Annotated Bibliography of Lambda-Calculi, Combinatory Logics and Type Theory

Inge Bethke

October 29, 1999

Contents

1 Fundamental theory

1.1 History and philosophy of mathematics

Potts, T. C. (1979). "The grossest confusion possible?"—Frege and the lambda calculus. *Rev. Internat. Philos.*, 33(4):761–785.

[MR82d:03004] The author asserts that A. Church's lambda calculus marks a wrong turn in the theory of meaning and that Frege's original Begriffsschrift system, as modified by the author (Studia Logica 37 (1978), no. 1, 7 - 26; MR 80a:03012), is the right path to follow. (Reviewer's remarks: Sometimes the author seems to distort Church's intentions, and his analysis is not incisive enough to determine whether his claims are correct.) Reviewed by E. Mendelson.

Curry, H. B. (1980). Some philosophical aspects of combinatory logic. In *The Kleene Symposium (Proc. Sympos., Univ. Wisconsin, Madison, Wis., 1978)*, pages 85–101. North-Holland, Amsterdam.

[MR82j:03012] This paper is developed from a talk the author gave at Florida State University in 1978. He first summarizes the main features of combinatory logic (Sections 1 - 7), and then answers in turn half a dozen miscellaneous points of criticism, some minor, some major, that have been made by H. Barendregt, D. Scott and others about philosophical aspects of his work (Sections 8 - 13). The topics are: (Section 8) the author's use of the assertion sign '$\vdash$', which has been criticized as confusing; (Section 9) the type-free structure of combinatory logic ("CL"); (Section 10) the accusation that CL has no intuitions behind it; (Section 11) the notion of function on which CL is based; (Section 12) the finite axiomatizability of CL; (Section 13) justification of CL as a part of logic. Because of time limitations, the author's points are made only in very quick outline. We shall comment on two of the topics below. Section 8: the assertion sign '$\vdash$'. In his works, the author uses this sign as a predicate, on a par with '$=$'. For example, from given obs X and Y, one can form the statements $\vdash X$, $\vdash Y$, $X = Y$. The author views the axioms and rules of a formal system S as defining the predicate $\vdash$ by recursion; $\vdash$ is the provability predicate for S. But he also uses $\vdash$ for deducibility: for a system S and obs $X_1, \dots, X_n, Y$, the statement $X_1, \dots, X_n \vdash Y$ means that if $\vdash X_1, \dots, \vdash X_n$ are adjoined as axioms to the definition of provability in S, then there is a deduction of $\vdash Y$ from these axioms [the author and R. Feys, Combinatory logic, Vol. I, Section 2B5c, see p. 50, North-Holland, Amsterdam, 1958; MR 20 #817]. In a review, Barendregt has criticized the author and his colleagues for confusing these two uses of '$\vdash$'; if X is an ob, then $\vdash X$ is a formula in the object language, but $X_1, \dots, X_n \vdash X$ is a statement in the metalanguage. The author replies making two points. Firstly, Barendregt's making the meta/object language distinction is a misstatement of the problem, because the author never makes it. Secondly, if it were made, then both uses of '$\vdash$' would be on the same level, namely, in the metalanguage (relative to a given S), and $\vdash X$ would be a special case of $X_1, \dots, X_n \vdash X$ with $n = 0$. In this reviewer's opinion, there is no confusion if one reads the author's works carefully. But one must be very careful. And if the meta/object language distinction had been used in the work of the author

and his colleagues, it would have helped a lot. For example, when the author writes of $\vdash X_i$ being adjoined to the axioms of S, the axioms are not obs of S but axioms of the recursive definition of provability over S. When these new axioms are adjoined, the relation (which we may call $\vdash_S$) of provability in S becomes a new provability relation $\vdash_{S'}$. And $X_1, \ldots, X_n \vdash X$ if and only if $\vdash_{S'} X$. A separation of languages would have helped to make this and similar points clearer. Sections 9 - 11: the intuitions behind combinatory logic. The "Curry school" has sometimes been criticized for constructing mere formal systems with no intuitive basis. While this has perhaps been true in some special cases, on the whole it has not. What is true, however, is that the intuitions the author uses are not those familiar to most modern mathematicians, who live in a world of ZF set theory. Firstly, the author's intuitions are based on the concept of function, not set; he points this out in Section 11. Secondly, his function concept is not the nowadays common one of a set of ordered pairs, but one which historically predates it, and in which computer science and constructive mathematics have produced a revived interest. A function is thought of as an operation which, when given an input, produces an output; it can be defined by describing the action of the operation on an arbitrary input, without necessarily saying whether this action will have a result for a particular input. An operation may be well defined without its domain being defined, or even definable. This view of functions has been explained very clearly by A. Church in the introduction to his Calculi of λ-conversion [Ann. Math. Studies, 6, Princeton Univ. Press, Princeton, N.J., 1941; MR 3, 129]. The operation concept has not yet been satisfactorily formalized. But in support of the author, the reviewer would stress that one cannot therefore dismiss it as nonexistent. The situation is like that in set theory before Zermelo; the powerful intuitions of Cantor had no known consistent formalization, but they were not abandoned on that account. In the reviewer's opinion the operator concept promises to give a deeper insight into the foundations than the now-standard set theories. And the members of the "Curry school" seem not to be alone in this view; there are many other formalizations of significant parts of mathematics in the literature which take the operator approach. (A recent example is a paper by P. Aczel and S. Feferman [To H.B. Curry: essays on combinatory logic, lambda calculus and formalism, pp. 67 - 99, Academic Press, London, 1980; MR 82b:03037].) (For the entire collection see MR 81j:03009.) Reviewed by R. Hindley.

Gandy, R. O. (1980). An early proof of normalization by A. M. Turing. In *To H. B. Curry: essays on combinatory logic, lambda calculus and formalism*, pages 453–455. Academic Press, London.
[MR81m:03018] The author reproduces an extract from an unpublished paper by Turing, probably written about 1940, showing that every formula of Church's simple type theory has a normal form. The author justifies his assertion that "Turing's proof antedates by many years any published proof of the theorem" with evidence from both published and unpublished sources. For the entire collection see MR 81j:03005.

Seldin, J. P. (1980). Curry's program. In *To H. B. Curry: essays on com-*

binatory logic, lambda calculus and formalism, pages 3–33. Academic Press, London.

[MR82j:03013] "Curry's program" is the author's name for the main themes running through the work of Haskell B. Curry. The author describes these themes, backed up by extensive quotations from Curry's published works, including some that are not generally well known. This article is well written, in the author's usual clear style, and is a very good account of Curry's approach to mathematical logic, with emphasis on historical development. It can be recommended to anyone who wishes to understand Curry's thought. (But be warned; like most nontrivial philosophy, it is not a work to be skimmed quickly.) Curry's work began in the 1920s when mathematical logic was still young, and many of today's concepts unknown. As the author points out, in the course of 50 years Curry's philosophical ideas have changed; but there are certain main strands running through his mathematical work. These seem to be the following. (1) (See especially Section 2 of the present paper.) An emphasis on formalization, and an analysis of the concept of "formal system". (This seems to be confused by some people with his philosophy of formalism, but is not quite the same. When Curry began his work, there were very few existing analyses of the concept of formal system in general, as opposed to developments of particular systems. And the virtues of formalization were not always appreciated.) Thanks to the work of Curry and his co-pioneers, this analysis is nowadays, with modifications, part of the "subconscious stock-in-trade" of every logician. But then it was new. (2) Included in the above, an analysis of substitution in systems with bound variables. This had previously been attempted, but with nontrivial errors. Curry's tools included the combinators, and these are what give Curry's formal systems their distinctive character. But in fact they are only a technical device, and any other equivalent linguistic device could have been used instead. The study of these combinators (pure combinatory logic) has taken up a large part of Curry's research effort, but they were never an end in themselves, only a stage on the way to deeper studies. (3) (See especially Section 3 of the present paper.) An analysis of the paradoxes, using the expressive power provided by the unrestricted type-free language. (In the reviewer's opinion, Curry's work here has led to deepened insight; though general agreement on the causes of the paradoxes is hardly to be expected.) (4) An emphasis on, though not a complete restriction to, "constructive" methods. Most of Curry's arguments (i.e. what most logicians would call his metatheory) can be formalized in first-order Peano arithmetic. And the subject of his investigations is principally (though not solely) constructive mathematics and logic. (5) (See especially Section 4 of the present paper.) The attempt to construct type-free formal systems adequate for significant parts of logic and mathematics. One of Curry's aims is to keep combinatory completeness (i.e. a minimally restricted language) as far as possible, and formulate all restrictions on (e.g.) the class of propositions explicitly by axioms of the system. (By the way, sometimes Curry has been accused of constructing mere formal systems without any intuitive motivation. This is a misunderstanding. Perhaps it comes partly from his insistence that our intuitions about a concept (e.g., that of function or set) are not absolutely pre-given, and may grow with time, and that attempts to formalize the concept (even failed inconsistent attempts) may play a positive role in developing such intuitions. Perhaps it comes also from the fact that Curry's intuitions on functions and sets are different from those familiar to most modern logicians [82j:03012 above]. They are older, and the reviewer believes they have possibilities of greater depth.) (For the entire collection see MR 81j:03005.) Reviewed by R. Hindley.

Rezus, A. (1982). *A bibliography of lambda-calculi, combinatory logics and related topics*. Mathematisch Centrum, Amsterdam. With a preface by Henk Barendregt.

[MR83c:03010] From the preface: "The items in this bibliography fall into the following categories: (1) the pure theory (syntax and semantics); (2) the theory related to foundations (of logic and mathematics); (3) applications (recursion theory, computer science, proof theory, category theory)."

Rosser, J. B. (1984). Highlights of the history of the lambda-calculus. *Ann. Hist. Comput.*, 6(4):337–349.

[MR86f:01030] This paper is a short history of the lambda calculus and combinatory logic by a participant in that history. After a brief introduction to both systems and their early history, the author takes up the inconsistency originally discovered by himself and S. C. Kleene [Ann. of Math. (2) 36 (1935), 630–636; Zbl 12, 146] and later simplified by H. B. Curry [J. Symbolic Logic 7 (1942), 115–117; MR 4, 125] and the effect of this inconsistency on the further development of the subject. He then gives the simplest known proof of the Church-Rosser theorem, which shows the consistency of both systems. Next follows a discussion of completeness, which is taken here to mean the representability of all partial recursive functions; instead of giving a proof, which is easily available in many other places, the author discusses the history of this representation and of Church's thesis, the original form of which was that all effectively computable functions are lambda representable. The author includes some personal recollections of his early work with Church and Kleene on this subject. This is followed by a discussion of models, including a simple presentation of a simple model due to G. D. Plotkin ["A set-theoretical definition of application", Memorandum MIP-R-95, School of Artificial Intelligence, Univ. Edinburgh, Edinburgh, 1972; per bibl.] and E. Engeler [Algebra Universalis 13 (1981), no. 3, 389–392; MR 83f:03012]. The paper closes with a discussion of the connection with computers, especially the early development of functional programming. There is an extensive list of references. Reviewer's remarks: 1. The paper is very well written and is a pleasure to read. 2. The reference to Engeler (1979) might be replaced with a reference to the published version cited above. Reviewed by J. P. Seldin.

Tichy, P. (1986). Constructions. *Philos. Sci.*, 53(4):514–534.

[MR88i:03015] The author claims that the subject matter of logic and mathematics is constructions rather than expressions or objects. Functions are regarded as incomplete constructions. This approach serves particularly to overcome difficulties that, according to the author, arise in the logical system of G. Frege 's Grundgesetze der Arithmetik (1893). A theory of constructions is developed based on rules for the formation (by abstraction) and the application of constructions. Within this theory, which is classical in nature (constructions being regarded as "ideal procedures"), partial functions can be handled. It is claimed that this theory escapes certain problems associated with A. Church 's and S. C. Kleene 's introduction of the λ-notation. Reviewer's remarks: A comparison of the author's theory of constructions with the intuitionistic theory of constructions as first outlined by G. Kreisel [in Foundations of intuitionistic logic, 198–210, Stanford Univ. Press, Stanford, Calif., 1962; MR 27 #3529] would be interesting. In his criticism of Frege the author fails to distinguish between metalinguistic talk about functions (in which Frege indicates argument places by lower case Greek letters), and open formulae within the theory (in which free variables are written as lower case Latin letters). Reviewed by Peter Schroeder-Heister.

Desclés, J.-P. and Ginisti, J.-P. (1988). Bibliographie commentée. *Math. Sci. Humaines*, (103):93–109.

[MR89k:03014] This bibliography lists 63 items under the heading "Combinatory logic and lambda-calculus" (including Frege's Grundgesetze der Arithmetik, which to the compilers' regret has never been translated into French), 37 items under the heading "Applicative languages, natural languages and cognitive representations", and 18 items under the heading "Applicative languages and programming". Most of the times are accompanied by descriptions and evaluations of their contents.

Ginisti, J.-P. (1988). Présentation de la logique combinatoire en vue des ses applications. *Math. Sci. Humaines*, (103):45–66, 113, 115.

[MR90d:03029] The development of combinatory logic as a branch of mathematical logic began with the lecture of the Soviet mathematician M. I. Schönfinkel "Über die Bausteine der mathematischen Logik", delivered in December of 1920 at Hilbert's seminar in Göttingen. Combinatory logic further evolved primarily as a theory of computations (algorithms) with a natural type-free operational semantics of an applicative kind as well as the calculi of the pure theory of combinators (Schönfinkel-Curry) and the λ-calculi of Church. It is supposed that the foundations of mathematics as well as of other disciplines can be constructed (according to Schönfinkel-Church-Curry) on the basis of the general applicative theory of abstract operators (considering sets as instances of such operators according to their functioning).

Combinatory logic, being a model of functional programming (according to the results of D. Scott, Y. L. Ershov, et al.), plays an important role in proof theory, algebra, topology, category theory, linguistics and other domains of modern knowledge. Such wide-scale applications of the Schönfinkel-Church-Curry apparatus are based on the unrestricted comprehension axiom scheme (in one of its forms) which is peculiar to it, as well as on the possibility of its deductive (for example, sequential in Gentzen's sense) extension (by restriction of the cut rule, but keeping the comprehension axiom scheme unrestricted). As a result, it has turned out that such sequential extensions possess properties of paraconsistent logics, elaborated by N. V. Vasilev, N. da Costa, A. Arruda, etc. [see, e.g., V. E. Volfengagen and V. Y. Yatsuk, Applikativnye vychislitelnye sistemy i kontseptualnyi metod proektirovaniya sistem znanii (Applicative computational systems and the conceptual method of designing knowledge systems), Minist. Oborony SSSR, Moscow, 1987; per revr.; N. A. Vasilev,

Voobrazhaemaya logika (Imaginary logic), "Nauka", Moscow, 1989; per revr.; the reviewer, Dokl. Akad. Nauk SSSR 270 (1983), no. 3, 537–541; MR 84m:03021; Math. Logik Grundlag. Math. 29 (1983), no. 5, 385–416; MR 85d:03026; Dokl. Akad. Nauk SSSR 273 (1983), no. 5, 1053–1057; MR 85f:03046; "Syntactically complete systems and a consistency theorem of formal arithmetics", in Kontseptualnoe proektirovanie bankov dannykh (Conceptual design of data bases), 94–102, Moskov. Dom Nauchn.-Tekhn. Propagand. F. E. Dzerzhinskogo, Moscow, 1986; per revr.; A. A. Kuzichev and the reviewer, Vestnik. Moskov. Gos. Univ. Ser. I Mat. Mekh. 1988, no. 6, 77–78].

In this paper the unrestricted comprehension axiom scheme is based on the minutely considered fundamental combinatorily complete collections of Schönfinkel-Curry combinators, in relation with their application to the humanities. Very interesting are the exposition of some aspects of the history of combinatory logic, the intuitive construction of applicative theories with combinators, yielding, in essence, unrestricted comprehension axioms, and the linguistic interpretation of such applicative theories with the separation of combinators which have arithmetical properties (see the above-mentioned references as well as the works of the Schönfinkel-Church-Curry school). The theory of the functionality operator F, which has properties close to those of the logical operator (not combinator) from the primary work of Schönfinkel , is expounded. In this paper, as in that of Schönfinkel , the theories do not appear in a syntactical form. As is generally known, the fundamental calculi of Church and Curry, which date from the 1930s, turned out to be inconsistent (Kleene-Rosser's paradox). However, in Schönfinkel 's work there is not any tendency towards the construction of Hilbert-type calculi; he rather notices ways to form deductive extensions, which have properties of paraconsistent logics—as already mentioned above—and are realized in the sequential constructions of the reviewer [see, e.g., Kombinatornyi Anal. No. 1 (1971), 105–119; MR 45 #8529]. Reviewed by A. S. Kuzichev.

Ginisti, J.-P. (1990). Erratum: "Presentation of combinatory logic with a view to its applications" [Math. Sci. Humaines No. 103 (1988), 45–66; MR 90d:03029]. *Math. Inform. Sci. Humaines*, (112):55.
[MR1096921] There will be no review of this item.

Schönfinkel, M. (1990). Sur les éléments de construction de la logique mathématique. *Math. Inform. Sci. Humaines*, (112):5–26, 59. Translated from the German by Geneviève Vandevelde; with an analysis and notes by Jean-Pierre Ginisti.
[MR92f:03013] This is a presentation of the French translation of Schönfinkel's paper "Über die Bausteine der mathematischen Logik" [Math. Ann. 92 (1924), 305–316; Jbuch 50, 23], prepared for publication in 1924 by H. Behmann on the basis of a lecture of Schönfinkel to the Mathematical Society of Göttingen in 1920. The main aim of Schönfinkel's lecture, and thus of the article, is the general elimination of variables (propositional, predicative, individual) by means of the utilization of several "particular functions" such as C, S, U, I, etc. Although the methods of Schönfinkel had a naive basis, his investigation founded what is now called "combinatory logic" in the sense of H. B. Curry. In the presentation of Schönfinkel's paper,

besides the translation of the paper there is an introduction to it, along with some critical conclusions at the end by Ginisti. Reviewed by M. S. Burgin.

Engeler, E. (1991). Zur wissenschaftstheoretischen Bedeutung der kombinatorischen Algebra. In *Jahrbuch 1990*, pages 45–53. Kurt-Gödel-Gesellschaft.

[MR92m:03016] The author of this paper puts forward two well-known theses: (1) Combinatory algebras must be considered, at least potentially, of equal importance to groups as ground structures in mathematics. (2) Combinatory logic provides a natural foundation for every mathematical theory and enriches these by its algorithmic aspects, in particular recursion and self-reference. He then illustrates "the role of combinatory logic as the natural culmination of the essential development of the mathematical abstraction process". This is done by considering some well-known and some unusual combinatory algebras. Reviewed by Martin W. Bunder.

Peña, L. (1991). De la logique combinatoire des Generales inquisitiones aux calculs combinatoires contemporains. *Theoria (San Sebastián) (2)*, 6(14-15):129–159. G. W. Leibniz (1646–1716) (Spanish).

[MR92m:03002] Summary: "In his 1686 essay Generales inquisitiones, Leibniz undertook to reduce sentences to noun-phrases, and truth to being. Such a reduction arose from his equating proof with conceptual analysis. Within limits, Leibniz's logical calculus provides a reasonable way of surmounting the dichotomy, thus allowing a reduction of hypothetical to categorical statements. However, it yields the disastrous result that, whenever A is possible and so is B, there can be an entity that is both A and B. Yet Liebniz in the Generales inquisitiones was the forerunner of 20th-century combinatory logic, which (successfully) practices—sometimes for reasons not entirely unlike Leibniz's own grounds—reductions of the same kinds he tried to carry out." For the entire collection see MR 92m:01004.

Seldin, J. P. (1991). In memoriam: Haskell Brooks Curry. In *Perspectives on the history of mathematical logic*, pages 169–175. Birkhäuser Boston, Boston, MA.

[MR94k:01033] This is a warm appreciation of Curry as a person and as one of the leading figures in mathematical logic in this century. In addition to sketching his academic career, broken only by four years as an applied mathematician in defense work during World War II, the author concentrates on Curry's development of the theory of combinatory logic. He emphasizes Curry's pragmatic approach, as opposed to those who have strong preconceptions about what is logically "correct". For the entire collection see MR 94b:03007. Reviewed by E. Mendelson.

Baaz, M., Brunner, N., and Svozil, K. (1993). Interpretations of combinatory algebras. In *Philosophy of mathematics (Kirchberg am Wechsel, 1992)*, pages 393–406. Hölder-Pichler-Tempsky, Vienna.

[MR95h:03027] The authors argue, by means of a somewhat loose, but interesting, collection of (informal) arguments and examples, in favour of the use of combinatory

logic as a tool in the philosophy of science. The paper is mainly concerned with the philosophical principles implicit in formal reasoning about experimental configurations, when concatenation of actions is taken as the basic notion. For the entire collection see MR 95a:00020. Reviewed by Harold Schellinx.

Engeler, E. (1993). *Foundations of mathematics.* Springer-Verlag, Berlin. Questions of analysis, geometry & algorithmics, Translated from the 1983 German original by Charles B. Thomas.

[MR94i:03001] The German original has been reviewed [1983; MR 85d:03024]. In the new preface, the author states that the purpose of the book is to combat "the lack of imagination that underlies the tendency to blind acceptance and thoughtless compliance with which many students approach the basic attitudes and concepts of mathematics" by presenting a cogent critique of these attitudes and concepts using the technical methods and results of mathematical logic.

Lalement, R. (1993). *Computation as logic.* Prentice Hall International, Englewood Cliffs, NJ. With a foreword by Michel Demazure, Translated from the 1990 French original by John Plaice.

[MR95f:68002] This book is a deep and rewarding exploration of the theoretical, practical, and historic connections among mathematical logic, computer science and programming. As an advanced text in computer science or logic, it offers a thorough treatment of standard fundamental topics and a unified introduction to results and techniques previously available only in scattered specialized works. As a bonus, the reader is offered an insightful and happily provocative philosophical perspective on logic as an experimental science, with computer science supplying the effective laboratory. Because of the scope and potentially wide appeal of this work, this review contains a more than cursory summary, while still omitting important topics. The title "Computation as logic" indicates two roles played by logic for computer science. From the outside, logic is a foundational discipline which offers formally and effectively equivalent models of the computable functions, representation of the computable functions and computational processes in formally axiomatized systems, uncomputability and undecidability results, and the classification of computational problems according to their difficulty. From the inside, the methods used in computer science are logical methods, both in the activities of programmers and in programs and how they are implemented.

After a first introductory chapter, Chapters Two through Seven are devoted to logical theory and methods inside computer science, while the final eighth chapter is an introduction to recursion theory and complexity theory. Two major contemporary computing paradigms, reduction and resolution, both originating from logic, are thoroughly studied. Reduction, associated with functional programming, includes β-reduction in the λ-calculus, and more general rewriting systems. Resolution, associated with logic programming, includes not only J. A. Robinson's famous proof procedure, but also the process, first studied by Herbrand in 1930, of solving equations during unification. The functional programming language used in the book is Standard ML (of New Jersey) while the logic programming languages are Prolog and Prolog-II.

Chapter Two, "The syntactic landscape", begins the formal treatment of computer science by recognizing its need to define and manipulate finite objects, such as integers, mathematical expressions, formulas, and program texts. The fundamental construction here is the finite term, which is approached from three equivalent and complementary vantage points: recursive definitions over the free monoid generated by a signature, tree derivations in inference systems, and tree domains (or occurrence languages). The author's examples show how the notion of finite term provides a unified treatment of first-order terms, data structures, applicative terms, and propositional formulas. Variables are introduced into signatures with an illuminating discussion of how different kinds of variables (logic, sequential, applicative, and functional) correspond to different programming paradigms. Discussion of the logic variable is taken up again in the fourth chapter on first-order logic and in the seventh chapter in connection with Prolog. The syntax of the λ-calculus is introduced as groundwork for the subsequent study of the λ-calculus as a paradigmatic programming language, and for classifying computational and logical objects as λ-terms (for example, functional programs and natural deduction proofs via the Curry-Howard correspondence). For programmers interested in implementing reduction systems, there is an introduction to shared implementations, including the de Bruijn λ-terms in AUTOMATH. Infinite terms inhabit the syntactic landscape of computer science as naturally as finite terms. Any decent programming language implements some form of the computable real numbers. So-called floating point numbers are truncated reals designating intervals. Some systems do exact rational arithmetic, and approximate irrationals. In either case, the problem is the finite representation of infinite terms. More significantly, infinite terms appear in computer science as data structures and in representing sequential processes. This motivates the study of formal terms encompassing finite terms and infinite terms, including the important rational terms. The notion of approximation of infinite terms leads directly to the notion of a partial term, an analogue of a partial function, which in turn introduces a crucial notion in program semantics: a term, as yet undefined, which may subsequently represent a later stage of a computation. The formal definition of this construction is viewed as a purely syntactic means of describing semantic behavior. The study of approximation is taken up again in Chapter Five on models, where Dana Scott's approach to program semantics is introduced. The final section of Chapter Two provides an elegant representation of sequential processes using rational terms, which isolates control and distinguishes algorithms from programs. The study of rational terms is taken up again in Chapter Six in connection with unification and equational logic, and in Chapter Seven in connection with the trees of Prolog-II.

Chapter Three on reduction studies β-reduction in the λ-calculus, the λ-calculus as a paradigmatic functional programming language, and more general rewriting systems based on the subsumption preorder over terms and call by pattern matching. Theoretical principles and methods of functional programming are illustrated by a rich variety of examples in ML. Compilation of λ-terms into applicative languages based on combinatory logic is then introduced in connection with the graph reduction machines used in implementing functional languages. The chapter concludes with a deeper study of termination and confluence.

Chapter Four on first-order logic studies the syntax of first-order languages, natural deduction proof systems, theories, and syntactic operations on theories. Basic syntactic matters are presented from the perspective of universal algebra, which delightfully pervades the entire book. Natural deduction systems are introduced both as formalizing reasoning, and as crucial components in the analysis of computation. The order of introduction of the proof systems reflects the current importance of intuitionistic logic in computer science. The order is: minimal propositional logic (without negation), intuitionistic first-order logic, and then classical first-order logic. The abstract syntax of derivations in minimal and intuitionistic logics is introduced via the Curry-Howard correspondence. Intuitionistic logic is viewed as the logic of sequential processes. Horn clauses (and thus clauses in pure Prolog) are analyzed as special cases of intuitionistic sequents. This offers an instructive alternative to the standard semantics of definite programs, which, historically, was presented using classical logic. This topic is taken up again in Chapter Seven on resolution. The final section emphasizes the analogues in computer science of syntactical operations on theories.

Chapter Five on models presents the classical and Kripke semantics for first-order logic, again, from the perspective of universal algebra with an eye on computer science. The treatment of Herbrand's theorem continues the theme of doing semantics with syntactic means, and the basic results are stated in terms of expansions, in line with Herbrand's original methods. A stronger result, "the real Herbrand theorem", is stated and identified as the key idea in the implementation of logic programming. The final section introduces Scott's framework for program semantics in the context of formal terms approximating infinite objects. The well-known models for the λ-calculus (due to Scott and Plotkin) are introduced.

When Chapter Six on equational logic is reached, algebras (or structures as they are also known in model theory) are the basic semantic objects in both logic and computer science. Algebraic structures (monoids, groups, rings, etc.) and data structures (lists, stacks, queues, etc.) are all formalized as algebras. The theory of algebras and equations is developed with attention to its historic roots in Birkhoff's work on varieties and stability. Unification is introduced in the general equational setting, with a welcome return to Herbrand's approach (rediscovered by Martelli and Montanari). Alternative approaches to unification algorithms are compared. The final section studies equational proofs and the Knuth-Bendix completion.

Chapter Seven on resolution is especially valuable for its presentation of topics in the semantics of logic programs not usually covered in the standard introductions to logic programming and Prolog. Among them are infinite terms in Prolog-II, interpretation of Horn clauses as intuitionistic sequents, Horn clauses and product algebras, narrowing, and a brief introduction to implementing equational rewriting systems in Prolog.

The final eighth chapter, "The computable landscape", begins by noting that the basic result of classical recursion theory (that the various formal models of computation are effectively equivalent) comes alive experimentally in computer science in comparing different paradigms of computation and analyzing different styles of programming. The computable functions are studied via the recursive functions, Turing machines and representation in first-order theories. A section on the classification of problems introduces the arithmetical hierarchy, reduction and creative sets. The final section introduces complexity theory in an algorithmic setting.

A careful reading of this book will reward logicians, computer scientists and philosophers of science in different ways. Logicians can witness the blossoming of logical concepts and methods in computer science. The practicing computer scientist can trace the logical and mathematical roots of programming. Philosophers of logic and computation will find fundamental issues relating to historic logical purposes and the everyday logical obligations of programmers, rather than the issues relating to the philosophy of mind inspired largely by artificial intelligence. And as the foreword reveals, the book can also be enjoyed by the nonspecialist mathematician with a fascination for logic and computer science.

Reviewer's remarks: The reviewer would like to offer a philosophical and historical caveat concerning the author's high-level discussion of the concept of function and its relation to the λ-calculus and computer science. The author states the widely held view that the classical set-theoretic concept of function is inadequate for computer science. By way of explanation, it is said that in computer science, it is necessary to represent infinite functions by their intensions, i.e., by computation rules. A return is thus made to the concepts of the nineteenth century. On the basis of this idea, the λ-calculus was introduced by A. Church in 1933, and developed by J. B. Rosser and S. C. Kleene (p. 48; see also p. 301). But was this really the idea? There are three foundational and philosophical questions here that should not be conflated: What are functions? What functions exist? How do we represent functions (in computer science or elsewhere)? Also, there are (at least) two important logical and philosophical players missing here, Curry and Scott. (The omission of Curry is curious given the author's wonderful penchant for an empirical view of logic and his apparent delight in doing semantic things in syntactic ways.) For Church and Curry, a fundamental question was whether functions should be defined as sets or sets defined as functions. Church's λ-calculi and Curry's (in crucial respects equivalent) combinatory logics were developed with the concept of function as primitive. But the issue is surely not settled. Scott's revolutionary work reverses the definitional and philosophical priority, while both perspectives continue to infuse logic and computer science with important ideas and methods. (For an introduction to these issues, the reviewer strongly recommends two papers by the missing players, both appropriately dedicated to Kleene on the occasion of his 70th birthday [H. B. Curry, in The Kleene Symposium (Madison, WI, 1978), 85–101, North-Holland, Amsterdam, 1980; MR 82j:03012; D. S. Scott, ibid., 223–265; MR 82d:03024]. The second (less important) part of the caveat concerns the author's description of the historical phases of the λ-calculus: "The λ-calculus has today become the paradigm of functional programming, after having been, first, an attempt to mechanize logic, then a general theory of functions, and finally, in the hands of computer scientists, a low-level programming language." (p. 48). The first two phases appear to be reversed. In fact, the λ-calculus was first proposed by Church, in 1933, in the spirit of the times, as a foundational symbolic logic embodying a general theory of functions. Then, in 1935, Church [Bull. Amer. Math. Soc. 41 (1935) no. 5, 332–333; Jbuch 61, 62] explicitly proposed to identify the class of effectively computable functions with the class of λ-definable functions (or, equivalently, with the class of recursive functions). Given the author's obvious knowledge of these things, so competently presented in the book, this is undoubtedly only a question of unfortunate phrasing. Caveat aside, the λ-calculus certainly has become the paradigm of functional programming, and this book is, among other good things, a systematic glimpse into how it is "used regularly to experiment with new computing paradigms, where certain properties, such as termination and confluence, play a crucial role comparable to that of the fundamental principles of physics". Having had the good fortune of participating in a few of these experiments, the reviewer heartily recommends the author's presentation. Reviewed by E. Ray Lucas.

Poncinie, L. (1993). Aristotle and Bressan on a number of things. *Erkenntnis*, 39(2):129–144.

[MR1250235] There will be no review of this item.

Kortright, E. V. (1994). Philosophy, mathematics, science and computation.

Topoi, 13(1):51–60.
[MR95m:00006] According to the author the metaphysical basis of computation is a prerequisite to any theory that describes the application of computer science to the world.

He first notes that the usual set-theoretic approach is not necessarily a satisfactory theory. It faces some well-known problems, such as incompleteness. Furthermore, quoting R. Thom, the author doubts that it can even describe the most elementary features of reasoning. And he notes that it is even worse than Aristotelian logic, which had at least a metaphysical foundation.

The author's proposal is to use the metaphysics of A. N. Whitehead. A central idea of Whitehead is a process called concrescence. By this process, which is a kind of synthesis involving all actual entities, new entities are created. The author then proposes a simulation language $\lambda\sigma$ based on Whitehead's metaphysics. This language is a typed λ-calculus which formally simulates the real-world phenomena as analysed by Whitehead. Reviewed by Jean-Yves Beziau.

Barendregt, H. (1996). Kreisel, lambda calculus, a windmill and a castle. In *Kreiseliana*, pages 3–14. A K Peters, Wellesley, MA.
[MR1435750] This item will not be reviewed.

Barendregt, H. (1997). The impact of the lambda calculus in logic and computer science. *Bull. Symbolic Logic*, 3(2):181–215.
[MR98k:03033] This paper is written to honor Church's great invention of the lambda calculus. The author is one of the most distinguished researchers contributing to the development of this theory, and he is also famous as the author of the book The lambda calculus. Its syntax and semantics [North-Holland, Amsterdam, 1981, 1984; MR 83b:03016; MR 86a:03012]. To start with, he presents the genesis of the lambda calculus. The notion of computability was first formalized in terms of definability on numerals represented in the lambda calculus. This is known as Church's Thesis. Nowadays we know various formulations of computability, such as Turing machines and recursive functions, which are all coincident with each another. The paper emphasizes that the formulation via the lambda calculus has the advantage of directly representing computations on various data types in addition to numerals. It is possible to represent data types in a very direct manner in various lambda calculi.

Next, the paper introduces two major areas of applications: the presentation of computations and the resulting functional programming languages on one hand, and the presentation of reasoning and the resulting systems of computer mathematics on the other hand. The lambda calculus is a theory formalizing the most primitive behavior of functions, and it is regarded as a prototype for programming languages. A number of actual programming languages have been designed and implemented on the basis of the lambda calculus. The typed version of the lambda calculus also provides a formal system for reasoning about mathematical propositions. We know a way to represent formal proofs as typed lambda terms. This is called the propositions-as-types interpretation, and was discovered by Curry, Howard, and de Bruijn. Several systems of mathematical reasoning based on this interpretation have been created.

The present paper is not only a good introduction to the lambda calculus for readers unfamiliar with this theory, but also a hint for readers familiar with the lambda calculus to discover new merits in it. Moreover, this paper presents a history of the lambda calculus, with interesting stories such as the reason why the letter λ was chosen to denote function abstraction. Reviewed by Hirofumi Yokouchi.

Ginisti, J.-P. (1997). *La logique combinatoire.* Presses Universitaires de France, Paris.

[MR98f:03012] This monograph on combinatory logic can be recommended to anyone interested in the foundations of mathematics and in logic, but it is in fact intended for a rather wide range of readers, including potential users in the social sciences. I would also recommend it to any student of λ-calculus, even though λ-calculus per se occupies no more than one page here.

The book provides a careful mathematical and philosophical presentation of combinators and of combinatory logic, which mainly relies on the pioneering work and views of Schönfinkel (1924) and Curry (from 1926 on). Besides an accurate mathematical development, the book contains many interesting mathematical and philosophical motivations and discussions and historical comments, and it also treats a few applications, essentially to linguistics. Finally the book ends with a presentation of Quine's related system PFL (predicate functor logic).

Combinatory logic (CL) was created as a mathematical tool for analyzing mathematical formalization; it provides an algebraic formalization of mathematics and logic from a unique binary operation (the "application" $x, y \to xy$) and a few basic constants which have a prescribed combinatorial action (see examples below). CL allows one to eliminate bound variables, either when expressing functions (the bound variable x is "abstracted" in $x \to x + 3$), or in formulas (the occurrences of x bound by some first-order quantifier). As a matter of fact, the latter kind of elimination can only be done in Schönfinkel 's "new calculus", which is CL augmented with a further constant U, which is to predicate calculus what Scheffer's connector is to propositional calculus: sufficient. Finally, CL, like λ-calculus, allows one to eliminate the usual mathematical constants (such as $+$ and $\times$ in arithmetic), and more generally allows one to represent all recursive functions.

The view of CL which is given in the present book is quite orthogonal to the modern mathematical view, which gives CL a secondary role and only considers it as the (useful) first-order presentation of Church's λ-calculus, which is itself of first interest in theoretical computer science. Unlike CL, λ-calculus, which is based on application and an "abstraction operator", gives an important role to bound occurrences of variables and hence allows one to retain many functional intuitions; λ-calculus is the basis of "functional programming" and, because of the facilities it offers both for computing and for encoding proofs in natural deduction style, is also the basis of the "programming with proofs paradigm".

CL can be based on various sets of constants, also called "basic combinators". The usual choice nowadays is to take (K, S), with $Kxy \to x$ and $Sxyz \to (xy)(yz)$; everywhere $Xxyz$ has to be read $(((Xx)y)z)$. This is a choice which allows a concise (although non-intuitive) presentation of CL and is particularly well-suited for proving the equivalence between CL and λ-calculus.

The author's choice, which essentially follows Schönfinkel , is to start from the five combinators (B, C, K, W, I), which respectively allow composition, commutation, deletion, duplication and identification; their actions are defined as follows: K as above, $Ix \to x$, $Cxyz \to xzy$, $Bxyz \to x(y(z))$, and $Wxy \to xyy$. This choice allows a deeper understanding of what is going on at the combinatorial level, is more efficient, and is better suited to applications.

The author's bias in his historical view is, however, sometimes heavy, in particular as regards Curry's typed version of CL in Chapter V, which could have been given a lighter treatment.

The monograph is written in a slightly literary style, and it has to be read linearly, which is not a real drawback since it is rather short and really interesting.

To put CL in a more recent mathematical context the reader could complement this monograph with earlier work of J.-L. Krivine [Lambda-calculus, types and models, translated from the 1990 French original by Rene Cori, Horwood, New York, 1993; MR 94h:03034 (Chapter 6)], H. P. Barendregt [The lambda calculus, Revised edition, North-Holland, Amsterdam, 1984; MR 86a:03012 (Chapter 7, Appendix B)], and, finally, if (s)he is interested in an implementation of CL as a (low-level) programming language (s)he could have a look at an earlier book of A. J. Field and P. G. Harrison [Functional programming, Addison-Wesley, Wokingham, 1988 (Chapter 12); Zbl 828.68033]. Reviewed by Chantal Berline.

Laan, T. (1997). *The evolution of type theory in logic and mathematics.* PhD thesis, Technische Universiteit Eindhoven.

[MR98k:03017] In this dissertation, the author undertakes the exposition of type theory along a historical line, but with a theoretical attitude. His careful reconstruction of the ideas of Frege and Russell, which gave rise to modern type theory, makes use of typed λ-calculus and pure type systems (PTSs) to obtain a rigorous formalization of the original exposition, which is difficult for the modern reader to appreciate, as it does not conform to modern mathematical standards.

The adoption of modern formal methods allows for a precise analysis of Russell's ramified theory of types (RTT) and of the deramification process, leading to simple types and λ-calculus. This culminates with the comparison of Russell's theory with Kripke's theory of truth, in the context of the discussion of the limits of the former theory as a theory of predicativity.

The modern part of the exposition is concerned with the theme of the propositions-as-types and proofs-as-terms paradigm. It encompasses a description of the Automath proof-checker and of related systems, based on the theory of PTSs. The thesis contributes to this last topic by proposing and studying a parametric extension of PTSs.

The spirit of this work is formalist and syntax-directed: it rarely addresses semantical questions, which are touched upon only in Chapter 3, which, to the reviewer's taste, is the most interesting one. The lack of consideration of, e.g., Martin-Löf type theory and of foundational topics not limited to Russell's RTT is, however, disappointing.

Those readers with interests in automated theorem proving and PTSs will find in this thesis material and hints for further research. It is the reviewer's opinion, however, that this thesis does not provide an introductory account of the large family of typed λ-calculi, and that it will be better appreciated by readers having some background knowledge of the field. Reviewed by Ugo de'Liguoro.

1.2 Untyped lambda calculus and combinatory logic

Barendregt, H. (1977). Solvability in lambda calculi. In *Colloque International de Logique (Clermont-Ferrand, 1975)*, pages 209–219. CNRS, Paris.

[MR80f:03020] This paper is a discussion of the notion of solvability for various versions of the type-free λ-calculus. In the case of the λI-calculus the solvable terms are just those having a normal form. In the λK-calculus the solvable terms are those which do not necessarily have a normal form, but which reduce to normal form if applied to a certain number of suitable arguments. They can be characterized as the terms in head normal form, a notion invented by C. Wadsworth. A head normal form is a term of the form $\lambda x_1 \cdots \lambda x_n.yN_1...N_k$ $(n, k \geq 0)$. The author gives a number of convincing arguments that the unsolvable terms are just the meaningless ones. A theorem of M. Hyland and Wadsworth says that in D. Scott's models D_∞ and P_ω for the λ-calculus all unsolvable terms are equated and mapped to the least element of D_∞ or P_ω, respectively. One can now view this fact either as an argument for considering the unsolvable terms as meaningless or as an argument for considering Scott's models to be the natural ones. It is not clear to the reviewer which of these views represents the opinion of the author. It is then shown that the (equational) theory obtained from the λ-calculus by equating all unsolvable terms has a unique Hilbert-Post-complete extension and this is just $Th(D_\infty)$, independently of the original lattice D. In the final paragraph a representation of the partial recursive functions in the λ-calculus is discussed where "undefined" corresponds to "unsolvable" and not just to "not having a normal form". Many of the results in this paper were already known, but putting them together gives very interesting new insights into the λ-calculus. Some misprints should be noted. On page 211 the first sentence after Definition 1.3 should be "To see the particular role of $I \cdots$" instead of "To see the particular role if $I \cdots$". In Theorem 1.18 replace "$D_\infty \vdash M = I$" by "$D_\infty \vdash M = \perp$". In Theorem 2.9. replace "D_ω" by "D_∞". In Corollary 2.10 replace "D^0_ω" by "D^0_∞". (For the entire collection see MR 58 #27191.) Reviewed by Gerd Mitschke.

Canal, R. (1978). Complexité de la réduction en logique combinatoire. *RAIRO Inform. Théor.*, 12(4):339–367.

[MR80a:03016] Combinatory logic has received considerable attention as a model for describing the semantics of programs, data and computations. Insufficient attention, however, was devoted to the computational properties of this algorithmic system until recently. Among the very few papers covering this aspect, this paper is particularly rich in detailed results. In it, the author presents his work as a continuation of some of the early results in combinatory logic (see H. B. Curry and R. Feys, Combinatory logic, Vol. I, North-Holland, Amsterdam, 1968; MR 39 #5368; Curry, J. R. Hindley and J. P. Seldin, Combinatory logic, Vol. II, 1972; Zbl 242 #2029) and some recent papers on this subject, aimed at analyzing complexity properties of terms and of normal form reductions, respectively interpreted as programs and computations. The computational power of basic combinators (such as those of duplicating, cancelling, composing and exchanging terms) brings, at a low complexity level, peculiar complexity properties in this formal system which have no counterpart in other formal systems, such as RAMs or Turing machines, which have received much more attention in connection with computational complexity. Simple results in this direction, expressing upper bounds to the "running time" of terms as a function of the size of the terms, in the case of linear combinations without a compositional effect and in the case of linear combinations having only a cancelling effect are considered in the first part of the paper. In the second part, deeper results state some sufficient and easily detectable conditions under which a combinator does not possess a normal form, particularly in the case of combinators with only a duplicating effect. Reviewed by Giorgio Ausiello.

Coppo, M., Dezani-Ciancaglini, M., and Ronchi Della Rocca, S. (1978). (Semi-)separability of finite sets of terms in Scott's D_∞-models of the λ-calculus. In *Automata, languages and programming (Fifth Internat. Colloq., Udine, 1978)*, pages 142–164. Springer, Berlin.

[MR80k:03018] A set of n lambda-terms is said to be separable if there exists a context $C(\ )$ such that $C(M_i) \to x_i$ for $i = 1, \ldots, n$, where the x_i are n distinct variables; the set is said to be semiseparable if there exist a context $C(\ )$ and an integer j, $1 \leq j \leq n$, such that $C(M_i) \to x_i$ for $i \neq j$ and $C(M_j) \to \Omega$. This paper presents a necessary and sufficient condition for a set to be separable when its terms are all different in Scott's model D_∞. This result improves the previously known results on (semi-)separability, namely Böhm's theorem (two different $\beta\eta$-normal forms are separable) and the result that any set of different $\beta\eta$-normal forms is separable (Barendregt and Böhm, Dezani-Ciancaglini, Peretti, Ronchi Della Rocca). The condition is called the (semi-)distinctness condition and is expressed in terms of Böhm trees modulo η. It is shown that any set is (semi-)separable if and only if it is (semi-) distinct. All the previously mentioned results immediately follow, and it also follows that two terms M and N are separable if and only if there exists no term P such that $M = P$ and $N = P$ hold in D_∞. (For the entire collection see MR 80c:68003.) Reviewed by G. Berry.

Hindley, R. (1978a). Reductions of residuals are finite. *Trans. Amer. Math. Soc.*, 240:345–361.

[MR80a:03017] A β-development of a λ-term M is a reduction sequence starting with M and contracting only redexes in M or their residuals. The finite development theorem (FD) for β(-reduction) states that for any M all β-developments are finite. FD for β was first proved for the λI-calculus by A. Church (The calculi of λ-conversion, Ann. of Math. Studies, No. 6, Princeton Univ. Press, Princeton, N.J., 1941; MR 3, 129). For the λK-calculus three independent proofs have been given by Schroer and Hyland (both unpublished) and in the present article. The author's proof depends on the fact that some β-development terminates (this follows easily by contracting innermost redexes). The author proves FD also for combinatory weak reduction and for $\beta\eta a$-reductions. The a-notion of reduction applies to several systems of contraction rules, e.g., for the recursor and Church's δ. Reviewed by Henk Barendregt.

Hindley, R. (1978b). Standard and normal reductions. *Trans. Amer. Math. Soc.*, 241:253–271.

[MR80a:03018] Let R, S be subterm occurrences of a λ-term M. R is senior to S if and only if S is a proper subpart of R or R is completely to the left of S. Consider a reduction path $\sigma : M_1 \to M_2 \to \cdots \to M_n$, where in each step $M_i \to M_{i+1}$ a redex R_i is contracted. Then σ is said to be standard if and only for $1 < i \leq n$ it is not the case that R_i is a residual of a redex senior to R_{i-1} (all redexes senior to R_{i-1} are "temporarily frozen"); σ is said to be strongly standard if and only if the preceding definition holds with "R_{i-1}" replaced by "some R_j with $j < i$" (the redexes senior to R_j are "permanently frozen"). The (strong) standardization theorem states that if M reduces to N, then there is a (strong) standard reduction from M to N. It is shown that for β-reductions the notions of standard and strongly standard are equivalent, but not so for $\beta\eta$-reductions or for terms involving the recursor. Then for β-reduction the standardization theorem is proved, completing the proof of H. B. Curry and R. Feys (Combinatory logic, Vol. I, North-Holland, Amsterdam, 1968; MR 39 #5368) by using the main result of the author's previous paper (03017 above). For $\beta\eta$ and $\beta\eta a$-reduction (including, e.g., the recursor or δ-redexes) the standardization theorem is proved, but for βa-reduction a counterexample is given to the strong version. The normal reduction theorem states that if a term has a normal form, then it can be reached by a reduction contracting always the leftmost redex. This is proved for β-reduction as an application of the strong standardization theorem. The paper leaves open the question whether for $\beta\eta$-reduction the strong standardization or normal reduction theorem holds. Both statements have been proved by J. W. Klop ("Reductions in λ-calculus and extensions", Dissertation, Utrecht, 1979). (Also the nearly normal reduction theorem for $\beta\eta$ is proved there; see the present paper for the definition.) G. Huet and J. J. Levy ("Computations in nonambiguous linear term rewriting systems", unpublished, 1978) have examined the question of strong standardization for (a natural class of) term rewriting systems (these are combinatory versions of βa-reduction systems). Reviewed by Henk Barendregt.

Venturini Zilli, M. (1978). Head recurrent terms in combinatory logic: a generalization of the notion of head normal form. In *Automata, languages and programming (Fifth Internat. Colloq., Udine, 1978)*, pages 477–493. Springer, Berlin.

[MR80g:03014] In Church's conception, only λ-terms with normal forms are defined (have meaning). More recently it has become common to regard as defined λ-terms which have a head normal form (equivalently, are solvable) even if they have no normal form, so that only terms without a head normal form are considered totally undefined. But even terms without a head normal form can carry some computational information: a term like $(\lambda x.xx)(\lambda x.xx)$, which reduces only to itself, seems in some sense "less" undefined than a term like $(\lambda x.xxx)(\lambda x.xxx)$, which reduces to terms of indefinitely increasing length. The author, dealing with combinators with weak reduction rather than with λ-calculus, defines a term T to be recurrent if every reductum of T reduces to T, and then defines a term to be head recurrent if it is in head normal form or else is of the form $RU_1 \cdots U_m$ where R is a recurrent redex. (Clearly a term in normal form is recurrent.) She then gives a reduction strategy which reveals both recurrence and head recurrence, and proves that there is no optimal strategy which does this. (The reviewer has made some suggestions and criticisms, which the author has accepted; the author is now working on a new paper which simplifies the theory for weak combinatory logic and extends the results to λ-calculus. In Examples 2 and 3 on p. 478, the terms in question are not recurrent (because B, W and C_+ are defined in terms of I, K and S and are not atomic), but they reduce to recurrent terms. In Theorem 5 (p. 483), $T^{(n)} \equiv R$ is a redex, and so constitutes an exception to the theorem as stated. Finally, the reviewer found the description of the reduction strategy h^+ on pp. 486 - 487 somewhat confusing and feels that the examples on pp. 487 - 488 would have been easier to follow if the clause of the algorithm applied at each step had been indicated.) (For the entire collection see MR 80e:68003.) Reviewed by J. P. Seldin.

Wadsworth, C. P. (1978). Approximate reduction and lambda calculus models. *SIAM J. Comput.*, 7(3):337–356.

[**MR81f:03020**] Author's summary: "We give the technical details and proofs for the notion of approximate reduction introduced in our earlier paper (same journal 5 (1976), no. 3, 488 - 521; MR 58 #21493). The main theorem asserts that every lambda expression determines a set of approximate normal forms of which it is the limit in the lambda calculus models discovered by D. Scott in 1969. The proof of this theorem rests on the introduction of a notion of type assignments for the lambda calculus corresponding to the projections present in Scott's models; the proof is then achieved by a series of lemmas providing connections between the type-free lambda calculus and calculations with these type assignments. As motivation for these semantic properties, we derive also some relations between the computational behavior of lambda expressions and their approximate normal forms, and we establish a syntactic analogue of the general considerations motivating the continuity of functions in Scott's lattice-theoretic approach."

Baeten, J. and Boerboom, B. (1979). ω can be anything it should not be. *Nederl. Akad. Wetensch. Indag. Math.*, 41(2):111–120.

[**MR80h:03018**] This paper contributes to the study of the phenomenon, noticed by G. Jacopini (λ-calculus and computer science theory (Proc. Sympos., Rome, 1975), pp. 213 - 219, Springer, Berlin, 1975; MR 57 #2887), that the term $(\lambda x.xx)(\lambda x.xx)$ of λ-calculi can be equated with any other closed term M without inconsistency. Jacopini's result was for the $\lambda\beta$-calculus. The present paper extends that result, by similar proof-theoretic techniques, to the $\lambda\beta\eta$-calculus. It also gives a model-theoretic proof of Jacopini's weaker result, by devising for each such M a modification of Scott's P_ω model in which $(\lambda x.xx)(\lambda x.xx) = M$. Reviewed by John Staples.

Bergstra, J. and Klop, J. W. (1979). Church-Rosser strategies in the lambda calculus. *Theoret. Comput. Sci.*, 9(1):27–38.

[**MR80i:03024**] A "reduction strategy" for $\lambda\beta$ is an $F : \lambda-\text{terms} \to \lambda-\text{terms}$ such that $F(X)$ is a term to which X $\lambda\beta$-reduces. Example 1: "Contract the leftmost redex in X or, if no redex exists, set $F(X) = X$." Example 2: "Do a complete development of all the redexes in X." A "Church-Rosser strategy" is an F such that if $X =_\beta Y$, then $F^m(X) \equiv F^n(Y)$ for some m, n. A "one-step strategy" is an F such that $F(X)$ is obtained by only one contraction in X (if X contains a redex). The authors construct (1) a recursive Church-Rosser strategy, (2) a nonrecursive one-step Church-Rosser strategy, and (3) a recursive one-step strategy which is Church-Rosser on the set Σ of applicative combinations of S ($S \equiv \lambda xyz.xz(yz)$.) The main problem in (1) and (2) is to avoid looping; to be Church-Rosser, the strategy must identify useless reduction-cycles and break out of them. The one-step strategy only does this at the cost of recursiveness. (For (3), the authors show that no loops are possible in Σ.) Authors' question: Does there exist a recursive Church-Rosser strategy for the set of all terms? Reviewed by R. Hindley.

Böhm, C., Dezani-Ciancaglini, M., Peretti, P., and Ronchi Della Rocca, S. (1979). A discrimination algorithm inside $\lambda\beta$-calculus. *Theoret. Comput. Sci.*, 8(3):271–291.

[MR80g:03012a] The following theorem is proved. Given a finite set of closed $\beta\eta$-normal forms $M_1, \ldots, M_n$, then one has $\forall N_1 \cdots N_n \exists F(FM_1 =_\beta N_1 \& \cdots \& FM_n =_\beta N_n)$. In fact the F that is constructed is of the form $\lambda x.xQ_1 \cdots Q_p$ for some fixed $\vec{Q}$ (depending on $\vec{M}, \vec{N}$). For $n = 2$ the statement was proved by Böhm ("Some properties of the $\beta\eta$-normal forms of the λK-calculus" (Italian), Publication No. 696, Ist. Naz. App. Calcolo, Rome, 1968). The method of that paper is used and extended to prove the present result. Reviewed by Henk Barendregt.

Dezani-Ciancaglini, M., Ronchi Della Rocca, S., and Saitta, L. (1979). Complexity of λ-term reductions. *RAIRO Inform. Théor.*, 13(3):257–287.

[MR81j:68016] Authors' summary: "The complexity of a λ-term is defined as the number of β-reductions needed to reach its normal form. In the present paper some general properties of the complexity of applications of normal forms are stated and this complexity is maximized, in some interesting cases, as a function of parameters describing the structures of the current normal forms."

Hindley, R. (1979). The discrimination theorem holds for combinatory weak reduction. *Theoret. Comput. Sci.*, 8(3):393–394.

[MR80g:03012b] The following theorem is proved in Böhm, Dezani-Ciancaglini, Peretti and Ronchi Della Rocca ("A discrimination algorithm inside $\lambda\beta$-calculus", Theoret. Comput. Sci. 8 (1979), no.3, 271–291). Given a finite set of closed $\beta\eta$-normal forms $M_1, \ldots, M_n$, then one has $\forall N_1, \ldots, N_n \exists F\ (\ FM_1 =_\beta N_1 \& \cdots \& FM_n =_\beta N_n\ )$. In fact the F that is constructed is of the form $\lambda x.xQ_1 \cdots Q_p$ for some fixed $\vec{Q}$ (depending on $\vec{M}, \vec{N}$). In this paper it is shown that the result also holds if the $M_1, \ldots, M_n$ are combinatory terms in strong normal form and $=_\beta$ is replaced by $=_w$ (convertibility corresponding to weak reduction). Reviewed by Henk Barendregt.

Mitschke, G. (1979). The standardization theorem for λ calculus. *Z. Math. Logik Grundlag. Math.*, 25(1):29–31.

[MR80g:03013] Using the theorem that developments of λ-terms are finite (R. Hindley, Trans. Amer. Math. Soc. 240 (1978), 345 - 361; MR 80a:03017), the author gives a fast proof of the useful standardization theorem (H.B. Curry and R. Feys, Combinatory logic. Vol. 1, second printing, North-Holland, Amsterdam, 1968; MR 39 #5368). Reviewed by Henk Barendregt.

Staples, J. (1979). A lambda calculus with naive substitution. *J. Austral. Math. Soc. Ser. A*, 28(3):269–282.

[MR81b:03015] The problem is to eliminate the need for changing bound variables

(to avoid collisions resulting from substitutions) and at the same time to preserve the processes of β-reduction and η-reduction (and as much as possible of the convenience of the λ-notation). The author's solution is to formulate λ-calculus so that each occurrence of the λ-operator binds a selection of the positions of free variables in a term, rather than a particular variable. The paper is clearly written with ample details. Reviewed by B. Lercher.

Turner, D. A. (1979). Another algorithm for bracket abstraction. *J. Symbolic Logic*, 44(2):267–270.

[MR80f:03021] In this short note the author provides a new bracket abstraction algorithm for combinatory logic. Bracket abstraction $(x)A$ is a combinatory version of λ-abstraction $\lambda x.A$, i.e., $(x)A$ is a combinatory term which has the property that $(x)AB$ reduces (in combinatory logic) to $(B/x)A$ (i.e., B substituted for x in A). The known bracket abstraction operators produce very long terms, i.e., $(x)A$ is usually much longer than A. The author's new algorithm is not much more complicated than older ones, and is much more efficient. Reviewed by Gerd Mitschke.

Barendregt, H. and Koymans, K. (1980). Comparing some classes of lambda calculus models. In *To H. B. Curry: essays on combinatory logic, lambda calculus and formalism*, pages 287–301. Academic Press, London.

[MR82b:03038] The authors discuss four varieties of models for combinatory logic and λ-calculus, as in the first author's article in the Handbook of mathematical logic (Part D, Proof theory and constructive mathematics, D7, North-Holland, Amsterdam, 1977; MR 58 #10343). The authors show: (1) The term model of (weak) combinatory logic is not a λ-algebra (i.e. model of β-equality). (2) The model of closed terms of λ-calculus in which the unsolvable terms are identified is a λ-algebra which is not a λ-model (i.e. rule (ξ) is not satisfied in the model). (3) However, every λ-algebra can be embedded in a λ-model. (4) Let T be the λ-calculus obtained by adding the axioms $\Omega(\lambda x.x) = \lambda x.x$, and $\Omega(\lambda xy.xy) = \lambda x.x(\lambda y.y)(\lambda zw.z)$, where $\Omega = (\lambda x.xx)(\lambda x.xx)$. Then T is consistent, and the term model of T is a λ-model that cannot be embedded in an extensional λ-model. (For the entire collection see MR 81j:03005.) Reviewed by B. Lercher.

Barendregt, H. and Longo, G. (1980). Equality of lambda terms in the model T^ω. In *To H. B. Curry: essays on combinatory logic, lambda calculus and formalism*, pages 303–337. Academic Press, London.

[MR82g:03017] The structure $T^\omega = [\omega \to T]$ is studied, where T is the three-element poset $(\perp, 0, 1)$ with $\perp \sqsubseteq 0$ and $\perp \sqsubseteq 1$, and where $f \sqsubseteq g$ if and only if $f(x) \sqsubseteq g(x)$ for all $x \in \omega$. It was introduced by G. Plotkin [J. Comput. System Sci. 17 (1978), no. 2, 209 – 238; MR 80d:68105] and shown to be a model of λ-calculus, joining the earlier models P_ω and D_∞. The present paper gives an important analysis of T^ω using ideas and methods of the first author [Handbook of mathematical logic, Part D, pp. 1091 - 1132, North-Holland, Amsterdam, 1977; see MR 58 #10343], especially those revolving about the notion of the Böhm tree $BT(M)$ of a λ-calculus term M.

The authors prove an approximation theorem for T^ω, stated as in Theorem 7.7 of the first author's article (op. cit.). The main result shows that $M = N$ holds in T^ω if and only if $BT(M) = BT(N)$ (as for P_ω), and that $M \sqsubseteq N$ holds in T^ω if and only if $BT(M) \sqsubseteq BT(N)$. Thus T^ω is in a sense more faithful than $P\omega$ as a model of λ-calculus. (For the entire collection see MR 81j:03005.) Reviewed by B. Lercher.

Bergstra, J. and Klop, J. W. (1980). Invertible terms in the lambda calculus. *Theoret. Comput. Sci.*, 11(1):19–37.

[MR81k:03016] The set of λ-terms whose elements are obtained by identifying two terms M and N whenever they are $\beta\eta$-convertible ($M = N$) becomes a semigroup if we define $M \circ N = BMN$, where $B \equiv \lambda xyz.x(yz)$. The identity element is $\lambda x.x$, named I, so that M is $\beta\eta$-invertible if and only if there exists N such that $M \circ N = N \circ M = I$. Invertible λ-terms are characterized in the paper. The main theorem states that M is $\beta\eta$-invertible if and only if M is a particular normal form named finite hereditary permutator. One direction of the proof of the main theorem is a consequence of a result by M. Dezani-Ciancaglini about invertible normal forms, which states that a normal form is $\beta\eta$-invertible if and only if it is a finite hereditary permutator. The other direction of the proof works through the following implications: M is $\beta\eta$-invertible $\Rightarrow$ M is D_∞ (Scott's model)-invertible $\Rightarrow$ M is a hereditary permutator, and also $M \circ N = I \Rightarrow M \circ N$ has $\beta\eta$-normal form $\Rightarrow M \circ N$ has β-normal form $\Rightarrow$ the depth of $BT(M)$ is finite, where $BT(M)$ is the Böhm tree of a term M. The paper contains a second proof of the implication: M is $\beta\eta$-invertible with inverse $N \Rightarrow M$ and N have a normal form. This second proof works inside the λI-calculus. Finally it is proved that the set of hereditary permutators is exactly the set of D_∞-invertible λ-terms, i.e. that M is a hereditary permutator $\Leftrightarrow$ M is invertible in D_∞. For the proof of this characterization of hereditary permutators, a formal inverse M^* of M, with respect to D_∞, is defined and is proved to exist as a hereditary permutator. The paper is self-contained modulo the basic notions about λ-calculus and its known models. Reviewed by Marisa Venturini Zilli.

Böhm, C. (1980). An abstract approach to (hereditary) finite sequences of combinators. In *To H. B. Curry: essays on combinatory logic, lambda calculus and formalism*, pages 231–242. Academic Press, London.

[MR82b:03039] This paper presents a new method of representing linear or multilevel (hereditary) finite sequences of combinatory terms. The method is an improvement over preceding approaches, such as that of H. B. Curry, J. R. Hindley and the reviewer (Combinatory logic, Vol. II, North-Holland, Amsterdam, 1972; Zbl 242:02029) in that two special terms are no longer needed: a special atom (here meaning an element of a sequence, not necessarily an atomic term) different from any other to denote the end of a sequence, and a discriminator between atoms and other terms. Since these special terms are no longer needed, any term can be an atom (i.e. can occur in a sequence). There are several standard operations on sequences which need to be represented. One is concatenation, forming the sequence $\langle x_1, \dots, x_n, y_1, \dots, y_m\rangle$ from $\langle x_1, \dots, x_n\rangle$ and $\langle y_1, \dots, y_m\rangle$. Another is $(unit)$, which, when applied to x, gives $\langle x\rangle$, the sequence whose only element is x. We also have the empty sequence $\langle\rangle$ or (nil). Also needed are $(head)$, which picks out the first element of a sequence, $(tail)$, which deletes the first element of a sequence, and $(null)$, which discriminates between (nil) and the other sequences. Since there are a number of ways to represent these sequences that satisfy the author's criteria, he begins by assuming the existence of a six-tuple of combinatory terms $((cat),(nil),(unit),(head),(tail),(null))$ satisfying the following six postulates: (1) $(cat)x((cat)yz)$=$(cat)((cat)xy)z$, (2) $(cat)x(nil) = (cat)(nil)x = x$, (3) $(head)((cat)((unit)x)y) = x$, (4) $(tail)((cat)((unit)x)y) = y$, (5) $(null)(nil) = 1$, and (6) $(null)((cat)((unit)x)y) = 0$, where 0 and 1 are the Church numerals $\lambda xy.y$ and $\lambda xy.xy$, respectively. The set $\mathcal{L}$ of finite linear sequences is defined by induction as follows: (i) $(nil) \in \mathcal{L}$, (ii) if t is any term then $(unit)t \in \mathcal{L}$, and (iii) if t is any term and $Y \in \mathcal{L}$ then $(cat)((unit)t)Y \in \mathcal{L}$. From these postulates it can be shown that for each $X \in \mathcal{L}$ there is a natural number n such that $(tail)^n X = (nil)$ (finiteness) and that for each $X \in \mathcal{L}$, $X = (null)X(K(nil))((cat)((unit)((head)X))((tail)X))$ (fixed point equation). The length lgh of $X \in \mathcal{L}$ can now be defined as $\mu i((tail)^i X = (nil))$, and it can be proved that for $X, Y \in \mathcal{L}$, $X = Y$ if and only if, for some n, $lgh(X) = lgh(Y) = n$ and for all $j = 1, 2, \ldots, n$, $(head)((tail)^j X) = (head)((tail)^j Y)$. It follows that each $X \in \mathcal{L}$ has the form (nil) (if $lgh(X) = 0$) or $(unit)x_1^\wedge \cdots ^\wedge (unit)x_n$ (if $lgh(X) = n > 0$), where '$^\wedge$' is an infix notation for concatenation. This justifies the notation $X = \langle x_1, \ldots, x_n\rangle$ or $X = \langle\rangle$. In addition, a term (lgh) representing lgh can be defined to be $\lambda xy.(null)x0(\sigma((lgh)((tail)x))y)$ (which exists by the fixed-point combinator), where σ is the Church successor function. Finally, the term $\lambda x.(cat)((tail)x)((unit)((head)x))$ has the effect of moving the first component of a sequence to the last position. The six postulates given above are satisfied by the six-tuple $(B, I, D, D\alpha K, \lambda xy.xy0, D0(K^2))$, where B is $\lambda xyz.x(yz)$, I is $\lambda x.x$, D is the pairing operator $\lambda xyz.zxy$, and α is an arbitrary term. In addition, they are satisfied by the six-tuple $(\Phi^n B, K^n I, C_{(n+2)}I, C_{(n+2)}I\alpha_1 \cdots \alpha^n 0), C_{(n+2)}I\alpha_1 \cdots \alpha_n 0(K^{n+2}0))$, where $\alpha_1, \ldots, \alpha_{n+1}$ are arbitrary terms, Φ is $\lambda fabx.f(ax)(bx)$, $\Phi^n = \lambda fabx_1 \cdots x_n.f(ax_1 ... x_n)(bx_1 \cdots x_n)$, $C_{(n)}$ is $\lambda fx_1 \cdots x_{n+1}.f(x_{n+1}x_1 \cdots x_n$, and $C_{(n+2)}I = \lambda x_1 \cdots x_{n+3}.x_{n+3}x_1 \cdots x_{n+2}$. The author then moves on to consider the case in which inside a linear finite sequence (lfs) each component is again an lfs and this recurs a finite number of times. This is a hereditary finite sequence (hfs). These hfs's may be thought of as trees. There are three possibilities: (1) the trees are unlabeled, in which case the only possible sequence at the end of a branch is $\langle\rangle$, (2) each node is labeled, in which case any term can be a label, and (3) only the ends of the branches are labeled. In the first case, we have structures such as $\langle\rangle$, $\langle\langle\langle\rangle\rangle\langle\rangle\rangle$, and $\langle\langle\langle\rangle\langle\rangle\rangle\langle\rangle\langle\rangle\rangle$. In the second case, the label is postfixed to the last component of every lfs forming the hfs, leading to such structures as $\langle x\rangle$, $\langle\langle\langle v\rangle u\rangle\langle w\rangle t\rangle$, and $\langle\langle\langle w\rangle\langle x\rangle v\rangle\langle y\rangle\langle z\rangle u\rangle$. In the third case, it is necessary to consider the possibility of the label occurring anywhere inside an lfs component and not only in the last position. This raises the problem of recognizing the labels, since they may be any terms. The solution is to identify these labels with their unit sequences, so that in

$\langle\langle\langle v\rangle\rangle\langle w\rangle\rangle$, the label is $\langle v\rangle$ and not v. Other examples are $\langle x\rangle$ and $\langle\langle\langle w\rangle\langle x\rangle\rangle\langle y\rangle\langle z\rangle\rangle$. (Tree diagrams for all these examples are given in the paper.) (Reviewer's remark: In footnote 2 on p. 241, the author states that in order to show that the given six-tuples of terms satisfy the six postulates it is necessary to use strong equality, which he identifies as having the rule $X = Y \Rightarrow \lambda x.X = \lambda x.Y$. In fact, the strong equality needs to satisfy the stronger rule that if x does not occur (free) in X or Y, then $Xx = Yx \Rightarrow X = Y$.) (For the entire collection see MR 81j:03005.) Reviewed by J. P. Seldin.

Hindley, R. and Longo, G. (1980). Lambda-calculus models and extensionality. *Z. Math. Logik Grundlag. Math.*, 26(4):289–310.

[MR81m:03019] This paper begins by noting the singular fact that the considerable quantity of work published on models of the lambda calculus prior to 1977 was carried out without the benefit of any published definition of what such models are. Moreover, the definitions implied by the works of various authors were inequivalent. Further, the definition published by H. Barendregt (Handbook of mathematical logic, Part D, pp. 1091 - 1132, North- Holland, Amsterdam, 1977; see MR 58 #10343) differs from the one favoured both by the present authors and by D. Scott in unpublished 1976 notes. That confused situation is resolved, in the reviewer's opinion, by this paper, which gives a meticuluous study of the issues. From the introduction: "In this paper we try to do four things: (i) to clarify the basic concepts involved, (ii) to give an acceptable and natural definition of λ-model, (iii) to explore extensionality and various forms of the ω-rule, and (iv) to set out clearly the relations between λ-models and combinatory models. All these goals are interdependent, and the key concept will turn out to be the relation of 'extensional equivalence', $a \sim b \Leftrightarrow \forall d\ ad = bd$." Further: "The present paper will be concerned only with type-free systems. But most of its comments will apply also to models of typed systems, with some obvious changes." Reviewed by John Staples.

Klop, J. W. (1980). Reduction cycles in combinatory logic. In *To H. B. Curry: essays on combinatory logic, lambda calculus and formalism*, pages 193–214. Academic Press, London.

[MR82b:03040] It is not hard to find λ-terms M whose reduction graph $\mathcal{G}(M)$ (the set of all reducts of M structured by the reduction relation) is finite and contains a cycle. Some examples are $\Omega \equiv (\lambda x.xx)(\lambda x.xx)$, WWW where $W \equiv \lambda xy.xyy$, $\omega(WI)$ where $\omega \equiv \lambda x.xx$ and $I \equiv \lambda x.x$, and $NNNN$ where $N \equiv \lambda abc.cbaa$. It is easy to find terms M_n such that $\mathcal{G}(M_n)$ is exactly a cycle of length n for all $n \geq 1$. Such a cycle is called a pure cycle. The main result of this paper is that in combinatory logic (CL) based on K, S or I, K, S, there are no pure cycles, and, in addition, there are no terms M such that $\mathcal{G}(M)$ is finite and contains a cycle. Actually, the main theorem is slightly stronger than this. Two terms are defined to be cyclic equivalent if each one reduces to the other. This is clearly an equivalence relation. The main theorem is that the cardinality of any cyclic equivalence class is 1 or $\aleph_0$ (in CL based on K, S or I, K, S). This theorem has the following corollaries: (i) if $\mathcal{G}(M)$

contains one cycle it contains infinitely many, (ii) finite reduction graphs are acyclic, (iii) there are no pure cycles, (iv) if a term has only finitely many reducts it must have a normal form, and (v) if $\mathcal{G}(M)$ contains a cycle it contains a noncyclic infinite reduction. The author also proves that terms lying on a prime cycle are unsolvable (i.e. have no head normal form; i.e. (in CL) have order 0). These results are not valid for other sets of basic combinators. If the basis is B, C, W, K, for example, then WWW reduces only to itself and thus is on a pure cycle of length one. If the basis is B, I, C, S for CLI (equivalent to the λI-calculus), then, for $\omega^* \equiv S(CI)(CI)$, $\omega^*\omega^*$ is on a cycle of length 6. If the basis is O, A, M, E, where the reduction rules are $Oxy \to y$, $Axyzu \to xz(yzu)$, $Mxyz \to x(yz)$, and $Exy \to yx$, although there appear to be no pure cycles, there are finite reduction graphs containing cycles. (This basis appeared in S. Stenlund's article (Combinators, λ-terms, and proof theory, Reidel, Dordrecht, 1972; MR 58 #21492).) The same is true of the basis I, J for CLI, where $Jabcd \to ab(adc)$, which is due to J.B. Rosser (Ann. of Math. (2) 36 (1935), 127 - 150; Zbl 11, 2; Duke Math. J. 1 (1935), 328 - 355; Zbl 12, 386). The author closes with some conjectures and questions. (For the entire collection see MR 81j:03005.) Reviewed by J. P. Seldin.

Lévy, J.-J. (1980). Optimal reductions in the lambda-calculus. In *To H. B. Curry: essays on combinatory logic, lambda calculus and formalism*, pages 159–191. Academic Press, London.

[MR83a:03012] Given a λ-term the problem arises of how to find in an efficient way its normal form (if it exists). Curry's leftmost reduction always finds the normal form but is not efficient, since it may happen that redexes are duplicated: (1) $D(Ia) \to Ia(Ia) \to a(Ia) \to aa$, where $D = \lambda x.xx$. Wadsworth studied so-called graph reductions in which copies of the same redex are contracted simultaneously. His algorithm is still not optimal, since redexes may be created: (2) $D(FI) \to FI(FI) \to Ia(Ia) \to a(Ia) \to aa$, where $F = \lambda x.xa$. Note that in (2) the two redex occurrences Ia are created in the same way. The principal definition in the present paper is that of the family of a redex (-occurrence in a certain reduction). It is shown that it is decidable whether two redexes have the same family. The author shows that if the members of a family are contracted simultaneously in one step and this goes on from the left to the right then an "optimal" reduction is obtained. Optimal here means of minimal "cost"; there are two assumptions on the cost of reductions: (a) If $\rho : M_0 \twoheadrightarrow M_1 \twoheadrightarrow \cdots \twoheadrightarrow M_n$ is a reduction such that in each transition $M_i \twoheadrightarrow M_{i+1}$ exactly the family in M_i of one redex is contracted simultaneously, then $cost(\rho) = n$. (b) If ρ is an arbitrary finite reduction, then $cost(\rho) \geq$ number of distinct families of redex occurrences that are contracted in ρ. In order to make requirement (a) realistic, an implementation of λ-reduction has to be constructed with a sharing mechanism. The author remarks that this problem remains open. But he has given a conceptual framework for a possible implementation of reduction that is optimal. (Although, by M. Blum's speed-up theorem J. Assoc. Comput. Mach. 1 4 (1967), 322 - 336; MR 38 #4213, there does not exist an absolutely optimal algorithm for finding normal forms, the notion of optimal in the sense that the number of contractions to be performed is minimized is still interesting.) Moreover, the notions on reduction introduced in the paper are interesting in their own right. (For the entire collection see MR 81j:03005.) Reviewed by Henk Barendregt.

Longo, G. and Venturini Zilli, M. (1980). A $\lambda\delta$-calculus with an algorithmic δ. *Arch. Math. Logik Grundlag.*, 20(1-2):41–52.

[MR81h:03032] Write $M \equiv N$ if M, N are terms of the λ-calculus that differ only by a change in bound variables. The δ-operator for $\lambda\beta$-calculus has the reduction rules (1) $\delta MN \to \lambda x.x$ if M and N are closed normal forms and $M \equiv N$, and (2) $\delta MN \to \lambda xy.y$ if M and N are closed normal forms and $M \not\equiv N$. Thus the δ-operator discriminates between equal and nonequal closed terms in normal form. The problem addressed here is how to define a new operator δ^* which discriminates between equal and nonequal closed terms in general, and to do this with reduction rules that satisfy the Church-Rosser property. The solution here is achieved by (a) restricting reductions to the contraction of left-most redexes, and (b) enlarging the set of terms to include "multiterms" which involve sequences $M_1, \ldots, M_n$, where each M_{i+1} is the result of contracting the left-most redex in M_i. The multiterms are defined so as always to include redexes (which may themselves be multiterms) so that the notion of normal form is unchanged from the usual one. Reviewed by B. Lercher.

Scott, D. (1980a). Lambda calculus: some models, some philosophy. In *The Kleene Symposium (Proc. Sympos., Univ. Wisconsin, Madison, Wis., 1978)*, pages 223–265. North-Holland, Amsterdam.

[MR82d:03024] The subject matter of lambda calculus started as a formal theory; only much later were models discovered and discussed. By describing a simple model of continuous set mappings (essentially the author's graph model (see SIAM J. Comput. 5 (1976), no. 3, 522 - 587; MR 55 #10262)) the author gives arguments that the historical order could have been the other way around. The model depends on coding finite sequences. If this coding is well-founded, then the paradoxical combinator Y denotes the least fixed point operation; otherwise this need not be so. The rule of induction is a tool for establishing valid equations (and inequalities) in the well-founded case. A construction is given of a model like D_∞ within the present model. It is discussed how (partial) equivalence relations give rise to type considerations. Finally the use of sets versus functions for foundational purposes is discussed. (For the entire collection see MR 81j:03009.) Reviewed by Henk Barendregt.

Scott, D. S. (1980b). Relating theories of the λ-calculus. In *To H. B. Curry: essays on combinatory logic, lambda calculus and formalism*, pages 403–450. Academic Press, London.

[MR82d:03025] The correspondence between typed λ-calculus and Cartesian closed categories (Ccc's) is well known. An object U in such a category C is called a domain if U^U is a retract of U. Given a domain U (and the retraction maps) in a Ccc C, one can extend the mentioned correspondence to type-free λ-calculus and construct a λ-theory $T(C, U)$. Conversely, by transforming a λ-theory T into a Ccc C_T, the author shows that all λ-theories can be obtained this way ($T = T(C_T, \lambda x.x)$). Because a Ccc C may fail to have enough points (is not concrete), in general one cannot construct a (weakly) extensional λ-model from C and a domain U. By embedding C into the topos set C^{op} and using the internal intuitionistic logic, the theory obtained from C, U "satisfies" (weak) extensionality. Moreover one may employ full higher order intuitionistic type theory. Since the embedding of C in the topos is full and faithful, the extended theory of the topos is conservative over the old one (corresponding to C). The author draws as conclusions that the typed λ-calculus is more fundamental than the type-free one; moreover the "functions" occurring in the λ-calculi may be considered as "real" functions that can be defined using, e.g., comprehension principles as long as this is done intuitionistically. The author gives full details of the necessary categorical constructions. Reviewer's remarks: If one allows λ-algebras, not necessarily satisfying (weak) extensionality, i.e. $(\forall x M = N) \to \lambda x.M = \lambda x.N$, then some of the results can be stated as follows. For each Ccc C with domain U there is a corresponding λ-algebra $\mathfrak{M}(C, U)$. (The author's construction of C_T is in fact $C_{\mathfrak{M}}$ with $\mathfrak{M}$ the closed term model of T.) If there are enough arrows $1 \to U$ (1 is the terminal object), then $\mathfrak{M}(C, U)$ is (weakly) extensional. Conversely for every λ-algebra $\mathfrak{M}$ there is a Ccc $C_{\mathfrak{M}}$ with domain $U = \lambda x.x$ such that $\mathfrak{M} \cong \mathfrak{M}(C_{\mathfrak{M}}, U)$. This gives an elegant description of the notions of λ-algebra vs. λ-model. Moreover, even if $\mathfrak{M}$ is not always (weakly) extensional, according to the internal logic of the topos corresponding to $C_{\mathfrak{M}}$ it is.) (For the entire collection see MR 81j:03005.) Reviewed by Henk Barendregt.

von Rimscha, M. (1980). Mengentheoretische Modelle des λK-Kalküls. *Arch. Math. Logik Grundlag.*, 20(1-2):65–73.

[MR81g:03012] In ordinary set theory it is impossible to construct models of the λ-calculus which are natural in the sense that the individuals of the model are functions in the set-theoretic sense and application is interpreted as ordinary set-theoretic application, because the axiom of foundation excludes self-application. Any set-theoretic models of the λ-calculus which are natural (in the above sense) have to be constructed in set theories without this axiom. The author shows that such a natural model can be constructed in ZFC^0 (ZFC without foundation) to which is added the universality axiom $U1$: For every extensional relation x there is a transitive set u such that x is isomorphic to $\epsilon \cap u^2$, where $\epsilon = \{\langle x, y\rangle : x \in y\}$ and a relation x is extensional just when, for all u and v in its field (the union of its domain and range), $u = v$ if and only if for all y, $\langle y, u\rangle \in x$ when and only when $\langle y, v\rangle \in x$. It is known that $ZFC^0 + U1$ is relatively consistent to ZFC^0. A model of the $\lambda\beta K$-calculus with the ω-rule (if $XZ = YZ$ for all closed terms Z, then $X = Y$) is then constructed as follows: The symbols, terms, and equations of the $\lambda\beta K$-calculus are coded in the set theory ($ZFC^0 + U1$) in such a way that the codes are related to one another like urelemente. An equivalence relation between codes for closed terms is defined by relating the codes of terms which can be proved equal in the $\lambda\beta K$-calculus with the ω-rule. The set C is defined to consist of one representative of each equivalence class of this equivalence relation. The application of one code in C to another is defined to be the code of the corresponding application term. For each element a of C there is a function f_a from C into itself whose value for x is the representative in C of a applied to x. Then each element a of C is identified with f_a by means of a relation E whose field D includes C. Because of the ω-rule, the relation E is extensional. Hence, by $U1$, there is a transitive set u and a bijection f from D onto u such that for any x and y in D, $\langle x, y\rangle \in E$ if and only if $f(x) \in f(y)$. The model is then the image of C under f. (C itself is a model, but it is not natural.) The author then extends the construction of this model in two ways. The first is to obtain a model of the $\lambda\beta K$-calculus without the ω-rule. In order to have a model in which the equality of this calculus is exactly reflected, it is necessary to distinguish between functions which are extensionally equal but which are not provably equal. This the author does by modifying the construction so that each term is represented by an ordered pair consisting of an ordinal number and a function. The second involves using a set theory in which there is assumed the axiom of strong extensionality, $Sext$: If there is an ϵ-isomorphism F from the transitive closure of x to the transitive closure of y such that $F(x) = y$, then $x = y$. (The transitive closure of x is the smallest set which includes x and all the elements of each of its elements.) This axiom is, in the presence of ZFC^0, incompatible with $U1$, but $U1$ can be weakened to an axiom which is consistent with $ZFC^0 + Sext$ relative to ZFC^0, namely $U2$: For each extensional relation x with the property that for any a and b in its field, if there is an x-isomorphism F from the transitive closure of a with respect to x to the transitive closure of b with respect to x such that $F(a) = b$ then $a = b$, there is a transitive set u such that x is isomorphic to $\epsilon \cap u^2$. The author concludes the paper by showing how to modify his construction to produce a model of the $\lambda\beta K$-calculus with the ω-rule in $ZFC^0 + U2$. (Reviewer's remark: The author refers to the $\lambda\beta K$-calculus with the ω-rule as the "extensional λK-calculus". This is not the usual terminology. Usually the extensional λK-calculus refers to the $\lambda\eta K$-calculus (which includes the axiom (η): If x does not occur free in U then $\lambda x.Ux = U$; this is equivalent to the rule: If x does not occur free in X and Y and $Xx = Yx$ then $X = Y$). It is known that the $\lambda\beta K$-calculus with the ω-rule is not as strong as the $\lambda\eta K$-calculus.) Reviewed by J. P. Seldin.

Wadsworth, C. P. (1980). Some unusual λ-calculus numeral systems. In *To H. B. Curry: essays on combinatory logic, lambda calculus and formalism*, pages 215–230. Academic Press, London.

[MR82b:03043] A sequence of λ-terms $(0), (1), \ldots$ is called a $\beta\eta$-adequate numeral system if all recursive functions can be λ-defined with respect to this sequence in the $\lambda\beta\eta$-calculus. It is easy to show that this is the case if there are terms s, p, z such that in $\lambda\beta\eta$ one has $s(n) = (n+1)$, $p(n+1) = (n)$, $z(0) = K$, $z(n+1) = KI$ for all n. The author constructs two $\beta\eta$-adequate numeral systems such that the number of initial λ's of (n) strictly increases with n: $(n)_1 = \lambda x_0 x_1 \cdots x_n . x_0(x_1(x_2(\cdots(x_n(KK))\cdots)))x_1 \cdots x_n$. (Take $s_1 = \lambda nxy.n(\lambda a.x(ya)y)$, $p_1 = \lambda nx.n(\lambda a.K(xa))I$, and $z_1 = \lambda n.nI(K(K(KI)))$.) $(n)_2 = \lambda x_0 x_1 \cdots x_{n+1}.x_1 x_2 \cdots x_n(x_0 x_{n+1})$. (Take $s_2 = \lambda nxyz.nx(yz)$, $p_2 = \lambda nx.nxI$, and

$z_2 = \lambda n.nK(n(\lambda a.aK)I)(KI)$.) Finally a $\beta\eta$-adequate system of Böhm is mentioned: $(n)_3 = \lambda x_1x_2\cdots x_{n+2}.x_1x_2\cdots x_nx_{n+2}x_{n+1}$. (Take $s_3 = \lambda nxy.n(xy)$, $p_3 = \lambda n.nI$, and $z_3 = \lambda n.n(n(K(KI)))(KK)$.) (For the entire collection see MR 81j:03005.) Reviewed by Henk Barendregt.

Barendregt, H. P. (1981). *The lambda calculus.* North-Holland Publishing Co., Amsterdam. Its syntax and semantics.

[MR83b:03016] There is no obvious claimant to the title of the standard model for the λ-calculus, so that the question arises of the "reality" behind the syntactical rules. To be sure, models are not the only kind of reality to be considered; the computations encoded in λ-calculus terms are also a reality, perhaps more fundamental than that of the set-theoretic models. This book is a massive and detailed examination of both kinds of reality for the pure λ-calculus, i.e. λ-calculus as a theory of functions or computations, without type distinctions or added logical notions. It brings together virtually all of the current knowledge on the subject and will be indispensable for serious students of the λ-calculus. There is surprisingly little in common between this book and those of H. B. Curry and R. Feys [Combinatory logic, Vol. I, North-Holland, Amsterdam, 1958; MR 20 #817] and Curry, J. R. Hindley and J. P. Seldin [Combinatory logic, Vol. II, North-Holland, Amsterdam, 1972; Zbl 242 #2029], which emphasize formal systems, especially those including "illative" (logical) notions. A useful feature of the book is the division between the general introduction in the first five chapters (124 pages) and the highly detailed and often intricate developments in the later chapters. The introduction outlines the main ideas, previews the later chapters, and includes some important proofs, e.g., the short proof of the Church - Rosser theorem due to W. W. Tait and P. Martin-Löf. Some of the high points of the remainder of the book are as follows. Chapters 6, 7, and 9 present approximately what was known in the 1930's: λ-definability and the relation to the theory of combinators. Chapters 8 and 10 cover solvability and Böhm's theorem that normal forms are separable. Chapters 11 through 15 analyze several notions of reduction. The Church-Rosser theorem is proved again, this time by the "classical" method of following the residuals of redexes through reductions. Reduction strategies, including standardization, are studied. "Sensible" theories (which identify the unsolvable terms) are proved consistent in Chapter 16. Chapter 17 includes Plotkin's theorem that λ-calculus is not ω-complete. In Chapter 18 we get the construction of the projective model D_∞ and the graph model P_ω. In Chapter 19 both models are studied and it is shown that both are sensible and that Curry's fixed-point combinator is the least fixed-point operator (Park's theorem). Chapter 20 includes the result that Church's δ operator is not λ-definable. The last chapter discusses combinatory semigroups and groups. Finally there are appendices on typed λ-calculus and illative combinatory logic (15 pages) and on schemes for making substitutions that involve changes in bound variables. Reviewed by B. Lercher.

Ronchi Della Rocca, S. (1981). Discriminability of infinite sets of terms in the D_∞-models of the λ-calculus. In *CAAP '81 (Proc. Sixth Colloq.,* *Genoa, 1981)*, pages 350–364. Springer, Berlin.

[**MR83c:03011**] This is a clearly and simply written paper collecting together some results about several kinds of discriminability and enumerability for infinite sequences of closed lambda-terms. All sequences are assumed to be "proper", i.e. their members denote distinct members of Scott's model D_∞. A sequence $F = (F_0, F_1, \dots)$ is said to be discriminable (relative to i) if $\exists i, D\ DF_i =_{\beta\eta} \lambda xy.x$ and $\forall j \neq i\ DF_j =_{\beta\eta} \lambda xy.y$; F is "completely discriminable" if $\exists D \forall i\ DF_iF_i =_{\beta\eta} \lambda xy.x$ and $\forall j \neq i\ DF_iF_j =_{\beta\eta} \lambda xy.y$. Also, F is called a "numeral system" if F has successor, predecessor and zero-test in an obvious sense. Sample results: Any completely discriminable sequence with successor is a numeral system. There exist sequences with any two of successor, predecessor, zero-test but not the third. Also conditions for discriminability of F are given in terms of the approximants of F's members. (Reviewer's remarks: (1) The author calls her topic "sets", but this is a slip of the pen; she really means "sequences" throughout (otherwise the proof of Theorem 1 would fail). (2) In Theorems 1 and 2, it is assumed that properness is defined as a property of sequence-generators, not of sequences (or sets) in general.) (For the entire collection see MR 82f:68002.) Reviewed by R. Hindley.

Staples, J. (1981). Efficient combinatory reduction. *Z. Math. Logik Grundlag. Math.*, 27(5):391–402.

[**MR83a:03013**] From the introduction: "In combinatory logics, terms with normal form generally have many different reductions to normal form, of differing lengths. In this paper we describe, for a conventional form of weak combinatory logic, a reduction procedure which is minimal (that is, prescribes reductions to normal form comprising a minimum number of contractions) and complete (that is, it always succeeds in reducing to normal form terms which have normal forms). "These properties are also possessed by trivial searching algorithms, in which one considers for a given term first the reduction of length zero, then all reductions of length one and so on until a reduction to normal form is encountered. The procedure we describe is less trivial, and its description does more to develop the study of efficiency, but it does not directly lead to algorithms any more practicable than the searching algorithms … because an impracticably large amount of computation may be required in order to find the optimal reduction prescribed by the theory." The author begins by proving independently of the Church-Rosser theorem that if a term has a reduction to a normal form then it has a left reduction (i.e. a reduction in which the redex contracted at each step is the leftmost one) to the same normal form. He then defines a dummy subterm of a term; it is a subterm which plays a role in the reduction similar to that of Y in a reduction to a normal form of KXY (any such reduction must eventually eliminate a descendent of Y). A subterm U of a term T with a normal form is said to be in quasinormal form if all subterms of U which are weak redexes (i.e. are of the form KXY or $SXYZ$) are dummy subterms of T. A quasinormal contraction (q-contraction) is a left contraction of a subterm U of T such that all proper subterms of U are in quasinormal form. The optimal reduction procedure consists of repeated q-contractions. This procedure is shown to be optimal by a proof that it satisfies the hypotheses of the main theorem in an earlier paper by the author Bull. Austral. Math. Soc. 16 (1977), no. 3, 341 - 349; MR 57 #11159. Reviewed by J. P. Seldin.

Bergstra, J. A. and Klop, J. W. (1982). Strong normalization and perpetual reductions in the lambda calculus. *Elektron. Informationsverarb. Kybernet.*, 18(7-8):403–417.

[MR85e:03028] Authors' summary: "A lambda term M is said to be strongly normalizing if every reduction of M stops eventually (in the unique normal form), and weakly normalizing if some reduction of M ends in the normal form. Here we are interested in characterizing those reduction steps $M \to M'$ such that if M has an infinite reduction then M' has one too. A sufficient condition is that in the step $M \to M'$ a redex $R \equiv (\lambda x.A)B$ is contracted, where x is free in A, i.e. R does not erase its 'argument'. (A corollary is the well-known fact that, in the λI-calculus, strong and weak normalization are equivalent.) An R whose contraction preserves the property of having an infinite reduction is said to be perpetual. In the present paper the perpetual redexes which do erase their argument are characterized."

Dezani-Ciancaglini, M. and Ermine, F. (1982). Maximal monoids of normal form. *Fund. Inform.*, 5(2):129–141.

[MR84b:03028] The set of λ-terms with composition $\circ \equiv \lambda z.M(Nz)$ forms a semigroup. The set NF of normal forms is not closed under $\circ$. The authors construct a maximal (with respect to $\subseteq$) subset $\mathcal{N}$ of NF closed under $\circ$. This is done via the generalized type assignment introduced by M. Coppo and Dezani-Ciancaglini [Arch. Math. Logik Grundlag. 19 (1978/79), no. 3-4, 139 - 156; MR 81d:03014]. Restricted to closed terms one has $\mathcal{N} = \{M \mid\vdash (0)0M\}$. Since $I \in \mathcal{N}$ it is a monoid. Reviewed by Henk Barendregt.

Koymans, C. P. J. (1982). Models of the lambda calculus. *Inform. and Control*, 52(3):306–332.

[MR85b:03020] The author develops a description of models of the untyped λ-calculus suggested by D. Scott [To H. B. Curry : essays on combinatory logic, lambda calculus and formalism, 403–450, Academic Press, London, 1980; MR 82d:03025], based on the notion of Cartesian closed category (C.c.c.). In recent years a number of definitions of λ-calculus models have been given. An outcome of many of these definitions (Hindley and Longo, Barendregt, Meyer) is the need for a distinction between λ-algebras (i.e. structures which satisfy all provable β-equalities) and λ-models (which also satisfy the axiom of weak extensionality). There are many reasons for this distinction: for example, the λ-algebras form an equational class but the λ-models do not. The categorical description has the advantage of unifying the two concepts. Each C.c.c. C with a reflexive object (i.e. an element $U \in \mathrm{Ob}(C)$ such that U^U is a retraction of U) yields a λ-algebra $\mathfrak{M}'_C$. Moreover $\mathfrak{M}'_C$ is a λ-model if and only if C has enough points at U (i.e. if and only if for all $f, g \colon U \to U$ with $f \neq g$ there exists an arrow $x \colon T \to U$, where T is the terminal object, such that $f \circ x \neq g \circ x$) and $\mathfrak{M}'_C$ is extensional if and only if U^U is isomorphic to U. Conversely,

every λ-algebra $\mathfrak{M}$ can be obtained from a C.c.c. $C_{\mathfrak{M}}$ with a reflexive object. The category $C_{\mathfrak{M}}$ is of very special nature and can be described in terms of an algebraic structure called a Cartesian closed monoid. In particular, with each λ-algebra $\mathfrak{M}$ can be associated a Cartesian closed monoid $M_{\mathfrak{M}}$ which generates $C_{\mathfrak{M}}$. Lastly, in the case of a concrete C.c.c. the interpretation of terms reduces to the well-known environmental interpretation. Note that in a concrete C.c.c. each object has enough points. Reviewed by M. Coppo.

Meyer, A. R. (1982). What is a model of the lambda calculus? *Inform. and Control*, 52(1):87–122.

[MR84g:03022] Although the λ-calculus is fifty years old, its model theory is the work of the last decade. Since the first model introduced by D. Scott about 1970, a number of models and techniques for constructing them have become available. But for a long time there was no clear statement of what a model of the λ-calculus is, and when the first definitions of such models began to appear, there was some disagreement as to just what the definition should be. This expository paper discusses why there is an apparent difficulty in making this definition and how that difficulty is overcome. The main definition used by the author is that of a combinatory model, which is a set D together with a binary operation $\cdot$ and three elements K, S, and ϵ of D satisfying: $(K\cdot d_0)\cdot d_1 = d_0$; $((S\cdot d_0)\cdot d_1)\cdot d_2 = (d_0\cdot d_2)\cdot(d_1\cdot d_2)$; $(\epsilon\cdot d_0)\cdot d_1 = d_0\cdot d_1$; if, for all $d \in D$, $d_0\cdot d = d_1\cdot d$, then $\epsilon\cdot d_0 = \epsilon\cdot d_1$; and $\epsilon\cdot\epsilon = \epsilon$. This is for the $\lambda\beta$-calculus; for the $\lambda\eta$-calculus, the required structure is an extensional combinatory model, which is like a combinatory model without ϵ which satisfies: if, for all $d \in D$, $d_0 \cdot d = d_1 \cdot d$, then $d_0 = d_1$. To keep the paper self-contained, the author begins with a review of the syntax of the λ-calculus. He next formulates the notion of an environment model, and introduces enough of its properties to explain the view that it is the natural, most general formulation of what mathematical models of the λ-calculus might be. Its drawback is that it uses induction on the syntactic structure of λ-terms to define purely algebraic conditions. For this reason, the notions of combinatory model and extensional combinatory model are preferable, and the author proves that there is an equivalence between these and environment models. As an easy application of the notion of combinatory model, the author shows that the construction of a λ-calculus model by E. Engeler [Algebra Universalis 13 (1981), 389 - 392; MR 83f:03012] follows more simply and more generally than was shown there. This completes the main part of the paper. But the author goes on to two further topics. The first concerns the fact that the combinators K and S of a combinatory model are not necessarily uniquely determined; the author defines a λ-model to be essentially a combinatory model in which K and S satisfy certain algebraic conditions which guarantee that they behave in all important respects like $\lambda xy.x$ and $\lambda xyz.xz(yz)$ and then proves an equivalence between these and combinatory models. The second concerns the fact that combinatory models are not equationally axiomatized structures and thus are not in general closed under taking substructures and morphisms. The author presents λ-algebras, which are essentially combinatory algebras (combinatory models without ϵ) in which K and S satisfy the equations which are added to weak equality to produce combinatory β-equality (e.g., the set called (ω_β) by H. B. Curry and R. Feys [Combinatory logic, Vol. I, see p. 203, North-Holland, Amsterdam, 1958; MR 20 #817]). The author shows that every λ-model is a λ-algebra and, given any λ-algebra, the free combinatory algebra generated by an infinite set of variables over the λ-algebra is a λ-model. A corollary due to H. Barendregt is that λ-algebras constitute the class of all substructures of λ-models and the class of all homomorphic images of λ-models. But in λ-algebras, the rule (ξ), $u = v \Rightarrow \lambda x.u = \lambda x.v$, is not necessarily satisfied, and hence, in the author's opinion, they are not all suitable as models for λ-calculus. All of the long proofs are placed in the appendix. (Reviewer's remark: The author credits to Curry the axioms used to define a λ-algebra. It is true that Curry is responsible for these particular axioms and also for the original idea of using such equations as axioms for obtaining extensional (i.e., η-) equality from weak equality. But the first set of such equations for obtaining β-equality from weak equality is really due to J.B. Rosser [Ann. of Math. (2) 36 (1935), 127 - 150; Zbl 11, 2]. J. Lambek [To H.B. Curry: essays on combinatory logic, λ-calculus and formalism, 375 - 402, Academic Press, London, 1980; MR 81m:03020] used the term "Curry algebras" for extensional combinatory models and "Schönfinkel algebras" for combinatory algebras; following this terminology it might be appropriate to use the term "Rosser algebra" for a λ-algebra.) Reviewed by J. P. Seldin.

Robinet, B. (1982). Sur des séquences itératives de combinateurs. *C. R. Acad. Sci. Paris Sér. I Math.*, 295(1):29–30.

[MR83h:03020] Author's summary: "In any iterative sequence of combinators, every term can be expressed uniquely as a function of Church's iterators and combinatory constants associated with the sequence."

Ronchi della Rocca, S. (1982). Characterization theorems for a filter lambda model. *Inform. and Control*, 54(3):201–216.

[MR85k:03009] Each term of the untyped lambda calculus can be assigned a set of types in such a way that the assignment of types to terms is complete, with respect to a natural semantics [see H. Barendregt et al., J. Symbolic Logic 48 (1983), no. 4, 931–940; MR 85j:03014; J. R. Hindley, Proceedings of the international symposium on programming (Turin, 1982), 212–226, Springer, Berlin, 1982; Zbl 514:03009]. Each of these sets of types is in a natural sense a filter. The completeness argument of Barendregt et al. constructs a model for the untyped lambda calculus (a λ-model) which is made up entirely of filters of types. In this paper the author studies the structure of that filter λ-model.

One of the two main results is that the equivalence relation on terms defined by equality in the filter λ-model is the relation of having the same set of approximate normal forms. This is the same equivalence relation as is defined by the equalities of the (previously known) lambda models P_ω and T^ω.

The other main result is a syntactic characterisation of the preorder defined on terms by the inclusion relation of the filter λ-model. This result shows that the filter λ-model inclusion relation is different from the inclusion relations of P_ω and T^ω. Reviewed by John Staples.

Longo, G. (1983). Set-theoretical models of λ-calculus: theories, expansions, isomorphisms. *Ann. Pure Appl. Logic*, 24(2):153–188.

[**MR85g:03025**] The author studies E. Engeler's variant D_A [Algebra Universalis 13 (1981), no. 3, 389–392; MR 83f:03012] of the graph λ-model $\mathcal{P}\omega$ of Plotkin and Scott in a general setting. A PSE-algebra is a nonempty set B such that (1) $\beta \subseteq B$ and $b \in B \Leftrightarrow (\beta, b) \in B$, where β ranges over finite sets and (-,-) is the ordered pair. Given such a B, one defines for elements of the power set $\mathcal{P}B$, $d.e = \{b : \exists\beta \subseteq e(\beta, b) \in d\}$ and $\lambda x.f(x) = \{(\beta, b) : b \in f(\beta)\}$ for Scott-continuous $f : \mathcal{P}B \to \mathcal{P}B$. Thus $\langle \mathcal{P}B, .\rangle$ becomes a λ-model. If one starts with a nonempty set A and constructs the least $B \supseteq A$ such that (1) holds, then the resulting model is D_A.

By using an appropriate notion of approximation it is shown that $D_A \models MRN \Leftrightarrow \mathrm{BT}(M) R\, \mathrm{BT}(N)$, where R is $\subseteq$ or $=$ and $\mathrm{BT}(M)$ is the Böhm tree of the term M. By defining alternative abstraction in D_A as $\lambda_A^+.f(x) = \lambda x.f(x) \cup A$, a related λ-model D_A^+ is obtained. Then the author shows that $D_A^+ \models M = N \Leftrightarrow T(M) = T(N)$, where $T(M)$ is a tree defined like the Böhm tree of M, except that the order of an unsolvable is taken into account: $T(M) = \lambda x_1 \cdots x_n.\perp$ if M is unsolvable of order n; $T(M) = T$ if M is unsolvable of order ∞. Moreover the terms of order n and also the unsolvable terms are characterized semantically in D_A^+.

The notion of λ-categorical combinatory algebra is introduced; in these structures there is a unique way to introduce λ-abstraction. It is shown that "atomless" PSE-algebras are λ-categorical. The models D_A are not, since the D_A^+ exist.

It is indicated how the algebras D_A can be collapsed by identifications on the ground set A. For example $a \sim (\{a\}, a)$ yields a model $(D_A^\sim, .\ , \lambda^+)$ isomorphic to the λ-model introduced by D. S. Scott [The Kleene symposium (Madison, Wis., 1978), 223–265, North-Holland, Amsterdam, 1980; MR 82d:03024]. The graph model $\mathcal{P}\omega$ is isomorphic to $D_A^\sim$ with $A = \{a\}$ and $a \sim (\varnothing, a)$.

It is shown that for countable nonempty A the structures $(D_A, .)$ and $(\mathcal{P}\omega, .)$ are embeddable into each other, but they are not isomorphic.

The paper states several open problems and contains many interesting discussions. Reviewed by Henk Barendregt.

Margaria, I. and Zacchi, M. (1983). Right and left invertibility in λ-β-calculus. *RAIRO Inform. Théor.*, 17(1):71–88.

[**MR85d:03027**] Authors' summary: "A characterization of λ-terms having left and/or right inverses in λ-β-calculus is given and the sets of all and only λ-terms left/right invertible are constructed. The results are obtained using the concept of Böhm tree, so this study is further used to characterize the λ-terms left/right invertible in the graph model P_ω."

Venturini Zilli, M. (1983). Cofinality in reduction graphs. In *CAAP '83 (L'Aquila, 1983)*, pages 405–416. Springer, Berlin.

[**MR85j:68071**] Author's summary: "Reductions from an arbitrary term of the $\lambda\beta$-calculus are considered in this paper in the framework of the reduction graph that can be associated to that term. The graph approach allows us to have a global view of all reductions starting from a term with a criterion of significance for reductions according to which cofinal reductions are the most significant ones. A new class of cofinal reductions is defined, namely, the quasicomplete reductions. A sufficient condition (conjectured to be also necessary) concerning cofinality in reduction graphs, which is weaker than one already known, is stated." For the entire collection see MR 85d:68006.

Barendregt, H. P. (1984a). Introduction to lambda calculus. *Nieuw Arch. Wisk. (4)*, 2(3):337–372.

[MR86k:03007] From the introduction: "This paper is an introduction to the type-free lambda calculus. Section 1 gives some intuition about the basic operations of the λ-calculus and some examples of inductive definitions and proofs. In Section 2, λ-terms and conversion (provable equality) are introduced and it is proved that all recursive functions are represented in the λ-calculus. In Section 3, the notion of λ-model in the category of complete lattices is introduced and the construction of a particular such model D_A is given. Section 4 is about reduction and gives a dynamical and proof-theoretical analysis of conversion. In the concluding Section 5, the notion of the Böhm tree of a λ-term is introduced. It is proved that terms with the same Böhm tree are equal in the model D_A, giving an interesting consistent extension of the λ-calculus.

"The methods in Section 2 and Section 4 are algebraic, that is, finitary in character. In Section 3 limits play a role, so it has an infinitary character. In Section 5 an interaction of the algebraic and limit aspects plays an important role. The sections end with some exercises; the more difficult ones have one or more stars."

Barendregt, H. P. (1984b). *The lambda calculus*. North-Holland Publishing Co., Amsterdam, revised edition. Its syntax and semantics.

[MR86a:03012] From the preface: "In the first edition of this book [1981; MR 83b:03016] the notions of lambda algebra and lambda model were introduced in a correct but somewhat elaborate way. Recent work of others has made it possible to give an elegant categorical description of both kinds of models. This leads to the main changes in this edition: a complete rewriting of Chapter 5 introducing models with some consequent changes in Chapter 18 constructing models. Moreover, some additional material has been included at the end of the book and several corrections have been made."

Bruce, K. and Longo, G. (1984). On combinatory algebras and their expansions. *Theoret. Comput. Sci.*, 31(1-2):31–40.

[MR86a:03013] This paper is concerned with models for the lambda calculus. The authors introduce the notion of a combinatory algebra, which is a semigroup $(D, \cdot)$ in which there are elements k and s such that, for all x, y, z in D, $kxy = x$ and $sxyz = xz(yz)$. A characterization is given of those combinatory algebras in which k and s are uniquely determined. There is a discussion of D. S. Scott's model of the lambda calculus [SIAM J. Comput. 5 (1976), no. 3, 522–587; MR 55 #10262] and its connection with other algebraic models of Plotkin and Engeler [E. Engeler, Algebra Universalis 13 (1981), no. 3, 389–392; MR 83f:03012]. (There is at least one misprint: on page 32, (1), the map is from $(D \to D)$ to $(D \to D)$.)

Giannini, P. and Longo, G. (1984). Effectively given domains and lambda-calculus models. *Inform. and Control*, 62(1):36–63.

[MR86j:68083] The notion of computability in abstract structures (in particular, T_0-spaces) has been studied by several authors, such as D. Scott and Yu. L. Ershov . This paper contains interesting results concerning computability properties of the interpretation of a formal language (the λ-calculus) in structures which give λ-algebras or λ-models [see H. P. Barendregt , The λ-calculus, its syntax and seman-

tics, North-Holland, Amsterdam, 1984; MR 86a:03012] as well as different notions of computability over models.

This paper is based on Ershov's notion of (effective) f-space [see, e.g., Logic colloquium 76 (Oxford, 1976), 455–467, North-Holland, Amsterdam, 1977; MR 58 #21541]. Many structures which give λ-models and λ-algebras are, indeed, (effective) f_0-spaces (observe that λ-algebras obtained as the interior of topological λ-models are typically not domains in Scott's sense).

In the first part of the paper the authors give the basic definitions relating to f-spaces and prove some basic properties, pointing out relations with domains. They also give the Rice-Shapiro and Myhill- Shepherdson theorems [see H. Rogers , Theory of recursive functions and effective computability, McGraw-Hill, New York, 1967; MR 37 #61] in the most general setting. Moreover, they discuss some well-known λ-models (P_ω, T_ω and B, the set of Böhm trees [see Barendregt, op. cit.]), pointing out some interesting properties from the point of view of computability. For example, the interiors of T_ω and B can be turned into constructive f_0-spaces, while the interior of P_ω cannot (roughly, an f_0-space is constructive if it is possible to give an effective enumeration of all its elements, not only of the finite ones).

Some of the previous results are then used to prove properties of the interpretation of λ-terms in λ-algebras. The interpretation of a λ-term turns out to be an effective morphism. Some consequences of this, mainly concerning the interpretation in P_ω, are investigated. Topological properties of application and of normalizable terms are also proved, as well as the genericity lemma [see Barendregt, op. cit.]. Moreover the relation of a "canonical" topology on some models (the tree topology, see Barendregt [op. cit.]) to their structures as f_0-spaces is studied.

From a more general point of view it is shown that the canonical (tree) topology can always be turned into an f-space structure for λ-algebras whose theory is an extension of the theory B of Böhm tree equality (i.e. $\mathrm{B} \vdash M = N$ if and only if $\mathrm{BT}(M) = \mathrm{BT}(N)$). In particular, given any λ-theory T which equates all unsolvable terms, the (term) model of "finite depth" extensions $T^\sim$ of B, obtained from T, has a canonical topology with an f-structure if and only if $T^\sim = \mathrm{B}$.

The last section deals with the notion of computable function in effectively given f_0-spaces which yield λ-models. It turns out that a function is computable as an element of the function space (which is again an effectively given domain) if and only if its image, under the required embeddings, is computable. The proof of the equivalence is carried out both in graph models (where a function is considered as an element via a coding of its graph) and in models obtained by an inverse limit construction. Reviewed by M. Coppo.

Kanda, A. (1984). Numeration models of λ-calculus. In *Ninth colloquium on trees in algebra and programming (Bordeaux, 1984)*, pages 155–168. Cambridge Univ. Press, Cambridge.

[MR787458] This item will not be reviewed individually. For details of the collection in which this item appears see 86b:68001.

Koymans, C. P. J. (1984). *Models of the lambda calculus*. Stichting Mathematisch Centrum Centrum voor Wiskunde en Informatica, Amsterdam.

[MR87m:03017] The untyped λ-calculus was developed by A. Church in the 1930s as a purely formal theory of functions under the operation of application (or evaluation). Closely related theories were independently discovered by Schönfinkel and

Curry in the 1920s. The extreme generality of these systems allows functional application $f(a)$ to make sense for any argument a, even self-application $f(f)$! In 1969, D. S. Scott constructed the first genuine mathematical models of such theories. The subsequent decade (1970–1980) saw rapid development of this semantics, together with many applications to computer science. This monograph surveys, extends, and classifies recent mathematical work.

In Chapter 1 the author discusses applicative structures, i.e. sets with a binary operation of "application" and sufficient additional data to interpret λ-terms and their equations. Extensive refinements of these "λ-algebras" due to Meyer and D. S. Scott characterize, among other things, "λ-models" satisfying the first-order weak extensionality axiom: (ξ) $\forall x(M = N) \Rightarrow \lambda xM = \lambda xN$. The author also discusses homomorphisms of λ-algebras as well as concrete models, e.g., Scott's D_∞ and the Plotkin-Scott graph model $P\omega$.

A mathematically more natural approach, using category theory, is discussed in Chapter 2. A categorical λ-algebra in a Cartesian closed category is an object U such that $[U \to U]$ is a retract of U ($[U \to U]$ being the internal function space). All constructions in Chapter 1 may be usefully viewed in this way. For example, extensional algebras (models of $\lambda\beta\eta$-calculus) satisfy the stronger condition that $[U \to U]$ is isomorphic to U, whereas weak extensionality (ξ) above says the terminal object is a kind of generator (restricted to $\mathrm{End}(U)$). The endomorphism monoid $\mathrm{End}(U)$ of the object U in a categorical λ-algebra has considerably influenced recent research. D. S. Scott, in unpublished lectures, axiomatized such "lambda monoids" and showed how they arise by a general categorical construction: forming the idempotent splitting completion of a category. The author surveys and extends this work to characterize λ-algebras and models.

In Chapter 3, the author, using categorical methods, explicates an interesting technique of D. S. Scott for iteratively constructing D_∞-like models internally in the graph model $P\omega$. Finally, Chapter 4 examines model-theoretic questions concerning Sanchis' "hypergraph" model, a variant of $P\omega$.

Reviewer's remarks: This book is a well-written survey, although one might appreciate a clearer delineation of expository from original material. A major bibliographic omission is the work of J. Lambek, who pioneered the category-theoretic viewpoint of λ-calculus. For surveys of this work, see a paper by Lambek and one by D. S. Scott [Lambek, To H. B. Curry: essays on combinatory logic, lambda calculus and formalism, 375–402, Academic Press, London, 1980; MR 81m:03020; Scott, ibid., pp. 403–450; MR 82d:03025]. λ-monoids in the extensional case were first studied (under the name "C-monoids") by Lambek and the reviewer in 1982, following unpublished joint work of Lambek and D. S. Scott [see Lambek and the reviewer, Mathematical applications of category theory (Denver, Colo., 1983), 145–174, Amer. Math. Soc., Providence, R.I., 1984; MR 85i:03196; Introduction to higher-order categorical logic, Cambridge Univ. Press, Cambridge, 1986]. Further recent references are given in the revised edition of H. P. Barendregt's book The lambda calculus, its syntax and semantics [North-Holland, Amsterdam, 1984; MR 86a:03012]. Reviewed by P. J. Scott.

Mezghiche, M. (1984). Une nouvelle $C\beta$-réduction dans la logique combinatoire. *Theoret. Comput. Sci.*, 31(1-2):151–163.

[MR86j:03009] Combinatory logic and λ-calculus are closely parallel, but not exactly so. A combinatory convertibility relation corresponding exactly to $\lambda\beta$-

convertibility can be defined, but there are problems when we try to define a combinatory ($C\beta$) reduction corresponding to $\lambda\beta$-reduction.

If we demand the most natural exact correspondence, then the resulting $C\beta$-reduction does not have the highly desirable Church-Rosser property [the reviewer, Z. Math. Logik Grundlag. Math. 23 (1977), no. 2, 169–180, see p. 174; MR 58 #5080]. In fact there is no "perfect" $C\beta$-reduction. All we can do is list the conditions we feel such a reduction should satisfy (e.g., the Church-Rosser property, and a relaxed form of correspondence with $\lambda\beta$-reduction), and hope to find a reduction satisfying most of them.

The present paper gives a list of conditions due to the reviewer. (Several of these involve the abstraction operation $[x]$.) The author discusses some candidate reductions (and modifications of $[x]$) from the published and unpublished literature. None satisfies all the conditions.

He then presents an interesting new definition of $[x]$, and a corresponding reduction, and shows that it satisfies almost all the conditions. Given that perfection is impossible, the author's $C\beta$-reduction seems to the reviewer to be a very acceptable substitute.

However, in Definition 4.2 on p. 157, the clause "$U \neq [x]_\beta.Y$" should be omitted from (ξ'). With it included, the proofs of Theorem 4.9 and Property 4 are incomplete (and probably uncompletable), but when it is omitted, all the proofs are valid (as far as the reviewer can see).

There are a few trivial misprints: for example on p. 151 in (f), omit "["; on p. 159 in line 17, add a subscript "λ" to "Y". Reviewed by R. Hindley.

Shabunin, L. V. (1984). *Kombinatornye ischisleniya.* Chuvash. Gos. Univ., Cheboksary.

[MR87k:03012] The book is an introduction to the basic facts in λ-theory. The material covered by the book is standard and includes various types of λ-calculi, the Church-Rosser theorem, λ-definability of partial recursive functions, combinatory algebras, $P\omega$ and $D\infty$ models. The book is clearly written and is intended for university students. Reviewed by A. S. Kuzichev.

Venturini Zilli, M. (1984). Reduction graphs in the lambda calculus. *Theoret. Comput. Sci.*, 29(3):251–275.

[MR85k:68063] The reduction graph $G(T)$ of a λ-term T is the (possibly infinite) labelled directed graph in which the vertices are all the possible reducts of T and the arcs represent the reduction relation. The notion of reduction graph, although often mentioned in the literature, has never been the object of a careful study in itself.

In this paper, after some general remarks on the structure of reduction graphs (some well-known properties of β-reduction forbid certain configurations to occur) the author investigates some more specific properties. Firstly a characterization of the terms which have a linear reduction graph is given. It seems hard, in general, to prove similar results for other classes of reduction graphs. Then, through the notion of a plane in a reduction graph (a plane is a cyclic subgraph which corresponds to a maximal set of interconvertible terms) the notion of the condensed subgraph $G^\circ(T)$ (the graph of all planes of $G(T)$) is introduced. Two interesting related definitions are those of bottleneck (a cutpoint of $G^\circ(T)$) and recurrent term (an element of a terminal plane). Some general properties of bottlenecks and recurrent terms are proved.

Lastly the notion of the spectrum of a term T is introduced. The spectrum of T is the partial order of all possible reductions starting from T (modulo a suitable equivalence), which is proved to be isomorphic to the completion of $G^\circ(T)$ (intuitively, a reduction σ_1 is greater than σ_2 if it yields more information about the term; cofinal reductions are maximal). Spectra can have a very rich structure; for example, there exists a term such that $(P_\omega, \subseteq)$ can be embedded in its spectrum which, hence, contains a densely ordered segment.

The paper is rounded off by several examples and stimulating conjectures. Reviewed by M. Coppo.

Barendregt, H. (1985). *Lambda-ischislenie.* "Mir", Moscow. Ego sintaksis i semantika. [Its syntax and semantics], Translated from the English by G. E. Mints, Translation edited by A. S. Kuzichev, With a preface by Kuzichev and Mints.

[MR86g:03026] The English original was reviewed [North-Holland, Amsterdam, 1981; MR 83b:03016]. This translation incorporates additions and revisions by the author. A second English edition has appeared [1984; MR 86a:03012].

Bellot, P. (1985). A new proof for Craig's theorem. *J. Symbolic Logic*, 50(2):395–396.

[MR87a:03033] Craig's theorem states that every proper basis for combinatory logic contains at least two elements. This brief paper gives a very simple proof by contradiction. It is shown that for every singleton proper basis $\{Q\}$, Q must be a projection; and contrarily that for each projection Q, $\{Q\}$ cannot be a basis. Reviewed by John Staples.

de Vrijer, R. (1985). A direct proof of the finite developments theorem. *J. Symbolic Logic*, 50(2):339–343.

[MR87k:03013] From the introduction: "Let M be a term of the type-free λ-calculus and let $\mathcal{R}$ be a set of occurrences of redexes in M. A reduction sequence from M which contracts first a member of $\mathcal{R}$ and afterwards only residuals of $\mathcal{R}$ is called a development (of M with respect to $\mathcal{R}$). The finite developments theorem says that developments are always finite. There are several proofs of this theorem in the literature.

"In the present proof we straightforwardly define for each pair $(M, \mathcal{R})$ a natural number, which can easily be seen to indicate the exact number of reduction steps in a development of maximal length of M with respect to $\mathcal{R}$."

Jacopini, G. and Venturini Zilli, M. (1985). Easy terms in the lambda calculus. *Fund. Inform.*, 8(2):225–233.

[MR87b:03029] A closed λ-term M is said to be easy if for any closed λ-term N the equation $M = N$ is consistent with the extensional λ-calculus $\lambda\eta$. Some results on easiness are proved: (1) If M does not reduce a proper context of a reduct of itself, then M is easy. (2) If M is easy and F has a right inverse, then FM is easy. Result (1) gives easy terms of order 0, e.g., $(\lambda x.xxx)(\lambda x.xxx)P$ with $P \not\equiv \lambda x.xxx$. Result (2) gives easy terms of orders > 0, e.g., $\lambda x_1 \cdots x_n.\Omega x_n x_1 \cdots x_{n-1}$, with $\Omega \equiv (\lambda y.yy)(\lambda y.yy)$, since it is known that Ω is easy. Reviewed by Henk Barendregt.

Kanda, A. (1985). Numeration models of λ-calculus. *Z. Math. Logik Grundlag. Math.*, 31(3):209–220.

[MR87d:03033] Following Ershov, a numeration is defined to be an arbitrary infinite sequence of objects, in which repetitions are allowed. A morphism from a sequence $\langle s_i \rangle$ to a sequence $\langle t_i \rangle$ (reviewer's notation) is defined to be a function f such that, for some total recursive function r, $f(s_i) = t_{r(i)}$ (f is then said to be realized by r). Let $\langle \varphi_i \rangle$ be a standard numeration of the partial recursive functions. For each pair $\langle s_i \rangle$, $\langle t_i \rangle$ of sequences, a numeration $\langle f_j \rangle$ of the set of all morphisms from $\langle s_i \rangle$ to $\langle t_i \rangle$ is said to be admissible if there exist recursive total functions U, R, N such that (a) $f_n(s_m) = t_{U(n,m)}$, (b) if φ_n realizes some f_j, then φ_n realizes $f_{N(n)}$, (c) $\varphi_{R(n)}$ realizes f_n. (In fact, as the author has pointed out in correspondence, (c) is derivable from (a) and (b).)

A numerated reflexive set (NRS) is defined to be a numeration $\gamma = \langle s_i \rangle$ such that there is an enumeration $(\gamma \to \gamma)$ of the set of all morphisms from γ to γ which is both admissible and isomorphic to γ (understanding "isomorphic" as in the category of numerations). The author shows that every NRS forms an extensional λ-model (i.e. a model of the $\lambda\beta\eta$-calculus). Conversely, every extensional λ-model whose domain is enumerable and whose application and abstraction operations are recursive (relative to some fixed enumeration of the domain) is an NRS. He conjectures that a modified definition of NRS should give a corresponding result for the class of all λ-models (models of $\lambda\beta$).

Reviewer's remarks: In Section 5.5 of H. Barendregt's The lambda calculus [revised edition, North-Holland, Amsterdam, 1984; MR 86a:03012] it is shown that for any object D in a concrete Cartesian closed category: (i) if $(D \to D)$ is a retract of D, then D determines a λ-model, and (ii) if $(D \to D)$ is isomorphic to D, then this λ-model is extensional. Although the category of numerations is not Cartesian closed, the proof of (ii) is like the proof of the main lemma in the present paper, and the proof of (i) can be applied in the present context to show that if we replace "isomorphic to" in the definition of NRS by "a retract of", then every NRS will form a λ-model, thus proving the author's conjecture. Reviewed by R. Hindley.

Révész, G. (1985). Axioms for the theory of lambda-conversion. *SIAM J. Comput.*, 14(2):373–382.

[MR86i:03016] The paper gives two alternative versions of the lambda calculus in which the substitution operator is not needed for the description of the contraction rules. This may be useful for the construction of lambda calculus machines, since substitution is rather expensive to implement. The first system is as follows: $\lambda x.P \to \lambda y.(\lambda x.P)y$ for $y \notin \mathrm{FV}(P)$; $(\lambda x.x)Q \to Q$; $(\lambda x.y)Q \to y$ if $x \not\equiv y$; $(\lambda x.(\lambda x.P))Q \to \lambda x.P$; $(\lambda x.(\lambda y.P))Q \to \lambda y.(\lambda x.P)Q$ if $x \not\equiv y$ and ($x \notin \mathrm{FV}(P)$ or $y \notin \mathrm{FV}(Q)$); $(\lambda x.P_1P_2)Q \to (\lambda x.P_1)Q((\lambda x.P_2)Q)$. The Church-Rosser theorem holds for this system, which has the same power as the usual lambda calculus.

In the second alternative version of the lambda calculus the author uses a variable-renaming operator (for free and bound variables: $[z//x]x(\lambda x.x) \equiv z(\lambda z.z)$). The fifth rule in this system becomes $(\lambda x.(\lambda y.P))Q \to \lambda z.(\lambda x.[z//y]P)Q$ if $y \not\equiv x$, $x \in \mathrm{FV}(P)$, z is not bound in P and $z \notin \mathrm{FV}((P)Q) \cup \{x\}$. It is claimed that these side conditions are more efficient to verify than the ones in the first alternative system. The second alternative system is also of the same power as the usual lambda calculus. A normalizing reduction strategy (a variant of leftmost reduction using memory) is indicated for the second system. Reviewed by Henk Barendregt.

Bethke, I. (1986). How to construct extensional combinatory algebras. *Nederl. Akad. Wetensch. Indag. Math.*, 48(3):243–257.

[MR88d:03019] A combinatory algebra is a model of the weak equality in combinatory logic. The Plotkin-Engeler construction of a combinatory algebra D_A from an arbitrary nonempty set A always produces a nonextensional result.

The present paper modifies that construction so that the result, now called $M(A)$, is always extensional. The author proves that $\langle M(A), \cdot\rangle$ is isomorphic to a substructure of the algebra $\langle D_A, \cdot\rangle$, and shows that sets A with different cardinalities give nonisomorphic algebras $M(A)$. She also shows that the Scott-Plotkin model $P\omega$, which is not extensional, nevertheless contains an isomorphic copy of $\langle M(A), \cdot\rangle$ for a suitable set A.

In the end, $M(A)$ turns out to be isomorphic, as a combinatory algebra, to a Scott D_∞-model (obtained by starting with the lattice $\mathcal{P}(\mathcal{A})$). However, this has the effect of making the $M(A)$-construction an interesting Plotkin-Engeler-style analysis of a class of D_∞-models.

The author writes entirely in terms of models of weak equality and does not mention λ-models. Reviewed by R. Hindley.

Hatcher, W. S. and Scott, P. J. (1986). Lambda-algebras and C-monoids. *Z. Math. Logik Grundlag. Math.*, 32(5):415–430.

[MR88k:03028] A model of the (type-free) λ-calculus consists of a structure $(A,')$ where $'$ is a binary operation on A, such that λ-terms can be interpreted in A and convertible terms become equal. There are two kinds of these models.

In the first kind, called λ-algebras, it is not necessarily true for λ-terms $t = t(x)$ and $s = s(x)$ that $(*)$ $A \models t(x) = s(x) \Rightarrow A \models \lambda x.t(x) = \lambda x.s(x)$. This in spite of the fact that $t(x) = s(x) \Rightarrow \lambda x.t(x) = \lambda x.s(x)$ is one of the rules of the λ-calculus. The explanation of this phenomenon is that in a λ-algebra A the equation $\lambda x.t(x) = \lambda x.s(x)$ should hold only if $t(x) = s(x)$ holds by means of a conversion in the theory and not if $A \models t(x) = s(x)$ holds because of some accidental property of A.

In a λ-model, the second kind of model for the λ-calculus, the implication $(*)$ always holds. The paper gives an alternative description of λ-algebras based on the category-theoretic description of these structures [see C. P. J. Koymans, Information and Control 52 (1982), no. 3, 306–332; MR 85b:03020; Models of the lambda calculus, Math. Centrum, Centrum Wisk. Inform., Amsterdam, 1984; MR 87m:03017]. It runs as follows. Lambda-algebras satisfying the η-rule ($\lambda x.tx = t$, provided $x \notin \mathrm{FV}(t)$) and having a surjective pairing correspond exactly to so-called C-monoids, i.e. structures of the form $\langle A, \circ, P, *, I, \pi_1, \pi_2, \epsilon\rangle$ where P and $\circ$ are binary operations, $*$ is a unary operation and I, π_1, π_2 and ϵ are elements of A satisfying: (1) $I \circ a = a \circ I = a$; (2) $f \circ (g \circ h) = (f \circ g) \circ h$; (3) $\pi_i \circ \langle a_1, a_2\rangle = a_i$, $i = 1, 2$; (4) $\langle \pi_1 \circ a, \pi_2 \circ a\rangle = a$; (5) $\epsilon \circ \langle *(f) \circ \pi_1, \pi_2\rangle = f$; (6) $*(\epsilon \circ \langle f \circ \pi_1, \pi_2\rangle) = f$. A similar description has been given in the second cited publication of Koymans that is more extensive than the first one. Reviewed by Henk Barendregt.

Hindley, J. R. and Seldin, J. P. (1986). *Introduction to combinators and λ-calculus.* Cambridge University Press, Cambridge.

[MR88j:03009] In December 1920, the Soviet mathematician M. Schönfinkel opened, by his report "Uber die Bausteine der mathematischen Logik" to the Mathematical Society in Gottingen [Math. Ann. 92 (1924), 305–316; Jbuch 50, 23], a new trend in the foundations of mathematics, which is based on the concept of a generalized function, the argument of which is also a function (without any restrictions). This mathematical discipline, named "combinatory logic" by Curry and the "theory (calculus) of λ-conversion" or "λ-calculus" by A. Church, during the past fifteen years has changed its appearance. Its connections with algebra, topology and category theory were revealed and became an important part of informatics. In particular, λ-calculi have turned out today to be convenient models, revealing fundamental aspects of the construction and operation of programs in algorithmic languages.

While the classical methods of research were purely syntactical, the discovery of models enabled us to apply semantic methods, which have turned out to be very effective, for example, in problems of extension and application of classical λ-calculi.

The authors of the book under review are outstanding specialists in this domain; in particular, they wrote, together with H. B. Curry, the second volume of Combinatory logic [North-Holland, Amsterdam, 1972; Zbl 242:02029] and, together with B. Lercher, the book Introduction to combinatory logic [Cambridge Univ. Press, London, 1972; MR 49 #25].

The book under review is divided into eighteen chapters: (1) λ-calculus; (2) Combinatory logic; (3) The power of λ and combinators; (4) Representing the recursive functions; (5) The undecidability theorem; (6) The formal theories $\lambda\beta$ and CLw; (7) Extensionality in λ-calculus; (8) Extensionality in combinatory logic; (9) The correspondence between λ and CL; (10) Models of CLw; (11) Models of $\lambda\beta$; (12) D_∞ and other models; (13) Typed terms; (14) Type assignment to CL-terms; (15) Type assignment to λ-terms; (16) Generalized type assignment; (17) Logic based on combinators; (18) Gödel's consistency proof for arithmetic.

The titles of the chapters characterize the book, on the one hand, as an introduction to this subject and, on the other hand, as a complete exposition of almost all the new (especially, model-theoretic) results.

We may note in particular the logical constructions, introduced as type-free sequential extensions (with unrestricted abstraction (comprehension), but with restricted cut rule) of the theory of combinators of Schönfinkel and Curry or the λ-calculi of Church [see the reviewer, Z. Math. Logik Grundlag. Math. 29 (1983), no. 5, 385–416; MR 85d:03026; and the references cited therein].

These systems are worth noticing in such a book, since their use makes it possible to prove the consistency of the well-known calculi ZF, NBG and FA (without presuming the consistency of any other calculi, i.e. in the absolute sense) by the method of reduction of one logico-mathematical system to another, elaborated by Kolmogorov, Bernays and Gödel. The provable syntactical completeness (according to Gödel) of such logical constructions, and the provable consistency (in the classical sense) of some subclasses of the classes of all their derivations, are essentially used. Reviewed by A. S. Kuzichev.

Kanda, A. (1986a). Classes of numeration models of λ-calculus. *Z. Math. Logik Grundlag. Math.*, 32(4):315–322.

[MR87m:03016] Let $\gamma\colon \mathbf{N} \to X$ be a numeration such that for some numeration $\gamma\uparrow\colon \mathbf{N} \to \mathrm{Hom}(\gamma,\gamma)$, γ and $\gamma\uparrow$ are isomorphic. The numeration γ is called a numeration model of λ-calculus (in this article it is $\beta\eta$-calculus) if it is possible to define an interpretation function with obvious properties. γ is λ-definable if, roughly speaking, every morphism $\gamma \to \gamma$ can be defined by some closed λ-term. The so-called numerated reflexive sets (NRSs), introduced previously by the author [same journal 31 (1985), no. 3, 209–220; MR 87d:03033], are examples of numeration models. Here it is shown that λ-definable models constitute a subclass of NRSs. In the last section of the article, fixed-point theorems are investigated. Reviewed by A. S. Kuzichev.

Kanda, A. (1986b). Numeration models of $\lambda\beta$-calculus. *Z. Math. Logik Grundlag. Math.*, 32(5):409–414.

[MR88h:03017] The author considers λ-models that are at the same time numerated sets in the sense of Yu. L. Ershov [Theorie der Numerierungen, I, Deutscher Verlag Wissensch., Berlin, 1973; MR 55 #10257], requiring among other things that application is a morphism. The following examples of such models are given: (1) Open term models, (2) the recursively enumerable sets in the graph model $P\omega$ or its code-free variant D_A.

The technical details are somewhat unbalanced. For example, too much attention is given to coded syntactic properties of terms; on the other hand, the relevant fact that the r.e. elements of $P\omega$ satisfy the Meyer-Scott axiom for λ-models is not even mentioned. Reviewed by Henk Barendregt.

Statman, R. (1986). Every countable poset is embeddable in the poset of unsolvable terms. *Theoret. Comput. Sci.*, 48(1):95–100.

[MR88g:03025] The author introduces a quasi-order relation between combinators which takes into account how close combinators come to having head normal forms. This relation yields a quotient poset $\mathcal{U}$ which has a bottom element, consisting of all solvable terms, and some maximal elements. The main result of this paper is that every countable poset can be isomorphically embedded in $\mathcal{U}$. This result is purely syntactic. This paper is very interesting since it highlights the computational behaviour of unsolvable terms, a topic almost ignored in the current literature. Reviewed by Mariangiola Dezani.

Hirokawa, S. (1987a). On a problem on the correspondence between lambda calculus and combinatory logic. *Math. Japon.*, 32(6):937–945.

[MR89b:03028] The transformations λ, which transforms terms in combinatory logic into λ-terms, and H, which transforms λ-terms into terms in combinatory logic, are well known, as is the result: For any λ-term M, $M_{H\lambda} =_\beta M$.

The author shows that $M_{H\lambda} >_\beta M$, and that this result also holds for BCK-lambda terms (i.e. lambda terms in which no variable x (not part of a λx) appears more than once). There are several misprints in the paper. Reviewed by Martin W. Bunder.

Hirokawa, S. (1987b). Some properties of one-step recurrent terms in lambda calculus. *Math. Japon.*, 32(6):927–935.

[MR89b:03027] A recurrent term in lambda calculus is a term each of whose reductions is reducible to the term itself. A one-step recurrent term is a term whose one-step reductions are reducible to itself.

The author proves a number of properties of one-step recurrent terms, including the following: The leftmost reduction of a one-step recurrent term is one-step recurrent, and if a term is one-step recurrent then either it is a normal form or it has no normal form.

A natural question is "Is every one-step recurrent term recurrent?". The author shows that the affirmative answer is equivalent to a conjecture of Klop. Reviewed by Martin W. Bunder.

Honsell, F. and Ronchi Della Rocca, S. (1987). Topological models are not complete for the λ-calculus. In *Proceedings of the conferences on mathematical logic, Vol. 3 (Italian) (Siena, 1985; Padova, 1985; and Siena, 1986)*, pages 399–402, Siena. Univ. Siena.
[MR904918] This item will not be reviewed.

Krishnamurthy, E. V. and Vickers, B. P. (1987). Compact numeral representation with combinators. *J. Symbolic Logic*, 52(2):519–525.
[MR88c:03017] The natural numbers in combinatory logic have many representations but most are defined using a common generative principle: there is a combinator σ such that if $\mathbf{n}$ represents n, then $\sigma\mathbf{n}$ represents $n+1$. Thus, the number of terms in $\mathbf{n}$ grows linearly with n. In contrast, the usual binary (or decimal, etc.) place notation gives a representation that grows logarithmically with n. The authors achieve a similar result for a system of combinatory numerals. Reviewed by B. Lercher.

Mirasyedioğlu, Ş. (1987). λ-Boolean theory. *Internat. Logic Rev.*, (35):13–27.
[MR89a:03025] Summary: "This article introduces the basic ideas of λ-Boolean functions and their applications in the analysis of recursively defined λ-terms which represent Boolean functions."

Piperno, A. (1987). A compositive abstraction algorithm for combinatory logic. In *TAPSOFT '87, Vol. 2 (Pisa, 1987)*, pages 39–51. Springer, Berlin.
[MR89g:03018] This paper presents an interesting algorithm for translating lambda-terms into combinators. This algorithm can be used for program compilation in the implementation of functional programming languages.

The author carefully sets his paper against the current literature. The main features of the proposed algorithm are efficiency (the complexity is a function of the number of variables occurring in the term to be translated) and compositionality (the resulting code may be viewed as a succession of procedure calls). The basis of combinators used is infinite but directly implementable, since each basic combinator depends on at most two parameters. Morever, the author presents two versions of this algorithm, one which repeatedly operates on one variable at a time and one which simultaneously operates on all variables. It is interesting to note that the length of the resulting code is invariant with respect to this choice (this is not true for the other algorithms in the literature). This paper contains valuable results that make it worth reading. For the entire collection see MR 88e:68011. Reviewed by Mariangiola Dezani-Ciancaglini.

Legrand, R. (1988). A basis result in combinatory logic. *J. Symbolic Logic*, 53(4):1224–1226.
[MR89i:03029] The author proves in this note that any basis for combinatory logic

must contain at least one combinator of rank strictly greater than two. Reviewed by Martin W. Bunder

Mirasyedioğlu, Ş. (1988). A reduction system with the lambda operator and combinators. *J. Karadeniz Tech. Univ. Fac. Arts Sci. Ser. Math.-Phys.*, 11:199–210 (1989). National Mathematics Symposium (Trabzon, 1987).
[**MR1103176**] This item will not be reviewed.

Sekimoto, S. and Hirokawa, S. (1988). One-step recurrent terms in λ-β-calculus. *Theoret. Comput. Sci.*, 56(2):223–231.
[**MR89h:68035**] In the λ-calculus, a term M is said to be (one-step) recurrent if and only if every term, such that M β-reduces (in one step) to it, β-reduces in its turn to M. The notion of recurrent term was studied by Böhm and Micali (who called it minimal form) and by Venturini-Zilli; they pointed out that, for a term, the property of being recurrent can be viewed as a generalization of the property of having a normal form. One of the authors, in a preceding paper, introduced the weakened notion of a one-step recurrent term, and gave, using this notion, a new and clearer formulation of a conjecture of Klop, which is the following: "Every one-step recurrent term is recurrent". In this note the authors (not surprisingly) disprove Klop's conjecture, by showing a counterexample, and give the necessary and sufficient conditions for a one-step recurrent term to be recurrent.

Both the problem and the proofs are very technical, but the note is clearly written and thus easy to understand. Reviewed by Simona Ronchi Della Rocca.

Stoughton, A. (1988). Substitution revisited. *Theoret. Comput. Sci.*, 59(3):317–325.
[**MR89j:68104**] Summary: "A definition of simultaneous substitution for the lambda calculus is presented that is easier to work with than standard single substitution because it is a definition by structural recursion, instead of recursion on the length of terms, and bound variables are always renamed. As a result, many proofs involving substitution can be given by structural induction, instead of induction on the length of terms, and are simpler than the corresponding standard proofs because of the reduction in the number of cases that must be considered. Furthermore, because of the uniform renaming of bound variables, identity substitutions normalize terms with respect to equivalence up to the renaming of bound variables (α-congruence), allowing induction-free proofs of some theorems that ordinarily would be proved by induction on the relation of α-congruence.

"A series of results relating simultaneous substitution and α-congruence are proved, and a simple proof of the 'substitution lemma' of denotational semantics is given."

Böhm, C. (1989). Subduing self-application. In *Automata, languages and programming (Stresa, 1989)*, pages 108–122. Springer, Berlin.
[**MR91a:03025**] Summary: "Recursive equational schemes, defining total functions on natural numbers, are embedded into a combinatory algebra producing an equation system having the form $f(0) = b, f(s(x)) = H(a, f, x)$. Two methods are described to derive the combinator representing f by means of a generalized morphism, avoiding the use of fixed point combinators and preserving strong normalizability, i.e. the same feature warranted by most type disciplines.

"Can a given combinator s represent a successor of some adequate algebraic numeral system? Answers to this question are exemplified and assembled to solve the problem

of embedding the infinite cyclic group **Z** of integers into a combinatory algebra." For the entire collection see MR 90m:68001.

Böhm, C. and Dezani-Ciancaglini, M. (1989). Combinatory logic as monoids. *Fund. Inform.*, 12(4):525–539.

[**MR90m:03028**] The algebraic structures discussed in this interesting paper are related to the set MS of ordered pairs (A, Id), where A is an associative combinator and Id an identity combinator under A. The authors discuss relationships among MS, a subset MS$'$ of MS which has its A's restricted, certain ordered triples, quadruples and sextuples consisting of A, Id and further special combinators, and combinator groups with operations and identity based on A and Id. Reviewed by Martin W. Bunder.

Böhm, C., Piperno, A., and Tronci, E. (1989). Solving equations in lambda-calculus. In *Logic Colloquium '88 (Padova, 1988)*, pages 139–160. North-Holland, Amsterdam.

[**MR90h:03013**] Part 1 of this paper reviews results on solvable and unsolvable equations in λ-calculus and combinatory logic, including the extensions of these systems that include the constant Ω with the properties $\Omega T =_{\beta\Omega} \Omega$ and $\lambda\nu.\Omega =_{\beta\Omega} \Omega$. Particular emphasis is given to the solvability of discriminability, separability and X-separability. All these notions are special cases of "X-weak separability".

F is said to be X-weakly separable if $F = \{F_1, \cdots, F_t\}$ is a multiset of combinators in the β-Ω logic, each having the same abstracted variables $X = \{x_1, \cdots, x_n\}$, and if for $Y = \{y_1, \cdots, y_t\}$ a multiset of variables the equations $F_i x_1 \cdots x_n =_{\beta\Omega} y_i$ $(i = 1, \cdots, t)$ are solvable for $x_1, \cdots, x_n$.

The decidability of "F is X-weakly separable" is proved in Part 2 for sensible theories and F λ-free. For the entire collection see MR 90d:03005. Reviewed by Martin W. Bunder.

Bunder, M. W., Hindley, J. R., and Seldin, J. P. (1989). On adding (ξ) to weak equality in combinatory logic. *J. Symbolic Logic*, 54(2):590–607.

[**MR90d:03028**] This is a very interesting paper which solves many open problems concerning the role played by rule (ξ) in the relations between combinatory weak equality and β-conversion.

It is well known that the main difference between combinatory weak equality and β-conversion is the rule (ξ) $X = Y \vdash \lambda x.X = \lambda x.Y$. Therefore it is natural to investigate the equality (ξ-equality) generated by adding (ξ) to the postulates for weak equality. But in combinatory logic the λ-abstraction that occurs in (ξ) is not primitive, and can be defined in many different ways. Each definition produces a different rule (ξ). The authors consider the ξ-equalities for six different definitions of abstraction, study their relations, distinguish between them, and axiomatize them. It turns out that if one uses the more common abstraction algorithms, the result will be a ξ-equality that is either equivalent to $\beta\eta$-conversion (and so strictly stronger than β-conversion) or else strictly weaker than β-conversion. However, two abstraction algorithms are given which succeed in the quest for a ξ-equality equivalent to β-conversion.

The style of the paper is very clear and smooth; it is a pleasure to read it. Reviewed by Mariangiola Dezani-Ciancaglini.

Jiang, M. D. and Wen, H. (1989a). Combinatory logic as a first-order mathematical theory. *Kexue Tongbao (Chinese)*, 34(12):950–953.

[**MR1030698**] There will be no review of this item.

Jiang, M. D. and Wen, H. (1989b). Combinatory logic as the [a] first-order mathematical theory. *Chinese Sci. Bull.*, 34(24):2079–2083.

[MR95b:03012a] The paper presents combinatory logic as a first-order theory, defines the notion of a model of that first-order theory, states as a lemma (without proof) that the first-order theory has a model, and from that concludes that the first-order theory is consistent, complete, and equiefficacious to the original combinatory logic.

Reviewer's remarks: (1) Jiang and Wen seem unaware that combinatory logic has been presented as a first-order logic before (see papers by H. P. Barendregt [Indag. Math. 35 (1973), 203–221; MR 58 #163] and M. Bel [Indag. Math. 39 (1977), no. 2, 69–72; MR 57 #75], both of which are listed in another work by Barendregt [The lambda calculus, revised edition, North-Holland, Amsterdam, 1984; MR 86a:03012]). (2) In the reviewer's opinion, the first paper would have been improved by a presentation of the proof of the lemma that the first-order theory of combinatory logic has a model; it would have been nice to know what model the authors had in mind. Reviewed by J. P. Seldin.

Mezghiche, M. (1989). On pseudo-cβ normal form in combinatory logic. *Theoret. Comput. Sci.*, 66(3):323–331.

[MR91e:03018] Summary: "A cβ-reduction in combinatory logic analogous to $\lambda\beta$-reduction in λ-calculus was defined by us in 1984. In this paper we define a notion of pseudo-cβ normal form and prove that for a CL-term X in pseudo-cβ normal form there exists a λ-term in $\lambda\beta$ normal form such that $X \equiv M_{H\beta}$."

Statman, R. (1989). On sets of solutions to combinator equations. *Theoret. Comput. Sci.*, 66(1):99–104.

[MR90j:03029] This paper shows, using a construction of Plotkin, the remarkable result that for every r.e. set $\mathcal{S}$ of combinators that is closed under β-conversion there is a common fixed point R, i.e., any such $\mathcal{S}$ is the set of solutions to an equation $Rx = R$. Reviewed by Martin W. Bunder.

Takahashi, M. (1989). Parallel reductions in λ-calculus. *J. Symbolic Comput.*, 7(2):113–123.

[MR90b:03017] Parallel reduction is a formulation of the reduction notion for the Tait-Martin-Löf proof of the Church-Rosser theorem. The author expresses the Church-Rosser proof using parallel reductions and then, again using parallel reductions, obtains a direct proof of the normal reduction theorem. If M has a normal form N, then N can be obtained from M by a leftmost reduction. Reviewed by B. Lercher.

Bunder, M. W. (1990). Some improvements to Turner's algorithm for bracket abstraction. *J. Symbolic Logic*, 55(2):656–669.

[MR91g:03033]

This paper presents modifications that can be made to Turner's (and other) algorithms for converting terms of the lambda calculus into combinatory terms (terms which do not include lambda abstractions). These algorithms involve transforming lambda abstractions (terms of the form $\lambda x.E$) into bracket abstractions (of the form

$[x]E$) and then providing an algorithm which allows one to expand the bracket abstraction as an abbreviation for a pure combinator term. The author notes first of all that Turner's algorithm does not preserve the property that $([x]X)x \to x$ (where $\to$ is the left-to-right application of combinatory reduction rules for the **S**, **K** and **I** combinators). Terms obtained by Turner's algorithm are, of course, $\beta\eta$-equivalent to the original.

The author extends Turner's algorithm, which involves combinators **S**, **K**, **I**, **B**, **C**, $\mathbf{B}'$, and $\mathbf{C}'$, 10 reduction rules, and 6 bracket abstraction rules, by an algorithm which uses the same combinators, 34 reduction rules, and 9 bracket abstraction rules. In this algorithm, the reduction rules are applied before the bracket abstraction rules. The author gives an example which shows that this algorithm does not always give as short an abstraction as Turner's rules, but notes that further optimizations (reduction rules) can be added to shorten any individual such counterexample. The main theorem of the paper shows that this extended algorithm always produces a unique fully reduced term, where "fully reduced" means that none of the reduction rules in the paper apply. The author also shows how similar improvements can be made to other bracket abstraction algorithms.

It is claimed that the new algorithm is more efficient than that of Turner, but no indication, aside from two simple examples, is given as to how much of an improvement in speed is made in general. As noted earlier, the new algorithm does not always return expressions as short as those given by the original algorithm. Turner's algorithm is known to have $O(n^2)$ space complexity in the worst case. Other algorithms are known to be $O(n \log n)$ (in both space and time). An analysis of complexity would seem essential to making the case for the value of this extended algorithm.

There appears to be a minor error on line 12 of page 667 of the paper. The Turner algorithm would return $\mathbf{B}x_1(\mathbf{B}x_2x_3)$, exactly the same as the extended algorithm (though Turner's algorithm does require more steps). Reviewed by Kim Bruce.

Ginisti, J.-P. (1990). La formation des notions en logique combinatoire. *Acta Univ. Lodz. Folia Philos.*, (7):33–47. La formation des notions de base en logique classique et non classique (Łódź, 1988).

[MR91k:03032] The author outlines Curry's basic functionality theory for combinatory logic and describes a series of complete combinatory bases obtained by modifying S and K in a simple way. He describes the Curry-Howard correspondence between types and provable formulae of intuitionist implicational logic, and makes the general point that the basic notions of one theory (for example, implicational logic) can sometimes be derived as nonbasic notions in another theory whose basis is at first sight completely different. For the entire collection see MR 91h:03006. Reviewed by R. Hindley.

Ke, T. (1990). Discussions about the report: "Combinatory logic as the [a] first-order mathematical theory" [Chinese Sci. Bull. **34** (1989), no. 24, 2079–2083; MR 95b:03012a] by M. D. Jiang and H. Wen. *Chinese Sci. Bull.*, 35(23):2012–2013.

[MR95b:03012b] This paper is a summary by an editor of correspondence about the report involving some objections. The most important objection is based on a misprint in that paper which is repeated in the attempt to correct it in this one. The misprint involves the report's axiom $\mathcal{A}_8$, which should read $(\forall x_1)(\forall x_2)(\forall x_3)(\mathrm{App}(x_1, x_2, x_3) \Leftrightarrow \mathcal{I}(x_1 x_2, x_3))$. Reviewed by J. P. Seldin.

Komori, Y. (1990). λ-calculus with simultaneous substitution as a primitive symbol. In *Proceedings of the 13th Symposium on Semigroups (Kyoto, 1989)*, pages 20–26, Kyoto. Kyoto Sangyo Univ.

[MR91m:03007] In the classical λ-calculus the operation of substitution is defined by means of the metalanguage. This is a very complicated notion from the formal point of view because of the employment of bound and free variables and the use of α-conversion (see Appendix C of H. P. Barendregt's book The lambda calculus [North-Holland, Amsterdam, 1984; MR 86a:03012], particularly the comparison of the theoretical method of Curry and the numerical method of de Bruijn for computer implementation). The author introduces the substitution symbol su in the language of λ-calculus as a primitive symbol with the intended interpretation of $(\mathrm{su}M(\mathrm{co}x_1(\mathrm{co}N_1(\cdots(\mathrm{co}x_n(\mathrm{co}N_n 0))\cdots))))$ as $[N_1/x_1, \cdots, N_n/x_n]M$ in the metalanguage. He defines three reduction rules for su itself and three mixed rules for the connections between su, the application operator ap and the λ-operator λ. For double substitution he needs the additional operator wa with three reduction rules. These nine rules define the author's βs-reduction.

However, the Church-Rosser theorem does not hold for βs, as was established by the author himself. Hence he restricts the set Λ of his λ-terms (essentially wider than the usual one) to the set Λe of expected λ-terms proposed by him. The author claims the Church-Rosser theorem for βs reduction restricted to Λe, using his notion of coincident reduction. But some of his expected λ-terms, such as $(\lambda(\lambda xx)x)$ and $(\mathrm{su}x(\mathrm{co}(\lambda xx)y))$, correspond to no conventional (i.e. meaningful) λ-terms. Moreover $(\lambda xx) \neq (\lambda yy)$ in βs-conversion since α-conversion is invalid in this system, and the βs-normal form is rather different from the β-normal form.

It seems that the artificial restrictions on λ-terms in the paper may be avoided when only meaningful λ-terms are used in the reduction rules. The author does not go into details in proving his theorems. However, the ideas and results of this paper are very interesting (especially in practice). The author has made a very fast computer interpreter of λ-calculus using the theory developed in this paper. For the entire collection see MR 90k:20003. Reviewed by A. S. Kuzichev and V. P. Shumikhin.

Parigot, M. (1990). Internal labellings in lambda-calculus. In *Mathematical foundations of computer science (Banská Bystrica, 1990)*, pages 439–445. Springer, Berlin.

[MR91m:03008] From the introduction: "We show that the normalisation for the apparent redexes and for (some) labelled lambda-calculi reduces in a quite straightforward way to the normalisation theorem for terms typable using $\to$ and $\wedge$. The method we use is a syntactic translation of the lambda-calculus into itself. We replace the usual 'external labellings' obtained by ad hoc extensions of the syntax of the lambda-calculus by 'internal labellings' obtained by expanding the terms with definable combinators (internal labels) in such a way that (1) the expanded term is always typable using $\to$ and $\wedge$ and (2) the reduction sequences we are interested in are faithfully reflected in the expanded term." For the entire collection see MR 91h:68011. Reviewed by Dorothy Bollman.

Barendregt, H. (1991). Self-interpretation in lambda calculus. *J. Funct. Programming*, 1(2):229–233.

[**MR1110298**] There will be no review of this item.

Barendsen, E. (1991). Theoretical pearls: an unsolvable numeral system in lambda calculus. *J. Funct. Programming*, 1(3):367–372.

[**MR93g:03012**] For numeral systems in untyped λ-calculus the definability of a successor, a predecessor, and a test for zero implies the definability of all recursive functions in that system. Towards a disproof of the converse statement, H. P. Barendregt and the author of the paper under review constructed a numeral system consisting of unsolvable λ-terms in which all unary partial recursive functions are representable. Such a system cannot have a test for zero if the truth values are chosen, as usual, to be two distinct normal forms. Then, independently, B. Intrigila ["Some results on numerical systems in λ-calculus", Notre Dame J. Formal Logic, to appear] found a numeral system consisting of unsolvable λ-terms in which all partial recursive functions, unary or not, are representable. The paper under review presents an example of such a system, combining the previous constructions, and gives all the necessary background. Reviewed by Alessandro Berarducci.

Böhm, C. and Tronci, E. (1991). About systems of equations, X-separability, and left-invertibility in the λ-calculus. *Inform. and Comput.*, 90(1):1–32.

[**MR91m:03031**] In the λ-calculus, many problems can be presented as a system of equations. To search for solutions of these systems, one has to treat the difficulty of self-application. The authors, in this paper, show how a disciplined use of self-application can be employed in the solution of functional equations without degenerating into infinite computations. They use an equivalence relation, which models the indistinctness of pairs of λ-terms in a finite set, to obtain a necessary condition for solvability of finite systems of equations. They present a class of systems for which this necessary condition is also sufficient for solvability. The authors point out that the next step seems to be to discover a larger class of such systems. These results are applied to the X-separability problem. Finally, the authors transform a left-invertibility problem into an X-separability problem. Reviewed by Fangmin Song.

Curien, P.-L. and Di Cosmo, R. (1991). A confluent reduction for the λ-calculus with surjective pairing and terminal object. In *Automata, languages and programming (Madrid, 1991)*, pages 291–302. Springer, Berlin.

[**MR1129914**] This item will not be reviewed individually. For details of the collection in which this item appears see 92g:68004 .

Galbiati, L. and Talcott, C. (1991). A simplifier for untyped lambda expressions. In *Conditional and typed rewriting systems (Montreal, PQ, 1990)*, pages 342–353. Springer, Berlin.

[MR1138502] This item will not be reviewed individually. For details of the collection in which this item appears see 92g:68076 .

Intrigila, B. (1991). A problem on easy terms in λ-calculus. *Fund. Inform.*, 15(1):99–106.

[MR92m:03018] Summary: "We say that a closed term M is easy in the $\lambda\beta\eta$-calculus if for every term N the theory $\lambda\beta\eta + \{M = N\}$ is consistent. The main result of the paper is that, for a certain term P, the term $P(\omega\omega)$ is easy, while $P\omega$ is not. It follows that the easiness of a given M does not imply either easiness or noneasiness of PM, contradicting an earlier conjecture."

Lu, R. Z., Zhang, Z., and Sun, Y. Q. (1991). Construction of the model of the lambda calculus system with algebraic operators. *J. Comput. Sci. Tech. (English Ed.)*, 6(1):108–112.

[MR92g:03021] The paper presents a family of lambda-calculi with pattern-matching abstraction. Patterns are defined by means of operators from the signature of an equationally defined algebraic structure, enjoying a property which makes them behave as constructors (although the definition of such structure operators is quite different from the usual notion of constructor in term-rewriting systems). A model definition for calculi in this family is given, and a specific model is obtained by an extension of the standard D_∞ construction familiar from pure λ-calculus, in which, at each step, a model of the first-order equations defining the algebraic structure is produced.

The conciseness of the paper, together with some inaccuracies in the formal development (e.g., in the treatment of the substitution operation) makes a full appreciation of the results difficult. Reviewed by Felice Cardone.

McCune, W. and Wos, L. (1991). The absence and the presence of fixed point combinators. *Theoret. Comput. Sci.*, 87(1):221–228.

[MR93a:03014] The authors study the fixed point property ($\forall F\, \exists T \;\; T = FT$) and the presence of fixed point combinators ($\exists\Theta\, \forall F \;\; \Theta F = F(\Theta F)$) in several systems of combinatory logic.

The research was sparked off by R. M. Smullyan's question whether the CL-system given by $\{Bxyz = x(yz),\;\; Mx = xx\}$ has a fixed point combinator. This question remains open, but some specific negative results are obtained by classifying various forms of combinator rules $A\overrightarrow{x} = T(\overrightarrow{x})$.

The results were inspired by an analysis of experiments with the automated theorem prover OTTER using the so-called kernel strategy to search for special fixed point combinators Θ having the property $\exists T \;\; \Theta f \twoheadrightarrow T \twoheadrightarrow fT$. Reviewed by Erik Barendsen.

Barendregt, H. (1992a). Representing "undefined" in lambda calculus. *J. Funct. Programming*, 2(3):367–374.

[MR94a:03017] This paper presents a general theorem of R. Statman about λ-definability of the partial recursive functions. Let F be a λ-term representing a partial recursive function ϕ in the sense that $\phi(n) = m$ if and only if $F\ulcorner n\urcorner =_\beta \ulcorner m\urcorner$. There are several proposals for what $F\ulcorner n\urcorner$ should be when $\phi(n)$ is undefined: a term without normal form (Church, 1941), an unsolvable term (Barendregt, 1971), an easy term (A. Visser, 1980) or a term of order 0 (Statman, 1987). It is proved that any nontrivial set of closed lambda terms which is co-r.e. and closed under $=_\beta$ can be used to represent 'undefined'. This covers the four possibilities above. The argument is based on a theorem of Visser about the precomplete numerations of Yu. L. Ershov [A. Visser, in To H. B. Curry: essays on combinatory logic, lambda-calculus and formalism, 259–284, Academic Press, London, 1980; MR 83c:03041]. Reviewed by Alessandro Berarducci.

Barendregt, H. (1992b). Theoretical pearls: enumerators of lambda terms are reducing. *J. Funct. Programming*, 2(2):233–236.

[**MR94e:03010**] A closed λ-term E is called an enumerator if

$$(\forall M \in \Lambda^0)(\exists n \in N) E\ulcorner n \urcorner =_\beta M.$$

Here Λ^0 is the set of closed λ-terms, $\ulcorner n \urcorner$ is Church's numeral encoding the natural number n, and $=_\beta$ denotes β-conversion in the λ-calculus. Such an enumerator E is said to be reducing if, moreover, for β-reduction $\twoheadrightarrow_\beta$, it satisfies

$$(\forall M \in \Lambda^0)(\exists n \in N)\ E\ulcorner n \urcorner \twoheadrightarrow_\beta M.$$

Any self-interpreter $E \in \Lambda^0$ that satisfies

$$(\forall M \in \Lambda^0)\ E\ulcorner M \urcorner =_\beta M$$

is obviously an enumerator. Analysis of the construction of the self-interpreters due to S. C. Kleene and P. de Bruin shows that they are reducing enumerators.

Another example of an enumerator is a universal generator, introduced in 1971 by the author in his Ph.D. thesis. In the paper under review the author describes its recursive construction. He also gives here the concrete form of another such λ-term proposed by Mulder in 1990. Both of these universal generators are also reducing enumerators. All these facts led the author to the following conjecture: Every enumerator is reducing. The author intuitively explained this conjecture to R. Statman in 1983, and he proved it rigorously in 1987 (in fact, much more generally).

Theorem (Statman): Let $\mathcal{A} \subseteq \Lambda^0$ be a recursively enumerable set (after coding). Suppose $(\forall M \in \Lambda^0)(\exists N \in \mathcal{A})\ N =_\beta M$. Then $(\forall M \in \Lambda^0)(\exists N \in \mathcal{A}) N \twoheadrightarrow_\beta M$. A complete proof of this theorem is given in the paper under review. It uses interesting details of λ-definability of recursive functions and is based on λ-calculus techniques. From the theorem the conjecture follows by taking $\mathcal{A} = \{E\ulcorner n \urcorner\colon\ n \in N\}$.

On the other hand one cannot conclude that every self-interpreter is reducing in the sense that $(\forall M \in \Lambda^0)\ E\ulcorner M \urcorner \twoheadrightarrow_\beta M$. This fact was pointed out to the author by de Bruin. Can someone give another counterexample? Reviewed by A. S. Kuzichev and V. P. Shumikhin.

Bunder, M. W. (1992). On adding further forms of (ξ) to weak equality in combinatory logic. *Austral. Comput. Sci. Comm.*, 14(1):125–149. Fifteenth Australian Computer Science Conference (ACSC-15) (Hobart, 1992).

[**MR94a:03018**] Summary: "It might be thought that if the rule

$$X =_{\xi^*} Y \to \lambda^* x.X =_{\xi^*} \lambda^* x.Y\ (\xi^*)$$

is used to strengthen weak combinatory equality, the new equality $=_{\xi^*}$ would be equivalent to $\lambda\beta$ equality. This is not the case. This paper examines the distinct equalities $=_{\xi^*}$ obtained when different forms of bracket abstraction $\lambda^* x$ are used. Also, a set of combinatory equations equivalent to each such rule (ξ^*) (and also (η^*)) is given." For the entire collection see MR 93i:68005.

Shen, B. Y. (1992). The abstraction operation in combinatory logic. *Chinese Ann. Math. Ser. A*, 13(4):451–457.

[**MR94a:03023**] Summary (translated from the Chinese): "In order to study the equivalence of combinatory logic (CL) and λ-calculus, it is very natural to define the abstraction operation $\lambda^* x$ in CL and to introduce the rule ξ_C: $P = Q \implies \lambda^* x.P = \lambda^* x.Q$. The theory thus obtained is denoted by CLξ_C. The study of the equivalence of λ-calculus and CLξ_C is commonly carried out using a special abstraction operation. We obtain general necessary and sufficient conditions on the abstraction operation $\lambda^* x$ for the λ-calculus to be equivalent to CLξ_C. Theorem 1 describes the necessary and sufficient conditions that the operation $\lambda^* x$ must satisfy in order for the general λ-calculus to be equivalent to CLξ_C. Theorem 2 describes the necessary and sufficient conditions that the operation $\lambda^* x$ must satisfy in order for the extended λ-calculus to be equivalent to CLξ_C."

Statman, R. (1992). Normal varieties of combinators. In *Logic from computer science (Berkeley, CA, 1989)*, pages 585–596. Springer, New York.

[**MR94g:03030**] Given combinators (= closed λ-terms) F, G, the variety defined by $Fx = Gx$ is the set of all combinators M such that $FM =_\beta GM$; the variety is normal if F and G are normal. A previous result of Klop states that a set Σ of combinators is a variety iff it is Visseral, which means recursively enumerable and closed under β-equivalence; but in general it is false that Σ is a normal variety (the simplest counterexample is $\{\Omega\}$).

In this elegant and concise paper the author first discusses normal varieties of solvable combinators. Binary languages may be viewed as Visseral sets of solvable combinators, and those which are normal varieties are characterized. From this characterization follows the fact that the collection of normal varieties is Σ^3_0-complete. Next he proves a number of results analogous to basic theorems of classical algebra, such as Hilbert's Nullstellensatz.

The paper concludes with several applications, one being the undecidability of whether $Fx = I$ admits a (normal) solution for normal combinators F. For the entire collection see MR 94c:03004. Reviewed by Chantal Berline.

Barendregt, H. (1993). Constructive proofs of the range property in lambda calculus. *Theoret. Comput. Sci.*, 121(1-2):59–69. A collection of contributions in honour of Corrado Böhm on the occasion of his 70th birthday.

[**MR94m:03027**] The first part of this short paper surveys the different proofs of the range property for the closed term model of lambda-calculus; then the author gives two constructive proofs of the property (namely, that the range of any combinator is either a singleton or an infinite set). The first one is direct and very natural, up to a coding of lambda-calculus within itself, while the second one, based on Ershov's enumerations, works in a more general context. The paper ends with the open question of whether the range property holds for the closed term model modulo the

theory that identifies all unsolvable terms. For the entire collection see MR 94e:68005. Reviewed by Chantal Berline.

Berarducci, A. and Böhm, C. (1993). A self-interpreter of lambda calculus having a normal form. In *Computer science logic (San Miniato, 1992)*, pages 85–99. Springer, Berlin.

[MR94i:03041] Summary: "We formalize a technique introduced by Böhm and A. Piperno to solve systems of recursive equations in lambda calculus without the use of the fixed-point combinator and using only normal forms. To this end we introduce the notion of a canonical algebraic term-rewriting system, and we show that any such system can be interpreted in the lambda calculus by the Böhm-Piperno technique in such a way that strong normalization is preserved. This allows us to improve some recent results of T. Mogensen concerning efficient Gödelizations $\ulcorner\urcorner\colon \Lambda \to \Lambda$ of lambda calculus. In particular we prove that under a suitable Gödelization there exist two lambda terms E (self-interpreter) and R (reductor), both having a normal form, such that $E\ulcorner M\urcorner \to M$ for every (closed or open) lambda term M, and if M has a normal form N then $\mathrm{R}\ulcorner M\urcorner \to \ulcorner N\urcorner$." For the entire collection see MR 94f:68011.

Berarducci, A. and Intrigila, B. (1993). Some new results on easy lambda-terms. *Theoret. Comput. Sci.*, 121(1-2):71–88. A collection of contributions in honour of Corrado Böhm on the occasion of his 70th birthday.

[MR95a:03024] The research in the paper continues work of Böhm and Jacopini on the untyped λ-calculus. Specifically, consistency of adding equations $X = Y$ between λ-terms X, Y to the calculus (with β-conversion) is studied. A term X is called [normal form] easy if for each [normal form] Y, the equation $X = Y$ can be added consistently. Obviously, every easy term is normal form easy, but the converse does not hold, as is shown. The authors prove a "separation theorem" which allows them to establish some pathological results: there are closed λ-terms X, Y for which $X = Z$ is consistent for each closed term Z except $\lambda x.x$, and $Y = N$ is consistent for every closed normal form N except Curry's fixed point combinator Y. For the entire collection see MR 94e:68005. Reviewed by Bart Jacobs.

Georgieva, N. (1993). Spaces with combinators. *Arch. Math. Logic*, 32(5):321–339.

[MR95a:03025] Spaces with combinators are similar to what are called combinatory algebras in a book by H. P. Barendregt [The lambda calculus, revised edition, North-Holland, Amsterdam, 1984; MR 86a:03012]; they are models of systems of combinators with weak equality (conversion). They are algebraic structures with a binary operation (application) and four basic combinators: K, B, $\mathsf{C}_* \equiv \mathsf{CI}$, and W (which are called A, B, C, and D by the author). This paper seems to be something of a survey paper. The author begins with a definition of spaces with combinators, gives two examples (one of which is similar to the construction of models of the λ-calculus by G. D. Plotkin ["A set theoretical definition of application", Tech. Rep. MIP-R-95, Univ. Edinburgh, Edinburgh, 1972; per bibl.] and E. Engeler [Algebra Universalis 13 (1981), no. 3, 389–392; MR 83f:03012]), and then proves a number of results about spaces with combinators, including a canonical form theorem, a normal form theorem (which seems to be the result that says that abstraction can always be defined), the fixed point theorem, and the result that numbers can be represented.

The representation of natural numbers was new to the reviewer; the natural number n is represented by a term that would, in λ-calculus, be written

$$\lambda x.x(\lambda y_1.y_1(\lambda y_2.y_2(\cdots(\lambda y_n.y_n \mathsf{IK})\cdots)\mathsf{K})\mathsf{K}).$$

A test for zero (called by the author a "while" operator) is also shown to be definable. An iterative space with combinators is then defined by replacing the basic combinator W with an iteration operator (which can be defined in terms of W), and it is proved that every partial recursive function can be represented. As applications of the theory, the author concludes with abstract versions of some basic results in recursion theory.

Reviewer's remarks: (1) The reviewer finds the statements and proofs of the canonical form theorem and normal form theorem a little confusing. The statement of the first theorem is not clear without a look at the proof of the second, and the proof of the second is not totally convincing, since it looks a little like a reference to the procedure described in the second and third last paragraphs of the English translation of a paper by M. Schönfinkel, "On the building blocks of mathematical logic", in the book From Frege to Gödel, edited by J. van Heijenoort [pp. 355–366, Harvard Univ. Press, Cambridge, MA, 1967], and that description proved to be insufficient to establish the definability of the abstraction operator. In view of the well-known definitions of abstraction in terms of combinators, it would appear easier to prove that S can be defined in terms of the basic combinators given by the author. This can be done easily: since the author's C_2 is the usual combinator C and is shown to be definable, the definition of S in terms of B, C, and W from the book by H. B. Curry and R. Feys [Combinatory logic, Vol. I, see p. 155, North-Holland, Amsterdam, 1958; MR 20 #817], which in the author's terms is $B(B(BD)C_2)(BB)$, is sufficient. (2) Once it is shown that abstraction is definable, all the remaining results in the paper follow from well-known results about combinators and the relation between them and spaces with combinators. (3) As for the representation of numbers, which appears to be new, it could just as easily have been done in terms of combinators or λ-terms, and such a presentation would, in the reviewer's opinion, have attracted more general interest than the presentation given here. (If it is not new, then it would have been nice if the author had given a reference for it.) Reviewed by J. P. Seldin.

Huet, G. (1993). An analysis of Böhm's theorem. *Theoret. Comput. Sci.*, 121(1-2):145–167. A collection of contributions in honour of Corrado Böhm on the occasion of his 70th birthday.

[MR95d:03011] This is a very elegant paper which gives an algorithmic development of all the fundamental notions of λ-calculus, such as substitution, β- and η-reductions, Böhm trees, the Böhm -out technique, etc. This allows the author to analyze constructively the separability theorem of λ-calculus (better known as Böhm 's theorem). Böhm 's theorem is an outstanding syntactic result which gives a very strong requirement for a structure to be a model of the λ-calculus.

The descriptive language is an actual programming language, CAML. CAML is a language of the ML family developed at INRIA Roquencourt. The choice of an actual programming language makes the treatment more rigorous and more constructive.

This paper is a very appropriate homage to Corrado Böhm on the occasion of his 70th birthday. For the entire collection see MR 94e:68005. Reviewed by Mariangiola Dezani-Ciancaglini.

Intrigila, B. (1993). The basis decision problem in λ-calculus. *Math. Logic Quart.*, 39(2):178–180.

[MR95a:03027] A set of combinators (closed lambda terms) X is called a basis if every combinator is convertible to an applicative combination of elements of X. The following is proved. (i) It is undecidable whether a finite X is a basis. (ii) It is undecidable whether a singleton M is a basis. The proof can be given in a shorter way than in the paper: statement (i) follows from (ii) and (ii) from Scott's theorem that every nontrivial set of combinators that is closed under conversion is undecidable. In spite of this short proof the result had not occurred to the reviewer before seeing the paper and is worth mentioning. Reviewed by Henk Barendregt.

López-Escobar, E. G. K. (1993). Graphical representation of the Church-Rosser theorem. *Rev. Colombiana Mat.*, 27(3-4):267–280.

[MR96b:03023] The author represents terms in combinatory systems by tree-like diagrams which he calls "templates" (plantillas). Using this notation, supposedly a modified version of notation used by G. Frege [Begriffsschrift, Nebert, Halle, 1879; Jbuch 11, 48], he gives a simple proof of the Church-Rosser theorem for a large class of combinatory systems that are characterized by certain conditions on the reduction rules. Reviewed by Dorothy Bollman.

Plotkin, G. D. (1993). Set-theoretical and other elementary models of the λ-calculus. *Theoret. Comput. Sci.*, 121(1-2):351–409. A collection of contributions in honour of Corrado Böhm on the occasion of his 70th birthday.

[MR95b:03014] This paper consists of two separate parts. Part I is a previously unpublished 1972 memorandum in which the author presents one of the first constructions of denotational models of the untyped λ-calculus and compares his models with the D^∞-models of D. Scott (1971) in the framework of continuous complete lattices. In Part II the author gives a very readable general overview of the developments in the subject since 1972, focusing on elementary constructions of models of the λ-calculus. Part II begins with a brief discussion of models of the λ-calculus in set theories with anti-foundation axioms. Then the author reviews the model $P(T_C)$ of Plotkin (1972) (studied in Part I) and the closely related models of Scott (1976, 1980) (among them the well-known $P(\omega)$ model) and of Engeler (1981). A general algebraic framework for the study of such models is provided by the Scott-Engeler algebras studied by Longo (1982) and the preordered Scott-Engeler algebras considered by Krivine (1990). The author reviews and strengthens some previous results by these authors and proves that the only isomorphisms between $\lambda\beta$-models arising from preordered SE-algebras are the obvious ones (the corresponding result for unordered SE-algebras is due to Shellinx, 1991). After remarking that SE-algebras cannot provide nontrivial extensional models, the author goes on to consider in depth one of the most general frameworks in which models of λ-calculus (extensional or not) can be constructed: the extended applicative type structures (EATS) studied by Coppo, Dezani-Ciancaglini, Honsell, Longo, Barendregt et al. in a series of papers starting in

1982-1983. The idea is that the denotation of a term is the set of all the types which can be assigned to it, and the power of the system lies in the presence of intersection among the type constructors. It is observed that all the extensional models in the category of algebraic complete lattices can be so obtained, but some nonextensional models of Scott (1986) cannot be induced by EATS. The relationship between EATS and the D^{∞}-models of Scott is also explored. Finally the author studies some general properties of models of the $\lambda\beta$-calculus in the category of complete partial orders, showing in particular that the order is first-order definable from application. For the entire collection see MR 94e:68005. Reviewed by Alessandro Berarducci.

Statman, R. (1993). Some examples of nonexistent combinators. *Theoret. Comput. Sci.*, 121(1-2):441–448. A collection of contributions in honour of Corrado Böhm on the occasion of his 70th birthday.

[**MR94j:03038**] This note shows that some additional conditions cannot be imposed on certain kinds of terms in λ-calculus and/or combinatory logic, such as fixed-point combinators, recurrent combinators, universal generators, and easy terms. In particular, the author states the nonexistence of "hypereasy" terms, of uniform universal generators and of "hypercurrent" combinators. Some of these facts were already proved in the literature and some are proved here. The proof methods are very interesting and display impressive virtuosity. Moreover, the original results are very strong. Therefore this is an outstanding paper. For the entire collection see MR 94e:68005. Reviewed by Mariangiola Dezani-Ciancaglini.

Takeuti, I. (1993). On the supremum of the values of λ+-terms. *Sūrikaisekikenkyūsho Kōkyūroku*, (818):121–130. Mathematical logic and applications '92 (Japanese) (Kyoto, 1992).

[**MR1251350**] This item will not be reviewed.

Wos, L. (1993). The kernel strategy and its use for the study of combinatory logic. *J. Automat. Reason.*, 10(3):287–343.

[**MR94d:03027**]

Summary: "Barendregt defines combinatory logic as an equational system satisfying the combinators S and K with $((Sx)y)z = (xz)(yz)$ and $(Kx)y = x$; the set consisting of S and K provides a basis for all of combinatory logic. Rather than studying all of the logic, logicians often focus on fragments of the logic, subsets whose basis is obtained by replacing S or K or both by other combinators. In this article, we present a powerful new strategy, called the kernel strategy, for studying fragments in the context of questions concerned with fixed point properties. Interest in such properties rests in part with the relation to normal forms and paradoxes. We show how the kernel strategy was used to answer a number of open questions, offering abundant evidence that the availability of the kernel strategy marks a singular advance for automated reasoning. In all of our experiments with this strategy applied by an automated reasoning program, the rate of success has been impressively high and the CPU time to obtain the desired information startlingly small. For each fragment we study, we use the kernel strategy to attempt to determine whether the strong or the weak fixed point property holds. Where **A** is a given fragment with basis **B**, the strong fixed point property holds for **A** if and only if there exists a combinator y such that, for all combinators x, $yx = x(yx)$, where y is expressed purely in terms of elements of **B**. The weak fixed point property holds for **A** if and only if for all combinators x there exists a combinator $y = xy$, where y is expressed purely in terms of the elements of **B** and the combinator x. Because the use of the kernel strategy is so effective in addressing questions focusing on either fixed point property, its formulation marks an important advance for combinatory logic. Perhaps of especial interest to logicians is an infinite class of infinite sets of tightly coupled fixed point combinators (presented here), whose unexpected discovery resulted directly from the application of the strategy by an automated reasoning program. We also offer various open questions for possible research and focus on an automated reasoning program and input files that may prove useful for such research."

Barendregt, H. (1994). Discriminating coded lambda terms. In *From universal morphisms to megabytes: a Baayen space odyssey*, pages 141–151. Math. Centrum Centrum Wisk. Inform., Amsterdam.

[MR99a:03017] This paper describes codings $\ulcorner M \urcorner$ of λ-terms M which can be decoded by self interpreters E, i.e. $E\ulcorner M\urcorner = M$. (Such a coding cannot be represented by a single combinator for arbitrary M.) Using these, discriminators Δ can be constructed such that, for λ-terms M and $N, \Delta\ulcorner M\urcorner\ulcorner N\urcorner = \text{true}$ if $M \equiv N$, and $\Delta\ulcorner M\urcorner\ulcorner N\urcorner = \text{false}$ otherwise.

The author constructs a simpler new discriminator for closed terms or terms with no more than a fixed number of variables, which allows the relation $\equiv$ to be replaced by α-equality. For the entire collection see MR 98h:00042. Reviewed by Martin W. Bunder.

Böhm, C. and Intrigila, B. (1994). The ant-lion paradigm for strong normalization. *Inform. and Comput.*, 114(1):30–49.

[MR95i:03027] According to the authors, an ant-lion is a trap (or waiting) state, which is a λ-term in normal form; if this term is applied to an "ant" term, then it eats it, entering in an active state, some action is performed, and at the end the trap state is reset. The authors propose the ant-lion paradigm in order to find strongly normalisable solutions of fixed point equations in λ-calculus. They use it here mainly to prove the adequacy theorem for numeral systems in the λ-I-calculus and in the λ-δ_{K}-calculus, which is an intermediate between λ-I-calculus and the usual (= λ-K) calculus, obtained by extending the former by a restricted K as a δ-rule. (The adequacy theorem, which is known to be true for the usual calculus, asserts that a sufficient condition for a set N of λ-terms to allow the representation of recursive functions is the existence of four combinators relative to N: 0, predecessor, successor, and test for 0.) Reviewed by Chantal Berline.

Böhm, C., Piperno, A., and Guerrini, S. (1994). λ-definition of function(al)s by normal forms. In *Programming languages and systems—ESOP '94 (Edinburgh, 1994)*, pages 135–149. Springer, Berlin.

[MR95f:03022] Summary: "Lambda-calculus is extended in order to represent

a rather large class of recursive equation systems, implicitly characterizing function(al)s or mappings of some algebraic domain into arbitrary sets. Algebraic equality will then be represented by $\lambda\beta\delta$-convertibility (or even reducibility). It is then proved, under very weak assumptions on the structure of the equations, that there always exist solutions in normal form (interpretation theorem). Some features of the solutions, such as the use of parametric representations of the algebraic constructors, higher-order solutions by currification, definability of functions on unions of algebras, etc., have been easily checked by a first implementation of the mentioned theorem, the CuCh machine." For the entire collection see MR 95e:68012.

Budimac, Z., Maćoš, D., and Ivanović, M. (1994). Another bracket abstraction algorithm. In *VIII Conference on Applied Mathematics (Tivat, 1993)*, pages 23–28. Univ. Montenegro, Podgorica.
[MR1318490] This item will not be reviewed.

Curien, P.-L. and Hardin, T. (1994). Yet yet a counterexample for $\lambda + SP$. *J. Funct. Programming*, 4(1):113–115.
[MR1279736] There will be no review of this item.

Ghilezan, S. (1994). Application of typed lambda calculi in the untyped lambda calculus. In *Logical foundations of computer science (St. Petersburg, 1994)*, pages 129–139. Springer, Berlin.
[MR95m:03030] Summary: "We discuss some properties of typed lambda calculi which can be related and applied to the proofs of some properties of the untyped lambda calculus. The strong normalization property of the intersection type assignment system is used in order to prove the finiteness of developments property of the untyped lambda calculus in [J.-L. Krivine, Lambda-calculus, types and models, English translation, Horwood, New York, 1993; MR 94h:03034]. Similarly, the strong normalization property of the simply typed lambda calculus can be used for the same purpose. Typability in various intersection type assignment systems characterizes lambda terms in normal form, normalizing, solvable and unsolvable terms. Hence, its application in the proof of the genericity lemma turns out to be appropriate." For the entire collection see MR 95j:03003.

Intrigila, B. (1994). Some results on numerical systems in λ-calculus. *Notre Dame J. Formal Logic*, 35(4):523–541.
[MR97b:03023] The paper under review is a systematic study of numeral systems in the $\lambda\beta\eta$-calculus and contains some new and interesting results concerning the existence or nonexistence of numeral systems with various properties. First it is shown that the adequacy conditions, namely the existence of a successor, a predecessor, and a test for zero, are to a large extent independent of each other. A difficult result obtained in this respect is the existence of a recursively enumerable sequence d_n of closed normal forms with a predecessor $p(d_{n+1}) = d_n$ but without successor or a test for zero. Then various examples of paradoxical numeral systems are considered. One of them settles negatively a problem of Böhm concerning strong normalization

(a different example was independently found by Statman): there is an adequate numeral system with terms in normal form such that for every choice of the predecessor, successor and test for zero, these functions applied to any numeral are not strong normalizing. It is also shown that the minimization operator cannot be defined from the successor and predecessor only, thus solving a problem in the book of H. B. Curry, J. R. Hindley and J. P. Seldin [Combinatory logic, Vol. II, North-Holland, Amsterdam, 1972; Zbl 242.02029]. Finally, an example is given of an undecidable numeral system without a test for zero which is nevertheless adequate for the representation of recursive functions of any number of arguments, thus solving a problem of Barendregt and Barensen. Reviewed by Alessandro Berarducci.

Johann, P. (1994). Normal forms in combinatory logic. *Notre Dame J. Formal Logic*, 35(4):573–594.

[MR96d:03025] Summary: "Let R be a convergent term rewriting system, and let CR-equality on (simply typed) combinatory logic terms be the equality induced by $\beta\eta R$-equality on terms of the (simply typed) lambda calculus under any of the standard translations between these two frameworks for higher-order reasoning. We generalize the classical notion of strong reduction to a reduction relation which generates CR-equality and whose irreducibles are exactly the translates of long βR-normal forms. The classical notion of strong normal form in combinatory logic is also generalized, yielding yet another description of these translates. Their resulting tripartite characterization extends to the combined first-order algebraic and higher-order setting the classical combinatory logic descriptions of the translates of long β-normal forms in the lambda calculus. As a consequence, the translates of long βR-normal forms are easily seen to serve as canonical representatives for CR-equivalence classes of combinatory logic terms for non-empty, as well as for empty, R." Reviewed by Martin W. Bunder.

Kerth, R. (1994). $2^{\aleph_0}$ modèles de graphes non équationnellement équivalents. *C. R. Acad. Sci. Paris Sér. I Math.*, 318(6):587–590.

[MR95d:03012] Summary: "In a book by H. P. Barendregt [The lambda calculus, Revised edition, North-Holland, Amsterdam, 1984; MR 86a:03012] we find a syntactic proof of the existence of $2^{\aleph_0}$ λ-theories. We give a semantic proof of this result by constructing $2^{\aleph_0}$ graph models which are pairwise not equationally equivalent."

Regnier, L. (1994). Une équivalence sur les lambda-termes. *Theoret. Comput. Sci.*, 126(2):281–292.

[MR95d:03013] This paper studies an equivalence relation between (untyped) λ-terms which identifies terms that differ only by permutation of β-redexes, the so-called σ-equivalence ($=_\sigma$), which is the equivalence relation corresponding to σ-reduction ($\to_\sigma$), the smallest compatible relation on λ-terms containing $((\lambda xU)V)W \to_\sigma (\lambda x(U)W)V$ (for $x \notin W$) and $(\lambda x\lambda yU)V \to_\sigma \lambda y(\lambda xU)V$ (for $y \notin V$). Let $\|T\|_S$ denote the length of T's β-reduction according to a strategy S. It is shown that $\|T_1\|_S = \|T_2\|_S$ for σ-equivalent terms T_i, where S is G (left-most reduction), H (head-reduction) or B (perpetual reduction, i.e. left-most reduction, except when the term is of the form $(\lambda xU)V$ with $x \notin U$ and V not normal, in which case one first applies perpetual reduction to V).

A theorem of Barendregt says that a term is strongly normalizing iff its perpetual reduction is finite. Here it is shown that if T has a finite perpetual reduction and $T \to_\beta T'$, then $\|T'\|_B < \|T\|_B$. Moreover, $\|T\|_B$ is proven to be in fact the largest possible β-reduction-length of T. Reviewed by Harold Schellinx.

Takahashi, M. (1994). A simple proof of the genericity lemma. In *Logic, language and computation*, pages 117–118. Springer, Berlin.
[MR95b:03015] Summary: "A short direct proof is given for the fundamental property of unsolvable λ-terms: if M is an unsolvable λ-term and $C[M]$ is solvable, then $C[N]$ is solvable for any λ-term N. (Here $C[]$ denotes an arbitrary context.)" For the entire collection see MR 95b:68015.

Trigg, P., Hindley, J. R., and Bunder, M. W. (1994). Combinatory abstraction using **b**, **b′** and friends. *Theoret. Comput. Sci.*, 135(2):405–422.
[MR95m:03034] Summary: "In this paper we characterise precisely the sets of terms whose abstractions can be defined using the following partial bases of combinators: $\{\mathbf{B}, \mathbf{B}', \mathbf{I}\}$, $\{\mathbf{B}, \mathbf{B}', \mathbf{I}, \mathbf{W}\}$, $\{\mathbf{B}, \mathbf{B}', \mathbf{I}, \mathbf{K}\}$, $\{\mathbf{B}, \mathbf{T}, \mathbf{I}\}$, $\{\mathbf{B}, \mathbf{T}, \mathbf{I}, \mathbf{W}\}$ and $\{\mathbf{B}, \mathbf{T}, \mathbf{I}, \mathbf{K}\}$. The reduction axioms for $\mathbf{B}'$ and $\mathbf{T}$ are $\mathbf{B}'XYZ \rhd Y(XZ)$, $\mathbf{T}XYZ \rhd YXZ$. The first two $\mathbf{B}'$-bases correspond via type-assignment to two interesting implicational logics. $\mathbf{T}$ has the re-ordering property of $\mathbf{B}'$ but not its bracketing property, and turns out to be strictly stronger than $\mathbf{B}'$ but strictly weaker than CI whose reduction axiom is $\mathbf{CI}XY \rhd YX$." Reviewed by Alessandro Berarducci.

Barendregt, H. (1995). Enumerators of lambda terms are reducing constructively. *Ann. Pure Appl. Logic*, 73(1):3–9. A tribute to Dirk van Dalen.
[MR96c:03028] Summary: "A closed λ-term E is called an enumerator if $\forall M \in \Lambda^\circ\, \exists n \in \mathbf{N}\, E\ulcorner n \urcorner =_\beta M$. Here Λ° is the set of closed λ-terms, $\mathbf{N}$ is the set of natural numbers and the $\ulcorner n \urcorner$ are the Church numerals $\lambda fx.f^n x$. Such an E is called reducing if moreover $\forall M \in \Lambda^\circ\, \exists n \in \mathbf{N}\, E\ulcorner n \urcorner \twoheadrightarrow_\beta M$. In 1983 I conjectured that every enumerator is reducing. An ingenious recursion-theoretic proof of this conjecture by Statman is presented in [H. P. Barendregt, J. Funct. Programming 2 (1992), no. 2, 233–236; MR 94e:03010]. The proof is not intuitionistically valid, however. Dirk van Dalen has encouraged me to find intuitionistic proofs whenever possible. In the lambda calculus this is usually not difficult. In this paper an intuitionistic version of Statman's proof is given. It took me somewhat longer to find it than in other cases."

Broda, S. and Damas, L. (1995). A new translation algorithm from lambda calculus into combinatory logic. In *Progress in artificial intelligence (Funchal, 1995)*, pages 359–370. Springer, Berlin.
[MR98g:03040] Summary: "Systems of lambda calculus are of importance for most knowledge representation theories and in particular for several systems for natural language processing. During the implementation of lambda systems several problems arise that are directly related to the presence of bound variables. These problems can be avoided by using translations from lambda calculus into combinatory systems, which give rise to extremely simple reduction machines. In this article we present

and prove the correctness of a translation algorithm that, when compared with other systems, has quite good properties in terms of memory space as well as in terms of length of evaluations." For the entire collection see MR 97a:68005.

David, R. (1995). Une preuve simple de résultats classiques en λ calcul. *C. R. Acad. Sci. Paris Sér. I Math.*, 320(11):1401–1406.

[**MR96c:03029**] Summary: "We give a simple proof of three basic theorems of pure lambda calculus: the Church-Rosser theorem, the standardisation theorem and the finiteness of developments theorem. We also give an extension—in the pure lambda calculus—of the latter that immediately gives the strong normalisation theorem for the type system D (simple types with intersection)."

Honsell, F. and Lenisa, M. (1995). Final semantics for untyped λ-calculus. In *Typed lambda calculi and applications (Edinburgh, 1995)*, pages 249–265. Springer, Berlin.

[**MR1477987**] This item will not be reviewed individually. For details of the collection in which this item appears see 98e:03020.

Jiang, Y. (1995). Consistency of a λ-theory with n-tuples and easy term. *Arch. Math. Logic*, 34(2):79–96.

[**MR97b:03024**] Let G be a finite group of order n considered as a subgroup of the symmetric group S_n. Add to the λ-calculus constants $c, \pi_1, \cdots, \pi_n$ and $\{c_\sigma\}_{\sigma \in G}$. The set $U(G)$ is the universal closure of the following set of equations, writing $\langle u_1, \cdots, u_n \rangle \equiv cu_1 \cdots u_n$: $\langle \pi_1 u, \cdots, \pi_n u \rangle = u$; $\pi_k \langle u_1, \cdots, u_n \rangle = u_k$ for $1 \leq k \leq n$; $c_\sigma \langle u_1, \cdots, u_n \rangle = \langle u_{\sigma 1}, \cdots, u_{\sigma n} \rangle$; $c_\sigma(uv) = (c_\sigma u)(c_\sigma v)$.

Let $\Omega \equiv (\lambda x.xx)(\lambda x.xx)$. Let t be an arbitrary λ-term. The main result in the paper is the consistency of $\lambda\beta\eta + U(G) + \Omega = t$. This is proved by a semantic argument, extending in a clever way the forcing argument of J. Baeten and B. Boerboom [Nederl. Akad. Wetensch. Indag. Math. 41 (1979), no. 2, 111–120; MR 80h:03018], who showed that $\lambda\beta + \Omega = t$ is consistent. The model construction is interesting in its own right.

Reviewer's remark: The notation is sometimes a bit difficult. For example, Ω is written as $(\lambda x.(x)x)\lambda x.(x)x$. Reviewed by Henk Barendregt.

Kauffman, L. H. (1995). Knot logic. In *Knots and applications*, pages 1–110. World Sci. Publishing, River Edge, NJ.

[**MR96k:03028**] In this readable paper the author introduces the use of knot and link diagrams for representing nonstandard sets and for representing the formalism of combinatory logic (lambda calculus). These diagrammatics present an interesting view of modern topics in topology and also of the foundations of mathematics and physics.

The titles and highlights of each section are the following. (1) Introduction. (2) Sets, knots, recursions. In this section the author reviews concepts of set theory from an original point of view, introduces ideas from knot theory and discusses re-entry and recursive forms in relation to knots, wild embeddings and fractals. The section also includes a discussion of indicational calculi, nonstandard logic, quantum logic and boundary logic. (3) Knot set theory. A set theory, introduced in this section and called "Knot set theory", is treated primarily in terms of the theory of knots and links in three-dimensional space. In this set theory, with the membership relation represented by one arc underpassing another, it is possible to indicate sets that are

members of themselves and sets that are members of each other. The diagrammatic representation of such knot sets is so constructed that topologically equivalent diagrams represent the same set. One of the consequences in involving the topology in this way is that knot sets use a "fermionic" convention that identicals cancel in pairs (while ordinary set theory is "bosonic": $\{a, a\} = \{a\}$, for all a). (4) Arrow epistemology. The author here discusses concepts of reference in relation to knot set theory. (5) Lambda calculus and topology. This section contains an interesting discussion of the lambda calculus in terms of a nonassociative composition in knot diagrams, represented by one arc underpassing another. The author shows how to write key constructions in the lambda calculus (such as the Church-Curry fixed point theorem) in terms of these diagrams, emphasizing the relationship of this formalism with the topology and with the algebraic structures of quandle, crystal and rack (used by knot theorists). (6) Interlock algebra. A complete classifier for knots in the lambda formulation is given. (7) The LD-Magma. The author sketches P. DeHornoy's inductive construction that embeds the LD-Magma (a free nonassociative and noncommutative algebra generated by one element and such that $a(bc) = (ab)\ (ac)$, for all a, b, c) into the Artin braid group. (8) On Gödel's theorem, self-reproducing machines, knots and lambda calculus. The author points out the pattern that relates the fixed point theorem in lambda calculus to recursive forms, self-reference and Gödel's incompleteness theorem.

He discusses mechanisms of replication, and even how to make knots reproduce themselves. (9) Quantum knots and topological quantum field theory. This is an introduction to the logic of Dirac brackets in the context of topological invariants. (10) Knots and circuits. The author discusses relations between knot theory, electricity and switching circuit theory. (11) Logic and circuit design—knot automata. The author considers a class of automata that are direct abstractions of digital circuitry disclosing analogies with knot theory. In this context he shows that quandles, crystals and racks implicate a concept of knot automata. (12) Pregeometry. The author interprets the diagrammatic languages for knots and links as J. Wheeler's pregeometry. In an appendix he discusses the bracket model for the Jones polynomial. For the entire collection see MR 96g:57001. Reviewed by R. Gylys.

Oberoi, S. (1995). $\lambda_{B'}$—a λ-calculus with a generalized β-reduction rule. *Inform. Process. Lett.*, 54(1):45–53.

[MR96b:03025] This paper presents a generalized notion of β-reduction (called β'-reduction) which essentially allows unordered Currying in the classical λ-calculus. The author proves that β'-reduction satisfies the fundamental properties of β-reduction, i.e. the Church-Rosser property, the normal order reduction property and the finiteness of developments. Moreover, the sets of normalizing and strongly normalizing terms coincide for the two notions of reduction. These results are not surprising, essentially because of the strong similarity between β-reduction and β'-reduction. In spite of this, the paper is a valuable contribution to the theory of pure λ-calculus, since the notion of β'-reduction is appealing and the proofs are carefully developed. Reviewed by Mariangiola Dezani-Ciancaglini.

Takahashi, M. (1995). Parallel reductions in λ-calculus. *Inform. and Comput.*, 118(1):120–127.

[MR96e:03017] Summary: "The notion of parallel reduction is extracted from the simple proof of the Church-Rosser theorem by Tait and Martin-Löf. Intuitively, this means to reduce a number of redexes (existing in a λ-term) simultaneously. Thus in the case of β-reduction the effect of a parallel reduction is the same as that of a 'complete development' which is defined by using 'residuals' of β-redexes. A nice feature of parallel reduction, however, is that it can be defined directly by induction on the structure of λ-terms (without referring to residuals or other auxiliary notions), and the inductive definition provides us exactly what we need in proving the theorem inductively. Moreover, the notion can be easily extended to other reduction systems such as Girard's second-order system F and Gödel's system T. In this paper, after reevaluating the significance of the notion of parallel reduction in Tait-and-Martin-Löf type proofs of the Church-Rosser theorems, we show that the notion of parallel reduction is also useful in giving short and direct proofs of some other fundamental theorems in the reduction theory of λ-calculus; among others, we give such simple proofs of the standardization theorem for β-reduction (a special case of which is known as the leftmost reduction theorem for β-reduction), the quasi-leftmost reduction theorem for β-reduction, the postponement theorem of η-reduction (in $\beta\eta$-reduction), and the leftmost reduction theorem for $\beta\eta$-reduction."

Bastonero, O. and Gouy, X. (1996). Stabilité forte et incomplétude de la classe des modèles stables du λ-calcul pur. *C. R. Acad. Sci. Paris Sér. I Math.*, 322(10):905–908.

[MR98b:03020] It is shown that the class of stable models is incomplete with respect to pure λ-calculus; more precisely: there is no stable model the theory of which is the theory of Park's strongly stable model. As a result, the authors obtain a semantic proof of this result, which can be adapted to the continuous case.

It is also known that pure λ-calculus + implicational calculus ($E \to E, ((E \to G) \to E) \to ((E \to G) \to G)$), MP (modus ponens): $E, (E \to G) \to G$, where E and G are formulas (with $\to$ as a constant of the λ-calculus) and λ-terms [see also S. Hirokawa, Y. Komori and I. Takeuti, Studia Logica 56 (1996), no. 3, 419–426; MR 98b:03021; see the following review] yields the paradox of Curry (1941). But: Pure λ-calculus + the sequential logics of Gentzen 1934 (without the postulated cut rule (formally equivalent to MP)) for λ-terms (obs) = absolute consistency systems. Hence: MP is the cause of inconsistency in the foundations of mathematics. (This is discussed further in unpublished work of the reviewer.) Reviewed by A. S. Kuzichev.

Bunder, M. W. (1996). Lambda terms definable as combinators. *Theoret. Comput. Sci.*, 169(1):3–21.

[MR97j:03032] This paper characterizes the class of λ-terms which can be translated into combinatory terms using several different sets of combinators. This is done by generalizing the classical abstraction algorithm for the set $\{S, K\}$. Thanks to the Curry-Howard isomorphism, this result allows one to formulate natural deduction systems equivalent to Hilbert-style systems for various implicational logics. This paper is interesting both for the given characterization and for the proof techniques used. It is well written and pleasant to read. Reviewed by Mariangiola Dezani-Ciancaglini.

Gaborovič, S. (1996). A note on discriminability of lambda terms. *Glas. Mat. Ser. III*, 31(51)(2):189–191.

[**MR98b:03024**] This short note proves that the λ-terms belonging to a discriminable set must have a bounded number of initial abstractions. The result is expected, but the proof is not trivial. Reviewed by Mariangiola Dezani-Ciancaglini.

Gouy, X. (1996). Une extension du théorème de Hyland et Wadsworth à une classe abstraite de modèles du λ-calcul pur. *C. R. Acad. Sci. Paris Sér. I Math.*, 322(5):419–422.

[**MR98g:03042**] Summary: "We present a very natural abstract class of semantics of pure λ-calculus (the regular Cartesian closed categories), which includes the continuous, stable and strongly stable semantics, and in which any model similar to Scott's model $\mathcal{D}_\infty$ has the same theory as the latter. As a matter of fact, we can extend to it the argument of Hyland and Wadsworth showing that the theory of $\mathcal{D}_\infty$ is the contextual theory induced by the set of solvable terms. This result is in sharp contrast to what happens in typed λ-calculus for the standard continuous, stable and strongly stable models of PCF, and in pure λ-calculus for Park's model and the analogous stable and strongly stable models: in both cases these models have different theories."

Jiang, Y. (1996). Consistency and axiomatization of a natural extensional combinatory logic. *Sci. China Ser. A*, 39(9):897–908.

[**MR1431226**] A review for this item is in process.

Nour, K. (1996). An example of a nonadequate numeral system. *C. R. Acad. Sci. Paris Sér. I Math.*, 323(5):439–442.

[**MR97d:03008**] Summary: "A numeral system is defined by three closed λ-terms: a normal λ-term d_0 for Zero, a λ-term S_d for Successor, and a λ-term for Zero Test, such that the λ-terms $(S_d^i\, d_0)$ are normalizable and have different normal forms. A numeral system is called adequate if and only if it has a λ-term for Predecessor. This note gives a simple example of a nonadequate numeral system."

Paoli, F. (1996). S is constructively complete. *Rep. Math. Logic*, (30):31–47 (1997).

[**MR1642161**] A review for this item is in process.

Plotkin, G. (1996). On a question of H. Friedman. *Inform. and Comput.*, 126(1):74–77.

[**MR97f:03021**] A subset A of an applicative structure $(X, \cdot)$ is separable if any function $f\colon A \to X$ is realized by some $\widehat{f}$ in X in the sense that for all $a \in A$, $f(a) = \widehat{f} \cdot a$. Friedman asked whether there is a model M of the $\lambda\beta\eta$-calculus in which every finite subset is separable. A positive answer is provided in this paper. The model M is constructed by adding new elements to the open term model satisfying certain δ-rules. The separability property implies that M admits no nontrivial partial order. (It does not even admit the structure of a category with an initial object $\perp$.) Another model M' is constructed which contains two absolutely inseparable elements, in the sense that there is no homomorphism of combinatory algebras from

M' to a combinatory algebra such that the images of the two elements are separable. Reviewed by Alessandro Berarducci.

Selinger, P. (1996). Order-incompleteness and finite lambda models (extended abstract). In *11th Annual IEEE Symposium on Logic in Computer Science (New Brunswick, NJ, 1996)*, pages 432–439. IEEE Comput. Soc. Press, Los Alamitos, CA.

[**MR1461854**] This item will not be reviewed.

Zhu, Y. Q. (1996). Constructive solutions for λ-definable functions. *J. Southeast Univ.*, 26(1):109–111.

[**MR1414106**] There will be no review of this item.

Akama, Y. (1997). A λ-to-CL translation for strong normalization. In *Typed lambda calculi and applications (Nancy, 1997)*, pages 1–10. Springer, Berlin.

[**MR99a:03016**] Summary: "We introduce a simple translation from λ-calculus to combinatory logic (CL) such that: A is a strong normalization (SN) λ-term iff the translation result of A is an SN term of CL (the reductions are β-reduction in λ-calculus and weak reduction in CL). None of the conventional translations from λ-calculus to CL satisfy the above property. Our translation provides a simpler SN proof for Gödel's λ-calculus by the ordinal number assignment method. By using our translation, we construct a homomorphism from a conditionally partial combinatory algebra which arises over SN λ-terms to a partial combinatory algebra which arises over SN CL-terms." For the entire collection see MR 98f:68017.

Broda, S. and Damas, L. (1997a). Compact bracket abstraction in combinatory logic. *J. Symbolic Logic*, 62(3):729–740.

[**MR98m:03027**] Summary: "Translations from lambda calculi into combinatory logics can be used to avoid some implementational problems of the former systems. However, this scheme can only be efficient if the translation produces short output with a small number of combinators, in order to reduce the time and transient storage space spent during reduction of combinatory terms. In this paper we present a combinatory system and an abstraction algorithm, based on the original bracket abstraction operator of M. Schönfinkel [Math. Ann. 92 (1924), 305–316; JFM 50.0023.01]. The algorithm introduces at most one combinator for each abstraction in the initial lambda term. This avoids explosive term growth during successive abstractions and makes the system suitable for practical applications. We prove the correctness of the algorithm and establish some relations between the combinatory system and the lambda calculus." Reviewed by Henri Volken.

Broda, S. and Damas, L. (1997b). On combinatory complete sets of proper combinators. *J. Funct. Programming*, 7(6):593–612.

[**MR98k:03034**] Summary: "A combinatory system (or equivalently the set of its basic combinators) is said to be combinatorially complete for a functional system if any member of the latter can be defined by an entity of the former system. In this paper the decision problem of combinatory completeness for finite sets of proper combinators is studied for three subsystems of the pure lambda calculus. Precise characterizations of proper combinator bases for the linear and the affine λ-calculus are given, and the respective decision problems are shown to be decidable. Furthermore, it is determined which extensions with proper combinators of bases for the linear λ-calculus are combinatorially complete for the λI-calculus."

David, R. and Nour, K. (1997). A syntactical proof of the operational equivalence of two λ-terms. *Theoret. Comput. Sci.*, 180(1-2):371–375.
[**MR98c:03036**] Summary: "In this paper we present a purely syntactical proof of the operational equivalence of $I = \lambda xx$ and the λ term J that is the η-infinite expansion of I."

Hasegawa, R. (1997). The generating functions of lambda terms (extended abstract). In *Combinatorics, complexity, & logic (Auckland, 1996)*, pages 253–263. Springer, Singapore.
[**MR1647313**] This item will not be reviewed.

Intrigila, B. (1997). Non-existent Statman's double fixed point combinator does not exist, indeed. *Inform. and Comput.*, 137(1):35–40.
[**MR98m:03028**] A fixed point combinator is a combinator Y such that $YM = M(YM)$. Since $M(YM) =$SIYM, this is equivalent to asserting that $Y =$ SIY, as C. Böhm pointed out [see H. P. Barendregt, The lambda calculus, Revised edition, North-Holland, Amsterdam, 1984; MR 86a:03012 (Section 6.5.3)]. Furthermore, as Böhm also pointed out, if Y is any fixed point combinator, so is Y(SI). R. Statman [Theoret. Comput. Sci. 121 (1993), no. 1-2, 441–448; MR 94j:03038] asked whether there is (in the $\lambda\beta$-calculus) a fixed point combinator Y such that $Y = Y$(SI). This paper shows that there is none. The proof uses a special argument, and the author remarks that there is no general method for proving combinators nonexistent. Reviewed by J. P. Seldin.

Kuper, J. (1997). On the Jacopini technique. *Inform. and Comput.*, 138(2):101–123.
[**MR98m:03029**] This interesting paper concerns the abstract and general formulation of a technique introduced by G. Jacopini (1975) to prove the consistency of extensions of lambda calculus. In this general setting, the Jacopini technique is not only able to cope with known examples of consistent extensions of the untyped lambda calculus but also gives rise to simpler consistency proofs. Moreover, since the formulation is rather abstract it applies to other calculi, as is shown by an example concerning the λ-calculus. Reviewed by Benedetto Intrigila.

Nour, K. (1997a). A conjecture on numeral systems. *Notre Dame J. Formal Logic*, 38(2):270–275.
[**MR99i:03014**] A numeral system is a sequence of λ-terms representing nonnegative integers in λ-calculus. For a numeral system, a function on nonnegative integers is said to be λ-definable when it can be represented by a closed λ-term. It is known

that, for a numeral system, all total recursive functions are λ-definable if and only if the successor, predecessor and zero test functions are λ-definable. The present paper proves the independence of these three functions. For example, there is a numeral system for which the successor and predecessor functions are λ-definable but the zero test function is not λ-definable. This means that, for a numeral system, it is not sufficient to represent successor and predecessor in order to represent all total recursive functions. Furthermore, the author conjectures that this fact can be generalized as follows: There is no pair of total recursive functions f and g such that, for all numeral systems, f and g are λ-definable if and only if all total recursive functions are λ-definable. Reviewed by Hirofumi Yokouchi.

Nour, K. (1997b). A conjecture on numeral systems. *Notre Dame J. Formal Logic*, 38(2):270–275.
[**MR1489413**] A review for this item is in process.

Song, F. M. (1997). A new (ξ) rule in combinatory logic. *Acta Math. Sinica*, 40(5):669–674.
[**MR1612587**] A review for this item is in process.

Statman, R. (1997). Effective reduction and conversion strategies for combinators. In *Rewriting techniques and applications (Sitges, 1997)*, pages 299–307. Springer, Berlin.
[**MR98j:03019**] From the summary: "We prove the following results concerning effective reduction and conversion strategies for combinators. (1) There is an effective one-step cofinal reduction strategy. (2) There is no effective confluence function, but there is an effective one-step confluence strategy. (3) There is an effective one-step enumeration strategy. (4) There is an effective one-step Church-Rosser conversion strategy." For the entire collection see MR 98i:68015.

Çağman, N. and Hindley, J. R. (1998). Combinatory weak reduction in lambda calculus. *Theoret. Comput. Sci.*, 198(1-2):239–247.
[**MR98k:03036**] Summary: "Combinatory logic claims to do the same work as λ-calculus but with a simpler language and a simpler reduction process. In a sense this claim is true: the classical reduction process in λ-calculus is indeed more complex than that in combinatory logic. But by changing its definition only slightly one can define in λ-calculus a perfect analogue of combinatory reduction. This analogue was first formulated 30 years ago but it is still not as well known as it deserves to be, so in the present purely expository paper we try to make it more accessible. We discuss its definition, motivation and its neat relation to substitution."

Dezani-Ciancaglini, M., Intrigila, B., and Venturini-Zilli, M. (1998). Böhm's theorem for Böhm trees. In *Theoretical computer science (Prato, 1998)*, pages 1–23. World Sci. Publishing, River Edge, NJ.
[**MR1700309**] A review for this item is in process.

Folkerts, E. (1998). Invertibility in $\lambda\eta$. In *Thirteenth Annual IEEE Symposium on Logic in Computer Science (Indianapolis, IN, 1998)*, pages 418–429. IEEE Computer Soc., Los Alamitos, CA.
[**MR99i:03012**] Summary: "This paper investigates invertibility properties of surjective and bijective terms in the closed term model of $\lambda\eta$. With the help of unsolvable terms, it is shown that some special surjective terms are right-invertible and that all bijective terms are invertible." For the entire collection see MR 99h:68008.

Kerth, R. (1998). The interpretation of unsolvable λ-terms in models of untyped λ-calculus. *J. Symbolic Logic*, 63(4):1529–1548.
[**MR1665771**] A review for this item is in process.

Pigozzi, D. and Salibra, A. (1998). Lambda abstraction algebras: coordinatizing models of Lambda calculus. *Fund. Inform.*, 33(2):149–200.
[**MR1654292**] A review for this item is in process.

Sørensen, M. H. (1998). Properties of infinite reduction paths in untyped λ-calculus. In *The Tbilisi Symposium on Logic, Language and Computation: selected papers (Gudauri, 1995)*, pages 353–367. CSLI Publ., Stanford, CA.
[**MR1682624**] A review for this item is in process.

Waldmann, J. (1998). Normalization of **s**-terms is decidable. In *Rewriting techniques and applications (Tsukuba, 1998)*, pages 138–150. Springer, Berlin.
[**MR1636695**] A review for this item is in process.

Zhao, X. and Wang, J. (1998). $Y_C\omega_{2n}$ are easy terms. *Chinese Sci. Bull.*, 43(20):1707–1709.
[**MR1667237**] A review for this item is in process.

Honsell, F. and Lenisa, M. (1999). Semantical analysis of perpetual strategies in λ-calculus. *Theoret. Comput. Sci.*, 212(1-2):183–209. Gentzen (Rome, 1996).
[**MR1672080**] A review for this item is in process.

Piperno, A. (1999). An algebraic view of the Böhm-out technique. *Theoret. Comput. Sci.*, 212(1-2):233–246. Gentzen (Rome, 1996).
[**MR1672088**] Summary: "Using an algebraic representation of closed β-normal forms in λ-calculus, we rephrase Böhm's theorem as an equality predicate between elements of a term algebra. The algebraic interpretation presented gives new insight into the Böhm-out technique and allows for original applications of the method." For the entire collection see MR 99h:68005.

1.3 Simply typed lambda calculus and combinatory logic

Gandy, R. O. (1980). Proofs of strong normalization. In *To H. B. Curry: essays on combinatory logic, lambda calculus and formalism*, pages 457–477. Academic Press, London.

[MR82i:03024] The author proves strong normalization of typed lambda terms (including 0, successor and +) as follows. Given the full type structure $T = (T_\alpha)$ over the natural numbers, a substructure $M = (M_\alpha, <_\alpha)$ can be defined consisting of the hereditarily strictly monotonic functions. It is proved that: (1) Typed λI-terms denote objects in M. (2) For each type α there exists a closed λI-term of type α. (3) Typed λ-terms t can be translated into typed λI-terms t^* of the same type such that t red s implies $t^* >_\alpha s^*$ in M_α. Now, by using a λI-term L of type $\alpha \to 0$ it follows from (1) that $>_\alpha$ is well founded. Hence by (3) there are no infinite reduction paths. Bounds on the maximal length of a reduction path starting with a given term can be computed from the proof. The method is extended to systems including the recursor, and also direct products and sums (covering $\supset$, & and $\vee$-reductions a la D. Prawitz [Proceedings of the Second Scandinavian Logic Symposium (Oslo, 1970), pp. 235 - 307, North- Holland, Amsterdam, 1971; MR 52 #7871]). (For the entire collection see MR 81j:03005.) Reviewed by Henk Barendregt.

Plotkin, G. D. (1980). Lambda-definability in the full type hierarchy. In *To H. B. Curry: essays on combinatory logic, lambda calculus and formalism*, pages 363–373. Academic Press, London.

[MR82a:03016] Starting from a ground set D let D_σ be a hierarchy. An environment is a map ρ from the variables to $\bigcup D_\sigma$. Then for each term M of the typed λ-calculus, ρ assigns a value $(M)(\rho)$ in $\bigcup D_\sigma$ in the natural way. If M is closed, $(M)(\rho)$ does not depend on ρ, so we write simply (M). An element d in $\bigcup D_\sigma$ is λ-definable if there is a closed M such that $d = (M)$. The problem is to characterize λ-definability. It is noted that λ-definable elements remain invariant under permutations of D, but such invariance is not sufficient for λ-definability. Let $(W, \leq)$ be a quasiorder. A relation $R \subseteq D^k \times W$ is an I-relation if $R(d_1, \ldots, d_k, w)$ implies $R(d_1, \ldots, d_k, w')$ for all $w' \geq w$. R is extended to a relation on $D_\sigma^k \times W$ in a natural way, and then d in D_σ is said to I-satisfy R if $R(d, \ldots, d, w)$ holds for all w. Theorem: Suppose D is infinite. Then d in D_σ is λ-definable if and only if d I-satisfies all I-relations $R \subseteq D^3 \times W$. (For the entire collection see MR 81j:03005.) Reviewed by B. Lercher.

Statman, R. (1980). On the existence of closed terms in the typed λ-calculus. I. In *To H. B. Curry: essays on combinatory logic, lambda calculus and formalism*, pages 511–534. Academic Press, London.

[MR82m:03014a] (Joint review with (Statman, 1981).) In the first paper, the author studies two relations between types over a single ground type in typed lambda-calculus. The first relation is that σ is reducible to τ if there is a closed term M in $\sigma \to \tau$ such that for all closed N_1 and N_2 in σ, $N_1 =_{\beta\eta} N_2 \Rightarrow MN_1 =_{\beta\eta} MN_2$. The second is that σ enumerates τ if there is a closed M in $\sigma \to \tau$ such that for all closed N_1 in τ there is a closed N_2 in σ such that $N_1 =_{\beta\eta} MN_2$. The author is able to give almost the complete structure of the reducibility relation. In particular he shows that the equivalence classes associated with it are well ordered in type $\omega + 2$ or $\omega + 3$ (depending on the answer to an open question). To state the ordering we need some definitions: $rank(0) = 0$ (0 is the ground type), and $rank(\sigma \to \tau) = max(rank(\sigma) + 1, rank(\tau))$; a type is small if it has no negative occurrence of a subtype with at least two components; and a type is large if it is not small. Then the well ordering is given as follows (where each type is reducible to those below it):

0.	types with closed terms	
1.	$0 \to 0$	
$\vdots$		
$n.$	$0 \to (\ldots(0 \to 0)\ldots)$	with $n + 1$ zeros
$\vdots$		
$\omega.$	$(0 \to 0) \to (0 \to 0)$	small types of rank 3 with at most one component of $rank \geq 1$ or $rank$ 2 with at most one component of $rank$ 1
$\omega + 1.$	$0 \to ((0 \to 0) \to (0 \to 0))$	all other small types of $rank$ 2 or 3
$\omega + 2(?).$	$(((0 \to 0) \to 0) \to 0) \to (0 \to 0)$	small types of $rank \geq 4$
$\omega + 3.$	$(0 \to (0 \to 0)) \to (0 \to 0)$	large types

The open question is whether $(((0 \to 0) \to 0) \to 0) \to (0 \to 0)$ is reducible to $(0 \to 0) \to ((0 \to 0) \to (0 \to 0))$. Much less is known with regard to enumeration. The author does prove the following results about it for types with closed terms: (1) If σ can be obtained from a third type $\neq 0$ by substituting τ for 0, then τ does not enumerate σ; (2) let $s(1) = 2$ and $s(n+1) = 2^{s(n)}$; then if $|\tau| > s(|\sigma|)^{\frac{1}{2}}$ then σ does not enumerate τ (where $|\sigma|$ is the number of occurrences of 0 in σ); and (3) if σ occurs positively in τ, then τ enumerates σ. Unlike the case with reducibility, the partial ordering of enumeration among equivalence classes is not a total ordering, since the equivalence class of all types not containing closed terms is incomparable with any other. An example of a pair of incomparable types containing closed terms is given by the author as $((0 \to 0) \to 0) \to 0$ and $(0 \to 0) \to (0 \to 0)$. The work on enumeration has led the author to a proof that the probability that a given type contains no closed term is between 0.14 and 0.375. The author concludes the first paper by considering the enumeration problem. The enumeration problem $\tau_1, \ldots, \tau_n; \tau$ is the set of all pairs of closed terms M_1, M_2 in $\tau_1 \to (\ldots(\tau_n \to \tau)\ldots)$ such that for $1 \leq i \leq n$ there exist N_i in τ_i satisfying $M_1 N_1 \ldots N_n =_{\beta\eta} M_2 N_1 \ldots N_n$. The author gives some conditions for this problem to be decidable and others for it to be undecidable. In the second paper, the author takes up unification problems. While it is known that the problem of unification of all orders ≥ 2 is undecidable, "if one considers the much finer classification according to type there are unification problems of arbitrarily high order which are decidable" (from the author's introduction). These problems are connected with the first paper by the following definitions: let τ be $\tau_1 \to (\tau_2 \to \ldots(\tau_t \to 0)\ldots)$ and σ be $\sigma_1 \to (\sigma_2 \to \ldots(\sigma_s \to 0)\ldots)$; then a unification problem (τ, k), where $0 \leq k \leq t$, is the set of all $\langle M_1, M_2 \rangle$ such that $M_1, M_2 \in \tau$ and for $1 \leq i \leq k$ there exist $N_i \in \tau_i$ such that $M_1 N_1 \ldots N_k =_{\beta\eta} M_2 N_1 \ldots N_k$, and the problem (σ, l)

transforms into (τ, k) if $k \geq l$ and there is an increasing function $\Phi : [1, l] \to [1, k]$ such that for $1 \leq i \leq l$, $\tau_{\Phi(i)} enumerates \sigma_i$ and $\sigma_{l+1} \to (\sigma_{l+2} \to \ldots (\sigma_s \to 0) \ldots)$ is reducible to $\tau_{k+1} \to (\tau_{k+2} \to \ldots (\tau_t \to 0) \ldots)$. The main result is that the set of decidable unification problems is polynomial-time decidable. The proof is based on an ordering relation $\leq$ between types, where $\sigma \leq \tau$ means essentially that σ can be "embedded" in τ by a map which preserves the sign of subtype occurrences. It is shown that this is a well quasi order (i.e., a quasi (pre)order which is well founded and does not have infinitely many incomparables), that it is polynomial time decidable, and that if $\sigma \leq \tau$ then τ enumerates σ. It follows that enumeration is a well quasi ordering, and hence so are reducibility and transforms into. From this it follows that any set open with respect to any of these orderings is polynomial-time decidable. The author still does not know if Hilbert's 10th problem for Diophantine equations in 8 variables is decidable. (Reviewer's remarks: The first paper is very concisely written; in some places the reader could use more help. (2) There are several misprints in the first paper (most of which were confirmed or supplied by the author); p. 514, line 8, 'U_λ' should be 'U_τ'; p. 515, line 14, '$rank \geq 1$' should be '$rank > 1$'; page 522, line 18, 'M' should be 'N'; p. 524, line 2 from bottom, before the two occurrences of '$x_i X_1 \ldots X_{s(i)}$' should be inserted '$\Lambda\sigma(11)$' and '$\Lambda\sigma(1r)$', respectively; p. 525, line 11 from bottom, '$y_i \in \tau(i)$' should be '$y_j \in \tau(j)$', and in the next line '$V_{\tau(1k)}$' should be '$U_{\tau(1k)}$'; p. 529, line 6, 'Lemma 5' should be 'Proposition 9'; and p. 532, line 3 from bottom, 'Proposition 1' should be 'Proposition 2'.) (For the entire collection containing part I see MR 81j:03005.) Reviewed by J. P. Seldin.

Pottinger, G. (1981). The Church-Rosser theorem for the typed λ-calculus with surjective pairing. *Notre Dame J. Formal Logic*, 22(3):264–268.

[MR82j:03015] The author proves the normal form theorem (every term has a normal form) and the Church-Rosser theorem (CRT) for the typed $\lambda\beta\eta$-calculus with extra constants P, L, R (pairing operator and projections) and axioms $P(Lt)(Rt) \to t$, $L(Ptu) \to t$, $R(Ptu) \to u$. Types include Cartesian product. (As the author says, J.W. Klop has shown in his thesis [Combinatory reduction systems, Math. Centrum, Amsterdam, 1980] that CRT fails for the corresponding untyped system.) The results are not deep, and indeed very similar theorems have appeared in the literature. (For combinators with P, L, R there is a proof of the normal form theorem by A.S. Troelstra [Metamathematical investigation of intuitionistic arithmetic and analysis, Section 2.3.7, pp. 97 - 174, Lecture Notes in Math., 344, Springer, Berlin, 1973; MR 56 #2793]. In Section 2.2.6, on which Section 2.3.7 depends, omit all reference to R; the proof then becomes constructive.) But the author's proofs are neat and direct, and his exposition is good. His normal form method is a variant of that in a paper by P.B. Andrews [J. Symbolic Logic 36 (1971), 414 - 432, see p. 417; MR 46 #1551]. He defines a class of so-called "predicative" reductions, and proves that all such reductions terminate in normal forms. To prove CRT, he first deduces CRT for predicative reductions in the obvious way from the above, and then extends it to arbitrary reductions very simply. Reviewed by R. Hindley.

Statman, R. (1981). On the existence of closed terms in the typed λ calculus. II. Transformations of unification problems. *Theoret. Comput. Sci.*, 15(3):329–338.

[MR82m:03014b] See (Statman, 1980).

Schwichtenberg, H. (1982). Complexity of normalization in the pure typed lambda-calculus. In *The L. E. J. Brouwer Centenary Symposium (Noordwijkerhout, 1981)*, pages 453–457. North-Holland, Amsterdam.

[MR85f:03013] Two estimates are given for the number of reduction steps necessary to reach the normal form in the pure typed λ-calculus (the types are generated from a ground type 0). First, a sequence of terms t_n (each of length $3n$) is defined such that it takes at least $2_{n-2} - n$ reduction steps ($2_0 := 1, 2_{n+1} := 2^{2_n}$) to bring t_n to its normal form. Then a particular normalization procedure is set up for which an $\mathcal{E}^4$ upper bound on the number of reduction steps can be obtained. For the entire collection see MR 84i:03010. Reviewed by A. S. Kuzichev.

Statman, R. (1983). λ-definable functionals and $\beta\eta$ conversion. *Arch. Math. Logik Grundlag.*, 23(1-2):21–26.

[MR84m:03022] Two results concerning the semantics of the typed λ-calculus are proved in this paper. (1) The λ-definable functionals in arbitrary type structures coincide with the stable solutions to systems of (λ-free) functional equations. The author understands by stable solutions those which are unique and remain so under perturbations of the structure by homomorphisms. (2) $\beta\eta$-conversion is the maximal consistent congruence relation satisfying typical ambiguity (by Friedman's result it is minimal). (The contents of the paper are precisely summarized in the author's introduction.) Reviewed by A. S. Kuzichev.

Bercovici, I. (1985). Unsolvable terms in typed lambda calculus with fix-point operators. In *Logics of programs (Brooklyn, N.Y., 1985)*, pages 16–22. Springer, Berlin.

[MR87h:68104] The author shows that the problem of determining whether a λY-term has a head normal form is decidable. As a corollary, she concludes that the problem of determining whether a given λY-term is equal to a given Y-free λ-term is also decidable. This supplements an unpublished result of Plotkin that the problem of determining whether a given λY-term is equal to some Y-free λ-term is decidable. For the entire collection see MR 86f:68007. Reviewed by Richard Statman.

Statman, R. (1985). Logical relations and the typed λ-calculus. *Inform. and Control*, 65(2-3):85–97.

[MR87h:03023] From the introduction: "The theme of this paper is that the story of the (simple) typed λ-calculus is the story of a certain class of hereditarily defined relations, here called 'logical' relations. A notion of logical relation has already been introduced by Plotkin. His notion is a special case of ours. The form of words 'logical relation' is particularly appropriate since the principal properties of such relations are closure under infinite conjunctions and existential quantifications, when these are suitably defined.

"Our intention is first to introduce the notion, and then to prove, somewhat sketchily, the main syntactic results on the typed λ-calculus from the fundamental theorem of logical relations. These include the normalization and the Church-Rosser theorems. The Church-Rosser theorem is a case in point. Our proof, which is almost instantaneous, is the first we know of which makes essential use of the type structure. Next we obtain some of the principal semantic results on the typed λ-calculus as corollaries to the fundamental theorem. The first of these is a semantic version of this theorem. Others include the existence of free models and the completeness of the full type structure over a ground domain of size k, when k is infinite. These corollaries are not surprising since models are special sorts of logical relations.

"Our main result is the characterization theorem. Both models of the typed λ-calculus and λ-definable elements of models are characterized in terms of logical relations. We also show that the characterization is in a certain sense optimal." Reviewed by A. S. Kuzichev.

Zlatuška, J. (1985). Normal forms in the typed λ-calculus with tuple types. *Kybernetika (Prague)*, 21(5):366–381.

[MR87g:03019] The author proposes a generalization of the standard typed λ-calculus containing not only application and abstraction, but also "tuple-forming" and projection operations. A similar idea has been used by J. Lambek from the point of view of Cartesian closed categories. It is shown that the corresponding notion of reduction is both strongly normalizing and Church-Rosser. Reviewed by A. S. Kuzichev.

Statman, R. (1986). Solving functional equations at higher types: some examples and some theorems. *Notre Dame J. Formal Logic*, 27(1):66–74.

[MR87d:03035] For any equation E between two terms of the typed lambda calculus Λ a sequence E_n of equations is constructed such that for any model $\mathfrak{A}$ of Λ the equation E is solvable in some extension of $\mathfrak{A}$ if and only if no E_n is solvable in $\mathfrak{A}$. This result is applied to lambda theories and models. Reviewed by G. E. Mints.

de Vrijer, R. (1987). Exactly estimating functionals and strong normalization. *Nederl. Akad. Wetensch. Indag. Math.*, 49(4):479–493.

[MR89g:03021] Introduction: "In this paper we show that one can establish strong normalization for the typed λ-calculus (λ^τ) by giving directly an expression which determines for each term t the number $h(t)$, indicating the exact height of its reduction tree (i.e. the number of reduction steps in a reduction sequence from t of maximal length). Just as in the proof of the finite developments theorem in an earlier paper of ours [J. Symbolic Logic 50 (1985), no. 2, 339–343; MR 87k:03013], the expression we obtain for $h(t)$ can best be understood as counting the reduction steps that result by reducing t according to a 'maximally uneconomic' reduction strategy—never allowing a short cut if a longer route could be taken as well. Such a reduction strategy is easily envisaged and in effect it turns out to correspond to the so-called perpetual strategy, which in the type-free λ-calculus always finds an infinite reduction path if there is one [cf. H. P. Barendregt, The lambda calculus, 13.4, North-Holland, Amsterdam, 1981; MR 83b:03016].

"Contrary to the case of finite developments in our earlier paper [op. cit.], the method of exact estimates does not automatically result in a simpler strong normalization proof for λ^τ. But the method can be modified by abandoning the demand for exactness of the estimates, and then it does give rise to an easy proof of strong normalization."

Zaionc, M. (1987). Word operation definable in the typed λ-calculus. *Theoret. Comput. Sci.*, 52(1-2):1–14.

[MR89d:68016] In the simply typed λ-calculus the set of closed terms of type $\mathbf{B} = (0 \to 0) \to (0 \to 0) \to 0 \to 0$ (where 0 is a basic type) is isomorphic (modulo β-η-conversion) to the set Σ^* of finite words on a binary alphabet $\Sigma = \{a, b\}$. A function $f\ (\Sigma^*)^n \to \Sigma^*$ is λ-definable if there exists a closed term M of type $\mathbf{B}^n \to \mathbf{B}$ which represents f, in the sense that $M\underline{w}_1 \cdots \underline{w}_n =_{\beta\eta} \underline{w}$ if and only if $f(w_1, \cdots, w_n) = w$ (where $\underline{w}$ is the λ-term of type $\mathbf{B}$ representing $w \in \Sigma^*$).

In this paper the author gives a complete characterization of the class of the λ-definable functions on binary words. This is the minimal class of functions closed under composition and containing all constant functions, all projections and the operations of concatenation, substitution, emptiness test, occurrence check for a or b, extraction of maximal prefixes of only a or b. The result can be easily extended to any finite alphabet.

This result is a natural extension of a theorem of H. Schwitchtenberg [Arch. Math. Logik Grundlagenforsch. 17 (1975/76), no. 3-4, 113–114; MR 54 #4928] which characterizes the class of λ-definable functions on the set of natural integers, represented by the type $\mathbf{N} = (0 \to 0) \to 0 \to 0$. Reviewed by M. Coppo.

Dekkers, W. (1988). Reducibility of types in typed lambda calculus. Comment on: "On the existence of closed terms in the typed λ-calculus, I" [in *To H. B. Curry: essays on combinatory logic, lambda calculus and formalism*, 511–534, Academic Press, London, 1980; MR 82m:03014a] by R. Statman. *Inform. and Comput.*, 77(2):131–137.

[MR89h:03021] The author answers an open question in the paper by Statman, who defined a term σ to be reducible to a term τ, notation $\sigma \leq \tau$, if and only if there is a closed term M in $\sigma \to \tau$ such that for all closed terms N_1, N_2 in σ, $N_1 =_{\beta\eta} N_2 \iff MN_1 =_{\beta\eta} MN_2$. He also defined two terms to be equivalent if each is reducible to the other. The equivalence class of σ under this equivalence relation is denoted $[\sigma]$, and these equivalence classes are ordered by the relation $[\sigma] < [\tau]$ if and only if $\sigma \leq \tau$ but not $\tau \leq \sigma$. Statman then proved that the equivalence classes of types are well ordered with type $\omega + 2$ or $\omega + 3$; however he could not settle which. The problem concerned two types, $\equiv (((0 \to 0) \to 0) \to 0) \to 0 \to 0$ and $\nu \equiv (0 \to 0) \to (0 \to 0) \to 0 \to 0$. Statman knew that $\nu \leq$, but he did not know whether $\leq \nu$. The author shows that $\not\leq \nu$, and that therefore the types are well ordered with type $\omega + 3$. Reviewed by J. P. Seldin.

Goerdt, A. (1988). On the expressive strength of the finitely typed lambda-terms. In *Mathematical foundations of computer science, 1988 (Carlsbad, 1988)*, pages 318–328. Springer, Berlin.

[MR1023436] This item will not be reviewed individually. For details of the collection in which this item appears see 90i:68010.

de Bruijn, N. G. (1991). Telescopic mappings in typed lambda calculus. *Inform. and Comput.*, 91(2):189–204.

[MR92m:03015] This paper treats multiple abstractions in a typed lambda calculus with dependent types. In such a calculus one can have contexts, and hence abstractions of the form "$[x_1 : A_1][x_2 : A_2(x_1)]$", in which the type of the second variable depends on the value of the first. The author calls these sequences "telescopes". In this paper he is concerned first with developing a notation to handle abstraction with respect to a telescope as a single operation, and second with developing a calculus which enables him to manipulate telescopes. This calculus involves substitution, beta and eta reduction, and concatenation. Reviewed by Edmund Robinson.

Dowek, G. (1991). A second-order pattern matching algorithm for the cube of typed λ-calculi. In *Mathematical foundations of computer science, 1991 (Kazimierz Dolny, 1991)*, pages 151–160. Springer, Berlin.

[MR1135356] This item will not be reviewed.

Huet, G. (1991). Introduction au λ-calcul typé. In *Logique et informatique: une introduction*, pages 137–162. INRIA, Rocquencourt.

[MR1146964] This item will not be reviewed individually. For details of the collection in which this item appears see 93d:68003.

Schwichtenberg, H. (1991). An upper bound for reduction sequences in the typed λ-calculus. *Arch. Math. Logic*, 30(5-6):405–408.

[MR92d:03007] A bound for the length of arbitrary reduction sequences of any term of the simply typed λ-calculus is given; it is better than that of R. O. Gandy [in To H. B. Curry: essays on combinatory logic, lambda calculus and formalism, 457–477, Academic Press, London, 1980; MR 82i:03024]. Reviewed by Chantal Berline.

Zaionc, M. (1991). λ-definability on free algebras. *Ann. Pure Appl. Logic*, 51(3):279–300.

[MR92b:03016] A free algebra A is determined up to isomorphism by the finite list $[\alpha_1, \cdots, \alpha_n]$ of the arities of its constructors. Consider now the set S of (simple) types, built on one basic type O with the help of $\rightarrow$, and let $\tau_i = O^{\alpha_i} \rightarrow O$ and $\tau^A = \tau_1, \cdots, \tau_n \rightarrow O$. We turn now to the simple typed λ-calculus (with types in S). It is well known that the closed normal terms of type τ^A are in one-to-one correspondence with the elements of A, and therefore that any λ-term of type $(\tau^A)^n \rightarrow \tau^A$ defines some n-ary mapping of A. The problem, which is completely solved here, is to characterize the functions which can be represented in this way; let us call them λ-definable for short. It is proved that the set of λ-definable operations is the minimal set that contains the constant functions and projections and is closed under composition and limited recursion. This generalizes the result of Schwichtenberg for $\mathbf{N} = [1, 0]$ (the natural numbers type, which is generated as an algebra by zero and the successor function), and previous results of the author on λ-definable word operations and tree operations (which correspond, respectively, to the algebras $[1, \cdots, 1, 0]$ and $[2, 0]$). In the case of $\mathbf{N}$, for example, this amounts to characterizing λ-definable functions as "extended polynomials", where the extension is by means of a simple restricted version of "if-then-else". Reviewed by Chantal Berline.

Bruce, K. B., Di Cosmo, R., and Longo, G. (1992). Provable isomorphisms of types. *Math. Structures Comput. Sci.*, 2(2):231–247. International Conference on Symbolic Computation (Zurich, 1990).

[MR93g:68072] Summary: "A constructive characterization is given of the isomorphisms which must hold in all models of the typed lambda calculus with surjective pairing. Using the close relation between closed Cartesian categories and models of these calculi, we also produce a characterization of those isomorphisms which hold in all CCCs. Using the correspondence between these calculi and proofs in intuitionistic positive propositional logic, we thus provide a characterization of equivalent formulae of this logic, where the definition of equivalence of terms depends on having 'invertible' proofs between the two terms. Some work of 1110293M. Rittri [J. Funct. Programming 1 (1991), no. 1, 71–89], on types as search keys in program libraries, provides an interesting example of the use of these characterizations." For the entire collection see MR 93a:68007.

Gray, J. W. (1992). Simultaneous substitution in the typed lambda calculus. In *Mathematical foundations of programming semantics (Pittsburgh, PA, 1991)*, pages 207–220. Springer, Berlin.

[MR1251316] This item will not be reviewed.

Kotov, S. V. (1992). Typed λ-calculus with recursive definitions. In *Logical foundations of computer science—Tver '92*, pages 246–257. Springer, Berlin.

[MR94j:03034] Summary: "This paper introduces an extension of a typed λ-calculus with local recursive definitions, that is, a language combining higher types and recursion. In contrast to Gödel's calculus of recursive functionals, the reduction presented arranges local definitions into a normal form. Essential properties of the reduction are soundness with respect to the outlined categorical semantics, confluence, and strong normalization. Suitability of the language to be a basis for higher-order specification languages is discussed." For the entire collection see MR 94h:68009.

de Groote, P. (1993). Defining λ-typed λ-calculi by axiomatizing the typing relation. In *STACS 93 (Würzburg, 1993)*, pages 712–723. Springer, Berlin.

[MR95a:03026] Summary: "We present a uniform framework for defining different λ-typed λ-calculi in terms of systems to derive typing judgements, akin to H. P. Barendregt's pure type systems [J. Funct. Programming 1 (1991), no. 2, 125–154; MR 93e:03019]. We first introduce a calculus called λ^λ and study its abstract properties. These are, among others, the Church-Rosser property, the property of subject reduction, and the property of strong normalization. Then we show how to extend λ^λ to obtain an inferential definition of R. P. Nederpelt's Λ ["Strong normalization in a typed lambda calculus with lambda structured types", Ph.D. Thesis, Techn.

Hogesch. Eindhoven, Eindhoven, 1973]. One may also extend λ^λ to get inferential definitions of D. T. van Daalen's Λ_β ["The language theory of Automath", Ph.D. Thesis, Techn. Hogesch. Eindhoven, Eindhoven, 1980; Zbl 422:68045], and N. G. de Bruijn's $\Lambda\Delta$ [in Mathematical logic and theoretical computer science (College Park, Md., 1984–1985), 71–92, Dekker, New York, 1987; MR 89b:03026], and we argue that these new inferential definitions are well suited for language-theoretic investigations." For the entire collection see MR 94i:68015.

Di Cosmo, R. and Kesner, D. (1993). A confluent reduction for the extension-typed λ-calculus with pairs, sums, recursion and terminal object. In *Automata, languages and programming (Lund, 1993)*, pages 645–656. Springer, Berlin.

[MR94i:03042] Summary: "We add extensional equalities for the functional and product types to the typed λ-calculus with not only products and terminal object, but also sums and bounded recursion (a version of recursion that does not allow recursive calls of infinite length). We provide a confluent and strongly normalizing (thus decidable) rewriting system for the calculus, which stays confluent when unbounded recursion is allowed. For this purpose, we turn the extensional equalities into expansion rules, and not into contractions as is done traditionally. We first prove the calculus to be weakly confluent, which is a more complex and interesting task than for the usual λ-calculus. Then we provide an effective mechanism to simulate expansions without expansion rules, so that the strong normalization of the calculus can be derived from that of the underlying, traditional, nonextensional system. These results give us the confluence of the full calculus, but we also show how to deduce confluence without the weak confluence property, using only our technique of simulating expansions." For the entire collection see MR 94h:68006.

Dougherty, D. J. (1993a). Higher-order unification via combinators. *Theoret. Comput. Sci.*, 114(2):273–298.

[MR94d:03022] Summary: "We present an algorithm for unification in the simply typed lambda calculus which enumerates complete sets of unifiers using a finitely branching search space. In fact, the types of terms may contain type variables, so that a solution may involve type-substitution as well as term substitution. The problem is first translated into the problem of unification with respect to extensional equality in combinatory logic, and the algorithm is defined in terms of transformations on systems of combinatory terms. These transformations are based on a new method (itself based on systems) for deciding extensional equality between typed combinatory logic terms."

Dougherty, D. J. (1993b). Some lambda calculi with categorical sums and products. In *Rewriting techniques and applications (Montreal, PQ, 1993)*, pages 137–151. Springer, Berlin.

[MR94h:03033] Summary: "We consider the simply typed λ-calculus with primitive recursion operators and types corresponding to categorical products and coproducts. The standard equations corresponding to extensionality and to surjectivity of

pairing and its dual are oriented as expansion rules. Strong normalization and ground (base-type) confluence are proved for the full calculus; full confluence is proved for the calculus omitting the rule for strong sums. In the latter case, fixed-point constructors may be added while retaining confluence." For the entire collection see MR 94f:68028.

Gilezan, S. (1993). A note on typed combinators and typed lambda terms. *Zb. Rad. Prirod.-Mat. Fak. Ser. Mat.*, 23(1):319–329.

[**MR95k:03015**] Summary: "The combinatory completeness of the typed theory of combinators and the equality in expressive power of the typed theory of combinators and typed λ-calculus are proved."

Jacopini, G. and Sontacchi, G. (1993). General recursive functions in a very simply interpretable typed λ-calculus. *Theoret. Comput. Sci.*, 121(1-2):169–178. A collection of contributions in honour of Corrado Böhm on the occasion of his 70th birthday.

[**MR94k:03016**] The typed λ-calculus lacks universality (i.e., it cannot represent all the recursive functions), whereas the untyped calculus does not have a lucid semantic interpretation. The authors strike a balance between the two by introducing, into the typed calculus, four new constants and two reduction rules. The new constants are θ and ρ, which stand for zero and the successor at the lowest type, and Ω_α and Δ_α for each type α, which are related to minimalization and case distinction. The authors give a representation of all the partial recursive functions in this calculus, at each type α, through the Church numerals, and also mention that a representation through numerals $\rho^k\theta$ can be given. On the semantic side, they start from an infinite set that contains θ and i and is closed under ρ, where i is an "indefinite" object. Domains of higher types are constructed in the natural way and the interpretations of terms are given. When the structure is closed under application, and contains the combinators K, S as well as Ω and Δ at various types, two terms are shown to have the same interpretation if they are convertible to each other (but not conversely). At the end of the paper, the authors indicate an extension by introducing the majority operator and show how the power of representation is enhanced. For the entire collection see MR 94e:68005. Reviewed by M. Yasuhara.

Knobel, A. (1993). Constructive set-theoretic models of typed combinatory logic. *J. Symbolic Logic*, 58(1):99–118.

[**MR94f:03020**] Using a combinatory model and the space of singletons of the model, which itself can be made into a model, the author proves that there exists a Kripke-style model of constructive set theory, where every closed extensional simply typed combinatory logic-theory is the theory of a combinatory model with certain sets at ground and total function spaces at higher types. This set universe is well-founded.

Functions at any type level that exist in the hierarchy but are not represented in the language are eliminated by extending them to representable functions, hence a full function space model is created. Reviewed by Martin W. Bunder.

Leivant, D. (1993). Functions over free algebras definable in the simple typed lambda calculus. *Theoret. Comput. Sci.*, 121(1-2):309–321. A collection of contributions in honour of Corrado Böhm on the occasion of his 70th birthday.

[MR95i:03030] From the summary: "We show that a function over a free algebra is definable in the simply typed λ-calculus (modulo the Böhm-Berarducci embedding) iff it is generated by predicative monotonic recurrence. Monotonic recurrence here is the generalization of iteration-with-parameters from $\mathbf{N}$ to arbitrary free algebras, and our predicativity condition uses the notion of tiers introduced by Leivant (1990)". Furthermore, it is shown that two tiers are enough.

The idea of the tier-hierarchy is to capture algebraically the distinct computational roles of a given element of a free algebra.

An independent subrecursive characterization of definable functions was given by M. Zaionc [Ann. Pure Appl. Logic 51 (1991), no. 3, 279–300; MR 92b:03016]. Both generalize Statman's and Schwichtenberg's result for numerical functions. The present one is related to monotonic computing devices, is claimed to be technically more natural, and "captures the foundational link with predicativity". For the entire collection see MR 94e:68005. Reviewed by Chantal Berline.

Leivant, D. and Marion, J.-Y. (1993). Lambda calculus characterizations of poly-time. *Fund. Inform.*, 19(1-2):167–184.

[MR95c:03095] This paper contains some nice characterizations of the PTIME functions inside λ-calculus. One of them identifies the PTIME functions as the functions representable in the simply-typed λ-calculus with pairing, with abstract input (represented by Church-like terms) and concrete output (represented by algebra terms). Another one states that the PTIME functions are those represented in the fragment of second-order typed λ-calculus in which type-quantification is restricted to multiplicative types (i.e., types built up from basic types and Cartesian products). The results of this paper yield the first characterization of a computational complexity class of practical interest within λ-calculus. A brief account of related characterizations of PTIME in the context of subrecursive functions is given. Reviewed by Alessandro Berarducci.

de Groote, P. (1994a). A CPS-translation of the $\lambda\mu$-calculus. In *Trees in algebra and programming—CAAP '94 (Edinburgh, 1994)*, pages 85–99. Springer, Berlin.

[MR96c:03031] Summary: "We present a translation of Parigot's λ-calculus [M. Parigot, in Logic programming and automated reasoning (St. Petersburg, 1992), 190–201, Lecture Notes in Comput. Sci., 624, Springer, Berlin, 1992; MR 94g:03029] into the usual λ-calculus. This translation, which is based on the so-called continuation passing style, is correct with respect to equality and with respect to evaluation. At the type level, it induces a logical interpretation of classical into intuitionistic logic, akin to Kolmogorov's negative translation. As a by-product, we get the normalization of second order typed λ-calculus." For the entire collection see MR 95b:68022.

de Groote, P. (1994b). Strong normalization in a non-deterministic typed lambda-calculus. In *Logical foundations of computer science (St. Petersburg, 1994)*, pages 142–152. Springer, Berlin.

[MR96b:03022] Summary: "In a previous paper [in Logical foundations of computer science (Tver, 1992), 105–116, Lecture Notes in Comput. Sci., Springer, Berlin, 1992], we introduced a non-deterministic λ-calculus (λ-LK) whose type system corresponds exactly to Gentzen's cut-free LK. This calculus, however, cannot be provided with a computational interpretation. Some of the constructs act as oracles and, for this reason, it is not possible to define an effective notion of reduction. In the present paper, we address this problem. We consider a weak version of the implicative fragment of λ-LK, and we define for it a relation of reduction that models, at the level of the terms, the appropriate proof-theoretic notion of proof reduction. This reduction relation satisfies several properties of interest, among others, the property of strong normalization. We prove this last result by using a reducibility argument a la Tait." For the entire collection see MR 95j:03003.

de Vrijer, R. C. (1994). Big trees in a λ-calculus with λ-expressions as types [MR **58** #5085]. In *Selected papers on Automath*, pages 469–492. North-Holland, Amsterdam.

[MR1429416] This item will not be reviewed individually. For details of the collection in which this item appears see 98g:03038.

Di Cosmo, R. and Kesner, D. (1994). Simulating expansions without expansions. *Math. Structures Comput. Sci.*, 4(3):315–362.

[MR96d:03024] Summary: "We add extensional equalities for the functional and product types to the typed λ-calculus with, in addition to products and terminal object, sums and bounded recursion (a version of recursion that does not allow recursive calls of infinite length). We provide a confluent and strongly normalizing (thus decidable) rewriting system for the calculus that stays confluent when allowing unbounded recursion. To do this, we turn the extensional equalities into expansion rules, and not into contractions as is done traditionally. We first prove the calculus to be weakly confluent, which is a more complex and interesting task than for the usual λ-calculus. Then we provide an effective mechanism to simulate expansions without expansion rules, so that the strong normalization of the calculus can be derived from that of the underlying, traditional, non-extensional system. These results give us the confluence of the full calculus, but we also show how to deduce confluence directly from our simulation technique without using the weak confluence property."

Hoofman, R. (1994). Comparing models of the non-extensional typed λ-calculus (extended abstract). In *Logical foundations of computer science (St. Petersburg, 1994)*, pages 164–172. Springer, Berlin.

[MR1321402] This item will not be reviewed.

Pollack, R. (1994). Closure under alpha-conversion. In *Types for proofs and programs (Nijmegen, 1993)*, pages 313–332. Springer, Berlin.

[**MR1285642**] This item will not be reviewed individually. For details of the collection in which this item appears see 95b:68023.

Valentini, S. (1994). A note on a straightforward proof of normal form theorem for simply typed λ-calculi. *Boll. Un. Mat. Ital. A (7)*, 8(2):207–213.

[**MR95g:03038**] This paper presents a new proof of the theorem that every term of the simply typed λ-calculus has a normal form. The proof is a modification of the proof based on the idea of computability due to W. W. Tait [J. Symbolic Logic 32 (1967), 198–212; MR 36 #2483]. At the end of the paper, the author asserts that the idea can be extended to a stronger type system with tuples and projections, and he raises as open problems the idea of proving strong normalization (that every reduction sequence terminates) and applying the idea of the proof to even stronger systems such as second-order typed λ-calculus and Girard's system F [J.-Y. Girard, P. Taylor and Y. Lafont, Proofs and types, Cambridge Univ. Press, Cambridge, 1989; MR 90g:03013].

Reviewer's remarks: (1) The proof of the theorem on closure under substitution is too condensed, as the author acknowledged to the reviewer by e-mail: the reviewer at first thought that there was a gap in the proof leading to a circular argument. (2) The reviewer has found the following misprints: (i) p. 208, conclusion of (app-rule), 'α' should be 'β'; (ii) p. 211, proof of second theorem, first line, 'b: β-exp' should be 'b: β λ-exp'. Reviewed by J. P. Seldin.

van Benthem Jutting, L. S. (1994a). The language theory of λ_∞, a typed λ-calculus where terms are types. In *Selected papers on Automath*, pages 655–683. North-Holland, Amsterdam.

[**MR1429418**] This item will not be reviewed individually. For details of the collection in which this item appears see 98g:03038.

van Benthem Jutting, L. S. (1994b). A normal form theorem in a λ-calculus with types. In *Selected papers on Automath*, pages 371–374. North-Holland, Amsterdam.

[**MR1429413**] This item will not be reviewed individually. For details of the collection in which this item appears see 98g:03038.

de Groote, P. (1995). A simple calculus of exception handling. In *Typed lambda calculi and applications (Edinburgh, 1995)*, pages 201–215. Springer, Berlin.

[**MR98f:68032**] Summary: "We introduce a simply typed λ-calculus ($\lambda^{\rightarrow}_{\mathrm{exn}}$) featuring an ML-like exception-handling mechanism. This calculus, whose type system corresponds to classical logic through the Curry-Howard isomorphism, possesses several interesting properties: among others, Church-Rosser, subject reduction, and strong normalisation. Moreover, its typing system ensures that the reduction of well-typed expressions cannot give rise to uncaught exceptions." For the entire collection see MR 98e:03020.

Dekkers, W. (1995). Inhabitation of types in the simply typed lambda calculus. *Inform. and Comput.*, 119(1):14–17.

[**MR96c:03030**] Summary: "In the 1960s, mathematicians from Eastern Europe showed that every admissible rule in the intuitionistic implicational calculus is derivable. We prove the corresponding proposition for the simply typed lambda calculus: The type ${}_1 \to {}_2 \to \cdots \to {}_m \to v$ is inhabited iff for each substitution s of types for the variables in ${}_1, \cdots, {}_m, v$ the inhabitation of $s({}_1), \cdots, s({}_m)$ implies the inhabitation of $s(v)$."

Di Cosmo, R. and Piperno, A. (1995). Expanding extensional polymorphism. In *Typed lambda calculi and applications (Edinburgh, 1995)*, pages 139–153. Springer, Berlin.
[MR1477980] This item will not be reviewed individually. For details of the collection in which this item appears see 98e:03020.

Doyen, J. (1995). Quelques propriétés du typage simple des fonctions des entiers dans les entiers. *C. R. Acad. Sci. Paris Sér. I Math.*, 321(6):663–665.
[MR96h:03035] Summary: "The main result of this paper is that a simply typed λ-term representing any numeric function 'simplifies' the types of the integers given in arguments (cf. Theorem 1). We show that if such a term is closed with type $N_a \to N_b$ where $N_a = (a \to a) \to (a \to a)$, then a and b must be somehow related. These results follow from a technical lemma which gives a generalization of a result of H. Schwichtenberg [Arch. Math. Logik Grundlagenforsch. 17 (1975/76), no. 3-4, 113–114; MR 54 #4928 (Theorem 2)]."

Kuper, J. (1995). Usability: formalising (un)definedness in typed lambda calculus. In *Computer science logic (Kazimierz, 1994)*, pages 76–90. Springer, Berlin.
[MR98e:03017] Summary: "We discuss the notion of usability, and propose to take it as a formalisation of (un)definedness in typed lambda calculi, especially in calculi based on PCF. We discuss some important properties that make usability attractive as a formalisation of (un)definedness. There is a remarkable difference between usability and solvability: in the untyped lambda calculus the solvable terms are precisely the terms with a head normal form, whereas in typed lambda calculus the usable terms are 'between' the terms with a normal form and the terms with a (weak) head normal form." For the entire collection see MR 98d:68013.

Nour, K. (1995a). Caractérisation opérationnelle des entiers classiques en λC-calcul. *C. R. Acad. Sci. Paris Sér. I Math.*, 320(12):1431–1434.
[MR96e:03014] Summary: "We generalize in this note, by syntactical methods, the result of J. L. Krivine about the operational characterization of classical integers in λC-calculus."

Nour, K. (1995b). Entiers intuitionnistes et entiers classiques en $\lambda\ C$-calcul. *RAIRO Inform. Théor. Appl.*, 29(4):293–313.
[MR96g:03021] Summary: "In this paper, we give a syntactical proof of J. L. Krivine's operational characterization of classical integers (as closed terms typable,

in a classical type system, by the type $N[s^n(0)]$). We also give conditions on the classical type system in order to obtain the intuitionistic integers."

Nour, K. (1995c). Quelques résultats sur le λC-calcul. *C. R. Acad. Sci. Paris Sér. I Math.*, 320(3):259–262.
[MR95k:03016] Summary: "In this note we characterize in an operational way the terms of λC-calculus of type $\forall X\{\perp\to X\}$ and $\forall X\{X \to X\}$."

Riecke, J. G. (1995). Statman's 1-section theorem. *Inform. and Comput.*, 116(2):294–303.
[MR96h:03040] Summary: "Statman's 1-section theorem [R. Statman, in Harvey Friedman's research on the foundations of mathematics, 331–338, North-Holland, Amsterdam, 1985; MR 87m:03018] is an important but little-known result in the model theory of the simply typed λ-calculus. The 1-section theorem states a necessary and sufficient condition on models of the simply typed λ-calculus for determining whether $\beta\eta$-equational reasoning is complete for proving equations that hold in a model. We review the statement of the theorem, give a detailed proof, and discuss its significance."

Simpson, A. K. (1995). Categorical completeness results for the simply-typed lambda-calculus. In *Typed lambda calculi and applications (Edinburgh, 1995)*, pages 414–427. Springer, Berlin.
[MR98f:03015] Summary: "We investigate, in a categorical setting, some completeness properties of beta-eta conversion between closed terms of the simply typed lambda calculus. A Cartesian-closed category is said to be complete if, for any two unconvertible terms, there is some interpretation of the calculus in the category that distinguishes them. It is said to have a complete interpretation if there is some interpretation that equates only interconvertible terms. We give simple necessary and sufficient conditions on the category for each of the two forms of completeness to hold. The classic completeness results of, e.g., H. Friedman and G. D. Plotkin are immediate consequences. As another application, we derive a syntactic theorem of R. Statman characterizing beta-eta conversion as a maximum consistent congruence relation satisfying a property known as typical ambiguity." For the entire collection see MR 98e:03020.

Valentini, S. (1995). On the decidability of the equality theory of simply typed λ-calculi. *Boll. Un. Mat. Ital. A (7)*, 9(1):83–93.
[MR95m:03035] Summary (translated from the Italian): "We present a new proof of the well-known result on the decidability of the equality theory for a simply typed λ-calculus; our proof is based on the extension to equality theory of the approach used in our previous article [Boll. Un. Mat. Ital. A (7) 8 (1994), no. 2, 207–213; MR 95g:03038] to prove the normal-form theorem. The principal novelty of our proof lies in the fact that to obtain this result we do not rely on the Church-Rosser theorem, which is used in the usual proof. Our method of proof is very simple, but it is also

very general and is easily applicable to a variety of simply typed λ-calculi whose equality theory satisfies the β-η-equality."

Zaionc, M. (1995). Lambda representation of operations between different term algebras. In *Computer science logic (Kazimierz, 1994)*, pages 91–105. Springer, Berlin.

[MR1471220] This item will not be reviewed.

Barendsen, E. and Bezem, M. (1996). Polymorphic extensions of simple type structures. With an application to a bar recursive minimization. *Ann. Pure Appl. Logic*, 79(3):221–280.

[MR98i:03023] Summary: "The technical contribution of this paper is threefold. First we show how to encode functionals in a 'flat' applicative structure by adding oracles to untyped λ-calculus and mimicking the applicative behaviour of the functionals with an impredicatively defined reduction relation. The main achievement here is a Church-Rosser result for the extended reduction relation. Second, by combining the previous result with the model construction based on partial equivalence relations, we show how to extend a λ-closed simple type structure to a model of the polymorphic λ-calculus. Third, we specialize the previous result to a countermodel against a simple minimization. This minimization is realized by a bar recursive functional, which contrasts with the results of Spector and Girard which imply that the bar recursive functions are exactly those that are definable in the polymorphic λ-calculus. As a spin-off, we obtain directly the nonconservativity of the extensions of Gödel's T with bar recursion, the fan functional, and Luckhardt's minimization functional, respectively. For the latter two extensions these results are new."

Kamareddine, F. and Nederpelt, R. (1996). A useful λ-notation. *Theoret. Comput. Sci.*, 155(1):85–109.

[MR97f:03020] In a series of papers the authors have proposed an essentially new notation for typed λ-calculi, the item notation. The main advantages of their approach are the following: (1) Some definitions (such as compatibility, the heart of a term and β-redexes) are easier and more natural; (2) it allows a notion of generalized reduction, which seems difficult to translate into the classical λ-calculus; (3) the computation of the canonical type of a given term in a given context becomes straightforward; (4) it is very easy to find the bindings, typings and substitutions of variable occurrences, as well as to determine the status of variables; (6) the description of the normal forms for a substitution calculus a la de Bruijn is dramatically simplified.

The small effort of learning this new notation is amply rewarded by the expressivity, generality and uniformity it provides. Reviewed by Mariangiola Dezani-Ciancaglini.

Takahashi, M., Akama, Y., and Hirokawa, S. (1996). Normal proofs and their grammar. *Inform. and Comput.*, 125(2):144–153. Theoretical aspects of computer software (Sendai, 1994).

[MR97f:03022] The authors study the structure of the set of β-normal forms of a type (in a given context) in the setting of the simply typed lambda calculus. As an application they give an algorithm to decide whether a type is empty or finite. For the entire collection see MR 97b:68011. Reviewed by Eugenio Moggi.

Barbanera, F. and Berardi, S. (1997). The simply typed theory of β-conversion has no maximum extension. *Inform. and Comput.*, 139(1):57–61.

[MR98m:03026] This paper deals with the problem of the existence of a maximum consistent extension of the simply typed theory of β-conversion. The main result is a proof of the nonexistence of maximum extension for the theory of β-conversion with an arbitrary set of atomic types. This is done by defining a family of models such that the sets of equations validated by them form a family of (non-extensional) theories whose union is inconsistent. A maximum extension for the theory of β-conversion should then necessarily contain such a union and hence be inconsistent. For the theory of $\beta\eta$-conversion, the answer depends on the number of types. With more than one atomic type, a negative answer is shown easily in the paper, while with just one atomic type a positive answer was provided by R. Statman [J. Symbolic Logic 47 (1982), no. 1, 17–26; MR 84e:03020]. Reviewed by Ying Jiang.

Bucciarelli, A. (1997a). Bi-models: relational versus domain-theoretic approaches. *Fund. Inform.*, 32(3-4):251–266.

[MR1657235] A review for this item is in process.

Bucciarelli, A. (1997b). Logical reconstruction of bi-domains. In *Typed lambda calculi and applications (Nancy, 1997)*, pages 99–111. Springer, Berlin.

[MR98i:68177] Summary: "We introduce a technique, based on logical relations, which, given two models M and N of a simply typed lambda-calculus L, allows us to construct a model M/N whose L-theory is a superset of both $\mathrm{Th}(M)$ and $\mathrm{Th}(N)$." For the entire collection see MR 98f:68017.

Xi, H. (1997a). Upper bounds for standardizations and an application. In *Computational logic and proof theory (Vienna, 1997)*, pages 335–348. Springer, Berlin.

[MR98j:03020] Summary: "We first present a new proof for the standardization theorem, a fundamental theorem in λ-calculus. Since our proof is largely built upon structural induction on lambda terms, we can extract some bounds for the number of β-reduction steps in the standard β-reduction sequences obtained by transforming a given β-reduction sequence. This result sharpens the standardization theorem. As an application, we establish a superexponential bound for the lengths of β-reduction sequences from any given simply typed λ-term." For the entire collection see MR 98h:03004.

Xi, H. (1997b). Weak and strong beta normalisations in typed λ-calculi. In *Typed lambda calculi and applications (Nancy, 1997)*, pages 390–404. Springer, Berlin.

[MR98g:03047] Summary: "We present translation techniques to study relations

between weak and strong beta-normalisations in various typed lambda-calculi. We first introduce a translation which translates lambda-terms into lambda-I-terms, and show that every lambda-term is strongly beta-normalisable if and only if its translation is weakly beta-normalisable. We then prove that the translation preserves typability of lambda-terms in various typed lambda-calculi. This enables us to establish the equivalence between weak and strong beta-normalisations in these typed lambda-calculi. This translation can handle Curry typing as well as Church typing, strengthening some recent closely related results. This may bring some insights toward answering whether weak and strong beta-normalisations in all pure type systems are equivalent." For the entire collection see MR 98f:68017.

Aczel, P. (1999). Notes on the simply typed lambda calculus. In *Computational logic (Marktoberdorf, 1997)*, pages 57–97. Springer, Berlin.
[**MR1698128**] A review for this item is in process.

1.4 Partial applicative theories

Shabunin, L. V. (1983). On the interpretation of combinators with weak reduction. *J. Symbolic Logic*, 48(3):558–563.
[**MR85g:03026**] The author discusses partial models of systems of combinatory logic (CL), and of λ-calculus with weak reduction. The aim of the article seems to be to show the relations between combinatory logic and partial recursion theory. Using numberings of all partial recursive functions the author defines in a very natural way partial models for the system under consideration and studies the relation between notions in combinatory logic and recursion theory. In particular, the following correspondences are obtained: weak equality in CL corresponds to extensional equality of partial recursive functions, and being in normal form corresponds to definedness. It is shown that the correspondences work best in the case of so-called regular systems of CL, which, loosely speaking, means for systems which can be formulated in the λI-calculus. It is also shown (Theorem 4) that total models for the λ-calculus cannot be constructed by this method. The results are not really surprising when one remembers the fact that CL or λ-calculus and recursion theory have always had very close connections. It seems that the author's results are an exact formulation of Church's intuitions when he invented the λ-calculus. There is a misprint in Lemma 1; $ax \to x$ should be replaced by $ax \to y$. Reviewed by Gerd Mitschke.

Bethke, I. (1987). On the existence of extensional partial combinatory algebras. *J. Symbolic Logic*, 52(3):819–833.
[**MR89b:03025**] A partial combinatory algebra (pca) is an applicative structure $(A,)$, in which $: A^2 \to A$ is partially defined with the property of combinatory completeness (cc). The author shows that some care is needed in the formulation of cc. Extensionality for a pca is defined as usual. The paper studies nontotal extensional pca's (ntepca's). It is shown that the reviewer's axiom for pca's, $sxy = sx'y' \Rightarrow x = x' \& y = y'$, cannot hold in such structures. Moreover these structures cannot be embedded in total pca's, nor have total pca's as substructures.

The main part of the paper is devoted to a construction of an ntepca. The author does this by a suitable modification of the Plotkin-Scott-Engeler graph model. She constructs a cpo A such that the cpo of strict (i.e. $f(\bot) = \bot$) continuous functions $fA \to_s A$ forms a retract of A. (Strict because the bottom element $\bot$ of A is to be considered as undefined.) In this resulting ordered ntepca one has $\mathbf{K}^\infty$ as the supremum of the increasing sequence $\mathbf{K}\Omega$, $\mathbf{K}^2\Omega, \cdots$ (where $\mathbf{K} = \lambda xy.x$, $\Omega = (\lambda x.xx)(\lambda x.xx)$ and $\mathbf{K}^\infty$ is the fixed point of $\mathbf{K}$). Reviewer's remark: A uniformly reflexive structure (URS) in the sense of Wagner and Strong [see, e.g., E. G. Wagner, Trans. Amer. Math. Soc. 144 (1969), 1–41; MR 40 #2543] is a pca with a discriminator δ such that $\delta xyab.$ if $x = y$ then a else b. It can be shown that a URS is never extensional: consider the fixed point of $\lambda x.\delta x(\lambda ab.b)(\lambda ab.a)(\lambda ab.b)$. Reviewed by Henk Barendregt.

Ballard, D. (1988). Combinatory completeness without classical equality. *J. Philos. Logic*, 17(2):115–132.

[MR90m:03027] The author is interested in using λ-calculus to clarify Gödel's diagonal lemma and the recursion theorem, since λ-calculus is "the correct algebra for studying fixed point phenomena". However, this is made difficult by the fact that the λ-calculus presumes that conversion is what the author calls a "classical equality": i.e., it has the properties $(*)a = b \Rightarrow ac = bc$ and $(**)b = c \Rightarrow ab = ac$, which in this context make it an equivalence relation. But the obvious application operator that would prove either of Gödel's diagonal lemma or the recursion theorem fails to satisfy $(**)$, and in fact "there are situations natural for a λ-calculus analysis" where both $(*)$ and $(**)$ fail with respect to the conversion relation natural to the situation.

Instead of reformulating the λ-calculus itself, the author generalizes its underlying semantic structure, the applicative system. In this generalization, the application operator is retained, but equality is replaced by an arbitrary equivalence relation $\approx$ bearing no apparent connection to anything else. The usual interaction of and $\approx$ via congruence axioms, which may partly or completely fail, is replaced by an interaction based on combinatory completeness. Thus, the author is doing a form of combinatory logic. The possible failure of $(*)$ and $(**)$ provides a mode of ranking these systems to measure their nonclassicalness.

The author defines an applicative system as a structure $(M, , \approx)$ where is a binary relation on M and $\approx$ is an equivalence relation. If $(*)$ holds, then the system is said to be semiclassical; if $(**)$ holds as well it is said to be classical. It is said to have strong combinatory completeness if for every term t and distinct variables $x_1, x_2, \cdots, x_n$ which include all those occurring in t, there is an element a of M such that $t \approx ax_1x_2\cdots x_n$; it is said to have weak combinatory completeness if this holds for all terms t whose head is a constant. (Here terms are formed from constants assigned to the elements of M, and ab indicates the application of this term formation.)

A short, transparent algebraic proof then shows the generalized diagonal lemma: If $(M, , \approx)$ is an applicative system with weak combinatory completeness, then for all $a_1, a_2, \cdots, a_n \in M$ there exist $b_1, b_2, \cdots, b_n \in M$ such that $a_ib_1b_2\ldots b_n \approx b_i$. This is followed by proofs of criteria for completeness. An applicative system has weak combinatory completeness if and only if, roughly, for each natural number n there are maps from M to M which, when given an appropriate number of arguments (depending on n), do the work of the combinators K, W, B, and C on terms with n arguments after the number appropriate to the combinator. (The author writes 'D'

for W.) A semiclassical system has weak combinatory completeness if and only if there are maps for K, W, and B which are independent of the number of arguments (and which can take additional arguments). An applicative system has strong combinatory completeness if and only if there are elements which do the work of K, W, and, for each n, the versions of B and C with $n+3$ arguments. And finally, a semiclassical system has strong combinatory completeness if and only if it has the combinators K, B, C, and W.

After a number of examples (which, among other things, show that the criteria for completeness proved in the paper cannot be improved), the author turns to truth combinators. A truth combinator is any I in M such that $Ix \approx x$. An applicative system has strong combinatory completeness if and only if it has weak combinatory completeness and a truth combinator. A negative of $a \in M$ is, by definition, any $a' \in M$ such that for all $b \in M$, $ab \not\approx a'b$. A truth combinator in an applicative system with strong combinatory completeness cannot have a negative. When these facts are applied to the examples, the results are proofs of Tarski's theorem on the undefinability of truth and the Gödel-Rosser theorems on the incompleteness of arithmetic.

The paper closes with some applications to Ershov's theory of numerations.

Reviewer's historical remark: The author refers on p. 121 to "Curry's S". In fact, S was introduced by M. Schönfinkel [Math. Ann. 92 (1924), 305–316; Jbuch 50, 23], and, although it appeared in Curry's first paper [H. B. Curry, Amer. J. Math. 51 (1929), 363–384; Jbuch 55, 33], Curry stopped using it in his papers and did not really understand the role of S until the spring of 1942, when he found he could use it to define abstraction by induction on the structure of the term, a result he did not publish until after the war [Synthese 7 (1949), 391–399; MR 12, 384]. In this work, he was following a suggestion of J. B. Rosser [J. Symbolic Logic 7 (1942), 18–27; MR 3, 289], which was based on the definition of abstraction by induction on the structure of the term given by A. Church [The calculi of lambda conversion, 12, Ann. of Math. Stud., 6, Princeton Univ. Press, Princeton, NJ, 1941; MR 3, 129]. This work of Rosser and Church did not use S explicitly. Reviewed by J. P. Seldin

Pino Pérez, R. (1991). An extensional partial combinatory algebra based on λ-terms. In *Mathematical foundations of computer science, 1991 (Kazimierz Dolny, 1991)*, pages 387–396. Springer, Berlin.

[MR1135378] This item will not be reviewed individually. For details of the collection in which this item appears see 92g:68008.

Pino Pérez, R. and Even, C. (1994). An abstract property of confluence applied to the study of the lazy partial lambda calculus. In *Logical foundations of computer science (St. Petersburg, 1994)*, pages 278–290. Springer, Berlin.

[MR95k:03017] Summary: "The partial λ-calculus, introduced by E. Moggi, is a formalism well adapted to study the logic of programs in which the evaluation mechanism is call-by-value. In this paper we consider a lazy version of the partial λ-calculus. The main result is the Church-Rosser property for the class of strongly normalizable terms. First, we find sufficient conditions for confluence in an abstract framework. Then, we prove that our calculus satisfies those conditions." For the entire collection see MR 95j:03003.

Asperti, A. and Ciabattoni, A. (1995). Effective applicative structures. In *Category theory and computer science (Cambridge, 1995)*, pages 81–95. Springer, Berlin.

[MR98c:03097] Summary: "An effective applicative structure (EAS) is a collection of partial functions over an arbitrary set M, indexed by elements of the same set, closed under composition, and containing projections, universal functions and the functions S^m_n of the s-m-n theorem of recursion theory. The notion of EAS is developed as an abstract approach to computability, filling a notational gap between functional and combinatorial theories." For the entire collection see MR 98c:68008.

Asperti, A. and Ciabattoni, A. (1996). On completability of partial combinatory algebras. In *Theoretical computer science (Ravello, 1995)*, pages 162–175. World Sci. Publishing, River Edge, NJ.

[MR98d:03067] Summary: "A partial combinatory algebra is said to be completable if it can be extended to a total one. J. W. Klop, in an earlier paper [Bull. European Assoc. Comput. Sci. 16 (1982), 472–482; per bibl.] and in a personal communication, gave a sufficient condition for completability of a PCA$\mathcal{M} = (M, , K, S)$ in the form of ten axioms (inequalities) on the terms of M. We prove that Klop's sufficient condition is equivalent to the existence of an injective s-m-n function over M (which in turn is equivalent to the padding lemma). This is proved by working with an alternative characterization of PCAs recently introduced by the authors (effective applicative structures). As a corollary, we show that nine of Klop's ten axioms are actually redundant (the so-called Barendregt axiom is enough to guarantee completability). Moreover, we prove that any uniformly reflexive structure is completable." For the entire collection see MR 97k:68013. Reviewed by Martin W. Bunder.

Bethke, I. and Klop, J. W. (1996). Collapsing partial combinatory algebras. In *Higher-order algebra, logic, and term rewriting (Paderborn, 1995)*, pages 57–73. Springer, Berlin.

[MR97g:68141] Summary: "Partial combinatory algebras occur regularly in the literature as a framework for an abstract formulation of computation theory or recursion theory. In this paper we develop some general theory concerning homomorphic images (or collapses) of pca's, obtained by identification of elements in a pca. We establish several facts concerning final collapses (maximal identification of elements). 'En passant' we find another example of a pca that cannot be extended to a total one." For the entire collection see MR 97f:68005.

Bethke, I., Klop, J. W., and de Vrijer, R. (1996). Completing partial combinatory algebras with unique head-normal forms. In *11th Annual IEEE Symposium on Logic in Computer Science (New Brunswick, NJ, 1996)*, pages 448–454. IEEE Comput. Soc. Press, Los Alamitos, CA.

[MR98d:68133] Summary: "In this note, we prove that having unique head-normal forms is a sufficient condition for partial combinatory algebras to be completable. As an application, we show that the pca of strongly normalizing CL-terms as well as the pca of natural numbers with partial recursive function application can be extended to total combinatory algebras." For the entire collection see MR 98a:68006.

Asperti, A. and Ciabattoni, A. (1997a). A sufficient condition for completability of partial combinatory algebras. *J. Symbolic Logic*, 62(4):1209–1214.

[MR99j:03055] A partial conbinatory algebra (PCA) $(M, -, K, S)$ is total if ab is defined for all $a, b \in M$. A PCA $(M, -, K, S)$ is completable if there are a total PCA $(M', -', K', S')$ and an injection $\phi\ M \to M'$ such that, for all defined $ab \in M$, $\phi(ab) = \phi(a)\phi(b)$. It is strongly completable if also $\phi(K) = K'$ and $\phi(S) = S'$.

It was known that a PCA is strongly completable if 9 inequalities and the axiom $a, b, c, d \in M$ & $Sab = Scd \to (a = c \wedge b = d)$ hold. The authors show that the axiom alone is sufficient to ensure completability. Reviewed by Martin W. Bunder.

Asperti, A. and Ciabattoni, A. (1997b). A sufficient condition for completability of partial combinatory algebras. *J. Symbolic Logic*, 62(4):1209–1214.

[MR1617969] A review for this item is in process.

Forti, M., Honsell, F., and Lenisa, M. (1997). An axiomatization of partial n-place operations. *Math. Structures Comput. Sci.*, 7(3):283–302.

[MR98b:08003] Summary: "We propose a general theory of partial n-place operations based solely on the primitive notion of the application of a (possibly partial) operation to n objects. This theory is strongly self-descriptive in that the fundamental manipulations of operations, that is, application, composition, abstraction, union, intersection and so on, are themselves internal operations. We give several applications of this theory, including implementations of partial n-ary λ-calculus, and other operation description languages. We investigate the issue of extensionality and give weakly extensional models of the theory."

Akama, Y. (1998a). SN combinators and partial combinatory algebras. In *Rewriting techniques and applications (Tsukuba, 1998)*, pages 302–316. Springer, Berlin.

[MR99f:03011] Summary: "We introduce an intersection typing system for combinatory logic. We prove soundness and completeness for the class of partial combinatory algebras. We show that a term of combinatory logic is typeable iff it is SN. Let $\mathcal{F}$ be the class of nonempty filters which consist of types. Then $\mathcal{F}$ is an extensional nontotal partial combinatory algebra. Furthermore, it is a fully abstract model with respect to the set of SN terms of combinatory logic. By $\mathcal{F}$, we can solve Bethke-Klop's problem: 'find a suitable representation of the finally collapsed partial combinatory algebra of $\mathcal{P}$'. Here, $\mathcal{P}$ is a partial combinatory algebra, and is the set of closed SN terms of combinatory logic modulo the inherent equality. Our solution is the following: the finally collapsed partial combinatory algebra of $\mathcal{P}$ is representable in $\mathcal{F}$. To be more precise, it is isomorphically embeddable in $\mathcal{F}$." For the entire collection see MR 99d:68012.

Akama, Y. (1998b). SN combinators and partial combinatory algebras. *Sūrikaisekikenkyūsho Kōkyūroku*, (1023):92–111. Type theory and its application to computer systems (Japanese) (Kyoto, 1997).
[MR1643752] This item will not be reviewed.

1.5 Illative combinatory logic and set theory

Seldin, J. P. (1976/77). Recent advances in Curry's program. *Rend. Sem. Mat. Univ. Politec. Torino*, 35:77–88 (1978).
[MR80a:03002b] Curry's program consists of extending pure combinatory logic by some "illative" notions in order to provide a basis for parts of mathematics (at least logic). Several resulting systems proposed by Curry, Church and others have turned out to be either inconsistent or inadequate (for logic). In this paper an illative system adequate for predicate logic is presented which is claimed to be consistent. (For the proof, reference is made to an unpublished talk.) A generalization of the given system is also presented; this theory turns out to be inconsistent. The paper ends with the remark that the time may have come to pay more attention to semantics. Some references in this direction are listed, to which may be added a forthcoming paper by P. Aczel ("Frege structures and the notions of proposition, truth and set", Proceedings of the Kleene Symposium, North- Holland, Amsterdam, to appear). Reviewed by Henk Barendregt.

Bridges, D. S. (1978). A note on Morse's lambda-notation in set theory. *Z. Math. Logik Grundlag. Math.*, 24(2):113–114.
[MR80b:03073] In A. P. Morse's set theory (A theory of sets, Academic Press, New York, 1965; MR 34 #39), the statement (**) domain $\lambda x(0 \in vx \rightarrow ux) = \{x \mid vx \wedge ux \in U\}$ can be proved with the help of the law of the excluded middle. The author proves conversely that (**) implies the law of the excluded middle. He also appends an argument for the "constructive" validity of the statement $(A \in U \wedge B \subseteq A \rightarrow B \in U)$. Reviewed by E. Mendelson.

Bunder, M. W. (1978). Equality in $\mathcal{F}_{21}^{*}$ with restricted subjects. *Z. Math. Logik Grundlag. Math.*, 24(2):125–127.
[MR80e:03011] This paper concerns some questions posed in Section 15C3 of the book by H. B. Curry, J. R. Hindley and the reviewer (Combinatory logic, Vol. II, North-Holland, Amsterdam, 1972; Zbl 242 #2029) concerning the addition of a new atomic term Q (to represent equality) and some axioms and rules for it to the system $\mathcal{F}_{21}^{*}$. The version of $\mathcal{F}_{21}^{*}$ considered by the author can be formulated on the basis of a system of λ-calculus or combinatory logic in which there are the (noncombinator) atomic terms Ξ (restricted generality), H (proposition), A (subjects; i.e., those terms of the system which are considered to have a value; this is called Υ in the work of Curry et al. (op. cit.)), E (arbitrary terms), and perhaps some additional canonical atoms (additional predicates) of various degrees. We take $X \supset_u Y$ as an abbreviation for $\Xi(\lambda u.X)(\lambda u.Y)$, F for $\lambda xyz.\Xi x(Byz) = \lambda xyz.xu \supset_u y(zu)$ (so that $FXYZ$ represents the statement that Z is a function from X to Y), H_n for

$FA(FA(\cdots(FAH)\cdots))$, where there are n occurrences of A, and L for $H_1 = FAH$. The axioms are EX for every term X, LE, LA, LH and, if θ is a canonical atom of degree n, $H_n\theta$. The rules are the following: $\Xi XY, XU \vdash YU$; if $\Delta, Xu \vdash Yu$, where Δ is any sequence of terms and u is a variable which does not occur (free) in Δ, X or Y, then $\Delta, LX \vdash \Xi XY$; $X \vdash HX$; and if $X = Y$ then $X \vdash Y$. (Curry, Hindley, and the reviewer (op. cit.) also postulated the rule $LX, LY \vdash H(\Xi XY)$, but the author does not need this one.) The subjects are restricted because there are terms X for which $\not\vdash AX$; the author has shown (J. Symbolic Logic 41 (1976), no. 2, 467 - 468; MR 53 #12889) that if $\vdash AX$ for all terms X then the system is inconsistent. The postulates for Q considered in this paper are the following: (1) $Ex \supset_x Qxx$; (2) $FE(FEH)Q$ (i.e., $Ex \supset_x Ey \supset_y H(Qxy)$); (3) $QXY, X \vdash Y$; and (4) $QXY, UX \vdash UY$. (Note that (3) can be deduced from (4) by putting I in place of U.) The author shows that (1) and (2) are consistent with the postulates for $\mathcal{F}^*_{21}$ by interpreting Q as $BK(KT)$, where T is any theorem. (He could have interpreted Q as any term X with the property that for any terms Y and X, $XYZ = T$.) He then goes on to show that if (1), (2), and either (3) or (4) are adjoined to $\mathcal{F}^*_{21}$, then the system is inconsistent (in the sense that every term becomes provable). He closes by pointing out that the contradiction can be avoided if (2) is replaced by $FA(FAH)Q$ or if the axiom LH is not postulated. (The author is in error in his statement at the end of the paper that the system $\mathcal{F}_{22}$ of Curry (J. Symbolic Logic 38 (1973), 489 - 492; MR 49 #2273) includes properties (1), (2) and (3). In fact, Q is not present in this system, and in the extensions in which Q is present, such as that of the reviewer (Logic Colloquium '73 (Bristol, 1973), pp. 433 - 444, North-Holland, Amsterdam, 1975; MR 52 #2835), (3) is false and (4), if it holds at all, holds only for those U for which $\vdash LU$.) Reviewed by J. P. Seldin.

Bunder, M. W. and Meyer, R. K. (1978). On the inconsistency of systems similar to $\mathcal{F}^*_{21}$. *J. Symbolic Logic*, 43(1):1–2.

[MR80a:03002a] Curry's program consists of extending pure combinatory logic by some "illative" notions in order to provide a basis for parts of mathematics (at least logic). Several resulting systems proposed by Curry, Church and others have turned out to be either inconsistent or inadequate (for logic). This paper reports on the failure of one such attempt to carry out Curry's program. Reviewed by Henk Barendregt.

Kuzičev, A. S. (1978). The Curry paradox in combinatorially complete systems. In *History and methodology of the natural sciences, No. XX (Russian)*, pages 81–98. Moskov. Gos. Univ., Moscow.

[MR81a:03014] The system considered here is one of λ-conversion with additional operation Ξ and Gentzen-type rules corresponding to the interpretation of Ξab as $\forall x(ax \supset bx)$. After imposing complicated restrictions on rules and derivations, the author presents a long proof of cut elimination. (For the entire collection see MR 80d:01002.) Reviewed by G. E. Mints.

Šumihin, V. P. (1978). The inconsistency of the system $\mathcal{F}^{*A}_{21}$ of combinatory logic. *Vestnik Moskov. Univ. Ser. I Mat. Mekh.*, (3):3–8.

[MR80d:03013] Author's summary: "An inconsistency in the systems $\mathcal{F}_{21}^{*A}$, $\mathcal{F}_{21}^{*T}$, $\mathcal{F}_{31}^{*}$ and $\mathcal{F}_{32}^{*}$ (H.B. Curry, J.R. Hindley and J.P. Seldin, Combinatory logic, Vol. II, North- Holland, Amsterdam, 1972; Zbl 242 # 2029) is proved by deriving a paradox. The class of systems in which this paradox is derivable is described."

Bunder, M. W. (1979a). Alternative forms of propositional calculus for a given deduction theorem. *Notre Dame J. Formal Logic*, 20(3):613–619.

[MR80m:03031] It is well known that in a propositional or predicate calculus based on combinatory logic it is necessary to have some restriction on the deduction theorem, in order to avoid Curry's paradox. In previous papers by the author (same journal 14 (1973), 341 - 346; MR 48 #5828; ibid. 15 (1974), 25 - 34; MR 49 #8818), the restricted form of the deduction theorem used was DTP: if $\Delta, X \vdash Y$ then $\Delta, HX \vdash X \supset Y$, where Δ is any sequence of terms and HX stands for "X is a proposition".

This form of the deduction theorem was motivated by the following three-valued tables, of which the table for implication is due to S. C. Kleene (Introduction to metamathematics, Van Nostrand, New York, 1952; MR 14, 525):

X	HX	X	ΓX		$X \supset Y$	T	F	N	$:Y$
T	T	T	F	$X:$	T	T	F	N	
F	T	F	T		F	T	T	T	
N	N	N	N		N	T	N	N	

Here N can stand for "neither T nor F", and Γ (negation) can be defined by $(x).x \supset .Hz \supset_z z$. If this definition for Γ is used, and if $Hz \supset_z z$ is assigned F, then the table for Γ follows from that for $\supset$. The question taken up in this paper is: to what extent are the entries in the third row and third column of the table for $\supset$ uniquely determined by DTP, modus ponens, and the (fairly obvious) rule $H : X \vdash HX$? The author answers the question by showing that the three N entries are in fact uniquely determined by these rules in the sense that in none of these positions can a T or F occur without contradiction. However, either or both of the T entries in the third row and column may be N. Rules are given which force each of these entries to be T; other rules are given which force each to be N. The author closes with some consequences of these results for the tables for $\wedge$ and $\vee$ if these operations are defined as the author has defined them previously (op. cit.; MR 49 #8818) by the following equivalences: $\wedge \equiv (x,y).Hz \supset_z .(x \supset .y \supset z) \supset z$, $\vee \equiv (x,y).Hz \supset_z : (x \supset z) \supset .(y \supset z) \supset z$. (On p. 617, in connection with his formula (18), the author states that if there is an X with $\vdash \Gamma(HX)$, then X has the value N and HX the value F. This seems to the reviewer to contradict the table for H and therefore to show that there is no X for which $\vdash \Gamma(HX)$, but this does not seem to be the author's conclusion. Also, there is a misprint in the author's first reference: "XI (1970), pp. 467 - 470" should be "XIV (1973), pp. 341 - 346".) Reviewed by J. P. Seldin.

Bunder, M. W. (1979b). Generalized restricted generality. *Notre Dame J. Formal Logic*, 20(3):620–624.

[MR80i:03025] H.B. Curry and his students usually base their systems of predicate logic on the restricted generality operator Ξ, instead of the classical $\forall, \supset$. ΞXY says that if XU is significant and true, then YU is significant and true. This is equivalent to $\forall u(Xu \supset Yu)$ in classical logic, but in a logic of "partial predicates", where XU need not be significant for all U, the equivalence is debatable, since if we assume that (i) $\forall uZ$ implies that Z is significant for all values of u and (ii) if $(P \supset Q)$ is significant then so are P and Q, then $\forall u(Xu \supset Yu)$ carries an implication about the significance of Xu and Yu that ΞXY does not. If one rejects (i) or (ii), then in analyzing the behaviour of $\forall$ and $\supset$, one of the interesting cases will be $\forall u(Xu \supset Yu)$, which brings attention back to Ξ as a key concept. The author states that besides Ξ, one needs also to postulate operators ${}^k\Xi^n$ for $k, n \geq 1$, which would correspond in classical logic to $\forall u_1 \cdots u_n((X_1 u_1 \cdots u_n \& \cdots \& X_k u_1 \cdots u_n) \supset Y u_1 \cdots u_n)$. Natural axioms for these operators are given, and a deduction theorem is proved. Reviewed by R. Hindley.

Bunder, M. W. (1979c). λ-elimination in illative combinatory logic. *Notre Dame J. Formal Logic*, 20(3):628–630.

[MR80h:03020] The title refers to rules for the conjunction connective in systems of propositional calculus based on combinatory logic. The author's earlier system (same journal 14 (1973), 341 - 346; MR 48 #5828) allowed the rules only under certain restrictions. Here new axioms are added for the primitive H, which expresses the property of being a proposition, with the result that the restrictions on the conjunction rules may be removed. Reviewed by B. Lercher.

Bunder, M. W. (1979d). On the equivalence of systems of rules and systems of axioms in illative combinatory logic. *Notre Dame J. Formal Logic*, 20(3):603–608.

[MR80h:03019] All "Hilbert-type" systems of illative combinatory logic which have been proposed so far are stronger than the corresponding natural deduction systems. By carefully investigating the sets of axioms of his own two systems (the author, same journal 14 (1973), 341 - 346; MR 48 #5828; "Predicate calculus of arbitrarily high order", Preprint No. 2/79, pp. 1 - 16, Univ. Wollongong, Wollongong, 1979), the author shows how to construct a system of axioms which is equivalent to a system of rules. Reviewed by A. S. Kuzichev.

Bunder, M. W. (1979e). Paraconsistent combinatory logic. *Polish Acad. Sci. Inst. Philos. Sociol. Bull. Sect. Logic*, 8(4):177–181.

[MR81a:03025] N. C. A. da Costa (Notre Dame J. Formal Logic 15 (1974), 497 - 510; MR 50 #6841) presented his paraconsistent logics $\mathcal{C}_n$, $1 \leq n \leq \omega$, which are based on the following conditions: (1) $\neg(A \& \neg A)$ should not in general be valid; (2) From A and $\neg A$ it should not be possible, in general, to deduce an arbitrary formula; (3) each calculus must contain the schemata and deduction rules of the classical propositional calculus that do not interfere with (1) and (2). In this paper the author defines a particular kind of paraconsistent logic in the following way: "A logic L, defined over a set of formulas F, is said to be paraconsistent if its set of theorems S contains X and $\neg X$ for some X but $F \neq S$". Then he shows that illative combinatory logic provides a number of paraconsistent logics that do not follow da

Costa's pattern and which satisfy some of the three da Costa conditions only in a certain sense. Reviewed by Ayda I. Arruda.

Bunder, M. W. (1979f). Scott's models and illative combinatory logic. *Notre Dame J. Formal Logic*, 20(3):609–612.

[MR80j:03022] The author examines how far results in illative combinatory logic remain valid if the properties $(\xi)(X = Y \Rightarrow \lambda u.X = \lambda u.Y)$ and $(\eta)(u$ not free in $X \Rightarrow \lambda u.(Xu) = X)$ are no longer assumed. He sketches how the development of logic and set theory based on combinatory logic (the author, same journal 15 (1974), 25 - 34; MR 49 #8818; ibid. 15 (1974), 192 - 206; MR 49 #8819) can be amended. For the finite formulation given in his earlier paper (ibid. 14 (1973), 341 - 346; MR 48 #5828), however, the equality rule has to be strengthened to an infinite scheme. Finally, some remarks are made about the interpretation of the illative systems in a system of D. Scott (Lambda-calculus and computer science theory (Proc. Sympos., Rome, 1975), pp. 1 - 26, Springer, Berlin, 1975; MR 58 #21489). One of these remarks is puzzling: "The full classical system (the author, op. cit., MR 49 #8818) can be interpreted in the model, while the intuitionistic subsystem cannot." Reviewed by J.W. Klop.

Bunder, M. W. (1979g). Variable binding term operators in λ-calculus. *Notre Dame J. Formal Logic*, 20(4):876–878.

[MR80i:03026] A variable-binding term operator in the sense of J. Corcoran, W. Hatcher and J. Herring (Z. Math. Logik Grundlagen Math. 18 (1972), 177 - 182; MR 46 #5098) is an operator such as Hilbert's epsilon which forms a term from a formula by binding one or more of the formula's variables. The author points out that any such operator may be represented in a lambda-calculus by λ and a nonbinding operator, provided we also add a constant term to represent the set of predicates. (The latter would not be needed for operators which form terms from terms.) The basic idea is due to A. Church (cf. Introduction to mathematical logic, Vol. I, Princeton Univ. Press, Princeton, N.J., 1956; MR 18, 631). The author defines Hilbert's epsilon as an example. Reviewed by R. Hindley.

Kuzičev, A. S. (1979). Classes of objects having normal forms in a system of λ-conversion with logical operators. *Dokl. Akad. Nauk SSSR*, 249(1):41–45.

[MR81h:03031] Author's summary: "The $\mathcal{A}$-system of λ-conversion is studied, and we introduce the concept of a $\lambda\mathcal{Q}$-theory, where $\mathcal{Q}$ is an equality operator. For $\lambda\mathcal{Q}$-theories we prove a cut theorem and a consistency theorem. We construct an example of a $\lambda\mathcal{Q}$-theory containing all translations of derivations of formal arithmetic." The ideas and notation of the author's earlier work (for example, same journal 236 (1977), no. 4, 796 - 799; MR 57 #15983) are used, mostly without explanation here. Results are stated with only the briefest suggestion of a method of proof. (English translation: Soviet Math. Dokl. 20 (1979), no. 6, 1202 - 1207 (1980).) Reviewed by B. Lercher.

Bunder, M. W. (1980a). The naturalness of illative combinatory logic as a

basis for mathematics. In *To H. B. Curry: essays on combinatory logic, lambda calculus and formalism*, pages 55–64. Academic Press, London.
[MR82a:03015] This paper provides a brief account of how mathematics can be formulated within the framework of illative combinatory logic. (For the entire collection see MR 81j:03005.) Reviewed by Garrel Pottinger.

Bunder, M. W. (1980b). Significance and illative combinatory logics. *Notre Dame J. Formal Logic*, 21(2):380–384.
[MR81g:03011] The author discusses several ways of extending the various significance logics S_n and C_n of L. Goddard and R. Routley (The logic of significance and context, Vol. 1, Scottish Academic Press, Edinburgh, 1973) by adding combinators. Reviewed by R. Hindley.

Fitch, F. B. (1980a). A consistent combinatory logic with an inverse to equality. *J. Symbolic Logic*, 45(3):529–543.
[MR82i:03023] In a previous paper [same journal 28 (1963), 87 - 97; MR 31 #1181] the author presented a consistent type-free system of combinatory logic $C\Delta$. In the present paper the author considers a stronger system $C\Gamma$, which also essentially includes the system Q of the author's book [Elements of combinatory logic, Yale Univ. Press, New Haven, Conn., 1974; MR 54 #2429]. In addition to the combinators, the systems $C\Delta$ and $C\Gamma$ contain the logical connectives and quantifiers and include complete arithmetic. In addition, $C\Gamma$ incorporates a description operator, but as this will require special attention at the end of this review, we first outline the construction of $C\Gamma$ when description is left out. The wffs (well-formed formulas) of $C\Gamma$ are expressions built up from the symbols $=, \sim, N, E, \vee, 0, T, B, \sigma$ using the application rule that if c and d are wffs then so is (cd). As usual, $a_1a_2\cdots a_n$ abbreviates $(\cdots(a_1a_2)\cdots a_n)$. An equation is a wff of the form $= cd$. $C\Gamma$ is defined in two stages. In the first stage a system CX is defined for each set of equations X. CX consists of a set PX of wffs called theorems and a set RX of wffs called antitheorems. These two sets are defined by a simultaneous positive elementary inductive definition whose clauses can be summarised as follows: (i) $= abc_1\cdots c_n$ is in PX $[RX]$ if $= ab$ is in X [is not in X]. (ii) $\sim ac_1\cdots c_n$ is in PX $[RX]$ if $ac_1\cdots c_n$ is in RX $[PX]$. (iii) $ec_1\cdots c_n$ is in PX $[RX]$ if $dc_1\cdots c_n$ is in PX $[RX]$ for some [every] d in $\Phi(e)$, whenever e has one of the forms (1) Na, (2) $\vee ab$, (3) Ea, (4) $0ab$, (5) Tab, (6) $Babc$, (7) σbc and in each case $\Phi(e)$ is the set (1) $(= 0a, = (\sigma 0)a, = (\sigma(\sigma 0))a, \cdots)$, (2) (a, b), (3) $(ac$: c any wff$)$, (4) (b), (5) (ba), (6) $(a(bc))$, (7) $(ab(bc))$. A standard proof by induction following these clauses shows that CX is always consistent in the sense that no expression is both a theorem and an antitheorem. The second stage in the definition of Γ involves a nonelementary monotone inductive definition of an impredicative and unusual sort. If X is a set of equations let $a =_X b$ if $= ab$ is in X. Call X a congruence if $=_X$ is an equivalence relation such that $a =_X b$ and $c =_X d$ implies $ac =_X bd$. Also let $a \sim_X b$ if $ac_1\cdots c_n$ is in PX $[RX]$ whenever $bc_1\cdots c_n$ is in PX $[RX]$. Γ is defined to be the smallest congruence such that $= ab$ is in Γ whenever $a \sim_X b$ for all congruences $X \sqsubseteq \Gamma$. Notice that if $a \sim_X b$ for all congruences X then

$a =_\Gamma b$. For example, $\sim\sim a =_\Gamma a$, $\vee aa =_\Gamma a$, $\vee ab =_\Gamma \vee ba$, etc. In the present paper the author seeks to incorporate the symbol 'ι', and a clause for it in the inductive definition of the system CX, so as to obtain, once Γ has been defined as above, $\iota\ (= b) =_\Gamma b$ for every b. (Here '$= a$' denotes the unit class whose only member is denoted by 'a', and 'ι' denotes a function or operator which operates on a unit class to give the only member of that class.) The clause 3.4.10 used by the author may be expressed as follows: "If $a =_X (= b)$ then $\iota a c_1 \cdots c_n$ is in PX $[RX]$ if and only if $bc_1 \cdots c_n$ is in PX $[RX]$". Unfortunately this clause is ambiguous, as a clause in an inductive definition, because $a =_X (= b)$ could hold for many wffs b. The author has suggested (in correspondence) the following reformulation of his clause. "$\iota a c_1 \cdots c_n$ is in PX $[RX]$ if and only if for some [every] b such that $a =_X (= b)$, $bc_1, \ldots, c_n$ is in PX $[RX]$". But as the author has noted, this reformulation no longer allows the proof, for arbitrary b, of $\iota(= b) =_\Gamma b$, but only the weaker $\iota(= b) \sim_\Gamma b$, and in fact the reformulated clause can be proved in the system without 'ι' as primitive if it is defined as '$[x]\exists y[= x(= y)\&y]$'." Reviewed by P. Aczel.

Fitch, F. B. (1980b). An extension of a system of combinatory logic. In *To H. B. Curry: essays on combinatory logic, lambda calculus and formalism*, pages 125–140. Academic Press, London.

[MR82g:03018] The system of combinatory logic referred to in the title is the author's system $C\Gamma$ [J. Symbolic Logic 45 (1980), no. 3, 529 - 543]. The extension involves the introduction of infinitely long well-formed formulas, variables and λ-operators of all finite and infinite degrees. The main goal of the paper is a proof of the consistency of the system. In that respect and many others it follows closely the style of the earlier paper cited above. (For the entire collection see MR 81j:03005.) Reviewed by John Staples.

Kearns, J. T. (1980). Fully explicit deductive systems. In *To H. B. Curry: essays on combinatory logic, lambda calculus and formalism*, pages 35–54. Academic Press, London.

[MR82g:03020] The main aim of this paper is to present the author's view of Curry's programme for the development of combinatory logic, "Curry's program as understood and adapted by Kearns", or CPK. In that view, Curry's programme is to develop and investigate fully explicit deductive systems intended to replace standard systems of logic and mathematics. A fully explicit deductive system, according to the author's introduction, is "a deductive system (1) whose expressions are given by listing or (effectively) describing the basic (simple) expressions, and (optionally) describing unrestricted procedures for obtaining complex expressions. This system is generated by construction processes which take expressions as inputs and yield expressions as results. A 'concrete' instance of a construction process is a construction; the constructions of a fully explicit deductive system include (2) (optionally) constructions generating complex expressions, (3) generalised proofs, and (4) computations. (The two options form an exclusive disjunction.)" (In the reviewer's opinion the definition of fully explicit deductive system just mentioned requires clarification and refinement. A paper such as this which seeks to clarify the work of others should itself achieve a high standard of clarity.) A secondary aim of the paper is to present an approach to paradox resolution which can deal with significant paradoxical sentences. In this approach, truth is "absolute", but falsity is relative to a language. 'False in language L' means incompatible with true sentences of L. The question of whether a negation operator $\neg$ exists which can allow false sentences to be represented by true sentences is treated separately. If the supposition that A is true allows the inference that $\neg A$ is supposed true, and conversely, then both A and $\neg A$ are false and one concludes that there is no such negation operator. This conclusion does not affect the true sentences. (For the entire collection see MR 81j:03005.) Reviewed by John Staples.

Kuzichev, A. S. (1980). Sequential systems of λ-conversion and of combinatory logic. In *To H. B. Curry: essays on combinatory logic, lambda calculus and formalism*, pages 141–155. Academic Press, London.

[MR82b:03041] This paper deals with what Curry calls "illative combinatory logic", and is concerned with systems of logic in the ordinary sense (with connectives and quantifiers) based on combinatory logic (or, what amounts to the same thing, λ-calculus). The main concern of this paper is with systems in which first-order arithmetic can be interpreted. The author begins with his $\mathcal{A}$-system (Dokl. Akad. Nauk SSSR 236 (1977), no. 4, 796 - 799; MR 57 #15983). This system (which is described in some detail in the review just cited) is essentially a sequent calculus without the cut rule but with the other structural rules, the usual rules for introducing the standard connectives and quantifiers in the antecedent and the succedent, rules permitting replacement of any term by one β-convertible to it, rules for introducing Ξab (interpreted as $ax \supset_x bx$) in the antecedent and succedent, and some rules for Q (identity). The author follows Curry in allowing full combinatory completeness. It naturally follows that the cut rule and modus ponens are not admissible. Furthermore, there is a term a such that $\Rightarrow a$ and $\Rightarrow \neg a$ are both provable. And finally, it is possible to prove $\Pi(\lambda x.Nx) \Rightarrow$, where N is $\lambda x.\Xi(\lambda y.\&(yZ_0)(\Pi(\lambda z.P(yz)(y(\sigma z)))))(\lambda t.tx)$, P is implication, Z_0 is KI (the Church numeral for zero), σ is $\lambda xyz.y(xyz)$ (the Church successor function), and Π is the universal quantifier. To deal with these problems, the author defines a restricted class of terms, patterned on the definition of first-order formulas of arithmetic, where the atomic terms are all of the form Qab and Na and where the function symbols include σ and terms for addition, multiplication, and the predecessor function. Some restrictions are then imposed on derivations; they are described in some detail in the review cited above. These restricted derivations are called $\mathcal{F}$-derivations. The author then proceeds with results stated or proved in his previous papers (ibid. 236 (1977), no. 5, 1072 - 1075; MR 58 #5082; Vestnik Moskov. Univ. Ser. I Mat. Meh. 1978, no. 2, 13 - 21; MR 80a:03019a; ibid. Ser. I Mat. Meh. 1978, no. 3, 60 - 66; MR 80a:03019b; Dokl. Akad. Nauk SSSR 238 (1978), no. 2, 269 - 272; MR 58 #10335; ibid. 243 (1978), no. 1, 19 - 21; MR 80b:03090a; ibid. 243 (1978), no. 5, 1123 - 1126; MR 80b:03090b). These are that the class of $\mathcal{F}$-derivations is closed under the cut rule, that there is no term a such that there are $\mathcal{F}$-derivations of both $\Rightarrow a$ and $\Rightarrow \neg a$, that if $\Gamma \Rightarrow \Theta$ in an appropriate sequent calculus for first-order classical arithmetic then there is an $\mathcal{F}$-derivation of $\Pi(\lambda x.Nx)$, $\Gamma \Rightarrow \Theta$, and that there is no $\mathcal{F}$-derivation of $\Pi(\lambda x.Nx) \Rightarrow$. It follows from the last two results that the empty sequent $\Rightarrow$ is not derivable in the sequent calculus for classical first-order arithmetic, and hence that classical first-order arithmetic is consistent. The system of $\mathcal{F}$-derivations is not a conservative extension of the sequent calculus for classical first-order arithmetic; the author therefore presents some further restrictions in order to obtain such a conservative extension. He then closes the paper with a discussion of restricting the terms and derivations of the $\mathcal{A}$-system to correspond with terms, formulas and derivations of ordinary first-order systems in a setting more general than arithmetic. (Reviewer's remarks: Because of Gödel's second incompleteness theorem, the proof of the consistency of arithmetic must involve methods of proof which cannot be formalized within first-order arithmetic. The reviewer is unable to determine what these methods are in the author's proof. The author does say in the introduction that they are "use of the type-free $\mathcal{A}$-system with a restriction on the cut rule (which uses essentially the notion of convertibility and its negation) and the unrestricted principle of combinatorial completeness". It is true that convertibility is not recursive, but it is r.e. and thus representable in first-order arithmetic, and because the system is classical and we have the law of excluded middle, the negation of convertibility is also representable. This needs further clarification from the author.) (For the entire collection see MR 81j:03005.) Reviewed by J. P. Seldin.

Bunder, M. W. (1981). Predicate calculus and naive set theory in pure combinatory logic. *Arch. Math. Logik Grundlag.*, 21(3-4):169–177.

[MR84f:03012] In standard systems of λ-calculus and combinatory logic, the term XY exists for any terms X and Y regardless of the computational effect of applying X to Y. In recursion theory, on the other hand, applying an algorithm to a particular argument may not lead to a value for the computation, for instance when the computation fails to terminate. To a recursion theorist, if applying X to Y leads to such a computation, then the term XY should fail to be defined. N. D. Goodman [J. Symbolic Logic 37 (1972), 225 - 246; MR 48 #3708] gave a system of combinatory logic in which some terms are defined in this sense and others are not, thus coming closer to the recursion-theoretic point of view than is usual in combinatory logic. The author of this paper shows how to interpret first-order logic and most of Zermelo - Fraenkel set theory in Goodman's system. He begins with Goodman's definition of a decidable predicate of k variables, a definition which can be made only in the metatheory, and defines a proposition as a decidable predicate of 0 variables. By making suitable definitions, and by strengthening Goodman's underlying system so that the equality (conversion) is extensional, he shows that the classical first-order predicate calculus can be interpreted in the system. By identifying sets with decidable predicates of one variable, he shows that all of the axioms of Zermelo - Fraenkel set theory can be proved except grounding, choice, and extensionality. The axiom of grounding can actually be proved false. (Reviewer's remark: In general the paper is well written. However, in the last part of the paper, the author refers to the axioms of Zermelo - Fraenkel set theory by numbers of the form ZF n, without presenting either the list of axioms or a reference to it. Both the axiom of infinity and the axiom of choice are called ZF 6; apparently one of them should be ZF 7.) Reviewed by J. P. Seldin.

Fitch, F. B. (1981). The consistency of system Q. *J. Symbolic Logic*, 46(1):67–76.

[MR82g:03019] The author shows that his system Q [Elements of combinatory logic, Yale Univ. Press, New Haven, Conn., 1974; MR 54 #2429], with minor corrections, is consistent and can be strengthened in various ways without destroying its consistency. The minor corrections are that the ζ-rule of cancellation should be excluded from the four subproofs in the rule of extensionality and that the rule of N-elimination, when used in the four subproofs of extensionality, must be restricted to the cases that the contexts indicated as '$(\cdots a \cdots)$' must be Q-expressions of the form '$c = b$', '$\sim t[c = b]$', 'Nb' or '$\sim t(Nb)$' where a occurs in b or c. The proof is based on the author's recent paper [J. Symbolic Logic 45 (1980), no. 3, 529 - 543]. The system given there is modified to a system Q_1 to which the consistency proof given there applies, and it is shown that Q is a subsystem of Q_1. Reviewed by J. P. Seldin.

Quine, W. V. (1981). Predicate functors revisited. *J. Symbolic Logic*, 46(3):649–652.

[MR83e:03028] The author presents a revision of the system of predicate functors he previously introduced [Proc. Amer. Philos. Soc. 104 (1960), 343 - 347; RZhMat 1961:6 A16; Logic and art: essays in honor of Nelson Goodman, pp. 214 - 238, Bobbs-Merrill, New York, 1971]. In this revision, the combinatory functors are separated from the truth-functional ones. The combinatory functors are Inv, inv, Pad, and Ref (one for each n, where n is the number of arguments); they satisfy $(InvF)x_2 \cdots x_n x_1 \equiv Fx_1 \cdots x_n$, $(invF)x_2 x_1 x_3 \cdots x_n \equiv Fx_1 \cdots x_n$, $(PadF)x_0 \cdots x_n \equiv Fx_1 \cdots x_n$, $(RefF)x_2 \cdots x_n \equiv Fx_2 x_2 x_3 \cdots x_n$. The truth functional functors are (for each n) $|$, satisfying $(F \mid\mid G)x_2 \cdots x_n \equiv (x_1)(Fx_1 \cdots x_n \mid Gx_1 \cdots x_n)$, where $|$ is Sheffer's stroke. (The reviewer thinks that more comment is needed on the author's statement that combinatory logic (a la Schönfinkel and Curry) "had the whole of higher set theory as its domain; the present scheme stays within the bounds of predicate logic". H. B. Curry [The critical approach to science and philosophy, pp. 127 - 143, Free Press, New York, 1964] dealt with this question in some detail. The reviewer also thinks it worth remarking that given the current state of what Curry called illative combinatory logic, the exact relationship between combinatory logic and higher set theory seems unclear; see an article by the reviewer [Rend. Sem. Mat. Univ. e Politec. Torino 35 (1976/77), 77 - 88 (1978); MR 80a:03002b].) Reviewed by J. P. Seldin.

Bunder, M. W. (1982). Illative combinatory logic without equality as a primitive predicate. *Notre Dame J. Formal Logic*, 23(1):62–70.

[MR83g:03015] Illative combinatory logic seeks to provide a framework for logic and foundations which has the utmost economy and simplicity. This paper is concerned with systems whose pure combinatory objects include the well-known K and S, and whose illative combinatory objects include an "equality" object Q and a "restricted generality" object Ξ. Assertions involving terms constructed from these, and perhaps other, objects are formed using a predicate '$\vdash$'. However, the usual additional assumption of a separate equality predicate is not made. Axioms and rules of inference are given, on the basis of which the author proves a deduction theorem. Reviewed by John Staples.

Kuzichev, A. S. (1982a). Arithmetically consistent λ-theories. *Dokl. Akad. Nauk SSSR*, 262(4):795–799.

[MR83i:03027] The author's construction of arithmetic based on combinatory logic with unrestricted λ-conversion may conveniently be compared with the familiar construction of arithmetic in the framework of second-order logic. Both are based on Dedekind's definition of the natural numbers: $Na \Leftrightarrow \forall X(X0\&\forall y(Xy \Rightarrow Xy') \Rightarrow Xa)$. An arithmetical derivation of a formula A is transformed first into a (first-order) predicate derivation, say d, of $J \Rightarrow A$, where J is a conjunction of arithmetical (mainly induction) axioms. Then d is replaced by a second-order derivation (say d^2) of A^N, the result of relativization to N of all quantifiers in A. If d was cut-free (and this can be achieved by cut elimination for first-order logic) then d^2 is also essentially cut-free. The author's complicated definition of a derivation in an arithmetical λ-theory seems to follow this line. His previous cut-elimination proofs establishing the consistency of arithmetic aroused some doubts (see J. P. Seldin's remark in his review of the paper by the author [To H. B. Curry: essays on combinatory logic, λ-calculus and formalism, pp. 141 - 155, Academic Press, London, 1980; MR 82b:03041]), since all explicitly presented steps were formalizable in (very weak subsystems of) first-order arithmetic. A proof presented in this paper uses a notion defined by second-order quantification, so one may wonder what is achieved even if it is correct. Only a very general sketch is given. (English translation: Soviet Math. Dokl. 25 (1982), no. 1, 122 - 127.) Reviewed by G. E. Mints.

Kuzichev, A. S. (1982b). A representation of first-order theories in type-free combinatorially complete systems. *Dokl. Akad. Nauk SSSR*, 266(1):23–27.

[MR84c:03032] The results whose proofs are outlined in this important paper yield new methods of constructing axiomatic theories on the basis of lambda-conversion and pure combinatory logic. They are used to prove the consistency of various well-known classical theories. (English translation: Soviet Math. Dokl. 26 (1982), no. 2, 295 - 300 (1983).) Reviewed by M. Greendlinger.

Bunder, M. W. (1983a). A one axiom set theory based on higher order predicate calculus. *Arch. Math. Logik Grundlag.*, 23(3-4):99–107.

[MR86i:03014a] Joint review with (Bunder, 1983c). In these papers the author uses a subsystem of his higher order predicate calculus [same journal 23 (1983), no. 1, 1–10; MR 85a:03015] as a basis for axiomatizing set theory in a way which identifies sets with first-order predicates. In this logic, there are two basic types (although the author does not call them types), A (individuals) and H (propositions), and compound types are of the form $F\alpha\beta$ (functions from α to β). There is universal quantification over each type and there is implication; the restriction on the rule of implication introduction of this natural deduction system is that the antecedent be in type H. The other connectives and quantifiers can be defined in the usual way for second or higher-order logic. Type assignment is not part of the rules for the formation of formulas, but instead is obtained as part of the deductive structure. The entire system is based on a system of combinatory logic or λ-calculus with atomic constants without reduction properties, and thus forms a part of illative combinatory logic in the sense of H. B. Curry and R. Feys [Combinatory logic, Vol. I, North-Holland, Amsterdam, 1958; MR 20 # 817]. As it happens, not all higher types are actually needed; for most parts of the papers only types A, H, and FAH are used, but in some places FAA and $FA(FAH)$ are also needed. The basic idea is to define a set to be anything with type FAH.

In the first paper, the author adds an axiom which says that every individual is a set. He then shows that it is possible to deduce the axioms of pairing, replacement, infinity, power set, and sum set. The axiom of grounding can be shown to be false. The axioms of extensionality and choice can be added. It is inconsistent to add an axiom stating that every set is an individual.

The equality used in stating these axioms is not the usual one for illative combinatory logic which represents convertibility. Instead, it is an extensional equality defined in the obvious way. It has some anomalous properties, since every set X can be proved extensionally equal to $X \cap A$, where A, the type of individuals, can be shown to be a set. It is shown in this paper that there is a set which contains something which cannot be proved to be an individual, and for this set X^* the extensional equality with $X^* \cap A$ might seem to be contradictory. It is not contradictory because in order to infer $V \in A$ from $V \in X^* \cap A$, it is necessary to have a proof that $V \in A$ is a proposition (i.e., has type H), and this proof appears to be lacking.

With regard to the (intensional) equality which represents convertibility, the reflexive property for it (over the type of individuals) is incompatible with the axiom of extensionality.

A further anomaly of the system is that it cannot be proved that every subclass of a set is a set.

In the second paper the author considers a system which is compatible with several properties (including the reflexive property) of the intensional equality and with the axiom that every set is an individual. Since the last axiom is incompatible with the axiom of the first paper (that every individual is a set), that axiom is not postulated. The paper considers a number of postulates for equality, but the main result is that from the axiom that X intensionally equals Y has type H for all X and Y with type A and from the axiom that every set is an individual, it is possible to prove the axioms of comprehension, pairing, replacement, sum set, and a special version of the axiom of power set (which seems to be weaker than the usual version). The axioms of extensionality and choice can be added as in the first paper, and the reflexive law of intensional equality implies that grounding is false. A trivial interpretation shows that many of the properties considered for intensional equality are consistent with the underlying logic.

Reviewer's remarks: (1) The reviewer is surprised that the axiom of comprehension is not mentioned in the first paper. (2) The exposition of the second paper would be improved by the presence of stated theorems summarizing the results. (3) It may be confusing that in several places on pages 101 and 104 of the first paper, '$\sim$' has been printed instead of '$\backsim$'. Reviewed by J. P. Seldin.

Bunder, M. W. (1983b). Predicate calculus of arbitrarily high finite order. *Arch. Math. Logik Grundlag.*, 23(1-2):1–10.

[MR85a:03015] Most mathematical theories are not expressible in the first order predicate calculus. Therefore various attempts have been made to establish convenient calculi of higher (finite) order. The author's approach generalizes earlier systems given by Beth, Church, Montague and Fourman. The basic techniques are from combinatory logic, and special emphasis is put on comprehensiveness. Reviewed by M. M. Richter.

Bunder, M. W. (1983c). Set theory in predicate calculus with equality. *Arch. Math. Logik Grundlag.*, 23(3-4):109–113.

[MR86i:03014b] See (Bunder, 1983a).

Bunder, M. W. (1983d). A weak absolute consistency proof for some systems of illative combinatory logic. *J. Symbolic Logic*, 48(3):771–776.

[MR86b:03021] From the text: "A large number of formal systems based on combinatory logic or λ-calculus have been extended to include first-order predicate calculus. Several of these, however, have been shown to be inconsistent, all, as far as the author knows, in the strong sense that all well-formed formulas (which here include all strings of symbols) are provable. We call the corresponding consistency notion—an arbitrary wff $\perp$ is [not] provable—weak consistency. We say that a system is strongly consistent if no formula and its negation are provable. (For systems that contain H, the class of propositions, strong consistency is equivalent to: not all propositions are provable. For more on consistency notions in illative combinatory logic see another paper by the author [same journal 42 (1977), no. 4, 527–529; MR 58 #16176].)

"Now for some systems, such as that of A. S. Kuzichev [To H. B. Curry essays on combinatory logic, lambda calculus and formalism, 141–145, Academic Press, London, 1980; MR 82b:03041], the strong and weak consistency notions are equivalent, but in the author's systems [Arch. Math. Logik Grundlag. 23 (1983), no. 3, 99–107; ibid. 23 (1983), no. 3, 109–113], which we consider, they are not. Each of these systems is strong enough to have all of ZF set theory, except the axioms of grounding and choice, interpretable in it, and one system of the author [op. cit.; 99–107] can also encompass first-order arithmetic. It therefore seems unlikely that a strong consistency result could be proved for these systems using elementary methods. In this paper, however, we prove the weak consistency of both of these systems by means that could be formulated, at least within the theory of the author [op. cit.; 99–107]. The method also applies to the typed systems of H. B. Curry, J. R. Hindley and Seldin [Combinatory logic, II, North-Holland, Amsterdam, 1972; Zbl 242:02029] and to Seldin's generalised types [Ann. Math. Logic 17 (1979), no. 1-2, 29–59; MR 82a:03017]."

The consistency result proved is sufficiently weak that it does not imply the consistency of the set theory and arithmetic which can be interpreted in the systems to which it applies. This is because all wff's of the first-order systems are interpreted in the latter systems as members of the class H of propositions, and this consistency result is too weak to prove that there is a proposition which is unprovable. This is also the reason that the fact that the proof can be formulated within one of the systems to which it applies does not violate Gödel's second theorem.

Reviewer's remarks: (1) The use of the symbol "$\perp$" for the indeterminate which cannot be proved seems to the reviewer to be a poor choice of notation, since this

symbol usually denotes a false proposition and it is essential to the consistency proof that the indeterminate cannot be proved to be a proposition. (2) It is useful from the point of view of exposition that a definition of a formal proof is given in the paper. Such a definition is lacking in the author's earlier papers. (3) The typed systems to which this proof applies have stronger consistency proofs in the form of cut-elimination theorems. Thus, the main interest of this proof is for the author's system, for which no other consistency proof is known. Reviewed by J. P. Seldin.

Kuzichev, A. S. (1983a). An arithmetically consistent λ-theory. *Z. Math. Logik Grundlag. Math.*, 29(5):385–416.

[MR85d:03026] The author has published more than twenty papers attempting to construct a consistent theory based on the λ-calculus in which arithmetic could be embedded. Most of these papers turned out to contain serious errors; see, e.g., a review by J. P. Seldin [the author, Sequential systems of λ-conversion and of combinatory logic, 141–145, Academic Press, London, 1980; MR 82b:03041]. The paper under review is yet another attempt along the same lines. The author recommends the earlier papers mentioned above for orientation. Reviewed by G. E. Mints.

Kuzichev, A. S. (1983b). Consistency of Quine's system nf. *Dokl. Akad. Nauk SSSR*, 270(3):537–541.

[MR84m:03021] This is a sketch of an argument for the result of the title. The argument is formalisable in third-order arithmetic, so its correctness would contradict Gödel's second incompleteness theorem. (English translation: Soviet Math. Dokl. 27 (1983), no. 3, 633 - 638.) Reviewed by G. E. Mints.

Kuzichev, A. S. (1983c). Set theory in type-free combinatorially complete systems. *Vestnik Moskov. Univ. Ser. I Mat. Mekh.*, (3):36–42.

[MR85b:03021] The author gives a very brief sketch of an embedding of set-theoretic systems (ZF and NF) into a system based on λ-calculus. It is difficult to decide whether this sketch can be completed, especially in view of serious doubts concerning a similar development of formal arithmetic [the author, To H. B. Curry: essays on combinatory logic, lambda calculus and formalism, 141–155, Academic Press, London, 1980; MR 82b:03041]. English translation: Moscow Univ. Math. Bull. 38 (1983), no. 3, 41–49. Reviewed by G. E. Mints.

Bunder, M. W. (1984). Conjunction without conditions in illative combinatory logic. *Polish Acad. Sci. Inst. Philos. Sociol. Bull. Sect. Logic*, 13(4):207–214.

[MR87b:03028] From the text: "We show that when the introduction rule for Ξ of one of our previous papers [Notre Dame J. Formal Logic 14 (1973), 341–346; MR 48 #5828; errata; MR 52 #10377] is replaced by one of two more general forms (which in fact allow a limited form of higher-order logic), a new definition of Λ can be written down which has unrestricted elimination rules as well as all other desirable properties of Λ."

Kuzichev, A. A. (1984). Addition of noncombinatory relations to the calculus of λ-conversion. *Vestnik Moskov. Univ. Ser. I Mat. Mekh.*, (6):34–36, 111.

[MR86h:03018] Author summary (translated from the Russian): "We introduce extensions of the λ-calculus by means of new constants and relations, give definitions of their consistency and correctness, and propose a criterion for correctness of an extension."

Bunder, M. W. (1985). Natural numbers in illative combinatory logic. In *Proceedings of the fifth Latin American symposium on mathematical logic (Bogotá, 1981)*, volume 19, pages 31–42.

[MR88k:03025] Introduction: "We attempt to develop natural numbers using the second-order predicate calculus and the three axioms also used to obtain the set theory of one of our earlier papers [Arch. Math. Logik Grundlag. 23 (1983), no. 3-4, 99–107; MR 86i:03014a]. Of these three axioms, two give us that the natural numbers, as we define them, are elements of the class of individuals A; the other says that all individuals are sets under the definition given earlier [op. cit.]. Besides the induction property and some elementary arithmetical properties of natural numbers, we can prove that the class of all natural numbers forms a set which is a subset of A.

"It has been thought that this theory would be strong enough to contain all of first-order arithmetic, and hence that it would be subject to Gödel's incompleteness results; this, however, seems not be the case. In order to obtain the remaining Peano type axioms for arithmetic, we need to assume a restricted form of substitution of equality and a 'type' for the paradoxical combinator Y." For the entire collection see MR 87h:03007.

Bunder, M. W. (1986/87). Some generalizations to two systems of set theory based on combinatory logic. *Arch. Math. Logik Grundlag.*, 26(1-2):5–12.

[MR89a:03097] InIntroduction: "Elsewhere [same journal 23 (1983), no. 3–4, 99–107; MR 86i:03014a] we showed that just one axiom (that every individual is a set) added to a higher-order logic based on combinatory logic allowed the derivation of a lot of set theory. This axiom and the identification of the notions 'set' and 'first-order predicate ranging over individuals', allowed the proof of the comprehension, pairing, replacement, power set and sum set axioms of set theory, while avoiding the usual paradoxes. In this paper we show that some of these proofs go through, even without the one axiom, for sets defined as 'first-order predicates ranging over a given type U'. All of the proofs go through with the axiom, for a large class of such types. We also wrote a paper [ibid. 23 (1983), no. 3-4, 109–113; MR 86i:03014b] based on the axiom that every set is an individual, as well as various axioms for equality. We correct an error in that paper and show that most of the equality axioms can be proved if equality is defined in one of a number of new ways. Also we show that the work in that paper can be generalised in a similar way to that of our first-cited paper. Next we show that a completely type-free notion of 'set' can be defined which incorporates all the sets of various types; however, virtually no set-theoretic results can be proved. Finally we mention some consistency results now known for these systems."

Bunder, M. W. (1987). Some consistency proofs and a characterization of inconsistency proofs in illative combinatory logic. *J. Symbolic Logic*, 52(1):89–110.

[MR88k:03026] This paper is part of Curry's program, which was described by the reviewer [in To H. B. Curry: essays on combinatory logic, lambda calculus and formalism, 3–33, Academic Press, London, 1980; MR 82j:03013]. It deals in particular with systems obtained by taking combinatory logic or λ-calculus with atomic constants for the logical connectives and quantifiers, taking the usual natural deduction rules for these connectives and quantifiers and an additional rule allowing arbitrary conversion of terms. It is well known that systems formed in this manner with no additional restrictions are inconsistent. The author considers adding additional premises to these rules and considers the form these additional premises must take in order for consistency proofs to become possible (via normalization of deduction in the sense of D. Prawitz [Natural deduction, Almqvist & Wiksell, Stockholm, 1965; MR 33 #1227]). In the case of systems of propositional logic without quantifiers, he shows that a deduction which cannot be normalized must contain a step with a term that has no normal form; he also shows that it must contain a copy of the cut formula and its deduction in the deduction of the minor premise for the elimination rule of which the cut formula is the major premise. He goes on to show that this situation can be prevented provided that one additional premise is added to the rule(s) for introducing implication (and, if it is present, negation), where the premise in question must contain the discharged assumption of the rule in question as a subterm. (The extra premise(s) are simple premise(s) not involving discharged assumptions.) In a system with quantifiers, it is possible to have a nonnormalizable deduction in which there is no step without a normal form in either the deduction or in any deduction to which it reduces. The author shows how to obtain consistency proofs for a system of universal quantifiers over all finite types (over an arbitrary finite set of atomic types which includes the type of propositions), but he is unable (with the methods of this paper) to obtain a consistency proof of a system including both the typed quantifiers and the propositional connectives. Reviewed by J. P. Seldin.

Flagg, R. C. (1987). κ-continuous lattices and comprehension principles for Frege structures. *Ann. Pure Appl. Logic*, 36(1):1–16.

[MR88i:03029] Let κ denote an infinite cardinal. A subset S of a poset X is κ-directed if every subset of S of cardinality less than κ has an upper bound in S. A poset X is κ-inductive if all κ-directed subsets have least upper bounds in X. A map $f\, X \to Y$ between κ-inductive posets is continuous if it preserves κ-directed sups. The κ-inductive posets form a Cartesian closed category. For two elements x and y the way-below relation $x \ll y$ holds if for every κ-directed subset D of X the relation $y \leq \sup D$ implies $x \leq d$ for some $d \in D$. A complete lattice L is said to be κ-continuous if every element x is the sup of the set of elements u with $u \ll x$. A subset U of a poset X is Scott open if $\sup D \in U$ for a κ-directed subset D of X implies $d \in U$ for some $d \in D$. The Scott open subsets of X form a topology for which any family of fewer than κ open subsets has an open intersection. The author calls T_0 topological spaces that have this property κ-spaces. He develops a theory of κ-continuous spaces and injective κ-spaces in complete parallelism to the now conventional theory of continuous lattices. For instance, every κ-continuous lattice can be embedded into a κ-continuous lattice of the type of a Scott domain D_∞. In particular, by the limit construction which produced D_∞ in the first place, the author constructs a domain D_κ which is isomorphic to $D_\kappa \times D_\kappa$ and to the function space $[D_\kappa \to D_\kappa]$ and which contains a copy of $\mathbf{N}$ as a discrete subspace. Due to the particular nature of the topology of this domain, the collection of discrete subspaces has certain closure properties which the author describes. The additional effort which has to go into the limit construction due to the presence of higher ordinals is warranted by the fact that D_κ shows the consistency of a certain type-free logic which suffices for a foundation of non-set-theoretic mathematics. The basis is the system EFL introduced by the author and Myhill, which he extends by adding a schema asserting that all finite types are discrete. Reviewed by K. H. Hofmann.

Kuper, J. (1987). A representation principle for sets and functions. In *Essays on concepts, formalisms, and tools*, pages 163–182. Math. Centrum Centrum Wisk. Inform., Amsterdam.

[MR88k:03014] Summary: "We present a representation principle for sets and functions, essentially meaning that sets and functions do exist in two different ways: as intuitive objects and as mathematical objects. In this paper some aspects of the relationship between these two ways are investigated. The principle has consequences for the concept of λ-calculus model and for the relationship between such models and set theory." For the entire collection see MR 88g:68005.

Beeson, M. J. (1988). Towards a computation system based on set theory. *Theoret. Comput. Sci.*, 60(3):297–340.

[MR90k:68101] Summary: "An axiomatic theory of sets and rules is formulated which permits the use of sets as data structures and allows rules to operate on rules, numbers, or sets. We might call it a 'polymorphic set theory'. Our theory combines the λ-calculus with traditional set theories. A natural set-theoretic model of the theory is constructed, establishing the consistency of the theory and bounding its proof-theoretic strength, and giving in a sense its denotational semantics. Another model, a natural recursion-theoretic model, is constructed, in which only recursive operations from integers to integers are represented, even though the logic can be classical. Some related philosophical considerations on the notions of set, type, and data structure are given in an appendix."

Bunder, M. W. (1988). Arithmetic based on the Church numerals in illative combinatory logic. *Studia Logica*, 47(2):129–143.

[MR90d:03027] This paper presents an important contribution to the development of predicate calculi within systems of illative combinatory logic. In particular, the inclusion of arithmetic within (higher-order) predicate logic is studied. A new definition of equality is presented, which allows the use of Church's numerals and the proofs of the basic properties of a recursion operator. Moreover, all primitive recursive functions can be defined in this logic and their properties can be established.

The paper is clearly written and pleasant to read. Reviewed by Mariangiola Dezani-Ciancaglini.

Cantini, A. (1988). Two impredicative theories of properties and sets. *Z. Math. Logik Grundlag. Math.*, 34(5):403–420.

[MR90b:03075] In this paper theories of properties are studied in which not only properties (intensions) are generated through inductive abstraction processes as is usually done, but also sets (extensions) are circumscribed by means of impredicative definitions. First $(1, 2)$ the theory SP is constructed; it is a sort of liberalized second-order system, where only a limited comprehension schema is available. In SP there are application and λ-abstraction for defining operations, the abstraction operator $\{ : \}$ for forming properties, set variables $X, Y, Z, \cdots$, and general variables $x, y, z, \cdots$, as well as variables $\alpha, \beta, \gamma, \cdots$ which range over set definitions. General variables range over everything and hence $\forall xA \to \forall XA$, $\exists XA \to \exists xA$, $\exists \alpha A \to \exists xA$, etc., hold. In contrast to second-order logic, atoms of the form $X \in Y$, $X = x$, are permitted.

In the extended theory IP, introduced in 3, sets form an initial stock of total properties (henceforth classes), but there are many classes which are not sets and many properties which are not classes. In 4 the author gives an interpretation of IP in suitable models of a theory of partial properties. This gives in turn a consistency proof for the theory IP (and hence for SP). In the appendix (5) a system SP(ε) is sketched. In this system an internal operation ε assigns to every "acceptable" property α its unique extension $\varepsilon(\alpha)$.

It is necessary to remark that both SP and IP are logical theories of dichotomous properties. At the same time dichotomous properties form only a subclass (although a very important one) of all properties that are studied in the theory of abstract properties [the reviewer, in Nonstandard semantic nonclassical logics (Russian) (Moscow, 1985), 99–107, Inst. Philos. Akad. Nauk SSSR, Moscow, 1986; RZhMat 1987:8 A54]. Reviewed by M. S. Burgin.

Bunder, M. W. (1989). Overlapping types in higher order predicate calculus based on combinatory logic. *J. Non-Classical Logic*, 6(1):33–44.

[MR1209355] There will be no review of this item.

Komori, Y. (1989a). Illative combinatory logic based on BCK-logic. *Math. Japon.*, 34(4):585–596.

[MR90i:03017] This paper outlines a system of illative combinatory logic similar to that in the reviewer's book, Set theory based on combinatory logic [Doctoral Dissertation, Univ. Amsterdam, Amsterdam, 1969; MR 49 #8817] and later papers. The main difference is that here the introduction rule for Ξ is restricted to cancel only one hypothesis at a time.

The consistency result that the author proves for his restricted system was already available directly from Theorem 8 of a paper by the reviewer [J. Symbolic Logic 52 (1986), no. 1, 89–110; MR 88k:03026]. The author uses the same methods except that he replaces finite induction by transfinite induction.

The work in this paper on models of this system is of more interest, as no set-theoretic or algebraic model of any system of illative combinatory logic had previously been published.

Though details of some proofs are omitted the author's soundness and completeness results for his "Kripke models" seem to be correct. Reviewed by Martin W. Bunder.

Komori, Y. (1989b). Logic based on combinators. *Polish Acad. Sci. Inst. Philos. Sociol. Bull. Sect. Logic*, 18(3):100–104.

[MR90i:03018] Unrestricted illative combinatory logic is inconsistent. In this paper the author considers a system that includes among its terms types very similar to those of a paper by the reviewer [Arch. Math. Logik Grundlag. 23 (1983), no. 1-2, 1–10; MR 85a:03015]. The author, however, uses two sets of inference rules, one for types and one for all terms. The restrictions on these rules are in some cases too strong and in other cases unclear. It is, for example, possible to prove that if X and Y have the type of propositions then so has $X \supset Y$, only if X and Y are distinct. The notion of "variables of type a" that is used in stating some rules is undefined. Reviewed by Martin W. Bunder.

Myhill, J. and Flagg, B. (1989). A type-free system extending (ZFC). *Ann. Pure Appl. Logic*, 43(1):79–97.

[MR90f:03085] From the introduction: "This is the fourth in a series of papers in which we are attempting to develop a natural, demonstrably consistent type-free foundation for mathematics, based on Frege's 'logical' notion of property or propositional function. The underlying theory for our approach is based on Aczel's idea of (**N**-standard) Frege structures. Frege structures themselves provide a foundation for (Bishop-style) constructive analysis if they are formulated with intuitionistic logic or for (Russian-style) recursive analysis if formulated with classical logic. However, they contain only a small part of classical mathematics. Earlier [in Mathematical logic and theoretical computer science (College Park, MD, 1984–1985), 197–217, Dekker, New York, 1987; MR 89i:03027] we constructed a natural, demonstrably consistent extension of the theory of Frege structures containing classical second-order arithmetic with full comprehension. This extension was based on a comprehension principle for discrete classes. In the present paper we add rules asserting closure properties for the collection of discrete classes. The resulting system extends ZFC. Consistency is proved using the model D_k, for k strongly inaccessible, constructed in a paper by Flagg [Ann. Pure Appl. Logic 36 (1987), no. 1, 1–16; MR 88i:03029]."

Turner, R. (1990). Logics of truth. *Notre Dame J. Formal Logic*, 31(2):308–329.

[MR92b:03014] This paper is about semantic theories of truth for languages which contain their own truth predicate. The author considers three such theories, and compares them with respect to their underlying logics. The theories are all based on λ-calculus. The first two such theories already had explicit logics; the author develops logics of the third for comparison and then proves some properties about it.

The three theories all use the same underlying language: starting with the terms of the $\lambda\beta$-calculus, the author defines an atomic formula to be one of $t = s$ (for equality of terms), $T(t)$ (for the truth of a term), and $F(t)$ (for the falsity of a term). Formulas are then defined in the usual way for classical logic, and models are easily defined from models for the $\lambda\beta$-calculus by interpreting '=' as the equality of the model as a model of λ-calculus, interpreting T and F as functions from the domain of the model into $\{0,1\}$, and using the standard definitions for the connectives and quantifiers. To make the truth predicate T useful, it is necessary to be able to treat wff's as terms; this can be done by a coding, so that for each wff A, ${}^\wedge A$ is a term. The intuitive interpretation of truth is embodied in the Tarski biconditionals $T(A) \leftrightarrow A$ (abbreviating $T({}^\wedge A) \leftrightarrow A$) for all wff's A, but adding this to the classical logic previously obtained is inconsistent. Hence, weaker postulates are needed.

The first theory is the Frege structure and is due jointly to D. Scott [in λ-calculus and computer science theory (Rome, 1975), 1–26, Lecture Notes in Computer Sci., 37, Springer, Berlin, 1975; MR 58 #21489] and P. Aczel [in The Kleene symposium (Madison, WI, 1978), 31–59, North-Holland, Amsterdam, 1980; MR 82e:03045], and is a theory of truth and propositions. Here, $P(t)$ (t is a proposition) is taken to mean $T(t) \vee F(t)$. The axioms for propositions are $P(s=t)$ and the closure axioms (with respect to the connectives and quantifiers) that one would expect. The axioms for truth are $T(s=t) \leftrightarrow s=t$, $\sim (T(A) \& F(A))$, and axioms giving the usual classical truth conditions for the connectives and quantifiers under the right "grammatical conditions", which say when the appropriate terms are propositions. For implication, this axiom is $P(A)\&(T(A) \rightarrow P(B)) \rightarrow (T(A \rightarrow B) \leftrightarrow (T(A) \rightarrow T(B)))$; for the universal quantifier it is $\forall x\, P(A) \rightarrow (T(\forall x A) \leftrightarrow \forall x T(A))$.

Models are defined by starting with the models for the language (as indicated above) and defining two new semantic relations for true and false giving the Kleene (strong) three-valued logic. These are not yet models of Frege structures, but the models of Frege structures are constructed from them by constructing a monotonic transfinite sequence of such models $\mathbf{M}(\alpha)$ and taking as the model of a Frege structure a model $\mathbf{M}^* = \mathbf{M}(\alpha) = \mathbf{M}(\alpha+1)$ (i.e., at a fixed point of the construction).

This model $\mathbf{M}^*$ is also a model of the second theory considered by the author, which is due to S. Kripke [J. Philos. 72 (1975), 690–716], P. C. Gilmore [in Axiomatic set theory (Los Angeles, CA 1967), 147–153, Proc. Sympos. Pure Math., XIII, Part II, Amer. Math. Soc., Providence, RI, 1974; MR 50 #12721] and S. Feferman [J. Symbolic Logic 49 (1984), no. 1, 75–111; MR 85i:03068], which the author calls KFG. It is classical in its external logic but has a residue of the Kleene three-valued logic in the internal logic of its truth predicate. The axioms include, for atomic formulas, $T(t) \leftrightarrow T(T(t)) \leftrightarrow F(F(t))$, $F(t) \leftrightarrow F(T(t)) \leftrightarrow T(F(t))$, $T(t=s) \leftrightarrow t=s$, and $T(t \neq s) \leftrightarrow t \neq s$, which give us the Tarski biconditionals for atomic formulas, $T(A) \rightarrow \sim F(A)$, and the truth-table conditions and standard quantifier rules without the grammatical conditions in the corresponding axioms for a Frege structure. The axioms for a Frege structure can be proved in this theory. Furthermore, if the usual Gentzen natural deduction rules are modified by making each formula the argument of T, and if a premise $P(A)$ is added to each of the rules for implication introduction and negation introduction (where $T(A)$ is the discharged assumption) and classical negation (where $T(\sim A)$ is the discharged assumption), then the resulting rules are all provable in this system. KFG can be viewed as a modal logic with T as the modal operator.

The third theory considered by the author differs from KFG in that the internal logic of its truth predicate is classical (instead of the three-valued internal logic in KFG). It is based on the work of A. Gupta [J. Philos. Logic 11 (1982), no. 1, 1–60; MR 84i:03020a] and H. G. Herzberger [J. Philos. Logic 11 (1982), no. 1, 61–102; MR 83h:03008]. Its models are obtained by a transfinite sequence of models similar to the $\mathbf{M}(\alpha)$ defined for Frege structures except that it is not monotonic. Instead of a fixed point in the construction, what is needed is a stabilization ordinal: an ordinal σ such that, for any element d of the domain, if there is an ordinal beyond which the value of T or F at d is always 1, then this value is 1 at σ. Herzberger [op. cit.]

proved the existence of a stabilization ordinal. A wff of the Gupta-Herzberger logic is considered sound if it is valid at every stabilization ordinal.

The rest of the paper is an analysis of modal logics. The logics considered include a weak deontic logic M, classical deontic logic D, and the logics T, S4, and S5. In none of these cases are all the expected postulates accepted without contradiction, but for those that cannot be accepted as is, approximations can be shown to be acceptable. The author concludes by showing that in the logic of stable truth, the notion of truth is not internally definable. Reviewed by J. P. Seldin.

Holmes, M. R. (1991). Systems of combinatory logic related to Quine's "New Foundations". *Ann. Pure Appl. Logic*, 53(2):103–133.

[**MR92g:03020**] Summary: "Systems TRC and TRCU of illative combinatory logic are introduced and shown to be equivalent in consistency strength and expressive power to Quine's set theory 'New Foundations' (NF) and the fragment NFU+Infinity of NF described by Jensen, respectively. Jensen demonstrated the consistency of NFU + Infinity relative to ZFC; the question of the consistency of NF remains open. TRC and TRCU are presented here as classical first-order theories, although they can be presented as equational theories; they are not constructive."

Kuzichev, A. A. (1991). Church-Rosser property for some extensions of the $\lambda\beta$-reducibility relation. *Z. Math. Logik Grundlag. Math.*, 37(6):547–559.

[**MR95a:03028**] This paper splits into two parts. In the first part the author proves (defining four different classes of terms) the Church-Rosser property for a λ-calculus enriched with constants (including the nonstandard elements of a fixed model of formal arithmetic). The reviewer conjectures that a simpler proof could be directly obtained from the result of Mitschke [H. P. Barendregt, The lambda calculus, North-Holland, Amsterdam, 1981; MR 83b:03016 (Theorem 15.3.3)]. The second part uses the result of the first one to prove that a set of derivations in a given system is a conservative extension of formal arithmetic. Unfortunately, the reviewer could neither completely appreciate this system nor check the proofs, since they depend on papers of the same author written in Russian. In particular, it is not clear where the author uses the system $\mathrm{H}\mathcal{A}^*$ in the proof of Theorem 3.3. Reviewed by Mariangiola Dezani-Ciancaglini.

Bunder, M. W. (1992). Combinatory logic and lambda calculus with classical types. *Logique et Anal. (N.S.)*, 35(137-138):69–79.

[**MR98k:03035**] The author considers a new combinator $U\colon ((a \to f) \to f) \to a$ to represent proofs for classical formulas, where f is a constant proposition to represent falsehood. It is known that any classical formula is represented as a type of a combinator X formed from $I\colon a \to a$, $B\colon (a \to b) \to (c \to a) \to (c \to b)$, $C\colon (a \to b \to c) \to b \to a \to c$, $K\colon a \to b \to a$, $W\colon (a \to a \to b) \to a \to b$ and U. The main theorem of the paper states that the combinator X can be transformed to the form $ZI \cdots IK \cdots KW \cdots WU \cdots U$, where Z is a BC combinator.

It is interesting that the combinator X represents four levels of proof, i.e., proof in linear logic, in BCK logic, in intuitionistic logic and in classical logic. It should be mentioned that the theorem holds not only for U but also other combinators, e.g., $P\colon ((a \to b) \to a) \to a$. However, the theorem would not be interesting for P, because W, K and I can be represented as $W = CB(CB)P$, $K = B(BP)(BC(CI))$ and $I = BP(CW)$. So, the combinator X in the main theorem does not contain a part $K \cdots KW \cdots W$. Reviewed by Sachio Hirokawa.

Grue, K. (1992). Map theory. *Theoret. Comput. Sci.*, 102(1):1–133.

[MR93j:03011] From the summary: "Map theory is a foundation of mathematics based on λ-calculus instead of logic and sets, and thereby fulfills Church's original aim of introducing λ-calculus. Map theory can do anything set theory can do. In particular, all of classical mathematics is contained in map theory. In addition, and contrary to set theory, map theory has unlimited abstraction and contains a computer programming language as a natural subset. This makes map theory more suited to deal with mechanical procedures than set theory. In addition, the unlimited abstraction allows definition, e.g., of the notion of truth and the category of all categories."

Map theory is an alternative to ZFC which relies on λ-calculus. The primitive notion is that of function or map (instead of set) and the primitive operations are those of λ-calculus: abstraction and application.

The language of Map is that of λ-calculus augmented with five individual constants: $\top, \bot, if, \phi, \epsilon$. The first two represent truth and undefinedness and the third is J. McCarthy's conditional, but the strength of the theory depends on the last two: ϕ is a map which codes a well-foundedness predicate Φ and ϵ is a choice function "a la Hilbert" which is linked to well-foundedness. The ideas behind ϕ and ϵ are very elegant, but the axioms are only able to express consequences of them (most are simple and natural; some are quite technical).

Formulas of ZFC are coded by terms of Map and theorems of ZFC by equations of the form $A = \top$ which are provable in Map, so consistency of Map implies consistency of ZFC. Conversely, in every model U of ZFC + SI, where SI asserts the existence of a strongly inaccessible cardinal, a model of the theory Map will be definable.

The paper is divided into three parts. The first one is a clear exposition of the ideas and intuitions behind the theory Map and of its expressive power, and it contains a survey of the other two parts. The second part deals with the precise axiomatisation of Map and develops ZFC and its logical environment within Map. The third one deals with the construction of a model of Map inside a model of ZFC + SI.

The idea of this consistency proof is the following: Let $D = V_\xi$, where ξ is the first inaccessible cardinal of U. By means of a complex ordinal induction the author defines two elements of U, Φ and *app*, such that $\Phi \subset D$ and *app* locally has the intended behaviour. A crucial link between Φ and D is that there will be a surjective function $s\colon \Phi \to D$ such that $s(\top) = \emptyset$ and, for $x \neq \top$, $s(x) = \{s(xy)\colon y \in \Phi\}$, where $xy = app(x, y)$. Thus well-foundedness of D will correspond to well-foundedness of Φ for the binary relation $x < y \iff (y \neq \top \text{ and } \exists z \in \Phi(x = yz))$.

Finally the model M is built around Φ. The construction uses relativisation, Gödel numberings, fixed point theorems for natural but rather complicated functionals, intricate definitions, and finally a quotient construction. But the model M which is thus obtained satisfies all the intuitive properties of Φ and ϵ which have guided the author but are not expressible at the level of axioms. The model M will satisfy any extension Map$^+$ of Map corresponding to an extension ZFC$^+$ of ZFC which is satisfied in D.

Reviewer's remarks: New consistency proofs have recently been found by the author and the reviewer, in the spirit of denotational semantics (paper in preparation). They rely on expansions of "ξ-denotational semantics", where ξ is a regular cardinal (the usual Scott semantics corresponds to $\xi = \omega$). The models which are given satisfy all the intuitive properties of Φ and ϵ which have guided the author, and should lead to more attractive versions of the theory Map. Reviewed by Chantal Berline.

Kamareddine, F. (1992a). λ-terms, logic, determiners and quantifiers. *J. Logic Lang. Inform.*, 1(1):79–103.

[MR95j:03025] The author studies models of a system of logic based on the type-free λ-calculus. After reviewing Curry's paradox to show that type freeness and logic do not mix easily, he presents a constructive version of a system of R. Flagg and J. Myhill ["Notes on a type-free system extending ZFC", privately circulated notes, Ohio State Univ., Columbus, OH and SUNY, Buffalo, NY, 1984; per bibl.], which is, in turn, based on work of D. S. Scott [in λ-calculus and computer science theory (Rome, 1975), 1–26, Springer, Berlin, 1975; MR 58 #21489] and P. Aczel [in The Kleene symposium (Madison, WI, 1978), 31–59, North-Holland, Amsterdam, 1980; MR 82e:03045]. This system is obtained from a system of $\lambda\eta$-calculus with the addition of constants for equality (=), and ($\wedge$), or ($\vee$), implies ($\rightarrow$), the two quantifiers ($\forall$, $\exists$), and the property of being a proposition (H). The logic is a standard natural deduction system for constructive logic with appropriate rules for terms being propositions in which the subformula introduced by disjunction introduction and the discharged assumption in the rule of implication introduction must be propositions, and any provable term is a proposition. Absurdity ($\perp$) is defined, and, as usual, negation is defined in terms of absurdity.

The main purpose of the paper is to develop the logic of this system. ΔP, meaning that P is a property or is a class, is defined as $\forall x H(Px)$. In terms of this definition, it is possible to define intersection, union, complement, and other set-theoretic notions such as the set of functions from one class to another and that of one property being a subproperty of another, and many of their usual properties are proved. However, some of the usual properties fail. To obtain these properties the author defines an intentional application, pred, with the property that $\text{pred}(P, x)$ holds [is a proposition] just when Px holds [is a proposition], and if $\text{pred}(P, a) = \text{pred}(Q, b)$ then $P = Q$ and $a = b$. If pred is used instead of ordinary application in these definitions, then many more of the usual properties hold.

DP (P is determinate) is defined as $\forall x(\text{pred}(P, x) \vee \text{pred}(P, x))$. Properties of determinate properties are proved. These results are then connected with the treatment of quantifiers in natural language due originally to Montague [see Formal philosophy: Selected papers of Richard Montague, edited by R. H. Thomason, Yale Univ. Press, New Haven, CT, 1974] and further developed by K. J. Barwise and R. Cooper [Linguistics and Philos. 4 (1981), no. 2, 159–219].

Reviewer's remarks: The reviewer is slightly surprised that the author does not mention H. B. Curry as the author of the original use of H to mean "is a proposition" and the restriction of the assumptions of implication introduction to propositions. See papers by H. B. Curry [J. Symbolic Logic 38 (1973), 489–492; MR 49 #2273] and J. R. Hindley and the reviewer [Introduction to combinators and λ-calculus, Cambridge Univ. Press, Cambridge, 1986; MR 88j:03009 (Chapter 17)] for additional references. To be completely fair to the author, however, it is important to point out that Scott [op. cit.] has no references, and the only reference in the paper by Aczel [op. cit.] is to a monograph by H. B. Curry, J. R. Hindley and the reviewer [Combinatory logic. Vol. II, North-Holland, Amsterdam, 1972]; the idea of using H this way is mentioned in this book, but the systems of the book in which the idea is used fall short of its use by Scott and Aczel and by Curry himself in his 1973 paper [op. cit.]. Reviewed by J. P. Seldin.

Kamareddine, F. (1992b). Set theory and nominalization. I. *J. Logic Comput.*, 2(5):579–604.

[MR94e:03050] This paper discusses formal theories of nominalization: that is, theories within which concepts—e.g., predicates, open well-formed formulas—correspond to objects. Set theories with an axiom of comprehension form the main class of examples. Applications might include, for instance, analogues of Montague semantics which can deal with the self-referential features of natural language.

Section 1 outlines a naive formalization and discusses the problems, such as Russell's paradox. It also sketches an intuitive notion of model: roughly, a universe U of objects, a collection P of predicates (functions $U \to \{0,1\}$) closed under various logical operations, and an injective function $P \to U$.

Section 2 briefly reviews some formal systems which have been put forward as solutions, such as Quine's NF and Feferman's theories; it is argued that any candidate system must allow self-application. Section 3 reviews possible untyped models for theories of nominalization: namely, Scott domains and P. Aczel's Frege structures [in The Kleene Symposium (Madison, WI, 1978), 31–59, North-Holland, Amsterdam, 1980; MR 82e:03045]. The formal definition of Frege structure is postponed to Part II [J. Logic Comput. 2 (1992), no. 6, 687–707; see the following review]. Reviewed by Wesley Phoa.

Kamareddine, F. (1992c). Set theory and nominalization. II. *J. Logic Comput.*, 2(6):687–707.

[MR94e:03051] This paper proposes Frege structures as a suitable type-free model for nominalization. It begins with the semantic definition of Frege structure, following Aczel (for a formal axiomatic theory, see a book by M. J. Beeson [Foundations of constructive mathematics, Section XVII. 7, Springer, Berlin, 1985; MR 86k:03055]). The author states Aczel's result that any model of the untyped λ-calculus can be extended to a Frege structure, and illustrates this in the case of the graph model $P\omega$.

Section 2 considers and discards Scott domains as a possible model for nominalization. Section 3 shows how Frege structures model nominalization. It describes a type system with subtype polymorphism, admitting self-application, and shows how it avoids Russell's and Curry's paradoxes. Any Frege structure provides an interpretation for this system. Section 4 mentions some alternative theories. For Part I see the preceding review. Reviewed by Wesley Phoa.

Barendregt, H., Bunder, M., and Dekkers, W. (1993). Systems of illative combinatory logic complete for first-order propositional and predicate calculus. *J. Symbolic Logic*, 58(3):769–788.

[MR95f:03020] Combinatory logic, like λ-calculus, originally made its appearance as part of a system of logic in the usual sense, with the usual logical connectives and quantifiers [see H. B. Curry, Amer. J. Math. 52 (1930), 509–536; Amer. J. Math. 52 (1930), 789–834; Jbuch 56, 48; Ann. of Math.(2) 32 (1931), 154–180; Zbl 1, 261; Ann. of Math. (2) 34 (1933), 381–404; Zbl 7, 194; Ann. of Math. (2) 35 (1934), 849–860; Zbl 10, 146; A. Church, Ann. of Math. (2) 33 (1932), 346–366; Zbl 4, 145; Ann. of Math. (2) 34 (1933), 839–864; Zbl 8, 289]. However, S. C. Kleene and J. B.

Rosser [Ann. of Math. (2) 36 (1935), 630–636; Zbl 12, 146] showed that the original systems of Curry and Church were inconsistent, and Curry [J. Symbolic Logic 7 (1942), 115–117; MR 4, 125] later showed that an inconsistency can be derived if the only logical connective is implication with its usual postulates added to either combinatory logic or λ-calculus. (Here, implication is represented by a constant P, so that $\mathrm{P}XY$ represents $X \supset Y$.) Although Church and his students lost interest in using λ-calculus as a basis for logic in this way, Curry did not: he spent the rest of his life looking for a way to base logic on combinatory logic. He originally coined the term "illative combinatory logic" for combinatory logic with extra constants and postulates for logic. (Later this term came to refer to any applied λ-calculus or combinatory logic.)

Curry's original proposals for finding consistent systems of illative combinatory logic [Proc. Nat. Acad. Sci. U.S.A. 28 (1942), 564–569; MR 4, 182] were based on the introduction of a category, represented by a constant H, for being a proposition and restricting some of the rules (especially the rule for introducing implication in the sense of natural deduction rules) so that the discharged assumption has to be in this category. (See also another paper by Curry [J. Symbolic Logic 7 (1942), 49–64; MR 3, 289] and Chapter 17 of the book by J. R. Hindley and J. P. Seldin [Introduction to combinators and λ-calculus, Cambridge Univ. Press, Cambridge, 1986; MR 88j:03009].) He also started looking for other constants to replace P and Π (the universal quantifier). One of these is Ξ, representing restricted generality: ΞXY stands for $(\forall u)(Xu \supset Yu)$. Another is F for what we now call the arrow type constructor: $\mathrm{F}XY$ stands for the "type" $X \to Y$. Curry thought of types as predicates, so that for a term X to be in type H was expressed as "$\mathrm{H}X$", where H is thought of as the predicate and X as the subject. In this framework, $\mathrm{F}XYZ$ stands for $(\forall x)(Xx \supset Y(Zx))$.

This paper deals with a variant of Curry's ideas due to M. Bunder (see his thesis ["Set theory based on combinatory logic", Ph.D. Thesis, Univ. Amsterdam, Amsterdam, 1969; per bibl.] and subsequent papers listed in the paper under review), in which H is defined in terms of a category L, which stands for "is a type"; H is defined to be $\mathrm{L} \circ \mathrm{K}$, where K is $\lambda xy.x$ and $\circ$ is composition of functions, so that $\mathrm{L} \circ \mathrm{K}$ is $\lambda x.\mathrm{L}(\mathrm{K}x)$. As extra logical constants, the authors consider P, Ξ, F, and a generalized type operator G, where $\mathrm{G}XY$ corresponds to the dependent function type usually denoted $(\forall x\colon X)(Yx)$. Four systems of logic are constructed, $\mathcal{I}\mathrm{P}$, $\mathcal{I}\Xi$, $\mathcal{I}\mathrm{F}$, and $\mathcal{I}\mathrm{G}$; each logic consists of the natural deduction rules for the introduction and elimination of the constant of the logic (with a restriction that the discharged premise of the introduction rule be limited by being in H or L as appropriate) and a rule in each giving conditions for a formula beginning with the constant in question to be in either H or L. Each of the logics has a rule for conversion of terms.

These systems are compared with two systems of ordinary logic, PROP, which is the $\supset$-fragment of intuitionistic propositional logic, and PRED, the $\{\supset, \forall\}$-fragment of first-order many-sorted intuitionistic predicate logic. The comparisons are by interpretations of two types: direct, and "propositions-as-types". Thus, PRED is interpreted in $\mathcal{I}\mathrm{P}$ directly and in $\mathcal{I}\mathrm{F}$ by "propositions-as-types", while PRED is interpreted directly in $\mathcal{I}\Xi$ and by "propositions-as-types" in $\mathcal{I}\mathrm{G}$. It is proved that all of these interpretations are sound. It is also shown that the two direct interpretations (of PROP in $\mathcal{I}\mathrm{P}$ and PRED in $\mathcal{I}\Xi$) are complete in the sense that the illative combinatory systems are conservative extensions of the respective ordinary systems interpreted in them. This shows that these illative combinatory systems are all essentially first-order.

The authors also consider briefly a system $\mathcal{I}_0$, due to Bunder, which extends $\mathcal{I}\Xi$ by means of the additional postulates $\Xi\mathrm{IH}$ (which says that any provable term is a proposition), LA for each sort A, and LH. These postulates make it possible to interpret second-order logic. However, so far there is no completeness proof, and the only consistency proof, which is due to Bunder [J. Symbolic Logic 48 (1983), no. 3, 771–776; MR 86b:03021], proves that every provable term has a normal form. Since a contradiction in the usual sense can have a normal form, this consistency proof is very weak.

Reviewer's remarks: 1. Part of the value of this paper is its exposition: the reader will get an excellent idea of the nature of illative combinatory logic and its current status. 2. One misprint: in the list of references, the pages for Curry [1934b] should be "849–860". Reviewed by J. P. Seldin.

Cantini, A. (1993). Extending the first-order theory of combinators with self-referential truth. *J. Symbolic Logic*, 58(2):477–513.

[MR94k:03015] Summary: "The aim of this paper is to introduce a formal system STW of self-referential truth, which extends the classical first-order theory of pure combinators with a truth predicate and certain approximation axioms. STW naturally embodies the mechanisms of general predicate application/abstraction on a par with function application/abstraction; in addition, it allows nontrivial constructions inspired by generalized recursion theory. As a consequence, STW provides a smooth inner model for J. Myhill's systems with levels of implication."

Holmes, M. R. (1993). Systems of combinatory logic related to predicative and "mildly impredicative" fragments of Quine's "New Foundations". *Ann. Pure Appl. Logic*, 59(1):45–53.

[MR93m:03020] Summary: "This paper extends the results of an earlier paper of ours [Ann. Pure Appl. Logic 53 (1991), no. 2, 103–133; MR 92g:03020]. New subsystems of the combinatory logic TRC shown in that paper to be equivalent to NF are introduced; these systems are analogous to subsystems of NF with predicativity restrictions on set comprehension introduced and shown to be consistent by M. Crabbe. For one of these systems, an exact equivalence in consistency strength and expressive power with the analogous subsystem of NF is established."

Lubarsky, R. S. (1993). μ-definable sets of integers. *J. Symbolic Logic*, 58(1):291–313.

[MR95b:03037] The main result of the paper is the following. Let L be the hierarchy of constructible sets. Theorem: A set of integers is Σ_n definable in the μ-calculus iff it is Σ_1 definable over L_α, where α is the least n-reflecting admissible ordinal. Reviewed by Yu. G. Ventsov.

Mo, S. K. (1994). The comprehension principle and its deletion—a complete logical calculus. *Sci. China Ser. A*, 37(5):513–522.

[MR95h:03016] The author thinks that "the comprehension procedure may be replaced in all aspects by the substitution operation and the comprehension principle itself may be completely abolished". Reviewed by G. Asser.

Lipton, J. (1995). Realizability, set theory and term extraction. In *The Curry-Howard isomorphism*, pages 257–364. Academia, Louvain.

[**MR97f:03070**] Set theory is still the universal language of choice for mathematics, although category theory has made significant advances. In this survey the author presents several examples of constructive set theory, all with a constructive context that agrees with Heyting's original axiomatization of intuitionistic logic. Constructive set theories allow for models that may be particularly useful in computer science.

We first meet IZF, a pragmatic constructive analogue of Zermelo-Fraenkel set theory, made popular by Myhill, Friedman, and others. The axioms of IZF are essentially chosen for their ability to give a reasonably strong theory, particularly of Powell ordinals and for the need of IZF to be self-validating, while retaining constructive aspects. These constructive aspects are clearly expressed by Kleene-style realizability interpretations for IZF. For example, the model $V(\omega)$ satisfies Church's thesis for total functions, the uniformity principle, and the statement "every metric space is subcountable". From unramified-forcing models we reach Ω-sets, presheaves, sheaves, and topoi. This puts us in the realm of type theory. But as with models for classical set theory, these notions can be lifted to proper class domains, hence to models of IZF, if the starting topos is sufficiently complete. Finally, Hyland's effective topos is presented as possibly the suitable framework for carrying out recursive mathematics.

A survey cannot be expected to be complete. So, instead, the author chooses to include several pieces of folklore, such as Myhill's derivation of excluded middle from "classical foundation", and Diaconescu's of excluded middle from the axiom of choice. A survey is also sensitive to misprints. There are a few, but almost always the intended meaning is clear. Extensive references to the literature enable the reader to study further the many fascinating aspects of constructive set theory and models. For the entire collection see MR 96h:03041. Reviewed by Wim Ruitenburg.

Aczel, P. (1996). Generalised set theory. In *Logic, language and computation, Vol. 1 (Moraga, CA, 1994)*, pages 1–17. CSLI Publ., Stanford, CA. [**MR98c:03108**]

In this article, the author introduces a generalization of untyped λ-calculus to a context in which terms may be sets and the domain of all possible terms may be a proper class. The larger context of this work is mathematical situation theory, though situation theory is not discussed in the paper except for allusions to it as the intended domain of application. The favored set theory is the modification of ZFC with an "anti-foundation axiom", AFA, popularized by the author (originally proposed by M. Forti and F. Honsell [Ann. Scuola Norm. Sup. Pisa Cl. Sci. (4) 10 (1983), no. 3, 493–522; MR 85f:03054]), and with the further modification that a proper class of atoms (nonsets with no elements) is postulated. This is the theory ZFA used in the recent book by K. J. Barwise and L. S. Moss [Vicious circles, CSLI Publ., Stanford, CA, 1996; MR 98a:03082]; this theory is called GST_0 in the article. This set theory is highly appropriate as a vehicle for a theory of generalized syntax. In fact, the anti-foundation axiom of Aczel was presented in Vicious circles as an assertion about the existence of solutions of infinitary equations; the use of atoms as atomic terms from which infinitary terms can be built by set-theoretical operations is essential to the statement of this form of AFA.

In the present article, the idea of interpreting sets built over a class of atoms as terms built over a class of parameters is further generalized to allow for the construction of abstractions over indexed (set) families of atoms (considered as parameters or variables). A preliminary to this is an abstract theory of substitution appropriate to this context.

A sequence of extended set theories is described: GST_2 adds a notion of substitution for parameters or indexed (set) families of parameters (GST_1 adds the notion of indexed family as a primitive to GST_0, a step which seems dispensible); GST_3 adds a notion of abstraction over parameters or indexed families of parameters; GST_4 adds a notion of application of abstractions, which are thus interpreted as λ-terms, and a proposal for a strong extensionality property for this notion of application. At the end of the paper there is a discussion of the definition and properties of a type of proposition in GST_4 which will appear familiar to a student of illative combinatory logic.

Proofs of the results claimed are not given, but all results claimed or conjectured appear plausible and not at all surprising to a reader familiar with the usual untyped λ-calculus. The reviewer sees no reason to doubt that the tools developed here will be useful in the intended domain of application. For the entire collection see MR 97a:03004. Reviewed by M. Randall Holmes.

Cantini, A. (1996). *Logical frameworks for truth and abstraction.* North-Holland Publishing Co., Amsterdam. An axiomatic study.

[**MR97f:03019**] This book deals with type-free logical systems based on combinatory logic with a truth predicate. While it is highly impressive in the range of material that can be represented in the formal systems developed here, it is interesting that the author, who claims to be influenced by the tradition of illative combinatory logic in the sense of Curry and Fitch, refers to virtually none of this work and uses quite a different approach, which leads to far more complex systems with often quite similar results. A clear advantage of the author's approach is that throughout the book he is able to find models for his formal systems, which are therefore consistent. Consistency proofs in illative combinatory logics, on the other hand, are very rare and cover far less powerful systems.

One of the attractions of combinatory logic is that with only three primitive symbols ('=' and the combinators K and S) and a few simple postulates all recursive functions can be defined. The Curry school's illative combinatory logic, obtained by adding one or two more primitive symbols, such as a subclass relation (Curry's "restricted generality") and a few more postulates, allows the development of higher-order predicate logics, first-order arithmetic, and much of set theory and category theory.

The author's basic system MF^-, developed in Part A, has the combinators, various arithmetical concepts, the connectives and $\wedge$, and the universal quantifier as primitive notions. Equations formed using these form the atomic formulas of a classical predicate calculus which also includes the predicate symbols N (natural numbers) and T (truth). Various additions lead to 59 formal systems, 144 axioms and 312 symbols as given in a very useful, but by no means complete, list at the back of the book!

The aim of the author's study is to distinguish the notions of property and set and predicate application and set-theoretic membership. This contrasts with the Curry-style systems where, at least, property and class and predicate application and class membership are identified. In these systems the paradoxes are avoided because theorems of logic hold only for a class of propositions. Interestingly this class behaves very similarly to the class of propositions defined by the author, provided his T is identified with the identity combinator I (an option that makes MF^- inconsistent).

Another interesting fact is that while both the Curry-style illative combinatory logics and the author's extensions of MF^- are type-free, terms in these systems can be shown to be elements of classes which can be interpreted as types.

The remainder of Part A of the book is taken up with developing a theory of classes and predicates that has many desirable features. Included is a proof that MF^- with a form of induction is stronger than Peano arithmetic.

Part B introduces a conservative extension $PW_C + GID$ of MF^- that includes induction over classes as well as a generalised induction scheme. Separation and reduction principles are proved, as well as an analogue of the Myhill-Shepherdson theorem for operations that are extensional with respect to the predication relation and a boundedness theorem for inductive sets.

In Part C it is shown that the type-free logic of Myhill with multiple levels of implications and that of Aczel and Feferman with definitional equivalence relations can be interpreted in $PW_C + GID$.

Part D concerns a logic TLR which, instead of just T, has various levels of truth T_i. TLR allows the interpretation of the subsystem ATR_0 of second-order arithmetic, which allows the proof of the comparability of well-orderings, the Luzin-Sierpinski theorem, the Gale-Stewart theorem and the Ulm structure theorem for countable reduced abelian p-groups. Systems related to TLR are also discussed.

As in illative combinatory logic, tautologies are valid in MF^- only under conditions involving the class of properties and propositionally can only be determined by the finite application of elementary rules. In Part E the author considers a system VF^- that is not finitely based in this way and contains all of the classical first-order logic with equality. VF^- is shown to be consistent and is related to the systems discussed in Part B. Extensions of VF^- are considered that include truth principles suggested by modal logic.

Finally, applications of the logical theories of Part D to the calculus of constructions and possible future uses in areas such as artificial intelligence and theoretical linguistics are discussed. Reviewed by Martin W. Bunder.

Berline, C. and Grue, K. (1997). A κ-denotational semantics for map theory in zfc + si. *Theoret. Comput. Sci.*, 179(1-2):137–202.

[MR99a:03018] The authors present a new consistency proof for map theory. The second author introduced and axiomatized this equational theory based on lambda-theory [Theoret. Comput. Sci. 102 (1992), no. 1, 1–133; MR 93j:03011]. He gave a syntactical translation of ZFC into map theory and constructed a model of it in set theory, assuming the existence of an inaccessible cardinal σ. The first author reviewed that paper and remarked that they had found a new consistency proof.

The current paper contains a clear introduction to map theory and an appendix listing the axioms and inference rules, so it can be read without reading the original paper. The authors refer to a non-existent index after this appendix and repeat many definitions, perhaps as a result of reordering the presentation.

In order to construct a model of map theory in ZF plus the existence of an inaccessible cardinal, the authors proceed as follows. In terms of a regular cardinal κ, an elementary construction of a partially ordered set $(M, \leq)$ is given (of cardinality κ). A topology is defined on M in terms of $\leq$ and κ. (For $\kappa = \omega$ it is the usual Scott topology; for $\kappa > |M|$ the continuous functions are exactly the monotone ones.) Using this topology, a model of the lambda calculus is constructed.

The terms of map theory are those of untyped λ-calculus, extended with 5 constants. These maps are interpreted as elements of M. In fact, each term with n free variables is interpreted as a continuous function from M^n to M.

The least element of M is the interpretation of the constant $\perp$ and represents undefinedness. A single element of M, incompatible with the rest, is the interpretation of the constant $\top$ and represents truth. The rest of M (proper maps, each representing falsehood) is isomorphic to the set of all continuous functions from M to M (ordered pointwise). This makes it easy to interpret application and λ-abstraction in such a way that β-equal terms have the same interpretation (and that the axioms $\bot B = \bot$ and $\top B = \top$ are satisfied). It also reduces the interpretation of the other constants to proving that certain functions are continuous.

McCarthy's conditional if is axiomatized by $if \top BC = B$, $if(\lambda x A)BC = C$ and $if \bot BC = \bot$. Strict three-valued propositional connectives are defined in terms of $\top$ and if. The third value $\bot$ is provably equal to a fixed point of negation, so map theory can be formulated without this primitive constant.

The constant ϕ is interpreted by the element of M that corresponds to a continuous function that has value $\top$ on some (open) subset Φ and $\bot$ on the rest of M. The elements of Φ represent sets: $\top$ represents the empty set and each other element f of Φ represents the set whose members are represented by fx for some x in Φ. The definition of Φ is complicated. The part of the universe represented by Φ is V_σ.

Finally, Hilbert's epsilon operator ϵ is interpreted using a choice function on Φ and the assumption $\kappa > \sigma$. (For the rest of the proof, $\kappa \geq \sigma$ is sufficient.) The universal and existential quantifier are defined in terms of $\top$, if and ϵ. These represent strict quantification over Φ. (One cannot quantify over all maps in map theory.)

In map theory, each theorem has to be coded as an equality between terms. For example, the axiom $\phi \top = \top$ expresses that $\top$ is a set. Since map theory is not formulated in a natural deduction style, implications must also be coded as equalities.

Satisfaction of all axioms and inference rules of map theory is reduced to the satisfaction of some semantic conditions (which formalize some intuitions behind map theory).

The following properties are proved. The model M clearly satisfies strong quartum non datur: each element is (the interpretation of) $\bot$, $\top$ or a λ-abstraction. The strong induction principle expresses that the elements of Φ represent well-founded sets. The generic closure principle expresses that the elements of Φ are the maps f such that, for each x in some "suitable" superset of Φ, fx is in Φ. Although its direct meaning is not obvious, it gives map theory the power of ZFC.

The paper shows that set theory (with logic) can be based on an extension of λ-calculus. It is not clear to the reviewer what parts of map theory are motivated by set theory to make the interpretation of ZF into map theory sound and what parts are motivated by the necessity to interpret general first order theories. It is still open whether the resulting map theory is strictly stronger than ZFC. Reviewed by Antonius J. C. Hurkens.

Bunder, M. W., Dekkers, W. J. M., and Geuvers, J. H. (1997). Equivalences between illative combinatory logics and pure type systems. *Sūrikaisekikenkyūsho Kōkyūroku*, (1021):119–135. Investigations of nonclassical logics based on sequent calculus and Kripke semantics (Japanese) (Kyoto, 1997).

[MR1643424] This item will not be reviewed.

Dunn, J. M. and Meyer, R. K. (1997). Combinators and structurally free logic. *Log. J. IGPL*, 5(4):505–537 (electronic).

[MR1465611] A review for this item is in process.

Dekkers, W., Bunder, M., and Barendregt, H. (1998a). Completeness of the propositions-as-types interpretation of intuitionistic logic into illative combinatory logic. *J. Symbolic Logic*, 63(3):869–890.
[**MR1649066**] A review for this item is in process.

Dekkers, W., Bunder, M., and Barendregt, H. (1998b). Completeness of two systems of illative combinatory logic for first-order propositional and predicate calculus. *Arch. Math. Logic*, 37(5-6):327–341. Logic Colloquium '95 (Haifa).
[**MR1634277**] A review for this item is in process.

Berline, C. and Grue, K. (1999). Addendum: "A κ-denotational semantics for map theory in ZFC + SI" [Theoret. Comput. Sci. **179** (1997), no. 1-2, 137–202; MR 99a:03018]. *Theoret. Comput. Sci.*, 211(1-2):397–398.
[**MR1663913**] A review for this item is in process.

Lenzi, G. (1999). A nontrivial model of Weydert's SF_3 minus the Leibniz rules. *Bull. Belg. Math. Soc. Simon Stevin*, 6(1):77–90.
[**MR1674703**] A review for this item is in process.

1.6 Subsystems of classical logic (including intuitionistic logic), modal and tense logics

Meredith, D. (1978). Positive logic and λ-constants. *Studia Logica*, 37(3):269–285.
[**MR80c:03019**] This is a contribution to the exploitation of the analogy between the positive propositional algebra of pure implication (abbreviated as ipa) and combinatory logic. The analogy was presented in the book by the reviewer and R. Feys (Combinatory logic, Vol. I, North-Holland, Amsterdam, 1958; MR 20 #817), hereafter cited as (CLg); a related result, published while (CLg) was in progress, was given in the reviewer's paper cited hereafter as (GDT), an abstract of which was published in 1954 (Proceedings of the International Congress of Mathematicians (Amsterdam, 1954), Vol. 2, pp. 399 - 400, Noordhoff, Groningen, 1954). The emphasis of the present paper is on applications of combinatory logic to ipa-theory. In (GDT), the following formula schemes are listed, in Lukasiewicz notation: (PB) $CCqrCCpqCpr$; (PC) $CCpCqrCqCpr$; (PW) $CCpCpqCpq$; (PK) $CpCqp$; and (PI) Cpp. The author considers three kinds of ipa systems: (1) the full ipa or $(BCWK)$ system, in which (PB), (PC), (PW), and (PK) are axiom schemes and (PI) is a consequence; (2) the weak ipa or $(BCWI)$ system, in which (PK) is dropped and replaced by (PI); and (3) the (BCI) system, in which (PB), (PC), and (PI) are the only axiom schemes; in all these cases substitution in the axiom schemes and modus ponens hold. The $(BCWK)$ system is due to D. Hilbert; indeed the axiom schemes appear in his 1922 paper (Abh. Math. Sem. Hamburg. Univ. 1 (1922), 157 - 177; Jbuch 48, 1188). The $(BCWI)$ system is due to A. Church (Kontrolliertes Denken, pp. 22 - 37, Alber, Munich, 1951). The (BCI) system, although treated in

(GDT), is ascribed to C. A. Meredith and A. N. Prior (Notre Dame J. Formal Logic 4 (1963), 171 - 187, especially p. 178; MR 30 #1033), with an acknowledgement in a postscript to S. Jaskowski (Z. Math. Logik Grundlag. Math. 9 (1963), 219 - 228; MR 27 #2397). In each of these cases the author first gives a characterization by "applicative expressions", which are combinations formed by application as in combinatory logic from ipa formulas as atoms. These serve as codifications of proof trees by modus ponens in which the major premise, i.e., the CPQ, of a modus ponens inference is always on the left, while the minor premise, i.e. the P, is on the right. If A is such an application expression, its atomic formulas are the top nodes of the tree in the order and with the multiplicity in which they appear from left to right; while if A_1 and A_2 are subtrees, $(A_1 A_2)$ is the subtree obtained from A_1 and A_2 by taking the end formula (called the result) of A_1 as major premise and that of A_2 as minor premise for a final modus ponens inference. Thus each of the three ipa systems is a set of formulas closed with respect to modus ponens and related to applicative expressions of specified types in certain specified ways. As a consequence, the author gives decision procedures for each of the three systems which appear to be simpler than those previously known. There follow a number of technical details concerning the correspondence between ipa formulas and systems and formulas and systems of combinatory logic. This correspondence is not one-to-one, and thus involves some complexity. Relations to the generalized detachment operator of C. A. Meredith are also discussed. At the end there is a discussion of modus ponens formulas, i.e., formulas obtained directly from the corresponding application expressions without supplementary uses of modus ponens – corresponding to the "direkt identische Implicationsformel" of Hilbert and Bernays, including an axiomatization of the set of such formulas. Reviewed by H. B. Curry.

Bartnick, J. (1980). *Prädikatenlogik ohne gebundene Variablen.* Verlag Peter D. Lang, Frankfurt. With a foreword by Dieter Rödding.

[MR82k:03010] Es gibt verschiedene Möglichkeiten, die Prädikatenlogik algebraisch-kombinatorisch zu behandeln, etwa mittels Zylinderalgebren, polyadischer Algebren, über die kombinatorische Logik oder schliesslich die "predicate functors". In der vorliegenden Schrift geht der Verfasser von dem Klassenbildungstheorem der Mengenlehre als Vorbild aus, welches besagt, dass sich jeder Klassenterm mittels endlich vieler Operationen aus endlich vielen Grundklassen gewinnen lässt. Für die Prädikatenlogik werden hier jedoch abzählbar viele Operationen benötigt. Für die Prädikatenlogik mit und ohne Gleichheit wird dann jeweils ein Gleichungs-kalkül erstellt sowie eine Übersetzung der Formeln in denselben. Der Vollständigkeitssatz besagt dann, dass gültige Formeln ableitbaren Gleichungen entsprechen. Die eigentliche Besonderheit ist dabei der Komprehensionsoperator des Verfassers. Lesenswert wird die Schrift auch durch die vielfachen Querverbindungen, die der Verfasser zu den eingangs erwähnten alternativen Ansätzen zieht. Reviewed by M. M. Richter.

Bunder, M. W. (1980). Quantified relevance logic and generalised restricted generality. *Logique et Anal. (N.S.)*, 23(90-91):319–321.

[MR84i:03036] It is shown how systems of relevance logic can be modified so that in such formulas as $\forall x \forall y[(A \rightarrow B) \rightarrow ((B \rightarrow C) \rightarrow (A \rightarrow C))]$ quantification can be restricted to those values of x and y for which the antecedent $(A \rightarrow B)$ holds, in accord with the intentions of relevance logic. A system of combinatory logic with the generalized restricted generality ${}^{k}\Xi^{n}$ [the author, Notre Dame J. Formal Logic 20 (1979), 620 - 624; MR 80i:03025] is used.

Bunder, M. W. (1981). Simpler axioms for BCK algebras and the connection between the axioms and the combinators $\mathbf{B}, \mathbf{C}$ and $\mathbf{K}$. *Math. Japon.*, 26(4):415–418.

[MR83f:03069] Author's summary: "We give a simple axiomatization of the BCK algebras introduced by K. Iseki and S. Tanaka [same journal 23 (1978/79), no. 1, 1 - 26; MR 80a:03081], in terms of axioms directly related to the combinators B, C and K. We also provide three axiomatizations of the positive implicative BCK-algebras introduced by Iseki and Tanaka. Two of these also have axioms directly related to combinators."

Kuzichev, A. S. (1982). Axiomatic theories in combinatorially complete systems. *Dokl. Akad. Nauk SSSR*, 264(3):538–542.

[MR83k:03020] The author defines a large class of sequential systems, each of which is proven to be absolutely consistent. For each of these systems, a class C of deductions is defined and shown to be internally consistent. Moreover, if the sequences $\Delta \Rightarrow \Lambda, a$ and $a, \Gamma \Rightarrow \Theta$ are deducible in C, then so is the sequence $\Delta, \Gamma \Rightarrow \Lambda, \Theta$. The author defines a Hilbert-like first-order predicate calculus and a combinatory calculus. They are both shown to be internally consistent. (English translation: Soviet Math. Dokl. 25 (1982), no. 3, 671 - 675.) Reviewed by M. Greendlinger.

Rezus, A. (1982). On a theorem of Tarski. *Libertas Math.*, 2:63–97.

[MR84b:03019] The theorem of the title was asserted as Theorem 8 in a paper by J. Lukasiewicz and A. Tarski [see Tarski, Logic, semantics and metamathematics, English translation, Clarendon Press, Oxford, 1956; MR 17, 1171]. It states: Every finitely axiomatizable system of propositional calculus with the rules of substitution and detachment in which the two formulas $CpCqp$ and $CpCqCCpCqrr$ are provable can be axiomatized by a single formula. According to the author, no proof of Tarski's theorem had been published prior to the present paper. The proof uses the formulas-as-types idea of H. B. Curry [Curry and R. Feys, Combinatory logic, Vol I, Theorem 9E, North-Holland, Amsterdam, 1958; MR 20 #817] and the condensed detachment operator of C. A. Meredith [see, e.g., Meredith and A. N. Prior, Notre Dame J. Formal Logic 4 (1963), 171 - 187; MR 30 #1033]. Specifically, if α and β are formulas of propositional calculus, and there is a substitution yielding α' and β' where $alpha'$ is $C\beta'\gamma$, then the value of the metaformula $(\alpha\beta)$ is the most general form of any such γ, whence we write $(\alpha\beta) = \gamma$. The metaformulas constructed from propositional calculus formulas as atoms by the operation of application thus encode proofs of the formulas which are their values. For example, let c and k abbreviate $CCpCqrCqCpr$ and $CpCqp$, respectively. Then for any α, $(ck)\alpha = Cpp$, and, whenever α, β, and

γ have the requisite forms to give values to the metaformulas, $((C\alpha)\beta)\gamma = (\alpha\gamma)\beta$. These examples suggest an analogy with the theory of combinators, and in fact the equality of metaformulas is largely analogous to combinatory equality, which the author exploits to obtain a proof of Tarski's theorem. Noting that $CpCqp$ is not valid in systems of relevant logic, the author modifies and refines his results to obtain single axioms for the purely implicational fragments of various relevant logics. He states as an open problem the existence of single axioms for relevant logics which include conjunction or disjunction. In an appendix he shows that every closed λ-calculus term is equal to an applicative combination of the single term $\lambda xyz.y(\lambda u.z)(xz)$. Reviewed by B. Lercher.

Shen, B. Y. (1983). A new system of axioms for BCI-algebras, connected with the combinators b, c, and i. *Nanjing Daxue Xuebao Ziran Kexue Ban*, 1983(4):616–619.

[MR85j:03115] Author's summary: "In 1966, K. Iseki introduced the notion of a BCI-algebra. We show that a BCI-algebra can be characterized by axioms that correspond in a natural way to the combinators B, C and I introduced by H. B. Curry and R. Feys [Combinatory logic, Vol. 1, see p. 264, North-Holland, Amsterdam, 1958; MR 20 #817]."

Buszkowski, W. (1984). A note on Lambek-van Benthem calculus. *Polish Acad. Sci. Inst. Philos. Sociol. Bull. Sect. Logic*, 13(1):31–37.

[MR85m:03009] Author's introduction: "J. F. A. K. van Benthem [The semantics of variety in categorial grammar, Simon Fraser Univ., Burnaby, B.C., 1983; per bibl.] introduced a variant of the Lambek syntactic calculus (LSC) proposed by J. Lambek [Amer. Math. Monthly 65 (1958), 154–170; MR 21 #4904], which we call the variant Lambek-van Benthem calculus (LBC). As proved by van Benthem, LBC is complete with respect to a semantics of λ-terms. In this note we indicate other relevant properties of LBC (decidability, generative power, completeness with respect to a standard algebraic semantics), supporting some expectations of van Benthem."

Bunder, M. W. and Meyer, R. K. (1985). A result for combinators, BCK logics and BCK algebras. *Logique et Anal. (N.S.)*, 28(109):33–40.

[MR87d:03032] BCK-combinators are combinations of three atomic combinators B, C, K, with reduction axiom schemes $\mathrm{B}XYZ \triangleright X(YZ)$, $\mathrm{C}XYZ \triangleright XZY$, and $\mathrm{K}XY \triangleright X$. They correspond to closed λ-terms in which each variable occurs only once (not counting occurrences just after a λ; e.g., only the second x is counted in $\lambda x.x$).

The authors prove in Theorem 1 that every BCK-combinator is stratified (i.e., can be assigned a type-scheme in the system of H. B. Curry [Dialectica 23 (1969), no. 2, 83–92]).

They then turn to BCK-logic. This is a pure implicational propositional calculus with three axiom schemes corresponding to the type schemes for B, C and K, and with modus ponens as its only rule. In Theorem 2, the authors claim that if $\vdash (A \supset B)$ and $\vdash C$ in this system, then there are substitutions σ, τ such that $\sigma(A) = \tau(C)$ and hence $\sigma(B)$ can be derived by modus ponens. This claim (and a corresponding one about BCK-algebras) is now known to be an error (according to a letter to the authors from A. Wronski, mentioned to the reviewer by the first author). Wronski's

counterexample is the pair of formulae $((a \supset a) \supset a) \supset a, b \supset b$. These are theorems of BCK-logic, being type schemes of CII and I (where I $\equiv$ CKK), but $(a \supset a) \supset a$ and $b \supset b$ have no common substitution-instances.

The authors also give an application of Theorem 1 to the "fool's model" of combinatory logic. As their name for it suggests, this is not really a model at all; but it has some of the features of the Engeler-Plotkin model D_A [E. Engeler, Algebra Universalis 13 (1981), no. 3, 389–392; MR 83f:03012]. Reviewed by R. Hindley.

Buszkowski, W. (1987). Lambda-semantics for categorical grammar. In *Essays on philosophy and logic (Cracow, 1984)*, pages 75–84. Uniw. Jagiellonski, Kraków.

[**MR925639**] This item will not be reviewed individually. For details of the collection in which this item appears see 88j:03002.

Komori, Y. (1987). BCK algebras and lambda calculus. In *Proceedings of the 10th symposium on semigroups (Sakado, 1986)*, pages 5–11, Sakado. Jōsai Univ.

[**MR88g:03024**] This paper is about BCK-logic, the implicational logic whose axiom schemes are the principal type-schemes of the combinators B, C and K, and whose only inference rule is modus ponens. This logic's theorems are exactly the type-schemes of closed λ-terms in which each variable occurs only once. (λ-terms in which each variable occurs only once are called BCK-λ-terms in this paper, and have sometimes been called linear λ-terms.)

The author summarises some basic facts about BCK-logic and BCK-λ-terms, and states some open problems on the topic of minimality. (In a formula system of implicational logic, a theorem α is said to be minimal if α is not a nontrivial substitution instance of another theorem.)

For example, he asks: (1) If an implicational formula α is a minimal theorem of LJ (intuitionistic logic), is α minimal also in classical logic? (2) If α is a theorem of BCK-logic, is it true that α is minimal in BCK-logic if and only if α is minimal in LJ? (3) If α is minimal in LJ, is the combinator whose type-scheme is α unique (modulo some reasonable conversion)? (4) Same as Question 3 but for BCK-logic. For the entire collection see MR 87j:20002. Reviewed by R. Hindley.

van Benthem, J. (1987). Categorial grammar and lambda calculus. In *Mathematical logic and its applications (Druzhba, 1986)*, pages 39–60. Plenum, New York.

[**MR89i:03026**] This paper aims to survey the connections between research in type theory and lambda calculus "exploiting analysis between categorical grammars, Gentzen calculi for implication and fragments of typed lambda languages".

The author, however, neglects to mention Curry, one of the originators and, with Church, the main developer of applicative systems. Curry was the first to assign types to applicative terms and the first, in 1934, to notice the analogy between types and logical derivations (now known as the formulas-as-types isomorphism). The author attributes this to Lambek (1958). Curry was probably also the first to use types as grammatical categories.

The paper is clear in its description of various categorical calculi, but not on what is meant by these "representing a mechanism for recognising natural languages". He shows, for example, that

in an applicative system, if z has type (a,b) then Bz (where $B = \lambda uvw.u(vw)$) has type $((c,a),(c,b))$. As z and Bz are distinct, it is not clear how this reflects his "Geach rule": expressions occurring in type (a,b) can also occur in type $((c,a),(c,b))$.

Still, the paper raises many interesting questions, some of which, such as the problem of higher-order quantification, can find their solution in a study of illative combinatory logic. There is also a clear connection between Lambek's subsystem of the lambda calculus and the BCI (linear) logics that are of current interest. For the entire collection see MR 89b:03004. Reviewed by Martin W. Bunder.

Bunder, M. W. (1988). Corrections to some results for BCK logics and algebras. *Logique et Anal. (N.S.)*, 31(121-122):115–122.

[MR93d:03036] Elsewhere [87d:03032Logique et Anal. (N.S.) 28 (1985), no. 109, 33–40; MR 87d:03032] the author and R. K. Meyer claimed the following "result" for BCK logic: If $A \supset B$ and C are theorems of BCK logic, then there are substitution instances $A_1 \supset B_1$ of $A \supset B$ and C_1 of C such that $A_1 = C_1$. B_1 is therefore a theorem of BCK logic.

However, A. Wronski has pointed out that the "result" fails for BCK logic (and algebra), for it must fail in any logic in which $A \supset A$ is provable for arbitrary A. In this note, a weaker version of BCK logic which does satisfy the above property is given. The system is in fact one considered by C. A. Meredith, and the result confirms a conjecture of his dating back to 1954. The theorem for BCK combinators on which the result is based states that every BCK combinator has a principal functional character or principal type scheme. A simpler direct (and new) proof of this theorem is also given here. Reviewed by Djordje Vukomanovic.

Lambek, J. (1989). On a connection between algebra, logic and linguistics. *Diagrammes*, 22:59–75. Journées E.L.I.T. (Esquisses, Logique et Informatique Théorique), Vol. 1 (Paris, 1988).

[MR91m:03030] This paper is concerned with the relations between context-free grammars, sequent calculus and multilinear algebra. A context-free grammar is defined as a set of derivations $f\colon A_1 \cdots A_n \to A$ (where the A_i's are syntactic types) containing an identity $1_A\colon A_n \to A$ and a cut rule: If $f\colon \Lambda \to A$ and $g\colon \Gamma A \Delta \to B$ are derivations, then $g\langle f\rangle\colon \Gamma\Lambda\Delta \to b$ is also a derivation. An internal language is introduced by adjoining variables for each type in order to express identity of derivations. Comparisons between derivations in linguistics, deductions in logic, forms in linear algebra and operations in universal algebra are established, and Gentzen's structural rules of interchange, contraction and thinning are considered. Montague semantics, linear logic, λ-calculus, etc., are characterized as forbidding or permitting these rules. The introduction of specific type-forming operations is then considered and two questions are raised about freely generated categorial grammars: (1) the calculation of the set of all sequents $f\colon \Gamma \to A$, and (2) the decision problem for identity of two sequents. An application to natural language is provided and the question of recursivity of the set of English sentences is raised. Reviewed by Francois Lepage.

Rabinov, A. (1989). First order theories of quantification. In *EPIA 89 (Lisbon, 1989)*, pages 51–62. Springer, Berlin.

[MR91b:68095] Summary: "Many applications of logic to artificial intelligence require that propositions be reified, i.e., made elements of the domain of discourse. This is needed, in particular, in formalizing reasoning about knowledge and about preconditions of actions. Difficulties arise when the quantificational structure of propositions is essential. We show how ideas from combinatory logic can be used to deal with this problem. This approach allows us to construct first-order theories in which lambda-abstraction and quantification can be easily expressed as terms of the language." For the entire collection see MR 90h:68009.

Simons, P. (1989). Combinators and categorial grammar. *Notre Dame J. Formal Logic*, 30(2):241–261.

[MR91j:03029] The starting point of this paper is a criticism of the usual treatment of quantification in the classical theory of categorial grammars. For example, according to the analysis given by Ajdukiewicz, the quantifier expression in a formula of the form $\forall x\, \varphi(x)$ is "$\forall x$", and there is no way to decompose this phrase further by assigning a suitable grammatical category to the symbol "$\forall$". It is well known that this problem can be solved by translating the object language into a suitable system of combinatory logic, in which the quantifier becomes an object of type $(\iota \to \mathrm{o}) \to \mathrm{o}$ where ι and o represent a type of "individuals" and a type of "propositions", respectively. This solution has the advantage of being applicable to quantifiers over objects of all finite orders, but it does not work when one wants to have many-place operation symbols in the language.

In this paper a system of combinators is introduced which permits the elimination of bound variables (of all finite orders) while preserving the compositionality of categorial grammar and allowing for many-place function symbols. The author applies his analysis to languages patterned after those introduced by the Polish logician Lesniewski (but the method can be adapted to other situations as well). Here the categories are constructed from basic ones (for instance, the categories s of sentences and n of names) by forming, for a category φ and a sequence of categories $\alpha_1, \cdots, \alpha_n$ the functor category $\varphi(\alpha_1, \cdots, \alpha_n)$ of functors taking arguments of categories $\alpha_1, \cdots, \alpha_n$ and giving a result of category φ. Categories are then used to specify the set of well formed expressions (wfes) in any category. The following combinators are introduced (assuming f, g, a_i and b_j of the appropriate category, for $1 \le i \le m$ and $1 \le j \le n$):

$$\begin{aligned} I(a) &= a, \\ W(f)(a_1 \cdots a_n) &= f(a_1 \cdots a_n a_n), \\ R(f)(a_2 \cdots a_n a_1) &= f(a_1 \cdots a_n), \\ K(f)(a_1 \cdots a_m b_1)(b_2 \cdots b_n) &= f(a_1 \cdots a_m)(b_1 \cdots b_n), \\ J(f)(a_1 \cdots a_{m-1})(a_m b_1 \cdots b_n) &= f(a_1 \cdots a_m)(b_1 \cdots b_n), \end{aligned}$$

$$B(g)(f_1 \cdots f_m)(a_{11} \cdots a_{1n_1} \cdots a_{m1} \cdots a_{mn_m}) = \\ g(f_1(a_{11} \cdots a_{1n_1}) \cdots f_m(a_{m1} \cdots a_{mn_m})), \quad (1)$$

and $U(f) = \forall a_1 \cdots a_n . f(a_1 \cdots a_n)$, where, in the last clause, f is assumed to be a functor of category $s(\alpha_1, \cdots, \alpha_n)$ and each a_i is a variable of category α_i. The main theorem then states that every well formed expression e in which there is no occurrence of combinators is equivalent to a well formed expression e' in which neither quantifiers nor bound variables of e occur. In the present context, this result is the counterpart of functional completeness results of combinatory logic [see, e.g., J. R. Hindley and J. P. Seldin, Introduction to combinators and λ-calculus, Cambridge Univ. Press, Cambridge, 1986; MR 88j:03009].

Combinators themselves do not belong to any single category, but rather are "systematically ambiguous". A discussion is included on the possibility of replacing the combinators introduced here with families of fixed category combinators. This is completely parallel to what happens in systems of typed combinatory logic (see the reference above for a description of some of them), where there is usually the possibility of choosing between a formulation which makes use of type schemes and one in which each term has exactly one type.

Finally, the author discusses how combinators may be of use also in clarifying the rules of type change, such as Geach's rule: (GR) Any expression with category $\gamma(\varphi)$ may be assigned category $\gamma(\alpha)(\varphi(\alpha))$, for any α, or Montague's rule: (MR) Any expression with category α may be assigned category $\varphi(\varphi(\alpha))$, for any φ.

In these cases combinators may be defined that act like "coercers" from values of one category to values of another category.

The introduction of combinators for situations in which one does not want to reduce all functions to unary ones, carried out in this paper, is interesting, and the paper is written very clearly. It contains several suggestions which are likely to lead to interesting further developments (see, for example, the discussion just mentioned on combinators and rules for type change), and a more formal treatment of them seems to be both possible and desirable. Reviewed by F. Cardone.

Hindley, J. R. and Meredith, D. (1990). Principal type-schemes and condensed detachment. *J. Symbolic Logic*, 55(1):90–105.

[MR91b:03020] This paper relates implicational propositional logic (with implication as the only connective) and type assignments in combinatory logic. Condensed detachment is a rule that combines substitution and modus ponens in implicational logic. Roughly, from formulas $A \to B$ and C such that there are substitution instances of A and C that coincide, condensed detachment allows the inference of the most general corresponding instance of B. The main question of the paper is: when is condensed detachment complete for various finite sets of axioms? i.e., when does condensed detachment as the rule suffice to derive all consequences of the axioms derivable by ordinary substitution and modus ponens? The authors show that condensed detachment corresponds exactly to a computation of the most general type assignment for combinators, and then use that correspondence to show that condensed detachment is complete for the classical and the intuitionistic implicational logics, but not complete for some weaker ones. The exposition of the paper is admirably clear. Reviewed by B. Lercher

Meyer, R. K., Bunder, M. W., and Powers, L. (1991). Implementing the "fool's model" of combinatory logic. *J. Automat. Reason.*, 7(4):597–630. Automated reasoning and nonclassical logics.

[MR93e:03021] This paper, of an expository character, reconsiders the links between combinatory logic and the algebra of formulas of propositional logic with only implication. The novel aspect is the focussing on C. A. Meredith's condensed detachment, which enables an immediate comparison with J. A. Robinson's resolution. Condensed detachment is a formalized combination of substitution and ordinary detachment (or modus ponens). For example, the propositional arrow-formulas $(q \to r) \to ((p \to q) \to (p \to r))$ and $s \to s$ are not immediately suited for detachment, but after the substitution of s for q and r in the first formula mentioned above, we obtain $(p \to s) \to (p \to s)$. For condensed detachment the substitutions must be such that a most general formula is obtained as detachment result. Hence, there is a strong parallel with the resolution technique, which employs most general unifiers.

The paper contains many notions and facts related to these subjects. In particular, the authors state results about Hindley's notion of D-completeness. This concerns the question: given a theorem in an (ordinary) logic with detachment and substitution, is it then necessarily also a theorem in the corresponding logic with condensed detachment? Moreover, the authors relate a number of combinatory logics to algebras of (families of sets of) arrow-formulas closed under condensed detachment. Since this relation is convincingly straightforward, the authors use the term "fool's model" for such classes. They show that it can be beneficial to "bring the apparatus of combinatory logic to the analysis of propositional logic". They finally describe an implementation of a fool's model. For the entire collection see MR 92g:03004. Reviewed by R. P. Nederpelt.

van Benthem, J. (1991). *Language in action.* North-Holland Publishing Co., Amsterdam. Categories, lambdas and dynamic logic.

[**MR92g:03002**] This book gives a sweeping and personal view of mathematical linguistics that may be of interest to linguists, mathematical logicians and computer scientists. The main unifying theme is categorial grammar, with considerable attention given to lambda calculus, type theory and "various logical paradigms directed toward the structural and procedural aspects of information", including relevance logic, modal logic and linear logic. As stated in the introduction, the book is written for "an audience with a certain sophistication in mathematical logic, and nonparochial interests in investigating logical structure across various fields of application". It is written in a conversational style, with technical passages pleasantly interrupted by informal discussion.

The book begins with a short presentation of the syntax and model theory of typed lambda calculus, followed by a review of categorial grammar. Categorial grammar is then analyzed using the structure of proofs in implicational logic. As is well known, lambda terms provide a convenient notation for implicational proofs. This firmly establishes the connection between lambda calculus and categorial grammar and leads into a study of proof theory, cut elimination and decidability. Following a short chapter on the expressive power of categorial formalisms, the book turns to a brief coverage of model theory. The remaining half of the book pays greater attention to systems of logic than connections to natural language. Higher-type and polymorphic type theories are presented, followed by systems of modal logic, relational algebra and dynamic logic. The book concludes with a survey chapter on logical systems that is meant to serve as reference for the other chapters. Reviewed by John C. Mitchell.

de Swart, H. C. M. and Nederpelt, R. P. (1992). Implication: a survey of the different logical analyses of "if ..., then ...". *Nieuw Arch. Wisk. (4)*, 10(1-2):77–104.

[**MR93k:03030**] What the authors have put together is a cursory and curiously eclectic overview of some of the formal developments in the field of conditional logic in the past decades. Five approaches are considered, corresponding to truth-functional (material) and counterfactual conditionals, and strict, relevant, and intuitionistic implications. The goal seems to have been to provide an already interested but as yet uninformed reader a taste of the different semantics and proof theories, and perhaps some sense of the motivation behind each. The limitations of space prevented adequate discussion of the impetus for these theories, so a reader without prior interest will probably not be engaged. Similar pressures may have weighted against a careful presentation of the details of the results included, but an informed reader might yet benefit from the gathering of references, particularly in regard to the sketched adaptations of the tableau method to effective decision procedures. Reviewed by Daniel H. Cohen.

Hirokawa, S. (1992). Balanced formulas, BCK-minimal formulas and their proofs. In *Logical foundations of computer science—Tver '92*, pages 198–208. Springer, Berlin.

[**MR94j:03033**] Summary: "The structure of normal form proof figures is investigated for implicational formulas in BCK-logic, in which each assumption can be used at most once. Proof figures are identified with λ-terms. A formula is balanced iff no type variable occurs more than twice in it. It is known that a proof figure in $\beta\eta$-normal form is unique for balanced formulas. In this paper, it is shown that closed λ-terms in β-normal form having balanced types are BCK-λ-terms in which each variable occurs at most once. A formula is BCK-minimal iff it is BCK-provable and it is not a nontrivial substitution instance of another BCK-provable formula. It is also shown that the set of BCK-minimal formulas is identical to the set of principal type-schemes of BCK-λ-terms in $\beta\eta$-normal form." For the entire collection see MR 94h:68009.

Schwarz, G. (1992). Reflexive autoepistemic logic. *Fund. Inform.*, 17(1-2):157–173.

[**MR94c:03034**] The reflexive autoepistemic logic introduced by the author is a variant of Moore's autoepistemic logic [R. C. Moore, Artificial Intelligence 25 (1985), no. 1, 75–94; MR 86h:03043]. According to Moore, T is a stable expansion of A iff

$$T = \{\phi\colon A \cup \{L\psi\colon \psi \notin T\} \cup \{L\psi\colon \psi \in T\} \vdash \phi\}$$

while T is a reflexive expansion of A iff

$$T = \{\phi\colon A \cup \{L\psi\colon \psi \notin T\} \cup \{L\psi \equiv \psi\colon \psi \in T\} \vdash \phi\}.$$

While Moore's autoepistemic logic coincides with nonmonotonic KD45 in the sense of D. V. McDermott [J. Assoc. Comput. Mach. 29 (1982), no. 1, 33–57; MR 83h:03027], reflexive autoepistemic logic coincides with the nonmonotonic logic obtained by adding the schema $\phi \supset (ML\phi \supset L\phi)$ to S4. The main result is the following: if, roughly speaking, S^b is obtained from S by rewriting each subformula of the form $L\phi$ as $L\phi \wedge \phi$, then T is a reflexive expansion of S iff it is a stable expansion of S^b. The converse result is obtained by replacing each occurrence of the operator L by

LL. However, as the author notices, reflexive autoepistemic logic is not immune to the problem of ungrounded expansions. Reviewed by Giovanni Criscuolo.

Schwichtenberg, H. (1992). Minimal from classical proofs. In *Computer science logic (Berne, 1991)*, pages 326–328. Springer, Berlin.
[MR1232898] This item will not be reviewed individually. For details of the collection in which this item appears see 94c:68001.

Valentini, S. (1992). The judgement calculus for intuitionistic linear logic: proof theory and semantics. *Z. Math. Logik Grundlag. Math.*, 38(1):39–58.
[MR95d:03006] The semantics of a logical calculus can be given at three levels: formulas, proofs and reduction of proofs. The author gives (1) a judgement calculus for intuitionistic linear logic, and (2) a representation theorem for structures giving the semantics of this calculus at the level of formulas. A more general approach exists based on quantales.

Some remarks on the semantics at the levels of proofs and reduction are sketched. Reviewed by Gianfranco Mascari.

Wansing, H. (1992). Formulas-as-types for a hierarchy of sublogics of intuitionistic propositional logic. In *Nonclassical logics and information processing (Berlin, 1990)*, pages 125–145. Springer, Berlin.
[MR94h:03123] Summary: "This paper contains a detailed presentation of W. A. Howard's 'formulas-as-types notion of construction' for fragments of various subsystems of intuitionistic propositional logic [in To H. B. Curry: essays on combinatory logic, lambda calculus and formalism, 480-490, Academic Press, London, 1980; MR 82g:03094]. W. Buszkowski's distinction between two kinds of lambda-abstractors [in Initiatives in logic, 180–206, Nijhoff, Dordrecht, 1987; per bibl.; in Categorial grammars and natural language structures, 69–94, Reidel, Dordrecht, 1988; per bibl.] is taken up and suitable fragments of typed terms are singled out [cf. also J. F. A. K. van Benthem, Essays in logical semantics, Reidel, Dordrecht, 1986; MR 89d:03024; W. Buszkowski, op. cit., 1987]. The relationship between cut elimination and normalization of terms is dealt with. Amongst other things, it is shown that in certain cases in which applications of the cut rule can be eliminated from an implicational fragment of the logics considered, cut elimination and normalization of terms with respect to β-reduction are homomorphic images of each other." For the entire collection see MR 94b:68016.

Benton, N., Bierman, G., de Paiva, V., and Hyland, M. (1993). Linear λ-calculus and categorical models revisited. In *Computer science logic (San Miniato, 1992)*, pages 61–84. Springer, Berlin.
[MR94j:03029] From the text: "Girard's intuitionistic linear logic is a refinement of intuitionistic logic, where formulae must be used exactly once. In other words, the familiar weakening and contraction rules of Gentzen's sequent calculus are removed. To regain the expressive power of intuitionistic logic, these rules are returned, but in a controlled manner. A logical operator '!' is introduced which allows a formula

to be used as many times as required (including zero). In this paper we consider multiplicative exponential linear logic (MELL), i.e., the fragment which has multiplicative conjunction or tensor, $\otimes$, linear implication, $\multimap$, and the logical operator 'exponential', !." For the entire collection see MR 94f:68011.

Borghuis, T. (1993). Interpreting modal natural deduction in type theory. In *Diamonds and defaults (Amsterdam, 1990/1991)*, pages 67–102. Kluwer Acad. Publ., Dordrecht.
[MR96i:03010] This paper proposes a proposition-as-types interpretation of normal modal logics into a typed lambda calculus with few extra primitives. The starting point is a "Fitch style" natural deduction formulation of modal logic, based on the notion of subordinate proof and rules for exchanging formulas between the main and the subordinate proofs. In the interpretation proposed by the author the notion of subordinate proof corresponds to a suitable notion of "subordinate context" in the associated type assignment system. The rules for importing and exporting formulas in subordinate proofs are then associated to special "import" and "export" constants which can appear in the lambda terms associated to the modal proofs. A notion of reduction corresponding to proof-normalization is finally introduced. For the entire collection see MR 96b:03003. Reviewed by Alessandro Berarducci.

Hindley, J. R. (1993). BCK and BCI logics, condensed detachment and the 2-property. *Notre Dame J. Formal Logic*, 34(2):231–250.
[MR94g:03059] Summary: "Some of the main properties of the BCK and BCI logics of implication are summarized, focusing on their connections with their condensed logics and with combinators and lambda-calculus. (A condensed logic is the set of all formulas deducible from the logic's axioms by the condensed detachment rule of Carew Meredith.) A full proof is given of the preservation of the 2- and 1-2-properties by condensed detachment, based on ideas of S. Jaskowski." Reviewed by Martin W. Bunder.

Hirokawa, S. (1993). The relevance graph of a BCK-formula. *J. Logic Comput.*, 3(3):269–285.
[MR94m:03028] This paper is about BCK-λ-calculus, where variables may occur at most once in a term. The author gives a complete characterization of the set of principal type-schemes of BCK-λ-terms in β-normal forms. He expects that a similar characterization should be possible for BCK-λ-terms (not necessarily in normal form), but the technique used in this paper does not generalize. Reviewed by Eugenio Moggi.

Komori, Y. and Hirokawa, S. (1993). The number of proofs for a BCK-formula. *J. Symbolic Logic*, 58(2):626–628.
[MR95b:03013] A BCK-formula can be simply characterised as a substitution instance of an implicational formula in which no propositional variable appears more than twice. A BCK formula β is a minimal formula of a BCK-formula α if α is a substitution instance of β and β is not a nontrivial substitution instance of another BCK-formula.

The authors show, by proving an equivalent result in typed lambda calculus, that the number of normal form proofs of a BCK-formula is equal to its number of minimal formulas. Reviewed by Martin W. Bunder.

Mackie, I., Román, L., and Abramsky, S. (1993). An internal language for autonomous categories. *Appl. Categ. Structures*, 1(3):311–343.
[MR95b:03005] Propositional logic with finite conjunctions $\wedge$, $\top$ and implication $\rightarrow$ is a language for Heyting algebras (i.e. for poset Cartesian closed categories). Under the propositions-as-types analogy, simply typed λ-calculus (with finite product- and exponent-types) is a language in which types can be seen as propositions, and terms as proof-objects (for this logic). Also, simply typed λ-calculus is an internal language for Cartesian closed categories.

In the paper under review, this situation is extended to (the multiplicative fragment of) linear logic (with finite tensors $\otimes$, I and implication $\multimap$). The authors propose a term calculus for linear logic, in which the rules are such that variables occur precisely once. As a result of this simplicity, normalization and Church-Rosser (confluence) results can be established without difficulty. It is then shown that the calculus can be used as an internal language for autonomous (i.e. symmetric monoidal closed) categories.

In the fifth section the term calculus is applied to coherence of autonomous categories. The main Lemma 5.4 is wrong; the following counterexample is due to Todd Trimble. The terms

$$M = \lambda t.\ \mathrm{let}\ t\ \mathrm{be}\ f \otimes g\ \mathrm{in}\ fg \quad \text{and}$$
$$N = \lambda t.\ \mathrm{let}\ t\ \mathrm{be}\ f \otimes g\ \mathrm{in}\ g(\lambda x.\, f(\lambda y.\, yx))$$

are different normal forms, both of type

$$[(((A \multimap I) \multimap I) \multimap I) \otimes ((A \multimap I) \multimap I)] \multimap I.$$

But they should be the same according to the lemma. (The authors are preparing a corrigendum.)

In the final section it is shown how recursion on the natural numbers can be incorporated in the term calculus for linear logic. Reviewed by Bart Jacobs.

Pigozzi, D. and Salibra, A. (1993). Polyadic algebras over nonclassical logics. In *Algebraic methods in logic and in computer science (Warsaw, 1991)*, pages 51–66. Polish Acad. Sci., Warsaw.
[MR97m:03095] Summary: "The polyadic algebras that arise from the algebraization of the first-order extensions of a standard implicational calculus (SIC) are characterized and a representation theorem is proved. SICs were considered by H. Rasiowa [An algebraic approach to non-classical logics, North-Holland, Amsterdam, 1974; MR 56 #5285] and include classical and intuitionistic logic and their various weakenings and fragments, the many-valued logics of Post and Lukasiewicz, modal logics that admit the rule of necessitation, BCK logic, etc." For the entire collection see MR 97k:03003.

Borghuis, V. A. J. (1994). *Coming to terms with modal logic.* Technische Universiteit Eindhoven, Eindhoven. On the interpretation of modalities in typed λ-calculus, Dissertation, Technische Universiteit Eindhoven, Eindhoven, 1994.

[MR95i:03036] The aim of this book (the author's Ph.D. thesis) is to extend type theory to intensional reasoning, by providing a class of "modal" type systems in which a modal logic interpretation of "propositions-as-types" can be obtained. Introducing the framework of pure type systems (PTSs) (H. P. Barendregt, 1992), Chapter 1 gives a number of generalizations of PTSs to modal PTSs (MPTSs) by extending the PTS-language with modal operators, adding structure to the contexts (generalized contexts), and providing new rules to handle the modal operators. Chapter 2 presents a detailed account of the propositions-as-types interpretation of a family of normal modal logics in MPTSs. Using the Fitch style (F. B. Fitch, 1952) of natural deduction proofs, mappings are defined from such proofs to typed lambda terms, and soundness and (other) invariance results on the composition of such mappings are obtained. In addition, the author proposes reduction rules on MPTS-terms, corresponding to proof reductions on the modal natural deductions. Chapter 3 shows that MPTSs are a well-behaved extension of PTSs, and the desirable metatheoretical properties of PTSs are proved to be preserved. Chapter 4 discusses how both the modal natural deduction systems and MPTSs can be generalized to deal with multiple agents and multiple modalities, using the logic $\mathrm{KB}_{\mathrm{CD}}$ of S. Kraus and D. Lehmann (1986) as an example. Chapter 5 investigates the interpretation of modal predicate logics in the setting of MPTSs. Chapter 6 indicates how the modal type systems can be used in the formalization of (dialogue) communication, bringing together the type-theoretical representation of natural language with the existing work in epistemic/doxastic logic pragmatics, inside the MPTS-framework. The final Chapter 7 contains concluding remarks and directions for future research. This monograph contributes to the development of type-based theoretical representation, and will be of real interest to those in fields such as modal logic, proof theory, natural language formal semantics, and type theory. Reviewed by Neculai Curteanu.

Shramko, Y. (1994). Relevant properties. *Logic Log. Philos.*, (2):103–115. Ontologic. Essays on formal ontology (Graal-Müritz, 1994).

[MR1409857] This item will not be reviewed individually. For details of the collection in which this item appears see 97c:03013.

Borisavljević, M. (1995). A connection between cut elimination and normalization. *Filomat*, (9, part 3):619–632. Algebra, logic & discrete mathematics (Niš, 1995).

[MR97c:03133] The author studies cut elimination (sequent calculus) and normalization (natural deduction) for intuitionistic (full) propositional logic, by means of multicategories and typed λ-calculus. For the entire collection see MR 96k:00015. Reviewed by Harold Schellinx.

Bunder, M. W. (1995). A simplified form of condensed detachment. *J. Logic Lang. Inform.*, 4(2):169–173.

[MR96e:03005] Summary: "This paper gives a simple, elegant statement of the condensed detachment rule that is independent of most general unifiers and proves that this is equivalent to the longer, more usual, formulation."

Feferman, S. (1995). Definedness. *Erkenntnis*, 43(3):295–320. Varia with a Workshop on the Foundations of Partial Functions and Programming (Irvine, CA, 1995).

[MR98f:03026] A divergence between formal logic and the actual practice of mathematicians exists with regard to the use of possibly undefined terms. An assertion such as "for all x, $x(x^{-1}) = 1$ if x^{-1} exists" makes perfect sense to a mathematician, but does not make sense in standard formal logic. To see this, consider that an immediate consequence of this sentence seems to be "$(0)(0^{-1}) = 1$ if 0^{-1} exists", and then ask precisely what the name '0^{-1}' refers to, and what the sense of the apparent predicate "exists" is.

The purpose of the paper under review is to define and suggest applications for logics in which formal reasoning about possibly undefined expressions is supported. The author defines two classes of candidate logics, which he calls "logics of existence" and "logics of definedness".

In logics of existence, free variables are taken to range over a larger realm of objects than we normally suppose (including objects that do not exist), and an existence predicate E is supplied to allow us to identify the objects in this range that really exist. The issue here is which objects in our range do exist; thus "logic of existence".

In logics of definedness, free variables are taken to range only over the objects which exist, and the "existence" operator is a special operator (not a predicate at all) which is applied to terms to tell us whether they have referents: "T is defined" is written "$T\downarrow$". Here everything exists, and the issue is which terms in our language are defined; thus "logic of definedness".

We can see the formal difference between these logics by considering the rules for reasoning with the universal quantifier. In both logics, one changes the rule of universal instantiation from $(\forall x\, P(x)) \vdash P(t)$ to $(\forall x\, P(x)), t\downarrow \vdash P(t)$ (one would have $E(t)$ instead of $t\downarrow$ in a logic of existence). In the example at the beginning of the review, one needs the additional (and false) hypothesis $0^{-1}\downarrow$ in order to draw the unpleasant conclusion given there.

A difference between the logics is seen in the status of the rule $A(x) \vdash A(t)$ of classical logic; in a logic of existence, this rule is retained, whereas an additional hypothesis must be added in a logic of definedness: $A(x), t\downarrow \vdash A(t)$.

The author is being somewhat confusing on p. 298 where he tells us that we have the rule $A(x) \vdash (\forall x\, A(x))$ in logics of definedness but not in logics of existence; in fact, this rule is valid in both logics, but a stronger rule is needed in a logic of existence, namely: $(E(x) \to A(x)) \vdash (\forall x\, A(x))$. The weaker rule is valid but is not the correct analogue to the classical rule.

The author prefers logics of definedness to logics of existence, and proceeds to develop and apply logics of definedness in the rest of the paper. The reviewer agrees with the author's claim that logics of definedness do a better job of capturing the everyday reasoning of mathematicians. The differences between the two approaches are not profound; logics of existence and definedness prove exactly the same closed formulas.

The author presents a complete formal definition of a logic of definedness called LPT ("logic of partial terms"). We discuss here only the formal axioms given for definedness of terms. In keeping with the fact that this is a logic of definedness, in which free variables range only over existing objects, each sentence $x\downarrow$, with x a free variable, is an axiom. It is natural to assert $c\downarrow$ for each constant. We have $s = t \to s\downarrow \wedge t\downarrow$, and $R(t_1, \cdots, t_n) \to t_i\downarrow$ for each atomic relation symbol R and appropriate i.

We have saved the most interesting definedness axiom for last, since the author gives it special consideration. This asserts that for each function symbol f, $f(t_1, \cdots, t_n)\downarrow$

$\rightarrow t_i \downarrow$ for each appropriate i. The author points out that this corresponds to "call-by-value semantics" in the theory of computation; the alternative "call-by-name semantics" would allow failures of this axiom.

An obvious thing to do in a logic of definedness (or in a logic of existence, for that matter) is to introduce Russell's definite description operator and the Hilbert symbol in terms $(\iota x\, \phi)$ and $(\epsilon x\, \phi)$ with meanings "the unique x such that ϕ" and "an arbitrarily selected x such that ϕ", respectively. These are naturally introduced as primitive operations, with obvious criteria of definedness. In the case of the definite description operator, the use of a logic of definedness (or existence) is an improvement over Russell's original interpretation of the operator via a complex contextual definition (it is interesting from a historical standpoint to observe that a logic of existence implements something resembling the views of Meinong to which Russell was reacting).

In the remainder of the paper, the author discusses the application of these ideas to systems of combinatory logic and λ-calculus, with the ultimate end in view of applications in computer science.

In combinatory logic (CL), the system which is proposed is one in which the usual axiom $\mathsf{S}xyz = xz(yz)$ is qualified to assert that $\mathsf{S}xy$ is always defined and $\mathsf{S}xyz$ is defined iff $xz(yz)$ is defined, in which case the usual equation $\mathsf{S}xyz = xz(yz)$ holds. The axiom for the K combinator takes its usual form $\mathsf{K}xy = x$. This implements a partial combinatory logic in which terms with normal forms exist, but "values" which do not have normal forms may turn out to be undefined. The author continues with a discussion of a similar treatment of λ-calculus; difficulties arise for the relationship between these "partial" versions of CL and λ-calculus for the usual reasons having to do with the property of "weak extensionality" possessed by the latter but not the former (embodied in the ability to substitute equals for equals inside λ-terms). The author refers to the existence of but does not describe a version of "partial" λ-calculus with explicit substitutions defined by Strahm which has a more satisfactory relationship with "partial" combinatory logic.

This is followed by a discussion of systems IOC and EOC intended for use in proofs of correctness and termination of programs. IOC is a two-sorted system (with individuals and classes), obtained by adding logical, arithmetic and class operations to a partial λ-calculus, with the strength of second-order arithmetic; EOC differs from IOC in having a weaker scheme of class comprehension and the strength of first-order arithmetic; it seems still to be adequate for the applications envisaged. There is an interesting discussion of the implementation of polymorphic types in these systems.

This is a well-written paper, carrying an important idea all the way from basic considerations to fairly complex applications. For the entire collection see MR 97a:03008. Reviewed by M. Randall Holmes.

Fujita, K.-e. (1995). On embedding of classical substructural logics. *Sūrikaisekikenkyūsho Kōkyūroku*, (918):178–195. Theory of rewriting systems and its applications (Japanese) (Kyoto, 1995).

[MR97a:03016] Summary: "There is an intimate connection between proofs of the natural deduction systems and typed lambda calculus. It is well known that in simply typed lambda calculus, the notion of formulas-as-types makes it possible to find fine structure of the implicational fragment of intuitionistic logic, i.e., relevance logic, BCK-logic and linear logic. In this paper, we investigate classical substructural logics consisting of implication and negation. However, our method is general enough

to apply to rich systems beyond the fragment. We show that proofs in Parigot's λ-calculus with proper constraints exactly correspond to proofs in substructural logics of Gentzen's LK. Moreover, we discuss three embeddings of classical substructural logics into the corresponding intuitionistic substructural logics." For the entire collection see MR 96k:68005.

Herbelin, H. (1995). A λ-calculus structure isomorphic to Gentzen-style sequent calculus structure. In *Computer science logic (Kazimierz, 1994)*, pages 61–75. Springer, Berlin.

[MR98f:03047] Summary: "We consider a λ-calculus for which applicative terms have, instead of the form $(\cdots((uu_1)u_2)\cdots u_n)$, the form $(u[u_1;\cdots;u_n])$, where $[u_1;\cdots;u_n]$ is a list of terms. While the structure of the usual λ-calculus is isomorphic to the structure of natural deduction, this new structure is isomorphic to the structure of Gentzen-style sequent calculus. To express the basis of the isomorphism, we consider intuitionistic logic with implication as the sole connective. However, we do not consider Gentzen's calculus LJ, but a calculus LJT which leads to a restriction of the notion of cut-free proofs in LJ. We also need to consider explicitly, in a simply typed version of this λ-calculus, a substitution operator and a list concatenation operator. In this way, each elementary step of cut-elimination exactly matches with a β-reduction, a substitution propagation step or a concatenation computation step.

"Though it is possible to extend the isomorphism to classical logic and to other connectives, we do not do so in this paper." For the entire collection see MR 98d:68013.

Komori, Y. (1995). Logics as sets of diagrams of proofs. *Sūrikaisekikenkyūsho Kōkyūroku*, (927):53–55. Problems concerning nonclassical logics and their Kripke semantics (Japanese) (Kyoto, 1995).

[MR1402047] This item will not be reviewed individually. For details of the collection in which this item appears see 97c:03014.

Pfenning, F. and Wong, H. C. (1995). On a modal λ-calculus for S4. In *Mathematical foundations of programming semantics (New Orleans, LA, 1995)*, page 20 pp. (electronic). Elsevier, Amsterdam.

[MR98m:03046] The paper presents a concise formulation of a proof term modal λ-calculus. It is shown that the calculus is equivalent, via a Curry-Howard isomorphism, to the intuitionistic modal logic S4. This calculus is also well suited for practical applications in functional programming and logical frameworks, due to a simple type checking algorithm. Among the main results of the paper are the subject reduction theorem and the proof of existence of a canonical form for every well-typed term. Some related results and possible applications are discussed. For the entire collection see MR 98g:68006. Reviewed by M. S. Burgin.

van Benthem, J. (1995). *Language in action.* MIT Press, Cambridge, MA. Categories, lambdas and dynamic logic, Revised reprint of the 1991 original.

[MR96g:03003] The 1991 original has been reviewed [North Holland, Amsterdam, 1991; MR 92g:03002]. In the addenda to this edition, the author surveys recent developments in what he calls the "categorial paradigm", providing additional references and a modified list of open problems.

Bierman, G. M. (1996). Towards a classical linear λ-calculus (preliminary report). In *Linear logic 96 (Tokyo)*, page 13 pp. (electronic). Elsevier, Amsterdam.

[MR1488162] This item will not be reviewed individually. For details of the collection in which this item appears see 98f:68009.

Blute, R. F. and Scott, P. J. (1996). Linear Läuchli semantics. *Ann. Pure Appl. Logic*, 77(2):101–142.

[MR96m:03037] H. Läuchli's semantics for intuitionistic logic, introduced in an earlier, somewhat less familiar paper [in Intuitionism and proof theory (Buffalo, NY, 1968), 227–234, North-Holland, Amsterdam, 1970; MR 42 #5778] and based on a more sophisticated notion of "set" and "element", models a formula of intuitionistic propositional calculus by a set with a distinguished permutation on it (which may be identified with a Z-set, a set with an action of the free cyclic group Z of integers), and models a proof by an invariant element. Läuchli's completeness theorem says: a formula is intuitionistically provable if and only if its interpretation in every abstract model contains an invariant element (i.e. an abstract proof). In the paper under review the authors introduce a linear analogue of Läuchli's semantics (strengthening it to the level of proofs, rather than provability), based upon an extension of functorial polymorphism [E. S. Bainbridge et al., Theoret. Comput. Sci. 70 (1990), no. 1, 35–64; MR 91g:03129a; corrigendum; MR 91g:03129b] to the multiplicative fragment of linear logic (MLL). The main result of the paper is the following "full completeness": interpreting formulas as definable multivariant functors on a category of topological vector spaces, the set of "diadditive" dinatural transformations which are uniform (= equivariant) with respect to actions of Z has a vector space structure, with basis the cut-free proofs in MLL with the mix rule. In conclusion, the authors show that diadditive Z-uniform dinatural transformations are also uniform with respect to the actions of arbitrary cocommutative Hopf algebras. The paper ends with a discussion of possible extensions of the work to noncommutative logic. Reviewed by R. Gylys.

Bunder, M. (1996). Logics without contraction. II. In *Logic and reality (Christchurch, 1989)*, pages 337–349. Oxford Univ. Press, New York.

[MR1462367] A review for this item is in process.

Buszkowski, W. (1996). The finite model property for BCI and related systems. *Studia Logica*, 57(2-3):303–323.

[MR98g:03021] The author proves the finite model property (fmp) for the implicational logic BCI (a Hilbert style axiomatization is provided by the combinators B, C and I of combinatory logic) and some extensions and variations, with respect to their "natural" algebraic semantics, in the case of BCI, and assignments in BCI-algebras. The arguments are somewhat involved technically and use as a key tool the method of so-called barriers, introduced earlier by the author to prove the fmp for fragments of the Lambek calculus.

BCI is, in fact, the implicational fragment of linear logic, and it is worthwhile to observe that, as an alternative to the author's direct proof, the results of this paper can be obtained as corollaries to the fmp for (nonexponential) propositional (commutative as well as noncommutative) linear logic with respect to Girard's phase semantics, a result due to Y. Lafont ["The finite model property for various fragments of linear logic", J. Symbolic Logic, to appear]. Interestingly, because of linear logic's symmetries, working in this broader context drastically simplifies the argument. Reviewed by Harold Schellinx.

Došen, K. (1996a). Deductive completeness. *Bull. Symbolic Logic*, 2(3):243–283.

[MR97j:03118] This clear and readable paper is an excellent exposition of Lambek's strengthening and generalization of the deduction theorem in categories related to intuitionistic propositional logic. Essential notions of category theory are introduced so as to yield a simple reformulation of Lambek's functional completeness theorem, from which its main consequences can be readily drawn. The connections of the theorem with combinatory, modal and substructural logics are also considered. Reviewed by Leo Esakia.

Došen, K. (1996b). Errata: "Deductive completeness" [Bull. Symbolic Logic **2** (1996), no. 3, 243–283; MR 97j:03118]. *Bull. Symbolic Logic*, 2(4):523.

[MR1460318] There will be no review of this item.

Hirokawa, S. (1996). The proofs of $\alpha \to \alpha$ in P-W. *J. Symbolic Logic*, 61(1):195–211.

[MR97d:03006] Via the formulas as types isomorphism, combinators represent Hilbert style proofs and λ-terms represent natural deduction style proofs of theorems of $\mathrm{H}_{\to}$. The author considers the system $\mathrm{P}-\mathrm{W}$, whose axioms are the types of the combinators B, B' and I, and shows that the class of hereditary right-maximal linear λ-terms (HRML) is the class that corresponds to $BB'I$-combinators and hence to natural deduction style proofs in this logic. Using the result of Martin and Meyer on the unprovability of any instance of $\alpha \to \alpha$ in $\mathrm{P}-\mathrm{W}-\mathrm{I}$ (or BB'-logic), he also shows that the proofs of instances of $\alpha \to \alpha$ in $\mathrm{P}-\mathrm{W}$ are the elements of HRML that $\beta\eta$-reduce to $\lambda x.x$. Reviewed by Martin W. Bunder.

Pliuškevičius, R. (1996). On saturation with flexible function symbols. In *Mathematical foundations of computer science 1996 (Cracow)*, pages 482–493. Springer, Berlin.

[MR98f:03019] Summary: "This paper extends the author's similarity saturation calculus by including flexible function symbols. The most attractive property of the similarity saturation calculus is that it allows us to build derivations constructively for both a finitary complete and finitary incomplete first-order linear temporal logic. The saturation calulus contains the so-called alternating rule and this rule splits off the logical part from the temporal one in the quasi-primary sequents. If the saturation process involves unification and flexible function symbols, then the alternating rule includes (in the temporal part of the rule) renaming of variables and a replacement of flexible function symbols by rigid ones. The employment of flexible symbols involves the introduction of some kind of constraints in the saturation process." For the entire collection see MR 98f:68011.

Čubrić, D. (1997). On the semantics of the universal quantifier. *Ann. Pure Appl. Logic*, 87(3):209–239.

[MR99a:03009] This paper is concerned with the presentation of categorical models for the universal fragment of intuitionistic logic, extending the well-known relations between Cartesian closed categories and the $(\top, \wedge, \rightarrow)$-fragment of intuitionistic propositional logic (proof theory, type theory, and the lambda calculus) introduced by J. Lambek and P. J. Scott [Introduction to higher order categorical logic, Cambridge Univ. Press, Cambridge, 1986; MR 88a:03032]. To capture the notion of universal quantification categorically and to model the $(\top, \wedge, \rightarrow, \forall)$-fragment of multisorted first-order intuitionistic logic, the author works with "$\forall$-fibrations", a kind of Grothendieck fibrations (or a "right" fragment of F. W. Lawvere's hyperdoctrines), and with typed "$\lambda\forall$-calculus", focusing on equality of proofs. He relates the latter concepts by two "opposite" constructions: in the first, starting with a cloven $\forall$-fibration, the author assigns to it the internal language and the theory, while, in the other, starting with a $\lambda\forall$-theory he constructs the canonical $\forall$-fibration. Moreover, the author obtains an interesting generalization of the famous Yoneda lemma and an extension of H. Friedman's completeness result for typed lambda calculus, giving exhaustive explanations. Reviewed by R. Gylys.

Dunn, J. M. and Meyer, R. K. (1997). Combinators and structurally free logic. *Log. J. IGPL*, 5(4):505–537 (electronic).

[MR99d:03013] As the authors say in their introduction, "In this paper we arrange a marriage between combinatory logic and substructural logics, showing how they can mutually support each other. Indeed, substructural logics, at least in their implication/fusion fragments, can be viewed as combinatory logics (with residuation) where the combinators are limited so as to license only certain 'structural rules'. And somewhat surprisingly, combinatory logic can itself be viewed as a substructural logic, and a semantics can be provided for it using a ternary accessibility relation, as has been done with other substructural logics."

The authors begin by introducing combinatory posets, which are built on left residuated posets with lower left identities. These combinatory posets are similar to the "combinatory algebras" of H. P. Barendregt [The lambda calculus, North-Holland, Amsterdam, 1981; MR 83b:03016]; the main difference is that the authors allow basic combinators other than S and K. These algebras lead to a "Kripke-style" semantics using frames with a ternary accessibility relation in a way similar to the Routley-Meyer semantics for relevance logic. The authors use them to give a completeness theorem for combinatory logic by proving a representation theorem for combinatory posets. This is followed by a philosophical interpretation of these models in which an element of a combinatory poset can be interpreted both as a set of states and as a set of untyped actions on states. This double interpretation makes it possible for one element to be "applied" to any other, including itself. The resulting interpretation of application is modeled the same way as "fusion" in relevance logic.

The authors also introduce dual combinators which apply from the right, so that, for example, $a(b(c\mathbf{s})) = (ab)(ac)$. They then explore the relationship between these combinators and some substructural logics. This is done by replacing structural rules by rules in which combinators are used to carry out the same operations. For example, the rule of weakening on the left is replaced by a rule called ($\mathbf{K}\vdash$): $\Gamma[\Phi] \vdash \chi \rightarrow \Gamma[\mathbf{K}, \Phi, \Phi] \vdash \chi$. The authors prove a cut elimination theorem for several such logics.

{Reviewer's remarks: The paper contains a number of misprints. These include several cited works which do not appear (or are misidentified) in the references, including "Dunn (1995)" on p. 506, "Ward and Dilworth (1939)" on p. 509, and "Dunn (1993)" on p. 511 and later. Also, "Curry and Feys (1972)" does not correctly indicate the date of publication: the book originally appeared in 1958 and was reprinted in 1968 and 1974. The date 1972 really belongs to the second volume by Curry, Hindley, and the reviewer. On p. 510, the characters "$\mathfrak{k}$, $\mathfrak{s}$, $\mathfrak{i}$, $\mathfrak{c}$, $\mathfrak{b}$, $\mathfrak{w}$" are identified as "lower-case Roman letters".} Reviewed by J. P. Seldin.

Fujita, K.-e. (1997). Strong normalization of pure gl_w-$\lambda\mu$-terms. *Sūrikaisekikenkyūsho Kōkyūroku*, (1021):1–13. Investigations of non-classical logics based on sequent calculus and Kripke semantics (Japanese) (Kyoto, 1997).

[MR1643388] This item will not be reviewed individually. For details of the collection in which this item appears see 99d:03003.

Swoyer, C. (1997). Complex predicates and conversion principles. *Philos. Stud.*, 87(1):1–32.

[MR98f:03030] In free logic, a statement $P(a)$ is false if a fails to denote. From this it follows that the usual rules for λ-abstraction and λ-conversion are not unconditionally valid, since $(\lambda x \sim P(x))(a)$ is false if a does not denote, but $\sim P(a)$ is true. The author argues that this is a virtue of free logic rather than a defect. To defend this view he reviews a number of well-known semantical puzzles and shows how free logic can deal with them. These concern, among other things, modal and temporal contexts and definite descriptions. For example, the formula $\Box((\exists! y)Py \rightarrow P(\iota x Px))$ is true, saying that necessarily the P is P on condition that there exists a unique P, but $[\lambda z \Box((\exists! y)Py \rightarrow Pz)](\iota x.Px)$ is false, saying that the P necessarily has P whenever there is a unique P. (To see that this is invalid, take Px to mean that x is the number of planets.)

The main application consists in a reassessment of Gareth Evans' demonstration that there can be no vague objects in language [see Analysis 38 (1978), no. 3, 208]. According to Swoyer, Evans' argument rests on an illicit abstraction. Namely, this is the step where Evans argues that if it is indeterminate whether $P(a)$ (in symbols $\nabla P(a)$) then a has the property that its being P is indeterminate, in symbols $[\lambda x \nabla P(x)](a)$. (In Evans' argument, Pa is '$b = a$'.) Swoyer argues that this works only if the source of indeterminacy is the predicate. Otherwise, if it is a (e.g. if a is given by a definite description that is indeterminate) then this implication is false. Reviewed by Marcus Kracht.

Avron, A., Honsell, F., Miculan, M., and Paravano, C. (1998). Encoding modal logics in logical frameworks. *Studia Logica*, 60(1):161–208. Natural deduction.

[MR1620059] A review for this item is in process.

Bimbó, K. and Dunn, J. M. (1998). Two extensions of the structurally free logic lc. *Log. J. IGPL*, 6(3):403–424.

[MR99f:03012] The system LC of Dunn and Meyer is a logic with fusion and implication that has no structural rules. These are replaced by the use of any complete set of combinators as formulas. (The combinators cannot in turn be replaced by the formulas that are their types if the cut rule is to be admissible.)

The present paper presents two extensions of LC that include conjunction ($\wedge$) and disjunction ($\vee$); one has the distributive laws for $\wedge$ over $\vee$ and $\vee$ over $\wedge$, the other does not. The admissibility of the cut rule is proved for both extensions and a Routley-Meyer style semantics is given for the distributive one. Reviewed by Martin W. Bunder.

Borghuis, T. (1998). Modal pure type systems: type theory for knowledge representation. *J. Logic Lang. Inform.*, 7(3):265–296.

[MR1639056] A review for this item is in process.

Boutilier, C. (1998). A unified model of qualitative belief change: a dynamical systems perspective. *Artificial Intelligence*, 98(1-2):281–316.

[MR99f:68185] In this extended paper, the author develops a formal account of belief change as a complex process, involving elements of revision (in the sense of Alchourron, Gardenfors and Makinson) and of update (in the sense of Katsuno and Mendelzon). By holding certain components fixed at certain values, the process takes each of these ingredients as a limiting case.

The central idea is as follows. Suppose we are given a belief state K and a statement α as input; we want to define the result of "changing K to accommodate α". A transition to α is defined as a passage from a world w, via an event e, to another world v that satisfies α. The plausibility of the first world w given K is specified by one function (indeed, K is represented as such a function), the plausibility of the event e (given w) by another function, and the plausibility of the resulting world v (given w, e) is given by a third function. Following Spohn, all of these functions take their values in the finite ordinals, with zero as most plausible. The plausibility of the entire transition is then defined as the sum of the three components. The result of the belief change in K to accommodate α then consists of those statements that are true in all of the most plausible transitions from K to α. AGM revision corresponds to the case where the only event considered is the null event. The relation to Katsuno-Mendelzon update is rather more complex. Suitable constraints on the author's semantics correspond to all but one of the Katsuno-Mendelzon postulates; the remaining postulate (U8) can be validated by a further constraint which, however, the author finds rather unnatural.

In final sections, the author considers the possibility of modifying his construction to allow the new information to be less than certain, and also compares his construction with one of similar intent due to N. Friedman and J. Y. Halpern [in Principles of knowledge representation and reasoning (Bonn, 1994), 190–201, Morgan Kaufmann, San Francisco, 1994], and also very briefly with that of K. Schlechta, D. Lehmann and M. Magidor [in Theoretical aspects of rationality and knowledge (De Zeeuwse Stromen, 1996), 137–145, Morgan Kaufmann, San Francisco, 1996]. Reviewed by David Makinson.

Fujita, K.-e. (1998). On proof terms and embeddings of classical substructural logics. *Studia Logica*, 61(2):199–221.

[**MR99f:03013**] Summary: "There is an intimate connection between proofs in natural deduction systems and typed lambda calculus. It is well known that in simply typed lambda calculus, the notion of formulae-as-types makes it possible to find the fine structure of the implicational fragment of intuitionistic logic, i.e., relevant logic, BCK-logic and linear logic. In this paper, we investigate three classical substructural logics ($\mathrm{GL}, \mathrm{GL}_c, \mathrm{GL}_w$) of Gentzen's sequent calculus consisting of implication and negation, which contain some of the right structural rules. In terms of Parigot's λ-calculus with proper restrictions, we introduce a proof term assignment to these classical substructural logics. According to these notions, we can classify the λ-terms into four categories. It is proved that well-typed GL_x-λ-terms correspond to GL_x proofs, and that a GL_x-λ-term has a principal type if stratified where x is nil, c, w or cw. Moreover, we investigate embeddings of classical substructural logics into the corresponding intuitionistic substructural logics. It is proved that the Gödel-style translations of GL_x-λ-terms are embeddings preserving substructural logics. As by-products, we obtain that an inhabitation problem is decidable and that well-typed GL_x-λ-terms are strongly normalizable."

1.7 Rewriting

Staples, J. (1979). A graph-like lambda calculus for which leftmost-outermost reduction is optimal. In *Graph-grammars and their application to computer science and biology (Internat. Workshop, Bad Honnef, 1978)*, pages 440–455. Springer, Berlin.

[**MR81i:03017**] From the introduction: "This paper gives a weak solution to the problem of giving an optimal reduction algorithm for a suitable representation of the lambda calculus, though the algorithm of the present paper has not been developed from J.-J. Levy's theory ("Reductions correctes et optimales dans le lambda -calcul", These de doctorat, Univ. Paris VII, Paris, 1978). The representation of the lambda calculus which we give faithfully models reduction to normal form of terms of the classical lambda calculus which have normal forms; we prove for this representation that leftmost-outermost reduction is optimal. The solution that we give is a weak one because optimal reductions in the calculus considered here are not always faster than optimal reductions in the classical lambda calculus."

Böhm, C. and Micali, S. (1980). Minimal forms in λ-calculus computations. *J. Symbolic Logic*, 45(1):165–171.

[**MR81h:03030**] The authors discuss and characterize what they call minimal forms in the λ-calculus. The notion of minimal form is an extension of the notion of normal form. Minimal forms are just the minimal terms with respect to the quasiordering generated by the reduction relation. Minimal forms which are unique (up to α-conversion) are normal forms. The first attempt to characterize terms having a minimal form by using cofinal reduction strategies fails. Theorem 2: If some cofinal strategy applied to a term F loops or stops, then F has a minimal form. Unfortunately the converse statement is true only for terms with only finitely many (up to

α-conversion) minimal forms, and the authors provide an example of a term with infinitely many minimal forms. Using ideas connected to Gross-reductions the authors obtain the following characterization of minimal forms (Theorem 5): A is a minimal form if and only if $G(A)$ reduces to A, where $G(A)$ is the result of a complete development of all redexes in A. As an application it is proved that the set of terms having minimal forms is recursively enumerable; that it is not recursive follows from a theorem of D. Scott. Reviewed by Gerd Mitschke.

Klop, J. W. (1980). *Combinatory reduction systems.* Mathematisch Centrum, Amsterdam.

[**MR83e:03026**] From the introduction: "This monograph is the author's Ph.D. thesis, written under the guidance of D. van Dalen and H. P. Barendregt. It is concerned exclusively with syntactical properties of lambda-calculus, combinatory logic, recursive program schemes, and in general, term rewriting systems with bound variables; especially those syntactical properties which concern reductions. 'Combinatory reduction systems' (CRSs) is our name for term rewriting systems with bound variables. The main results are: (I) the introduction of the concept of CRS and the basic syntactical theorems for CRSs, notably the Church-Rosser theorem (CR), the lemma of parallel moves, and the theorem on finite developments (FD); (II) the generalization of a method due to R. Nederpelt which enables one to reduce strong normalization proofs for certain CRSs to weak normalization proofs (this device is not only interesting in itself, but also enabled us to obtain the theorems FD, CR, etc.); (III) the negative result that CR fails for certain non-left-linear CRSs, e.g., $\lambda(\eta) \oplus$ Surjective Pairing $\not\models$ CR, on the other hand, the positive result that, e.g., $CL \oplus D(M, M) \to M \models$ CR." Contents: Chapter I. $\lambda\beta$-calculus and definable extensions. Chapter II. Regular combinatory reduction systems. Chapter III. Irregular combinatory reduction systems. Chapter IV. $\lambda\beta\eta$-calculus.

Pettorossi, A. (1981a). Comparing and putting together recursive path ordering, simplification ordering and nonascending property for termination proofs of term rewriting systems. In *Automata, languages and programming (Akko, 1981)*, pages 432–447. Springer, Berlin.

[**MR83g:03016**] Strong termination of a rewriting system means that there is no infinite sequence of rewriting steps in that system. It is interesting because a wide class of algorithms can be expressed as rewriting systems, in such a way that termination of the algorithm for all inputs is equivalent to strong termination of the rewriting system. In this paper a "nonascending" property, previously introduced by the author, is shown to be sufficient for the strong termination of systems of proper combinators. Such systems have rewriting rules of the form $Xx_1 \ldots x_n \to v$, where X is a primitive symbol, $x_1, \ldots, x_n$ are variables and v is an applicative combination of some or all of those variables. The paper concludes with an account and comparison of various different methods of proving strong termination, illustrated by examples. A methodology named "diagram of matchings" is proposed as a means of combining

the strengths of the various methods. (For the entire collection see MR 82i:68003.) Reviewed by John Staples.

Pettorossi, A. (1981b). A property which guarantees termination in weak combinatory logic and subtree replacement systems. *Notre Dame J. Formal Logic*, 22(4):344–356.

[MR83e:03027] Let combinatory weak reduction be defined using any set of basic combinators, with reduction rules of the form $Xx_1 \dots x_m \to Y$, where Y is a combination of $x_1, \dots, x_m$ only, and m depends on X. X is said to be "without compositive effect" if Y has the form $x_{i_1} \dots x_{i_k}$ where for all $j, i_j \leq m$, and there may be repetitions. Suppose we represent terms T as binary trees as usual, with the root at the bottom. Let the "index" of an occurrence of an atom in T be the number of left-branch choices made in climbing up the tree from the root to the atom. A combinator X has the "nonascending" (NA) property if for all $i \leq m$, the indices of all occurrences of x_i in Y are $\leq$ the index of x_i in $Xx_1 \dots x_m$. (For X without compositive effect, the NA property says that for all $i \leq m, m-i \geq max\{k-j : i_j = i\}$; note $\geq$, not $>$.) The author proves that for every set $\{X_1, X_2, \dots\}$ of basic combinators with the NA property and without compositive effect, every combination of $X_1, X_2, \dots$ has normal form, and all its reductions terminate. Reviewed by R. Hindley.

Pettorossi, A. (1982). Tree rewriting systems, combinatory weak reduction systems and type free languages. *Fund. Inform.*, 5(3-4):279–299.

[MR85c:68042] Author's summary: "We consider combinators as tree transducers: this approach is based on the one-to-one correspondence between terms of combinatory logic and trees and on the fact that combinators may be considered as transformers of terms. Since combinators are terms themselves, we deal with trees as objects to be transformed and tree transformers as well. Methods for defining and studying tree rewriting systems inside combinatory weak reduction systems and weak combinatory logic are also analyzed, and particular attention is devoted to the problem of finiteness and infinity of the generated tree languages (here defined). This implies the study of the termination of the rewriting process (i.e., reduction) for combinators."

Bunder, M. W. (1985). An extension of Klop's counterexample to the Church-Rosser property to λ-calculus with other ordered pair combinators. *Theoret. Comput. Sci.*, 39(2-3):337–342.

[MR87h:03021] The Church-Rosser property for λ-calculus states that if a term A reduces to two terms B and C then there is a term D such that both B and C reduce to D. This is one of the fundamental results on reduction in the λ-calculus. In 1979, Klop showed that λ-calculus with a surjective pairing function could not satisfy the Church-Rosser property. Klop's proof, using Böhm trees, in fact showed that certain other even simpler extensions to the λ-calculus also failed to be Church-Rosser. The author's contribution in this paper is to give a new more elementary proof of this result. A side benefit is that the new proof holds for a wider class of reduction systems than Klop's did. A condition is given on lambda terms Y such that for a Y satisfying these conditions, if the rule $DAA \to Y$ is added to the lambda calculus, then the resulting system is not Church-Rosser. These conditions on Y hold for all of the cases discussed by Klop and seem likely to hold for most Y's of interest. Reviewed by Kim Bruce.

Karr, M. (1985). "Delayability" in proofs of strong normalizability in the typed lambda calculus. In *Mathematical foundations of software development, Vol. 1 (Berlin, 1985)*, pages 208–222. Springer, Berlin.
[MR87i:03016] From the introduction: "The goal here is to have a general theory that allows us to adjoin new reductions to the β-reduction, and by simple structural examination of the new reductions, ascertain that terms remain normalizable in the new system. To do this, we have developed a new strategy for proving strong normalizability, which as a special case yields an independent proof of normalizability of terms under β-reduction. While we use the pairing reductions to exemplify the general results, the emphasis is on clarifying the properties of reductions that allow the proof of normalizability." For the entire collection see MR 86j:68005.

Hardin, T. and Laville, A. (1986). Proof of termination of the rewriting system SUBST on CCL. *Theoret. Comput. Sci.*, 46(2-3):305–312.
[MR88h:68053] Summary: "P. L. Curien [Categorical combinators, sequential algorithms and functional programming, Pitman, London, 1985; MR 87k:68002] defined a translation of the λc-calculus in the pure combinatory categorical logic and established an equivalence theorem between these two theories. The rewriting system SUBST simulates, in particular, the substitution of the λc-calculus. This system is locally confluent. We show here that it is also Noetherian."

Barendregt, H. P., Kennaway, J. R., Klop, J. W., and Sleep, M. R. (1987). Needed reduction and spine strategies for the lambda calculus. *Inform. and Comput.*, 75(3):191–231.
[MR89h:03020] This paper suggests a new approach toward finding more efficient normalizing strategies for the λ-calculus. A redex R in a term M is said to be needed if every reduction of M to a normal form reduces some residual of R. It is shown that these needed redexes can be contracted in any order without risking nontermination. It is also shown that (1) a redex is needed in a term if and only if it is contracted in the leftmost reduction path of that term; (2) a reduction path with the property that beyond any point in the path is a redex that is needed in the term which it contracts is normalizing; and (3) being needed is an undecidable property. A family of algorithms (called spine strategies) is identified which delivers increasingly better (but increasingly costly) approximations of the set of needed redexes. The entire idea is related to "strictness analysis" and to two concepts of sequentiality, that of G. Berry [in Transformations de programmes (Paris, 1978), 67–80, Dunod, Paris, 1978; MR 80k:68017] and that of G. Huet and J.-J. Levy ["Call by need computations in nonambiguous linear term rewriting systems", Rep. R 359, INRIA, Le Chesnay, 1979]. It is also shown to hold for an extension of the λ-calculus with strict operators.

An appendix relates the entire idea to the labelled λ-calculus of Levy [in λ-calculus and computer science theory (Rome, 1975), 147–165, Lecture Notes in Comput. Sci., 37, Springer, Berlin, 1975; MR 58 #5144]. Reviewed by J. P. Seldin.

Kennaway, J. R. (1989). Sequential evaluation strategies for parallel-or and related reduction systems. *Ann. Pure Appl. Logic*, 43(1):31–56.

[MR90m:03030] This paper deals with evaluation strategies for term rewriting systems, abstract reduction systems, and combinatory reduction systems. Combinatory reduction systems are defined and properties of such systems are discussed. A good strategy is a recursive normalising one-step evaluation strategy, i.e., a recursive function which chooses a redex for every term not in normal form, such that repeated reduction of chosen redexes will eventually arrive at the normal form of any term which has one. A general condition on abstract reduction systems is formulated. The author proves that the combinatory reduction systems of a large class satisfy this condition and that every abstract reduction system satisfying this general condition has a good strategy. Furthermore, a term rewriting system having the Church-Rosser property is given which has no good strategy. Reviewed by Hans-Joachim Goltz.

Klop, J. W. and de Vrijer, R. C. (1989). Unique normal forms for lambda calculus with surjective pairing. *Inform. and Comput.*, 80(2):97–113.

[MR90e:03016] This paper concerns the equational theory $\lambda\pi$ of λ-calculus extended with constants π, π_0, π_1, and axioms for surjective pairing: $\pi_i(\pi X_0 X_1) = X_i$ and $\pi(\pi_0 X)(\pi_1 X) = X$. It is well known that if the last axiom is omitted then the resulting system is a definitional extension of the pure λ-calculus, but that $\lambda\pi$ is not.

Equational λ-calculi are usually studied in conjunction with corresponding reduction systems. The obvious reduction system to associate with $\lambda\pi$ is $\lambda\pi^c$, which is obtained by adding to $\lambda\beta$-reduction ($\lambda\beta\eta$-reduction is not considered in the present paper) the reduction axioms $\pi_i(\pi X_0 X_1) \to X_i$ and $\pi(\pi_0 X)(\pi_1 X) \to X$. However, Klop [Combinatory reduction systems, Math. Centrum, Amsterdam, 1980; MR 83e:03026] showed that this system does not satisfy the Church-Rosser property. The presentation of the counterexample (without a complete proof) is the first main order of business of the paper.

The reason that the Church-Rosser theorem fails for $\lambda\pi^c$ is that in the axiom $\pi(\pi_0 X)(\pi_1 X) \to X$, the variable X appears twice; hence, if $X \to Y$, a redex $\pi(\pi_0 X)(\pi_1 X)$ can be contracted (by a contraction inside one of its arguments) to a term $\pi(\pi_0 X)(\pi_1 Y)$ which is not a redex. This phenomenon also occurs in the system $\lambda\delta^h$, which is obtained from the pure $\lambda\beta$-calculus by adding a constant δ and the axiom $\delta XX \to X$. This system also fails to satisfy the Church-Rosser theorem. Nevertheless, it has the property of uniqueness of normal forms (any two convertible normal forms are identical). This can be seen by considering the system $\lambda\delta^l$, which is obtained from $\lambda\delta^h$ by replacing the reduction axiom $\delta XX \to X$ by the rule $\delta XY \to X$ if $\lambda\delta \vdash X = Y$, where $\lambda\delta$ is the equational version of $\lambda\delta^h$ and is obtained from the $\lambda\beta$-calculus by adding the constant δ and the axiom $\delta XX = X$. Note that the axiom $\delta XX \to X$ is a special case of the rule $\delta XY \to X$ if $\lambda\delta \vdash X = Y$. Since this rule is stable under reductions inside its arguments, it is relatively easy to prove that the Church-Rosser theorem holds for $\lambda\delta^l$, and the conversion relation generated by both $\lambda\delta^h$ and $\lambda\delta^l$ is the conversion of $\lambda\delta$. The uniqueness of normal forms for $\lambda\delta^h$ follows from the Church-Rosser theorem for $\lambda\delta^l$ and the fact that $\lambda\delta^h$ and $\lambda\delta^l$ have the same normal forms. (The same result can be obtained by using, instead of $\lambda\delta^l$, the system $\lambda\delta^r$, which is defined by the rule $\delta XY \to Y$ if $\lambda\delta \vdash X = Y$.)

de Vrijer ["Surjective pairing and strong normalization: two themes in lambda calculus", Dissertation, Univ. Amsterdam, Amsterdam, 1987; per bibl.] used a modification of this technique to prove that $\lambda\pi$ is a conservative extension of the λ-calculus. The system $\lambda\pi^{lr}$ is defined by adding to $\lambda\beta$-calculus the contraction axioms $\pi_i(\pi X_0 X_1) > X_i$ and rules $\pi(\pi_0 X)Y > X$ if $\lambda\pi \vdash \pi_1 X = Y$ and $\pi Y(\pi_1 X) > X$ if $\lambda\pi \vdash \pi_0 X = Y$. An equivalence relation $\approx$ is defined as the least equivalence relation on the terms of $\lambda\pi$ satisfying the condition that $X_0 = Y_0 \,\&\, X_1 = Y_1 \Rightarrow C[\pi X_0 X_1] = C[\pi Y_0 Y_1]$ for any "context" $C[]$. In effect, $\approx$ disregards replacement occurrences of subterms in the scope of a π by convertible ones. (On pure λ-terms without occurrences of π, this is syntactic identity.) de Vrijer then proved a modified form of the Church-Rosser theorem: if $\lambda\pi \vdash M = N$, then there exist $\approx$-equivalent terms Q_0 and Q_1 such that $M \geq Q_0$ and $N \geq Q_1$. From this, it follows that $\lambda\pi$ is a conservative extension of the pure λ-calculus, that $\lambda\pi^{lr}$ satisfies the normal form property (any term which converts to a normal form reduces to it) and the uniqueness of normal forms, and (by showing that $\lambda\pi^c$ and $\lambda\pi^{lr}$ have the same normal forms) that $\lambda\pi^c$ satisfies the uniqueness of normal forms property.

The authors mention the following interesting open questions: (i) Is $\lambda\eta\pi$ ($\lambda\pi$ with the η-axiom) conservative over the $\lambda\beta\eta$-calculus? (ii) Does $\lambda\eta\pi^c$ satisfy the uniqueness of normal forms property? (iii) Does the Church-Rosser theorem hold for $\lambda\pi^{lr}$? (iv) Is there a restriction of the contraction relation $>$ of $\lambda\pi^{lr}$ which is decidable, generates the convertibility relation of $\lambda\pi$, satisfies the normal form property, and (perhaps) has the same normal forms as $\lambda\pi^{lr}$ (and hence as $\lambda\pi^c$)? And finally, the related question (v): Is there an effective normal form strategy for $\lambda\pi^{lr}$?

The authors conclude with some positive results for some related systems. (1) The Church-Rosser theorem and all the other results of interest mentioned in the paper hold for the typed versions of $\lambda\pi^c$ and $\lambda\eta\pi^c$ [see G. Pottinger, Notre Dame J. Formal Logic 22 (1981), no. 3, 264–268; MR 82j:03015]. (2) Another is the conjecture that the results mentioned in the paper also hold for corresponding systems of combinatory logic. Furthermore, if π_0 and π_1 are restricted to have only one argument and π to have only two arguments, then CLπ^c is confluent (by the results of Y. Toyama [J. Assoc. Comput. Mach. 34 (1987), no. 1, 128–143; MR 88e:68072]). A similar result holds for $\lambda\pi^c$ provided that the arguments of π_i and π are restricted to closed terms [see Klop, op. cit.]. (3) P. Chew [in Proceedings of the 13th Annual ACM Symposium on the Theory of Computing (Milwaukee, WI, 1981), 7–18, ACM, New York, 1981] showed that term rewriting systems including reduction rules with repeated occurrences of variables have unique normal forms provided that the left-hand sides of the rules satisfy a suitable "nonoverlapping" property. A corollary is that CLδ^n has this property. Chew's result also applies to CL plus applicative "parallel if"; i.e., CL with three new constants, C (for "conditional"), T (for "true"), and F (for "false") and the contraction rules $CTXY \to X$, $CFXY \to Y$, and $CZXX \to X$. (This case is substantially more complicated than CLδ^n.) This system is not confluent [Klop, op. cit.], but it becomes confluent if the number of arguments of C is restricted to 3. (4) The λ-calculus with Church's δ-reduction, which extends λ-calculus with the rules $\delta XX \to 0$ if X is a closed $\beta\delta$-normal form and $\delta XY \to 1$ if X and Y are closed $\beta\delta$-normal forms and $X \not\equiv Y$ (where 0 and 1 can be taken to be $\lambda xy.x$ and $\lambda xy.y$, respectively) and in which a $\beta\delta$-normal form is a term without β-redexes and without subterms of the form δXY, was shown to be confluent by G.

Mitschke ["Discriminators in combinatory logic and λ-calculus", Preprint No. 336, Tech. Hochsch. Darmstadt, Darmstadt, 1977; per bibl.]. (5) Related to the last example is a question posed in person by the reviewer to Klop in 1981: suppose in $\lambda\pi^c$, the rule $\pi(\pi_0 X)(\pi_1 X) \to X$ is restricted to the case that X is a closed $\lambda\pi^c$-normal form. The authors show that then confluence holds. Reviewed by J. P. Seldin.

Yokouchi, H. (1989). Church-Rosser theorem for a rewriting system on categorical combinators. *Theoret. Comput. Sci.*, 65(3):271–290.

[MR91a:68067] P.-L. Curien introduced rewriting systems of categorical combinators, inspired by the categorical semantics of lambda calculi [the author, Inform. and Control 69 (1986), no. 1, 188–254; MR 88f:03012]. One of Curien's systems, CCL β, which corresponds to the type-free lambda calculus with coupling, is not confluent. In this paper the author introduces the notion of a regular categorical combinator, and shows that the set of regular terms is closed under reduction and confluent. The definition of regularity and the proof are related with the subsystem SUB of CCL β. A paper by T. Hardin [Theoret. Comput. Sci. 65 (1989), no. 3, 291–342] (in which CCL β is named SL β) gave another such subset in the study of the correspondences between categorical combinators and lambda calculi, and extending this result the author also gives a hierarchy of subsets with the same properties. Reviewed by Teruo Hikita.

Barbanera, F. (1990). Combining term rewriting and type assignment systems. *Internat. J. Found. Comput. Sci.*, 1(3):165–184. Third Italian Conference on Theoretical Computer Science (Mantova, 1990).

[MR92c:68083] When combining term rewriting and lambda calculus systems, the fundamental properties of confluence and strong normalization may be disturbed. The disturbance appears to be less for typed lambda calculus systems. In a paper by V. Breazu-Tannen and J. Gallier [in Automata, languages and programming (Stresa, 1989), 137–150, Lecture Notes in Comput. Sci., 372, Springer, Berlin, 1989; MR 91d:68060], preservation of confluence and strong normalization has been proved for mixed systems when types are regarded as built into the terms. In the current paper, similar results are shown for combinations of term rewriting systems with simple and second order typed λ-calculi where typing is done by type assignment systems. One main theorem is that every term in such a combined system is $\beta\eta R$-strongly normalizable. This is proved by a variant of Tait's proof. A confluence preservation result can only be proved by disallowing η-reduction. However, in another of the main results the author shows that if the term rewriting relation is strongly normalizable and confluent on algebraic terms, then $\beta\eta R$-convertibility of mixed terms is decidable.

The paper is well organized. This new journal apparently does not edit for English, so some minor peculiarities of expression remain (e.g., "it descends that" for "it follows that"), although nothing really to impair understanding. For the entire collection see MR 91m:68005. Reviewed by John Pedersen.

Khasidashvili, Z. O. (1990). β-reductions and β-developments of λ-terms with the least number of steps. In *COLOG-88 (Tallinn, 1988)*, pages 105–111. Springer, Berlin.

[MR92a:03016] This paper gives an algorithm for determining the greatest lower

bound of the set of lengths of the reductions of a λ-term to normal form, as well as an algorithm for constructing a reduction strategy with this minimal length for the given term. For the entire collection see MR 91e:03004. Reviewed by Martin W. Bunder.

Klop, J. W. (1990). Term rewriting systems from Church-Rosser to Knuth-Bendix and beyond. In *Automata, languages and programming (Coventry, 1990)*, pages 350–369. Springer, New York.

[MR1076830] This item will not be reviewed individually. For details of the collection in which this item appears see 91e:68004 .

Yokouchi, H. and Hikita, T. (1990). A rewriting system for categorical combinators with multiple arguments. *SIAM J. Comput.*, 19(1):78–97. **[MR92b:03015]**

There is a well-known correspondence between typed lambda calculi and Cartesian closed categories (ccc's). In the untyped case, the ccc's are rather special: they must contain a reflexive object (i.e., one in which the function space $U \Rightarrow U$ is a retract of U, or sometimes even isomorphic to U if we wish extensionality). The translations of various such lambda calculi into ccc's then permit new kinds of functional formalisms for lambda terms, the so-called categorical combinators. The categorical formalism is of course based on composition as the primitive notion.

Alas, the translation to ccc's is far from unique, and many variations have been examined. The authors' bibliography cites numerous works on the subject. Care must be taken: there are many choices that could be made, e.g., (1) whether to have rule η, (2) whether to have products with surjective pairing, (3) whether to make the untyped translation a special case of the typed one, or to use special properties intrinsic to the untyped formalism, etc. For example, the interplay of products with the untyped formalism permits different interpretations, particularly in handling sequences of variables and lambda abstraction.

In this paper, the authors examine some extensions and variations of systems of categorical combinators due to Curien, with the emphasis on viewing them as rewriting systems. In particular, systems of such combinators are developed to handle multiple argument functions for practical computation purposes. On the other hand, in most versions of untyped lambda calculi with products, i.e., pairing and projections satisfying surjective pairing, the Church-Rosser (CR) property fails. The authors look at restricted subsets of terms for which CR holds, establish CR for their multiple argument combinators, and show how to extend these results to permit embeddings of their system into the Curien system. Numerous technically interesting remarks on translating into categorical combinators are discussed. Reviewed by Philip J. Scott.

Barbanera, F. (1991). Adding algebraic rewriting to the calculus of constructions: strong normalization preserved. In *Conditional and typed rewriting systems (Montreal, PQ, 1990)*, pages 260–271. Springer, Berlin.

[MR1138495] This item will not be reviewed individually. For details of the collection in which this item appears see 92g:68076 .

Dougherty, D. J. (1991). Adding algebraic rewriting to the untyped lambda calculus (extended abstract). In *Rewriting techniques and applications (Como, 1991)*, pages 37–48. Springer, Berlin.

[**MR1104468**] This item will not be reviewed individually. For details of the collection in which this item appears see 92b:68018 .

Nipkow, T. (1991). Higher-order unification, polymorphism, and subsorts (extended abstract). In *Conditional and typed rewriting systems (Montreal, PQ, 1990)*, pages 436–447. Springer, Berlin.

[**MR1138511**] This item will not be reviewed individually. For details of the collection in which this item appears see 92g:68076.

Bertot, Y. (1992). Origin functions in λ-calculus and term rewriting systems. In *CAAP '92 (Rennes, 1992)*, pages 49–65. Springer, Berlin.

[**MR94j:68028**] Summary: "We use origin functions to describe the notions of descendance and residuals in reduction systems such as the λ-calculus and linear term rewriting systems. We compare the origin functions for the λ-calculus and for term rewriting systems, $\lambda\sigma$ and λEnv, that implement this calculus. We show that the notions of origin do not correspond exactly, but we describe an extension of the notion of origin that permits the correct computation of λ-calculus origins for derivations of λEnv. We show that this extension is not sufficient to give the same result for $\lambda\sigma$ and we give another extension for that system. This work is interesting as it provides a distinction between the two term rewriting systems. This work has applications in the debugging of languages based on the λ-calculus or environments." For the entire collection see MR 94f:68009.

Dougherty, D. J. (1992). Adding algebraic rewriting to the untyped lambda calculus. *Inform. and Comput.*, 101(2):251–267.

[**MR94g:68060**] Summary: "We investigate the system obtained by adding an algebraic rewriting system R to an untyped lambda calculus in which terms are formed using the function symbols from R as constants. On certain classes of terms, called here 'stable', we prove that the resulting calculus is confluent if R is confluent, and is terminating if R is terminating. The termination result has the corresponding theorems for several typed calculi as corollaries. The proof of the confluence result suggests a general method for proving confluence of typed β-reduction plus rewriting; we sketch the application to the polymorphic lambda calculus."

Felty, A. (1992). A logic programming approach to implementing higher-order term rewriting. In *Extensions of logic programming (Stockholm, 1991)*, pages 135–161. Springer, Berlin.

[**MR94i:68141**] Summary: "Term rewriting has proven to be an important technique in theorem proving. In this paper, we illustrate that rewrite systems and strategies for higher-order term rewriting, which includes the usual notion of first-order rewriting, can be naturally specified and implemented in a higher-order logic programming language. We adopt a notion of higher-order rewrite system which uses the simply typed λ-calculus as the language for expressing rules, with a restriction on the occurrences of free variables on the left-hand sides of rules so that matching of terms with rewrite templates is decidable. The logic programming language contains an implementation of the simply typed lambda calculus including $\beta\eta$-conversion and higher-order unification. In addition, universal quantification in queries and the bodies of clauses is permitted. For higher-order rewriting, we show how these operations implemented at the meta-level provide elegant mechanisms for the object-level operations of descending through terms and matching terms with rewrite templates. We discuss tactic style theorem proving in this environment and illustrate how term rewriting strategies can be expressed as tactic-style search." For the entire collection see MR 94d:68002.

Jay, C. B. (1992). Modelling reduction in confluent categories. In *Applications of categories in computer science (Durham, 1991)*, pages 143–162. Cambridge Univ. Press, Cambridge.
[MR94h:68107] Confluent categories are introduced as a model of confluent rewriting systems. A category with ordered hom-sets is called confluent provided that the relation "to have a common upper bound" is transitive on each hom-set. Weakenings of the usual concepts of functor, natural transformation, adjoint, etc. are considered; e.g., a confluent functor F is defined like a functor except that, instead of preservation of composition, it is required that $F(fg)$ have a common upper bound with $F(f)F(g)$, and analogously with $F(\mathrm{id}_A)$ and id_{FA}. The corresponding theory is illustrated by the simply typed λ-calculus with pairing. For the entire collection see MR 93d:68001. Reviewed by Jiri Adamek.

Laneve, C. and Montanari, U. (1992). Axiomatizing permutation equivalence in the λ-calculus. In *Algebraic and logic programming (Volterra, 1992)*, pages 350–363. Springer, Berlin.
[MR1252870] This item will not be reviewed individually. For details of the collection in which this item appears see 94f:68005 .

Müller, F. (1992). Confluence of the lambda calculus with left-linear algebraic rewriting. *Inform. Process. Lett.*, 41(6):293–299.
[MR93f:68055] The author considers applicative term rewriting systems (ATRSs), i.e., the untyped λ-calculus with β-reduction, augmented with algebraic operations. These additional operations obey certain laws given as rewriting rules on terms. Then these rewriting rules are assumed to be left-linear and without any application of variables in their left-hand sides. Under these assumptions the combined reduction relation of an ATRS turns out to be confluent (Church-Rosser) whenever the added algebraic part alone is confluent. Reviewed by Peter R. J. Asveld.

Ogawa, M. (1992). Chew's theorem revisited: uniquely normalizing property of nonlinear term rewriting systems. In *Algorithms and computation (Nagoya, 1992)*, pages 309–318. Springer, Berlin.
[MR94j:68142] Summary: "This paper gives a purely syntactical proof, based on proof normalization techniques, of an extension of Chew's theorem. The main theorem is that a weakly compatible TRS is uniquely normalizing. Roughly speaking, the weakly compatible condition allows possibly nonlinear TRSs to have nonroot

overlapping rules that return the same results. This result implies the consistency of CL-pc, which is an extension of the combinatory logic CL with parallel-if rules." For the entire collection see MR 94h:68004.

Akama, Y. (1993). On Mints' reduction for ccc-calculus. In *Typed lambda calculi and applications (Utrecht, 1993)*, pages 1–12. Springer, Berlin.
[**MR94c:03018**] Summary: "In this paper, we present a divide-and-conquer lemma to infer the SN + CR (strong normalization and Church-Rosser) property of a reduction system from that property of its subsystems. Then we apply the lemma to show the property of Mints' reduction for ccc-calculus with restricted η-expansion and restricted π-expansion. In the course of the proof, we obtain some relations of the two restricted expanions vis-a-vis traditional reductions. Among other things, we get a simple characterization of the restricted η-expansion in terms of traditional β- and η-reductions, and a similar characterization for the restricted π-expansion." For the entire collection see MR 94b:68025.

Asperti, A. and Laneve, C. (1993a). Optimal reductions in interaction systems. In *TAPSOFT '93: theory and practice of software development (Orsay, 1993)*, pages 485–500. Springer, Berlin.
[**MR94f:68105**] Summary: "J. Lamping's optimal graph reduction technique for the λ-calculus is generalized to a new class of higher-order rewriting systems, called interaction systems. This provides a uniform description, in Lamping's style, of other basic computational constructs such as conditionals and recursion." For the entire collection see MR 94c:68008.

Asperti, A. and Laneve, C. (1993b). Paths, computations and labels in the λ-calculus. In *Rewriting techniques and applications (Montreal, PQ, 1993)*, pages 152–167. Springer, Berlin.
[**MR94j:03028**] Summary: "We provide a new characterization of J.-J. Levy's redex-families in the λ-calculus as suitable paths in the initial term of the derivation. The idea is that redexes in the same family are created by 'contraction' (via β-reduction) of a unique common path in the initial term. This fact gives new evidence about the 'common nature' of redexes in a family, and about the possibility of sharing their reduction. From this point of view, our characterization underlies all recent works on optimal graph reduction techniques for the λ-calculus, providing an original and intuitive understanding of optimal implementations.
"As an easy by-product, we prove that neither overlining nor underlining is required in Levy's labelling." For the entire collection see MR 94f:68028.

Barbanera, F. and Fernández, M. (1993a). Combining first and higher order rewrite systems with type assignment systems. In *Typed lambda calculi and applications (Utrecht, 1993)*, pages 60–74. Springer, Berlin.
[**MR94c:68104**] Summary: "General computational models obtained by integrating different specific models have recently become a stimulating and promising research subject for the computer science community. In particular, combinations of alge-

braic term rewriting systems and various λ-calculi, either typed or untyped, have been deeply investigated. In the present paper this subject is addressed from the point of view of type assignment. A powerful type assignment system, the one based on intersection types, is combined with term rewriting systems, first and higher order, and relevant properties of the resulting system, like strong normalization and confluence, are proved." For the entire collection see MR 94b:68025.

Barbanera, F. and Fernández, M. (1993b). Modularity of termination and confluence in combinations of rewrite systems with λ_ω. In *Automata, languages and programming (Lund, 1993)*, pages 657–668. Springer, Berlin.

[**MR94k:68098**] Summary: "We prove that termination and confluence are modular properties in combinations of the typed λ-calculus of order ω with (first or higher order) term rewrite systems, provided that the first order rewrite system is conservative (nonduplicating) and the higher order rewrite system satisfies some suitable conditions (the general schema) and does not introduce critical pairs." For the entire collection see MR 94h:68006.

Danos, V. and Regnier, L. (1993). Local and asynchronous beta-reduction (an analysis of Girard's execution formula). In *Eighth Annual IEEE Symposium on Logic in Computer Science (Montreal, PQ, 1993)*, pages 296–306. IEEE Comput. Soc. Press, Los Alamitos, CA.

[**MR94e:03011**] Summary: "We build a confluent, local, asynchronous reduction on λ-terms, using infinite objects (partial injections of Girard's algebra L^*), which is simple (only one move), intelligible (semantic setting of the reduction), general (based on a large-scale decomposition of β), and may be mechanized." For the entire collection see MR 94d:68005.

Dershowitz, N. and Mitra, S. (1993). Higher-order and semantic unification. In *Foundations of software technology and theoretical computer science (Bombay, 1993)*, pages 139–150. Springer, Berlin.

[**MR1295040**] This item will not be reviewed individually. For details of the collection in which this item appears see 95d:68003 .

Klop, J. W., van Oostrom, V., and van Raamsdonk, F. (1993). Combinatory reduction systems: introduction and survey. *Theoret. Comput. Sci.*, 121(1-2):279–308. A collection of contributions in honour of Corrado Böhm on the occasion of his 70th birthday.

[**MR94k:68100**] Summary: "Combinatory reduction systems, or CRSs for short, were designed to combine the usual first-order format of term rewriting with the presence of bound variables as in pure λ-calculus and various typed λ-calculi. Bound variables are also present in many other rewrite systems, such as systems with simplification rules for proof normalization. The original idea of CRSs is due to Aczel, who introduced a restricted class of CRSs and, under the assumption of orthogonality, proved confluence. Orthogonality means that the rules are nonambiguous (no overlap leading to a critical pair) and left-linear (no global comparison of terms necessary).

"We introduce the class of orthogonal CRSs, illustrated with many examples, discuss its expressive power, and give an outline of a short proof of confluence. This proof is a direct generalization of Aczel's original proof, which is close to the well-known confluence proof for λ-calculus by Tait and Martin-Löf. There is a well-known connection between the parallel reduction which plays a role in the latter proof and the concept of 'developments', and a classical lemma in the theory of λ-calculus is the one on 'finite developments', a strong normalization result. It turns out that the notion of 'parallel reduction' used in Aczel's proof gives rise to a generalized form of developments which we call 'superdevelopments' and on which we briefly comment." For the entire collection see MR 94e:68005.

Nipkow, T. (1993). Orthogonal higher-order rewrite systems are confluent. In *Typed lambda calculi and applications (Utrecht, 1993)*, pages 306–317. Springer, Berlin.
[MR1233156] This item will not be reviewed individually. For details of the collection in which this item appears see 94b:68025 .

van Bakel, S. (1993). Partial intersection type assignment in applicative term rewriting systems. In *Typed lambda calculi and applications (Utrecht, 1993)*, pages 29–44. Springer, Berlin.
[MR94j:68131] Summary: "This paper introduces a notion of partial type assignment on applicative term rewriting systems that is based on a combination of an essential intersection type assignment system, and the type assignment system as defined for ML [R. Milner, J. Comput. System Sci. 17 (1978), no. 3, 348–375; MR 81g:68022], both extensions of Curry's type assignment system [H. B. Curry and R. Feys, Combinatory logic, Vol. I, North-Holland, Amsterdam, 1958; MR 20 #817]. Terms and rewrite rules are written as trees, and type assignment consists of assigning intersection types function symbols, and specifying the way in which types can be assigned to nodes and edges between nodes. The only constraints on this system are local: they are imposed by the relation between the type assigned to a node and those assigned to its incoming and outgoing edges. In general, given an arbitrary typeable applicative term rewriting system, the subject reduction property does not hold. We formulate a sufficient but undecidable condition typeable rewrite rules should satisfy in order for this property to hold." For the entire collection see MR 94b:68025.

van Raamsdonk, F. (1993). Confluence and superdevelopments. In *Rewriting techniques and applications (Montreal, PQ, 1993)*, pages 168–182. Springer, Berlin.
[MR94h:68113] Summary: "In this paper a short proof is presented for confluence of a quite general class of reduction systems, containing λ-calculus and term rewrite systems: the orthogonal combinatory reduction systems. Combinatory reduction systems (CRSs for short) were introduced by Klop, generalizing an idea of Aczel. In CRSs, the usual first-order term rewriting format is extended with binding structures for variables. This permits us to express, besides first order term rewriting, also λ-calculus, extensions of λ-calculus and proof normalizations. Confluence

is proved for orthogonal CRSs, that is, for those CRSs having left-linear rules and no critical pairs. The proof proceeds along the lines of the proof of Tait and Martin-Löf for confluence of λ-calculus, but uses a different notion of 'parallel reduction' as employed by Aczel. It gives rise to an extended notion of development, called 'superdevelopment'. A superdevelopment is a reduction sequence in which, besides redexes that descend from the initial term, also some redexes that are created during reduction may be contracted. For the case of λ-calculus, all superdevelopments are proved to be finite. A link with the confluence proof is provided by proving that superdevelopments characterize exactly Aczel's notion of 'parallel reduction' used in order to obtain confluence." For the entire collection see MR 94f:68028.

Ariola, Z., Kennaway, R., Klop, J. W., Sleep, R., and de Vries, F.-J. (1994). Syntactic definitions of undefined: on defining the undefined. In *Theoretical aspects of computer software (Sendai, 1994)*, pages 543–554. Springer, Berlin.

[**MR1296691**] This item will not be reviewed individually. For details of the collection in which this item appears see 95d:68006 .

Breazu-Tannen, V. and Gallier, J. (1994). Polymorphic rewriting conserves algebraic confluence. *Inform. and Comput.*, 114(1):1–29.

[**MR95i:68066**] The authors study in this paper combinations of many-sorted algebraic term rewriting systems and polymorphic lambda term rewriting with respect to the preservation of the confluence property of algebraic rewriting. It turns out that strongly normalizing type disciplines interact nicely with algebraic data type specifications.

Algebraic and lambda terms are mixed by adding the symbols of the algebraic signature to the polymorphic lambda calculus as higher-order constants. It is shown that if a many-sorted algebraic rewrite system R is confluent, then $R+\beta+\text{type-}\beta+\text{type-}\eta$ rewriting of mixed terms is confluent too. In contrast η reduction does not commute with algebraic reduction, in general. However, using long normal forms, it is shown that if R is confluent and strongly normalizing (canonical) then equational provability from $R+\beta+\eta+\text{type-}\beta+\text{type-}\eta$ is still decidable. Reviewed by Klaus Madlener.

Gardner, P. (1994). Discovering needed reductions using type theory. In *Theoretical aspects of computer software (Sendai, 1994)*, pages 555–574. Springer, Berlin.

[**MR1296692**] This item will not be reviewed individually. For details of the collection in which this item appears see 95d:68006.

Khasidashvili, Z. (1994). The longest perpetual reductions in orthogonal expression reduction systems. In *Logical foundations of computer science (St. Petersburg, 1994)*, pages 191–203. Springer, Berlin.

[**MR95m:68094**] Summary: "We consider reductions in orthogonal expression reduction systems (OERS), that is, orthogonal term rewriting systems with bound variables and substitutions, as in the λ-calculus. We design a strategy that for any

given term t constructs a longest reduction starting from t if t is strongly normalizable, and constructs an infinite reduction otherwise. The conservation theorem for OERS follows easily from the properties of the strategy. We develop a method for computing the length of a longest reduction starting from a strongly normalizable term. We study properties of pure substitutions and several kinds of similarity of redexes. We apply these results to construct an algorithm for computing lengths of longest reductions in strongly persistent OERS that does not require actual transformations of the input term. As a corollary, we have an algorithm for computing lengths of longest developments in OERS." For the entire collection see MR 95j:03003.

Nesi, M., de Paiva, V., and Ritter, E. (1994). Rewriting properties of combinators for rudimentary linear logic. In *Higher-order algebra, logic, and term rewriting (Amsterdam, 1993)*, pages 256–275. Springer, Berlin.
[MR96d:68117] From the text: "In this paper we investigate the possibility of developing a (semi-)automatic rewriting tool for manipulating and reasoning about combinators for intuitionistic linear logic. In particular, we develop a term rewriting system for a theory of categorical combinators for rudimentary linear logic. We make use of the Knuth-Bendix completion algorithm to transform the equational theory for the combinators into an equivalent canonical rewrite system. This means that a set of categorical combinators for linear logic has first to be derived, and then the resulting system of combinators can be checked for rewriting properties using rewriting techniques.

"The paper is organized as follows. First, we mention some basic facts about intuitionistic linear logic, categorical combinators and term rewriting systems. We then describe how our system of categorical combinators has been derived. The next sections discuss some of the rewriting properties of such a system, namely termination and normal forms. Finally, we draw some conclusions and mention some further work." For the entire collection see MR 95j:68012.

van Oostrom, V. (1994). Confluence by decreasing diagrams. *Theoret. Comput. Sci.*, 126(2):259–280.
[MR95j:68080] This paper gives a confluence theorem for abstract reduction systems which unifies several known confluence criteria. The author introduces a notion of decreasing diagram using the multiset ordering. For finite multisets M and N over a partially ordered set I, $M \prec N$ if M is obtained from N by replacing an element of N by smaller elements of I. This ordering was used by N. Dershowitz and Z. Manna [Comm. ACM 22 (1979), no. 8, 465–476; MR 80g:68017] to prove termination of programs and rewriting systems.

Let $\mathcal{A} = (\mathcal{A}, \langle \to_\alpha \rangle_{\alpha \in \mathcal{I}})$ be an abstract reduction system consisting of reduction relations $\to_\alpha$ labelled by $\alpha \in I$. A reduction sequence $x \twoheadrightarrow y$ is labelled by a string $\sigma \in I^*$. For $\sigma \in I^*$ define a multiset $|\sigma|$ by $|\sigma| = []$ if $\sigma = \epsilon$ and by $|\sigma| = [\alpha] \cup (|\tau| - \curlyvee\alpha)$ if $\sigma = \alpha\tau$, where $\curlyvee\alpha = \{\beta \in I | \ \beta \prec \alpha\}$. A diagram ($x \overset{\sigma}{\to}\!\!\twoheadrightarrow y$, $x \overset{\tau}{\to}\!\!\twoheadrightarrow z$, $z \overset{\sigma'}{\to}\!\!\twoheadrightarrow w$, $y \overset{\tau'}{\to}\!\!\twoheadrightarrow w$) is called decreasing if $|\sigma\tau'| \preceq [\sigma, \tau] \succeq |\tau\sigma'|$. $\mathcal{A}$ is said to be locally decreasing if there are subsets I_v and I_h of I such that for any pair of reductions $x \overset{\alpha}{\to}\!\!\to y$ and $x \overset{\beta}{\to}\!\!\to z$, there are sequences of reductions $z \overset{\sigma}{\to}\!\!\twoheadrightarrow w$ and $y \overset{\tau}{\to}\!\!\twoheadrightarrow w$ with $\sigma \in I_v^*$, $\tau \in I_h^*$ such that $(\alpha, \beta, \sigma, \tau)$ forms a decreasing diagram. Now, the main theorem asserts that a locally decreasing system is confluent. Reviewed by Yuji Kobayashi.

van Oostrom, V. and van Raamsdonk, F. (1994). Weak orthogonality implies confluence: the higher-order case. In *Logical foundations of computer science (St. Petersburg, 1994)*, pages 379–392. Springer, Berlin.
[MR96b:68093] Summary: "In this paper we prove confluence for weakly orthogonal higher-order rewriting systems. This generalises all the known 'confluence by orthogonality' results." For the entire collection see MR 95j:03003.

Asperti, A. and Laneve, C. (1995a). Comparing λ-calculus translations in sharing graphs. In *Typed lambda calculi and applications (Edinburgh, 1995)*, pages 1–15. Springer, Berlin.
[MR1477971] This item will not be reviewed individually. For details of the collection in which this item appears see 98e:03020.

Asperti, A. and Laneve, C. (1995b). Paths, computations and labels in the λ-calculus. *Theoret. Comput. Sci.*, 142(2):277–297. Selected papers of the Fifth International Conference on Rewriting Techniques and Applications (Montreal, PQ, 1993).
[MR96h:03032] Summary: "We provide a new characterization of Levy redex-families in the λ-calculus as suitable paths in the initial term of the derivation. The idea is that redexes in the same family are created by 'contraction' (via β-reduction) of a unique common path in the initial term. This fact gives new evidence about the 'common nature' of redexes in the same family, and about the possibility of sharing their reduction. In general, paths seem to provide a very friendly and intuitive tool for reasoning about redex-families, as well in theory (using paths, we shall provide a remarkably simple proof of the equivalence between extraction and labeling) as in practice (our characterization underlies recent work on optimal graph reduction techniques for the λ-calculus (Lamping), providing an original and intuitive understanding of optimal implementations). Finally, as an easy by-product of the path-characterization, we prove that neither overlining nor underlining are required in Levy's labeling." For the entire collection see MR 96b:68005.

Briaud, D. (1995). An explicit Eta rewrite rule. In *Typed lambda calculi and applications (Edinburgh, 1995)*, pages 94–108. Springer, Berlin.
[MR1477977] This item will not be reviewed individually. For details of the collection in which this item appears see 98e:03020.

Dougherty, D. J. and Johann, P. (1995). A combinatory logic approach to higher-order E-unification. *Theoret. Comput. Sci.*, 139(1-2):207–242.
[MR96d:68113] Summary: "Let E be a first-order equational theory. A translation of higher-order E-unification problems into a combinatory logic framework is presented and justified. The case in which E admits presentation as a convergent term rewriting system is treated in detail: in this situation, a modification of ordinary narrowing is shown to be a complete method for enumerating higher-order E-unifiers. In fact, we treat a more general problem, in which the types of terms contain type variables." Reviewed by Zhenyu Qian.

Jay, C. B. and Ghani, N. (1995). The virtues of eta-expansion. *J. Funct. Programming*, 5(2):135–154.

[MR97a:68089] Summary: "Interpreting η-conversion as an expansion rule in the simply-typed λ-calculus maintains the confluence of reduction in a richer type structure. This use of expansions is supported by categorical models of reduction, where β-contraction, as the local counit, and η-expansion, as the local unit, are linked by local triangle laws. The latter form reduction loops, but strong normalization (to the long $\beta\eta$-normal forms) can be recovered by 'cutting' the loops." Reviewed by Ralph W. Wilkerson.

Kuske, S. (1995). Implementing β-reduction by hypergraph rewriting. In *Joint COMPUGRAPH/SEMAGRAPH Workshop on Graph Rewriting and Computation (SEGRAGRA '95) (Volterra, 1995)*, page 8 pp. (electronic). Elsevier, Amsterdam.

[MR1489277] This item will not be reviewed individually. For details of the collection in which this item appears see 98h:68012 .

Asperti, A. and Laneve, C. (1996). Interaction systems. II. The practice of optimal reductions. *Theoret. Comput. Sci.*, 159(2):191–244.

[MR97d:68124] Summary: "Lamping's optimal graph reduction technique for the λ-calculus is generalized to a new class of higher-order rewriting systems, called interaction systems. Interaction systems provide a nice integration of the functional paradigm with a rich class of data structures (all inductive types), and some basic control flow constructs such as conditionals and (primitive or general) recursion. We describe a uniform and optimal implementation, in Lamping's style, for all these features. The paper is the natural continuation of Part I [Math. Structures Comput. Sci. 4 (1994), no. 4, 457–504; MR 96h:68104], where we focused on the theoretical aspects of optimal reductions in interaction systems (family relation, labeling, extraction)."

Coquand, C. and Coquand, T. (1996). On the definition of reduction for infinite terms. *C. R. Acad. Sci. Paris Sér. I Math.*, 323(5):553–558.

[MR97d:68125] Summary: "The original motivation of this work was to give a co-inductive definition of reduction between infinite terms. We define a special reduction relation, so-called semi-standard reduction, which is used to prove the main properties of reduction between infinite terms. For instance, though this reduction is not Church-Rosser, a normal form, if it exists, is uniquely determined. This notion of semi-standard reduction is interesting even for finite terms and allows for a simple proof of the standardisation theorem."

Danos, V. and Regnier, L. (1996). Reversible, irreversible and optimal λ-machines (extended abstract). In *Linear logic 96 (Tokyo)*, page 21 pp. (electronic). Elsevier, Amsterdam.

[MR1488165] This item will not be reviewed.

Ferreira, M. C. F., Kesner, D., and Puel, L. (1996). λ-calculi with explicit substitutions and composition which preserve β-strong normalization (extended abstract). In *Algebraic and logic programming (Aachen, 1996)*, pages 284–298. Springer, Berlin.

[**MR1462914**] This item will not be reviewed individually. For details of the collection in which this item appears see 98c:68005.

Glauert, J. and Khasidashvili, Z. (1996). Minimal relative normalization in orthogonal expression reduction systems. In *Foundations of software technology and theoretical computer science (Hyderabad, 1996)*, pages 238–249. Springer, Berlin.

[**MR99a:68098**] Summary: "In previous papers, the authors studied normalization relative to desirable sets $\mathcal{S}$ of 'partial results', where it was shown that such sets must be stable. For example, the sets of normal forms, head-normal-forms, and weak head-normal-forms in the λ-calculus are all stable. They showed that, for any stable $\mathcal{S}$, $\mathcal{S}$-needed reductions are $\mathcal{S}$-normalizing. This paper continues the investigation into the theory of relative normalization. In particular, we prove existence of minimal normalizing reductions for regular stable sets of results. All the above-mentioned sets are regular. We give a sufficient and necessary criterion for a normalizing reduction (with respect to a regular stable $\mathcal{S}$) to be minimal. Finally, we establish a relationship between relative minimal and optimal reductions, revealing a conflict between minimality and optimality: for regular stable sets of results, a term need not possess a reduction that is minimal and optimal at the same time." For the entire collection see MR 98i:68006.

Hilken, B. P. (1996). Towards a proof theory of rewriting: the simply typed 2λ-calculus. *Theoret. Comput. Sci.*, 170(1-2):407–444.

[**MR97j:68068**] Summary: "This paper describes the simply typed 2λ-calculus, a language with three levels: types, terms and rewrites. The types and terms are those of the simply typed λ-calculus, and the rewrites are expressions denoting sequences of β-reductions and η-expansions. An equational theory is imposed on the rewrites, based on 2-categorical justifications, and the word problem for this theory is solved by finding a canonical expression in each equivalence class.

"The canonical form of rewrites allows us to prove several properties of the calculus, including a strong form of confluence and a classification of the long-$\beta\eta$-normal forms in terms of their rewrites. Finally we use these properties as the basic definitions of a theory of categorical rewriting, and find that the expected relationships among confluence, strong normalisation and normal forms hold."

Kahrs, S. (1996). The variable containment problem. In *Higher-order algebra, logic, and term rewriting (Paderborn, 1995)*, pages 109–123. Springer, Berlin.

[**MR97g:68112**] Summary: "The essentially free variables of a term t in some λ-calculus, $\mathrm{FV}_\beta(t)$, form the set $\{x|\forall u, t =_\beta u \Rightarrow x \in \mathrm{FV}(u)\}$. This set is significant once we consider equivalence classes of λ-terms rather than λ-terms themselves, as for instance in higher-order rewriting.

"An important problem for (generalised) higher-order rewrite systems is the variable

containment problem: given two terms t and u, do we have for all substitutions θ and contexts $C[]$ that $\mathrm{FV}_\beta(C[t^\theta]) \supseteq \mathrm{FV}_\beta(C[u^\theta])$? This property is important when we want to consider $t \to u$ as a rewrite rule and keep n-step rewriting decidable. Variable containment is in general not implied by $\mathrm{FV}_\beta(t) \supseteq \mathrm{FV}_\beta(u)$. We give a decision procedure for the variable containment problem of the second-order fragment of $\lambda^\to$. For full $\lambda^\to$ we show the equivalence of variable containment to an open problem in the theory of PCF; this equivalence also shows that the problem is decidable in the third-order case." For the entire collection see MR 97f:68005.

Kennaway, R., van Oostrom, V., and de Vries, F.-J. (1996). Meaningless terms in rewriting. In *Algebraic and logic programming (Aachen, 1996)*, pages 254–268. Springer, Berlin.

[MR98d:68115] Summary: "We present an axiomatic approach to meaninglessness in finite and transfinite term rewriting and lambda calculus. We justify our axioms in two ways. First, they are shown to imply important properties of meaninglessness: genericity of the class of meaningless terms, the consistency of equating all meaningless terms, and the construction of Böhm trees. Second we show that they can be easily verified for existing notions of meaninglessness." For the entire collection see MR 98c:68005.

Laneve, C. and Montanari, U. (1996). Axiomatizing permutation equivalence. *Math. Structures Comput. Sci.*, 6(3):219–249.

[MR97k:68095] This paper generalizes a result of Laneve and Montanari [in Algebraic and logic programming (Volterra, 1992), 350–363, Lecture Notes in Comput. Sci., 632, Springer, Berlin, 1992; see MR 94f:68005], where permutation equivalence is axiomatized for λ-calculus; the axiomatization is based on the approach proposed in a paper by J. Meseguer and Montanari [Inform. and Comput. 88 (1990), no. 2, 105–155; MR 91h:68140]. As a consequence, the definition of permutation equivalence does not need residual calculi.

The basic idea of the approach of Meseguer and Montanari [op. cit.] is that the algebraic structure of terms is extended to derivations; the desired equivalence is then obtained by adding suitable axioms. It is shown that Meseguer's axioms completely define permutation equivalence in the case of term rewriting systems. Then it is shown, by a counterexample, that this result does not hold for the more general case of term rewriting modulo a set of axioms. Subsequently it is demonstrated that the axiomatization proposed by Meseguer can be made complete (by adding extra axioms) for permutation equivalence in orthogonal combinatory reduction systems. To this aim it is necessary to flatten an OCRS into a TRS where substitution is given the status of an algebraic operation. Reviewed by Dirk Janssens.

Sørensen, M. H. (1996). Effective longest and infinite reduction paths in untyped λ-calculi. In *Trees in algebra and programming—CAAP '96 (Linköping, 1996)*, pages 287–301. Springer, Berlin.

[MR97e:68079] Summary: "A maximal reduction strategy in untyped λ-calculus computes for a term a longest (finite or infinite) reduction path. Some types of reduction strategies in untyped λ-calculus have been studied, but maximal strategies have

received less attention. We give a systematic study of maximal strategies, recasting the few known results in our framework and giving a number of new results, the most important of which is an effective maximal strategy in $\lambda\beta\eta$. We also present a number of applications illustrating the relevance and usefulness of maximal strategies." For the entire collection see MR 97d:68007.

van de Pol, J. (1996). Two different strong normalization proofs? Computability versus functionals of finite type. In *Higher-order algebra, logic, and term rewriting (Paderborn, 1995)*, pages 201–220. Springer, Berlin.

[**MR97g:03056**] Summary: "A proof of $\forall t \ni n\, \mathrm{SN}(t, n)$ (term t performs at most n reduction steps) is given, based on strong computability predicates. By using modified realizability, a bound on reduction lengths is extracted from it. This upper bound is compared with the one R. O. Gandy defined, using strictly monotonic functionals. This reveals a remarkable connection between his proof and W. W. Tait's. We show the details for simply typed λ-calculus and Gödel's system T. For the latter system, program extraction yields considerably sharper upper bounds." For the entire collection see MR 97f:68005.

Barbanera, F., Fernández, M., and Geuvers, H. (1997). Modularity of strong normalization in the algebraic-λ-cube. *J. Funct. Programming*, 7(6):613–660.

[**MR98k:03032**] Summary: "We present the algebraic-λ-cube, an extension of Barendregt's λ-cube with first- and higher-order algebraic rewriting. We show that strong normalization is a modular property of all the systems in the algebraic-λ-cube, provided that the first-order rewrite rules are nonduplicating and the higher-order rules satisfy the general schema of J.-P. Jouannaud and M. Okada. We also prove that local confluence is a modular property of all the systems in the algebraic-λ-cube, provided that the higher-order rules do not introduce critical pairs. This property and the strong normalization result imply the modularity of confluence."

Khasidashvili, Z. and Ogawa, M. (1997). Perpetuality and uniform normalization. In *Algebraic and logic programming (Southampton, 1997)*, pages 240–255. Springer, Berlin.

[**MR1639639**] This item will not be reviewed individually. For details of the collection in which this item appears see 1 639 623 .

Melliès, P.-A. (1997). Axiomatic rewriting theory. III. A factorisation theorem in rewriting theory. In *Category theory and computer science (Santa Margherita Ligure, 1997)*, pages 49–68. Springer, Berlin.

[**MR1640337**] A review for this item is in process.

Sørensen, M. H. (1997). Strong normalization from weak normalization in typed λ-calculi. *Inform. and Comput.*, 133(1):35–71.

[**MR97k:03017**] Summary: "For some typed λ-calculi it is easier to prove weak normalization than strong normalization. Techniques to infer the latter from the former have been invented over the last twenty years by R. Nederpelt, J. W. Klop, Z. Khasidashvili, M. Karr, P. de Groote, and A. J. Kfoury and J. Wells. However, these techniques infer strong normalization of one notion of reduction from weak normalization of a more complicated notion of reduction. This paper presents a new technique to infer strong normalization of a notion of reduction in a typed λ-calculus from weak normalization of the same notion of reduction. The technique is demonstrated to work on some well-known systems, including second-order λ-calculus and the system of positive recursive types. It gives hope for a positive answer to the Barendregt-Geuvers conjecture stating that every pure type system which is weakly normalizing is also strongly normalizing. The paper also analyzes the relationship between the techniques mentioned above, and reviews, in less detail, other techniques for proving strong normalization."

Guerrini, S. (1998). Sharing-graphs, sharing-morphisms, and (optimal) λ-graph reductions. In *The Tbilisi Symposium on Logic, Language and Computation: selected papers (Gudauri, 1995)*, pages 317–332. CSLI Publ., Stanford, CA.
[MR1682622] A review for this item is in process.

Melliès, P.-A. (1998). A stability theorem in rewriting theory. In *Thirteenth Annual IEEE Symposium on Logic in Computer Science (Indianapolis, IN, 1998)*, pages 287–298. IEEE Computer Soc., Los Alamitos, CA.
[MR99j:68062] Summary: "One key property of the λ-calculus is that there exists a minimal computation (the head-reduction) $M \overset{e}{\twoheadrightarrow} V$ from a λ-term M to the set of its head-normal forms. Minimality here means categorical 'reflectivity', i.e., that every reduction path $M \overset{f}{\twoheadrightarrow} W$ to a head-normal form W factors (up to redex permutation) to a path $M \overset{e}{\twoheadrightarrow} V \overset{h}{\twoheadrightarrow} W$. This paper establishes a stability a la Berry or poly-reflectivity theorem which extends the minimality property to rewriting systems with critical pairs. The theorem is proved in the setting of axiomatic rewriting systems where sets of head-normal forms are characterised by their frontier property, in the spirit of an earlier paper by J. Glauert and Z. Khasidashvili." For the entire collection see MR 99h:68008.

Ferreira, M. C. F., Kesner, D., and Puel, L. (1999). λ-calculi with explicit substitutions preserving strong normalization. *Appl. Algebra Engrg. Comm. Comput.*, 9(4):333–371.
[MR1683292] A review for this item is in process.

Intrigila, B. and Venturini Zilli, M. (1999). Orders, reduction graphs and spectra. *Theoret. Comput. Sci.*, 212(1-2):211–231. Gentzen (Rome, 1996).
[MR1672084] A review for this item is in process.

van Raamsdonk, F., Severi, P., Sørensen, M. H. B., and Xi, H. (1999). Perpetual reductions in λ-calculus. *Inform. and Comput.*, 149(2):173–

225.
[**MR1675817**] A review for this item is in process.

Vrijer, R. C. d. (1999). Conditional linearization. *Indag. Math. (N.S.)*, 10(1):145–159.
[**MR1691431**] A review for this item is in process.

1.8 Infinitary lambda calculus

Mitschke, G. (1980). Infinite terms and infinite reductions. In *To H. B. Curry: essays on combinatory logic, lambda calculus and formalism*, pages 243–257. Academic Press, London.
[**MR82g:03021**] Reducibility in the ordinary λ-calculus can be defined in two different ways – either by means of replacement rules which specify which sequences of terms are reductions [cf., for example, J.R. Hindley, B. Lercher and J.P. Seldin, Introduction to combinatory logic, see p. 8, Cambridge Univ. Press, London, 1972; MR 49 #25] or axiomatically [e.g., op. cit., see pp. 9 - 10]. Although various applications for systems involving infinite terms have been found since W.W. Tait's paper [Formal systems and recursive functions (Oxford, 1963), pp. 176 - 185, North-Holland, Amsterdam, 1965; MR 33 #3925], reducibility relations for such systems have hitherto been defined only in the second of these two ways. The author shows how to introduce a notion of reduction appropriate to the infinite case. Beginning with the system of W. Maass [ISILC Proof Theory Symposium (Kiel, 1974), pp. 257 - 263, Lecture Notes in Math., 500, Springer, Berlin, 1975; MR 53 #10555], he first provides a measure of the complexity of proofs of reducibility statements by assigning countable ordinals to such proofs. Reductions are then taken to be sequences of pairs of terms and redex occurrences, also indexed by means of countable ordinals. Using the complexity measure previously defined, the author specifies the terminating reduction sequences by means of a transfinite induction over the countable ordinals. A nonterminating sequence is then taken to be a sequence indexed by a limit number which is not a terminating sequence but every proper initial segment of which is a terminating sequence. Using these definitions, the author shows that if a term reduces to a finite term, then there is a finite reduction leading from one to the other. He also shows how to extend the methods of Curry and R. Feys [Combinatory logic, Vol. 1, North-Holland, Amsterdam, 1958; MR 20 #817; J. Symbolic Logic 34 (1969), 545 - 560; MR 46 #1578] so as to obtain a proof of the Church-Rosser theorem. (Reviewer's remarks: These results are certainly interesting, but they are less than completely satisfying because the definition of reduction sequences depends so heavily on the axiomatic definition of reducibility which the author borrows from Maass (op. cit.). This is a distinct contrast with the situation in the finite case, and the point is to provide analogies, not contrasts.) (For the entire collection see MR 81j:03005.) Reviewed by Garrel Pottinger.

Beylin, I. and Diskin, Z. (1994). Introducing infinitary lambda calculus. In *Mathematics*, pages 11–22. Latv. Univ., Riga.

[**MR96k:03034**] Summary: "The paper introduces a notion of infinitary lambda calculus. Whereas in the ordinary λ-calculus functions can be applied only to a single argument, our version allows multiple (even infinite) applications and abstractions. So some λ-terms can involve infinitely many (or even all) variables, which makes the syntactical machinery rather subtle. An algebraic semantics for the new calculus is constructed and the corresponding completeness and representation theorems are presented."For the entire collection see MR 96j:00012.

Berarducci, A. (1996). Infinite λ-calculus and non-sensible models. In *Logic and algebra (Pontignano, 1994)*, pages 339–377. Dekker, New York.

[**MR97h:03016**] The first time I encountered Berarducci's infinite λ-calculus was in Torino, in February 1994, at the BRA-Esprit meeting "Common Foundations of Functional and Logic Programming". I was immediately impressed by the deep new ideas that are at the basis of the construction of this model, and that were illustrated in a fascinating way by Berarducci's talk. After that I studied the paper as soon as I obtained a preprint, reinforcing my initial opinion.

I think that this is an outstanding work on the forefront of λ-models. It is gratifying to see papers such as this giving a really new and interesting contribution. The proposed λ-model takes into account the "asymptotic behaviors" of terms through their infinite normal forms. This throws light upon the inner structure of the unsolvable terms in a completely original way. The above construction naturally leads to the notion of mute terms, which have a totally undefined operational behavior. Moreover, some open problems on easy terms are solved using the analysis of infinite normal forms. The paper is carefully written in a clear and pleasant style, in spite of the highly technical matter. For the entire collection see MR 97c:00047. Reviewed by Mariangiola Dezani-Ciancaglini.

Berarducci, A. and Intrigila, B. (1996). Church-Rosser λ-theories, infinite λ-terms and consistency problems. In *Logic: from foundations to applications (Staffordshire, 1993)*, pages 33–58. Oxford Univ. Press, New York.

[**MR97m:03027**] Berarducci has previously introduced an infinitary version of λ-calculus which allows terms that include infinite applications, in which an infinite number of subterms can be reduced at once.

Here this is used to define a large class of standard λ-terms which can be equated with every other term, without leading to triviality. The Church-Rosser property, for various forms of reduction, is maintained. For the entire collection see MR 97g:03002. Reviewed by Martin W. Bunder.

Kennaway, J. R., Klop, J. W., Sleep, M. R., and de Vries, F. J. (1997). Infinitary lambda calculus. *Theoret. Comput. Sci.*, 175(1):93–125. Non-standard logics and logical aspects of computer science (Kanazawa, 1994).

[**MR98c:03038**] Summary: "In a previous paper [Inform. and Comput. 119 (1995), no. 1, 18–38; MR 96g:68062] we established the theory of transfinite reduction for orthogonal term rewriting systems. In this paper we perform the same task for the lambda calculus.

"From the viewpoint of infinitary rewriting, the Böhm model of the lambda calculus can be seen as an infinitary term model. In contrast to term rewriting, there are

several different possible notions of infinite term, which give rise to different Böhm-like models, which embody different notions of lazy or eager computation." For the entire collection see MR 97k:68009.

Berarducci, A. and Dezani-Ciancaglini, M. (1999). Infinite λ-calculus and types. *Theoret. Comput. Sci.*, 212(1-2):29–75. Gentzen (Rome, 1996).
[**MR1672060**] A review for this item is in process.

1.9 Explicit substitution

de Bruijn, N. G. (1978). Lambda calculus with namefree formulas involving symbols that represent reference transforming mappings. *Nederl. Akad. Wetensch. Indag. Math.*, 40(3):348–356.
[**MR80b:03016**] This is a generalization of an earlier article by the author (same journal 34 (1972), 381 - 392; MR 48 #71). Reviewed by A. S. Kuzichev

Balsters, H. (1987). Lambda calculus extended with segments. In *Mathematical logic and theoretical computer science (College Park, Md., 1984–1985)*, pages 15–27. Dekker, New York.
[**MR89g:03015**] From the introduction: "The aim of this paper is to give a description of a system of λ-calculus called $\lambda\sigma$. The system $\lambda\sigma$ is an extension of ordinary type free λ-calculus, the extension being that a new class of terms called segments is incorporated in the system. Segments are only parts of terms in the traditional sense and act as a sort of generalized contexts. Segments were originally introduced by N. G. de Bruijn in order to provide for certain abbreviational facilities in the mathematical language AUTOMATH. In this paper we treat the $\lambda\sigma$-system as an interesting extension of the λ-calculus in its own right and do not discuss connections with AUTOMATH. We start from a simple system called λV and then gradually move on to the full $\lambda\sigma$-system." For the entire collection see MR 88i:03005.

Ohtsuka, H. (1989). A categorical model of λ-calculus based on de Bruijn's name-free expressions. *Bull. Inform. Cybernet.*, 23(3-4):129–140.
[**MR90m:03031**] de Bruijn's notation for terms in λ-calculus uses numbers instead of names for variables. For a bound variable, this number is a nonnegative integer measuring the "distance" of a certain occurrence to the binding λ in the term [N. G. de Bruijn, Nederl. Akad. Wetensch. Indag. Math. 40 (1978), no. 3, 348–356; MR 80b:03016]. Free variables can be treated analogously by adding a list of imaginary binders to the term. The name-free terms obtained can be considered as α-congruence classes of λ-terms.

The λ-calculus with β- and η-rules and surjective pairing is equivalent to a Cartesian closed category. This connection has been extensively investigated by P.-L. Curien [Inform. and Control 69 (1986), no. 1-3, 188–254; MR 88f:03012]. Curien used name-free terms. The present article is based on the C-monoid model for untyped de Bruijn terms. The author shows how the "reference transforming mappings", belonging to the explicit substitution procedure for de Bruijn's λ-terms, can be translated into operations of the categorical model. Reviewed by R. P. Nederpelt.

Abadi, M., Cardelli, L., Curien, P.-L., and Lévy, J.-J. (1991). Explicit substitutions. *J. Funct. Programming*, 1(4):375–416.

[MR92j:68068] This paper introduces three lambda-calculi with explicit substitution (an untyped, a simply typed and a second-order lambda-calculus), and establishes some basic results about them.

These calculi provide a rigorous framework to address implementation issues involving substitution (such as type-checking of languages with a complex type system, or reduction strategies for functional languages). Indeed, establishing the correctness of a type-checker for Quest was one of the original motivations for this work.

The bad news is that the list of typing rules for the second-order calculus is rather long. Reviewed by Eugenio Moggi.

Cartwright, R. (1991). Lambda: the ultimate combinator. In *Artificial intelligence and mathematical theory of computation*, pages 27–46. Academic Press, Boston, MA.

[MR92i:03012]

λ-calculus is a succinct notation for reasoning about functional programs. Usually, to incorporate λ-calculus in the term language of the predicate calculus is to interpret λ-expressions as terms constructed from the application function, the combinators K and S, and primitive functions on program data. However, this approach may lead to an unwieldy mechanism for reasoning about particular programs. In this paper, the author puts forward a new method of reconciling λ-notation with the combinatory framework imposed by the predicate calculus. The main idea is to interpret λ-expressions as combinatory terms without performing any translation, by taking λ as a binary combinator *lambda* and every variable v as a unary combinator v. By means of this approach, the author constructs models for the λ-calculus with dynamic binding and the $\lambda\sigma$-calculus with variables. Another motivation for interpreting λ as a combinator is that this permits denotational definitions of programming languages to be formulated as homomorphisms from syntactic algebras to semantic algebras. The system, in which λ is interpreted as a function, is easily understood and intuitively natural. For the entire collection see MR 92f:68002. Reviewed by Fangmin Song.

Curien, P.-L. and Ríos, A. (1991). Un résultat de complétude pour les substitutions explicites. *C. R. Acad. Sci. Paris Sér. I Math.*, 312(6):471–476.

[MR92e:03015] The $\lambda\sigma$-calculus, introduced by M. Abadi, L. Cardelli, Curien and J.-J. Levy [in Conference record of the Seventeenth Annual ACM Symposium on Principles of Programming Languages (San Francisco, CA, 1990), 16–30, ACM, New York, 1990; CCA 1991:2819], is a refinement of λ-calculus (in de Bruijn's notation), extended with an explicit operator applying substitutions to $\lambda\sigma$-terms. A set σ of reduction rules is given which allows us to manipulate substitutions explicitly and gives rise to a Noetherian notion of reduction, also called σ. We note here that σ-closed normal $\lambda\sigma$-terms are closed terms of the pure λ-calculus. A simply typed version of $\lambda\sigma$-calculus was also presented in the above-mentioned paper. Let $\sigma(a)$ denote the σ-normal form of a $\lambda\sigma$-term a. Then if a is typable in the $\lambda\sigma$-calculus, $\sigma(a)$ is typable in the simply typed λ-calculus; however, the converse is obviously false because of a loss of information on the substitutions. The authors propose here a better translation, which we denote by T, from $\lambda\sigma$-calculus to λ-calculus, based on rewriting rules with no loss of information. T is sound and complete in the sense that a is $\lambda\sigma$-simply typable if and only if $T(a)$ is λ-simply typable. Reviewed by Chantal Berline.

Ohtsuka, H. (1991). Comparison of two categorical models of typed λ-calculus. *Bull. Inform. Cybernet.*, 24(3-4):147–164.
[MR92g:03022] Summary: "This paper discusses a relation between two categorical models of typed λ-calculus which are both Cartesian closed categories. One of them has a concept of variables in it. The other does not have such a concept and is based on de Bruijn's name-free expression. We show that the second one is obtained by a certain construction over a Cartesian closed category and they are isomorphic from the categorical point of view."

Curien, P.-L., Hardin, T., and Ríos, A. (1992). Strong normalization of substitutions. In *Mathematical foundations of computer science 1992 (Prague, 1992)*, pages 209–217. Springer, Berlin.
[MR94j:03031] Summary: "$\lambda\sigma$-calculus is an extended λ-calculus where substitutions are handled explicitly. We prove the strong normalization of its subcalculus σ which computes substitutions." For the entire collection see MR 94f:68022.

Hardin, T. (1992). Eta-conversion for the languages of explicit substitutions. In *Algebraic and logic programming (Volterra, 1992)*, pages 306–321. Springer, Berlin.
[MR94m:68095] Summary: "Some new calculi referred to by the collective name of $\lambda\sigma$-calculus have been recently introduced to provide an explicit treatment of substitutions in the λ-calculus. They are term rewriting systems, with two sorts: substitution and term. The λ-terms are exactly the ground $\lambda\sigma$-terms of sort term containing no substitutions, and the β-reduction is decomposed in these calculi, into a starting reduction with a rule called (beta) followed by a derivation computing explicitly the substitution. These calculi differ by their treament of substitution. In this paper, we extend the $\lambda\sigma$-calculi with a conditional rewriting relation, called $c\eta$. This relation coincides, on λ-terms, with the classical η-reduction of λ-calculus. We prove that the confluent $\lambda\sigma$-calculus, augmented by $c\eta$, remains confluent and that the ground confluent version, extended by $c\eta$, is still ground confluent. The proof is done by the interpretation method introduced in another paper of ours [Theoret. Comput. Sci. 65 (1989), no. 3, 291–342; MR 91h:03021]." For the entire collection see MR 94f:68005.

Gray, J. W. (1993). Semantics of the typed λ-calculus with substitution in a Cartesian closed category. *J. Pure Appl. Algebra*, 89(1-2):107–126.
[MR95c:03034] In standard presentations of the λ-calculus, the notion of substituting a term for the free occurrences of a variable in another term is defined by recursion in the metatheory. By a λ-calculus with substitution, the author means one in which this notion is built into the object language, say with the notation $[a/x]b$, and the metatheoretic recursion defining it is replaced by reduction rules in

the object theory. (The author cites two other papers [T. Hardin and J.-J. Levy, "A confluent calculus of substitutions", Preprint, 1990; per bibl.; M. Abadi et al., J. Funct. Programming 1 (1991), no. 4, 375–416; MR 92j:68068] as sources for this sort of system.) In view of the crucial role played by substitution in the λ-calculus's central rule of β-reduction, making substitution an explicit part of the calculus allows a more accurate modeling of the computational steps involved in β-reduction. The author defines, for a typed λ-calculus with substitution, two sorts of semantics in Cartesian closed categories. Both semantics interpret types τ as objects $D(\tau)$ of the category and interpret terms of type τ as morphisms into $D(\tau)$ in a natural way. They differ in the domains of these morphisms. One, the global environment semantics, uses the product, indexed by all variables, of the corresponding $D(\tau)$; the other, the local environment semantics, uses, for the interpretation of a term a, the product of only those $D(\tau)$ corresponding to finitely many variables, including the free variables of a. The author proves that both interpretations are sound in the sense that reductions preserve the interpretations of terms. The reductions considered here are the usual α- and β-reductions of λ-calculus, now formulated using the explicit substitution operation, and the rules for reducing explicit substitutions. Reviewed by Andreas Blass.

Kamareddine, F. and Nederpelt, R. (1993). On stepwise explicit substitution. *Internat. J. Found. Comput. Sci.*, 4(3):197–240.

[MR94m:03029] In the usual presentation of λ-calculus the $\lambda\beta$-reduction is defined by $(\lambda x.A)B \to A[x := B]$, where the operation $[x := B]$ (substitute B for the free occurrences of x) is kept in the metatheory, that is, it is not a legitimate term of the object language. This can be a drawback if one is interested in describing how the details of substitution and renaming of bound variables are handled in a concrete implementation. A possible solution to this problem was proposed by M. Abadi et al. [J. Funct. Programming 1 (1991), no. 4, 375–416; MR 92j:68068] using "de Bruijn indexes" and an explicit formalism for substitutions. In this paper the authors propose a different formalism, still based on de Bruijn indexes, and show that the system of Abadi et al. can be interpreted in the new system. The approach is based on a new notation for (typed) lambda terms, where $\lambda x : \alpha.t$ is written as $(\alpha\lambda_x)t$ and tt' is written as $(t'\delta)t$ (where δ represents the application operation). Objects of the form $(\alpha\lambda_x)$ are called λ-items, and objects of the form $(t'\delta)$ are called substitution items. Thus each lambda-term is a sequence of "items" applied to a variable. The δ-item and λ-item involved in a β-reduction occur adjacently in the term without being separated by the body of the term. The system is then expanded by introducing appropriate "substitution items" and the relevant rewriting rules governing them. The system also takes into account local substitutions in which only some occurrences of a variable are substituted for. Reviewed by Alessandro Berarducci.

Balsters, H. (1994). *Lambda calculus extended with segments*: Chapter 1, Sections 1.1 and 1.2 (Introduction). In *Selected papers on Automath*, pages 339–367. North-Holland, Amsterdam.

[**MR1429412**] This item will not be reviewed individually. For details of the collection in which this item appears see 98g:03038.

Hardin, T. (1994). Eta-conversion for the languages of explicit substitutions. *Appl. Algebra Engrg. Comm. Comput.*, 5(6):317–341.

[**MR96d:68137a**] (Joint review with (Hardin, 1995).) Summary: "Some new calculi, referred to by the collective name of $\lambda\sigma$-calculus, have been recently introduced to provide an explicit treatment of substitutions in the λ-calculus. They are term rewriting systems, with two sorts: substitution and term. The λ-terms are exactly the ground $\lambda\sigma$-terms of sort term containing no substitutions, and the β-reduction is decomposed in these calculi into a starting reduction with a rule called (beta) followed by a derivation explicitly computing the substitution. These calculi differ by their treatment of substitution. In this paper, we extend the $\lambda\sigma$-calculi with a conditional rewriting relation, called $c\eta$. This relation coincides, on λ-terms, with the classical η-reduction of λ-calculus. We prove that the confluent $\lambda\sigma$-calculus, augmented by $c\eta$, remains confluent and that the ground confluent version [M. Abadi et al., in Conference Record of the Seventeenth Annual ACM Symposium on Principles of Programming Languages (San Francisco, CA, 1990), 31–46, ACM, New York, 1990; CCA 1991:2819], extended by $c\eta$, is still ground confluent. The result extends readily to categorical combinatory logic. The proof is done by the interpretation method introduced in [T. Hardin, Theoret. Comput. Sci. 65 (1989), no. 3, 291–342; MR 91h:03021]."

From the text of the erratum: "In the original paper, it is claimed that the system $\lambda\sigma$, augmented with the rewriting relation $c\eta$, is confluent. This result does not hold. $\lambda\sigma$ and $c\eta$ are confluent. But, these two rewriting relations do not commute: Proposition 5.7 is wrong. However, this proposition remains true on ground terms. So, the correct result is the following: The system $\lambda\sigma$, augmented with the rewriting relation $c\eta$, is ground confluent."

Nishizaki, S.-y. (1994). Simply typed lambda calculus with first-class environments. *Publ. Res. Inst. Math. Sci.*, 30(6):1055–1121.

[**MR96h:68120**] Summary: "We propose a lambda calculus $\lambda^{\rightarrow}_{\mathrm{env}}$ in which it is possible to handle first-class environments. This calculus is based on the idea of explicit substitution, that is, on $\lambda\sigma$-calculus. The syntax of $\lambda^{\rightarrow}_{\mathrm{env}}$ is obtained by merging the class of terms and that of substitutions. Reduction is made from the weak reduction of $\lambda\sigma$-calculus. Its type system also originates in the one of $\lambda\sigma$-calculus. Confluence of $\lambda^{\rightarrow}_{\mathrm{env}}$ is proved by Hardin's interpretation method which is originally used for proving confluence of $\lambda\sigma$-calculus. We prove strong normalizability of $\lambda^{\rightarrow}_{\mathrm{env}}$ by reducing it to strong normalizability of a simply typed record calculus. Finally, we propose a type inference algorithm which produces a principal typing for each typable term."

Ritter, E. (1994). Normalization for typed lambda calculi with explicit substitution. In *Computer science logic (Swansea, 1993)*, pages 295–304. Springer, Berlin.

[**MR96e:03016**] From the text: "This paper presents the best solution so far for an important open problem in the typed λ-calculus: Does the addition of explicit substitution preserve strong normalization? It shows that any notion of reduction that pushes substitution only under outermost λ-abstractions is strongly normalizing. The proof of this result uses a reducibility argument and not a syntax-specific combinatorial analysis. Hence this method works for more powerful λ-calculi such as the system F and the calculus of constructions. The result is also independent of the representation of the substitutions. It also holds for versions where de Bruijn indices are used only for bound variables and global variables are indexed separately. If the names of the variables are added to the substitution, i.e. a substitution now has the form $(x_n = t_n, x_1 = t_1)$, the result is still true." For the entire collection see MR 95j:68009.

Sato, M. and Kameyama, Y. (1994). Conservativeness of λ over $\lambda\sigma$-calculus. In *Logic, language and computation*, pages 73–94. Springer, Berlin.
[MR95f:68150] Λ is a SCHEME-like language previously defined by the first author of this paper. The $\lambda\sigma$-calculus, introduced by Abadi, Cardelli, Curien and Levy, is a refinement of the classical λ-calculus in which substitutions can be explicitly manipulated. In this paper a new version of Λ, with explicit substitutions, is defined, and a translation is given from the functional fragment of Λ to the $\lambda\sigma$-calculus (and vice versa). By using this translation it is proved that Λ is conservative over the $\lambda\sigma$-calcuus. This result is to be expected, since the functional part of Λ, as originally defined, is a sugared version of λ-calculus, and the introduction of the explicit substitutions in its syntax was done just by mimicking the $\lambda\sigma$-calculus syntax. For the entire collection see MR 95b:68015.

Hardin, T. (1995). Erratum: "Eta-conversion for the languages of explicit substitutions". *Appl. Algebra Engrg. Comm. Comput.*, 6(4-5):325.
[MR96d:68137b] See (Hardin, 1994).

Benaissa, Z.-E.-A., Briaud, D., Lescanne, P., and Rouyer-Degli, J. (1996). $\lambda\nu$, a calculus of explicit substitutions which preserves strong normalisation. *J. Funct. Programming*, 6(5):699–722.
[MR97m:68129] Summary: "Explicit substitutions were proposed by Abadi, Cardelli, Curien, Hardin and Levy to internalise substitutions into λ-calculus and to propose a mechanism for computing on substitutions. $\lambda\nu$ is another view of the same concept which aims to explain the process of substitution and to decompose it into small steps. It favours simplicity and preservation of strong normalisation. This way, another important property is missed, namely confluence on open terms. In spirit, $\lambda\nu$ is closely related to another calculus of explicit substitutions proposed by de Bruijn and called $C\lambda\xi\phi$. In this paper, we introduce $\lambda\nu$, we present $C\lambda\xi\phi$ in the same framework as $\lambda\nu$ and we compare both calculi. Moreover, we prove properties of $\lambda\nu$; namely, $\lambda\nu$ correctly implements β reduction; $\lambda\nu$ is confluent on closed terms, i.e. on terms of classical λ-calculus and on all terms that are derived from those terms; and, finally, $\lambda\nu$ preserves strong normalisation in the sense that strongly β normalising terms are strongly $\lambda\nu$ normalising."

Bloo, R. and Rose, K. H. (1996). Combinatory reduction systems with explicit substitution that preserve strong normalisation. In *Rewriting techniques and applications (New Brunswick, NJ, 1996)*, pages 169–183.

Springer, Berlin.

[**MR98h:68128**] Summary: "We generalise the notion of explicit substitution from the λ-calculus to higher-order rewriting, realised by combinatory reduction systems (CRSs). For every confluent CRS, R, we construct an explicit substitution variant, Rx, which we prove confluent.

"We identify a large subset of the CRSs, the structure-preserving CRSs, and show for any structure-preserving CRS R that Rx preserves strong normalisation of R.

"We believe that this is a significant first step towards providing a methodology for reasoning about the operational properties of higher-order rewriting in general, and higher-order program transformations in particular, since confluence ensures correctness of such transformations and preservation of strong normalisation ensures that the transformations are always safe, in both cases independently of the reduction strategy used." For the entire collection see MR 98h:68011.

Curien, P.-L., Hardin, T., and Lévy, J.-J. (1996a). Confluence properties of weak and strong calculi of explicit substitutions. *J. ACM*, 43(2):362–397.

[**MR98b:68094**] The $\lambda\sigma$-calculus has been introduced for an explicit treatment of substitutions in the λ-calculus [M. Abadi et al., J. Funct. Programming 1 (1991), no. 4, 375–416; MR 92j:68068]. It is also another formulation of calculi for categorical combinators which have been induced from the study of categorical models of the λ-calculus. The present paper reintroduces the $\lambda\sigma$-calculus and its variants in a systematic manner, with special attention to confluence properties. The authors first introduce two calculi for explicit substitution, corresponding to the λ-calculus, with the restriction that a term in the scope of a λ cannot be rewritten. They show that the calculi are both confluent. Next they deal with the full system of $\lambda\sigma$-calculus. They show that the $\lambda\sigma$-calculus is not confluent, although it is confluent on ground terms only. To overcome the difficulty of this nonconfluence result, they propose an extension of $\lambda\sigma$-calculus, called $\lambda\sigma_{\Uparrow}$-calculus, and they prove that $\lambda\sigma_{\Uparrow}$-calculus is fully confluent. Reviewed by Hirofumi Yokouchi.

Curien, P.-L., Hardin, T., and Ríos, A. (1996b). Strong normalization of substitutions. *J. Logic Comput.*, 6(6):799–817.

[**MR97g:03016**] Summary: "$\lambda\sigma$-calculus is an extended λ-calculus in which substitutions are handled explicitly. It is similar to, and inspired by, categorical combinatory logic (CCL). The strong normalization of σ, the subcalculus which computes substitutions, may be inferred from the strong normalization of the similar subsystem SUBST of CCL. We present here an independent proof of the termination of several subsitution calculi, including σ and SUBST."

Muñoz, C. (1996). Confluence and preservation of strong normalisation in an explicit substitutions calculus. In *11th Annual IEEE Symposium on Logic in Computer Science (New Brunswick, NJ, 1996)*, pages 440–447. IEEE Comput. Soc. Press, Los Alamitos, CA.

[**MR1461855**] This item will not be reviewed individually. For details of the collection in which this item appears see 98a:68006 .

Strahm, T. (1996). Partial applicative theories and explicit substitutions. *J. Logic Comput.*, 6(1):55–77.

[**MR97m:03030**] In the literature, partial combinatory logic and partial lambda calculus are used as a basis for first-order theories with partial self-application. These theories are used in the formalization of constructive mathematics and as logical foundations for functional programming languages.

The author points out that partial combinatory logic and partial lambda calculus are not equivalent, as is sometimes thought. In fact, the partial lambda calculus is not embeddable into partial combinatory logic and does not have the standard recursion-theoretic model.

This problem is resolved in this paper by extending the partial lambda calculus to include terms formed by finite "explicit substitutions". The new system is embeddable into partial combinatory logic and inherits all its models. A reduction relation over the new partial lambda calculus is shown to be Church-Rosser. Reviewed by Martin W. Bunder.

Bloo, C. J. (1997). *Preservation of termination for explicit substitution.* Eindhoven University of Technology, Eindhoven. Dissertation, Technische Universiteit Eindhoven, Eindhoven, 1997.

[**MR98g:68015**] Summary: "The λ-calculus is a language for the formal description of mathematical functions and application of functions to arguments. It lies at the base of functional programming languages. The reduction relation β-reduction allows for computation of the result of application of a function to an argument.

"For the study of the operational behaviour of β-reduction it is desirable to refine the reduction relation into several smaller steps each of which represent a fixed amount of work. This refined notion of reduction is called 'explicit substitution' and is the main subject of this thesis.

"There are several different ways of obtaining a refined reduction, each leading to a different explicit substitution calculus. In most explicit substitution calculi the reduction relation can be divided into two disjoint parts, one for the introduction of explicit substitutions, and one for the distribution and elimination of them. The latter is called the 'substitution calculus'.

"Important properties for explicit substitution calculi are confluence and termination of the substitution calculus, confluence of explicit substitution reduction, the ability of explicit substitution reduction to simulate β-reduction, and soundness of explicit substitution reduction with respect to β-reduction.

"Furthermore it is desirable for explicit substitution calculi to satisfy the preservation of strong normalization property (abbreviated to PSN). This property states that terms which are strongly normalizing with respect to β-reduction are strongly normalizing with respect to explicit substitution reduction as well. We show for a handful of explicit substitution calculi that they possess all of these properties.

"The PSN property receives special attention. We first present a direct, constructive proof. This has the disadvantage that for any new calculus with explicit substitution, the proof has to be redone, which is nontrivial to do. Therefore, we present a different method which uses a recursive path order to show PSN. This method is very general and can be applied in a rather straightforward way to new explicit substitution calculi. "Until recently, the study of explicit substitution has concentrated on λ-calculus. We present a generalization of the concept to higher-order rewriting. We use the concept of combinatory reduction systems (CRSs) and define when a CRS can be called explicit. Then we present a method that gives for each CRS a corresponding explicit CRS. We show that the explicit CRS satisfies all the properties mentioned above except for PSN. We identify a large subclass of CRSs that do satisfy PSN. This is proven by a method using the recursive path order.

"In the final chapter we briefly study type systems with explicit substitution. We extend the notion of pure type system to λ-calculus with explicit substitution and show that the resulting explicit pure type systems behave as well as ordinary pure type systems, except for the subject reduction property. We discuss two very different approaches to overcome this exception and conclude with a short discussion on directions for future research."

Kamareddine, F. and Ríos, A. (1997). Extending a λ-calculus with explicit substitution which preserves strong normalisation into a confluent calculus on open terms. *J. Funct. Programming*, 7(4):395–420.

[MR99g:03013] Lambda calculi with explicit substitution are interesting both in theory and in practice, as they make the mechanisms of substitution part of the calculus. Classical lambda terms are encoded in such calculi as closed terms, using de Bruijn's indices. Variables may also be considered, so that open terms represent calculi of classical contexts. Two relevant questions are whether the encoding preserves strong normalisation of lambda terms, and whether the calculus extended with variables is confluent.

This paper proves confluence for λs_e, an extension to open terms of a calculus with explicit substitutions known as λs, and—building on previous results of the same authors—argues that λs is the only calculus enjoying the properties above that, in addition, simulates classical single-step β-reduction.

To establish confluence, the authors need to extend Hardin's interpretation method. This yields a technique that works under a milder hypothesis and that—not being specific to calculi of explicit substitutions—may in principle find applications to other rewrite theories. The paper reads quite well, though the exposition could be improved in places. Reviewed by Vladimiro Sassone.

Kirchner, C. and Ringeissen, C. (1997). Higher-order equational unification via explicit substitutions. In *Algebraic and logic programming (Southampton, 1997)*, pages 61–75. Springer, Berlin.

[MR99d:68218] Summary: "We show how to reduce the unification problem modulo $\beta\eta$-conversion and a first-order equational theory E, into a first-order unification problem in a union of two nondisjoint equational theories including E and a calculus of explicit substitutions. A rule-based unification procedure in this combined theory is described and may be viewed as an extension of the one initially designed by G. Dowek, T. Hardin and C. Kirchner for performing unification of simply typed λ-terms in a first-order setting via the $\lambda\sigma$-calculus of explicit substitutions. Additional rules are used to deal with the interaction between E and $\lambda\sigma$." For the entire collection see MR 99b:68002.

Hardin, T., Maranget, L., and Pagano, B. (1998). Functional runtime systems within the lambda-sigma calculus. *J. Funct. Programming*, 8(2):131–176.

[MR99c:68034] Summary: "We define a weak λ-calculus, $\lambda\sigma_w$, as a subsystem of the full λ-calculus with explicit substitutions $\lambda\sigma_{\Uparrow}$. We claim that $\lambda\sigma_w$ could be the archetypal output language of functional compilers, just as the λ-calculus is their universal input language. Furthermore, $\lambda\sigma_{\Uparrow}$ could be the adequate theory to establish the correctness of functional compilers. Here we illustrate these claims by proving the correctness of four simplified compilers and runtime systems modelled as abstract machines. The four machines we study are the Krivine machine, the SECD, the FAM and the CAM. Thus, we give the first formal proofs for Cardelli's FAM and for its compiler."

Kamareddine, F. and Ríos, A. (1998). Bridging de Bruijn indices and variable names in explicit substitutions calculi. *Log. J. IGPL*, 6(6):843–874.

[**MR1659850**] A review for this item is in process.

Bloo, R. and Geuvers, H. (1999). Explicit substitution. On the edge of strong normalization. *Theoret. Comput. Sci.*, 211(1-2):375–395.

[**MR99k:68025**]

From the summary: "We use the recursive path ordering (RPO) technique of semantic labelling to show the preservation of the strong normalization property (PSN) for several calculi of explicit substitution. PSN states that if a term M is strongly normalizing under ordinary β-reduction (using 'global' substitutions), then it is strongly normalizing if the substitution is made explicit ('local'). There are various ways of making global substitution explicit and PSN is a quite natural and desirable property for the explicit substitution calculus. Our method for proving PSN is very general and applies to several known systems of explicit substitutions, both with named variables and with De Bruijn indices. We also look at two small extensions of the explicit substitution calculus that allow the permutation of substitutions. For one of these extensions, PSN fails (by the counterexample in Mellies 1995). For the other we can prove PSN using our method, thus showing the subtlety of the subject and the generality of our method.

"One of the key ideas beind our proof is that, for λx the set of terms of the explicit substitution calculus, we look at the set $\lambda x^{<\infty}$, consisting of the terms A such that the substitution-normal-form of each subterm of A is β-SN. This is a kind of 'induction loading': if we prove that λx-reduction is SN on the set $\lambda x^{<\infty}$, then we have proved PSN for λx. To prove λx-SN on the set $\lambda x^{<\infty}$, we define the β-size of a term $A \in \lambda x^{<\infty}$ as the maximum length of a β-reduction path from the substitution-normal-form of A. Using this β-size, we define a translation from $\lambda x^{<\infty}$ to some well-founded order $>_{\mathrm{rpo}}$ on labelled terms, such that any infinite λx-reduction path starting from an $A \in \lambda x^{<\infty}$ translates to an infinite $>_{\mathrm{rpo}}$-descending sequence. The well-founded order $>_{\mathrm{rpo}}$ is defined by using a technique similar to semantic labelling."

1.10 Higher-order logic and type theory

Salle, P. (1978). Une extension de la théorie des types en λ-calcul. In *Automata, languages and programming (Fifth Internat. Colloq., Udine, 1978)*, pages 398–410. Springer, Berlin.

[**MR80k:03019**] Type assignment may be a way of statically testing properties of λ-terms. For example it is well known that any term which has a type in Curry's functional typing system has a normal form. However, the converse is false, and indeed very few terms have a type in this system. The author presents a more comprehensive type theory where every term has a nonempty set of types and in which the properties of having a head normal form and having a normal form are completely characterised, extending previous constructions and results of M. Coppo and M. Dezani-Ciancaglini (Arch. Math. Logik Grundlag. 19 (1978/79), no. 3-4, 139 - 156). The types are built from three basic types $\omega, 0, 1$. The type 0 is intuitively the type of terms having a normal form, while the type 1 is the type of normal forms M such that $MN_1N_2 \cdots N_n$ has a normal form for all normal forms $N_1, N_2, \ldots, N_n$. The set of types is quotiented by some equivalences which are intuitively clear. It is then turned into a lattice by setting $\omega < 0 < 1$. In a first approach, at most one type is assigned to each term, and partial results are obtained. However, some terms still have no type. Then every term receives a set of types, still following an idea of Coppo and Dezani-Ciancaglini, and the final results are obtained: Every term has at least the type ω. Two $\beta\eta$-interconvertible terms have the same set of types. A term is unsolvable if and only if it has only type ω. A term has a normal form if and only if it has a type greater than 0. Another general typing system has been developed by Coppo and Dezani-Ciancaglini (op. cit.). Some relations between the two systems have been studied by Coppo, Dezani-Ciancaglini and the author (Automata, languages and programming (Proc. Sixth Colloq., Graz, 1979), pp. 133 - 146, Springer, Berlin, 1979). In particular it was shown that two terms have the same set of types in the author's system if and only if they are equivalent in the sense of Morris and that they have the same set of types in the system of Coppo and Dezani-Ciancaglini if and only if they have the same Böhm tree. (For the entire collection see MR 80c:68003.) Reviewed by G. Berry.

Seldin, J. P. (1978). A sequent calculus formulation of type assignment with equality rules for the $\lambda\beta$-calculus. *J. Symbolic Logic*, 43(4):643–649.

[MR80e:03012] The rules for assigning types or type-schemes to λ-terms are usually presented as a natural-deduction system, including an equality rule: "If type τ is assigned to X and $X = Y$, then assign τ to Y." For the case where equality is $\lambda\beta\eta$, or combinatory equality, an equivalent Gentzen-sequence-style formulation was given in Section 14E of the book by H.B. Curry, the reviewer and the author (Combinatory logic, Vol. II, North-Holland, Amsterdam, 1972; Zbl 242 #2029), and a cut-elimination theorem was proved. The author gives a sequence formulation for the case of $\lambda\beta$-equality and proves cut elimination. It is interesting that the cut-elimination proof for these rules would fail (as the author points out) if equality were $\lambda\beta\eta$, not $\lambda\beta$. Reviewed by R. Hindley.

Coppo, M. and Dezani-Ciancaglini, M. (1978/79). A new type assignment for λ-terms. *Arch. Math. Logik Grundlag.*, 19(3-4):139–156.

[MR81d:03014] A nonstandard assignment of types to terms of the (type-free) λ-calculus is introduced. The set of types T is defined as follows: $0, 1 \in T$; $\sigma, \tau \in T \Rightarrow (\sigma)\tau \in T$. On T one introduces a pre-ordering: $0 \sqsubseteq \sigma \sqsubseteq 1$; $1 \sqsubseteq (0)1$; $(1)0 \sqsubseteq 0$;

$\sigma \sqsubseteq \sigma', \tau \sqsubseteq \tau' \Rightarrow (\sigma)\tau' \sqsubseteq (\sigma')\tau$. Besides the rules of basic functionality (H. B. Curry and R. Feys, Combinatory logic, Vol. I, Chapter 9, North-Holland, Amsterdam, 1958; MR 20 #817), one requires that if a term M has type σ and $\tau \sqsubseteq \sigma$, then M has type τ . The following theorems are proved. (1) If M is a normal form (nf), then M has a type. (2) If M has a type, then M has an nf. (3) For M in nf, it is decidable whether M has type τ. Reviewed by Henk Barendregt.

Seldin, J. P. (1979). Progress report on generalized functionality. *Ann. Math. Logic*, 17(1-2):29–59.

[MR82a:03017] This paper is a summary of the author's work in progress on systems of combinatory logic or λ-calculus that include the generalized functionality operator G, which provides interpretations of various type theories. The characteristic rule for G is that $G\alpha\beta X$ and αY imply $\beta Y(XY)$, where α is a type, β is a type function, so that βY is a type, and $G\alpha\beta$ is a type. Thus the type of XY depends on Y itself as well as on the types of X and Y. The ordinary functionality operator F can be defined in terms of G as follows: $F \equiv \lambda xy.Gx(Ky)$. The author describes several systems, Hilbert-style and Gentzen-style, extensional and nonextensional, mostly stating results without proof. Many results are analogous to results of the theory of ordinary functionality. Reviewed by B. Lercher.

Coppo, M. and Dezani-Ciancaglini, M. (1980). An extension of the basic functionality theory for the λ-calculus. *Notre Dame J. Formal Logic*, 21(4):685–693.

[MR82d:03023] From the introduction: "This paper is concerned with the assignment of types to terms of the λ-calculus. It extends Curry's basic functionality theory of typing in a simple way by defining a more powerful rule for type formation and modifying the deduction rules accordingly. The main idea is to define from any arbitrary finite set of types $(\tau_1, \dots, \tau_n)$ a 'sequence' $(\tau_1, \dots, \tau_n)$ whose underlying set of terms can be interpreted as the intersection of those of $\tau_1, \dots, \tau_n$. This leads to a system in which the assignment rules are left substantially unchanged, but the set of typed terms is much larger than in the basic functionality theory (in particular, all strongly normalisable terms possess at least one type)." Indeed, the typed terms are exactly the strongly normalisable ones. The typing is not preserved under η-convertibility. Reviewed by John Staples.

Coppo, M., Dezani-Ciancaglini, M., and Venneri, B. (1980). Principal type schemes and λ-calculus semantics. In *To H. B. Curry: essays on combinatory logic, lambda calculus and formalism*, pages 535–560. Academic Press, London.

[MR82d:03022] The authors have previously proposed what is essentially an extension of Curry's basic functionality theory (e.g., 82d:03023 below). The extension adds a "universal" type ω, and types of the form $(\sigma_1, \dots, \sigma_n)\tau$, $n \geq 1$, where $\sigma_1, \dots, \sigma_n, \tau$ are types. A type $(\sigma_1, \dots, \sigma_n)\tau$ is intended to type a function whose (single) argument has all of the types $\sigma_1, \dots, \sigma_n$ and whose value has type τ. The extended typing is preserved under β-conversion, and all solvable terms have non-ω types. In this paper it is shown that Curry's concept of principal type scheme does not extend to this more general setting, and a generalised notion of principal type scheme is developed. Finally a λ-calculus model is defined whose domain is the power set of the set of type schemes; the value of a term in the model is just its set of type schemes. (For the entire collection see MR 81j:03005.) Reviewed by John Staples.

Pottinger, G. (1980). A type assignment for the strongly normalizable λ-terms. In *To H. B. Curry: essays on combinatory logic, lambda calculus and formalism*, pages 561–577. Academic Press, London.

[MR82j:03014] The author considers a calculus S_1 for statements of the form $t \models A$, where t is a λK-term and A is a formula built up from propositional variables, $\supset$ and a new connective $\hat{\&}$ with the following intended meaning: to assert $A \hat{\&} B$ is to assert that one has a reason for asserting A which is also a reason for asserting B. This meaning is different from the intuitionistic meaning of &: to assert $A\&B$ is to assert that one has a pair of reasons, the first of which is a reason for asserting A and the second of which is a reason for asserting B. The rules of S_1 include $(\hat{\&}E)$ $\frac{P \vdash t \models A \hat{\&} B}{P \vdash t \models A}$, $\frac{P \vdash t \models A \hat{\&} B}{P \vdash t \models B}$, $(\hat{\&}I)$ $\frac{P \vdash t \models A \quad P \vdash t \models B}{P \vdash t \models A \hat{\&} B}$, (η) $\frac{P \vdash \lambda x.tx \models A}{P \vdash t \models A}$; these make it hard to prove results about S_1. In order to overcome this difficulty the author introduces an (extensionally) equivalent calculus S_2 which is obtained by enlarging the stock of axioms and altering the form of $(\supset I)$. Let $X_(t,P) = \{A \mid P \vdash t \models A$ is derivable in S_2 (or equivalently in S_1)$\}$, $X_t = \cup_P X_(t,P)$ and $TERM = \{t \mid X_t \neq \emptyset\}$. The main result is that $TERM = \{t \mid t$ is $\beta\eta$-strongly normalizable$\}$. In the final section the author provides evidence to the effect that $\hat{\&}$ behaves quite differently from &. If A is a theorem means that, for some t, $\vdash t \models A$ is derivable in S_2 (or equivalently in S_1), then formulas like $p \supset (q \supset p \hat{\&} q)$, $(p \supset q) \supset ((p \supset r) \supset (p \supset q \hat{\&} r))$, $(p \hat{\&} q \supset r) \supset (p \supset (q \supset r))$ are not theorems. (For the entire collection see MR 81j:03005.) Reviewed by Carlo Cellucci.

Ben-Yelles, C. B. (1981). G-stratification is equivalent to F-stratification. *Z. Math. Logik Grundlag. Math.*, 27(2):141–150.

[MR82i:03022] The theory of basic functionality (see the book by H. B. Curry and R. Feys [Combinatory logic, Vol. I, North-Holland, Amsterdam, 1958; MR 20 #817] for a complete description) provides rules for assigning type symbols to combinatory terms and λ-terms, relative to assignments of type symbols to variables. Given such an assignment, a term is said to be stratified relative to it if the rules permit at least one type symbol to be assigned to that term. The complex type symbols employed in the theory of basic functionality have the form $F\alpha\beta$, where α and β are atomic type symbols or are built up from such symbols by means of F. Intuitively, $F\alpha\beta$ is thought of as standing for a class of functions the common domain of which is denoted by α and the ranges of which are included in what β denotes. The intuitive import of a statement $(F\alpha\beta)U$ is that the term U stands for such a function. These intuitive considerations are formally expressed by means of a rule which allows one to infer the statement $\beta(UV)$ from the statements $(F\alpha\beta)U$ and αV. J. P. Seldin ["The theory of generalized functionality", to appear] has generalized these ideas by introducing type symbols of the form $G\alpha\beta$, where α is a type symbol but, in contrast with what was said above, β is thought of as standing for a function having types as values, rather than as standing for a type. This way of looking at things led him to introduce a rule which allows $(\beta V)(UV)$ to be inferred from $(G\alpha\beta)U$ and αV. Note that, contrary to the procedure followed in the theory of basic functionality, in Seldin's theory the type symbol βV assigned to UV depends on V. For both combinatory terms and λ-terms the author shows that there is an assignment of type symbols to variables in the theory of basic functionality relative to which a term is stratified if and only if there is an assignment of type symbols to variables in Seldin's theory relative to which that term is stratified. The exposition is quite clear and informative. Reviewed by Garrel Pottinger.

Coppo, M., Dezani-Ciancaglini, M., and Venneri, B. (1981). Functional characters of solvable terms. *Z. Math. Logik Grundlag. Math.*, 27(1):45–58.

[MR82j:03011] One of the reasons that typed λ-calculus is interesting is that the types relate to the intended meanings of the terms when they are interpreted. But the ordinary typed λ-calculus is too limited in assigning these "meanings", in that the only terms which are assigned types form a (proper) subset of the set of terms with normal forms, while it is clear from other contexts that all terms that have normal forms, and some that do not, can have meanings. For example, for many purposes all terms with head normal forms have meaning (see Corollary 5.3 of C. P. Wadsworth's article [SIAM J. Comput. 5 (1976), no. 3, 488 - 521; MR 58 #21493]). It is thus desirable to extend ordinary typed λ-calculus so that at least all terms with head normal forms have types. The authors do better than this. They have found a version of the typed λ-calculus in which every term has a type, but in which only terms with certain kinds of types have normal forms or head normal forms. Furthermore, two terms which are β-convertible have exactly the same types. This theory is obtained from the theory of functionality of H.B. Curry [Curry and R. Feys, Combinatory logic, Vol. I, Chapter 9, North-Holland, Amsterdam, 1958; MR 20 #817; Curry, J. R. Hindley and the reviewer, Vol. II, Chapter 14, 1972; Zbl 242:02029] by making two modifications: (1) a type ω is introduced to be the type of all terms (it is Curry's E), and (2) sequences of types of the form $[\sigma_1, \dots, \sigma_n]$ are introduced to represent, in effect, the "intersection" of types $\sigma_1, \dots, \sigma_n$. The compound type $F\sigma\tau$ is replaced by the type $F'[\sigma_1, \dots, \sigma_n]\tau$ or $[\sigma_1, \dots, \sigma_n]\tau$. The rules of the system are the obvious introduction and elimination rules for F' (which are obtained from the corresponding rules for F), a rule which assigns ω to every term, and the rule $\sigma_1 X, \dots, \sigma_n X \vdash [\sigma_1, \dots, \sigma_n]X$. (Here ' $\sigma_i X$' is the statement that X has type σ_i. The converse of this last rule can be proved as a derived rule.) To state the main results the following definitions are needed: first, note that each type can be written uniquely in the form $[\tau_1] \cdots [\tau_n]\phi$, where $[\tau_i]$ is a sequence of some finite length, $n \geq 0$, and ϕ is an atomic type; this ϕ is called the tail of the type. A type is tailproper if its tail is not ω. Next, the level of an occurrence of a type τ in a type σ is defined by induction on the construction of σ from τ as follows:

if τ is all of σ , this level is 0; if σ is $[\nu_1, \ldots, \nu_n]\mu$, the level of τ in σ is either one more than the level of τ in ν_i or exactly the level of τ in μ, depending on where τ occurs in σ. Moreover, a type is proper if the only occurrences of ω in it are at odd levels. Clearly a proper type is tailproper. Finally, a basis $\mathfrak{B}$ is a set of statements of the form σx, where x is a variable and σ is a type; and a basis is said to be proper if every occurrence of ω in a statement σx of the basis is at an even level of σ. The main results can now be stated as follows: (1) A term X possesses a head normal form if and only if there are a basis $\mathfrak{B}$ and a tailproper type τ such that $\mathfrak{B} \vdash \tau X$; and (2) a term X possesses a normal form if and only if there are a proper basis $\mathfrak{B}$ and a proper type τ such that $\mathfrak{B} \vdash \tau X$. Both of these results are proved by showing that deductions can be normalized (in a sense equivalent to that of D. Prawitz [Natural deduction, Almqvist & Wiksell, Stockholm, 1965; MR 33 #1227] for first- and second-order logic; here each reduction step causes a contraction in the term of the conclusion) and analyzing the ways in which redexes can occur in the conclusions of such normalized deductions. The authors conclude their paper by giving a restricted version of their system in which the syntax of types and the assignment rules are limited so as to rule out all types which are meaningless when types are thought of as properties of terms (the terms themselves being regarded as functions from terms to terms). The restrictions amount essentially to identifying with ω all nontailproper types, identifying with each other all types which differ only in the order of their sequences or which differ by insertions or deletions of nontailproper types in their sequences, and by restricting vacuous abstraction by rule $F'i$ to the case in which the conclusion has the form $[\omega]\tau(\lambda x.X)$. This restricted system is not equivalent to the original system and is not, in fact, a generalization of Curry's theory of functionality, since in the restricted system $[\tau][\sigma]\tau(\lambda xy.x)$ can be deduced only if σ is ω, whereas in ordinary functionality and in the original system of this paper this can be deduced for any type σ. However, the main results of the paper hold for the restricted system. In addition, in the restricted system types are invariant under η (they are not in the original system). Reviewed by J. P. Seldin.

Hindley, J. R. (1982). The simple semantics for Coppo-Dezani-Sallé types. In *International symposium on programming (Turin, 1982)*, pages 212–226. Springer, Berlin.

[MR86k:68066] Author summary: "The Coppo-Dezani-Salle type-language has '$\cap$' (intersection) and 'ω' (universal type), in addition to the usual '$\rightarrow$' (exponentiation). Coppo, Dezani and Salle have presented formal rules for assigning types to type-free λ-terms, and have proved metatheorems which give their system significance and interest [P. Salle , Automata, languages and programming (Udine, 1978), 398–410, Springer, Berlin, 1978; MR 80k:03019; M. Coppo , M. Dezani-Ciancaglini and B. Venneri , Z. Math. Logik Grundlag. Math. 27 (1981), no. 1, 45–58; MR 82j:03011]. However, no precise semantics has been given for their system yet, though the authors have made it fairly obvious informally what semantics they intended. We define a precise semantics in an arbitrary model of type-free λ-calculus. The rules of Coppo, Dezani and Salle turn out to be incomplete with respect to this semantics, but

they become complete when three extra rules (two trivial, one not) are added. The completeness proof uses the term-model only." For the entire collection see MR 86f:68010.

Statman, R. (1982). Completeness, invariance and λ-definability. *J. Symbolic Logic*, 47(1):17–26.

[MR84e:03020] A functional is lambda-definable in a given type structure [H. Friedman, Logic Colloquium (Boston, Mass., 1972 - 73), 22 - 37, Lecture Notes in Math., 453, Springer, Berlin, 1975; MR 52 #2833] if it is the interpretation of a closed typed lambda-term (of the appropriate type). In the first section of this paper a class of structures (obtained essentially as term models) is constructed in which the definable elements are exactly those invariant under permutations (this is known to be not true in full type structures; see a paper by G. Plotkin [To H. B. Curry: essays on combinatory logic, lambda calculus and formalism, 363 - 373, Academic Press, London, 1980; MR 82a:03016] for more discussion). From considerations on these models a number of interesting corollaries are obtained, such as an "omega-rule" for beta-eta-conversion, a maximal consistent extension of beta-eta-conversion, definability and undefinability results for typed versions of Church's delta and a completeness theorem for hereditarily finite full type structures (which generalizes one of Plotkin's results [op. cit.]). In the second section the author considers some consequences of a conjecture [Plotkin, op. cit.] that lambda-definability of a functional in a full type structure on a finite ground domain is decidable (it is not known whether Plotkin's conjecture is true). For example, assuming Plotkin's conjecture, the problem of deciding whether a given statement of type theory (in the language defined by Friedman [op. cit.]) has a hereditarily finite model is r.e. In the third section one of the models constructed in Section 1 is used to obtain some complexity results. For example, no nontrivial set of terms (of the same type) closed under beta-equality can be decided in elementary time. Reviewed by M. Coppo.

Barendregt, H., Coppo, M., and Dezani-Ciancaglini, M. (1983). A filter lambda model and the completeness of type assignment. *J. Symbolic Logic*, 48(4):931–940 (1984).

[MR85j:03014] Over the last several years there has been growing interest in types in programming languages. One of the many areas of research growing from that interest has been the type assignment problem: Given a term in a programming language and an assignment of types to identifiers, is it possible to assign that term a type? Given such a system one need not be as careful in declaring the types of all variables as is currently required by languages such as Pascal. One could declare the types of some of the variables and let the computer deduce the types of the others from the ways they occur in the program. In 1958 H. B. Curry (see paper for all references) described a formal system for assigning types to terms of the type-free λ-calculus. In 1975 D. S. Scott gave a natural semantics for type assignment and asked whether a completeness result could be obtained. This paper shows that the answer is yes. The same result was obtained independently by R. Hindley .

The Curry type schemes are the smallest set containing all type variables and closed under $\to$. A type assignment is a statement of the form $B \vdash \sigma M$ where σ is a type, M is a term of the type-free lambda calculus and B is a set of statements assigning types to variables. For example, $\{(\tau \to \sigma)x, \tau y\} \vdash \sigma(xy)$. The Curry type assignment

rules given in the paper can be used to assign types to statements as in the example above. Let M be a model of the type-free lambda calculus. Thus one can interpret λ-terms in M. One can also interpret types in M by assigning type variables to subsets of M and then extending this by letting the meaning of $\sigma \to \tau$ be the set of all d in M such that for all e in the meaning of σ, $d \cdot e$ is in the meaning of τ. One can then write $B \models \sigma M$ if and only if for all type-free models M such that the assignment B is valid, M has type σ. The completeness theorem says that $B \vdash \sigma M$ if and only if $B \models \sigma M$.

The authors accomplish their proof by extending the types to include a new operation $\sigma \cap \tau$ on types and the new type ω. They also define a new preorder $\leq$ on types such that the intended meaning of $\sigma \leq \tau$ is that the meaning of σ is a subset of the meaning of τ. The type assignment rules are extended to include ω and $\cap$. By a very clever construction, the authors form a model of the type-free λ-calculus from ideals of types. This model is constructed so that the meaning of a λ-term M corresponds to the set of all types that the type assignment rules assign to it. This model is then used to prove the completeness theorem for the extended type assignment system. A Prawitz normalization argument is then used to show that the extended type assignment system is conservative over Curry's. The completeness of Curry's system follows.

The paper is interesting not only for the completeness proof but also for the new ideal models of the type-free λ-calculus constructed therein. Reviewed by Kim Bruce.

Fortune, S., Leivant, D., and O'Donnell, M. (1983). The expressiveness of simple and second-order type structures. *J. Assoc. Comput. Mach.*, 30(1):151–185.

[MR84j:03029] This is the first in a series of four papers on the polymorphic typed λ-calculus (P.T.L.). The sequels are by Leivant [Proceedings of the 13th annual ACM symposium on theory of computing (Milwaukee, Wis., 1981), 38 - 45, Assoc. Comput. Mach., New York, to appear], the reviewer [22nd annual symposium on foundations of computer science (Nashville, Tenn., 1981), 279 - 282, IEEE, New York, 1981], and Leivant ["Reasoning about functional programs", Foundations of computer science (Tucson, Ariz., 1983), to appear; per revr.]. This paper provides the background for the principal results of the third and fourth papers [op. cit.]. The most interesting of these results [Leivant, 1983] goes as follows. Let A and B be r.e. data types and let $f : A \to B$ be recursive. Then f is computable by a P.T.L. term if and only if f is provably total in classical analysis. Reviewed by Richard Statman.

Hindley, R. (1983a). The completeness theorem for typing λ-terms. *Theoret. Comput. Sci.*, 22(1-2):1–17.

[MR85e:03030a] The author gives a clear and concise summary of the present state of the art of the theory of types in λ-calculus. His precise bibliography allows the reader to refer to the literature necessary for a general understanding of the topic. He also provides an original proof of the completeness theorem for λ-calculus and combinatory logic with respect to three different semantics of the untyped λ-calculus (the conjecture with respect to the third semantics is proved in detail in his article "Curry's type-rules are complete with respect to the F-semantics too", Theoret.Comput.Sci 22 (1983), no. 1-2, 127–133).

The reviewer's only criticism concerns the author's statement that the usual language of types is not strong enough to express the difference between the models of λ-

calculus, and that the problem remains open. In fact, the reviewer has shown ["A generalization of the theory of functionality in λ-calculus" (French), These d'Etat, Univ. Paul Sabatier, Toulouse, 1980; BullSig(110) 1981:13358] that the inclusion and equivalence relations in G. D. Plotkin's models T^{ω}, P_{ω} and D. S. Scott's model D_{∞}, as well as the inclusion and operational equivalence relations of J. H. Morris , correspond exactly to relations defined on the sets of types and the sets of schemes of types. Reviewed by Patrick Salle.

Hindley, R. (1983b). Curry's type-rules are complete with respect to the F-semantics too. *Theoret. Comput. Sci.*, 22(1-2):127–133.

[MR85e:03030b] See Hindley, *The completeness theorem for typing λ-terms.* Theoret. Comput. Sci. 22 (1983), no.1–2, 1–17.

(1984). *Lambda calculus and computer science theory*, Kyoto. Kyoto University Research Institute for Mathematical Sciences. Sûrikaisekikenkyûsho Kôkyûroku No. 515 (1984).

[MR85g:68005] Contents: A. Aiba and M. Nakanishi, Evaluation of an applicative language on cooperating infinite number of processors (pp. 1–15); Shigeo Oyagi, A mathematical theory of Prolog (pp. 16–61); Atsushi Togashi and Shoichi Noguchi, A program transformation from equational programs into logic programs (pp. 62–88); Susumu Hayashi, Adjunction of semifunctors: categorical structures in nonextensional lambda calculus (pp. 89–98); Sachio Hirokawa, Some properties of one-step recurrent terms in lambda calculus (pp. 99–109); Yoshihito Toyama, On the Church-Rosser property for the direct sum of term rewriting systems (pp. 110–133); Hirofumi Yokouchi, General construction of lambda calculus models (pp. 134–176); Takanori Adachi, Powerposets (pp. 177–196). The papers will not be reviewed individually.

Astesiano, E. and Costa, G. (1984). Distributive semantics for nondeterministic typed λ-calculi. *Theoret. Comput. Sci.*, 32(1-2):121–156.

[MR86h:68103] The applicative language PCF, which is based upon the λ-calculus, is extended by a binary choice operator or to serve as a prototype for the study of nondeterminism. Various natural ways of defining operational semantics of this language are discussed, and it is shown that they are not equivalent. A distinguishing feature of the semantics is the preservation of semantic equivalence, which means that programs without occurrences of the or operator which are equivalent with respect to the deterministic semantics are still equivalent with respect to the nondeterministic semantics. The main part of the paper formally develops an operational semantics which is based on outermost derivations together with sharing of arguments and a denotational semantics which is fully abstract with respect to the operational one. It is shown that these semantic definitions preserve deterministic equivalence.

The authors present a thorough treatment of the model construction for the denotational semantics, proving its main properties. Reviewed by E. Fehr.

Burstall, R. and Lampson, B. (1984). A kernel language for abstract data types and modules. In *Semantics of data types (Valbonne, 1984)*, pages 1–50. Springer, Berlin.

[MR86i:68092] Author summary: "A small set of constructs can simulate a wide

variety of apparently distinct features in modern programming languages. Using typed lambda calculus with bindings, declarations, and types as first-class values, we show how to build modules, interfaces and implementations, abstract data types, generic types, recursive types, and unions. The language has a concise operational semantics given by inference rules." For the entire collection see MR 85m:68003.

Coppo, M. (1984). Completeness of type assignment in continuous lambda models. *Theoret. Comput. Sci.*, 29(3):309–324.

[MR85e:03029] Most known models of untyped λ-calculus have a built-in partial order and satisfy a "continuity" condition. This is because they originated from models, not only of λ-calculus, but of various theories of computable functionals. In contrast, continuity has played no role at all in the published semantics for type-assignment systems (systems of rules for assigning types to untyped λ-terms).

For continuous λ-models, the author describes a very natural semantics ("continuous semantics") which remedies this defect. He states two type-assignment rules which must be added to the usual Curry rules to agree with this semantics, and proves the resulting system complete. (Types in this paper are $\to$-types only, not Coppo-Dezani types.)

The author also proves that the fixed-point combinator Y has type $(\alpha \to \alpha) \to \alpha$ in his system. (In fact, to get this result was his original motivation for the semantics.)

Finally, continuous versions of the F-semantics and of Scott's equivalence-relation semantics are defined, and the system is proved complete with respect to these, too. (The result for Scott's semantics is proved by an observation which was first published by the reviewer, but came originally from notes by the author.)

The author begins his paper with a new proof that Curry's original rules are complete. Earlier proofs had used the term- and the filter-model, both strongly tied to syntax. This new proof uses Scott's totally nonsyntactic model, P_ω.

The paper is well-organised and a pleasure to read. Reviewed by R. Hindley.

Coppo, M., Honsell, F., Dezani-Ciancaglini, M., and Longo, G. (1984). Extended type structures and filter lambda models. In *Logic colloquium '82 (Florence, 1982)*, pages 241–262. North-Holland, Amsterdam.

[MR87g:03018] The paper is not well written at all. Some of the main results are not formulated but only left as implicit corollaries; the English is bad; results borrowed from the literature are poorly referenced; et cetera. The persistent reader will be much rewarded, however. The authors considerably extend the method of constructing filter λ-models given by the reviewer, Coppo and Dezani-Ciancaglini [J. Symbolic Logic 48 (1983), no. 4, 931–940; MR 85j:03014], thereby obtaining new characterizations of known models.

An extended abstract type structure (EATS) $\mathcal{S}$ is of the form $\langle S, \leq, \wedge, \to, \omega \rangle$, where $\leq$ is a reflexive, transitive relation on the set S with largest element ω; $a \wedge b$ is the infimum of a and b in S and $\to$ is a binary operation on S satisfying the three conditions $\omega \leq \omega \to \omega$; $(a \to b) \wedge (a \to c) \leq a \to (b \wedge c)$; $a' \leq a$, $b' \leq b' \Rightarrow (a \to b) \leq (a' \to b')$. EATSs are usually considered modulo the relation $a \sim b \Leftrightarrow a \leq b \leq a$. So we can consider $\leq$ as a partial order.

The free EATS over the generators $\{\varphi_0, \varphi_1, \cdots\}$ is denoted by $\mathcal{T}$. If Σ is a set of equations between elements of $\mathcal{T}$, then $\mathcal{S}(\Sigma) = \mathcal{T}/\Sigma$ is the obvious EATS obtained from $\mathcal{T}$ by making some identifications.

Given an EATS $\mathcal{S}$ we can form the applicative structure $|\mathcal{S}|$ as follows. The domain of $|\mathcal{S}|$ consists of all filters in $\mathcal{S}$, i.e. all subsets d of S satisfying the conditions $\omega \in d$;

$a, b \in d \Rightarrow a \wedge b \in d$; $a \in d$, $a \leq b \Rightarrow b \in d$. Application in $|\mathcal{S}|$ is defined by $d.e = \{\sigma \mid \exists \tau \in e\ \sigma \rightarrow \tau \in d\}$. The authors isolate a condition on $\mathcal{S}$ that implies that $|\mathcal{S}|$ is a λ-model: (1) $(a_1 \rightarrow b_1) \wedge \cdots \wedge (a_n \rightarrow b_n) \leq c \rightarrow d, \omega \neq d \Rightarrow \bigwedge\{b_i \mid c \leq a_i\} \leq d$. (It is shown also that this condition is not necessary, only sufficient.) The free EATS $\mathcal{T}$ satisfies (1), as was shown by the reviewer, Coppo, and Dezani-Ciancaglini [op. cit.]. If moreover we have in $\mathcal{S}$ that (2) every a in S can be written in the form $(b_1 \rightarrow c_1) \wedge \cdots \wedge (b_n \rightarrow c_n)$, then $|\mathcal{S}|$ is even an extensional λ-model.

Given a complete lattice $(D, \sqsubseteq)$ with continuous application $. : D \times D \rightarrow D$, the EATS $\mathcal{K}(D) = (K(D), \cap, \subseteq, \rightarrow, D)$ is defined as follows: $K(D) = \{c^{\vee} \mid c$ a compact element of $D\}$; $c^{\vee} = \{d \mid c \sqsubseteq d\}$; $A \rightarrow B = \{d \mid d.A \subseteq B\}$. It is shown that if D is an algebraic lattice which is a λ-model satisfying $\lambda y.xy \sqsubseteq x$, then $|\mathcal{K}(D)| \cong D$. Moreover, for these D one has $\mathcal{K}(D) = \mathcal{S}(\Sigma_D)$, where $\Sigma_D = \{\sigma = \tau \mid$ for all $\xi[\![\sigma]\!]_\xi = [\![\tau]\!]_\xi\}$, for every type valuation $\xi\ \{\varphi_0, \varphi_2, \cdots\} \rightarrow \{X \mid X \subseteq D\}$ being a type valuation. (This valuation is extended to all elements of $\mathcal{T}$ by setting $[\![\sigma \wedge \tau]\!] = [\![\sigma]\!] \cap [\![\tau]\!]$, $[\![\sigma \rightarrow \tau]\!] = [\![\sigma]\!] \rightarrow [\![\tau]\!]$, $[\![\omega]\!] = D$.)

Combining the last two results, the authors obtain interesting characterizations of various versions of Scott's lattice models D_∞. For example, for Park's version D_∞^P in which the fixed-point operator Y is not the least fixed-point operator one has (starting with a two-element D_0) that $D_\infty^P \cong |\mathcal{S}(\Sigma^P)|$, with Σ^P axiomatized by $\varphi_n = \varphi_m$, $\varphi_n = \varphi_n \rightarrow \varphi_n$, for $n, m \geq 0$. For the entire collection see MR 85g:03006. Reviewed by Henk Barendregt.

Dezani-Ciancaglini, M. and Margaria, I. (1984). F-semantics for intersection type discipline. In *Semantics of data types (Valbonne, 1984)*, pages 279–300. Springer, Berlin.

[MR86k:68063] Author summary: "The aim of this paper is to investigate the soundness and completeness for the F-semantics (F-soundness and F-completeness) of some modifications of the intersection type discipline for terms of the (untyped) λ-calculus. As pointed out by Scott, the key of a λ-model is the set F of the elements representing functions. The F-semantics of types takes into account that the intuitive meaning of '$\sigma \rightarrow \tau$' is 'a function with domain σ and range τ' and interprets $\sigma \rightarrow \tau$ as a subset of F.

"The type theories which induce F-complete type assignments are characterized. It turns out that a type assignment is F-complete if and only if the following are derived rules for it:

$$\frac{\tau^M \quad M =_\beta N}{\tau^N}, \quad \frac{\varphi \wedge \omega^n \rightarrow \omega^M}{\varphi \lambda z_1 \cdots z_n . M z_1 \cdots z_n} \quad \text{if } z_1, \cdots, z_n \notin \mathrm{FV}(M).\text{"}$$

For the entire collection see MR 85m:68003.

Fehr, E. (1984). Expressive power of typed and type-free programming languages. *Theoret. Comput. Sci.*, 33(2-3):195–238.

[MR86g:68108] Author summary: "The classes of typed and type-free lambda-schemes are studied with respect to their use as control structures of programming languages. The semantics of both classes are analyzed in the same semantical domains. In particular, it is shown that typed λ-schemes are translatable into equivalent type-free λ-schemes but not vice versa. Furthermore, we prove that the class of type-free λ-schemes is universal in the sense that in the initial models, all recursively enumerable Σ-trees (Σ is the set of operation symbols) are definable."

Hikita, T. (1984). On the average size of Turner's translation to combinator programs. *J. Inform. Process.*, 7(3):164–169.

[MR86g:68134] Author summary: "D. A. A. Turner proposed [Software Practical Experience 9 (1979), 31–49; per bibl.] an interesting method of implementation for functional programs by first translating them to combinator expressions without variables, and then reducing the graphs they represent. One of the points of concern for this method was the expansion of the sizes of expressions resulting from the translation. J. R. Kennaway has recently shown that the worst case of the size of this translation is of order n^2, where n is the size of the original program. In this paper we choose a theoretical definition of the average size of a translation, and show that the order of the average size is at most $n^{3/2}$. Partial results on lower bounds of the average size are also shown in the case of programs with one distinct variable. Finally, numerical results for the average size are exhibited."

Hindley, J. R. (1984). Coppo-Dezani types do not correspond to propositional logic. *Theoret. Comput. Sci.*, 28(1-2):235–236.

[MR86i:03015] Author summary: "This note gives a simple counterexample to back up G. Pottinger's explanation of the difference between propositional logic and the type theories of Pottinger [To H. B. Curry essays on combinatory logic, lambda calculus and formalism, 561–577, Academic Press, London, 1980; MR 82j:03014] and M. Coppo and M. Dezani-Ciancaglini [Arch. Math. Logik Grundlag. 19 (1978/79), no. 3-4, 139–156; MR 81d:03014]. The example depends on a theorem of C.-B. Ben-Yelles ["Type-assignment in the lambda calculus", Ph.D. Thesis, Inst. de Math., Univ. des Sciences, Algiers, 1979; per bibl.]."

Mitchell, J. (1984). Type inference and type containment. In *Semantics of data types (Valbonne, 1984)*, pages 257–277. Springer, Berlin.

[MR86i:68096] Author summary: "Type inference, the process of assigning types to untyped expressions, may be motivated by the design of a typed language or semantical considerations on the meanings of types and expressions. A typed language GR with polymorphic functions leads to the GR inference rules. With the addition of an 'oracle' rule for equations between expressions, the GR rules become complete for a general class of semantic models of type inference. These inference models are models of untyped lambda calculus with extra structure similar to models of the typed language GR. A more specialized set of type inference rules, the GRS rules, characterize semantic typing when the functional type $\sigma \to \tau$ is interpreted as all elements of the model that map σ to τ and the polymorphic type $\forall t.\sigma(t)$ is interpreted as the intersection of all $\sigma(\tau)$. Both inference systems may be reformulated using rules for deducing containments between types. The advantage of the type inference rules based on containments is that proofs correspond more closely to the structure of terms." For the entire collection see MR 85m:68003.

Poigné, A. (1984). Higher order data structures—Cartesian closure versus λ-calculus. In *STACS 84 (Paris, 1984)*, pages 174–185. Springer, Berlin.

[MR86b:68038] Author's summary: "We discuss connections between typed λ-

calculus and Cartesian closure and prove equivalence of the theories 'up to abstraction'. This is a working-out of ideas of Scott and Lambeck but in an abstract data type environment. The results serve as a basis for the discussion of higher-order specifications. We demonstrate that higher-order equations based on λ-calculus are more appropriate if the equivalence of λ-calculus and Cartesian closure is to be preserved. We construct higher-order theories for higher-order specifications. For higher-order models we discuss existence of initial models and completeness of higher-order theories." For the entire collection see MR 85i:68002.

Reynolds, J. C. (1984). Polymorphism is not set-theoretic. In *Semantics of data types (Valbonne, 1984)*, pages 145–156. Springer, Berlin.

[MR86e:03016] Author summary: "The polymorphic, or second-order, typed lambda calculus is an extension of the typed lambda calculus in which polymorphic functions can be defined. In this paper, we prove that the standard set-theoretic model of the ordinary typed lambda calculus cannot be extended to model this language extension." For the entire collection see MR 85m:68003.

Ronchi Della Rocca, S. and Venneri, B. (1984). Principal type schemes for an extended type theory. *Theoret. Comput. Sci.*, 28(1-2):151–169.

[MR86d:03017] From the introduction: "M. Coppo, M. Dezani-Ciancaglini and Venneri [Z. Math. Logik Grundlag. Math. 27 (1981), no. 1, 45–58; MR 82j:03011] introduced a type theory as an extension of basic functionality theory [H. B. Curry and R. Feys , Combinatory logic, Vol. I, North-Holland, Amsterdam, 1958; MR 20 #817; ibid., second printing, 1974; MR 39 #5368; Curry, J. R. Hindley and the reviewer, ibid., Vol. II, Chapter 14, North-Holland, Amsterdam, 1972; Zbl 242:02029]. The system increases the power of Curry's theory (by generalizing both type structure and assignment rules) so that only unsolvable terms have a trivial functional characterization. Moreover, Coppo, Dezani-Ciancaglini and Venneri [To H. B. Curry: essays on combinatory logic, lambda calculus and formalism, 535–560, Academic Press, London, 1980; MR 82d:03022] proved, for any term X, the existence of a 'principal type scheme' (p.t.s.) from which all (and only) the type schemes of X can be generated, by suitable operations. This shows the internal coherence of the (in general infinite) set of type schemes which can be assigned to a term. Lastly, in a paper by H. Barendregt, Coppo and Dezani-Ciancaglini [J. Symbolic Logic 48 (1983), no. 4, 931–940; MR 85j:03014] the system has been further enhanced. It becomes natural to ask if a notion of p.t.s. can be defined in this last theory. In the present paper we prove that the p.t.s. of a term X, as defined by Coppo et al. [op. cit.; MR 82d:03022], is also the p.t.s. of X in this theory. Let us recall that, in the basic functionality theory, only one operation (that of substitution) is sufficient to obtain from the p.t.s. of a term X all the type schemes of X, and this operation also has the property that it can be applied to an arbitrary type scheme of X to obtain again a type scheme of X [Hindley, Trans. Amer. Math. Soc. 146 (1969), 29–60; MR 40 #7118]. In the paper by Coppo et al. [op. cit.; MR 82d:03022] it was necessary to introduce, besides substitution, a new operation which, however, lacks the property above, i.e., it can be applied only to p.t.s.'s. An improvement given in this paper is that all operations here defined (in order to obtain from the p.t.s. of a term all the type schemes of the term) generate deducible type schemes for X starting from arbitrary type schemes of X."

The types of this system include, in addition to the arrow types $(\sigma \to \tau)$ (functions from σ to τ), intersection types $(\sigma \cap \tau)$ (in the set-theoretic sense) and a universal type ω. There is also an inclusion relation $\leq$ between types. Types are assigned to

terms by means of natural deduction introduction and elimination rules for $\to$ and $\cap$, an axiom ωM for each term M, and the rule $(\leq)$ σM, $\sigma \leq \tau \vdash \tau M$. A basis scheme is a set of assignment statements assigning types to variables.

The operations used to obtain instances of the principal type schemes take a pair $\langle B, \tau \rangle$ of a basis scheme and a type scheme to another such pair and correspond to operations on deductions. The three operations that allow any type scheme to be obtained from the principal type scheme are the following: (1) substitution (replacing type variables by type schemes), (2) rise (adding a new instance of rule $(\leq)$ to the end of a deduction), and (3) expansion (replacing a subdeduction by new subdeductions, all of which, including the original one, are substitution instances of a common subdeduction, and then combining the new subdeductions by the introduction rule for $\cap$). These operations are sound in the sense that if one of them takes $\langle B, \tau \rangle$ to $\langle B', \tau' \rangle$ and if $B \vdash \tau M$, then $B' \vdash \tau' M$.

The main proof makes heavy use of the set of approximate normal forms and the set of approximates of a term [Barendregt, The lambda calculus: its syntax and semantics, North-Holland, Amsterdam, 1981; MR 83b:03016]. The principal basis scheme and principal type scheme are initially defined only for approximate normal forms and later extended to arbitrary terms.

Reviewer's remarks: The definition of an $\to$-cut on p. 155 is unclear. This unclear wording appeared previously [Barendregt et al., op. cit.; MR 85j:03014], but there was a diagram there to help make the meaning clear. In this paper, where there is no diagram, the statement should be corrected. The reviewer suggests the following wording: "An $\to$-cut in D is a statement occurrence ρZ in D which is the major premise of $(\to E)$ and is obtained by $k \geq 0$ applications of $(\beta\eta)$ immediately preceded by $(\to I)$. The length of this cut is $k+1$." (The rule $(\beta\eta)$ is a rule of an associated system in which normalization is defined, not the main system.) Reviewed by J. P. Seldin.

Smith, J. (1984). An interpretation of Martin-Löf's type theory in a type-free theory of propositions. *J. Symbolic Logic*, 49(3):730–753.

[MR85j:03106] This paper is concerned with a new interpretation of Martin-Löf's type theory (TT). For this purpose a logical theory (LT) is introduced which is based on propositions and combinator terms and has Frege structures as models. The main result states that TT can be interpreted in LT. Reviewed by Gerhard Jäger.

Bunder, M. W. (1985). Possible forms of evaluation or reduction in Martin-Löf type theory. *Theoret. Comput. Sci.*, 41(1):113–120.

[MR88a:03150] The background of this paper is given by the procedures which reduce expressions in Martin-Löf type theory to normal form. These reduction rules can also be interpreted in different ways, e.g., as reductions in the λ-calculus. This paper studies the properties that such general reductions must have in order to satisfy Martin-Löf's conditions. Reviewed by M. M. Richter.

Cocchiarella, N. B. (1985). Two λ-extensions of the theory of homogeneous simple types as a second-order logic. *Notre Dame J. Formal Logic*, 26(4):377–407.

[MR87c:03014] The author further develops ideas which he first mooted in 1972

[Nous 6 (1972), no. 2, 165–187; MR 58 #27296], and elaborated in several subsequent papers [Notre Dame J. Formal Logic 15 (1974), 552–568; MR 50 #6784; Logique et Anal. (N.S.) 17 (1974), 61–87; MR 50 #12643; Nous 9 (1975), no. 1, 33–53; MR 57 #9473; especially Notre Dame J. Formal Logic 20 (1979), no. 3, 505–524; MR 80k:03058]. Two extensions of that theory are here presented as λ-calculi of the second order. Type restrictions at the second order apply only to adicity, and each predicate can occur also as a subject, of other predicates and of itself. The two extensions differ in that one of them requires homogeneous stratification of wffs, that is, the arguments of any predicate can all be taken to be of the same level; the other does not, but avoids paradox by the free logical move of not insisting that every subject term denotes. The logical paradoxes are narrowly avoided not, therefore, by type restriction but essentially by J. F. Thomson's theorem [Analysis 15 (1954/55), no. 1, 1–13] to the effect that there is no individual matching the paradoxical properties. Semantics are given for the two systems, and completeness and relative consistency results are proved. Reviewed by Stephen Read.

Constable, R. L. and Mendler, N. P. (1985). Recursive definitions in type theory. In *Logics of programs (Brooklyn, N.Y., 1985)*, pages 61–78. Springer, Berlin.

[MR87a:68112] Author summary: "We offer a new account of recursive definitions for both types and partial functions. The computational requirements of the theory restrict recursive type definitions involving the total function-space constructor ($\rightarrow$) to those with only positive occurrences of the defined type. But we show that arbitrary recursive definitions with respect to the partial function-space constructor are sensible. The partial function-space constructor allows us to express reflexive types of Scott's domain theory (as needed to model the lambda calculus) and thereby reconcile parts of domain theory with constructive type theory." For the entire collection see MR 86f:68007.

Coppo, M. (1985). A completeness theorem for recursively defined types. In *Automata, languages and programming (Nafplion, 1985)*, pages 120–129. Springer, Berlin.

[MR819247] This item will not be reviewed individually. For details of the collection in which this item appears see 87a:68002.

Koletsos, G. (1985). Church-Rosser theorem for typed functional systems. *J. Symbolic Logic*, 50(3):782–790.

[MR87c:03025] Most known proofs of the Church-Rosser (CR) theorem for the typed λ-calculus are based on an analysis of the reduction relation and do not use the type structure in an essential way. In this paper the author gives a new proof of the CR theorem for the typed λ-calculus (and some well-known extensions) using the technique introduced by W. W. Tait in the proof of the normalization theorem for the functionals of finite type [same journal 32 (1967), 198–212; MR 36 #2483], thus exploiting the type structure in an essential way. In particular, the proof is based on the definition of a predicate $\mathrm{MON}^\rho(a)$ ("a is monovalued"), where ρ is a type and a a term of type ρ, corresponding to Tait's notion of convertibility. $\mathrm{MON}^\rho(a)$ is defined by induction on types in the following way: $\mathrm{MON}^0(a)$ if and only if a has

CR (where 0 is the basic type); $\mathrm{MON}^{\sigma\to\tau}(a)$ if and only if a has CR and, for every term b, $\mathrm{MON}^{\sigma}(b)$ implies $\mathrm{MON}^{\tau}(a(b))$. Then it is proved, by an induction on types (split into two lemmas) and an induction on terms, that each term a is monovalued, which implies that a has CR.

An advantage of this technique is that the proof easily generalizes to (typed) infinitely long terms. In this case the other proofs of CR are complicated by the fact that a reduction rule with infinite premises (or, alternatively, infinite reductions) must be considered [see, e.g., G. Mitschke, To H. B. Curry: essays on combinatory logic, lambda calculus and formalism, 243–257, Academic Press, London, 1980; MR 82g:03021].

Using the same technique, the Church-Rosser theorem is also proved for Gödel's functionals of finite type and for Girard's impredicative F-system (in this latter case, following Girard, the notion of candidate of monovaluedness is introduced). Reviewer's remark: A proof of the CR theorem for the simply typed λ-calculus which uses a variant of Tait's technique has also been given independently by R. Statman [Inform. and Control 65 (1985), no. 2, 85–97]. Reviewed by M. Coppo.

Reynolds, J. C. (1985). Three approaches to type structure. In *Mathematical foundations of software development, Vol. 1 (Berlin, 1985)*, pages 97–138. Springer, Berlin.

[MR818017] This item will not be reviewed individually. For details of the collection in which this item appears see 86j:68005 .

Statman, R. (1985). Equality between functionals revisited. In *Harvey Friedman's research on the foundations of mathematics*, pages 331–338. North-Holland, Amsterdam.

[MR87m:03018] Summary: "In this note we try to place a remarkable little paper by H. M. Friedman [Logic colloquium (Boston, Mass., 1972/1973), 22–37, Lecture Notes in Math., 453, Springer, Berlin, 1975; MR 52 #2833] in the context of what we know today about the model theory of the typed λ-calculus. In order to do this, it is appropriate to survey recent work in this area, in so far as it touches on issues raised by Friedman. We do not intend a general survey; much will be left out. In particular, we omit discussion of unification, fragments of L.C.F., and polymorphic types, which are areas of interest to the author, and the specialist will surely find other omissions. However, we try to show the reader how Friedman's paper lays the foundation for the general model theory of the typed λ-calculus.

"The plan of this note is the following. We begin with a brief, informal introduction to the subject. The reader is then advised to read Friedman's paper. It is, after all, quite accessible and easy to read. We then proceed to consider the three major issues touched on in Friedman's paper as we view them today. These issues are (1) completeness theorems, (2) the solvability of higher type functional equations, and (3) logical relations." For the entire collection see MR 87b:03004.

Asperti, A. and Longo, G. (1986). Relating type-structures: partial variations on a theme of Friedman and Statman (preliminary version). In *CAAP '86 (Nice, 1986)*, pages 266–287. Springer, Berlin.

[MR848800] This item will not be reviewed individually. For details of the collection in which this item appears see 87d:68003.

Coquand, T. (1986). Sur l'analogie entre les propositions et les types. In *Combinators and functional programming languages (Val d'Ajol, 1985)*, pages 71–84. Springer, Berlin.

[MR88i:03021] This expository article analyzes from the point of view of a computer scientist the analogy between propositions and types that was first noticed by H. B. Curry [Curry and R. Feys, Combinatory logic, Vol. I, North-Holland, Amsterdam, 1958; MR 20 #817], developed by W. A. Howard [in To H. B. Curry: essays on combinatory logic, lambda calculus and formalism, 480–490, Academic Press, London, 1980; MR 82g:03094] and others (for a survey of the literature see a book by J. R. Hindley and the reviewer [Introduction to combinators and λ-calculus, see p. 194, Cambridge Univ. Press, Cambridge, 1986]), and used in a systematic manner by N. G. de Bruijn [in To H. B. Curry: essays on combinatory logic, lambda calculus and formalism, 579–606, Academic Press, London, 1980; MR 81m:03017].

The paper begins with the implication fragment of the minimal propositional calculus, and shows that the natural deduction formulation is essentially analogous to ordinary type assignment to λ-terms, where the analogy is that the formulas of the logic correspond to the types of the type assignment and the proofs correspond to the λ-terms. Similarly, a Hilbert-style formulation is analogous to type assignment to combinators. This analogy is really an exact correspondence which, along with the well-known normalization theorem, makes it possible to show that Peirce's law, viz. $((x \Rightarrow y) \Rightarrow x) \Rightarrow x$, is not provable in minimal logic. The author then shows that conjunction can be added by looking at the usual pairing and projection operators, that disjunction can be added by looking at the disjoint sum with its injection and case functions, and that negation can be obtained by the well-known method of adding a formula $\perp$ corresponding to an empty type.

The author then passes on to first-order logic by considering the special case of first-order arithmetic. This is followed by a discussion of P. Martin-Löf's type theory, as presented in his book Intuitionistic type theory [Bibliopolis, Naples, 1984; Zbl 571:03030], especially its existential type. This is followed by a discussion of the existential type and the notion of an "abstract type"; i.e., asserting that there is a type such that .... This leads to a discussion of the second-order polymorphic typed λ-calculus, introduced independently by J.-Y. Girard ["Interpretation fonctionnelle et elimination des coupures de l'arithmetique d'ordre superieur", These d'Etat, Univ. Paris, VII, Paris, 1972] and J. C. Reynolds [in Programming symposium (Paris, 1974), 408–425, Lecture Notes in Comput. Sci., 19, Springer, Berlin, 1974; MR 56 #17187] and to the author's own extension of that system to higher orders ["Une theorie des constructions", These de 3eme cycle, Univ. Paris VII, Paris, 1985; per bibl.].

The general level of exposition in the paper is good, but there are a few difficult places. In the first paragraph on p. 74, the letter "B" seems to be used in two different senses. In the second theorem on p. 75, the hypothesis is that A is a formula, but nothing is said about B. The reviewer is unable to follow the comments at the top of p. 76 about interpreting a functional of type $((A \to \perp) \to \perp) \to A$ as a choice function. Finally, the English-speaking reader may be interested in the fact that an English translation of this paper has been done by Walt Hill of Hewlett-Packard Laboratories; it was circulated with the working papers of the Institute on Logical Foundations of Functional Programming held at the University of Texas, Austin, 8–12 June 1987. For the entire collection see MR 87j:03004. Reviewed by J. P. Seldin

Dezani-Ciancaglini, M. and Margaria, I. (1986). A characterization of F-complete type assignments. *Theoret. Comput. Sci.*, 45(2):121–157.

[MR88k:68058] Summary: "The aim of this paper is to investigate the soundness and completeness of the intersection type discipline (for terms of the (untyped) λ-calculus) with respect to the F-semantics (F-soundness and F-completeness).

"As pointed out by Scott, if D is the domain of a λ-model then there is a subset F of D whose elements are the 'canonical' representatives of functions. The F-semantics of types takes into account that the intuitive meaning of '$\sigma \to \tau$' is 'the type of functions with domain σ and range τ' and interprets $\sigma \to \tau$ as a subset of F.

"The type theories which induce F-complete type assignments are characterized. It follows that a type assignment is F-complete if and only if equal terms get equal types and, whenever M has a type $\phi \wedge \omega^n \to \omega$, where ϕ is a type variable and ω is the 'universal' type, the term $\lambda_{z_1 \cdots z_n} M_{z_1 \cdots z_n}$ has type ϕ. Here we assume that $z_1, \cdots, z_n$ do not occur free in M."

Girard, J.-Y. (1986). The system F of variable types, fifteen years later. *Theoret. Comput. Sci.*, 45(2):159–192.

[MR88j:68105] Summary: "The semantic study of system F stumbles on the problem of variable types for which there was no convincing interpretation; we develop here a semantics based on the category-theoretic idea of direct limit, so that the behaviour of a variable type on any domain is determined by its behaviour on finite ones, thus getting rid of the circularity of variable types. To do so, one has first to simplify somehow the extant semantic ideas, replacing Scott domains by the simpler and more finitary qualitative domains. The interpretation obtained is extremely compact, as shown on simple examples. The paper also contains the definitions of a very small 'universal model' of lambda-calculus, and investigates the concept totality."

Leivant, D. (1986). Typing and computational properties of lambda expressions. *Theoret. Comput. Sci.*, 44(1):51–68.
[MR88a:03033]

There are two ways of introducing types into a system of λ-calculus. In the traditional way, each typed λ-term is defined with a unique built-in type, given to it as part of its definition. The other approach is that of considering types as predicates representing properties of untyped terms. Types, in this case, can be "assigned" to terms by means of formal inference rules. This notion of typing was first introduced in Curry's functionality theory and has been incorporated in some (functional) programming languages, such as ML and HOPE. A second-order universal quantifier over type variables, as in J.-Y. Girard's F-system ["Interpretation functionelle et elimination des coupures dans l'arithmetique d'ordre superieur", These d'Etat, Univ. Paris VII, Paris, 1972] can also be introduced in the type inference approach. Extensions of basic functionality theory have also been proposed by the reviewer, M. Dezani-Ciancaglini and B. M. Venneri [Z. Math. Logik Grundlag. Math. 27 (1981), no. 1, 45–58; MR 82j:03011], who introduced an intersection operator $\wedge$ for type formation and a basic type ω intended as the type of all terms (remark: there is no natural counterpart of these operators in typed systems).

In a type inference system it makes sense to ask which untyped terms can be given a type or whether there are interesting properties of (untyped) terms that can be characterized by the set of their types.

In this paper the author integrates and generalizes results by Girard and Coppo-Dezani-Venneri to give a characterization, in terms of typing, of the normalizable, strongly normalizable and solvable terms (a term is said to be solvable if it reduces to a term of the form $\lambda x_1 \cdots x_n . x_j N_1 \cdots N_m$ with $n, m \geq 0,\ j \geq 0$). For example, a term is strongly normalizable if and only if it can be given a type without using type ω. Moreover, generalizing Girard's technique for proving strong normalization in his F-system, he gives a uniform and elegant framework to prove convergence properties (such as (strong) normalizability and solvability) implied by particular typings. In order to obtain a full characterization of these properties, however, the essential operators are the $\wedge$ type constructor and the type ω. When the second-order quantifier over types is added to the system, the characterizations of the strongly normalizable and solvable terms still hold while that of the normalizable terms fails.

Most results for type inference systems can easily be transported to the corresponding typed systems. In the paper, an alternative proof of the strong normalization theorem for the second-order typed λ-calculus is obtained as a corollary of the analogous theorem for the inference version of the system. Reviewed by M. Coppo.

Longo, G. and Martini, S. (1986). Computability in higher types, $P\omega$ and the completeness of type assignment. *Theoret. Comput. Sci.*, 46(2-3):197–217.

[MR88h:03018] As is well known, $P\omega$ is a universal Scott-Ershov domain in the sense that any countably based (in particular, effective) domain can be embedded into it by an (effective) retraction. This means that for any [effective] domain X there are $A_X \subseteq P\omega$ and continuous [effective] mappings $g\colon X \to A_X$ and $f\colon A_X \to X$ such that $f \circ g = \mathrm{id}_X$ (denoted $X \lhd A_X$). It is also well known that $P\omega$ can be seen as an applicative structure, that is, there is an effective binary operation $\circ$ on $P\omega$ corresponding to function application. Thus for subsets A, B of $P\omega$ we obtain the corresponding function space $A \to B = \{d \in P\omega\colon \text{ for all } a \in A,\ d \circ a \in B\}$. The authors show that if $X \lhd A_X$ and $Y \lhd A_Y$ then $\mathrm{Cont}(X,Y) \lhd A_X \to A_Y$, where $\mathrm{Cont}(X,Y)$ is the domain of continuous functions from X to Y. They then extend this result to show that $X^\sigma \lhd A^\sigma$ for each type symbol σ built up from atomic type symbols using $\to$, provided this holds for the atomic types, where $\{X^\sigma\}$ is the usual type structure of domains and $\{A^\sigma\}$ the corresponding type structure in $P\omega$. Using Yu. L. Ershov's work [in Logic colloquium '76 (Oxford, 1976), 455–467, North-Holland, Amsterdam, 1977; MR 58 #21541], the authors then embed the type structure of the continuous functionals into $P\omega$, the claim being that $P\omega$ is less vague than the original type structure. In the final section of the paper it is shown that the type structure $\{P^\sigma\}$ in $P\omega$ built on the atomic type corresponding to the partial functions on ω gives a sound and complete model for typable λ-terms.

MacQueen, D., Plotkin, G., and Sethi, R. (1986). An ideal model for recursive polymorphic types. *Inform. and Control*, 71(1-2):95–130.

[MR88h:68058] The purpose of this paper is to give a domain-theoretic semantics, together with sound rules of inference, for typing expressions of the untyped λ-calculus (equipped with a suitable set of constants). The types that can be assigned to expressions include basic types (such as Booleans and integers), product, sum, union, intersection, and function space types, universally and existentially quantified types, and recursively defined types. The technical approach is as follows: There is a semantic domain V constructed in the usual manner as the solution of a domain

equation so that (for each environment) every expression denotes an element of V. Types are ideals of V closed under limits of chains. The main technical difficulty consists in interpreting all the type constructions as operations on ideals, including formation of recursive types. The approach taken by the authors is an elegant application of the Banach fixed-point theorem for contractive maps on complete metric spaces. They exhibit a natural complete metric space structure for the space $\mathcal{I}(V)$ of ideals of V, and prove that all the type forming operations are contractive (or nonexpansive). This then provides a semantic function $\mathcal{T}T\mathrm{Exp} \to T\mathrm{Env} \to \mathcal{I}(V)$ assigning (for each type environment) a type to each type expression. The authors then give a Gentzen-like type inference system for typing λ-calculus expressions, prove its soundness, and illustrate its use by typing the Y combinator. An appendix treats the case of strong ideals that are, in addition, closed under finite sups whenever they exist.

Nordström, B. (1986). Multilevel functions in Martin-Löf's type theory. In *Programs as data objects (Copenhagen, 1985)*, pages 206–221. Springer, Berlin.

[**MR846666**] This item will not be reviewed individually. For details of the collection in which this item appears see 87b:68002.

Rezus, A. (1986). Semantics of constructive type theory. *Libertas Math.*, 6:1–82.

[**MR88a:03034**] The author presents an interpretation of P. Martin-Löf's type theory CST [Logic, methodology, and philosophy of science, VI (Hannover, 1979), 153–175, North-Holland, Amsterdam, 1982; MR 85d:03112] with one universe and without identity types. In the author's approach, following D. S. Scott [SIAM J. Comput. 5 (1976), no. 3, 522–587; MR 55 # 10262], types are translated into objects in a model of the type-free lambda calculus. The goal is to show that at least a subsystem of constructive type theory is intelligible from a classical nonconstructivistic viewpoint. A second goal is to provide an abstract setting for the design of a large class of typed programming languages. The author's presentation includes an extensive discussion of related work on models of constructive type theory. Reviewed by Henry Africk.

Breazu-Tannen, V. and Coquand, T. (1987). Extensional models for polymorphism. In *TAPSOFT '87, Vol. 2 (Pisa, 1987)*, pages 291–307. Springer, Berlin.

[**MR900622**] This item will not be reviewed individually. For details of the collection in which this item appears see 88e:68011.

Coppo, M., Dezani-Ciancaglini, M., and Zacchi, M. (1987). Type theories, normal forms, and D_∞-lambda-models. *Inform. and Comput.*, 72(2):85–116.

[**MR88m:03025**] In a paper by the reviewer, Coppo and Dezani-Ciancaglini [J. Symbolic Logic 48 (1983), no. 4, 931–940; MR 85j:03014] the notion of filter model was introduced, based on partially ordered types including type variables, a type ω

and closed under $\to$ and $\wedge$. Coppo, Dezani-Ciancaglini, F. Honsell and G. Longo [Logic Colloquium '82 (Florence, 1982), 241–262, North-Holland, Amsterdam, 1984; MR 87g:03018] extended the method by allowing in addition to ω other atomic types and different partial orderings. In their paper it was shown how a large class of D_∞ models could be obtained as filter models on these generalized types. The present authors proceed in the opposite direction by showing that the filter model of a type structure T^* studied by Coppo and Dezani-Ciancaglini [Arch. Math. Logik Grundlag. 19 (1978/79), no. 3-4, 139–156; MR 81d:03014] can be described as a model D^*_∞ with a nonstandard initial projection. The authors show for this model the approximation theorem, using as a lemma that if a λ-term M has type σ, then already an approximant of M has that type. It is interesting to see that the lemma is proved using, in a different context, the well-known computability method of W. Tait [J. Symbolic Logic 32 (1967), 198–212; MR 36 #2483]. As a second main result the authors show that the theory of the model D^*_∞ is J. H. Morris' extensional theory T_{NF} ["Lambda calculus models of programming languages", Ph.D. Thesis, Massachusetts Inst. Tech., Cambridge, Mass., 1968; per bibl.] in which for closed terms one has $M = N$ if and only if $\forall F$ [FM has an nf $\Leftrightarrow$ MFN has an nf]. A third result concerns the class $\mathcal{N} = \{M\colon\ \forall\vec{N}\ (\vec{N}$ have nf's $\Rightarrow M\vec{N}$ has an nf$)\}$introduced in a paper by C. Böhm and Dezani-Ciancaglini [in λ-calculus and computer science theory (Rome, 1975), 96–121, Springer, Berlin, 1975; MR 58 #5143]. It is shown that exactly those terms are in $\mathcal{N}$ that have a given type $\phi_\perp$ in the type structure T^*. Finally, extending a result of Dezani-Ciancaglini and F. Ermine [Fund. Inform. 5 (1982), no. 2, 129–141; MR 84b:03028], the authors obtain some maximal monoids of combinators as sets of terms having specified types in T^*. Reviewed by Henk Barendregt.

Coquand, T. and Ehrhard, T. (1987). An equational presentation of higher order logic. In *Category theory and computer science (Edinburgh, 1987)*, pages 40–56. Springer, Berlin.

[MR89k:03006] From the introduction: "We propose an equational version, based on a paper by R. A. G. Seely [J. Symbolic Logic 52 (1987), no. 4, 969–989; MR 89g:03019], of the higher-order type system of J.-Y. Girard ["Interpretation fonctionnelle et elimination des coupures de l'arithmetique d'ordre superieur", These d'Etat, Univ. Paris VII, 1972]. The situation here is similar to the one in a Cartesian closed category with respect to a simple typed calculus: there now exist a number of equational presentations of this theory. One of the most obvious advantages of this kind of presentation is that it avoids the problem of α-conversion (which is crucial for an implementation). This is particularly important in higher-order calculi, since the traditional restriction for the binding of type variables seems difficult to solve.

"The main idea of Seely's formalism is to see quantifications as adjunctions. It is an old paradigm, first discovered by F. W. Lawvere [Dialectica 23 (1969), no. 3-4, 281–296], and developed by Seely in two papers [Z. Math. Logik Grundlag. Math. 29 (1983), no. 6, 505–542; MR 86c:03054; op. cit., 1987]. What we present here is just a syntactic and equational formalization of Seely's categorical presentation. We illustrate our formalism by showing that it furnishes a nice way of checking that the constructions of Girard [Theoret. Comput. Sci. 45 (1986), no. 2, 159–192; MR

88j:68105] and Coquand, C. Gunter and G. Winksel ["Polymorphism and domain equations", Tech. Rep. No. 107, Cambridge; per bibl.] are models of $F\omega$." For the entire collection see MR 88j:68002.

Dezani-Ciancaglini, M. and Venneri, B. (1987). A system for assigning partial types. In *Proceedings of the conferences on mathematical logic, Vol. 3 (Italian) (Siena, 1985; Padova, 1985; and Siena, 1986)*, pages 389–393, Siena. Univ. Siena.

[**MR904916**] This item will not be reviewed.

Girard, J.-Y. (1987). Le lambda-calcul du second ordre. *Astérisque*, (152-153):4, 173–185 (1988). Séminaire Bourbaki, Vol. 1986/87.

[**MR89i:03028**] In this paper, which has an expository character, the author discusses his system F (1970) and the main theoretical results concerning this system. He also briefly sketches some recent developments in this area: linear logic and functional arithmetics.

The author's system F is a second-order typed lambda calculus which incorporates in the well-known (Curry/Howard) manner a concise coding of intuitionistic logic in a natural deduction style. (Around 1970, de Bruijn, the author and Martin-Löf were the first to exploit these ideas.) The addition of a few definitions (about Booleans, integers, product types, etc.) enables one to represent in the system F all recursive functions which terminate provably, the proof being given in PA_2. The system F has the properties of confluence (Church-Rosser) and strong normalization. It has a comparatively simple denotational semantics.

Systems like this may turn out to be of considerable importance for computer science, as recent research shows: the underlying pure (nontyped) lambda-calculus can act as a language for representing the algorithmic contents of computer programs, while the types represent the specifications of such algorithms. For the entire collection see MR 88k:00019. Reviewed by R. P. Nederpelt.

Hagino, T. (1987). A typed lambda calculus with categorical type constructors. In *Category theory and computer science (Edinburgh, 1987)*, pages 140–157. Springer, Berlin.

[**MR89e:03021**] The author describes a lambda calculus with simple types and type constructors using a category-theoretic mechanism. Types are of the form $\sigma \to \tau$, $\underline{\mu}\rho.(\sigma_1, \cdots, \sigma_n)$ or $\overline{\mu}\rho.(\sigma_1, \cdots, \sigma_n)$, where $\to$ is the usual simple type constructor, $\underline{\mu}$ the least, $\overline{\mu}$ the greatest fixed point operator and $\sigma_1, \cdots, \sigma_n$ a sequence of types.

A natural extension of the notions of T-algebra and T-coalgebra, called an F, G-dialgebra, then enables one, as the author shows, to define a number of interesting types, e.g., the product $\sigma \times \tau \equiv \overline{\mu}\rho.(\sigma, \tau)$, the coproduct $\sigma + \tau \equiv \underline{\mu}\rho.(\sigma, \tau)$, the type of natural numbers $\omega \equiv \underline{\mu}\rho.(1, \rho)$, the type for ordinals and the type for finite lists.

The Church-Rosser theorem and the strong normalization theorem hold for the described system. The latter is proved by a computability argument. The usefulness of the possibility of defining the above-mentioned types is of course that theoretical results such as the normalization theorem stay valid for the system with the product operator, the coproduct operator, etc. Finally, the author relates his system to ML. For the entire collection see MR 88j:68002. Reviewed by R. P. Nederpelt.

Krivine, J.-L. (1987). Un algorithme non typable dans le système F. *C. R. Acad. Sci. Paris Sér. I Math.*, 304(5):123–126.

[MR88f:03013] Let F be the system of second-order typed lambda-calculus introduced by J.-Y. Girard [Proceedings of the second Scandinavian logic symposium (Oslo, 1970), 63–92, North-Holland, Amsterdam, 1971; MR 53 #12895]. F is the same as J. C. Reynolds' system [Programming symposium (Paris, 1974), 408–425, Lecture Notes in Comput. Sci., 19, Springer, Berlin, 1974; MR 56 #17187]. In F, let N be the type $(\forall\phi)((\phi\to\phi)\to(\phi\to\phi))$, the type of the Church numerals.

In untyped lambda-calculus, let $M \equiv uC_*(\lambda x.a)(vC_*(\lambda x.b))$, where a, b, u, v, x, y are distinct variables and $C_* \equiv \lambda xy.yx$. M represents the numerical relation $m \le n$, in the sense that for all Church numerals Z_m, Z_n, $[Z_m/u, Z_n/v]M$ converts to a if $m \le n$, and to b otherwise.

The author proves that if u, v are assumed to have type N, then no type can be assigned to M in the system F.

He then describes an extension of F which does assign a type to M under the assumption that u, v have type N, and for which the normalisation theorem, which holds for F, still holds. Reviewed by R. Hindley.

Martini, S. (1987). An interval model for second-order lambda calculus. In *Category theory and computer science (Edinburgh, 1987)*, pages 219–237. Springer, Berlin.

[MR89g:03017]

From the introduction: "In the past few years there has been a lot of interest in the semantics of second-order lambda calculus, and several models have been discovered: the closure model of D. Scott, the finitary retractions model of N. McCracken, the finitary projection model of R. Amadio, K. Bruce and G. Longo, the PER (or "modest") model of Moggi, and the qualitative domains model of J.-Y. Girard. All these models are extensional, i.e. the (η) rule is sound in all of them, so that every function has only one representative in each type.

"In the first-order type-free calculus, on the other hand, no one ever doubted that (η) was a very strong requirement, and much effort was devoted to building nonextensional models in which only a weak extensionality property (expressed by the (ξ) rule) holds. For the first-order simply typed lambda calculus, a categorical characterization of nonextensional models was given by S. Hayashi [Theoret. Comput. Sci. 41 (1985), no. 1, 95–104; MR 87i:18005] (semi (or weak) Cartesian closed categories), and S. Martini has shown (in a preprint) how to obtain a weak Cartesian closed category from the ideal model of D. MacQueen, G. Plotkin and R. Sethi [Inform. and Control 71 (1986), no. 1-2, 95–130; MR 88h:68058], thus obtaining a 'concrete' nonextensional model.

"The main aim of this paper is to determine a nonextensional second-order lambda model by 'generalizing' in a sense, the ideal model." For the entire collection see MR 88j:68002.

Ronchi Della Rocca, S. (1987). An unification semi-algorithm for intersection type schemes. In *TAPSOFT '87, Vol. 1 (Pisa, 1987)*, pages 37–51. Springer, Berlin.

[MR88h:68073] Summary: "We study the unification problem for intersection type schemes. This problem is semidecidable. The semi-algorithm UNIFY for solving it is presented, and it is proved that, when a solution exists, UNIFY finds the most general

one. While UNIFY also uses operations different from substitution, it is conservative with respect to Robinson's unification algorithm. Moreover, a semi-algorithm PP is presented, using UNIFY as an essential tool, which, given a term X that is strongly normalizing, computes its principal type scheme (pts). Since there is a one-to-one correspondence between a term and its pts (if it exists), PP can also be viewed as a reduction machine, using an innermost reduction strategy. The use of unification between type schemes instead of β-reduction in computing the normal form of a term avoids the necessity of α-conversions." For the entire collection see MR 88e:68010.

Seely, R. A. G. (1987). Categorical semantics for higher order polymorphic lambda calculus. *J. Symbolic Logic*, 52(4):969–989.

[MR89g:03019] A categorical structure suitable for interpreting polymorphic lambda calculus is defined. A preliminary announcement of the results of this paper appeared in two earlier papers by the author [C. R. Math. Rep. Acad. Sci. Canada 8 (1986), no. 2, 135–139; MR 88e:03018; ibid. 8 (1986), no. 3, 197–201; MR 88e:03019].

In 1971–1972 J.-Y. Girard introduced the polymorphic lambda calculus, which goes beyond ordinary typed lambda calculus in that type abstraction is permitted. There are two lambda operators, one for ordinary first-order abstraction and a second one for type abstraction [Girard, in Proceedings of the Second Scandinavian Logic Symposium (Oslo, 1970), 63–92, North-Holland, Amsterdam, 1971; MR 53 #12895]. Polymorphic lambda calculus has four sorts of objects, called orders, operators, types and terms. The types make up an order called Omega. Operators are functions from one kind of object to another. Operators from an object A to the order Omega are themselves types, forming another order. One can create product types and function types. Interest in polymorphic lambda calculus increased with the introduction of programming languages with facilitities for defining generic or polymorphic routines in which types can be passed as parameters [Girard, Theoret. Comput. Sci. 45 (1982), no. 2, 159–192; MR 88j:68105].

In 1984 J. C. Reynolds showed that the standard interpretation of first order typed lambda calculus in the category of sets cannot be extended to a model of the second order typed lambda calculus [Reynolds, in Semantics of data types (Valbonne, 1984), 145–156, Lecture Notes in Comput. Sci., 173, Springer, Berlin, 1984; MR 86e:03016].

The semantics of ordinary lambda calculus has a nice formulation using Cartesian closed categories [J. Lambek and P. J. Scott, Introduction to higher order categorical logic, Cambridge Univ. Press, Cambridge, 1986; MR 88a:03032]. The author's intent is to provide a corresponding categorical semantics for the polymorphic lambda calculus using the concept of a PL category (S, G), which by definition consists of a category S with finite products, distinguished object Omega, and exponentiation, and a weakly complete and cocomplete indexed category G over S. The author shows that PL categories provide a sound and complete algebraic semantics for polymorphic lambda calculus. One may regard G as a contravariant functor from S to the category of small categories, functors and natural transformations, effectively giving a categorical structure externally to the object Omega of the category S.

An example of a PL category is N. J. McCracken's model of the second-order polymorphic lambda calculus using closure operators in the power set of the set of natural numbers [McCracken, "An investigation of a programming language with a polymorphic type structure", Ph.D. Thesis, Syracuse Univ., Syracuse, 1979]. This model uses ideas of Donahue and builds on D. S. Scott's model of the ordinary lambda calculus [SIAM J. Comput. 5 (1976), no. 3, 522–587; MR 55 #10262]. See also a paper by K. B. Bruce and A. R. Meyer [in Semantics of data types (Valbonne, 1984), 131–144, Lecture Notes in Comput. Sci., 173, Springer, Berlin, 1984; see MR 85m:68003].

Any PL category has a natural candidate for a weak natural numbers object, the type N of Church numerals. By throwing away nonstandard numerals, one can make a strong natural numbers object. On the technical side, the author modifies the original Girard/Reynolds system by adding expansion (eta) as well as reduction (beta) rules for equality. He presents a definitional extension of polymorphic lambda calculus which includes a notion of subtypes; a construction on the categorical structures furnishes a semantics for the extended language. This construction creates a type of natural numbers, providing a context for Girard's Dialectica interpretation.

Inspired by the author's categorical presentation, but seeking to avoid the semantic problem of equality "up to isomorphism" of category theory, T. Coquand and T. Ehrhard have since proposed a syntactical and equational formalism for the higher-order polymorphic type system [in Category theory and computer science (Edinburgh, 1987), 40–56, Lecture Notes in Comput. Sci., 283, Springer, Berlin, 1987; see MR 88j:68002]. Their treatment of the proof theory of higher-order logic avoids the syntactic problem of bound variables. Reviewed by Johan G. Belinfante.

Wand, M. (1987). A simple algorithm and proof for type inference. *Fund. Inform.*, 10(2):115–121.

[MR89b:03029] The problem of determining the most general type-scheme (if any) assignable to an untyped lambda term was first formulated and solved by Curry. A number of other investigators have given algorithms and proofs. What appears to be new here is a short proof using a loop invariant, which asserts that at each iteration of the procedure, corresponding to the decomposition of the term into subterms, the most general possible type-scheme is generated. Reviewed by B. Lercher.

Ageron, P. (1988). Sémantique catégorique des types: comprendre le système f. *Diagrammes*, 19:PA1–PA40.

[MR90g:18002] The author develops a concept of types in a category $\underline{C}$ which unifies the algebraic and logical theories of types. Applications to data types of informatics and to J.-Y. Girard's system F (an extension of typed λ-calculus) for intuitionistic (propositional) 2nd-order logic are indicated. The usual "type = functor" point of view is replaced by a more expressive framework in the spirit of Ch. Ehresmann's sketches. The basic notion is that of a weft (in French: "trame") of order $\leq n+1$ in the sense of C. Lair [same journal 18 (1987), CL1–CL47; MR 90e:18007]: $\mathcal{T} = (\underline{T}, \mathcal{H}^-(\mathcal{T}), \mathcal{H}^+(\mathcal{T}))$, where $\underline{T}$ is a compograph (i.e. a multiplicative graph in the sense of Ehresmann). The category of all models M of $\mathcal{T}$ in $\underline{C}$ is the full subcategory $\underline{C}^{\mathcal{T}}$ of the category $\mathrm{Func}(\underline{T}, \underline{C})$ of all functors $\underline{T} \to \underline{C}$ with object class $\mathrm{Mod}(\mathcal{T}, \underline{\mathcal{C}})$ given by $M \in \mathrm{Mod}(\mathcal{T}, \underline{\mathcal{C}})$ if and only if for all (F', H, F) in $H^-(\mathcal{T})$ (case 1) or in $H^+(\mathcal{T})$ (case 2) such that $F'\ \underline{T}'_H \to \underline{T}$, $F\ \underline{T}_H \to \underline{T}$, $H\ \mathcal{T}'_{\mathcal{H}} \to \mathcal{T}_{\mathcal{H}}$, and $HF = F'$, one has $F'M \in \mathrm{Mod}(\mathcal{T}'_{\mathcal{H}}, \underline{\mathcal{C}})$, $FM \in \mathrm{Mod}(\mathcal{T}_{\mathcal{H}}, \underline{\mathcal{C}})$, and for each $N \in \mathrm{Mod}(\mathcal{T}_{\mathcal{H}}, \underline{\mathcal{C}})$ such that $HN = F'M$ there is a unique natural transformation $\delta\ N \to FM$ (case 1) [respectively, $\delta\ FM \to N$ (case 2)] such that $H\delta = \mathrm{Id}_{F'M}$. (That is, $F'M$ is the co-universal [respectively, universal] prolongation of FM along H.) (Inductive definition on n.)

Given any syntax of types, i.e., a homomorphism $H\ \mathcal{I} \to \mathcal{J}$ in the category Wef of all wefts, in Section I the author introduces the notion of an inductive [respectively, projective] type in $\underline{C}$ of syntax H as a certain mapping $\tau\ \mathrm{Mod}(\mathcal{I}, \underline{\mathcal{C}}) \to \mathrm{Mod}(\mathcal{J}, \underline{\mathcal{C}})$. Special cases of an inductive [projective] type of syntax H are: a choice of inductive [projective] limits for the functors $\underline{I} \to \underline{C}$; a choice of exponential diagrams for the pairs of objects of $\underline{C}$ (in the projective case); and the structure of a Peano category on $\underline{C}$ (L. Coppey, 1984) (in the inductive case).

For a mixed pair $\Sigma = (\Sigma^-, \Sigma^+)$ of classes of syntaxes of inductive [projective] types the category $\underline{\mathrm{Sem}}\,\Sigma$ of categorical type semantics is introduced. For example, the category Cartc_N of all small Cartesian closed categories with a natural number object can be represented as $\underline{\mathrm{Sem}}\,\Sigma$.

In Section II a new category-theoretical interpretation of Girard's system F is described. In this setting the assertion "a category $\underline{C}$ is a semantics for a type A" can be defined. It is shown that "the $\forall$-type A of system F is semantisable in a Cartesian closed category $\underline{C}$" can be expressed purely equationally. Reviewed by H.-J. Hoehnke.

Breazu-Tannen, V. and Coquand, T. (1988). Extensional models for polymorphism. *Theoret. Comput. Sci.*, 59(1-2):85–114. International Joint Conference on Theory and Practice of Software Development (Pisa, 1987).

[MR90k:68102] Summary: "We present a general method for constructing extensional models for the Girard-Reynolds polymorphic lambda calculus—the polymorphic extensional collapse. The method yields models that satisfy additional, computationally motivated constraints, such as having only two polymorphic Booleans and having only the numerals as polymorphic integers. Moreover, the method can be used to show that any simply-typed lambda model can be fully and faithfully embedded into a model of the polymorphic lambda calculus." For the entire collection see MR 89f:68004.

Coquand, T., Gunter, C., and Winskel, G. (1988). dI-domains as a model of polymorphism. In *Mathematical foundations of programming language semantics (New Orleans, LA, 1987)*, pages 344–363. Springer, Berlin.

[MR90f:68105] Summary: "We investigate a model construction recently described by J.-Y. Girard. This model differs from the models of McCracken, Scott, etc., in that the types are interpreted (quite pleasingly) as domains rather than closures or finitary projections on a universal domain. Our objective in this paper is two-fold. First, we would like to generalize Girard's construction to a larger category called dI-domains which was introduced by Berry. The dI-domains possess many of the virtues of the domains used by Girard. Moreover, the dI-domains are closed under the separated sum and lifting operators from denotational semantics; this is not true of the domains of Girard. We intend to demonstrate that our generalized construction can be used to develop denotational semantics in the ordinary way, but with the added feature of type polymorphism with a 'types as domains' interpretation. Our second objective is to show how Girard's construction (and our generalization) can be done abstractly. We also give a representational description of our own construction using the notion of a prime event structure." For the entire collection see MR 89a:68005.

Girard, J.-Y. (1988). Normal functors, power series and λ-calculus. *Ann. Pure Appl. Logic*, 37(2):129–177.

[MR90e:03015] This paper develops the first of the author's novel approaches to the semantics of various λ-calculi and/or logical calculi. (Some related papers by the author [Theoret. Comput. Sci. 45 (1986), no. 2, 159–192; MR 88j:68105; ibid.

50 (1987), no. 1; MR 89m:03057] were published earlier, but written later, than the paper under review.) In order to model computable functions of higher type one has to find some kind of domains and a decent notion of a space $X \Rightarrow Y$ of functions between such domains X and Y itself forming a domain of this kind. The classical solutions (Scott domains, Ershov f-spaces, etc.) use various notions of continuous functionals to build such spaces. The author's approach—as earlier in the theory of ordinal functions—is to replace topological notions by category-theoretic ones.

For modeling λ-calculi the basic question to be answered is "What is a type?" In this paper two answers are given, leading to two kinds of semantics: quantitative and qualitative. In the quantitative approach a type is viewed as a collection of atomic "states", and an object of a given type as an arbitrary superposition of its atomic states. Formally: for any set A (the atomic "states") the following category SET^A is viewed as a type. The objects of SET^A are formal sums $\sum_{a\in A} x_a a$ with x_a a set for all $a \in A$. The morphisms from $\sum_{a\in A} x_a a$ to $\sum_{a\in A} y_a a$ are formal sums $\sum_{a\in A} f_a a$ with f_a a function from x_a to y_a for all $a \in A$. In the qualitative approach any so-called qualitative domain is viewed as a type. The notion of a qualitative domain—a simplification of the notion of a Scott domain—is introduced in an appendix. This paper focuses on the quantitative semantics, whereas the qualitative semantics is developed in the cited successor to this paper [op. cit., 1986].

In the quantitative approach normal functors are used to build the desired "function spaces". A functor from SET^A to SET^B is termed normal if it preserves direct limits, (possibly infinite) pull-backs and kernels. The main theorem about normal functors is a normal form theorem leading to a sort of "power series expansion". By virtue of this, a normal functor from SET^A to SET^B can be represented by an object of the category $\mathrm{SET}^{\mathrm{Int}(A)B}$ (namely, the family of coefficients of its "power series expansion"). Here $\mathrm{Int}(A)$ denotes the set of all objects $\sum_{a\in A} x_a a$ of SET^A such that x_a is an integer for all $a \in A$.

Choosing $\mathrm{SET}^{\mathrm{Int}(A)B}$ as the "function space" $\mathrm{SET}^A \Rightarrow \mathrm{SET}^B$ is the key to the construction of models of pure (untyped) λ-calculus as well as of Gödel's T. For λ-calculus a set A_∞ is constructed such that $\mathrm{SET}^{A_\infty} \Rightarrow \mathrm{SET}^{A_\infty}$ is isomorphic to SET^{A_∞}. For Gödel's T a set K is defined such that SET^K can serve as the type of integers. The solution is a bit unexpected and is developed by discussing the representation of recursive functions by recursive normal functors.

Further topics treated in the paper are fixed point operators in models of λ-calculus, finiteness properties of the objects in the models denoted by normalizable terms, so-called consistency measures on objects of SET^A, and in a second appendix, last but not least, an extension of the interpretation of Gödel's T to a system closed under forming the sum of any two given types. Reviewed by Peter Pappinghaus.

Martino, A. (1988). A quantitative interpretation of Girard's system F. *Theoret. Comput. Sci.*, 56(3):303–320.

[MR89j:68101] J.-Y. Girard [Ann. Pure Appl. Logic 37 (1988), no. 2, 129–177] developed a new class of models, based on category theory, for various λ-calculi. The present author shows that a variant of one of these is a model of Girard's system F. Such models are of special interest because they cover higher order λ-calculi, or, in other words, illative λ-calculi. Reviewed by Martin W. Bunder.

Mitchell, J. C. (1988). Polymorphic type inference and containment. *Inform. and Comput.*, 76(2-3):211–249.

[MR91c:68021] There have been various attempts to introduce explicit quantifiers into λ-calculus type expressions, in order to model so-called polymorphic objects used in programming languages. Finding consistent models for such type languages has proven to be quite difficult, the main difficulty being essentially the way in which to interpret $\forall$. Does $M\colon \forall t\sigma(t)$ mean that $M\colon s(U)$ for all types u? If so (and this seems fairly reasonable), then it is natural to model $\forall t\sigma(t)$ as the intersection $\bigcap_{u\in D}[\![\sigma(u)]\!]$, where $[\![\sigma(u)]\!]$ is the set corresponding to the type $\sigma(u)$ in some model. But then $\forall tt$ must be the empty set in any model, and this is limiting: for example, there are interesting models in which an appropriate model of $\forall tt$ is $\{\bot\}$, where $\bot$ is the least element of a c.p.o.

This paper presents some formal inference systems for quantified lambda type expressions, describes the semantics of such type expressions, and presents some completeness proofs for the inference system. The inference systems presented are a generalization of the ideal models of D. MacQueen and R. Sethi [in 1982 ACM Symposium on LISP and Functional Programming (Pittsburgh, 1982), 243–252, ACM, New York, 1982]. The paper also establishes some interesting connections between functions that manipulate type expressions (called retyping functions) and type containment (subset). For example, if the type of a lambda expression is "forgotten", its unquantified principal type (called its Curry-type in the paper), will be contained in the original quantified type. Reviewed by M. R. Levy.

Coquand, T., Gunter, C., and Winskel, G. (1989). Domain theoretic models of polymorphism. *Inform. and Comput.*, 81(2):123–167.

[MR91d:68079] Summary: "We give an illustration of a construction useful in producing and describing models of Girard and Reynolds' polymorphic λ-calculus. The key unifying ideas are that of a Grothendieck fibration and the category of continuous sections associated with it, constructions used in indexed category theory; the universal types of the calculus are interpreted as the category of continuous sections of the fibration. As a major example a new model for the polymorphic λ-calculus is presented. In it a type is interpreted as a Scott domain. In fact, understanding universal types of the polymorphic λ-calculus as categories of continuous sections appears to be useful generally. For example, the technique also applies to the finitary projection model of Bruce and Longo , and a recent model of Girard. (Indeed the work here was inspired by Girard's and arose through trying to extend the construction of his model to Scott domains.) It is hoped that by pinpointing a key construction this paper will help towards a deeper understanding of models for the polymorphic λ-calculus and the relations between them."

Curien, P.-L. (1989). Alpha-conversion, conditions on variables and categorical logic. *Studia Logica*, 48(3):319–360.

[MR91k:03153] This is a long and somewhat rambling paper. Underpinning it is the author's experience with his "categorical abstract machine", in which he took the categorical presentation of the simply typed lambda calculus as Cartesian closed categories and used it to furnish a collection of combinators which were well adapted to be used as the abstract machine language for an implementation of a language with the type structure like that of ML. In the present paper the author extends the type system to allow existentially quantified, second-order, and dependent types. He

observes that using a de Bruijn notation for bound variables allows him to present the syntactic rules in a form which is closer to the categorical adjunctions that will be used to model them. Categorical equivalents of the type theories are then presented, together with the appropriate derived combinatory calculi. There is a discussion of the equations satisfied by the combinators, and a fairly sophisticated treatment of the relationship between these and various naturality conditions on the adjunctions. The paper concludes with a discussion of the use of Moggi's ω-sets [cf. J. M. E. Hyland , in The L. E. J. Brouwer Centenary Symposium (Noordwijkerhout, 1981), 165–216, North-Holland, Amsterdam, 1982; MR 84m:03101] in providing a useful concrete model for these type theories. As the preceding sentence illustrates, the reviewer does not feel that the author's historical remarks and attributions are completely reliable. Reviewed by Edmund Robinson.

Hindley, J. R. (1989). BCK-combinators and linear λ-terms have types. *Theoret. Comput. Sci.*, 64(1):97–105.

[MR91c:03009] The linear lambda terms considered in this note are lambda terms in which each variable occurs at most once. A corresponding combinatory calculus is obtained by taking combinators B, C and K as a basis. The primary topic is the set of type schemes (as in papers by the author [Trans. Amer. Math. Soc. 146 (1969), 29–60; MR 40 #7118] and R. Milner [J. Comput. System Sci. 17 (1978), no. 3, 348–375; MR 81g:68022]) which may be assigned to a linear lambda term or BCK term. The main results are (i) every such term has a type, and (ii) types are preserved by weak conversion. Although neither result is completely surprising, the technical details are more complicated (through no fault of the author) than one might initially suspect. Since the note mentions Girard's linear logic [J.-Y. Girard, Theoret. Comput. Sci. 50 (1987), no. 1; MR 89m:03057] in passing, it is worth pointing out that the logic induced by regarding the types of linear terms as propositional formulas (and the terms themselves as proofs) is closed under weakening, while weakening is omitted from Girard's linear logic. Weakening results from the fact that a lambda-bound variable need not occur at all in a linear lambda term, while in Girard's system, each hypothesis must be used exactly once, unless marked with the modal operator '!', which is not considered here. Reviewed by John C. Mitchell.

Jäger, G. (1989). Type theory and explicit mathematics. In *Logic Colloquium '87 (Granada, 1987)*, pages 117–135. North-Holland, Amsterdam.

[MR90k:03013] This is a very interesting and stimulating survey paper which shows that explicit mathematics can be a useful and flexible framework for discussing concepts of present-day computer science. In particular the author proposes a variant of Feferman's explicit mathematics and shows how monomorphic- and polymorphic-type structures can be interpreted in this formalism. Moreover, the extension of a given explicit type theory by an abstract data type is formulated in general and exemplified in the case of the natural numbers. Lastly, computable types and universes are discussed. For the entire collection see MR 90d:03004. Reviewed by Mariangiola Dezani-Ciancaglini.

Pitts, A. M. and Taylor, P. (1989). A note on Russell's paradox in locally Cartesian closed categories. *Studia Logica*, 48(3):377–387.

[**MR91k:03018**] The aim of this paper is to show the inconsistency of a type theory with dependent products, strong equality types, and a universal type. Type theories may be "inconsistent" in two possible ways. The weak form of inconsistency is for each type to possess a closed term (which is bad only for proof theorists, since it corresponds to each proposition having a proof), while the stronger form has additionally that any two terms of the same type are provably equal. When the calculus has types which internalise the notion of equality, as is the case with the type theories studied here, these two notions coincide. It was known previously that a type theory with dependent products and a universal type was inconsistent in the weak sense. This was proved by constructing a "looping combinator", and it is interesting to read the present paper in conjunction with one by D. J. Howe [in Proceedings of the Symposium on Logic in Computer Science (Ithaca, NY, 1987), 205–214, IEEE Comput. Soc. Press, Washington, DC, 1987; CCA 1987:58946], in which this proof is discussed. The present paper shows that in the presence of equality types it is possible to formalise a variant of Russell's paradox, and to obtain a fixed point combinator rather than a looping one (it is not known whether a similar argument is possible without the equality types). This is obviously highly significant for those interested in interpreting languages with recursion. The paper also discusses the categorical interpretation of the result. Reviewed by Edmund Robinson.

Rozière, P. (1989). Un résultat de non-typabilité dans Fω. *C. R. Acad. Sci. Paris Sér. I Math.*, 309(13):799–802.

[**MR91c:03010**] Summary (translated from the French): "A very simple term of the pure λ-calculus, found by B. Maurey, computes, for two Church integers m and n, the Boolean value of the relation '$m \leq n$' in a 'computation time' (β-reduction number, for example by reduction of the head redex) linear in $\inf(m, n)$. J.-L. Krivine [same journal Ser. I Math. 304 (1987), no. 5, 123–126; MR 88f:03013] has shown that, if m and n are considered to be of integer type, this term is not typable in the second-order typed λ-calculus, or system F, but rather is typable in an extension of this system obtained via the addition of recursive types.

"We show here that, in the same context, this term is also not typable in the finite-order typed λ-calculus, or system Fω, another extension of the system F also introduced by J.-Y. Girard. One will get the same result in the system of constructions of T. Coquand and G. Huet [Inform. and Comput. 76 (1988), no. 2, 95–120; MR 89j:68096], which types the same terms as Fω.

"It has been shown [P. Giannini and S. Ronchi, "Characterisation of typing in polymorphic type discipline", LICS, Edinburg, 1988; per bibl.] that the system Fω types strictly more closed terms than the system F."

Scedrov, A. (1989). Normalization revisited. In *Categories in computer science and logic (Boulder, CO, 1987)*, pages 357–369. Amer. Math. Soc., Providence, RI.

[**MR90g:03014**] The normalization theorem for second-order polymorphic λ-calculus (typed λ-calculus with the operations of abstraction and application over types) is a very important theorem, which implies in an elementary way the normalization of analysis. It was proved for the first time by J. Y. Girard in his thesis ["Interpretation fonctionelle et elimination des coupeurs de l'arithmetique d'ordre superieur", These d'Etat, Univ. Paris VII, Paris, 1972], where the proof was given for the system F, which can be seen as an extension of the second-order polymorphic λ-calculus.

The proof presented in this paper, which is remarkably simple, uses a variant of Girard's technique which has been developed by the interaction of a number of researchers in the field (including the author). It is based on the idea of giving an interpretation of polymorphic types as subsets of the set of normalizable untyped λ-terms. The proof presented in the paper shows only weak normalization, but it can be adapted to prove strong normalization as well.

In the last section the author points out that the domain of this interpretation (namely, a collection of subsets of the normalizable terms modulo β-conversion) can be seen as a sort of weak Cartesian closed category with weak indexed products. For the entire collection see MR 89m:18001. Reviewed by M. Coppo.

Solitro, U. (1989). A typed calculus based on a fragment of linear logic. *Theoret. Comput. Sci.*, 68(3):333–342.

[**MR91c:03011**] In this interesting article the author gives a functional language (the ν-calculus) which interprets the second-order multiplicative fragment of J.-Y. Girard's linear logic (LL) [same journal 50 (1987), no. 1; MR 89m:03057]. More specifically, the author proves there is a bijective correspondence between terms of the ν-calculus and Girard's multiplicative proof nets [op. cit.]. The author then sets up an appropriate notion of conversion (rewrite rules). Strong normalization and Church-Rosser properties for the ν-calculus are then inferred from the general theory of proof-nets.

As in any typed functional language, the ν-calculus has variables of every type, in particular "co-variables" $y : A^\perp$. The key operation of the ν-calculus is the formation of the term $\nu xy.t$ where $x : A$ and $y : A^\perp$ are simultaneously bound in t. The author claims that his functional view of the normalization process clarifies some of Girard's use of "information flow" in proof nets. The author's arguments to this effect are not detailed enough to be convincing.

There is increasing interest among many researchers in using typed functional calculi to study (fragments of) LL. We may expect to see more languages in the style of the author's suggestive study. Such languages would be useful in studying so-called "coherence problems" for categorical models of LL, in extensions of Lambek's noncommutative linear logics arising from linguistics [cf. J. Lambek, in Categories in computer science and logic (Boulder, CO, 1987), 217–239, Contemp. Math., 92, Amer. Math. Soc., Providence, RI, 1989; MR 90e:18010].

In a related vein, a calculus of linked pairs of co- and contravariant variables arises in recent categorical studies of polymorphism [cf. E. S. Bainbridge, P. J. Freyd, A. Scedrov and P. J. Scott, Theoret. Comput. Sci. 70 (1990), no. 1, 35–64] where the notion of dinatural transformation plays a fundamental role (cf. early categorical studies of such variance problems in papers by G. M. Kelly [in Coherence in categories, 66–105, Lecture Notes in Math., 281, Springer, Berlin, 1972; MR 49 #5126] and S. Eilenberg and Kelly [J. Algebra 3 (1966), 366–375; MR 32 #7618]). Reviewed by Philip J. Scott.

Abadi, M. and Plotkin, G. D. (1990). A per model of polymorphism and recursive types. In *Fifth Annual IEEE Symposium on Logic in Computer Science (Philadelphia, PA, 1990)*, pages 355–365. IEEE Comput. Soc. Press, Los Alamitos, CA.

[MR1099188] This item will not be reviewed individually. For details of the collection in which this item appears see 92b:68007.

Bruce, K. B., Meyer, A. R., and Mitchell, J. C. (1990). The semantics of second-order lambda calculus. *Inform. and Comput.*, 85(1):76–134.

[MR92e:68110] Summary: "In the second-order (polymorphic) typed lambda calculus, lambda abstraction over type variables leads to terms denoting polymorphic functions. Straightforward cardinality considerations show that a naive set-theoretic interpretation of the calculus is impossible. We give two definitions of semantic models for this language and prove them equivalent. Our syntactical 'environment model' definition and a more algebraic 'combinatory model' definition for the polymorphic calculus correspond to analogous model definitions for untyped lambda calculus. Soundness and completeness theorems are proved using the environment model definition. We verify that some specific interpretations of the calculus proposed in the literature indeed yield models in our sense."

Curien, P.-L. and Ghelli, G. (1990). Coherence of subsumption. In *CAAP '90 (Copenhagen, 1990)*, pages 132–146. Springer, Berlin.

[MR91g:03034] Summary: "Subsumption used in subtyping breaks the terms-as-proofs paradigm. Semantics most naturally are associated with proofs. Thus a problem of coherence arises: different typing proofs of the same term must have related meanings. We propose a proof-theoretical, rewriting approach to this problem. We focus on $\mathbf{F}_{\leq}$, a second-order lambda calculus with bounded quantification, which is rich enough to make the problem interesting. We define a normalizing rewriting system on the proofs, in which different proofs of the same typing judgement are transformed into a unique normal proof, and in which normal proofs of judgements assigning different types to the same term are strongly related. This rewriting system is not defined on the proofs themselves, but on the terms of an auxiliary type system, in which the terms carry complete information about their typing proof. This technique also gives a simple proof of the existence of a minimum type for each term. From an analysis of the proofs in normal form we obtain a deterministic type-checking algorithm, which is sound and complete by construction." For the entire collection see MR 91e:68005.

Dezani-Ciancaglini, M. and Venneri, B. (1990). Partial types and intervals. *SIAM J. Comput.*, 19(4):644–658.

[MR91g:03035] This paper studies an inference system for assigning types to untyped lambda terms. The motivation for this work stems from a polymorphic type system related to the programming language ML, as studied by D. MacQueen, G. Plotkin and R. Sethi [Inform. and Control 71 (1986), no. 1-2, 95–130; MR 88h:68058]. (This paper will be cited as [MPS].) Although the syntactic type system of [MPS] is straightforward, the semantics has some technical complications. Types are interpreted semantically as "ideals" (downward closed sets closed under meets with

arbitrary elements) over a partially-ordered model of the untyped lambda calculus. This choice does not originate with [MPS] and is well motivated by certain properties of types. However, taking types as ideals, it is not possible to interpret recursively defined types as least fixed points of monotonic functions, since type expressions do not generally define monotonic functions. The alternative used in [MPS] is to define a metric with the property that each type expression defines a contractive function of each free type variable. This solves the problem of giving a satisfactory interpretation to each type expression. However, although the metric seems natural for the lambda model chosen in [MPS], the technique does not generalize to other lambda models. In subsequent work by R. Cartwright [in Twelfth Annual ACM Symposium on Principles of Programming Languages (New Orleans, LA, 1985), 22–36, ACM, New York, 1985], the second precursor to the paper under review, a generalization of the [MPS] type assignment system is considered, and types are interpreted as intervals of ideals, rather than single ideals. This has the desired effect of making type expressions monotonic functions of their arguments, but at a cost to the syntactic type discipline: the standard type assignment rules are no longer sound. This brings us to the present paper.

This paper addresses the shortcomings of Cartwright's work by developing a type assignment system with polymorphic and recursive types that is sound and complete for a general form of interval type interpretation developed in the paper. The main contributions are the type assignment system itself, the general semantic setting, and the soundness and completeness theorems. Since intervals are sets of ideals, the type assignment system involves two forms of membership. The value of a lambda term may belong to all ideals within an interval, or only some. These are treated as two modalities of type membership. The authors prove a subject reduction theorem, an important property showing that the semantics respects reduction, or "program execution". In the process of presenting the soundness and completeness theorems, some interesting properties of the type assignment and interval interpretation are revealed. The paper also discusses the relationship to more standard type assignment systems and suggests some similarities with type assignment based on subtyping. Reviewed by John C. Mitchell.

Feferman, S. (1990). Polymorphic typed lambda-calculi in a type-free axiomatic framework. In *Logic and computation (Pittsburgh, PA, 1987)*, pages 101–136. Amer. Math. Soc., Providence, RI.
[MR92c:03014]

The general purpose of the paper is to provide a unitary conceptual axiomatic framework for representing programs, taken from functional programming languages based on untyped or polymorphic λ-calculi, and for stating and proving properties of programs. The author introduces two axiomatic theories: IOC_Λ and EOC_Λ ($\subseteq \mathrm{IOC}_\Lambda$), which are respectively the impredicative and the predicative (or elementary) versions of a "theory of operations and classes". Both incorporate the λ-calculus on partial terms, abstraction over types, and various typed λ-calculi (more precisely, the realisability version of these calculi). While EOC_Λ is a conservative extension of Peano arithmetic, IOC_Λ contains, via realisability, current forms of polymorphic λ-calculi (essentially Girard's system F), and IOC_Λ is as strong as analysis (= second-order Peano arithmetic). Both systems are strongly related to systems worked out previously by the author.

Concerning the expressive power of the theories: EOC_Λ permits finitary inductive definitions of types, polymorphic sorting of lists, and records (all this is possible at the elementary level since the type N of integers is taken here as a basic type). On the other hand IOC_Λ permits infinitary inductive definitions of types, higher ordinal number classes and recursion on them, subtyping and bounded products.

The paper is directed towards two applications. The first is to show that the author's systems are much more expressive than system F and permit in particular new type constructions; some comments on this point will be made below. The second is to give evidence that, since questions of predicativity and impredicativity do not arise in computational practice, the uses of system F in practice can be accounted for on elementary grounds: a large class of "semipredicative" typed terms is isolated with the property that (roughly speaking) if t is of type T in F then one can prove in EOC_Λ that t° realises T, where t° is the term of pure λ-calculus obtained by erasing all types in t (t° represents the computational behaviour of t). Finally, the author conjectures, on the basis of some test examples, that only semipredicative terms are met in practice.

Let us come now to the comparison between IOC_Λ and system F as to their typing possibilities. It is clear that the expressive power of IOC_Λ, which is that of second-order arithmetic (or second-order combinatory logic), is far greater than that of F, which is that of second-order propositional calculus. So, to be meaningful, the comparison has to be made, not with system F, but with its second-order arithmetic extension AF_2, which was introduced independently by D. Leivant and J.-L. Krivine and was developed by Krivine in Chapter 9 of his book λ-calcul, types et modeles [Masson, Paris, 1990; per revr.; see also Krivine and M. Parigot, J. Inform. and Process. Cybernet. 26 (1990), no. 3, 149–167; MR 91j:68086; Leivant, in Twenty-Fourth Annual Symposium on Foundations of Computer Science (Tucson, AZ, 1983), see p. 201, IEEE, New York, 1984].

As regards polymorphic typing, both systems have the same power. For example, both can express the types $\mathrm{LIN}\, X$ of linear orderings on X, and $S = \forall X(\mathrm{LIN}\, X \to \forall n \in N(X^n \to X^n))$, which is realised by any uniform way of sorting sequences $x \in X^n$ with respect to any linear ordering of X. Now S is the example given by the author of a type not available in F. This is obviously true but not very convincing, since S is indeed a very weak specification which is in fact realised by any uniform way of giving a result in X^n, for example by any term which, given any $x \in X^n$, calculates $(\min x, \cdots, \min x) \in X^n$, where $\min x$ is the smallest element of x; in that respect there are types of F which are as good, or as bad, as S: for example, the type $S' = \forall X((X^2 \to B) \to (\mathrm{LIST}\, X \to \mathrm{LIST}\, X))$, where B is the type of Booleans and $\mathrm{LIST}\, X$ the type of finite lists of elements of X (nevertheless IOC_Λ, as well as AF_2, allows us to link programs to their specifications much more precisely). For the entire collection see MR 91a:03002. Reviewed by Chantal Berline.

Kfoury, A. J. and Tiuryn, J. (1990). Type reconstruction in finite-rank fragments of the polymorphic λ-calculus (extended abstract). In *Fifth Annual IEEE Symposium on Logic in Computer Science (Philadelphia, PA, 1990)*, pages 2–11. IEEE Comput. Soc. Press, Los Alamitos, CA. **[MR1099155]** This item will not be reviewed individually. For details of the collection in which this item appears see 92b:68007.

Krivine, J.-L. (1990). *Lambda-calcul. Types et modèles. [Types and models]*. Masson, Paris.

[**MR93i:03019**] This book collects some classical and some new topics in lambda-calculus theory. The presentation is fluent while preserving the mathematical rigour.

There are essentially two original contributions: (1) Intersection types are used as a new and elegant tool for proving classical properties of pure lambda-calculus. (2) A type assignment system for pure lambda-calculus whose types are formulas of second-order predicate calculus is introduced. The resulting language is suitable both for representing data structures and for writing programs (in the "proofs as programs" approach).

There is only one point on which the reviewer disagrees. The author says that the intersection type assignment system does not seem useful for modeling programming languages. On one hand, intersection types have been used by J. C. Reynolds ["Preliminary design of the programming language FORSYTHE", Rep. No. CMU-CS-88-159, Carnegie Mellon Univ., Pittsburgh, PA, 1988; per revr.; in Automata, languages and programming (Stresa, 1989), 704–722, Lecture Notes in Comput. Sci., 372, Springer, Berlin, 1989; MR 91a:68030] to get expressive power for his programming language FORSYTHE. The technique described in a paper by B. Pierce ["Programming with intersection types and bounded polymorphism", Ph.D. Thesis/Rep. No. CMU-CS-91-205, Carnegie Mellon Univ., Pittsburgh, PA, 1991; per revr.] improves the typing of FORSYTHE and gives interesting examples. On the other hand, taking as meanings of lambda-terms the set of types which can be assigned to them is a way of building most classical lambda-calculus models. More precisely, by suitable type inclusions and by adding a subtyping rule one obtains all models built up using Scott's inverse limit construction [see F. Cardone and M. Coppo, "Two extensions of Curry's type inference system", in Logic and computer science, edited by P. Odifreddi, Academic Press, New York, 1990; per revr.; and references therein] and the models of G. Plotkin [in Theoretical aspects of computer software (Sendai, 1991), 1–17, Lecture Notes in Comput. Sci., 526, Springer, Berlin, 1991; MR 93a:68086] and Engeler. This approach could have been used by the author to describe lambda-calculus models.

In any case it is really a pleasure to read this book. Also, it has been fruitfully used in postgraduate courses. Reviewed by Mariangiola Dezani-Ciancaglini.

Malecki, S. (1990). Generic terms having no polymorphic types. In *Automata, languages and programming (Coventry, 1990)*, pages 46–59. Springer, New York.

[**MR91h:68118**] Summary: "We define a set S_2 of lambda-terms for which we decide typing in Girard's system F. As the algorithm uses normalizability in an essential way, we also characterize normalizable terms in S_2. This allows us to connect the two properties and to build many nontypable terms, normalizable and otherwise." For the entire collection see MR 91e:68004.

Normann, D. (1990). Formalizing the notion of total information. In *Mathematical logic*, pages 67–94. Plenum, New York.

[**MR92f:68104**] From the introduction: "Suppose that Λ is a model for untyped λ-calculus and that T is some typed λ-calculus (transfinite, second order or whatever). One standard procedure for obtaining a model for T is to regard all terms in T as untyped terms, interpret each type as a subset of Λ and then prove that each typed term is interpreted into the interpretation of the type.

"When the types are inductively defined it is possible to isolate the total objects of the type, while if T is second-order type theory this is not obvious. If T permits

certain recursive definitions of types, the isolation of the total or well-founded objects becomes more vital. As mentioned, it is possible by straightforward induction on the type-formation to isolate the total (or well-founded) objects of the type, and the proofs that λ-terms define total objects are often simple though the result is essential for the applicability of the theory in question. It does not, however, involve a systematic analysis for the concept of totality itself.

"In this paper we focus on the concept of totality in defining the structures we investigate. We view a totality-domain as a set of total processes giving definite answers to a fixed class of questions. We isolate certain properties the class of total objects might have, and we discuss which properties of the 'processes' these reflect." For the entire collection see MR 91h:03007.

Scedrov, A. (1990). A guide to polymorphic types. In *Logic and computer science (Montecatini Terme, 1988)*, pages 111–150. Springer, Berlin.
[MR92c:03035] This is a survey of and introduction to polymorphic lambda calculi. The author concentrates on two important examples, the second-order lambda calculus of Girard and Reynolds, and the Coquand-Huet calculus of constructions. There are detailed proofs of the Church-Rosser property and strong normalisation for the second-order lambda calculus, and an extensive discussion of models for both calculi. There are copious references and the style is clear and concise. The major drawback of the paper is, however, its age. It was written for a summer school held in Italy in the summer of 1988. Since then, the area with which it deals has developed rapidly (a fact for which the author himself is partly responsible), and this, while not affecting its value as a tutorial, has rendered the paper somewhat out-of-date as an exposition of the current research situation. A more accurate picture can now be obtained from the proceedings of the annual "logic in computer science" conferences [IEEE, Computer Soc. Press]. For the entire collection see MR 91e:03006. Reviewed by Edmund Robinson.

Tiuryn, J. (1990). Type inference problems: a survey. In *Mathematical foundations of computer science (Banská Bystrica, 1990)*, pages 105–120. Springer, Berlin.
[MR92c:68131] From the introduction: "The type inference problem can be stated as follows. Given an expression M without any types in it, decide whether types can be inserted into M so that it becomes a correctly typed program. The strong type inference problem differs from the above problem in allowing some of the global identifiers of M to have fixed, predefined types. The answer to these problems generally depends on the available types, available programming constructs and the rules for typing them. In the present paper we survey the area of type inference. This area has been a domain of very intensive research for the last five years. In our survey, in addition to old results we try to present some very recent ones.

"The paper is divided into five sections. Section 2 is devoted to strong type inference for the finitely-typed calculus. Section 3 deals with ML, an extension of the calculus of finite types. Section 4 contains some recent results on an extension of ML whose essence is polymorphic recursion. Finally, Section 5 discusses (strong) type inference for the second-order lambda calculus and some of its fragments." For the entire collection see MR 91h:68011.

Zaionc, M. (1990). A characterization of lambda definable tree operations. *Inform. and Comput.*, 89(1):35–46.

[MR91m:03009] The author characterizes the operations on binary branching trees that are lambda definable when the collection of trees is represented in Church's simple theory of types as the type $(O \to (O \to O)) \to (O \to O)$. The main result is a proof that they constitute the least set containing the constant functions and projections and closed under composition and a strange, limited form of recursion. The author contrasts his result with previously known results for operations on integers and strings and asks whether these can be put together in a common formulation for operations on general free algebraic structures. Readers may therefore be interested in the papers by D. Leivant [in Special Issue: Twenty-fourth Annual Symposium on the Foundations of Computer Science (Tucson, AZ, 1983), 201, IEEE, New York, 1984] and C. Böhm and A. Berarducci [Theoret. Comput. Sci. 39 (1985), no. 2-3, 135–154; MR 87h:68012] which contain the solution of this general problem in the more extended context of the second-order lambda calculus. Reviewed by Edmund Robinson.

Alessi, F. and Barbanera, F. (1991). Strong conjunction and intersection types. In *Mathematical foundations of computer science, 1991 (Kazimierz Dolny, 1991)*, pages 64–73. Springer, Berlin.

[MR1135350] This item will not be reviewed individually. For details of the collection in which this item appears see 92g:68008.

Barbanera, F. and Dezani-Ciancaglini, M. (1991). Intersection and union types. In *Theoretical aspects of computer software (Sendai, 1991)*, pages 651–674. Springer, Berlin.
[MR93g:68071]

From the introduction: "Intersection types were introduced by M. Coppo, Dezani-Ciancaglini and B. M. Venneri [Z. Math. Logik Grundlag. Math. 27 (1981), no. 1, 45–58; MR 82j:03011] as a generalization of the predicate type discipline by Curry, mainly with the aim of describing the functional behaviour of all solvable λ-terms. In the intersection type discipline the usual '$\to$'-based type language for λ-calculus was extended by adding a constant ω as a universal type and a new connective '$\wedge$' for the intersection of two types. With suitable axioms and rules to assign types to λ-terms, this resulted in a system in which: (i) the set of types given to a λ-term does not change under β-conversion, and (ii) the sets of solvable and normalizing λ-terms can be characterized very neatly by the types of their members. Moreover, this system was proved to be sound and complete.

"The aim of the present paper is foundational: to study the intersection type discipline enriched with a new type constructor '$\vee$' allowing us to form union types. The typing rules for introduction and elimination of '$\vee$' are essentially inspired by the 'formulas-as-types' analogy, viewing $\vee$ as the logical connective 'or'. New clauses, which take into account the intuitive meaning of '$\vee$' as union, are added to the preorder relation on types. Moreover, we add a clause which is not implied by set-theoretic arguments, but which is useful to preserve typing under reductions of λ-terms. The pre-order relation so obtained is used in a type assignment rule.

"We prove that our type assignment system with intersection and union types still satisfies the properties (i) and (ii). These proofs are done by associating to each type a finite set of intersection types, and to each deduction in the full system a set of deductions in the intersection type assignment system. Moreover, we prove the approximation theorem, which says that the types of a λ-term are all and only the types of its approximants. "A natural way of defining the semantics of a predicate type theory is to interpret the underlying untyped language in a domain D and the types as subsets of D, in such a way that the interpretation of a λ-term is contained in the interpretation of every type which can be assigned to that λ-term. A classical way of giving the semantics of disjunction is to consider possible worlds where knowledge can expand, i.e. where there are different states of knowledge. Of course, there are incompatible ways of extending the knowledge: a tree of possibilities will be a possible world, and a branch in such a tree will determine a possible history of knowledge in that possible world. These are the basic ideas leading to Beth-Kripke models for intuitionistic logic. We introduce a notion of type interpretation inspired by Beth models. Essentially, we make sure that whenever a λ-term M has a union type $\sigma \vee \tau$ in a given state of knowledge w, then M will eventually have one of σ and τ in every state extending w. We prove the soundness and completeness of our type assignment system with respect to this notion of type interpretation. Moreover, we consider two other notions of type interpretation: one inspired by the second order interpreation given in logic to the disjunction, and the other inspired by the translation into types without unions. Our type assignment system is proved to be sound and complete with respect to both these type interpretations.

"As by-products of these results we have also interesting properties for the type assignment systems obtained by weakening the pre-order relation on types. In fact all these systems characterize the sets of solvable and normalizing λ-terms and are sound with respect to the four notions of type interpretations introduced in the present paper." For the entire collection see MR 92h:68009.

Barendregt, H. (1991). Introduction to generalized type systems. *J. Funct. Programming*, 1(2):125–154.

[MR93e:03019] This paper constitutes an introduction to the author's programme for unifying many of the typed lambda calculi currently being used in theoretical computer science and proposed as foundations for computational systems (perhaps more theorem-provers than programming languages). For some time these systems have been classified in a more or less informal way by the strength of the type constructions they allow. The author has to some extent formalised this. In this paper he concentrates on the general product types (and β- as opposed to $\beta\eta$-equivalence). He introduces a generic calculus, and classifies the calculi by some simple restrictions on the form of the general product introduction rules. The result is a lattice of calculi arranged in the form of a 3-dimensional cube (the "λ-cube"). The advantage of this approach is that it allows uniform proofs of various metatheoretic results (the Church-Rosser theorem, strong normalisation, subject reduction, unicity of types), although the proofs of these are given elsewhere [H. P. Barendregt, "Lambda calculi with types", in Handbook of logic in computer science, edited by S. Abramsky, T. Maibaum and D. Gabbai, Oxford Univ. Press; per bibl.]. The main thrust of this particular paper is an exploration of the propositions-as-types paradigm, with logical systems being introduced to correspond to the type systems in the λ-cube. Reviewed by Edmund Robinson.

Cardelli, L., Martini, S., Mitchell, J. C., and Scedrov, A. (1991). An extension of system F with subtyping. In *Theoretical aspects of computer software (Sendai, 1991)*, pages 750–770. Springer, Berlin.

[MR92k:03010] Summary: "System F is a well-known typed λ-calculus with polymorphic types, which provides a basis for polymorphic programming languages. We study an extension of F, called $F_{<:}$, that combines parametric polymorphism with subtyping.

"The main focus of the paper is the equational theory of $F_{<:}$, which is related to PER models and the notion of parametricity. We study some categorical properties of the theory when restricted to closed terms, including interesting categorical isomorphisms. We also investigate proof-theoretical properties, such as the conservativity of typing judgments with respect to F.

"We demonstrate by a set of examples how a range of constructs may be encoded in $F_{<:}$. These include record operations and subtyping hierarchies that are related to features of object-oriented languages."

For the entire collection see MR 92h:68009.

Cardone, F. and Coppo, M. (1991). Type inference with recursive types: syntax and semantics. *Inform. and Comput.*, 92(1):48–80.

[MR92f:03012] In the article under review, the authors study Curry's type system extended with a recursion operation working on all types, not just on those where the relevant type variable occurs positively. This makes the semantics a delicate matter.

The semantics of these types is given in terms of ideals in specific c.p.o.'s, which come equipped with a notion of approximation (like that for D. Scott's original D_∞ construction). This approach goes back to a paper of Coppo [in Automata, languages and programming (Nafplion, 1985), 120–129, Lecture Notes in Comput. Sci., 194, Springer, Berlin, 1985; see MR 87a:68002]. The interpretation of types is such that two types have the same value if and only if they have the same infinite unfolding.

The authors further succeed in extending the type assignment system in such a way that a completeness result can be obtained. The approximation rule they add yields a type for a term in case all direct approximants of the term have that type. It is then shown that in the resulting system a term has a type if and only if the value of the term is an element of the value of the type. Reviewed by Bart Jacobs.

Coquand, T. (1991a). An algorithm for testing conversion in type theory. In *Logical frameworks (Sophia-Antipolis, 1990)*, pages 255–279. Cambridge Univ. Press, Cambridge.

[MR1139787] This item will not be reviewed individually. For details of the collection in which this item appears see 92f:68003.

Coquand, T. (1991b). An introduction to type theory. In *Logique et informatique: une introduction*, pages 117–135. INRIA, Rocquencourt.

[MR1146963] This item will not be reviewed individually. For details of the collection in which this item appears see 93d:68003.

Dybjer, P. (1991). Inductive sets and families in Martin-Löf's type theory and their set-theoretic semantics. In *Logical frameworks (Sophia-Antipolis, 1990)*, pages 280–306. Cambridge Univ. Press, Cambridge.

[**MR1139788**] This item will not be reviewed individually. For details of the collection in which this item appears see 92f:68003.

Geuvers, H. and Nederhof, M.-J. (1991). Modular proof of strong normalization for the calculus of constructions. *J. Funct. Programming*, 1(2):155–189.

[**MR92k:03009**] Two fundamental properties of many rewriting theories are strong normalization, SN, and Church-Rosser, CR. SN says that all reductions starting from any given term t eventually halt at an irreducible term (= normal form). CR (= the diamond property or confluence) implies that irreducibles are unique. A typed functional language (qua rewrite system) which satisfies both SN and CR admits an intrinsic operational semantics: to compute the value of a function f applied to an argument a, simply reduce $f(a)$ to normal form.

The authors provide a careful and detailed new proof of SN for Coquand's calculus of constructions, λC. This type theory, one of the most subtle known, includes Girard's systems $\mathcal{F}$ and $\mathcal{F}^\omega$ as well as certain dependent type systems a la Martin-Löf. The authors' argument is based on a reduction-preserving mapping from λC to $\mathcal{F}^\omega$, followed by a careful proof of SN for the various kinds of abstraction thus coded into $\mathcal{F}^\omega$. Along the way, the authors eventually prove SN for $\mathcal{F}^\omega$ itself by the Tait et al. method of using saturated sets of strongly normalizing untyped lambda terms, rather than by Girard's original method of candidates.

Reviewer's remarks: The proof is very technical, but with all details clearly spelled out. The paper includes a very good, although terse, introduction to the area of generalized type systems, and is an excellent primer on sophisticated techniques for normalization proofs. Also, as is typical of this genre of proof, the authors assume that the underlying collection of pseudoterms satisfies CR; for this latter, one refers back to the untyped lambda calculus again. From a categorical viewpoint, what is missing in all the heavy syntactic machinery here is an account of appropriate product types with surjective pairing (at the simply typed as well as higher levels). Presumably this would entail an extension of one of the known techniques for handling products for simple types. Nonetheless, a proof of SN for a categorical-style theory of calculus of constructions as in a paper by J. M. E. Hyland and A. M. Pitts [in Categories in computer science and logic (Boulder, CO, 1987), 137–199, Contemp. Math., 92, Amer. Math. Soc., Providence, RI, 1989; MR 90h:03023] must still await some intrepid explorer. Reviewed by Philip J. Scott.

Giannini, P. and Ronchi Della Rocca, S. (1991). Type inference in polymorphic type discipline. In *Theoretical aspects of computer software (Sendai, 1991)*, pages 18–37. Springer, Berlin.

[**MR92i:03013**] Summary: "A hierarchy of type assignment systems is defined which is a complete stratification of the polymorphic type assignment system. For each of these a type inference algorithm is given." For the entire collection see MR 92h:68009.

Hirokawa, S. (1991a). Principal type assignment to lambda terms. *Internat. J. Found. Comput. Sci.*, 2(2):149–162.

[**MR93a:03013**] The paper is concerned with giving simple types to untyped lambda terms. It contains an algorithm for transforming any type of a term into the principal, or simplest possible type. This is not the optimal algorithm for producing the principal type, since it requires $O(k^4m^4)$, as opposed to linear, time. However, the algorithm gives some insight into the relationships between different types for the same term. Reviewed by John C. Mitchell.

Hirokawa, S. (1991b). Principal type-schemes of BCI-lambda-terms. In *Theoretical aspects of computer software (Sendai, 1991)*, pages 633–650. Springer, Berlin.
[MR92m:03017] The author gives a structural characterization of the types that are principal type schemes (PTSs) of a linear λ-term. A similar characterization is also given for minimal type schemes of linear λ-terms. (The reviewer is not aware of similar characterizations of PTSs for wider classes of λ-terms.)

The paper is rather technical and gives most of the proofs (though pointers to unpublished work may frustrate some readers). On the other hand, no mention is made of possible applications of these results, which may leave the impression of a rather sterile piece of mathematics. For the entire collection see MR 92h:68009. Reviewed by Eugenio Moggi.

Koletsos, G. (1991). Polymorphic lambda calculus: the Church-Rosser property. *Theoret. Comput. Sci.*, 79(2, (Part B)):365–371.
[MR92h:03019] There are two important properties a calculus of terms (and types) may have: Church-Rosser (CR) and strong normalization (SN). The first says that given a term t which reduces both to t' and to $t"$, there is a "common reduct" $t"'$ to which both t' and $t"$ reduce; the second says that there are no infinite reduction paths. Girard proved both these results for his system F (the second-order polymorphic lambda-calculus). His proof of SN is semantical in the sense that terms are interpreted as elements of appropriate sets—using "candidats de reducibilite", which extends a method of W. Tait . Subsequently both J. Mitchell and A. Scedrov gave a somewhat simpler proof using an erasure function which removes types. Girard's proof of CR for system F is syntactical and extends the Tait and Martin-Löf proof for the untyped lambda calculus. The paper under review contains an alternative proof of CR: it is semantical and uses so-called "candidats de monovalence". As remarked by the author, it is open whether there is also an erasure proof for CR. Reviewed by Bart Jacobs.

Longo, G. and Moggi, E. (1991). Constructive natural deduction and its "ω-set" interpretation. *Math. Structures Comput. Sci.*, 1(2):215–254.
[MR92j:03004] Surely the definition of models of Girard's system F by internal categories, originally due to Moggi, is one of the most interesting results in the field of mathematical semantics. The main problem in modelling Girard's system is the semantic understanding of impredicativity. Moggi's semantics allows a "set-theoretic" understanding of a calculus which has no classical set-theoretic model. The present paper provides an elementary but technically neat development of Moggi's semantics using modest sets and ω-sets. This is essentially the content of Sections 6–8. The authors suggest that the experienced reader may skip the previous sections, but the

reviewer believes that they should be read. In fact, the introduction contains a very interesting discussion on the role of mathematical investigation of linguistic concepts in computer science. The remaining sections introduce the most interesting type theories by stressing the analogy "propositions-as-types" and contain deep remarks. Moreover there are many historical remarks and philosophical reflections. The style is cultured and pleasant. In conclusion, this paper is outstanding for its readability and deep mathematical content. Reviewed by Mariangiola Dezani-Ciancaglini.

Mendler, N. P. (1991). Inductive types and type constraints in the second-order lambda calculus. *Ann. Pure Appl. Logic*, 51(1-2):159–172. Second Annual IEEE Symposium on Logic in Computer Science (Ithaca, NY, 1987).

[MR92h:03020] The author puts forward a formal system which arises as an extension of the second-order lambda calculus (e.g., Girard's F system), obtained by adding two type constructors and ν. Strengthening the expressive power, these operations give the least and the greatest solutions to positively defined type expressions. Using Girard's method of reducibility candidates, the author proves that all typed terms of the system presented are strongly normalizable. Apart from the strong normalization for this second-order calculus with general inductive type constructors, he gives a syntactic condition on equational type constraints which holds exactly when the resulting terms are strongly normalizable. For the entire collection see MR 91k:03004.

Mezghiche, M. (1991). Weak completeness of type assignment in λ-calculus models: a generalization of Hindley's result. *Theoret. Comput. Sci.*, 87(1):203–208.
[MR92m:03019]

This paper concerns the completeness problem in Curry's type assignment and introduces a notion of "weak completeness". J. Baeten and B. Boerboom [Indag. Math. 41 (1979), no. 2, 111–120; MR 80h:03018] proved that for any closed term M, there exists a λ-model D in which M and $(\lambda x.xx)(\lambda x.xx)$ have the same interpretation. This fact shows that the relation

$$(\exists D.D \models \alpha M) \Rightarrow \vdash \alpha M$$

does not hold for any closed term M.

The author proves that, given a λ-model D, there is a rule to be added to Curry's type assignment in order to obtain "completeness relative to D", i.e.,

$$(D \models \alpha M) \Rightarrow \vdash_D \alpha M,$$

where the new type assignment $\vdash_D$ is obtained by adding to the usual rules $(\to I)$, $(\to E)$ the rule (Eq_D): from αX and $\forall \rho\, [\![X]\!]_\rho = [\![Y]\!]_\rho$, infer αY.

Completeness is proved by Hindley's standard technique, using a term model Λ_D whose elements are the D-equality classes of terms.

This result makes it possible to show a weak completeness property for Curry's type assignment, i.e., for all closed terms M,

$$(\exists D.D \models \alpha M) \Rightarrow (\exists M^* =_D M:\ \vdash \alpha M^*).$$

In other words, if, in a λ-model D, $[\![M]\!] \in [\![\alpha]\!]_\nu$, then there exists a term M^* in the D-equality class of M for which the type α is deducible by no application of the rule (Eq_D). Reviewed by Ines Margaria.

Palmgren, E. (1991). A construction of Type:Type in Martin-Löf's partial type theory with one universe. *J. Symbolic Logic*, 56(3):1012–1015.
[MR92h:03090] The author shows that it is possible to define an interpretation of a version of the type theory Type:Type in Martin-Löf's partial type theory with a single universe. The significance of partial type theory is that it has a general fixed point operator for operations on types. Combining this with the universe allows solutions of "domain equations". The author observes that the same techniques can be used to give interpretations of other impredicative type theories such as the second-order lambda calculus. Reviewed by Edmund Robinson.

Plotkin, G. (1991). A semantics for type checking. In *Theoretical aspects of computer software (Sendai, 1991)*, pages 1–17. Springer, Berlin.
[MR93a:68086] Summary: "Curry's system for F-deducibility is the basis for implicit type checking for programming languages such as ML. If a natural 'preservation of types by conversion' rule is added it becomes undecidable, but complete relative to a variety of model classes. We show completeness for F-deducibility itself, relative to an extended notion of model which validates reduction but not conversion. Both term-model and filter-model proofs are given, and the extension to polymorphic typing is also considered." For the entire collection see MR 92h:68009.

Streicher, T. (1991). *Semantics of type theory.* Birkhäuser Boston Inc., Boston, MA. Correctness, completeness and independence results, With a foreword by Martin Wirsing.
[MR93d:68042] Based on the author's Ph.D. thesis, this book studies the semantics of type theories, or typed lambda calculi, with dependent types and/or polymorphism. (Some well-known examples are automath, Martin-Löf's constructive theories, the Nuprl type system, Girard's System F and the calculus of constructions.) These systems may be used in foundational studies of constructive mathematics or as languages for writing functional programs. When considered as programming languages, these type theories provide a framework for asserting and proving program correctness, or for developing programs in accordance with their formal specifications.

The main semantic framework is a category-theoretic concept called a doctrine of constructions, which is an extension of J. Cartmell's contextual categories [88j:03046Ann. Pure Appl. Logic 32 (1986), no. 3, 209–243; MR 88j:03046]. Two particular doctrines of constructions are considered, one based on realizability and the other domain-theoretic. This framework is then used to study a variant of Coquand and Huet's calculus of constructions, first by showing how to interpret the calculus in any doctrine and then by constructing and analyzing a term model. The book, which will be most useful to those familiar with the area, begins with a short historical discussion and ends with a comparison to related efforts. Reviewed by John C. Mitchell.

Barendregt, H. P. (1992). Lambda calculi with types. In *Handbook of logic in computer science, Vol. 2*, pages 117–309. Oxford Univ. Press, New York.

[MR97m:03026] I received the present paper from Mathematical Reviews at the end of April 1997. It was published in 1992, and previous versions of it have circulated among interested researchers since 1989. I asked myself about the meaningfulness of reviewing a paper five years after its publication. Maybe it makes sense when a good paper was unfairly ignored. But "Lambda calculi with types" soon became the "bible" of everybody working on this subject. It also stimulated research: all the open problems listed on page 183 have been solved. The nomenclature used for the first time in this paper, for example the distinction between typing a la Curry versus typing a la Church, the thinning lemma and the generation lemma, etc., are currently adopted in the majority of the papers dealing with typing, not only for lambda calculi. The classifications of typed lambda calculi and of systems of intuitionistic logic through cubes are usually called "Barendregt's cubes". More remarkably, the notion of a pure type system, which previously had been almost ignored, became an object of study for many researchers.

As a matter of fact, this paper completely deserves its popularity. So in what follows I will try to briefly explain why. The writing style is clear and friendly, but at the same time totally accurate and mathematically precise. A masterpiece is the proof of Girard's paradox, a highly technical result explained in all details with great simplicity. This writing style reflects the broad (not only mathematical) culture of the author, which allows him to state interesting and deep connections in a fascinating way. For the entire collection see MR 96j:68006. Reviewed by Mariangiola Dezani-Ciancaglini.

Benton, P. N. (1992). Strictness logic and polymorphic invariance. In *Logical foundations of computer science—Tver '92*, pages 33–44. Springer, Berlin.

[MR94j:03030] Summary: "We describe a logic for reasoning about higher-order strictness properties of typed lambda terms. The logic arises from axiomatising the inclusion order on certain closed subsets of domains. The axiomatisation of the lattice of strictness properties is shown to be sound and complete, and we then give a program logic for assigning properties to terms. This places work on strictness analysis via type inference on a firm theoretical foundation. We then use proof-theoretic techniques to show how the derivable strictness properties of different instances of polymorphically typed terms are related." For the entire collection see MR 94h:68009.

Coquand, T. (1992). The paradox of trees in type theory. *BIT*, 32(1):10–14.

[MR94c:03008] Call a set x normal if $x \notin x$. Russell's paradox tells us that there is no set of normal sets. Similarly, one can call a rooted tree normal if it is not equal to any of the subtrees immediately below the root. Again there is a paradox: putting all normal trees under a single root does not yield a tree. In the paper under review it is shown how this paradox of trees can be translated into (dependent) type theory—using the W type constructor—and how it gives rise to a fixed point combinator. For the entire collection see MR 93h:68089. Reviewed by Bart Jacobs.

Curien, P.-L. and Ghelli, G. (1992). Coherence of subsumption, minimum typing and type-checking in $F_{\leq}$. *Math. Structures Comput. Sci.*, 2(1):55–91.

[MR93i:68120] Coherence is a technical problem that arises in assigning meaning to expressions. Specifically, it is sometimes natural to assign meaning to an expression using some additional property of the expression that is not uniquely determined. In this case, it is necessary to establish that the meaning assigned to each expression is uniquely determined, independent of the way additional properties of the term are identified.

This paper is concerned with a polymorphic functional language with subtyping. Subtyping provides a pre-order on the types of expressions. If A is a subtype of B, then subsumption means that if e is an expression of type A then e is also an expression of type B. In this setting, there may be several ways of establishing that an expression has a certain type, with a coherence problem resulting from the fact that the meaning of a term is most naturally given according to the way we establish that it has a type.

The technique used for solving the coherence problem is to transform a proof that an expression has a type into a certain standard form. This is used to prove three interesting and important properties of the language: (i) a semantic interpretation is coherent if and only if the transformations on proofs preserve the meaning of expressions, (ii) every expression has a minimum type, with respect to the type preorder, given a unique type for each variable occurring in the term, and (iii) there is a deterministic type checking algorithm for expressions. Reviewed by John C. Mitchell.

Dezani-Ciancaglini, M. and Hindley, J. R. (1992). Intersection types for combinatory logic. *Theoret. Comput. Sci.*, 100(2):303–324.

[MR93i:03018] Summary: "Two different translations of the usual formulation of intersection types for λ-calculus into combinatory logic are proposed; in the first one the rule ($\leq$) is unchanged, while in the second one the rule ($\leq$) is replaced by three new rules and four axiom-schemes, which seem to be simpler than rule ($\leq$) itself."

Fredholm, D. and Serafimovski, S. (1992). Partial inductive definitions as type-systems for λ-terms. *BIT*, 32(1):30–44.

[MR95b:03010] Summary: "With every partial inductive definition D, a natural deduction calculus ND(D) is associated. Not every such system will have the normalization property; specifically, there are definitions D' for which ND(D') permits non-normalizable deductions. A lambda calculus is formulated where the terms are used as objects realizing deductions in ND(D), and is shown to have the Church-Rosser property. Since ND(D) permits non-normalizable deductions, there will be typed terms which are non-normalizable. It will, for example, be possible to obtain a typed fixed-point operator." For the entire collection see MR 93h:68089.

Hirokawa, S. (1992). The converse principal type-scheme theorem in lambda calculus. *Studia Logica*, 51(1):83–95.

[MR93k:03012] Summary: "A principal type-scheme of a λ-term is the most general type-scheme for the term. The converse principal type-scheme theorem of R. Hindley [Trans. Amer. Math. Soc. 146 (1969), 29–60; MR 40 #7118] states that

every type-scheme of a combinatory term is a principal type-scheme of some combinatory term.

"This paper shows a simple proof for the theorem in λ-calculus, by constructing an algorithm which transforms a type assignment to a λ-term into a principal type assignment to another λ-term that has the type as its principal type-scheme. The clearness of the algorithm is due to the characterization theorem of principal type-assignment figures. The algorithm is applicable to BCIW-λ-terms as well. Thus a uniform proof is presented for the converse principal type-scheme theorem for general λ-terms and BCIW-λ-terms." (Here, BCIW-λ-terms are also known as λI-terms; they are terms in which $\lambda x.M$ is a well-formed term only when there is a (free) occurrence of x in M. Also, type-assignment figures are constructed from the following two rules: ($\to$ E): $M\colon \alpha \to \beta, N\colon \alpha \vdash MN\colon \beta$, and ($\to$ I): from $\Gamma, x : \alpha \vdash M\colon \beta$ to infer $\Gamma \vdash \lambda x.M\colon \alpha \to \beta$.)

The basic idea behind the proof is to note that every type-scheme is obtained from a principal type-scheme by a substitution, and to consider the kinds of substitutions that can occur in this process. Thus, suppose that in a principal type-scheme α of a term M, the type $d \to e$ is substituted for a to obtain the type α', and suppose that in M the variable y is assigned type a (by a discharged assumption). (Here lower-case Greek letters stand for type-schemes and lower-case Roman letters for type variables.) Then if y in M is replaced by $\lambda v.yv$, the result is a term whose principal type-scheme is α'. Similarly, if α' is obtained from α by substituting b for a, where a occurs in α, and if, in M, x is assigned $a \to b$ and y is assigned a (so that xy is assigned b), then in M replace y and xy by fy and $f(xy)$ respectively, abstract with respect to f, and apply the resulting term to $\lambda v.v$, and the resulting term will have principal type-scheme α'.

Reviewer's remarks: In general the paper is well written. The reviewer found three minor misprints: on p. 85, line 10, 'formul' should be 'formula'; on p. 87, top line, 'each other' should be 'every other'; and on p. 87, Theorem 2, line 3, 'each maximal' should be 'every maximal'. Reviewed by J. P. Seldin.

Jacobs, B., Margaria, I., and Zacchi, M. (1992). Filter models with polymorphic types. *Theoret. Comput. Sci.*, 95(1):143–158.

[MR93j:03012] Summary: "Using ideas and results from earlier papers [H. Barendregt, M. Coppo and M. Dezani-Ciancaglini, J. Symbolic Logic 48 (1983), no. 4, 931–940 (1984); MR 85j:03014; M. Coppo et al., in Logic colloquium '82 (Florence, 1982), 241–262, North-Holland, Amsterdam, 1984; MR 87g:03018] on intersection types, a comparable theory is developed for (second-order) polymorphic types. The set of filters constructed with polymorphic types forms, with inclusion, a continuous lattice which yields a model of what we call $\beta\eta$-expansion (i.e., the value of a term increases under $\beta\eta$-reduction), but not of β-conversion. Combining intersection with polymorphic types does give filter λ-models, but the two standard ways of interpreting λ-terms do not coincide."

Kfoury, A. J. and Tiuryn, J. (1992). Type reconstruction in finite rank fragments of the second-order λ-calculus. *Inform. and Comput.*, 98(2):228–257. Selections from the 1990 IEEE Symposium on Logic in Computer Science.

[MR94a:03020] One of the main open problems concerning the polymorphic λ-

calculus F_2 of J.-Y. Girard ["Interpretation fonctionnelle et elimination des coupures de l'arithmetique d'ordre superieur", These d'Etat, Univ. Paris VII, Paris, 1972] and J. C. Reynolds [in Programming Symposium (Paris, 1974), 408–425, Lecture Notes in Comput. Sci., 19, Springer, Berlin, 1974; MR 56 #17187] is whether the typability problem for F_2 is decidable. More precisely, it is not known whether there is an algorithm which, given a λ-term M, decides whether $A \vdash M : \sigma$ for some type σ and some environment A which assigns at most one type to each variable. The authors consider the typability problem, and other type reconstruction problems, in various subsystems of F_2 of finite "rank". The definition of rank considered here is equivalent to the one of D. Leivant [in Tenth Annual ACM Symposium on Principles of Programming Languages (Austin, TX, 1983), 88–98, ACM, New York, 1983]. One of the main results of the paper under review is that typability in Λ_2, the polymorphic λ-calculus of rank 2, is polynomial-time equivalent to typability in the system ML. It follows that typability in Λ_2 is (decidable and) DEXPTIME-complete by unpublished results of Kanellakis, Mairson and Mitchell, and of the authors and P. Urzyczyn. An algorithm to decide typability in Λ_2 was presented by N. McCracken [in Semantics of data types (Valbonne, 1984), 301–315, Lecture Notes in Comput. Sci., 173, Springer, Berlin, 1984; see MR 85m:68003] using previous work of Leivant. In the paper under review it is also shown that, for every extension of Λ_2 with constants which are assigned universally polymorphic types, the typability problem is still decidable. A final result is that for every $k > 2$ the typability problem in the polymorphic λ-calculus of rank k extended with a suitably chosen set of constants with types of rank 1 is undecidable. This last result uses the undecidability of semi-unification [A. J. Kfoury, J. Tiuryn and P. Urzyczyn, Inform. and Comput. 102 (1993), no. 1, 83–101] together with a reduction to the typability problem in ML^+, which is ML extended with "polymorphic recursion" as defined by A. Mycroft [in International Symposium on Programming (Toulouse, 1984), 217–228, Lecture Notes in Comput. Sci., 167, Springer, Berlin, 1984; see MR 85k:68008]. There are related papers by F. Henglein [in Proceedings of the ACM Conference on LISP and Functional Programming (Snowbird, UT, 1988), 184–197, ACM, New York, 1988] and by the authors and Urzyczyn [in Proceedings of the 15th ACM Symposium on Principles of Programming Languages, 58–69, ACM, New York, 1988; per bibl.]. For the entire collection see MR 92m:68004. Reviewed by Alessandro Berarducci.

Ma, Q. and Reynolds, J. C. (1992). Types, abstraction, and parametric polymorphism. II. In *Mathematical foundations of programming semantics (Pittsburgh, PA, 1991)*, pages 1–40. Springer, Berlin.
[MR1251307] This item will not be reviewed.

Madry, M. (1992). λ-representability of integer, word and tree functions. *Fund. Inform.*, 17(3):253–270.
[MR93m:68115] Summary: "Six λ-languages over function types between algebras B, N and Υ are considered. Type $N = (0 \to 0) \to (0 \to 0)$ is called a nonnegative integers type; $B = (0 \to 0) \to ((0 \to 0) \to (0 \to 0))$ is called a binary words type;

$\Upsilon = (0 \to (0 \to 0)) \to (0 \to 0)$ is called a binary trees type. These associations come from the isomorphism between the types and corresponding algebraic structures. Closed terms whose types are the above-mentioned function types represent unary functions of appropriate types. The problem is to determine what class of functions is represented by the closed terms of the examined type. It is proved that for $B \to N$, $N \to B$, $\Upsilon \to N$ and $\Upsilon \to B$ there exists a finite base of functions such that any λ-definable function is some combination of the base functions. An algorithm which, for every closed term, returns the function in the form of a combination of the base functions is given. For two other types, $B \to \Upsilon$ and $N \to \Upsilon$, a method of constructing λ-representable functions using primitive recursion is shown."

Nelson, N. (1992). Primitive recursive functionals with dependent types. In *Mathematical foundations of programming semantics (Pittsburgh, PA, 1991)*, pages 125–143. Springer, Berlin.

[MR1251312] This item will not be reviewed individually. For details of the collection in which this item appears see 94g:68008.

Palmgren, E. and Stoltenberg-Hansen, V. (1992). Remarks on Martin-Löf's partial type theory. *BIT*, 32(1):70–83.

[MR94c:03024] This paper gives a domain-theoretic model of partial type theory with one universe. In such a model one can define functions by general recursions and solve recursive domain equations up to equality. The basic idea is to introduce a large dcpo of information systems (this idea is not new), and to define on it continuous functions for interpreting the type constructors of type theory.

The authors have ignored related work on the solution of domain equations up to equality using information systems [K. G. Larsen and G. Winskel, in Semantics of data types (Valbonne, 1984), 109–129, Lecture Notes in Comput. Sci., 173, Springer, Berlin, 1984; MR 86d:68046], and on domain-theoretic models for calculi with dependent types [P. Taylor, "Recursive domains, indexed category theory and polymorphism", Ph.D. Thesis, Univ. Cambridge, Cambridge, 1987; per revr.], which they should perhaps have cited. For the entire collection see MR 93h:68089. Reviewed by Eugenio Moggi.

Phoa, W. (1992). Using fibrations to understand subtypes. In *Applications of categories in computer science (Durham, 1991)*, pages 239–257. Cambridge Univ. Press, Cambridge.

[MR94d:68068] Summary: "Fibrations have been widely used to model polymorphic λ-calculi. In this paper we describe the additional structure on a fibration that suffices to make it a model of a polymorphic λ-calculus with subtypes and bounded quantification; the basic idea is to single out a class of maps, the inclusions, in each fibre. Bounded quantification is made possible by imposing a condition that resembles local smallness. Since the notion of inclusion is not stable under isomorphism, some care must be taken to make everything strict.

"We then show that PER models for λ-calculi with subtypes fit into this framework. In fact, not only PERs, but also any full reflective subcategory of the category of modest sets ('PERs' in a realizability topos) provides a model; hence all the small complete categories of 'synthetic domains' found in various realizability toposes can be used to model subtypes." For the entire collection see MR 93d:68001.

Salvesen, A. (1992). On specifications, subset types and interpretation of proposition in type theory. *BIT*, 32(1):84–101.

[MR94a:03022] Summary: "In this paper we discuss some practical aspects of using type theory as a programming and specification language, where the viewpoint is to use it not only as a basis for program synthesis but also as a programming language with a programming logic allowing us to do ordinary verification.

"The subset type has been added to type theory in order to avoid irrelevant information in programs. We give an example of a proof which illustrates the problems that may occur if the subset type is used in specifications when we have the standard interpretation of propositions as types. Harrop formulas and the operation Squash are then discussed as solutions to these problems. It is argued that they are not acceptable from a practical point of view.

"An extension of the theory to include the two new judgment forms 'A is a proposition' and 'A is true' is then given and explained in terms of the old theory. The logical constants are no longer identified with the corresponding type-theoretical constants, but propositions are interpreted as Gödel formulas, which allows us to introduce and justify logical rules similar to rules for classical logic. The interpretation is extended to include predicates defined by using reflections of the ordinary definition of Gödel's formulas in a 'type of small propositions'. The programming example is then revisited and stronger elimination rules are discussed." For the entire collection see MR 93h:68089.

van Bakel, S. (1992). Complete restrictions of the intersection type discipline. *Theoret. Comput. Sci.*, 102(1):135–163.

[MR94b:03025] In this paper the intersection type discipline for the λ-calculus is studied. Because of its undecidability properties the intersection type discipline is not practicable for type checkers of functional languages. In order to obtain a type checker based on this system some restrictions have to be made. In this paper, two complete restrictions of the intersection type discipline are carefully studied. The first is based on a set of strict types and the second is mainly characterized by the absence of the type ω. For both systems several important results, such as completeness and typeability, are proved. Unfortunately, both yield undecidable systems. Thus these attempts to restrict the system in preparation for the construction of a type checker failed. Reviewed by E. Fehr.

Abadi, M., Cardelli, L., and Curien, P.-L. (1993). Formal parametric polymorphism. *Theoret. Comput. Sci.*, 121(1-2):9–58. A collection of contributions in honour of Corrado Böhm on the occasion of his 70th birthday.

[MR95e:68048] The authors introduce a formal system $\mathcal{R}$, an extension of Girard's system F, in order to develop a syntactic approach to parametricity. A polymorphic function is parametric if, roughly, its behaviour does not depend on the type at which it is instantiated. The formal notion is the one suggested by Reynolds' semantic approach: a polymorphic function is parametric if, for any relation between types, its instances at related types behave in related ways. Therefore the authors employ variables $\mathcal{W}$ for (binary) relations and judgements of the forms $E \vdash \overset{A}{\to} \underset{B}{\to} \mathcal{W}$, $E \vdash \overset{a:A}{\to} \underset{b:B}{\to} \mathcal{W}$, where E is an environment and A, B are types. Instead of equality

judgements such as $E \vdash b = c : A$, there is a relation A^* corresponding to the identity on type A, and judgements such as $E \vdash^{b:A}\!\!\rightarrow \underset{c:A}{\rightarrow} A^*$. The rules of the system $\mathcal{R}$ are the natural ones, the aim being to deal with Reynolds' semantics. In this system it is possible to formally derive theorems about functions from their types only; there is no need to look at their codes ("theorems for free" in Wadler's wording), even for nonclosed terms. After showing how to formalize some technical tools in $\mathcal{R}$, the authors derive syntactic, induction-free proofs of a number of results, proved semantically, for closed terms, by Wadler, as well as new ones, and also give a syntactic version of Reynolds' abstraction theorem. For the entire collection see MR 94e:68005. Reviewed by Aldo Ursini.

Baker-Finch, C. (1993). Relevance and strictness. In *Proceedings of the Sixteenth Australian Computer Science Conference (ACSC-16) (Brisbane, 1993)*, volume 15, pages 703–709.

[MR94k:03014] The author studies the concept of strictness in the context of the simply typed lambda calculus. He follows the approach taken by D. A. Wright [TAPSOFT '91. Vol. II (Brighton, 1991), 235–258, Lecture Notes in Comput. Sci., 494, Springer, Berlin, 1991], which essentially consists in extending the language of types by using two distinct kinds of type constructors, $\rightarrow$ and $\not\rightarrow$, where a term M is given type $\sigma \rightarrow \tau$ to indicate that M is strict, i.e., M needs its argument, or is given type $\sigma \not\rightarrow \tau$ to indicate that M is constant, i.e., it returns the same value regardless of its argument's value. Note that the simply typed lambda calculus is strongly normalizable, so strictness is not to be intended in the sense of Scott (according to Scott a function is strict if it always returns an undefined value when one of its arguments is undefined). An inconvenient feature of the system with $\rightarrow$ and $\not\rightarrow$ is that some Curry-typable terms are not typable (since a variable can have both strict and nonstrict occurrences in the same term). To overcome this problem the author studies a system possessing also a third kind of type constructor $\supset$, which gives no strictness information when no such information is possible. He develops a Curry-Howard isomorphism for the resulting system and gives a soundness and completeness result with respect to a suitable semantics. Roughly, this result states that the type assignment system can derive as much strictness information as is consistent with the given semantics. For the entire collection see MR 94d:68009. Reviewed by Alessandro Berarducci.

Berardi, S. (1993). Encoding of data types in pure construction calculus: a semantic justification. In *Logical environments (Edinburgh, 1991)*, pages 30–60. Cambridge Univ. Press, Cambridge.

[MR1255109] This item will not be reviewed individually. For details of the collection in which this item appears see 94f:03004.

Breazu-Tannen, V. and Meyer, A. R. (1993). Conservativity of equational theories in typed lambda calculi. *Fund. Inform.*, 19(1-2):1–49.

[MR95e:03045] A great deal is known about the metatheory of typed lambda calculi, but almost all of it concerns pure type theories. Comparatively little is known

about theories built on some collection of basic types and operations; most of what is known is due to the authors of the current paper. One of the main results here is that, given a theory of the simply typed lambda calculus $\lambda^{\rightarrow}(\Sigma, E)$, where Σ is a signature and E is a set of equations, the extension to a theory of the second-order lambda calculus $\lambda^{\forall}(\Sigma, E)$ is conservative. Similarly, given an algebraic theory (Σ, E), the extension to the second-order lambda calculus is conservative. However, this extension does not add any new first-order functions. If we want this extra computational power, then we have to link the base types to polymorphic types. The authors explore one way of doing this. They pick a base type of observables, along with a countable sequence of observable elements. They then introduce coding and decoding functions from the type of polymorphic integers which code this sequence (for each observable there is one equation to say that coding followed by decoding is the identity, along with a single equation to say that decoding followed by coding is the identity on the polymorphic integers). It is obvious that this increases the number of functions definable on the type of observables (since there is one corresponding to each function provably terminating in second-order arithmetic); however, the authors show that this is still a conservative extension. Reviewed by Edmund Robinson.

Di Cosmo, R. (1993). Deciding type isomorphisms in a type-assignment framework. *J. Funct. Programming*, 3(4):485–525.

[MR95h:68117] From the text: "This paper develops a complete proof system for type isomorphism in a type-assignment framework and gives a decision procedure. The main contribution of the paper is to set up a formalism for this proof system, to discover a new isomorphism (split), and to develop an efficient algorithm to check isomorphism, which can be used to improve library search systems, as investigated by Rittri and others. These results contribute to a better understanding of type isomorphism in ML-style type systems." Reviewed by Peter Bachmann.

Forster, T. (1993). A semantic characterization of the well-typed formulae of λ-calculus. *Theoret. Comput. Sci.*, 110(2):405–418.

[MR94b:03026] The author adapts techniques developed by set theorists to give a semantic characterization of typable λ-terms. In order to use these techniques, application is viewed as a ternary relation, and typability of a term is expressed by an existential formula. Perhaps the main contribution of this paper is to show the possibility of a "technology transfer" from an area of mathematics apparently unrelated to computer science. Reviewed by Eugenio Moggi.

Fujiwara, Y. (1993). Dinaturality of simple subtyping. *Sūrikaisekikenkyūsho Kōkyūroku*, (851):139–152. Type theory and its applications to computer systems (Japanese) (Kyoto, 1993).

[MR95i:68084] Summary: "Semantic parametricity of a simply-typed λ-calculus with subtyping is discussed. Viewing subtyping as implicit coercion, we give a category-theoretic formulation of semantic parametricity. Although semantic parametricity may fail in arbitrary Cartesian closed categories (CCCs), a simple proof-theoretic argument gives us a sufficient condition on subtyping contexts such that terms satisfy semantic parametricity in arbitrary CCCs." For the entire collection see MR 95f:68011.

Gavilanes-Franco, A., Lucio-Carrasco, F., and Rodríguez Artalejo, M. (1993). Reasoning with higher order partial functions. In *Computer science logic (San Miniato, 1992)*, pages 167–181. Springer, Berlin.

[MR94i:03043] Summary: "In this paper we introduce the logic PHOL, which embodies higher-order functions through a simply-typed λ-calculus and deals with partial objects by using partially ordered domains and three truth values. We define a refutationally complete tableau method for PHOL and we show how to derive a sound and complete cut-free sequent calculus through a systematic analysis of the rules for tableau construction." For the entire collection see MR 94f:68011.

Ghilezan, S. (1993). Inhabitation in intersection and union type assignment systems. *J. Logic Comput.*, 3(6):671–685.

[MR95a:68072] This paper relates inhabitation in two λ-calculi: that with intersection and union, and that with product and sum (or, logically speaking, with conjunction and disjunction). Decidability of inhabitation in the first calculus is not known, while the second is decidable (it corresponds to validity in intuitionistic propositional logic).

The author introduces two translations: $\phi^{\wedge}$ and $M^{\cap}$. The translation $\phi^{\wedge}$ replaces intersection and union in a type ϕ with conjunction and disjunction, while $M^{\cap}$ (which may be undefined) erases from the term M the constants corresponding to the elimination and introduction rules for conjunction and disjunction.

The main result says that $M : \phi$ iff $N : \phi^{\wedge}$ for some N such that $N^{\cap} = M$. However, this is not so informative, as $N^{\cap}$ may be undefined. Reviewed by Eugenio Moggi.

Giannini, P., Honsell, F., and Ronchi Della Rocca, S. (1993). Type inference: some results, some problems. *Fund. Inform.*, 19(1-2):87–125.

[MR95f:03024] This paper is a survey of the type inference problem for λ-calculus, which is the problem of determining whether a given λ-term is typable in a given type assignment system.

It is proved that it is enough to consider the system F_ω induced by Girard's explicit type system F^ω, since all other systems belonging to "Barendregt's cube" are equivalent to F_ω or to one of its first two subsystems F_1 and F_2, from the typability point of view. In other words, dependencies do not increase the computational strength of the system.

The notion of type scheme is then defined and used for investigating the type inference problem for F_ω, and to reduce it to a restriction of the higher-order semi-unification problem.

Much progress has been made since the paper was written: recently J. B. Wells proved, using the undecidability of the semi-unification problem [A. J. Kfoury, J. Tiuryn and P. Urzyczyn, Inform. and Comput. 102 (1993), no. 1, 83–101; MR 94b:03080], that typability for F_2 is undecidable ["Typability and type checking in the second-order λ-calculus are equivalent and undecidable", in Proceedings of the Ninth Annual IEEE Symposium of LICS, (Paris 1994), 176–185; per revr.]. P. Urzyczyn succeeded earlier in showing that typability in F_n is undecidable for $n \geq 3$, by reducing it to the halting problem of a Turing machine [in Typed lambda calculi and applications (Utrecht, 1993), 418–432, Lecture Notes in Comput. Sci., 664, Springer, Berlin, 1993; MR 94c:03026]. However, the question of whether the computational strength of F_ω is already contained in F_3, which is conjectured in the last section of the present paper, is still open. Reviewed by Sophie Malecki.

Hirokawa, S. (1993). Principal types of BCK-lambda-terms. *Theoret. Comput. Sci.*, 107(2):253–276.

[MR94g:03028] Summary: "BCK-λ-terms are the λ-terms in which each variable occurs at most once. The principal type of a λ-term is the most general type of term. In this paper we prove that if two BCK-λ-terms in β-normal form have the same principal type then they are identical. This solves the following problem (Y. Komori, 1987) in more general form: If two closed $\beta\eta$-normal form BCK-λ-terms are assigned to the same minimal BCK-formula, are they identical? (A minimal BCK-formula is the most general formula among BCK-provable formulas with respect to substitutions for type variables.)

"To analyze type assignment, the notion of a connection is introduced. A connection is a series of occurrences of a type in a type assignment figure. Connected occurrences of a type have the same meaning. The occurrences of the type in distinct connection classes can be rewritten separately; as a result, we have more general type assignment.

"By the formulae-as-types correspondence, our result implies the uniqueness of the normal proof figure of principal BCK-formulas. This result is valid for BCI-logic and for the implicational fragment of linear logic as well."

Krivine, J.-L. (1993). *Lambda-calculus, types and models.* Ellis Horwood, New York. Translated from the 1990 French original by René Cori.

[MR94h:03034] The French original has been reviewed [Masson, Paris, 1990; MR 93i:03019].

Longo, G., Milsted, K., and Soloviev, S. (1993). The genericity theorem and parametricity in the polymorphic λ-calculus. *Theoret. Comput. Sci.*, 121(1-2):323–349. A collection of contributions in honour of Corrado Böhm on the occasion of his 70th birthday.

[MR94m:03030] It is traditional to make a distinction between "parametric" and "ad hoc" polymorphism. A parametric polymorphic value behaves uniformly in its various instantiations, while an ad hoc one does not. There have been various attempts to capture the difference, both semantically and proof-theoretically. This paper is much more at the proof-theoretic end. The authors consider extending Girard's second-order lambda calculus with an extra equational axiom, Axiom C: $M\tau = M\tau'$ for $\Gamma \vdash M : \forall X.\sigma$ and $X \notin \mathrm{FV}(\sigma)$. This axiom says that if M is a term of polymorphic type ($\forall X.\sigma$), and σ does not depend on the interpretation of the type variable X, then M itself does not depend on the interpretation of X. This rules out (for instance) definitions by branching on types, and not surprisingly it holds in many of the models of the calculus.

The main result of this paper is the "genericity theorem": Assume M and N live in the same type $\forall X.\sigma$. If $M_\tau =_{F^c} N_\tau$ for some type τ, then $M =_{F^c} N$. (Here " $=_{F^c}$" is the (provable) equality in the second-order lambda calculus augmented by Axiom C above.)

This theorem says that in order to test the equality of two polymorphic operations, it suffices to test them at a single type, and that any type will do. This is a powerful and striking result, and comes very close to formalising the parametricity of the

second-order lambda calculus in Strachey's sense. Its limitation is that it refers to the syntactic calculus, rather than to equality in a model. In fact, the genericity theorem fails in most models, even those for which Axiom C holds. This again is not too surprising. If the model contains a type with one element, then the zero iterator and the single iterator will coincide for that type. The genericity theorem then implies that they coincide for all types. Hence all endofunctions are the identity. This includes negation on the Booleans, which is enough to imply that any two elements of a type are equal.

Now, a consequence of most of the semantic definitions of parametricity is that the interpretation of the type $\forall X.X \to X$ has a single element, and so the definition of parametricity as "satisfying the genericity theorem" conflicts with these other approaches. The moral is probably that equality in the formal calculus is a rather weak notion, and that it cannot be confused with the more extensional notion of equality in a model. For the entire collection see MR 94e:68005. Reviewed by Edmund Robinson.

Malecki, S. (1993). Sur le problème de la typabilité dans le système F. *C. R. Acad. Sci. Paris Sér. I Math.*, 316(11):1111–1116.
[MR94d:03024] Summary: "A system of typing is defined which is a subsystem of the second-order typed lambda-calculus (or system F). We prove that this system (system F) is strictly included in the original one by studying the typability of some pure lambda-terms. Moreover, we show that the typability problem in F is reducible to the typability problem in F′."

Otake, K. (1993). Towards imperative type systems. *Sūrikaisekikenkyūsho Kōkyūroku*, (851):127–138. Type theory and its applications to computer systems (Japanese) (Kyoto, 1993).
[MR1304529] This item will not be reviewed individually. For details of the collection in which this item appears see 95f:68011.

Palmgren, E. (1993). An information system interpretation of Martin-Löf's partial type theory with universes. *Inform. and Comput.*, 106(1):26–60.
[MR94m:68117] Summary: "Martin-Löf's partial type theory, i.e., type theory with fixed point operator, is extended with universes and given a domain-theoretic interpretation using Scott's information systems. The dependent types are interpreted as monotone families (parametrizations) of information systems and elements as relativized approximable mappings. Universes are interpreted as fixed points of continuous operators (on the class of parametrizations) that are induced by the introduction rules for universes. Partial type theory has a rich type structure. In fact, as is shown elsewhere, Martin-Löf's inconsistent type theory, Type : Type, is definable in it. However, in this paper we give a direct interpretation of (an extension of) Type : Type as an application of the domain-theoretic method of solving type universe equations introduced here." Reviewed by Horst Reichel.

Phoa, W. (1993). Adequacy for untyped translations of typed λ-calculi. In

Eighth Annual IEEE Symposium on Logic in Computer Science (Montreal, PQ, 1993), pages 287–295. IEEE Comput. Soc. Press, Los Alamitos, CA.

[**MR1238291**] This item will not be reviewed individually. For details of the collection in which this item appears see 94d:68005.

Pierce, B. C. (1993). Intersection types and bounded polymorphism. In *Typed lambda calculi and applications (Utrecht, 1993)*, pages 346–360. Springer, Berlin.

[**MR94e:68112**] Summary: "Intersection types and bounded quantification are complementary extensions of a first-order statically typed programming language with subtyping. We define a typed λ-calculus combining these extensions, illustrate its properties, and develop proof-theoretic results leading to algorithms for subtyping and typechecking." For the entire collection see MR 94b:68025.

Poll, E., Hemerik, C., and ten Eikelder, H. M. M. (1993). cpo-models for second order lambda calculus with recursive types and subtyping. *RAIRO Inform. Théor. Appl.*, 27(3):221–260.

[**MR94m:68118**] Constructions of CPO models are shown for second-order lambda calculi with recursive types and/or subtyping. Types are interpreted as CPOs. The combination and interaction of recursive types and subtyping does not pose any problems. The constructions are based on a standard fixed-point result for ω-categories, solving a set of recursive domain equations by an inverse-limit construction in a product category. Coercion functions are used to give the semantics of subtyping. A key point here is to prove coherence, which means that all type derivations for a term give the same meaning. In order to satisfy both the recursive domain equations and the coherence conditions one has to consider an inverse-limit construction in a functor category from the subtype relation on types (viewed as a category) to CPO. Reviewed by Teruo Hikita.

Springintveld, J. (1993). Lower and upper bounds for reductions of types in $\lambda\underline{\omega}$ and λP (extended abstract). In *Typed lambda calculi and applications (Utrecht, 1993)*, pages 391–405. Springer, Berlin.

[**MR1233162**] This item will not be reviewed individually. For details of the collection in which this item appears see 94b:68025.

van Bakel, S. (1993a). Essential intersection type assignment. In *Foundations of software technology and theoretical computer science (Bombay, 1993)*, pages 13–23. Springer, Berlin.

[**MR95j:68105**] Summary: "This paper introduces a notion of intersection type assignment on the lambda calculus that is a restriction of the BCD system as presented in a paper by H. P. Barendregt, M. Coppo and M. Dezani-Ciancaglini [J. Symbolic Logic 48 (1983), no. 4, 931–940 (1984); MR 85j:03014]. This restricted system is essential in the following sense: it is an almost syntax directed system that satisfies all major properties of the BCD system. The set of typeable terms can be characterized in the same way, the system is complete with respect to the simple type semantics, and it has the principal type property." For the entire collection see MR 95d:68003.

van Bakel, S. (1993b). Principal type schemes for the strict type assignment system. *J. Logic Comput.*, 3(6):643–670.

[MR94h:68126] Summary: "We study the strict type assignment system, a restriction of the intersection type discipline, and prove that it has the principal type property. We define, for a term M, the principal pair (of basis and type). We specify three operations on pairs, and prove that all pairs deducible for M can be obtained from the principal one by these operations, and that these operations map deducible pairs to deducible pairs."

van Benthem Jutting, L. S. (1993). Typing in pure type systems. *Inform. and Comput.*, 105(1):30–41.

[MR94h:03032] Generalised (or pure) type systems provide a taxonomy of type systems with various forms of product type. If a term in such a system can be typed, then its type has the form $\prod x_1 : C_1 \cdots \prod x_m : C_m.e$. In "functional" systems, this type is unique up to β-equivalence. In this paper the author shows that even in nonfunctional systems two inequivalent types A and B for a term t have a degree of common structure in that A reduces to $\prod x_1 : C_1 \cdots \prod x_m : C_m.s$ and B to $\prod x_1 : C_1 \cdots \prod x_m : C_m.s'$, where s and s' are sorts. As a corollary, "strengthening" holds:

$$\text{If}\Gamma_1, x : A, \Gamma_2 \vdash b : B \text{and } x \notin \mathrm{FV}(\Gamma_2) \cup \mathrm{FV}(b) \cup \mathrm{FV}(B) \qquad \text{then } \Gamma_1, \Gamma_2 \vdash b : B.$$

Moreover, if all terms normalise and the set of sorts is finite, then the typing relation is decidable. Reviewed by Edmund Robinson.

Wolfram, D. A. (1993). *The clausal theory of types.* Cambridge University Press, Cambridge.

[MR94j:68003] The clausal theory of types is based on an applied simply typed lambda-calculus, in which the base types include o (for propositions) and ι for individuals. Logical connectives (including equality) are interpreted as constants of appropriate types. For example, for any type α there is a constant Π of type $(\alpha \to o) \to o$ representing universal quantification. The terms allowed in the clausal theory of types form a restricted subset of the long $\beta\eta$-normal forms for the calculus, specifically those in which no constant or variable appears above an instance of Π in the term (except where the variable is being bound by a λ). Formulae are simply terms of type o. These can be written into a normal form which consists of quantifiers and binders on the outside with a term in conjunctive normal form on the inside.

The first major result of the book is a form of the "Skolem-Herbrand-Gödel" theorem for the clausal theory of types. This is enough for the theory to be usable in principle as a programming language. To make it usable in practice, however, some form of unification is required, rather than an exhaustive search, and this leads to a discussion of the current state of knowledge in the field of higher-order unification. This forms a major part of the book, and is comprehensive and informative. But it gives a picture of a theory which is still in active development, and so cannot be said to be a final account.

Reviewer's remarks: There is currently a great deal of interest in combining functional and logic programming in a higher-order logic programming language, and this book makes an interesting contribution to the theoretical side of that project, but does not, in the reviewer's opinion, contain a definitive treatment. One feature that makes the book attractive is the author's extensive and careful use of references. This has to be balanced against the author's notation and formulation of definitions, which often seem to be derived from term-rewriting rather than the lambda-calculus tradition, and tend, at least to the reviewer's taste, to be rather too congested, obscuring some of the ideas behind a screen of detail. Reviewed by Edmund Robinson.

Alessi, F. (1994). Type preorders. In *Trees in algebra and programming—CAAP '94 (Edinburgh, 1994)*, pages 37–51. Springer, Berlin.
[**MR95e:03044**] The category ITS of intersection type structures and approximable mappings is equivalent to the ccc of algebraic domains and Scott-continuous functions, which are commonly used to model λ-calculus (whereas intersection type systems correspond to particular solutions of recursive domain equations). Most commonly used are the prime-algebraic domains, which are generated from the ordered set of their prime elements and also form a ccc PA; in this case the ITS presentation is redundant. To eliminate this redundancy the author introduces the much simpler "type preorders" category TP, which is equivalent to PA.

This paper clarifies the domain-theoretic description of the ITSs and makes directly available in the intersection typing framework a reduction which is now classical at the level of domains. It can also clearly be seen here how any morphism on a prime algebraic domain may be, even in the continuous semantics, reconstructed from a "linear" one and a simple power domain operator (here, the "lifted Hoare power domain" operator). For the entire collection see MR 95b:68022. Reviewed by Chantal Berline.

Alessi, F. and Barbanera, F. (1994). Toward a semantics for the QUEST language. *RAIRO Inform. Théor. Appl.*, 28(6):513–555.
[**MR95k:68120**] Summary: "A model is given for the second-order lambda calculus extended with inheritance, bounded quantification, recursive types, constructors and kinds. This language can be viewed as the core of the QUEST language defined by L. Cardelli [in Formal description of programming concepts (Rio de Janeiro, 1989), 431–507, Springer, Berlin, 1991; see MR 92i:68002]. Types are interpreted as intervals of partial equivalence relations. In such a model the meaning of the operator μ, the constructor of recursive types, turns out to be the natural one, i.e., the least fixed point operator."

Cardelli, L., Martini, S., Mitchell, J. C., and Scedrov, A. (1994). An extension of system F with subtyping. *Inform. and Comput.*, 109(1-2):4–56. International Conference on Theoretical Aspects of Computer Software (TACS '91) (Sendai, 1991).
[**MR96d:68135**] Summary: "System F is a well-known typed λ-calculus with polymorphic types, which provides a basis for polymorphic programming languages. We study an extension of F, called $\mathrm{F}_{<:}$ (pronounced ef-sub), that combines parametric polymorphism with subtyping. The main focus of the paper is the equational theory of $F_{<:}$, which is related to PER models and the notion of parametricity.

We study some categorical properties of the theory when restricted to closed terms, including interesting categorical isomorphisms. We also investigate proof-theoretical properties, such as the conservativity of typing judgments with respect to F. We demonstrate by a set of examples how a range of constructs may be encoded in $F_{<:}$. These include record operations and subtyping hierarchies that are related to features of object-oriented languages." For the entire collection see MR 94j:68004.

Cardone, F., Dezani-Ciancaglini, M., and de'Liguoro, U. (1994). Combining type disciplines. *Ann. Pure Appl. Logic*, 66(3):197–230.

[MR95c:03033] The authors consider a type inference system with type constructors including arrow (function space), union, intersection, universal quantification and existential quantification. They note that this system has already been described by D. B. MacQueen, G. D. Plotkin and R. Sethi [Inform. and Control 71 (1986), no. 1-2, 95–130; MR 88h:68058]. Here the authors give the natural deduction formulation of the rules for type inference, following the earlier authors, and introduce a new sequent calculus (Gentzen-style) formulation. They prove a restricted form of the Hauptsatz for the sequent calculus formulation (proofs do not normalize; they show that types are invariant under a restricted class of "parallel" reductions). They present a model for their type system as a collection of subsets of D_∞, Scott's model of the untyped lambda-calculus. Their model relies on the approximation properties of this model alone, not, as in the work of the earlier authors, on a complete metric space structure imposable on the denotations of the types.

To read this paper, one should already probably be familiar with type inference systems and their application in computer science (specifically in the semantics of programming languages), although there is a discussion of earlier work and the direction of applications in the introduction. The proof of the modified Hauptsatz and the construction of the model and verification of its properties are carried out in considerable detail (there is a lot of detail because of the number of type constructors allowed); one lemma in the Hauptsatz proof has its proof postponed to an appendix. The reviewer found the paper readable (though further readings would be required to check every detail of the proofs) and interesting. The reviewer thinks that the system of this paper would be nice for studying the interactions between many different type constructions coexisting in the same system, as the authors suggest. Reviewed by M. Randall Holmes.

Chen, K. and Odersky, M. (1994). A type system for a lambda calculus with assignments. In *Theoretical aspects of computer software (Sendai, 1994)*, pages 347–364. Springer, Berlin.

[MR1296682] This item will not be reviewed individually. For details of the collection in which this item appears see 95d:68006.

Coquand, T. (1994). A new paradox in type theory. In *Logic, methodology and philosophy of science, IX (Uppsala, 1991)*, pages 555–570. North-Holland, Amsterdam.

[MR96h:03034] The author presents a new paradox for type theory by proving inconsistent the system of polymorphic higher-order logic, an extension of minimal higher-order logic, which is provably consistent.

Minimal higher-order logic is defined as a simply typed lambda calculus with one basic type o and function types of the form $\alpha \to \beta$, where there are exactly two term constants, $\Rightarrow$ of type $o \to o \to o$ and, for each type τ, a constant $\forall_\tau$ of type $(\tau \to o) \to o$. A proposition is interpreted as a term of type o. The rules are the standard natural deduction introduction and elimination rules for $\Rightarrow$ (implication) and $\forall_\tau$ (typed universal quantifier). Polymorphic higher-order logic is defined the same way except that the types are taken to be the types of polymorphic second-order lambda calculus. The system is very expressive: the author shows this by showing that if natural numbers, their type, and Leibniz equality are defined as usual in second- and higher-order typed lambda-calculus, then it is possible to prove the Peano postulate that says that zero is not the successor of any number.

The proof that polymorphic higher-order logic is inconsistent follows the argument by J. C. Reynolds [in Semantics of data types (Valbonne, 1984), 145–156, Lecture Notes in Comput. Sci., 173, Springer, Berlin, 1984; MR 86e:03016] by constructing what amounts to an isomorphism between a set and its power set, which is impossible by Cantor's theorem. The author actually refers to the intuitionistic proof of Cantor's theorem as presented by A. M. Pitts ["Non-trivial power types can't be subtypes of polymorphic types", in Proceedings of the 4th Annual Symposium on Logic in Computer Science (Pacific Grove, CA, 1989), 6–13, IEEE Comput. Sci. Press, Washington, DC, 1989; Zbl 716.03005].

This result is then applied to "impredicative type theory" (which the author, the reviewer, and others have previously called the "calculus of constructions") to show that excluded middle and the principle of definition descriptions implies the principle of proof irrelevance. The latter principle says that any two terms which have the same small type are equal in the sense of Leibniz equality.

The paper concludes with a discussion of related results and problems. For the entire collection see MR 96a:03004. Reviewed by J. P. Seldin.

Geuvers, H. (1994). Conservativity between logics and typed λ calculi. In *Types for proofs and programs (Nijmegen, 1993)*, pages 79–107. Springer, Berlin.

[MR95h:03028] The eight systems of typed λ-calculi inside "Barendregt's cube" correspond, via the Curry-Howard embedding, to as many intuitionistic logical systems. The embedding associates, with each provable formula in the logical system, an inhabited type in the corresponding typed λ-theory. For the simply typed λ-calculus and the second-order polymorphic λ-calculus, the embedding is complete, in the sense that each inhabited type corresponds to a provable formula (the Curry-Howard embedding is then called an isomorphism). Berardi and Geuvers observed that for the Calculus of Constructions, the highest system in Barendregt's cube, the Curry-Howard embedding is not complete. In this interesting and well-written paper the author studies the question of conservativity between systems of typed λ-calculi inside Barendregt's cube, namely the question of whether new type-forming operations can create inhabitants of previously empty types. On the logical side, at least when the embedding is complete, this corresponds to the question of whether one logical system is conservative over a smaller one, in the usual sense that a formula taken from the smaller system which is provable in the larger system is already provable in the smaller one. Berardi, Paulin, Ruys and Verschuren had also studied this question, obtaining various partial results. In this paper the author completes the picture, showing that conservativity always holds except in the case of the Calculus

of Constructions over its second-order subsystem λP2. In one case, the conservativity of $\lambda\omega$ over $\lambda 2$, the proof is semantical. The author exploits the Curry-Howard isomorphism and establishes the conservativity using an algebraic semantics for the corresponding intuitionistic systems. For the entire collection see MR 95b:68023. Reviewed by Alessandro Berarducci.

Giannini, P. and Ronchi Della Rocca, S. (1994). A type inference algorithm for a stratified polymorphic type discipline. *Inform. and Comput.*, 109(1-2):115–173. International Conference on Theoretical Aspects of Computer Software (TACS '91) (Sendai, 1991).
[MR1269374] This item will not be reviewed individually. For details of the collection in which this item appears see 94j:68004.

Hayashi, S. (1994a). Logic of refinement types. In *Types for proofs and programs (Nijmegen, 1993)*, pages 108–126. Springer, Berlin.
[MR95i:03028] The system ATTT, introduced by the author in an earlier paper, is a conservative extension of Girard-Reynolds' second-order polymorphic lambda-calculus F. Its main feature is the notion of refinement type, first introduced in a sligthly different form by Freeman and Pfenning. A refinement type of type A is intended to represent a subset of the ordinary type A. Refinement types are closed under unions, intersections and other formation rules (in the spirit of Martin-Lof type theory) and they form a Heyting algebra, thus allowing the interpretation of second-order logic via techniques analogous to those of categorical logic. It is shown that every intuitionistically provable second-order formula is provable in ATTT and it is conjectured that the converse holds (it does for first-order formulas). This interpretation of logic does not keep any computational information and is compatible with classical logic. Realizability interpretations of constructive logic are also represented naturally in ATTT. For the entire collection see MR 95b:68023. Reviewed by Alessandro Berarducci.

Hayashi, S. (1994b). Singleton, union, and intersection types for program extraction. *Inform. and Comput.*, 109(1-2):174–210. International Conference on Theoretical Aspects of Computer Software (TACS '91) (Sendai, 1991).
[MR95m:68115] Two systems for typed lambda calculus are defined and discussed in this article: ATT, an impredicative type theory with implicit typing, and ATTT, a system similar to ATT, but presented with so-called "refinement types". For both systems the meta-theory is discussed in quite some detail and it is shown how these systems can be used for extracting programs from constructive proofs.

The type constructions that are allowed in ATT are arrow, (big) intersection, (big) union and singleton types. As the system has implicit typing, only the application of arrow rules is made explicit in the terms (by abstraction and application). However, many well-known constructions from explicit typing are definable, like the Π, the Σ and the subset type constructor: for A and B types (B possibly containing a variable x of type A), define $\Pi x{\in}A.B := \bigwedge x{\in}A.(\{x\}_A \to B)$, the type of terms f such that for all $a \in A$, $fa \in B[a/x]$, define $\Sigma x{\in}A.B := \bigvee x{\in}A.(\{x\}_A \times B)$, the type of terms $\langle a, p\rangle$ with $a \in A$ and $p \in B[a/x]$, and define $\{x{\in}A|B\} := \bigvee x{\in}A. \bigvee y{\in}B\{x\}_A$, the type of terms $a \in A$ such that there exists a term of type $B[a/x]$. Here, $\{x\}_A$ denotes

the singleton type that contains just x; the product-type constructor $\times$ is definable with the expected introduction and elimination rules. Comparing the definition of the Σ-type and the subset-type, one can observe that in ATT information can be hidden (by using the $\bigvee$-constructor) or can be made explicit (by using the $\times$-construction). The idea is that, in a specification, some parts bear computational content (and these should be made explicit in the term) and some parts bear only logical content (which does not have to made explicit in the term).

The system ATT is translated to a system called ibMPS, a variant of a system defined by MacQueen, Plotkin and Sethi. This translation shows that the terms of ATT are strongly normalizable and it also shows how to extract programs from proofs. For example, let $f \in \bigwedge x{\in}A.(\{x\}_A \to \bigvee y{\in}B.\bigvee z{\in}C\{y\}_B)$. Then the term f itself is the untyped term (program) that satisfies the specification $\forall a \in A[C[fa/y]]$. Due to the use of the union type, only the information about the mapping from A to B is explicit in f. This corresponds to the fact that the translation to ibMPS yields $f \in A \to B$. (The derivation in ATT that "proves" that f is of the above-mentioned type is in fact the proof that f satisfies the specification.) The system ATTT is similar to ATT, but now augmented with refinement types. The idea here is that, if $B \in$ type, then refine(B) is the kind of "subtypes" of B (the types A such that, if $M \in A$, then $M \in B$). In the syntax, the arrow and the Π-constructor can be used for both types and refinement types (A such that $A \in$ refine(B)). The (big) union, (big) intersection and singleton type constructions can only be used for refinement types. So we have, e.g., $\Pi x \in \text{type}.A$ for both $A \in$ type and $A \in$ refine(B) but $\bigvee x \in \text{type}.A$ only for $A \in$ refine(B). The system ATTT can be translated to the polymorphic lambda calculus. Just as for ATT, this shows how to extract programs from proofs and it also proves strong normalization for ATTT.

The system ATT does not have decidability of typing. In contrast, derivability of $\Gamma \vdash M\colon A$ in ATTT is decidable unless A is a refinement type. This conforms with the intuitive understanding that a refinement type represents a specification and one can not expect it to be decidable whether a program M meets a specification A. If A is not a refinement type (in ATTT), then M has the type A in a programming-language sense. This can be checked at compile time, so this should be decidable. For the entire collection see MR 94j:68004. Reviewed by Herman Geuvers.

Kamareddine, F. and Nederpelt, R. (1994). A unified approach to type theory through a refined λ-calculus. *Theoret. Comput. Sci.*, 136(1):183–216. Selected papers of the Meeting on the Mathematical Foundations of Programming Semantics (MFPS '92), Part II (Oxford, 1992).

[MR95m:03031] In a series of papers the authors have proposed an essentially new notation for typed λ-calculi. The main advantages of their approach are the following. (1) Types and terms are treated in an uniform way; for example, types also β-reduce. (2) Some definitions are easier and more natural, but there are also new definitions; for example, substitution now becomes an explicit operation via substitution operators. Moreover, this notation allows a notion of generalized reduction, which seems difficult to translate in the classical λ-calculus.

The present paper shows how various existing systems can be expressed in a uniform way in the proposed setting. In particular, the fine structure of the typing relation is examined in depth.

I think that the small effort of learning this new notation will be greatly rewarded by the expressivity, generality and uniformity so obtained.

The paper is well written and pleasant to read. For the entire collection see MR 95h:68002. Reviewed by Mariangiola Dezani-Ciancaglini.

Nederpelt, R. P. (1994a). Strong normalization in a typed lambda calculus with lambda structured types. In *Selected papers on Automath*, pages 389–468. North-Holland, Amsterdam.

[**MR1429415**] This item will not be reviewed individually. For details of the collection in which this item appears see 98g:03038.

Nederpelt, R. P. (1994b). Type systems—basic ideas and applications. In *Selected papers on Automath*, pages 229–247. North-Holland, Amsterdam.

[**MR1429404**] This item will not be reviewed individually. For details of the collection in which this item appears see 98g:03038.

Ong, C.-H. L. and Ritter, E. (1994). A generic strong normalization argument: application to the calculus of constructions. In *Computer science logic (Swansea, 1993)*, pages 261–279. Springer, Berlin.

[**MR96g:03011**] A way to obtain uniform (or, generic) proofs of normalization for various (higher-order, polymorphic, dependent) type theories is to interpret them in a suitably rich universe of untyped strongly normalizing lambda terms. The categorical construction of such universes inside certain realizability toposes was initiated by J. M. E. Hyland and the first author [in Typed lambda calculi and applications (Utrecht, 1993), 179–194, Lecture Notes in Comput. Sci., 664, Springer, Berlin, 1993; see MR 94b:68025], and applied to higher-order polymorphic type theory. In the present paper this approach is extended to include higher order dependent type theory (specifically, the calculus of constructions). This requires an appropriately well-behaved "stripping map", removing type information from terms. The definition of this map is the main technical contribution of the paper. For the entire collection see MR 95j:68009. Reviewed by Bart Jacobs.

Plotkin, G. (1994). A semantics for static type inference. *Inform. and Comput.*, 109(1-2):256–299. International Conference on Theoretical Aspects of Computer Software (TACS '91) (Sendai, 1991).

[**MR95j:03026**] In a type assignment system for untyped lambda calculus, which is complete with respect to any "decent" semantics, an equality rule assuring that interconvertible terms have the same type would be valid. Unfortunately this rule is not admissible for many systems, and its addition, although it gives the desired completeness result, makes the systems undecidable. In particular, this is true for the basic Curry system, which can be made the basis of static type inference algorithms that calculate the type of a term without any reduction.

The author splits the equality rule up into two parts: a reduction rule and an expansion rule. The former states that, if a term P reduces to a term Q, any type for P is also a type for Q; the latter states, symmetrically, that, if P reduces to Q, any type for Q is a type for P.

In type systems in which the subject reduction theorem holds, the reduction rule is admissible. So Curry's system, for example, can be semantically analyzed and proved complete for a class of models that (possibly) do not model conversion by equality, but that model reduction by inequality. These models are models of reduction: partially ordered sets in which types are interpreted as upper-closed sets in the ordering.

The completeness of some type assignment systems (Curry, polymorphic and intersection systems) with respect to different notions of semantics is analyzed by using two kinds of reduction models: a term model and a type-expression model, which can be seen as the natural version of the intersection filter model [H. P. Barendregt, M. Coppo and M. Dezani-Ciancaglini, J. Symbolic Logic 48 (1983), no. 4, 931–940 (1984); MR 85j:03014]. For the entire collection see MR 94j:68004. Reviewed by Ines Margaria.

Severi, P. and Poll, E. (1994). Pure type systems with definitions. In *Logical foundations of computer science (St. Petersburg, 1994)*, pages 316–328. Springer, Berlin.

[MR95k:68131] Summary: "In this paper, an extension of pure type systems (PTSs) with definitions is presented. We prove this extension preserves many of the properties of PTSs. The main result is a proof that for many PTSs, including the calculus of constructions, this extension preserves strong normalisation." For the entire collection see MR 95j:03003.

Takahashi, M., Akama, Y., and Hirokawa, S. (1994). Normal proofs and their grammar. In *Theoretical aspects of computer software (Sendai, 1994)*, pages 465–493. Springer, Berlin.

[MR95j:68099] Summary: "First we give a grammatical (or equational) description of the set $\{M \text{ in normal form} | \Gamma \vdash M : A\}$ for a given basis Γ and a given type A in the simple type system, and give some applications of the description. Then we extend the idea to systems in λ-cube and more generally to normalizing pure type systems. The attempt results in derived (or 'macro') rules, the totality of which is sound and complete for type assignments of normal terms. A feature of the derived rules is that they reflect the syntactic structure of legal terms in normal form, and thus they may give us a more global view than the original definition of the systems." For the entire collection see MR 95d:68006.

van Bakel, S., Liquori, L., Ronchi Della Rocca, S., and Urzyczyn, P. (1994). Comparing cubes. In *Logical foundations of computer science (St. Petersburg, 1994)*, pages 353–365. Springer, Berlin.

[MR95m:03029] Summary: "We study the cube of type assignment systems, as introduced in [P. Giannini, F. Honsell and S. Ronchi Della Rocca, Fund. Inform. 19 (1993), no. 1-2, 87–125; MR 95f:03024]. This cube is obtained from Barendregt's typed λ-cube via a natural type erasing function E, that erases type information from terms. We prove that the systems in the former cube enjoy good computational properties, such as subject reduction and strong normalization. We study the relationship between the two cubes, which leads to some unexpected results in the field of systems with dependent types." For the entire collection see MR 95j:03003.

van Benthem Jutting, L. S., McKinna, J., and Pollack, R. (1994). Checking algorithms for pure type systems. In *Types for proofs and programs (Nijmegen, 1993)*, pages 19–61. Springer, Berlin.
[MR96d:68134]

The authors discuss typechecking algorithms for the class of pure type systems. The latter is a collection of typed lambda calculi that have a similar syntactic format, the only type-forming operation being the Π-type-construction and the only reduction being the β-reduction. The interest of the results presented in this paper lies in the fact that in many implementations of type theory, the formal system that is implemented is a pure type system. An—at least theoretically—important tool for the use of implementations of pure type systems is the decidability of typechecking. More practical questions that arise are how to find a feasible algorithm for typechecking and to what extent the implemented system is faithful to the underlying (formal) pure type system. Both these questions are addressed in this paper.

The question of typechecking (does term a have type A in the context Γ?) is reduced to a type-synthesis problem (compute a type A_0 for a in context Γ, if it exists, and a type for A in context Γ, if it exists) and an equality problem (check whether A is convertible to A_0). A large part of the article is devoted to finding a good (feasible) type-synthesis algorithm. The main technique for finding such a type-synthesis algorithm is to modify the syntax of the type system in order to find a so-called "syntax directed" set of rules. A set of derivation rules is "syntax directed" if for every context Γ and term a there is at most one rule with a conclusion $\Gamma \vdash a : A$ and moreover, if such a rule exists, then it produces exactly one type A for a. (If this extra proviso does not always hold, the rules are called "nearly syntax directed".) The key problem in finding a (nearly) syntax directed set of rules for a pure type system is to show the equivalence with the original set of rules.

The authors show that for two interesting classes of pure type systems (the "semi-full" ones and the "functional" ones), a nearly syntax directed version of the set of rules can be given. (Using that, they describe a type-synthesis algorithm.) Motivated by this, they show how to give a nearly syntax directed set of rules for general pure type systems. As a matter of fact, they describe a nearly syntax directed system for every natural number n, such that the union of all these systems is the original pure type system. For the entire collection see MR 95b:68023. Reviewed by Herman Geuvers.

Venneri, B. (1994). Intersection types as logical formulae. *J. Logic Comput.*, 4(2):109–124.

[MR95b:68068] Summary: "The aim of this paper is to investigate, using the Curry-Howard isomorphism approach, a logical characterization of the intersection-type discipline. First a novel formulation of intersection type inference for combinatory logic is presented which is equivalent to the original version of the system, while the intersection operator is no longer dealt with as a proof-theoretic connective. Then a Hilbert-style axiomatization is defined and proved to totally parallel intersection-derivability in such a way that the inhabited intersection-types are precisely the provable formulae in the logic system."

Yokouchi, H. (1994). f-semantics for type assignment systems. *Theoret. Comput. Sci.*, 129(1):39–77.

[**MR95k:68133**] The subject of this paper is the completeness of some type assignment systems for terms of untyped λ-calculus, with respect to the F-semantics (F-completeness).

The first system investigated is the polymorphic type assignment system, for which the problem of F-completeness was open [J. C. Mitchell, Inform. and Comput. 76 (1988), no. 2-3, 211–249; MR 91c:68021]. The author shows that polymorphic type assignment is not F-complete and introduces two new additional rules that give F-completeness. The proof of F-completeness is based on the construction of an F-model, obtained by modifying Hindley's term model definition. To this aim an equivalence relation $=_{\beta f}$, such that $M =_{\beta f} \lambda x.Mx$ iff M has functional types, is introduced.

The definition of F-semantics is extended also to polymorphic type assignment systems without the (Eq_β) rule; it is proved that the addition of a new rule to a polymorphic type assignment system without (Eq_β) makes it F-complete; the proof is done by using F-models of β-reduction (semi-models).

Finally the natural translation of the results about polymorphic type assignment systems to an intersection type assignment system gives new insight into the problem of F-completeness for the intersection discipline, previously investigated by M. Dezani-Ciancaglini and I. Margaria [Theoret. Comput. Sci. 45 (1986), no. 2, 121–157; MR 88k:68058]. The relations between the present results and the results of Dezani-Ciancaglini and Margaria are studied. Reviewed by M. Zacchi.

Aspinall, D. (1995). Subtyping with singleton types. In *Computer science logic (Kazimierz, 1994)*, pages 1–15. Springer, Berlin.

[**MR1471214**] This item will not be reviewed individually. For details of the collection in which this item appears see 98d:68013.

Barbanera, F., Dezani-Ciancaglini, M., and de'Liguoro, U. (1995). Intersection and union types: syntax and semantics. *Inform. and Comput.*, 119(2):202–230.

[**MR96c:68111**] The investigation of various kinds of type constructors is important because typed programming languages have many advantages. This paper is a lambda calculus and proof-theoretic investigation of the problematic but useful type constructors union and intersection. These type constructors are problematic because the usual "subject-reduction" property and "formulas as types" analogy fail. In Section 1 a natural deduction and equivalent sequent presentation of a basic type assignment system is given; in Section 2 the problems of intersection and union types are mollified; in Sections 3 and 4 the proof theory of various type theories is investigated; in Section 5 various interesting semantics are introduced and the soundness and completeness of type theories are proved; in Section 6 related work of Pierce, Abramsky and others is discussed. Reviewed by B. H. Mayoh.

Barthe, G. (1995). Extensions of pure type systems. In *Typed lambda calculi and applications (Edinburgh, 1995)*, pages 16–31. Springer, Berlin.

[**MR1477972**] This item will not be reviewed individually. For details of the collection in which this item appears see 98e:03020.

Bellucci, R., Abadi, M., and Curien, P.-L. (1995). A model for formal parametric polymorphism: a PER interpretation for system ∇. In *Typed lambda calculi and applications (Edinburgh, 1995)*, pages 32–46. Springer, Berlin.

[MR1477973] This item will not be reviewed individually. For details of the collection in which this item appears see 98e:03020.

Benton, P. N. (1995). Strong normalisation for the linear term calculus. *J. Funct. Programming*, 5(1):65–80.

[MR97b:03073] Assignments of typed terms to derivations in intuitionistic linear logic have been given by Benton, G. Bierman, V. de Paiva and M. Hyland [in Typed lambda calculi and applications (Utrecht, 1993), 75–90, Lecture Notes in Comput. Sci., 664, Springer, Berlin, 1993; MR 94g:03116]. By introducing the additives, the author presents a new proof of strong normalization for the resulting linear term calculus LTC, with the exclusion of commuting conversions. The novelty is methodological: one interprets (types and) terms t of LTC by (types and) terms t° of second-order polymorphic λ-calculus, in such a way that if t reduces to t', then t° reduces to $(t')^\circ$. Thus strong normalization follows from the well-known proof of Girard. The key point is in the translation of exponential types: the author informally argues that here the full power of the target system is really needed. Guided by the linear logic equivalence of $!A$ and $1\&A\&(!A\otimes !A)$ to define $(!A)^\circ$, one first considers (roughly speaking) the type $\Phi(X) = I \times A^\circ \times (X \times X)$ and then defines $(!A)^\circ$ to be the greatest fixed point of $\Phi(X)$.

Bono, V. and Liquori, L. (1995). A subtyping for the Fisher-Honsell-Mitchell lambda calculus of objects. In *Computer science logic (Kazimierz, 1994)*, pages 16–30. Springer, Berlin.

[MR1471215] This item will not be reviewed individually. For details of the collection in which this item appears see 98d:68013.

Coppo, M. and Giannini, P. (1995). Principal types and unification for simple intersection type systems. *Inform. and Comput.*, 122(1):70–96.

[MR96m:68103] Summary: "In this paper we introduce a decidable restriction of the intersection type discipline. The restriction is based on a uniform principle rather than on a notion of rank. For this system we give a type checking algorithm that we prove to be sound and complete. An extended language of types with labelled intersections and intersection schemes is needed to describe the principal type of terms which turns out to be unique modulo a set of equations between labels. The unification algorithm for labelled types, which extends first order unification, could be used as a basis to define inference algorithms for other (more powerful) systems with intersection."

de Bruijn, N. G. (1995). On the roles of types in mathematics. In *The Curry-Howard isomorphism*, pages 27–54. Academia, Louvain.

[MR97c:03039] This article presents the "propositions as types" idea (though the author expresses reservations about this terminology, preferring "proofs as objects") through a description of a fragment of the author's Automath proof checker.

The article begins with an engaging discussion of the contrast between the untyped foundations of mathematics to which most of us officially subscribe (the standard set theory ZFC) and the important role that types play in our actual mathematical practice. The reviewer notes that the Mizar project also found types to be indispensable in the development of a proof checker intended to handle actual mathematical practice.

The discussion which follows this is intended to indicate how "the idea of treating proofs as objects is almost forced upon anyone who tries to explain mathematics to a machine" (quoting the article). The illustration of this via a fragment of Automath (the subset without λ-calculus) should not be allowed to daunt the reader who is interested mainly in mathematical or philosophical rather than computer science issues; the reviewer suspects that the author would think that dropping the clause "to a machine" from the quotation above would be appropriate.

The reviewer recommends the article highly, and also the book in which it appears, which is a collection of articles on the same topic. For the entire collection see MR 96h:03041.

Di Cosmo, R. (1995). Second order isomorphic types: a proof-theoretic study on second order λ-calculus with surjective pairing and terminal object. *Inform. and Comput.*, 119(2):176–201.

[MR97a:03022] Summary: "We investigate invertible terms and isomorphic types in the second-order lambda calculus extended with surjective pairs and a terminal (or Unit) type. These two topics are closely related: on one hand, the study of invertibility is a necessary tool for the characterization of isomorphic types; on the other hand, we need the notion of isomorphic types to study the typed invertible terms. The result of our investigation is twofold: we give a constructive characterization of the invertible terms, extending previous work by Dezani and Bruce-Longo, and a decidable equational theory of the isomorphisms of types which hold in all models of the calculus, which is a conservative extension to the second-order case of the results previously achieved for the case of first-order typed calculi. Via the Curry-Howard correspondence, this work also provides a decision procedure for strong equivalence of formulae in second-order intuitionistic positive propositional logic; this procedure is suitable for searching for equivalent proofs in automated deduction systems."

Dowek, G. (1995). Lambda-calculus, combinators and the comprehension scheme. In *Typed lambda calculi and applications (Edinburgh, 1995)*, pages 154–170. Springer, Berlin.

[MR98f:03011] Summary: "The presentations of type theory based on a comprehension scheme, a Skolemized comprehension scheme and λ-calculus are equivalent, both in the sense that each one is a conservative extension of the previous one and that each one can be coded in the previous one preserving provability. A similar result holds for set theory." For the entire collection see MR 98e:03020.

Gallier, J. (1995). Proving properties of typed λ-terms using realizability, covers, and sheaves. *Theoret. Comput. Sci.*, 142(2):299–368. Selected

papers of the Fifth International Conference on Rewriting Techniques and Applications (Montreal, PQ, 1993).
[MR96h:03036] Summary: "The main purpose of this paper is to take apart the reducibility method in order to understand how its pieces fit together, and in particular, to recast the conditions on candidates of reducibility as sheaf conditions. There has been a feeling among experts on this subject that it should be possible to present the reducibility method using more semantic means, and that a deeper understanding would then be gained. This paper gives mathematical substance to this feeling, by presenting a generalization of the reducibility method based on a semantic notion of realizability which uses the notion of a cover algebra (as in abstract sheaf theory). A key technical ingredient is the introduction of a new class of semantic structures equipped with preorders, called pre-applicative structures. These structures need not be extensional. In this framework, a general realizability theorem can be shown. Kleene's recursive realizability and a variant of Kreisel's modified realizability both fit into this framework. We are then able to prove a metatheorem which shows that if a property of realizers satisfies some simple conditions, then it holds for the semantic interpretations of all terms. Applying this theorem to the special case of the term model yields a general theorem for proving properties of typed λ-terms, in particular, strong normalization and confluence. This approach clarifies the reducibility method by showing that the closure conditions on candidates of reducibility can be viewed as sheaf conditions. The above approach is applied to the simply-typed λ-calculus (with types $\rightarrow$, $\times$, $+$, and $\perp$), and to the second-order (polymorphic) λ-calculus (with types $\rightarrow$ and $\forall^2$), for which it yields a new theorem." For the entire collection see MR 96b:68005.

Geuvers, H. (1995). The calculus of constructions and higher order logic. In *The Curry-Howard isomorphism*, pages 139–191. Academia, Louvain.
[MR97f:03011] Among the typed lambda calculi, the Calculus of Constructions (CC) of Coquand and Huet occupies a major place. It is the "top element" in Barendregt's Cube, since it embodies three important generalizations of Church's simply typed lambda calculus, enabling polymorphism, predication and constructor-building. CC therefore combines the generalisations of Girard (system F_ω) with dependent typed lambda calculi in the style of de Bruijn (Automath) and Martin-Löf. CC can be used for higher-order intuitionistic predicate logic, in which proofs are interpreted as lambda terms and propositions as types. This interpretation is known as the Curry-Howard or formulae-as-types embedding of predicate logic into type theory.

This paper describes how higher-order predicate logic (HOPL) can be represented by a typed lambda calculus (λHOPL), or, alternatively, by a pure type system (λPREDω), both of which are faithful with respect to the Curry-Howard embedding. Next, CC is defined, followed by the formulae-as-types embedding from λPREDω into CC. An important question is whether this embedding is complete (soundness is easy). Counterexamples of the author and, independently, Berardi, show that this is not the case: in CC there are more "propositions" (types) for which an inhabitant (a term of that type) can be derived than there are propositions provable in higher-order predicate logic. (An intuitive reason is that "information is lost", because the embedding maps both propositions and domains of predicate logic to types.) However, CC is still consistent. Moreover, when CC is used as a system for higher-order reasoning about a mildly restricted set of inductive data types, then completeness holds. This paper, which gives a detailed account of the questions described above, with discussion and examples, ends with an overview of existing extensions and variations of CC. For the entire collection see MR 96h:03041. Reviewed by R. P. Nederpelt.

Ghani, N. (1995). $\beta\eta$-equality for coproducts. In *Typed lambda calculi and applications (Edinburgh, 1995)*, pages 171–185. Springer, Berlin.
[MR1477982] This item will not be reviewed individually. For details of the collection in which this item appears see 98e:03020.

Ghelli, G. (1995). Divergence of $F_{\leq}$ type checking. *Theoret. Comput. Sci.*, 139(1-2):131–162.
[MR96m:68105] Summary: "System $F_{\leq}$ is an extension of second-order typed lambda calculus, where a subtype hierarchy among types is defined, and bounded second-order lambda abstraction is allowed. This language is a basis for much of the current research on integration of typed functional languages with subtypes and inheritance. An algorithm to perform type checking for $F_{\leq}$ expressions has been known since the language Fun was defined. The algorithm has been proved complete, by the author and P.-L. Curien, which means that it is a semi-decision procedure for the type-checking problem. In this paper we show that this algorithm is not a decision procedure, by exhibiting a term which makes it diverge. This result was the basis of Pierce's proof of undecidability of typing for $F_{\leq}$. We study the behavior of the algorithm to show that our diverging judgement is in some sense contained in any judgement which makes the algorithm diverge. On the basis of this result, and of other results in the paper, we claim that the chances that the algorithm will loop while type-checking a 'real program' are negligible. Hence, the undecidability of $F_{\leq}$ type-checking should not be considered as a reason to prevent the adoption of $F_{\leq}$ as a basis for defining programming languages of practical interest. Finally, we show the undecidability of an important subsystem of $F_{\leq}$."

Hermida, C. and Jacobs, B. (1995). Fibrations with indeterminates: contextual and functional completeness for polymorphic lambda calculi. *Math. Structures Comput. Sci.*, 5(4):501–531. Fifth Biennial Meeting on Category Theory and Computer Science (Amsterdam, 1993).
[MR97k:03016] Summary: "Lambek used categories with indeterminates to capture explicit variables in simply typed λ-calculus. He observed that such categories with indeterminates can be described as Kleisli categories for suitable comonads. They account for 'functional completeness' for Cartesian (closed) categories.

"Here we refine this analysis, by distinguishing 'contextual' and 'functional' completeness, and extend it to polymorphic λ-calculi. Since the latter are described as certain fibrations, we are led to consider indeterminates, not only for ordinary categories, but also for fibred categories. Following a 2-categorical generalisation of Lambek's approach, such fibrations with indeterminates are presented as 'simple slices' in suitable 2-categories of fibrations: more precisely, as Kleisli objects. This allows us to establish contextual and functional completeness results for some polymorphic calculi." For the entire collection see MR 96h:68007. Reviewed by R. P. Nederpelt.

Hofmann, M. (1995). A simple model for quotient types. In *Typed lambda calculi and applications (Edinburgh, 1995)*, pages 216–234. Springer, Berlin.

[MR1477985] This item will not be reviewed individually. For details of the collection in which this item appears see 98e:03020.

Holmes, M. R. (1995). Untyped λ-calculus with relative typing. In *Typed lambda calculi and applications (Edinburgh, 1995)*, pages 235–248. Springer, Berlin.

[MR98f:03013] Summary: "A system of untyped λ-calculus with a restriction on function abstraction using relative typing analogous to the restriction on set comprehension found in Quine's set theory NF ('New Foundations') is discussed. The author has shown elsewhere that this system is equiconsistent with Jensen's NFU ('New Foundations' with urelements) + Infinity, which is in turn equiconsistent with the simple theory of types with infinity. The definition of the system is given and the construction of a model is described. A semantic motivation for the stratification criterion for function abstraction is given, based on an abstract model of computation. The same line of semantic argument is used to motivate an analogy between the notion of 'strongly Cantorian set' found in NF and the notion of 'data type'; an implementation of absolute types as domains of retractions with strongly Cantorian ranges is described. The implementation of these concepts in a theorem prover developed by the author is sketched." For the entire collection see MR 98e:03020.

Hurkens, A. J. C. (1995). A simplification of Girard's paradox. In *Typed lambda calculi and applications (Edinburgh, 1995)*, pages 266–278. Springer, Berlin.

[MR1477988] This item will not be reviewed individually. For details of the collection in which this item appears see 98e:03020.

Kozen, D., Palsberg, J., and Schwartzbach, M. I. (1995). Efficient recursive subtyping. *Math. Structures Comput. Sci.*, 5(1):113–125.

[MR96j:68024] Summary: "Subtyping in the presence of recursive types for the λ-calculus was studied by R. M. Amadio and L. Cardelli [in Proceedings of the Eighteenth Annual ACM Symposium on Principles of Programming Languages, 104–118, ACM Press, New York, 1991]. They showed that the problem of deciding whether one recursive type is a subtype of another is decidable in exponential time. In this paper we give an $O(n^2)$ algorithm. Our algorithm is based on a simplification of the definition of the subtype relation, which allows us to reduce the problem to the emptiness problem for a certain finite automaton with quadratically many states.

"It is known that equality of recursive types and the covariant Böhm order can be decided efficiently by means of finite automata, since they are just language equality and language inclusion, respectively. Our results extend the automata-theoretic approach to handle orderings based on contravariance."

Kristiansen, L. and Normann, D. (1995). Semantics for some constructors of type theory. In *Proceedings of the 2nd Gauss Symposium. Conference*

A: Mathematics and Theoretical Physics (Munich, 1993), pages 201–224, Berlin. de Gruyter.
[MR98g:03017] In this paper the semantics of two basic constructors of Martin-Löf type theory (the dependent sum and dependent product) is discussed. These constructors were introduced for developing a constructive variant of set theory.

The author considers the hierarchy for describing subdomains of an universal domain. Two versions of a universal domain are investigated. One of them is called a qualitative domain, and the second one is called a Scott domain. In both cases, the set of Boolean values and the set of natural numbers are the base types. For the entire collection see MR 96d:00027. Reviewed by Feodor Alexandrovich Murzin.

Longo, G. (1995). Parametric and type-dependent polymorphism. *Fund. Inform.*, 22(1-2):69–92.
[MR96g:68071] Under the pragmatic point of view, the classification of polymorphism is done in accordance with the dependence of functions on the kind of types. This paper contains a detailed discussion of polymorphism based on a sound theoretical background given in several papers. As a consequence, the reader needs to be familiar with the content of the cited papers in order to understand completely the interesting discussion in which open problems play an important role.

At first, an introduction on polymorphic λ-calculus is given. Then, a short informal discussion on types as invariants and types as sets follows. A relational parametricity is presented in Chapter 3 which follows the concept of types as invariants. Chapter 4, however, shows that the understanding of parametricity as "effective" parametricity is incompatible with parametricity as invariance. In Chapter 5, PER models are used to give an understanding of "uniformity" of polymorphic terms. PER is a category where partial equivalence relations are the objects and morphisms are defined by Kleene's application.

Finally, the contradiction between the negative results in theory and the practical use of methods is considered. It remains an aim to embed "ad hoc" polymorphism into a sound mathematical frame. An approach is presented as a "preliminary attempt". Reviewed by Peter Bachmann.

Margaria, I. and Zacchi, M. (1995). Principal typing in a $\forall\wedge$-discipline. *J. Logic Comput.*, 5(3):367–381.
[MR96h:03037] Summary: "In this paper we study the problem of finding a principal typing for a λ-term in a polymorphic intersection type assignment system. Following the approach of Coppo et al., Ronchi and Venneri, the principal type is proved to exist for any λ-term with a finite set of approximants. In the same way the existence of the principal type is shown for $BCK\lambda$-terms in a polymorphic type assignment system."

Mo, S. K. (1995). Operators and the comprehension procedure—a complete type-free logical calculus. *Chinese Ann. Math. Ser. A*, 16(3):296–302.
[MR97c:03031] It is shown that higher-order functions are the same as operators in nature (Theorem 4.2). Every operator possesses three kinds of arguments: companion, guiding and scope. If we replace higher-order functions by operators, then the functions and operators themselves used as arguments should be replaced by their name forms (used as scopes, and the corresponding name variables should be

added as new guiding variables), the latter being formterms, i.e. formulas or terms (Theorem 4.4). Consequently, in any formterms, all occurrences of functions or of operators, except as guiding variables, should be in filled form, never standing in isolation (Theorem 4.4). The guiding (function or operator) variables may be replaced by guiding dependent variables which stand for the name forms of variable functions (and operators), and are formterm variables (Theorem 4.5). Consequently, when we use the comprehensive function $\lambda x A$, we may instead use its filled form $'\lambda x A y$, i.e. $S(y/x)A$; here $S(y/x)A$ means that we substitute everywhere y for x in A. Since we may get rid of λx, we may dispense with the comprehension procedure altogether (Theorem 4.7), and use permanent abbreviations (together with substitution) instead. The type distinction is of no use at all. If we make such a change for the full (higher-order) predicate calculus, we get a (functional) complete type-free logical calculus. Roughly speaking, this system is the full predicate calculus with higher functions replaced by operators, and also is the restricted predicate calculus with elementary operators (including substitution) added, both without the comprehension procedure (replaced by the permanent abbreviations). Since the symbol S of substitution may be eliminated from all formterms, this system is as safe as the restricted predicate calculus. Reviewed by Bai Ying Shen.

Nour, K. (1995). A general type for storage operators. *Math. Logic Quart.*, 41(4):505–514.

[MR96j:03016] This paper introduces various type systems assigning to untyped lambda terms formulas of second-order predicate logic. The main result is that all terms that can be assigned a certain type are storage operators for the Church numerals. The result seems very specific and the author makes no effort to place it in a broader context. Reviewed by Eugenio Moggi.

Palmgren, E. (1995). The Friedman-translation for Martin-Löf's type theory. *Math. Logic Quart.*, 41(3):314–326.

[MR97m:03018] A Friedman-translation is obtained for Martin-Löf's unrestricted basic type theory and some extensions. This basic type theory is the monomorphic type theory ML_0 without universes [cf. E. Palmgren and V. Stoltenberg-Hansen, Ann. Pure Appl. Logic 48 (1990), no. 2, 135–196; MR 91h:03015], but without the iteration type Ω. For these theories a higher type version of Markov's rule holds. The translation itself works on the level of definitional equality, and for the identity types. Reviewed by Wim Ruitenburg.

Terlouw, J. (1995). Strong normalization in type systems: a model theoretical approach. *Ann. Pure Appl. Logic*, 73(1):53–78. A tribute to Dirk van Dalen.

[MR96g:03023] Summary: "Tait's proof of strong normalization for the simply typed λ-calculus is interpreted in a general model-theoretical framework by means of the specification of a certain theory T and a certain model $\mathfrak{A}$ of T. The argumentation is partly reduced to formal predicate logic by the application of certain derivability properties of T. The resulting version of Tait's proof is, within the same

framework, systematically generalized to the calculus of constructions and other advanced type systems. The generalization proceeds along the formal level in the sense that T is modified first and that the new theory partly dictates the subsequent transformation of $\mathfrak{A}$, which is carried out in stages. The uniform and standardized character of the argumentation contributes to its perspicuity."

van Bakel, S. (1995). Intersection type assignment systems. *Theoret. Comput. Sci.*, 151(2):385–435. Thirteenth Conference on Foundations of Software Technology and Theoretical Computer Science (Bombay, 1993).

[**MR96m:68102**] This paper gives a comprehensive overview of intersection type assignment systems for the lambda calculus. It also presents a new system, the essential type assignment system, which has the same properties as the well-known Barendregt-Coppo-Dezani system, but whose typing mechanism is more syntax directed in the sense that the types used are the representatives of equivalence classes of types in the BCD-system. In total five notions of intersection type assignment are presented and compared: the Coppo-Dezani (1980) system, the Coppo-Dezani-Venneri (1981) system, the already mentioned BCD-system (1983), the strict system of Bakel (1992) and the essential system. The latter turns out to be an extension of the strict system and a restriction of the BCD-system. It is a system closed under β-equality and η-reduction. The results presented include the principal type property, an approximation theorem leading to a characterization of terms having a (head) normal form, and a soundness and completeness theorem using a filter semantics. For the entire collection see MR 96f:68004. Reviewed by Alessandro Berarducci.

Wand, M., O'Keefe, P., and Palsberg, J. (1995). Strong normalization with non-structural subtyping. *Math. Structures Comput. Sci.*, 5(3):419–429.

[**MR96m:68022**] Summary: "We study a type system with a notion of subtyping that involves a largest type $\top$, a smallest type $\perp$, atomic coercions between base types, and the usual ordering of function types. We prove that any λ-term typable in this system is strongly normalizing, which solves an open problem of Thatte. We also prove that the fragment without $\perp$ types has strictly fewer terms. This demonstrates that $\perp$ adds power to a type system."

Abadi, M. and Fiore, M. P. (1996). Syntactic considerations on recursive types. In *11th Annual IEEE Symposium on Logic in Computer Science (New Brunswick, NJ, 1996)*, pages 242–252. IEEE Comput. Soc. Press, Los Alamitos, CA.

[**MR1461838**] This item will not be reviewed individually. For details of the collection in which this item appears see 98a:68006.

Altenkirch, T., Hofmann, M., and Streicher, T. (1996). Reduction-free normalisation for a polymorphic system. In *11th Annual IEEE Symposium on Logic in Computer Science (New Brunswick, NJ, 1996)*, pages 98–106. IEEE Comput. Soc. Press, Los Alamitos, CA.

[**MR1461825**] This item will not be reviewed individually. For details of the collection in which this item appears see 98a:68006.

Aspinall, D. and Compagnoni, A. (1996). Subtyping dependent types (summary). In *11th Annual IEEE Symposium on Logic in Computer Science (New Brunswick, NJ, 1996)*, pages 86–97. IEEE Comput. Soc. Press, Los Alamitos, CA.

[**MR1461824**] This item will not be reviewed individually. For details of the collection in which this item appears see 98a:68006.

Barbanera, F. and Fernández, M. (1996). Intersection type assignment systems with higher-order algebraic rewriting. *Theoret. Comput. Sci.*, 170(1-2):173–207.

[**MR98b:68006**] The subject of the paper is interesting and the results obtained are worthwhile in the research area of algebraic functional languages. The authors investigate a type assignment system using intersection types for languages definable by adding first- and higher-order algebraic rewriting to lambda calculus. As a main property, it is proved that (i) the condition that first-order rules are non-duplicating implies strong normalization; (ii) the condition that higher-order rules satisfy the so-called "general schema" implies the completeness property, that is, both strong normalization and confluence.

The novelty of the paper is that these properties concern the type assignment approach, while analogous results in the literature work in the setting of algebraic extensions of typed lambda calculus.

Unfortunately, the authors fail to explain by example the significance for these integrated languages of the type assignment approach, whose advantages are already well known for functional programming. Reviewed by Betti Venneri.

Barthe, G. and Geuvers, H. (1996). Modular properties of algebraic type systems. In *Higher-order algebra, logic, and term rewriting (Paderborn, 1995)*, pages 37–56. Springer, Berlin.

[**MR97g:68111**] Summary: "We introduce the framework of algebraic type systems, a generalisation of pure type systems with higher order rewriting a la Jouannaud-Okada, and initiate a generic study of the modular properties of these systems. We give a general criterion for one system of this framework to be strongly normalising. As an application of our criterion, we recover all previous strong normalisation results for algebraic type systems." For the entire collection see MR 97f:68005.

Bellè, G. (1996). Syntactical properties of an extension of Girard's system F where types can be taken as "generic" inputs. In *Theoretical computer science (Ravello, 1995)*, pages 134–145. World Sci. Publishing, River Edge, NJ.

[**MR1445801**] This item will not be reviewed individually. For details of the collection in which this item appears see 97k:68013.

Bloo, R., Kamareddine, F., and Nederpelt, R. (1996). The Barendregt cube with definitions and generalised reduction. *Inform. and Comput.*, 126(2):123–143.

[**MR97c:03045**] This interesting paper studies the properties of the Barendregt cube extended with definitions and generalized reduction. The use of item notation, which makes more redexes visible and describes nested definitions in a neat way, is essential.

First, the authors add generalized reduction to the cube. The main motivation is that generalized reduction makes it possible to contract redexes more freely than does standard beta-reduction. All the usual properties (Church-Rosser, subtyping, strong normalization) are preserved, but subject reduction fails. This failure is due to a loss of information, which can be recovered by adding definitions. The system obtained enjoys all the standard properties, but only a restricted form of subtyping holds. This is not a real drawback, since the subterms (of a typed term) which cannot be typed are involved. The main feature of the definitions is that type derivations become considerably shorter, without slowing down type-checking algorithms. Moreover, the authors prove the following conservativity result: Whenever a judgment is derivable in an extended theory, it is also derivable in the original theory by unfolding all definitions.

The paper is well written and pleasant to read. Reviewed by Mariangiola Dezani-Ciancaglini.

Bunder, M. W. and Hindley, J. R. (1996). Two beta-equal lambda-I-terms with no types in common. *Theoret. Comput. Sci.*, 155(1):265–266.

[**MR96j:03015**] Summary: "It is known that in the Curry-style type-assignment system for simple type theory there exist two closed lambda-K-terms that are beta-equal but have no types in common. This note extends that result to lambda-I-terms."

Compagnoni, A. B. and Pierce, B. C. (1996). Higher-order intersection types and multiple inheritance. *Math. Structures Comput. Sci.*, 6(5):469–501.

[**MR98b:68109**] This paper gives essentially two new and interesting contributions. First a system combining intersection types with higher-order subtyping is introduced. The soundness of this system is proved by constructing a semantic model based on partial equivalence relations. The metatheory of this system has been studied in much greater detail by Compagnoni ["Higher-order subtyping with intersection types", Ph.D. thesis, Catholic Univ., Nijmegen, 1995; per bibl.]. Secondly, this calculus is used to extend the model for object-oriented programming described by Pierce and D. N. Turner [J. Funct. Programming 4 (1994), no. 2, 207–247; Zbl 817.68052] in order to handle inheritance and multiple inheritance. These are characteristic features of common object-oriented languages, allowing the definition of a new class which inherits methods from one or more than one superclass.

The paper is carefully written in a clear and pleasant style, in spite of the highly technical content. Reviewed by Mariangiola Dezani-Ciancaglini.

Curien, P.-L. and Di Cosmo, R. (1996). A confluent reduction for the λ-calculus with surjective pairing and terminal object. *J. Funct. Programming*, 6(2):299–327.

[**MR99b:03025**] The extension of second-order lambda calculus with the help of product types, pairing and the projection operators is considered. Some fragments of this calculus are shown to be effectively weakly normalizing. It is also proved that some systems of reductions are confluent. Reviewed by Feodor Alexandrovich Murzin.

Di Cosmo, R. and Kesner, D. (1996). Rewriting with extensional polymorphic λ-calculus. In *Computer science logic (Paderborn, 1995)*, pages 215–232. Springer, Berlin.

[**MR98c:03037**] Summary: "We provide a confluent and strongly normalizing rewriting system, based on expansion rules, for the extensional second-order typed lambda calculus with product and unit types; this system corresponds to the intuitionistic positive calculus with implication, conjunction, quantification over propositions and the constant True. This result is an important step toward a new theory of reduction based on expansion rules, and gives a natural interpretation to the notion of second-order η-long normal forms used in higher-order resolution and unification, which are here just the normal forms of our reduction system." For the entire collection see MR 98a:68002.

Fettig, R. and Löchner, B. (1996). Unification of higher-order patterns in a simply typed lambda-calculus with finite products and terminal type. In *Rewriting techniques and applications (New Brunswick, NJ, 1996)*, pages 347–361. Springer, Berlin.

[**MR1489026**] This item will not be reviewed individually. For details of the collection in which this item appears see 98h:68011 .

Ghilezan, S. (1996a). Generalized finiteness of developments in typed lambda calculi. *J. Autom. Lang. Comb.*, 1(4):247–257.

[**MR98a:03016**] Summary: "The strong normalization property of the intersection type assignment system has been used in order to prove the finiteness of developments property of the untyped lambda calculus [J.-L. Krivine, Lambda-calcul, Masson, Paris, 1990; MR 93i:03019]. Similarly, the strong normalization property of the simply typed lambda calculus can be used for the same purpose. We extend both of these methods in order to prove the generalized finiteness of developments."

Ghilezan, S. (1996b). Some topological properties of the lambda calculus. *Sci. Rev. Ser. Sci. Eng.*, (19-20):65–73. Symposium Dedicated to the Memory of Duro Kurepa (Belgrade, 1996).

[**MR98b:03025**] The author considers a type topology defined on lambda terms by their typability in the type assignment system involving intersection types. The sets of terms which are typable by a given type in the same context are a natural basis for this topology. In this setting, by using many well-known results concerning the intersection type discipline, some properties of lambda terms are proved, for instance, the continuity of application and the genericity lemma. The author also suggests some advantages of this topology with respect to the more standard filter lambda model defined by intersection types. For the entire collection see MR 98a:00010. Reviewed by Betti Venneri.

Ghilezan, S. (1996c). Strong normalization and typability with intersection types. *Notre Dame J. Formal Logic*, 37(1):44–52.

[**MR97h:03018**] In this paper the author claims to prove in a simple way the well-known result that the strongly normalizing lambda terms are precisely the typable terms in an intersection type system. The result consists of two parts, namely: (i) all typable terms are strongly normalizable, (ii) all strongly normalizable terms are typable. The proof of point (i) is really simple, since it is done by saturated sets instead of using computability, which is not suitable when the partial ordering on types is not considered. This proof is along the lines of Krivine's proof [J.-L. Krivine, Lambda-calcul, Masson, Paris, 1990; MR 93i:03019], but simpler since no notion of type interpretation is used. However, the proof of (ii) presented here is wrong, as are many other proofs of the same result in the literature. Namely, it is based on a key lemma, Proposition 4.2, which is false, asserting that "if $M\beta$-reduces to N by contracting the λ-redex $(\lambda x.P)Q, B \vdash N : \sigma$ and Q is typable in the same basis B, then also $B \vdash M : \sigma$".

The author remarks, in connection with the example on page 50, that the technical difficulty is the fact that types are not preserved by β-expansion; however, the eel slips through his fingers when he believes that the crucial point in the key proposition is the fact that Q is required to be typable in the same basis as N. In fact, this requirement is of no use, since a weakening property on bases holds in the type inference, namely: (weakening) for any basis B, any term N, and any larger basis B' such that $x : \beta \in B$ implies either $x : \beta \in B'$ or $x : \beta \wedge \gamma \in B'$ (for some $\gamma), B \vdash N : \sigma$ implies $B' \vdash N : \sigma$. Then, by weakening, two assignments of types to terms can always be done in the same basis. The following counterexample to Proposition 4.2 is so derived from the example given on page 50. Counterexample to Proposition 4.2: $y : (\alpha \to \beta) \wedge \alpha \vdash \lambda y.y : \alpha \to \alpha$ and $y : (\alpha \to \beta) \wedge \alpha \vdash yy : \beta$ but $y : (\alpha \to \beta) \wedge \alpha \nvdash \lambda y.(\lambda x.y)yy : \alpha \to \alpha$. On the other hand, let us notice that the weakening property must be used in proving point (ii), namely Theorem 4.3, from Proposition 4.2. Reviewed by Betti Venneri.

Hofmann, M. and Pierce, B. C. (1996). Positive subtyping. *Inform. and Comput.*, 126(1):11–33.

[MR97c:68091] From the summary: "The statement $S \leq T$ in a λ-calculus with subtyping is traditionally interpreted by a semantic coercion function of type $[\![S]\!] \to [\![T]\!]$ that extracts the 'T part' of an element of S. If the subtyping relation is restricted to covariant positions, this interpretation may be enriched to include both the implicit coercion and an overwriting function $\mathrm{put}[S,T] \in [\![S]\!] \to [\![T]\!] \to [\![S]\!]$ that updates the T part of an element of S. We give a realizability model and a sound equational theory for a second-order calculus of positive subtyping. Though weaker than familiar calculi of bounded quantification, positive subtyping retains sufficient power to model objects, encapsulation, and message passing, and inheritance." Reviewed by Chantal Berline.

Jacobs, B. (1996). On cubism. *J. Funct. Programming*, 6(3):379–391.

[MR97h:03011] This paper is concerned with a new proposal for classifying typed λ-calculi which is an alternative to the formalism of pure type systems well known as Barendregt's cube. The key notion of dependency is the starting point. Instead of deriving dependencies from typing rules, "settings", that is, sets of sorts explicitly ordered by their dependencies, are assumed as primitive. Then a type system can be characterised as a certain setting plus a number of "features" (formation rules for types and terms) fitting on it.

This setting-plus-features classification turns out to be very elegant. It shows an interesting order between type systems, gives a smooth solution to the problem of handling constants, and clarifies new correspondences between typed λ-calculi and logical systems. The paper is neat and pleasant to read. Reviewed by Betti Venneri.

Kamareddine, F. and Nederpelt, R. (1996). Canonical typing and π-conversion in the Barendregt cube. *J. Funct. Programming*, 6(2):245–267.

[MR97j:03033] This paper generalizes the notion of β-reduction for types to Π-reduction for types and terms. The main features of this approach are the following: (1) Terms and types are dealt with in a uniform way; (2) the introduction of a compatibility property for typing the application; (3) the clear distinction between the question of typability and that of type checking; (4) a simple algorithm for calculating the canonical types of terms; (5) the redundancy of the conversion rule; (6) the possibility of generalizing to abstractions of higher order.

The Barendregt cube extended with Π-conversion has all the properties of the original cube, except for subject reduction. However, a weak subject reduction still holds, which suffices to preserve unicity of typing. Moreover, the full subject reduction can be recovered either by allowing definitions or adding a typing rule.

The paper is well written and pleasant to read. Reviewed by Mariangiola Dezani-Ciancaglini.

Laan, T. and Nederpelt, R. (1996). A modern elaboration of the ramified theory of types. *Studia Logica*, 57(2-3):243–278.

[MR97j:03017] This paper gives an accurate formalization of the original ramified type theory, which is faithful to the original ideas as presented by Whitehead and Russell in Principia mathematica. This description uses λ-calculus, which allows one to clarify the notion of free variable and that of substitution. As a result the paper proves for ramified type theory some variants of well-known theorems which hold for the modern type systems. In particular the authors show strong normalization, the free variable lemma, the thinning lemma, the unicity of types and the subterm lemma. The paper is interesting, well written and pleasant to read. Reviewed by Mariangiola Dezani-Ciancaglini.

Martí-Oliet, N. and Meseguer, J. (1996). Inclusions and subtypes. II. Higher-order case. *J. Logic Comput.*, 6(4):541–572.

[MR99d:03026b] Summary of Part II: "The first-order theory of subtypes as inclusions developed in Part I is extended to a higher-order context. This involves providing a higher-order equational logic for (inclusive) subtypes, a categorical semantics for such a logic that is complete and has initial models, and a proof that this higher-order logic is a conservative extension of its first-order counterpart. This higher-order categorical semantics includes a new notion of homomorphism between models that is both very natural in terms of its preservation properties and substantially more general than other notions of higher-order homomorphism proposed previously. The categorical semantics of higher-order inclusive subtypes is then generalized to a notion of model with two subtype relations $\tau \leq: \tau'$ (inclusion) and

$\tau \leq \tau'$ (implicit conversion), thus reconciling and relating the two different intuitions that have so far prevailed in the first-order and higher-order cases. Axioms are then given that integrate the $\leq$ and $\leq:$ relations in the unified categorical semantics. Besides enjoying the benefits provided by each of the notions without their respective limitations, this framework supports rules for structural subtyping that are more informative and can discriminate between inclusions and implicit conversions."

Nour, K. (1996). Storage operators and $\forall$-positive types in TTR type system. *Math. Logic Quart.*, 42(3):349–368.

[MR97f:68113] The paper gives a sufficient type-theoretic condition for a term T of TTR (the theory of recursive types) to be a storage operator. The statement of the theorem uses a generalization of Gödel translations. The introduction states the main theorem as a generalization of a result due to Krivine, while the rest of the paper gives the precise definitions and technical details. Reviewed by Eugenio Moggi.

Rehof, J. (1996). Strong normalization for non-structural subtyping via saturated sets. *Inform. Process. Lett.*, 58(4):157–162.

[MR97f:68114] Summary: "We show that the standard method of saturated sets for proving strong normalization of β-reduction in the simply typed and second-order polymorphic lambda calculus incorporates non-structural subtyping systems in a natural way. This shows that strong normalization for non-structural subtyping proved by Wand, O'Keefe and Palsberg (1995) via coercion interpretations can be obtained in a straightforward extension of the standard method. The proof presented here is compared to other proofs of strong normalization for subtyping systems."

Sayag, E. and Mauny, M. (1996). Characterization of the principal type of normal forms in an intersection type system. In *Foundations of software technology and theoretical computer science (Hyderabad, 1996)*, pages 335–346. Springer, Berlin.

[MR98k:03040] Summary: "We introduce a restriction of the intersection type discipline that leads to a principal type property in the classical sense for normal forms. We characterize completely the structure of principal types of normal forms and we give an algorithm that reconstructs normal forms from types." For the entire collection see MR 98i:68006.

Severi, P. G. (1996). *Normalisation in lambda calculus and its relation to type inference*. Technische Universiteit Eindhoven, Eindhoven. Dissertation, Technische Universiteit Eindhoven, Eindhoven, 1995.

[MR1392226] There will be no review of this item.

Springintveld, J. (1996). Third-order matching in the polymorphic lambda calculus. In *Higher-order algebra, logic, and term rewriting (Paderborn, 1995)*, pages 221–237. Springer, Berlin.

[MR97g:03017] Summary: "We show that it is decidable whether a third-order matching problem in the polymorphic lambda calculus has a solution. We give an algorithm that, given such a problem, returns a substitution if one exists, and returns 'fail' otherwise." For the entire collection see MR 97f:68005.

Stefanova, M. (1996). A schematic proof of strong normalization for the systems of the λ-cube. *Annuaire Univ. Sofia Fac. Math. Inform.*, 90(1-2):17–40 (1998). Fourth Logical Biennial (Gjuletchitza, 1996).
[**MR1700125**] A review for this item is in process.

Stefanova, M. and Geuvers, H. (1996). A simple model construction for the calculus of constructions. In *Types for proofs and programs (Torino, 1995)*, pages 249–264. Springer, Berlin.
[**MR98e:03008**] Summary: "We present a model construction for the calculus of constructions (CC) where all dependencies are carried out in a set-theoretical setting. The soundness theorem is proved and as a consequence of it strong normalization for CC is obtained. Some other applications of our model constructions are: showing that CC + classical logic is consistent (by constructing a model for it) and showing that the axiom of choice is not derivable in CC (by constructing a model in which the type that represents the axiom of choice is empty)." For the entire collection see MR 98d:03006.

Tiuryn, J. (1996). A sequent calculus for subtyping polymorphic types. In *Mathematical foundations of computer science 1996 (Cracow)*, pages 135–155. Springer, Berlin.
[**MR1478956**] This item will not be reviewed individually. For details of the collection in which this item appears see 98f:68011.

Tiuryn, J. and Urzyczyn, P. (1996). The subtyping problem for second-order types is undecidable. In *11th Annual IEEE Symposium on Logic in Computer Science (New Brunswick, NJ, 1996)*, pages 74–85. IEEE Comput. Soc. Press, Los Alamitos, CA.
[**MR98b:68011**] Summary: "We prove that the subtyping problem induced by Mitchell's containment relation for second-order polymorphic types is undecidable. It follows that type-checking is undecidable for the polymorphic lambda-calculus extended by an appropriate subsumption rule." For the entire collection see MR 98a:68006.

Urzyczyn, P. (1996). Positive recursive type assignment. *Fund. Inform.*, 28(1-2):197–209.
[**MR98a:03017**]

The author studies inference systems for assigning recursive types to untyped lambda terms. In order to preserve strong normalization of typeable terms, only positive recursive types are considered, that is, types in which the recursion does not occur to the left of an odd number of arrows.

The motivation for this work stems from the existence of different approaches for defining recursive type systems in the literature. These approaches and their relationships are investigated in order to obtain the main result (Theorem 5.1) that the class of typeable terms remains the same for all the different type disciplines and that typeability is decidable.

In the process of proving this result, several properties of the type systems are stated. However, the proof of the subject reduction property (Theorem 2.1-2) is only sketched and needs to be complemented with some subtle technical points (see the discussion about subject reduction and invertibility in Marz's paper cited in the references). The paper is interesting mainly for introducing a new tree-like representation of positive recursive types, since the usual model of regular trees for arbitrary recursive types is not adequate for representing information about the positivity condition. This model of positive regular trees, besides being a crucial tool for proving Theorem 5.1, seems to be the most interesting contribution of the paper. Reviewed by Betti Venneri.

Aoun, A. and Mirasyedioğlu, Ş. (1997). Type assignment system and unification. *J. Inst. Math. Comput. Sci. Comput. Sci. Ser.*, 8(1):1–12.
[**MR1467619**] There will be no review of this item.

Barthe, G. and Melliès, P.-A. (1997). On the subject reduction property for algebraic type systems. In *Computer science logic (Utrecht, 1996)*, pages 34–57. Springer, Berlin.
[**MR98m:68023**] Summary: "Algebraic type systems provide a general framework for the study of the interaction between typed λ-calculi and typed rewriting systems. A major problem in the development of a general theory for algebraic type systems is to prove that typing is preserved under reduction (subject reduction lemma). In this paper, we propose a general technique to prove subject reduction for a large class of algebraic type systems. The idea is to consider for every (functional) algebraic type system a labelled syntax for which subject reduction is easy to prove and then prove the equivalence between the labelled and standard syntaxes whenever the labelled system is strongly normalising. The equivalence can then be used to recover confluence, strong normalisation and subject reduction for the standard syntax." For the entire collection see MR 98k:03005.

Barthe, G. and Sørensen, M. H. (1997). Domain-free pure type systems. In *Logical foundations of computer science (Yaroslavl, 1997)*, pages 9–20. Springer, Berlin.
[**MR1611405**] This item will not be reviewed individually. For details of the collection in which this item appears see 98i:68009.

Brandt, M. and Henglein, F. (1997). Coinductive axiomatization of recursive type equality and subtyping. In *Typed lambda calculi and applications (Nancy, 1997)*, pages 63–81. Springer, Berlin.
[**MR99b:03024**] From the summary: "We present new sound and complete axiomatizations of type equality and subtype inequality for a first-order type language with regular recursive types. The rules are motivated by coinductive characterizations of type containment and type equality via simulation and bisimulation, respectively.

The main novelty of the axiomatization is the fixpoint rule (or coinduction principle), which has the form: 'from $A, P \vdash P$ infer $A \vdash P$', where P is either a type equality $\tau = \tau'$ or a type containment $\tau \leq \tau'$. We define what it means for a proof (formal derivation) to be formally contractive and show that the fixpoint rule is sound if the proof of the premise $A, P \vdash P$ is contractive. (A proof of $A, P \vdash P$ using the assumption axiom is, of course, not contractive.) The fixpoint rule thus allows us to capture a coinductive relation in the fundamentally inductive framework of inference systems." For the entire collection see MR 98f:68017.

Chen, G. (1997). Subtyping calculus of construction (extended abstract). In *Mathematical foundations of computer science 1997 (Bratislava)*, pages 189–198. Springer, Berlin.

[MR1640220] This item will not be reviewed individually. For details of the collection in which this item appears see 99d:68009.

Courant, J. (1997). A module calculus for pure type systems. In *Typed lambda calculi and applications (Nancy, 1997)*, pages 112–128. Springer, Berlin.

[MR1480467] This item will not be reviewed individually. For details of the collection in which this item appears see 98f:68017.

Despeyroux, J., Pfenning, F., and Schürmann, C. (1997). Primitive recursion for higher-order abstract syntax. In *Typed lambda calculi and applications (Nancy, 1997)*, pages 147–163. Springer, Berlin.

[MR1480469] This item will not be reviewed individually. For details of the collection in which this item appears see 98f:68017.

Dezani-Ciancaglini, M., Ghilezan, S., and Venneri, B. (1997a). The "relevance" of intersection and union types. *Notre Dame J. Formal Logic*, 38(2):246–269.

[MR99e:03015] The authors investigate a system $\mathrm{TA}_{\wedge\vee}$ of typed combinatory logic with intersection and union types, especially with respect to the formulas-as-types notion (the Curry-Howard isomorphism). This means that the goal is to find a formal logic in the Hilbert style to which this typed combinatory logic corresponds. The major problem to be overcome in obtaining this correspondence is that in the rules for $\wedge$-introduction ($\Gamma \vdash M : \sigma \& \Gamma \vdash M : \tau \ \Rightarrow\ \Gamma \vdash M : \sigma \wedge \tau$) and $\vee$-elimination ($\Gamma, x : \sigma \vdash M : \rho \& \Gamma, x : \tau \vdash M : \rho \& \Gamma \vdash N : \sigma \vee \tau \ \Rightarrow\ \Gamma \vdash M[N/x] : \rho$), the subproofs of the formulas corresponding to σ and τ are required to be identical, whereas in ordinary logic they are only required to have the same conclusions. The authors solve this problem by relating $\mathrm{TA}_{\wedge\vee}$ to a sequence of systems which employ a partial order relation on types to eliminate the use of these rules. This process uses results of Venneri [J. Logic Comput. 4 (1994), no. 2, 109–124; MR 95b:68068], F. Barbanera and Dezani-Ciancaglini [in Theoretical aspects of computer software (Sendai, 1991), 651–674, Lecture Notes in Comput. Sci., 526, Springer, Berlin, 1991; MR 93g:68071] and Barbanera, Dezani-Ciancaglini and U. de'Liguoro [Inform. and Comput. 119

(1995), no. 2, 202–230; MR 96c:68111]. The resulting logic is a modification of the relevance logic
rmB^+ of R. K. Meyer and R. Routley [Logique et Analyse (N.S.) 15 (1972), 407–428; MR 48 #5857; J. Philos. Logic 1 (1972), no. 2, 192–208; MR 53 #12878]. (The modifications are that the "Church constant" is not present and one extra axiom is added.) The authors close the paper by comparing the logic they obtain with intuitionistic logic and with M. Dummett's logic [J. Symb. Logic 24 (1959), 97–106; MR 23#A801]. Reviewed by J. P. Seldin.

Dezani-Ciancaglini, M., Ghilezan, S., and Venneri, B. (1997b). The "relevance" of intersection and union types. *Notre Dame J. Formal Logic*, 38(2):246–269.

[MR1489412] A review for this item is in process.

Di Cosmo, R. and Ghani, N. (1997). On modular properties of higher order extensional lambda calculi. In *Automata, languages and programming (Bologna, 1997)*, pages 237–247. Springer, Berlin.

[MR1616189] This item will not be reviewed individually. For details of the collection in which this item appears see 98k:68002.

Farkh, S. and Nour, K. (1997). Résultats de complétude pour des classes de types du système $\mathcal{AF}\in$. *RAIRO Inform. Théor. Appl.*, 31(6):513–537.

[MR99d:68037] The paper under review is divided into two parts. The first half contains an introduction to the Krivine system of second order functional arithmetic, $\mathcal{AF}\in$; the second half is devoted to the authors' results. To comment on these results we shall follow the authors on a short trip around their topic.

$\mathcal{AF}\in$ is a type assignment system, namely a formal system associating type expressions to type-free λ-terms. In $\mathcal{AF}\in$ types are formulas of second-order predicate logic, but the form and proof-theoretic properties are the same as those of (the type assignment version of) Girard's system $\mathcal{F}$, in which types are just formulas of second-order propositional logic.

One of the most relevant properties of typeable terms is indeed, both for system $\mathcal{F}$ and $\mathcal{AF}\in$, that they are strongly normalizing, i.e. any β-reduction out of them is finite.

The formulas of $\mathcal{AF}\in$ admit a standard model-theoretic interpretation. A nonstandard model is, on the other hand, a structure in which types are interpreted as certain sets of λ-terms, with the intended meaning that true formulas have nonempty interpretation. As a matter of fact these sets come from the model theory of untyped λ-calculus, and satisfy the minimal property that if $t[u/x]t_1 \cdots t_k \in X$ then $(\lambda x.t)ut_1 \cdots t_k \in X$ (saturated sets). Now the key fact is that a type A can be interpreted by a saturated set $|A|_{\mathcal{I}}$ (where $\mathcal{I}$ is an interpretation of free variables in A) which is built along the logical connectives of A. An adequacy theorem states the correctness of such an interpretation: if $\vdash t \colon A$ (say for closed t and A), then $t \in |A|$. Such an interpretation is known as the realizability interpretation, and its first application is in the proof of strong normalization. The converse of adequacy does not hold, in general, and this seems to be a consequence of the fact that the syntax of untyped λ-terms is poorer than what is needed to reflect the structure of (intuitionistic) derivations of their types. Such a correspondence does exist in the case of typed λ-calculi, where it is known as the Curry-Howard interpretation of formulas as types and proofs as terms.

There is, however, a case in which the correspondence between the derivation of $\vdash t\colon A$ and t is somehow recovered. Suppose that A has the form $\forall x_1 \cdots \forall x_n . D_1[x_1] \wedge \cdots \wedge D_n[x_n] \to E[f(x_1, \cdots, x_n)]$ and $D_1, \cdots, D_n, E$ are data types. Roughly speaking, data types are second-order formulas inductively defining concrete objects, like Booleans, integers, lists, etc. Now, if A is as above, the term t represents the function f, in the sense that, in the model, $tu_1 \cdots u_n = f(u_1, \cdots, u_n)$ whenever $u_1 \in |D_1|, \cdots, u_n \in |D_n|$. As a matter of fact this follows not only from adequacy, but, essentially, from completeness, namely from the reverse implication, which holds in the case of data types.

Now the authors of the present paper raise the question of when completeness holds, and give a (rather complex) definition of a superclass of data types for which they prove completeness, even if they do not achieve a characterization of the whole class of types enjoying such a property.

It is perhaps a naive question to ask the point of such an effort. Rather surprisingly, the answer we are given (in two lines) is that this would clarify the relation between the storage operators as originally defined by Krivine, and the (weaker) formulation of this concept by one of the authors.

Storage operators are combinators simulating call-by-value into head-reduction. If D is a data type and $\vdash T\colon D \to D^*$ (where D^* is the Gödel translation of D) then T is a storage operator for terms in $|D|$. Since yet another variant of the storage operator concept (due to a researcher on the same equipe) satisfies this property for any $\mathcal{AF}\in$ type A, the main concern of the authors seems to be to investigate where the difference lies. In fact all versions of storage operators coincide on $|A|$ if completeness holds for A.

The theme treated deserves attention from the point of view of the logical analysis of actual (imperative and call-by-value) programming languages, which could also be relevant for program extraction techniques. The present paper, however, reports on partial results on what is seemingly an internal debate of a small community. Reviewed by Ugo de'Liguoro.

Gallier, J. (1997). Kripke models and the (in)equational logic of the second-order λ-calculus. *Ann. Pure Appl. Logic*, 84(3):257–316.

[MR98j:03018] J. C. Mitchell and E. Moggi [Ann. Pure Appl. Logic 51 (1991), no. 1-2, 99–124; MR 92a:03017] introduced a Kripke-style semantics for the second-order polymorphic lambda calculus, together with a completeness result (with respect to conversion) dealing with possibly empty types. In the paper under review, these models are generalised to Kripke models of reduction, essentially by using preorders, in the form of functors to the category of preorders (instead of to the category of sets). Soundness and completeness results are obtained for an appropriate collection of rewrite rules. Reviewed by Bart Jacobs.

Ghani, N. (1997). Eta-expansions in F^ω. In *Computer science logic (Utrecht, 1996)*, pages 182–197. Springer, Berlin.

[MR98k:03037] Summary: "The use of expansionary η-rewrite rules in typed λ-calculi has become increasingly common in recent years as their advantages over contractive η-rewrite rules have become more apparent. Not only does one obtain simultaneously the decidability of $\beta\eta$-equality and a natural construction of the long $\beta\eta$-normal forms, but rewrite relations using expansions retain key properties when

combined with first-order rewrite systems, generalise more easily to other type constructors and are supported by a categorical theory of reduction. However, one area where η-contractions have until now remained the only possibility is in the more powerful type theories of the λ-cube. This paper begins to rectify this situation by considering a higher-order polymorphic λ-calculus known as F^{ω}." For the entire collection see MR 98k:03005.

Ghelli, G. (1997). Termination of system F-bounded: a complete proof. *Inform. and Comput.*, 139(1):39–56.
[MR99a:68024] "System F-bounded" is a typed, second-order λ calculus that has been defined to deal with the type systems of object-oriented languages. This paper gives a detailed proof of an important property of reduction in "system F-bounded". This property, namely the strong normalisation, or termination, property, is essentially the fact that no infinite reduction starts from a typed term. It is related to the possibility of solving recursive type equations. The proof itself uses the Tait-Girard method of saturated sets in a rather straightforward manner, but it is carried out in great detail. Reviewed by Henri Volken.

Hindley, J. R. (1997). *Basic simple type theory.* Cambridge University Press, Cambridge.
[MR98k:03038] Polymorphic type assignment systems continue to be of major importance in programming language design. These systems are essentially extensions of basic simple type theory, concerning function types, from which they inherit both problems and solving techniques. This book presents in full the current state-of-the-art in simple type theory, including basic ideas, advanced results and also two algorithms which are not in the accessible literature. Relations to intuitionistic and weaker logics, semantics, type-inference and algorithms for counting inhabitants are the main topics. All these subjects are treated in a very rigorous way with exceptional care for mathematical details. Therefore, the book makes useful and stimulating reading and it will be an essential tool for computer scientists working in type theory and related areas. Reviewed by Betti Venneri.

Jay, C. B. (1997). Covariant types. *Theoret. Comput. Sci.*, 185(2):237–258.
[MR98m:68026] A famous negative result of Reynolds states that the second order polymorphic lambda calculus has no set-theoretic models. In the paper under review, a restricted system of "covariant types" is presented which does have a set-theoretic semantics (in terms of the author's data functors and natural transformations). The key point is to replace the ordinary exponent type $S \to T$ by a "transformation type" $S \Rightarrow T$ in which all free type variables of S are bound. This restricts expressivity. For example, many polymorphic operations (such as filtering on lists) are available only for individual types. But many standard terms are still available. Reviewed by Bart Jacobs.

Kurata, T. (1997). A type theoretical view of Böhm-trees. In *Typed lambda calculi and applications (Nancy, 1997)*, pages 231–247. Springer, Berlin.

[**MR99a:03019**] Summary: "Two variations of the intersection type assignment system are studied in connection with Böhm-trees. One is the intersection type assignment system with a nonstandard subtype relation, by means of which we characterize the location of $\perp$ in Böhm-trees. The other is a refinement of the intersection type assignment system whose restricted typability is shown to coincide with finiteness of Böhm-trees." For the entire collection see MR 98f:68017.

Loader, R. (1997). Equational theories for inductive types. *Ann. Pure Appl. Logic*, 84(2):175–217.
[**MR98k:03018**] The theories of standard models of the untyped lambda calculus can be nicely characterised [see H. P. Barendregt, The lambda calculus, Revised edition, North-Holland, Amsterdam, 1984; MR 86a:03012 (Chapter 19)]. There is relatively little such work for models of typed lambda calculi. The paper under review forms an exception. It considers a simply typed calculus with type variables (but without abstraction over types), including fixed points of (covariant) types. The latter are described as initial algebras. It is shown that the equational theory of the standard model of partial equivalence relations (PERs) is the minimal extensional theory, and is at the same time the largest consistent theory. This result is proved via an intermediate model of "effective totality domains". Reviewed by Bart Jacobs.

Malecki, S. (1997). Proofs in system F_ω can be done in system F^1_ω. In *Computer science logic (Utrecht, 1996)*, pages 297–315. Springer, Berlin.
[**MR1611505**] This item will not be reviewed.

McCusker, G. (1997). Games and definability for FPC. *Bull. Symbolic Logic*, 3(3):347–362.
[**MR99f:03015**] In the past few years, "games" have become a hot topic in logic and theoretical computer science. As a true intermediary situatable somewhere halfway between syntax and semantics, games were at the heart of the construction of convincing 'fully abstract' models (this means more or less that one asks for a bijection between elements of the model and terms) for PCF: by S. Abramsky, P. Malacaria and R. Jagadeesan [in Theoretical aspects of computer software (Sendai, 1994), 1–15, Lecture Notes in Comput. Sci., 789, Springer, Berlin, 1994; see MR 95d:68006], following earlier work on games for linear logic, and by J. M. E. Hyland and C.-H. L. Ong ["On full abstraction for PCF: I, II, III", submitted]. In the paper under review the author constructs a similar games model not for PCF but for FPC—a type theory with products, sums, function spaces and recursive types. Technically the work leans heavily upon that of the authors mentioned above, and a main part of the paper is devoted to a concise description of the games. In this part all proofs are omitted. In the final section it is shown that every finite element of the games model obtained is indeed the interpretation of some term of the language.

Though it was listed as a "communication" in the Bulletin of Symbolic Logic, I would say that the main interest of this small paper is expository: anyone new to such games can find here a nice survey of a method that in virtually no time has proven itself as a technique which, with some perseverance, can be forced to work not only for PCF or FPC (or CFP?), but for many variations on quite a few other systems as well. Reviewed by Harold Schellinx.

Nour, K. (1997a). La valeur d'un entier classique en $\lambda\mu$-calcul. *Arch. Math. Logic*, 36(6):461–473.

[**MR98m:03062**] Recently, there has been in the theoretical computer science community a great interest in the computational features of classical logic. Various formal systems and, in particular, calculi, have been proposed to provide an algorithmic interpretation for classical proofs. One such calculus is the λ-calculus of M. Parigot. A good measure of the expressiveness of a calculus is the possibility of encoding data types and their related operations in the calculus (an example is the Church numerals for the λ-calculus). In the λ-calculus (its second-order version) we can encode natural numbers. The peculiar characteristics of classical logic, however, make their representations nonunique. Then, if we wish to compute in the λ-calculus using the encoded natural numbers, we need procedures to obtain numbers from their possible representations. Three such procedures are provided in the paper. Their interest is mainly theoretical, since for actual computations in a calculus it is unlikely that one would use logical representations of natural numbers. Nonetheless, these procedures are a step (perhaps small, but still a step) towards a full understanding of the λ-calculus.

The paper, written in French, is a bit lacking in clarity in general and lacks formality in a few definitions. Reviewed by Franco Barbanera.

Nour, K. (1997b). Une réponse négative à la conjecture de E. Tronci pour les systèmes numériques typés. *RAIRO Inform. Théor. Appl.*, 31(6):539–558.

[**MR1607520**] This paper concerns the interplay between adequate numeral systems and storage operators, called in French "operateurs de mise en memoire". We briefly recall these two notions.

The pure, type-free λ-calculus is Turing-complete in the following sense: There exists a denumerable sequence of distinct closed terms $d \equiv d_0, d_1, \cdots$, encoding natural numbers, such that for all p and each partial recursive p-ary function φ there exists a closed term F such that for all natural numbers $n_1, \cdots, n_p$, if $\varphi(n_1, \cdots, n_p)$ is defined, then $Fd_{n_1} \cdots d_{n_p} =_\beta \mathbf{d}_{\varphi(n_1,\cdots,n_p)}$. Otherwise, $Fd_{n_1} \cdots \mathbf{d}_{n_p}$ has no head normal-form (is unsolvable). Conversely, if such an F exists for a partial numerical function φ, then φ is λ-definable with respect to d.

According to H. Barendregt's book, the sequence d is called a numeral system if the successor function and the test for zero are λ-definable. Barendregt showed that all (total) recursive functions are λ-definable with respect to the numeral system d if and only if a predecessor function $P(n+1) = n$ is λ-definable with respect to the same numeral system. In this case, d is said to be adequate.

Storage operators were introduced by J.-L. Krivine [Arch. Math. Logic 30 (1990), no. 4, 241–267; MR 91j:03011] and have been studied by other researchers, including the author of the present paper. If X is a set of λ-terms then by a storage operator for X is meant a term O_X such that for all M and N, if N is β-equivalent to some term in X, then $O_X MN$ head-reduces to MN', where N' is, essentially, the normal form of N. (By the way, storage operators simulate call-by-value in call-by-name λ-calculus with head-reduction.) As Krivine showed, there is a connection between storage operators and data types (roughly speaking, data types are second-order propositional formulas inductively defining Booleans, integers, lists, etc.) which are definable in Girard's system $\mathcal{F}$: if D is a data type and $\vdash O_D\colon D \to D^*$ (where D^* is the Gödel translation of D), then O_D is a storage operator for terms of type D.

In the present paper it is proved that each adequate numeral system whose terms are normal forms has a storage operator. This means that, given an adequate numeral system d, there exists a storage operator for the set $\{d_0, d_1, \cdots\}$. The converse implication has been conjectured by Tronci, but it remains open. Indeed, the author shows that there exists a numeral system e such that all e_n are uniformly typable in system $\mathcal{F}$, which is also the case for the successor and test for zero; e has a storage operator of the right type, but there exists no predecessor having the expected type (namely, the type of the successor combinator for e) in system $\mathcal{F}$.

A particular case of an adequate numeral system is given by Church numerals: $\overline{n} = \lambda xy.x^n y$. It is with respect to the typing of these numerals that representable functions are classified in type theories (for a survey see a paper by S. Fortune, D. Leivant and M. O'Donnell [J. Assoc. Comput. Mach. 30 (1983), no. 1, 151–185; MR 84j:03029]). For example, it is known that the numerical functions representable in simply typed λ-calculus without constants are exactly the generalized polynomials (Schwichtenberg); those representable in system $\mathcal{F}$ are the functions that are provably total in second-order arithmetic (Girard).

Now a relevant point is the following: Does the existence of storage operators for typable numerical systems affect the class of representable functions? If yes, how? After all, it is the answers to these still open questions that would arouse interest in the results in the present paper, which otherwise are of a restricted technical nature. Reviewed by Ugo de'Liguoro.

Pierce, B. and Steffen, M. (1997a). Corrigendum to: "Higher-order subtyping". *Theoret. Comput. Sci.*, 184(1-2):247.

[**MR98h:68031b**] See (Pierce and Steffen, 1997b).

Pierce, B. and Steffen, M. (1997b). Higher-order subtyping. *Theoret. Comput. Sci.*, 176(1-2):235–282.

[**MR98h:68031a**] This is an interesting paper on the analysis of system $F^{\omega}_{\leq}$. This system is an extension of Girard's higher-order polymorphic λ-calculus with subtyping. Many formal models for statically typed object-oriented languages are based on system $F^{\omega}_{\leq}$. The present paper develops the fundamental metatheory of this calculus, leading up to sound and complete algorithms for checking the subtyping and typing relations. This is a difficult task, essentially because of the interaction of the rule of conversion between types with the rule of transitivity.

As noticed by the authors, the present proofs are broadly similar to those of A. B. Compagnoni [in Computer science logic (Kazimierz, 1994), 46–60, Lecture Notes in Comput. Sci., 933, Springer, Berlin, 1995; see MR 98d:68013CMP 1 471 217], where the combination of system $F^{\omega}_{\leq}$ with intersection types was analyzed.

The paper is written in a clear and pleasant style, in spite of the highly technical content. Reviewed by Mariangiola Dezani-Ciancaglini.

Pierce, B. C. (1997). Intersection types and bounded polymorphism. *Math. Structures Comput. Sci.*, 7(2):129–193.

[**MR98m:03031**] Summary: "Intersection types and bounded quantification are complementary extensions of a first-order programming language with subtyping. We define a typed λ-calculus combining these extensions, illustrate its unusual properties and develop basic proof-theoretic and semantic results leading to algorithms for subtyping and typechecking."

Poll, E. (1997). System F with width-subtyping and record updating. In *Theoretical aspects of computer software (Sendai, 1997)*, pages 439–457.

Springer, Berlin.

[**MR1608911**] This item will not be reviewed individually. For details of the collection in which this item appears see 98i:68019.

Urzyczyn, P. (1997a). Inhabitation in typed lambda-calculi (a syntactic approach). In *Typed lambda calculi and applications (Nancy, 1997)*, pages 373–389. Springer, Berlin.

[**MR98i:68128**] Summary: "A type is inhabited (nonempty) in a typed calculus iff there is a closed term of this type. The inhabitation (emptiness) problem is to determine whether a given type is inhabited. This paper provides direct purely syntactic proofs of the following results: the inhabitation problem is PSPACE-complete for simply typed lambda-calculus and undecidable for the polymorphic second-order and higher-order lambda calculi (systems F and F_ω)." For the entire collection see MR 98f:68017.

Urzyczyn, P. (1997b). Type reconstruction in F_ω. *Math. Structures Comput. Sci.*, 7(4):329–358.

[**MR99e:03040**] The author presents in an exemplary manner, making this paper truly a pleasure to read, some interesting technical results concerning Girard's calculus F_ω viewed as a 'Curry style' type assignment system for pure lambda terms. Its main concern is the question of whether it is recursively decidable whether a given pure lambda term can be typed by an F_ω-term: this is the so-called type reconstruction problem for F_ω. The presentation in this paper modifies and simplifies earlier versions of the author's proof that the type reconstruction problem for F_ω is undecidable.

In Section 3 it is shown that the type inference rules of F_ω are not sufficient to type all strongly normalizable terms (a necessary first step when tackling the problem, as the set of strongly normalizable F_ω-terms is already known to be undecidable): the term

$$(\lambda x.z(x(\lambda fu.fu))(x(\lambda vg.gv)))(\lambda y.yyy),$$

which is easily seen to be strongly normalizable, turns out to be untypable.

It is well known that in general an integer function representable in the untyped lambda calculus by a term F does not have to be representable in a typed calculus: it might very well be the case that F is not typable by the appropriate function-type. However, something weaker might be true: the function could still be nonuniformly representable, i.e. the type assigned to F may be different for different n. And yes, indeed, the author proves that every partial recursive function is nonuniformly representable in F_ω (Theorem 5.6). It then follows (from the proof) that the type reconstruction problem is undecidable, for F_ω as well as for all of its subsystems $\mathrm{F}_\omega^{(n)}$ (with constructors of level at most n occurring in types). Moreover, the set of terms typable in $\mathrm{F}_\omega^{(1)}$ is not recursively separable from the set of normalizable terms. Reviewed by Harold Schellinx.

van Bakel, S., Liquori, L., Ronchi della Rocca, S., and Urzyczyn, P. (1997). Comparing cubes of typed and type assignment systems. *Ann. Pure Appl. Logic*, 86(3):267–303.

[**MR98i:03016**] Types are introduced in λ-calculus following two different approaches: the first one (called fully typed) is to define terms directly decorated

with types, while the second one (called type assignment) is to assign types to terms of the untyped calculus. Since λ-calculus can be viewed as a paradigmatic programming language, these two approaches correspond to explicitly and implicitly typed languages, respectively. This paper is an excellent contribution to the analysis of the relationships between fully typed and type assignment systems.

Quite surprisingly, it is proved that, in the presence of polymorphism, the correspondence between the two approaches is not given by the simple erasure of type information. A new erasing function is then introduced, allowing the construction of a cube of type assignment systems that is isomorphic to Barendregt's cube. Reviewed by Adolfo Piperno.

Xi, H. (1997). Simulating η-expansions with β-reductions in the second-order polymorphic λ-calculus. In *Logical foundations of computer science (Yaroslavl, 1997)*, pages 399–409. Springer, Berlin.
[MR1611441] This item will not be reviewed individually. For details of the collection in which this item appears see 98i:68009.

Amadio, R. M. and Coupet-Grimal, S. (1998). Analysis of a guard condition in type theory (extended abstract). In *Foundations of software science and computation structures (Lisbon, 1998)*, pages 48–62. Springer, Berlin.
[MR1641331] This item will not be reviewed individually. For details of the collection in which this item appears see 99c:68005.

Chen, G. (1998). Dependent type system with subtyping. I. Type level transitivity elimination. In *Proceedings of the International Summer School and Workshop on Formal Models of Programming and their Applications (Beijing, 1997)*, volume 13, pages 564–578.
[MR1677508] A review for this item is in process.

Di Gianantonio, P. and Franco, G. (1998). A type assignment system for the game semantics. In *Theoretical computer science (Prato, 1998)*, pages 37–47. World Sci. Publishing, River Edge, NJ.
[MR1700311] A review for this item is in process.

Farkh, S. and Nour, K. (1998). Un résultat de complétude pour les types $\forall^+$ du système $\mathcal{F}$. *C. R. Acad. Sci. Paris Sér. I Math.*, 326(3):275–279.
[MR99d:03014] Summary: "We present a completeness result for the types with positive quantifiers of the J.-Y. Girard type system $\mathcal{F}$. This result generalizes a theorem of R. Labib-Sami."

Fernández, M. (1998). Type assignment and termination of interaction nets. *Math. Structures Comput. Sci.*, 8(6):593–636. Logic and models of computation (Marseille, 1996).
[MR1672402] A review for this item is in process.

Fu, Y. (1998). Structures definable in polymorphism. In *Proceedings of the International Summer School and Workshop on Formal Models of Programming and their Applications (Beijing, 1997)*, volume 13, pages 579–587.

[MR1677509] A review for this item is in process.

Gallier, J. (1998). Typing untyped λ-terms, or reducibility strikes again! *Ann. Pure Appl. Logic*, 91(2-3):231–270.

[MR99b:03026] The author uses a new uniform approach in studying known, and some new, connections between the typability of λ-terms, in the full conjunctive type theory $(\mathcal{D}\Omega)$ and that without $\omega(\mathcal{D})$, and certain properties of the λ-terms. This "new and more modular version of the reducibility method" leads efficiently to theorems of the form: If the set of λ-terms $\mathcal{P}$ satisfies certain properties, then $\mathcal{P}$ contains all the λ-terms: (i) with nontrivial types in $(\mathcal{D}\Omega)$, (ii) with ω-free types in $(\mathcal{D}\Omega)$, (iii) typable in $\mathcal{D}$, (iv) typable in $(\mathcal{D}\Omega)$ with a weakly nontrivial type. Of these the last is new. Also, characterizations of terms: (a) having head normal form, (b) that are normalizable, (c) that are strongly normalizable, and (d) having weak head normal form are obtained in a uniform way. The last characterization is new. Some open problems are listed. Reviewed by Martin W. Bunder.

Ghelli, G. and Pierce, B. (1998). Bounded existentials and minimal typing. *Theoret. Comput. Sci.*, 193(1-2):75–96.

[MR99f:03014] $F_{\leq}$ is a typed lambda-calculus which allows subtyping and second-order bounded quantification. This allows the definition of bounded quantifiers over types. $F_{\leq}$ has the property that every term that has a type has a unique minimal type relative to $\leq$. A natural extension of $F_{\leq}$, with primitive bounded quantifiers over types, and with more natural and less restrictive rules, is shown not to have this minimal type property, while a similar extension of a fragment of $F_{\leq}$ (Kernel Fun) does have this property. The authors also give an algorithm which, for all typed terms, determines the minimal type in Kernel Fun. Reviewed by Martin W. Bunder.

Jouannaud, J.-P. and Rubio, A. (1998). Rewrite orderings for higher-order terms in η-long β-normal form and the recursive path ordering. *Theoret. Comput. Sci.*, 208(1-2):33–58. Rewriting techniques and applications (New Brunswick, NJ, 1996).

[MR99i:68071] An earlier version of this paper that appeared as part of the conference proceedings has been reviewed [in Rewriting techniques and applications (New Brunswick, NJ, 1996), 108–122, Lecture Notes in Comput. Sci., 1103, Springer, Berlin, 1996; MR 98i:68143]. For the entire collection see MR 99f:68010.

Pitts, A. (1998). Parametric polymorphism and operational equivalence (preliminary version). In *Second Workshop on Higher-order Operational Techniques in Semantics (HOOTS II) (Stanford, CA, 1997)*, page 26 pp. (electronic). Elsevier, Amsterdam.

[**MR1649822**] This item will not be reviewed individually. For details of the collection in which this item appears see 99e:68003.

Poll, E. (1998). Expansion postponement for normalising pure type systems. *J. Funct. Programming*, 8(1):89–96.

[**MR98m:68028**] Summary: "Expansion postponement is a tantalisingly simple conjecture about pure type systems which has so far resisted all attempts to prove it for any interesting class of systems. We prove the property for all normalising pure type systems, and discuss the connection with typechecking."

Yokouchi, H. (1998). Completeness of type assignment systems with intersection, union, and type quantifiers. In *Thirteenth Annual IEEE Symposium on Logic in Computer Science (Indianapolis, IN, 1998)*, pages 368–379. IEEE Computer Soc., Los Alamitos, CA.

[**MR99i:03015**] Summary: "This paper develops type assignment systems for intersection and union types, and type quantifiers. The known system for these types is not semantically complete. We introduce a certain class of typing statements, called stable statements, which includes all statements without type quantifiers, and we show that the known system is complete for stable statements if we add two axiom schemas expressing the distributive laws of intersection over union and the existential quantifier, respectively. All the results are obtained in a systematic way with sequent calculi for type assignment and the cut-elimination theorem for them." For the entire collection see MR 99h:68008.

Kamareddine, F., Bloo, R., and Nederpelt, R. (1999). On π-conversion in the λ-cube and the combination with abbreviations. *Ann. Pure Appl. Logic*, 97(1-3):27–45.

[**MR1682067**] A review for this item is in process.

Kfoury, A. J., Ronchi della Rocca, S., Tiuryn, J., and Urzyczyn, P. (1999). Alpha-conversion and typability. *Inform. and Comput.*, 150(1):1–21.

[**MR1681704**] A review for this item is in process.

Minari, P. (1999). Theories of types and names with positive stratified comprehension. *Studia Logica*, 62(2):215–242. Selected papers in honour of Ettore Casari.

[**MR1696000**] A review for this item is in process.

Song, F. and Qian, Y. (1999). A syntactic proof of the conservativity of λ_ω over λ_2. *J. Comput. Sci. Tech.*, 14(2):129–134.

[**MR1675870**] A review for this item is in process.

Wells, J. B. (1999). Typability and type checking in System F are equivalent and undecidable. *Ann. Pure Appl. Logic*, 98(1-3):111–156.

[**MR1696849**] A review for this item is in process.

Xi, H. (1999). Upper bounds for standardizations and an application. *J. Symbolic Logic*, 64(1):291–303.

[MR1683909] A review for this item is in process.

1.11 Proof theory and constructive mathematics

Kuzichev, A. A. and Kuzichev, A. S.. A conservative extension of a formal arithmetic. *Vestnik Moskov. Univ. Ser. I Mat. Mekh.*, 1988(6):77–78.

[MR90j:03027] On the basis of earlier papers [A. A. Kuzichev, Z. Math. Logik Grundlag. Math. 25 (1979), no. 4, 299–314; MR 81a:03058; the authors, Dokl. Akad. Nauk SSSR 250 (1980), no. 6, 1310–1315; MR 82k:03090], the authors establish that an arithmetical formula A is derivable in H$\mathcal{A}$ if and only if the sequent $\Pi N \Rightarrow A$ has an ω_1-derivation.

Kuzičev, A. S. (1978a). The consistency of formal arithmetic. *Dokl. Akad. Nauk SSSR*, 243(5):1123–1126.

[MR80b:03090b] This paper is a new version of the author's earlier consistency proof (same Dokl. 238 (1978), no. 2, 269 - 272; MR 58 #10335) for arithmetic using a system of λ-conversion with additional constants for equality and logical connectives. λ-conversion and Gentzen-type rules for connectives are postulated. Many of the familiar effects of unrestricted comprehension are obtained via diagonal construction. Among these is the interpretation of arithmetic using Dedekind's definition of natural numbers and an (unintended) derivation of contradiction. The author claims that after restricting the notion of derivation in a rather cumbersome way he is still able to interpret arithmetic and to obtain a kind of cut-elimination theorem allowing him to prove the nontriviality of the system (i.e., not every sequent is derivable). The reviewer was unable to follow the proof, but cut-elimination for the translations of arithmetic formulas can be proved in a simpler way using the Kreisel-Takeuti trick: a cut can be replaced by antecedent rules applied to a formula obtained from an antecedent instance of induction. (English translation: Soviet Math. Dokl. 19 (1978), no. 6, 1305 - 1308; ibid. 19 (1978), no. 6, 1486 - 1490.) Reviewed by G. E. Mints.

Kuzičev, A. S. (1978b). Formal arithmetic in combinatorially complete systems. I. *Vestnik Moskov. Univ. Ser. I Mat. Mekh.*, 1978(2):13–21.

[MR80a:03019a] The author adds the usual Gentzen rules for predicate logic with identity to the language of λ-conversion. Using a combinatory analogue of the recursion theorem, he is able not only to introduce the natural numbers and prove the Peano axioms for arithmetical functions, but also to obtain many results of second-order logic. In particular, he introduces an arithmetical operator N, imitating Dedekind's definition of the set of natural numbers as the smallest set containing 0 and closed under the successor function. A deduction of a formula Ain ordinary arithmetic is translated into a deduction of the sequent $N(e_1), \dots, N(e_k) \to A$ in the author's system, where $e_1, \dots, e_k$ are the free variables in A, by practically the same method as in second-order logic. In both parts of the paper, the primary concern is to prove cut elimination for a special class of deductions; this class includes the translations of all deductions in formal arithmetic. Reviewed by G. E. Minc.

Kuzičev, A. S. (1978c). Formal arithmetic in combinatorially complete systems. II. *Vestnik Moskov. Univ. Ser. I Mat. Mekh.*, 1978(3):60–66.
[MR80a:03019b] See (Kuzičev, 1978b).

Kuzičev, A. S. (1978d). Theorem on the mean sequent in the $\dashv$-system of λ-conversion. *Dokl. Akad. Nauk SSSR*, 243(1):19–21.
[MR80b:03090a] From the text of the paper: "We study the $\mathcal{A}$-system of λ-conversion (the author, same journal 236 (1977), no. 5, 1072 - 1075; MR 58 #5082; ibid. 236 (1977), no. 4, 796 - 799; MR 57 272; MR 58 #10335), introduce the concept of $\mathcal{F}$-derivation, and prove a theorem on the mean sequent for $\mathcal{F}$-derivations."

Kuzičev, A. S. (1979). Formal arithmetic in a system of λ-conversion with logical operators. *Z. Math. Logik Grundlag. Math.*, 25(4):299–314.
[MR81a:03058] This paper summarises the author's earlier work (Vestnik Moskov. Univ. Ser. I Mat. Meh. 1978, no. 2, 13 - 21; MR 80a:03019a; ibid. 1978, no. 3, 60 - 66; MR 80a:03019b) on his Gentzen-type system for combinatory logic with additional operators corresponding to logical connectives and quantifiers. A proof of the cut elimination theorem is sketched for a subclass of deductions sufficient to derive arithmetic. This reviewer was unable to follow it, but it seems to use only primitive recursive arithmetic. If the formula proved is of the form $\forall x Nx \rightarrow C$, Nx being Dedekind's definition of natural numbers (the case the author is most interested in) then cut elimination can be proved easily by the Kreisel-Takeuti trick. Reviewed by G. E. Mints.

O'Donnell, M. (1979). A practical programming theorem which is independent of Peano arithmetic. In *Begriffsschrift—Jena Frege Conference (Friedrich-Schiller-Univ., Jena, 1979) (German)*, pages 284–298. Friedrich-Schiller-Univ., Jena.
[MR82i:03066] A programming language has the termination property if all programs not containing unbounded loops or circular function definitions must halt. The author introduces a programming language based on the polymorphic typed λ-calculus of J. C. Reynolds for which the termination property is true, but not provable in Peano arithmetic (plus all true Π_1 sentences of arithmetic). He calls this result "the first example of a natural and practical unprovable statement from computer science or any other concrete science". The proof proceeds by showing that the functions defined by a certain class of programs grow too fast to be provably recursive. A corollary, obtained by using the correspondence between proofs and typed λ-terms [cf. S. Stenlund, Combinators, λ-terms and proof theory, Reidel, Dordrecht, 1972; MR 58 #21492], is that cut elimination (normalization) theorems for propositional logic with quantification over propositional variables are independent of Peano arithmetic (plus all true Π_1 sentences of arithmetic). (For the entire collection see MR 81m:03002.)

Venneri, B. M. (1979). Arbres de déduction naturelle et programmation en LCF: possibilités de "normalisation". In *Les arbres en algèbre et en* *programmation (4ème Colloq., Lille, 1979)*, pages 261–274. Univ. Lille I, Lille.

[MR81c:68074] Author's summary: "This work constitutes an inquiry into proof theory and the use of the logic of computable functions as a programming language. We show that the proofs are trees of 'natural deduction' and that the rules are rules of introduction and elimination, connected by a principle of inversion. Thus it is possible to normalize (i.e., simplify) these trees." (For the entire collection see MR 80j:68004.)

Aczel, P. and Feferman, S. (1980). Consistency of the unrestricted abstraction principle using an intensional equivalence operator. In *To H. B. Curry: essays on combinatory logic, lambda calculus and formalism*, pages 67–98. Academic Press, London.

[MR82b:03037] From the introduction: "The usual explanation of '$\{x \mid \phi(x)\}$' is that it is supposed to denote a class whose members are just those y for which $\phi(y)$ holds. This is here regarded as an 'explanation by definition'. Instead of expressing the unrestricted abstraction (comprehension) principle in the usual inconsistent form (1) $y \in \{x \mid \phi(x)\} \leftrightarrow \phi(y)$ we express it here in the form (2) $y \in \{x \mid \phi(x)\} \equiv \phi(y)$. This makes use of a new connective $\equiv$ in place of the truth-functional biconditional $\leftrightarrow$. In general, $\phi \equiv \psi$ is read as expressing that 'ϕ is equivalent to ψ in consequence of given basic definitions' (in this case (2)). A logic is presented here for $\equiv$ which is simple and plausible under this reasoning. "With each system S in ordinary logic is associated a system S^* based on the laws of $\equiv$ and the abstraction principle of (2) above, together with S. The main result is that if S has only infinite models, then S^* is a conservative extension of S. As a corollary, one has that for empty S the theory S^* is consistent." In one of the appendices, the second author presents a very interesting motivation for the construction of S^*, given S. (For the entire collection see MR 81j:03005.) Reviewed by N. C. A. da Costa.

Bunder, M. W. (1980). The consistency of a higher order predicate calculus and set theory based on combinatory logic. In *Mathematical logic in Latin America (Proc. IV Latin Amer. Sympos. Math. Logic, Santiago, 1978)*, pages 73–82. North-Holland, Amsterdam.

[MR81h:03106] The author considers a version of higher order predicate calculus based on combinatory logic and shows that proofs in the calculus can be normalised in a certain way. Consistency of the calculus follows as a corollary. The author also shows that the addition of extensionality, pairing, comprehension, replacement, sum set, power set and infinity does not affect consistency. (For the entire collection see MR 81d:03002.) Reviewed by Carlo Cellucci.

Diller, J. (1980). Modified realization and the formulae-as-types notion. In *To H. B. Curry: essays on combinatory logic, lambda calculus and formalism*, pages 491–501. Academic Press, London.

[MR82g:03093] The author deals with a natural deduction version of $N - HA^\omega$, formulated along the lines of P. Martin-Löf [Logic Colloquium '73 (Bristol, 1973), pp.

73 - 118, North-Holland, Amsterdam, 1975; MR 52 #7856]. He defines a primitive recursive term a' and a term a^* which proves that a' modified realizes the type of a, i.e. the formula proved by a. The main theorem of the paper says that normalization commutes with both of these operations $'$ and *. The theorem of G. E. Minc [Zap. Nauchn. Sem. Leningrad. Otdel. Mat. Inst. Steklov. (LOMI) 40 (1974), 110 - 118; MR 53 #2646] (who disregards *) follows as a corollary. The author also notes that his result has been extended by M. Stein ["Interpretationen der Heyting-Arithmetik endlicher Typen", Dissertation, Univ. Munster, Munster, 1976] and P. Rath ["Eine verallgemeinerte Funktionalinterpretation der Heyting- Arithmetik", Dissertation, Univ. Münster, Münster, 1978] to other functional interpretations. (For the entire collection see MR 81j:03005.) Reviewed by H. Schwichtenberg.

Howard, W. A. (1980). The formulae-as-types notion of construction. In *To H. B. Curry: essays on combinatory logic, lambda calculus and formalism*, pages 480–490. Academic Press, London.

[MR82g:03094] From the introduction: "The following consists of notes which were privately circulated in 1969. Since they have been referred to a few times in the literature, it seems worth while to publish them. The ultimate goal was to develop a notion of construction suitable for the interpretation of intuitionistic mathematics. The notion of construction developed in the notes is certainly too crude for that, so the use of the word construction is not very appropriate. However, the terminology has been kept in order to preserve the original title and also to preserve the character of the notes." In part I the ideas are illustrated for the intutionistic propositional calculus. By a construction of a sequent $\Gamma \to \beta$ is meant a λ-term F^β of type β such that for every free variable X^α occurring in F^β there is a corresponding occurrence of α in Γ. It is then shown that for any derivation of $Gamma \to \beta$ in positive implicational propositional logic one can find a construction of $\Gamma \to \beta$ and conversely. Also a connection between normalization of terms and cut elimination is established. These ideas are then extended to propositional logic with more connectives, and (in part II) to Heyting arithmetic. In particular, the treatment of $\exists$ appears to be a nontrivial question. As an unpublished preprint, this paper has had a great influence, and hence it is good to see it published at last. Its ideas have been taken up in a number of subsequent papers, maybe most consequently by P. Martin-Löf ["Constructive mathematics and computer programming", Proceedings of the 6th International Congress of Logic, Methodology and Philosophy of Science (Hannover, 1979), North-Holland, Amsterdam, to appear]. (For the entire collection see MR 81j:03005.) Reviewed by H. Schwichtenberg.

Kuzičev, A. A. (1980). On the permissibility of a cut in combinatorially complete systems with equality. *Vestnik Moskov. Univ. Ser. I Mat. Mekh.*, 1980(5):31–34, 84.

[MR81k:03059] Author's summary: "We propose an algorithm for constructing a cut in a Q-system of λ-conversion (Q is the equality operator). This algorithm corresponds to A. S. Kuzichev's proof of the cut elimination theorem (Dokl. Akad.

Nauk SSSR 236 (1977), no. 4, 796 - 799; MR 57 #15983; Vestnik Moskov. Univ. Ser. I Mat. Meh. 1978, no. 2, 13 - 21; MR 80a:03019a; ibid. Ser. I Mat. Meh. 1978, no. 3, 60 - 66; MR 80a:03019b) carried out by induction on the construction of derivations."

Kuzičev, A. A. and Kuzičev, A. S. (1980). The embedding of formal arithmetic in combinatorially complete systems. *Dokl. Akad. Nauk SSSR*, 250(6):1310–1315.

[MR82k:03090] Drawing heavily on earlier results on $\mathcal{A}$-systems by the second author [e.g., same journal 236 (1977), no. 4, 796 - 799; MR 57 #15983; ibid. 236 (1977), no. 5, 1072 - 1075; MR 58 #5082], the authors prove several theorems on a special $\mathcal{A}$-system constructed sequentially with a restriction on the cut rule and forming a fragment of the $\mathcal{A}$-system in which rules 3 and 13 of the $\mathcal{A}$-extension are not used. Using induction, the authors show the typical result that if $\mathfrak{A}$ is a derivation of the sequent $\Gamma \Rightarrow \Theta$ in the calculus $L\mathcal{A}$ of formal arithmetic then one can construct an L-derivation of the sequent $\prod \lambda x.Nx, \Gamma \Rightarrow \Theta$. Using a result from the first author's thesis (written under the guidance of V. A. Uspenskii) on the embedding of formal arithmetic, it is shown that a certain sequent T is not derivable. A corollary of the nonderivability of the sequent T and the typical result given earlier in this review is another proof of the consistency of formal arithmetic in the spirit of Gentzen in view of Gödel's second incompleteness theorem. The reviewer believes that the proof could be streamlined by using distributed logic, in which special $\mathcal{A}$-systems are constructed concurrently rather than sequentially as fragments of the master $\mathcal{A}$-system. (English translation: Soviet Math. Dokl. 21 (1980), no. 1, 311 - 316.) Reviewed by A. A. Mullin.

Kuzichev, A. S. (1980). On consistency proofs for first-order theories. In *History and methodology of the natural sciences, No. XXV (Russian)*, pages 97–103. Moskov. Gos. Univ., Moscow.

[MR83m:03065] Author's summary: "This paper contains the results of an analysis of type-free extensions of λ-conversion and pure combinatory logic, constructed sequentially with a restriction on the cut rule. It is shown that the apparatus of these extensions can be used to prove the consistency of calculi of first order." (For the entire collection see MR 82c:01004.)

Lambek, J. (1980). From λ-calculus to Cartesian closed categories. In *To H. B. Curry: essays on combinatory logic, lambda calculus and formalism*, pages 375–402. Academic Press, London.

[MR81m:03020] This is a very detailed and readable exposition of now familiar connections between basic notions of combinatory logic, proof theory and category theory. No previous knowledge of the first two fields and very little of the third is required. The main attention is devoted to various aspects of the possibility of defining lambda -abstraction (called the functional completeness theorem in combinatory logic) which is equivalent to the deduction theorem for positive intuitionistic propositional calculus. The notion of prerecursive category (Cartesian-closed category with natural number object N) is defined and a result due to M. Tibault is mentioned:

The set F of functions represented by morphisms $N^k \to N$ in the free prerecursive category generated by the empty category coincides with the set of type-recursive functions discussed by Grzegorczyk. This seems to imply that F is essentially the set of Gödel's functions of finite type. (For the entire collection see MR 81j:03005.) Reviewed by G. E. Mints.

Meredith, D. (1980). A positive logic proof procedure. In *To H. B. Curry: essays on combinatory logic, lambda calculus and formalism*, pages 503–510. Academic Press, London.

[MR82a:03014] From the text: "We present a procedure which has applications in both propositional and combinatory logic. In the former it provides a proof procedure for the Hilbert positive implicational logic which involves no reduction to canonical form, and which gives proofs which are very concise. In the latter it provides a method for finding, for any functional type isomorphic with a positive logic thesis, a combinator of that type." (For the entire collection see MR 81j:03005.)

Kuzichev, A. S. (1981). Arithmetical theories constructed on the basis of λ-conversion. *Dokl. Akad. Nauk SSSR*, 261(4):792–796.

[MR83c:03053] In the first part of the paper a deductive extension Asub0 of the calculus of lambda-conversion is introduced. It continues the studies of the author on lambda-conversion from a series of his former works in the field. The main result of this part is the consistency of the system Asub0 (Theorem 1). In the second part of the paper the definition of an arithmetical lambda-theory is given by taking lambda-conversion as a base. Then the author obtains some properties of such lambda-theories. The main ones are: inner consistency (Theorem 3) and a section property (Theorem 2). In the last (third) part of the paper an example of an arithmetical lambda- theory is constructed and it is shown that the formal arithmetic HA is embeddable in this lambda-theory (Theorem 4). (English translation: Soviet Math. Dokl. 24 (1981), no. 3, 584 - 589 (1982).) Reviewed by M. S. Burgin.

Bunder, M. W. (1982). A formal metatheory in which Gödel's first but not his second incompleteness theorem can be proved. *J. Non-Classical Logic*, 1(1):43–69.

[MR84j:03112] This paper presents in great detail a formal metatheory M such that Gödel's first incompleteness theorem can be formalized in M. The author then goes on to discuss whether it is possible to get a formalized version of the standard proof (he refers to the book by E. Mendelson [Introduction to mathematical logic, Van Nostrand Reinhold, New York, 1964; MR 29 # 2158]) of Gödel's second incompleteness theorem within M. However, no underivability result for M is obtained; the author just reports about obstacles to formalizing the standard proof. Reviewed by H. Schwichtenberg.

Kuzichev, A. S. (1983a). Arithmetic completeness of type-free logic. *Dokl. Akad. Nauk SSSR*, 270(6):1323–1327.

[MR85d:03025] The type-free logic A_0 introduced in a previous paper by the author [Z. Math. Logik Grundlag. Math. 25 (1979), no. 4, 299–314; MR 81a:03058] as a sequential calculus with the restricted cut rule and unbounded abstraction is investigated. The author defines a certain subclass AL of theorems of A_0 and shows that it contains all theorems of the systems LA and HA of formal arithmetic, which were defined in the above-cited paper.

English translation: Soviet Math. Dokl. 27 (1983), no. 3, 767–772. Reviewed by P. Stepanek.

Kuzichev, A. S. (1983b). Arithmetically consistent λ-theories of type-free logic. *Dokl. Akad. Nauk SSSR*, 268(2):288–292.

[MR85j:03015a] In earlier papers, the author [same journal 261 (1981), no. 4, 792–796; MR 83c:03053; ibid. 262 (1982), no. 4, 795–799; MR 83i:03027; ibid. 266 (1982), no. 1, 23–27; MR 84c:03032] considered the problem of constructing axiomatic theories by means of Church's λ-conversion and pure combinatory logic. He suggested that type-free logics could be used to give syntactic proofs of the consistency of various Hilbert-type calculi. In the paper under review, he investigates further the class of L-derivations [op. cit.; MR 83i:03027] and syntactic proofs of the consistency of formal arithmetic. Five typographical errors are given in the correction. English translation of the original: Soviet Math. Dokl. 27 (1983), no. 1, 72–77.

Kuzichev, A. S. (1983c). Correction: "Arithmetically consistent λ-theories of type-free logic". *Dokl. Akad. Nauk SSSR*, 272(1):10.

[MR85j:03015b] See MR 85j:03015a.

Kuzichev, A. A. (1984). The representation of arithmetic in combinatorially complete systems. *Dokl. Akad. Nauk SSSR*, 278(3):537–541.

[MR86c:03011] Author introduction (translated from the Russian): "The author and A. S. Kuzichev [same journal 250 (1980), no. 6, 1310–1315; MR 82k:03090] established a connection between formal arithmetic and a class of L-derivations. The natural numbers and arithmetic operations are represented in L-derivations by objects $Z_n, \sigma, [+], [], \pi$ and π_0 [see H. P. Barendregt , The lambda calculus. Its syntax and semantics, North-Holland, Amsterdam, 1981; MR 83b:03016; A. S. Kuzichev, Z. Math. Logik Grundlag. Math. 25 (1979), no. 4, 299–314; MR 81a:03058]. In this paper we propose a representation of natural numbers in a more general form, introducing the notion of an arithmetical signature Ω with respect to which a class of $L(\Omega)$-derivations is defined. We prove that this class is consistent with arithmetic."

English translation: Soviet Math. Dokl. 30 (1984), no. 2, 417–421.

Bezem, M. (1985). Strong normalization of bar recursive terms without using infinite terms. *Arch. Math. Logik Grundlag.*, 25(3-4):175–181.

[MR87k:03071] The author gives a new proof of the strong normalization theorem for bar recursive terms, using, instead of continuity, W. Howard's compactness of functionals of finite type. Infinite terms are not needed. Reviewed by A. S. Kuzichev.

Päppinghaus, P. (1985). A typed λ-calculus and Girard's model of ptykes. In *Foundations of logic and linguistics (Salzburg, 1983)*, pages 245–279. Plenum, New York.

[MR87h:03022] From the introduction: "In a fundamental paper inaugurating Π^1_2-logic, J.-Y. Girard introduced the notion of a dilator as a functor from the category of ordinals into itself preserving direct limits and pullbacks. In Chapter 12 of a forthcoming book [Proof theory and logical complexity, Bibliopolis, Naples, to appear; per bibl.] he generalizes this notion to all finite types. For every finite type σ a category PT^σ is defined whose objects form a (proper) class Pt^σ and are called ptykes of type σ. In his book Girard proves the ptykes of finite types to be a model of a variant T′ of Gödel's T. Using this new model he proposes an intrinsic 'ordinal assignment' to terms of T′. To (closed) terms of type $o \to o$, in particular, their value at ω is assigned. We are interested in determining these values. From earlier work on theories of functionals of finite type over the countable ordinals, the following answer to this question is to be expected: $\{t\omega\colon t$ is a closed T′-term of type $o \to o\} = \eta_0$, where $\eta_0 = \varphi_{\varepsilon_{\Omega+1}}(0)$, the so-called Bachmann-Howard ordinal. The aim of this paper is to prove the inclusion '$\subseteq$' of the equation above. The reverse inclusion will be proved elsewhere.

"The types and the categories PT^σ are inductively defined as follows: (i) o is a type, and PT^o is the category ON of ordinals. Pt^o is the class On of all ordinals. For $x, y \in \mathrm{Pt}^o$ the set of morphisms $I^o(x,y)$ is the set of all order-preserving maps from x to y. (ii) If σ and τ are types, then $(\sigma \to \tau)$ is a type. $\mathrm{Pt}^{\sigma\to\tau}$ is the class of all functors from PT^σ to PT^τ preserving direct limits and pullbacks. For $A, B \in \mathrm{Pt}^{\sigma\to\tau}$ the set of morphisms $I^{\sigma\to\tau}(A,B)$ is the set of all natural transformations from A to B. (iii) If σ and τ are types, then $(\sigma \times \tau)$ is a type. $\mathrm{Pt}^{\sigma\times\tau} =_{\mathrm{df}} \mathrm{Pt}^\sigma \times \mathrm{Pt}^\tau$, and $I^{\sigma\times\tau}((A,B),(A',B')) =_{\mathrm{df}} I^\sigma(A,A') \times I^\tau(B,B')$.

"In order to prove the result we set up a typed λ-calculus of ordinal operator terms, λ-OT, which extends T′, and define a notion of reduction reflecting the defining equations of the constants."

The terms of this system are ordinary typed λ-terms with additional constants. In addition to the constants for pairing and projection functions, there are constants for the ordinal zero, successor, addition, primitive recursion, and supremum. Because only nondecreasing functions can be induced by ptykes of type $o \to o$, the recursion equations have been modified. If R_σ is the recursion operator of type σ, then the equations used by the author are $R_\sigma ab0 = a$, $R_\sigma ab(x+1) = R_\sigma abx +_\sigma b(R_\sigma abx)x$, where $+_\sigma$ is a ptyx of type $\sigma \to \sigma \to \sigma$ defined in terms of the other constants.

From the introduction again: "By routine methods we prove this notion of reduction to be Church-Rosser and strongly normalizing. A closed term of type o in normal form is of one of the following forms: $0, St$ with t a closed term of type o in normal form, or $\sup_{x<\alpha} t[x]$, where α is a limit ordinal and $t[x]$ a normal term of type o. So we can use the normal form of a closed term of type o to determine $|t|$, the intended value of t, inductively: $|0| =_{\mathrm{df}} 0, |St| =_{\mathrm{df}} |t|+1$, and $|\sup_{x<\alpha} t[x]| =_{\mathrm{df}} \sup_{\beta<\alpha} |\mathrm{NF}(t[\underline{\beta}])|$, where $\underline{\beta}$ is a closed term of type o denoting β. This definition is by induction on an inductively defined class OC_0 of 'ordinally computable' closed terms of type o. We generalize the notion of ordinal computability to all finite types and prove that every closed term is ordinally computable. This proof is modeled on Zucker's proof of computability of the closed terms of his theory T_2 of trees. In order to obtain a bound for $|t|$, we restrict the system λ-OT to terms of the form $\sup_{x<\alpha} t$ with $\alpha < \Lambda$ for some fixed $\Lambda \le \eta_0$, and observe that for this restricted system the proof of '$t \in \mathrm{OC}_0$' can be formalized in the theory ID_1 of one inductive definition. By applying the proof theory of ID_1, it follows that for terms of the restricted system, $|t|$ is bounded above by η_0. The same bound holds for the value of t in an arbitrary canonical model of λ-OT (i.e., a model where the domain of type o is an ordinal). For it is easy to see that in such models $|t|$ equals the value of t in the model. Unfortunately, the type structure of the ptykes is not a model of the full language of λ-OT. This is because the sup of a family of ptykes does not always exist. This difficulty is overcome by restricting λ-OT to a suitable syntactically defined subclass of terms. We show that the ptykes are a model of the resulting subsystem PT $< \Lambda$ of λ-OT. Together these results establish that η_0 is closed under primitive recursive ptykes of type $o \to o \to \cdots \to o$." For the entire collection see MR 86h:03003. Reviewed by J. P. Seldin.

Bunder, M. W. (1986). Corrections to: "A formal metatheory in which Gödel's first but not his second incompleteness theorem can be proved" [J. Non-Classical Logic **1** (1982), no. 1, 43–69; MR 84j:03112]. *J. Non-Classical Logic*, 3(1):69–72.

[MR87m:03092] The author lists two major changes to pp. 63–65 of the cited paper (which do not affect the conclusions reached) and some further minor changes elsewhere, and adds two remarks based on private communications from S. Feferman and from J. P. Seldin.

Troelstra, A. S. (1986). Strong normalization for typed terms with surjective pairing. *Notre Dame J. Formal Logic*, 27(4):547–550.

[MR87m:03081] This elegant paper is built up around the following simple observation. Let $t \mapsto t^*$ be the familiar translation of types with product into (finite sequences of) pure functional types given by $0^* = 0, (s \times t)^* = (s^*, t^*), (s \to t)^* = ((s)^* t_1, \cdots, (s)^* t_m)$ for $t^* = t_1, \cdots, t_m$. Then $\mapsto$ preserves the reduction relation not only for lambda abstraction and (simultaneous) recursion, but also for the surjective pairing. The latter is important in various applications of proof theory, for example to category theory. The resulting proof of strong normalisation seems to be simpler than previous ones and applies to a wide class of systems. Reviewed by G. E. Mints.

de Vrijer, R. (1987). Strong normalization in N-HA$_p^\omega$. *Nederl. Akad. Wetensch. Indag. Math.*, 49(4):473–478.

[MR89g:03020] From the introduction: "The purpose of this note is to amend the notion of computability employed in proofs of strong normalization, so as to make it work also for system which include a surjective pairing. To facilitate comparison with existing proofs (i.e. for systems without surjective pairing), we work with the system N-HA$_p^\omega$ as formulated in Metamathematical investigation of intuitionistic arithmetic and analysis [edited by A. S. Troelsta, Lecture Notes in Math., 344, Springer, Berlin, 1973; MR 48 #3699], and roughly follow the setup of the proof of strong normalization of N-HA$^\omega$ given there.

"The new definition of computability has an extra clause for the product types, of course. But apart from that, it deviates from the usual definitions in two more respects. First, the computable terms are explicitly required to be strongly normalizing. Secondly, by the very form of the computability conditions it is enforced at once that computability is preserved under reduction."

Fitch, F. B. (1987). Formalized self-reference. In *Self-reference*, pages 113–122. Nijhoff, Dordrecht.

[MR89e:03020] From the introduction: "We constructed [see J. Symbolic Logic 45 (1980), no. 3, 529–543; MR 82i:03023] a system of logic CΓ which can be shown to be consistent if the rule for '*eta*' is omitted, and within which at least some aspects of self-reference can be represented. We now sketch this logic and show how it deals with the Russell paradox and the Grelling paradox." For the entire collection see MR 88k:03006. Reviewed by G. E. Mints.

Flagg, R. C. and Myhill, J. (1987a). An extension of Frege structures. In *Mathematical logic and theoretical computer science (College Park, Md., 1984–1985)*, pages 197–217. Dekker, New York.

[MR89i:03027] From the introduction: "In recent years much attention has been given to type-free foundations for mathematics (see in particular a paper by P. Aczel [in The Kleene Symposium (Madison, WI, 1978), 31–59, North-Holland, Amsterdam, 1980; MR 82e:03045]). While the existent systems permit unlimited self-reference and are demonstrably consistent, the mathematics they contain is rather weak; it consists, roughly speaking, of the mathematics of E. Bishop [Foundations of constructive analysis, McGraw-Hill, New York, 1967; MR 36 #4930] plus many instances of excluded middle. Strengthening of these systems so far has yielded theories of hyperarithmetical reals [P. Lorenzen and Myhill, J. Symbolic Logic 24 (1959), 37–49; MR 22 #14] which not only still fall short of classical analysis, but involve tortuous technical complications which prevent any sizable body of classical mathematics being taken over directly. What we would like is a self-referential system extending ZF and provably consistent relative to some well-known 'slight' extension of ZF.

"We do not yet know how to solve this problem; in this paper we make a first step. Specifically we start with axioms for a 'Frege structure' in approximately the sense of Aczel [op. cit.], and then adjoin a comprehension rule and a principle of choice. The resulting system extends classical second-order arithmetic with full comprehension and the axiom of dependent choices. Consistency is proved using Scott's lattice-theoretic models of the lambda calculus." For the entire collection see MR 88i:03005.

Flagg, R. C. and Myhill, J. (1987b). Implication and analysis in classical Frege structures. *Ann. Pure Appl. Logic*, 34(1):33–85.

[MR89f:03046] The authors construct a foundational system for mathematics on the Fregean concept of property, i.e. such that for any formula $\Phi(x)$ there is a property Ψ such that $\Phi(M)$ and "M has property Ψ" are interdeducible for every M. Systems have been constructed on this basis by Fitch and Aczel. According to the authors the most reasonable foundation of classical mathematics based on Frege's concept of property is to define reals as Dedekind cuts and to use a hierarchy of implications. Such a system was proposed in 1975 by Myhill, without a consistency proof, which is now supplied by exploiting Aczel's notion of Frege structure. In this paper the authors develop the proof theory and model theory of classical Frege structures, formalize the hierarchy of implications, prove the consistency of the system and outline the resulting analysis with Dedekind reals, which resembles to a large extent classical (elementary) analysis, and with Cauchy reals, which resembles recursive analysis. Reviewed by B. van Rootselaar.

Bezem, M. (1988). Equivalence of bar recursors in the theory of functionals of finite type. *Arch. Math. Logic*, 27(2):149–160.

[MR90k:03061] This is a technical paper about bar recursion in the intuitionistic theory of finite-type functionals. Bar recursion is a principle that allows definition of functions by recursion over a well-founded tree of finite sequences of functionals. This can be represented in the type theory by introducing a bar-recursion combinator whose role is to take the tree as argument and produce the function as result (in much the same spirit as an ordinary recursion combinator). Perhaps it is not surprising that there are several different formalizations of this idea in the literature. They differ, for example, in the types of the functions allowed as arguments to the combinator, and in the representation chosen for finite sequences. The main aim of this paper is to show the formal equivalence of five of these different formalizations. It also contains an interesting lemma on the equivalence of two forms of the rule of extensionality. Reviewed by Edmund Robinson.

de Queiroz, R. J. G. B. (1988). A proof-theoretic account of programming and the role of reduction rules. *Dialectica*, 42(4):265–282.

[MR90b:03090] This paper is an important contribution to natural deduction theory, where rules in a proof system can be divided into introduction, reduction and elimination rules. The author argues convincingly that the meaning of a logical connective is completely given by its introduction and reduction rules, which tell when and how a connective can be used. Assigning a more prominent role to reduction rules, the author gives a more natural general method of deriving elimination rules than that given by Prawitz and improved by Schroeder-Heister. The connection with the lambda calculus and Aczel's Frege system is described in the first part of the paper; in the second, Martin-Löf's intuitionistic type theory is illuminated by the new approach. Quoting from the conclusions, "Taking our point of view on the importance of the 'reduction' rules, and the insufficiency of the 'introduction' rules, one can understand why the language of category theory has gained a certain popularity within the computer science community: it makes available in the language not only each symbol to be 'introduced' but also its projections and its corresponding 'reduction'. That is probably what is missing in proof theory in order to make it a 'comfortable' language to express constructions. We see the work of Martin-Löf as a remarkable step towards this objective." Reviewed by B. H. Mayoh.

Girard, J.-Y., Taylor, P., and Lafont, Y. (1989). *Proofs and types.* Cambridge University Press, Cambridge.

[MR90g:03013] Beginning with a sketch of Tarski's and Heyting's semantics, the book aims at an analogy between propositions and types. First comes an introduction to natural deduction. In the view of Heyting's semantics, a proof of a proposition A is identified as an object of type A, and the rules of natural deduction as constructions of a function. This is the Curry-Howard isomorphism.

This book presents a formal system of typed terms for discussing the functional objects which correspond to proofs. This simple typed calculus has too little expressive power to present the integers or the Booleans. Then the authors discuss Gödel's system T, which has enough expressive power. For instance, let f be a function from $\mathbf{N}$ to $\mathbf{N}$; then f is representable in T if and only if f is a recursive function which is provably total in PA. Although the expressive power of the system T is enormous, it is still possible to enrich the expressive power. The book presents the system F, introduced by Girard in 1971, which has extreme expressive power. All kinds of computer data—lists, trees and pairs—are definable in F. The book deals extensively with the definability of usual types in F. From the point of view of the Curry-Howard isomorphism between typed systems and natural deduction, F corresponds to second-order intuitionistic logic. Another topic discussed is the strong normalisation theorems for the simple typed calculus, the system T and the system F. These theorems are proved by improving the method of Tait. There are also chapters devoted to Gentzen's sequent calculus and cut elimination. A detailed account of semantics is given. The semantics, called coherence semantics, is a new kind of domain theory and is based on category theory. It is a refinement of Scott's domain theory. As compared with Scott's topological semantics, the coherence semantics is simple and easy to handle. In the book, the coherence semantics interprets the simple typed calculus, the system T and the system F. It is worthwhile to note that the interpretation of the sum type leads to the origin of linear logic, which is a new approach to constructive logics and programming. The book will be helpful to anyone who studies linear logic.

Summing up, the book is well written and an excellent reference. It will be useful for anyone interested in proof theory and computer science; researchers and students in this area should thank the authors for their great effort. Reviewed by Fangming Song.

López-Escobar, E. G. K. (1990). Remarks on the Church-Rosser property. *J. Symbolic Logic*, 55(1):106–112.

[MR90m:03097] It is well known that to the proof of every theorem of first-order intuitionistic propositional logic there corresponds, via the Curry-Howard isomorphism, a λ-term and that reduction of the λ-term to normal form corresponds to normalisation of the proof.

The author of this paper introduces a more general notion of reduction (as in the work of Klop and others) that satisfies the Church-Rosser property, and applies this to give a direct proof of the strong normalisation theorem for the (impredicative) second-order intuitionistic propositional calculus. Reviewed by Martin W. Bunder.

Ono, H. (1990). Structural rules and a logical hierarchy. In *Mathematical logic*, pages 95–104. Plenum, New York.

[MR91j:03073] From the introduction: "Gentzen-type sequent calculi usually contain three structural rules, viz., exchange, contraction and weakening rules. In recent years, however, there have been various studies on logics that have omitted some or all of these structural rules. The motives or purposes of these studies have been so diverse that sometimes close connections between them have been overlooked. Here we give a brief survey of recent results on these logics in an attempt to make these interrelationships clearer." For the entire collection see MR 91h:03007.

Renardel de Lavalette, G. R. (1990). Extended bar induction in applicative theories. *Ann. Pure Appl. Logic*, 50(2):139–189.

[MR91i:03117] Summary: "TAPP is a total applicative theory, conservative over intuitionistic arithmetic. In this paper, we first show that the same holds for TAPP + the choice principle EAC; then we extend TAPP with choice sequences and study the principle EBI_0^a (arithmetical extended bar induction of type zero). The resulting theories are used to characterise the arithmetical fragment of EL (elementary intuitionistic analysis) + EBI_0^a. As a digression, we use TAPP to show that P. Martin-Löf's basic extensional theory $\mathbf{ML}_0$ is conservative over intuitionistic arithmetic."

Avron, A. (1991). Hypersequents, logical consequence and intermediate logics for concurrency. *Ann. Math. Artificial Intelligence*, 4(3-4):225–248.

[MR93a:03028] Using hypersequents, i.e., sequences of ordinary sequents, the author constructs cut-free Gentzen-type systems for some propositional logics (in particular, Dummett's system LC) which are intermediate between intuitionistic and classical ones. The author believes that these logics could serve as bases for parallel λ-calculi and tries to interpret them in terms of parallel computation. Reviewed by Michael V. Zakharyashchev.

McCarty, D. C. (1991). Polymorphism and apartness. *Notre Dame J. Formal Logic*, 32(4):513–531 (1992).

[MR93b:03110] The author constructs an intuitionistic model for the second-order polymorphic lambda calculus. Such a model is constructed, working within the Kleene realizability universe, as a small category of sets which, among other things, allows apartness (the concept of apartness is an intuitionistic positivization of the classical notion of inequality). It is also shown that the construction presented is different from another known construction of the model, which is based on modest sets. Reviewed by A. Kucera.

Smullyan, R. M. (1991). Some unifying fixed point principles. *Studia Logica*, 50(1):129–141.

[MR93b:03001] The author displays his virtuosity as a magician by deriving Gödel's first incompleteness theorem, Tarski's theorem on the undefinability of truth, and three forms of the second recursion theorem, among other results, from a series of increasingly general fixed point principles with increasingly trivial proofs, and ultimately from the following totally trivial Theorem 4.3: A sufficient condition for an arbitrary binary relation R on an arbitrary set N to have a fixed point (that is, an element p of N such that $R(p,p)$ holds) is that there exist a mapping $d\colon N \to N$ and a binary relation S on N such that (i) $S(d(b),b)$ holds for some b in N, and (ii) $S(x,y)$ implies $R(x,d(y))$ for all x and y in N. All students and teachers of recursion theory and metamathematics will enjoy reading this illuminating paper.

Reviewer's remarks: The simplification process can be carried even further by eliminating the auxiliary relation S. Theorem 5: A sufficient condition for a binary relation Q on a set N to have a fixed point is that there exists a mapping $d\colon N \to N$ such that (iii) $Q(d(b),b)$ holds for some b in N, and (iv) $Q(d(x),x)$ implies $Q(d(x),d(x))$ for all x in N. The proof is like the proof for Theorem 4.3. We can derive Theorem 4.3 from Theorem 5 by assuming the hypothesis of Theorem 4.3, defining

$$Q = \{\langle d(x), d(x)\rangle\colon\ R(d(x),d(x))\} \cup \{\langle d(x), x\rangle\colon\ S(d(x),x)\},$$

verifying (iii) and (iv), and showing that Q and R have the same fixed points. Conversely, we can derive Theorem 5 from Theorem 4.3 by defining $R = \{\langle d(x), d(x)\rangle\colon\ Q(d(x),d(x))\}$ and $S = \{\langle d(x), x\rangle\colon\ Q(d(x),x)\}$. The sufficient conditions for a fixed point in Theorems 4.3 and 5 are necessary as well: choose d to be the identity mapping and S to be R. Reviewed by Perry Smith.

Di Cosmo, R. and Longo, G. (1992). Constructively equivalent propositions and isomorphisms of objects, or terms as natural transformations. In *Logic from computer science (Berkeley, CA, 1989)*, pages 73–94.

Springer, New York.
[**MR1236006**] This item will not be reviewed individually. For details of the collection in which this item appears see 94c:03004.

Feferman, S. (1992). Logics for termination and correctness of functional programs. II. Logics of strength PRA. In *Proof theory (Leeds, 1990)*, pages 195–225. Cambridge Univ. Press, Cambridge.
[**MR95a:68076**] Introduction: "We continue the work of Part I [in Logic from computer science (Berkeley, CA, 1989), 95–127, Springer, New York, 1992; see MR 94c:03004]. More specifically, we take up the logics of strength PRA introduced in 19, but in a modified form, more suitable to practice. While the present paper is designed to be read independently, the reader is advised to look at Part I (and its predecessor [S. Feferman, in Logic and computation (Pittsburgh, PA, 1987), 101–136, Contemp. Math., 106, Amer. Math. Soc., Providence, RI, 1990; MR 92c:03014]) for motivational discussions.

"For simplicity in the following, only logics for implicit polymorphism ('Λ' logics) are considered, not explicit polymorphisms ('Λ' logics) as in the paper just cited and Part I, 18. Also (again for simplicity), we do not include the relation $\dot{\epsilon}$ for the representation of types by objects. In place of the logics $\mathrm{QL}_0(\lambda)$ and $\mathrm{PL}_0(\lambda)$ we introduce here new systems $\mathrm{QL}(\mathrm{F}_0\text{-IR})$ and $\mathrm{PL}(\mathrm{F}_0\text{-IR})$, also shown to be of strength PRA. The main modification is in the replacement of the induction axiom for N, IA_N, by a function-induction rule for N, $\mathrm{F}_0\text{-IR}_N$. From this we also derive a rule F_0-IR_T for recursively generated types T." For the entire collection see MR 94b:03008.

Gabbay, D. M. and de Queiroz, R. J. G. B. (1992). Extending the Curry-Howard interpretation to linear, relevant and other resource logics. *J. Symbolic Logic*, 57(4):1319–1365.
[**MR95c:03139**] IT is known (G. H. Helman) that relevance restriction for implication corresponds to the following condition: x actually occurs freely in t for the term λxt corresponding (under the Curry-Howard interpretation) to $\rightarrow$-introduction. Similarly, linear restriction (relevance + no contraction rule) is as follows: x has exactly one free occurrence in t (J.-Y. Girard). The authors note several similar restrictions for less popular logics. The paper contains numerous motivations and semiformal explanations. Reviewed by G. E. Mints.

Girard, J.-Y., Scedrov, A., and Scott, P. J. (1992). Normal forms and cut-free proofs as natural transformations. In *Logic from computer science (Berkeley, CA, 1989)*, pages 217–241. Springer, New York.
[**MR96e:03065**] In this paper the authors address some relationships between the fundamental notions of (1) cut-elimination (proof-theory), (2) natural transformations (category theory) and (3) parametricity (foundations of polymorphism). It is shown that, given a typed λ-calculus $\mathcal{L}$ and a CCC $\mathcal{C}$, a type $\tau(\alpha_1, \cdots, \alpha_n)$ containing type-variables α_i induces a multivariant ('definable') functor $[\tau]$: $(\mathcal{C}^o)^n \times \mathcal{C}^n \rightarrow \mathcal{C}$, and any legal typing judgement $\overline{x} \colon \overline{\sigma} \vdash t \colon \tau$ induces a family of lambda-terms which is a ('definable') dinatural transformation $[\sigma_1] \times \cdots \times [\sigma_n] \rightarrow [\tau]$; $\beta\eta$-equal terms result in the same family. These definable functors and dinatural transformations between them form a CCC.

The proof that the families induced by terms yield dinatural transformations that

moreover compose to form a category proceeds via Gentzen's sequent calculus and cut elimination.

By specializing to the CCC freely generated by $\mathcal{L}$ one obtains as a corollary that arbitrary terms t, when validly typed, in addition to the usual equations of the typed λ-calculus, provably satisfy the appropriate dinaturality equations.

Reviewer's remarks: The published version of the paper is almost unreadable in places. The correct original version of the paper is available by anonymous ftp, e.g. at theory.doc.ic.ac.uk/theory/papers/Scott, or on the WWW, at http://www.csi.uottawa.ca/phil/extra/papers. For the entire collection see MR 94c:03004. Reviewed by Harold Schellinx.

Parigot, M. (1992). $\lambda\mu$-calculus: an algorithmic interpretation of classical natural deduction. In *Logic programming and automated reasoning (St. Petersburg, 1992)*, pages 190–201. Springer, Berlin.
[MR94g:03029]

The λ-calculus is intended to embody the algorithmic content of cut elimination in a sequent form of natural deduction calculus for classical logic. The logic involved has sequents of the form $\Gamma \vdash \Delta$, where Γ and Δ are sets of formulae, and, being a natural deduction calculus, has introduction and elimination rules for formulae on the right. The λ-calculus is an extension of the lambda calculus in which there are two forms of variables, denoting individuals and labels, respectively. A term t can be labelled: $[\alpha]t$, and labels can be bound: $\alpha.t$. The interesting part of the reduction system for this calculus is the rule of "structural reduction": $(\beta.u)v \triangleright_c \beta.[[\beta](wv)/[\beta]w]$, which substitutes wv for all terms w in u labelled by β. To connect the λ-calculus to the sequent calculus, there is a typing system in which formulae to the left of the symbol $\vdash$ are indexed with individual variables, and formulae to the right are indexed with labels (with the exception that a single right-hand formula is allowed to remain unindexed). A proof of $\Gamma \vdash \Delta$ is then associated with a term in the λ-calculus which has those free variables. As expected, reduction in the calculus corresponds to cut elimination. A proof of confluence is given and a subject reduction result is stated, but there is no mention of strong normalisation.

This work falls between the "continuation-passing" approach of Griffin and Murthy, and the more mathematically motivated approach of Girard. Connections with the former are indicated, but they are not really developed, and there is no mention of the latter in the paper. Moreover, although there are a few scattered remarks, the author does not develop a clear intuitive picture of the role of his new sort of variables, or the part they play in the dynamics of execution. For the entire collection see MR 94c:68004. Reviewed by Edmund Robinson.

Streicher, T. (1992). Independence of the induction principle and the axiom of choice in the pure calculus of constructions. *Theoret. Comput. Sci.*, 103(2):395–408.
[MR93k:03013] Summary: "We define a model of the pure Calculus of Constructions (CoC) in which the induction principle over data types defined by second-order quantification and several versions of the axiom of choice are not valid. Instead the model validates the nonconstructive principle of proof by contradiction, i.e., in the model under consideration the proposition

$$(\forall p : \mathrm{Prop})\,((p \to \bot) \to \bot) \to p$$

contains a proof object (where $\perp$ is the empty proposition $(\forall p : \mathrm{Prop})\, p$, representing falsity)."

Swarup, V. and Reddy, U. S. (1992). A logical view of assignments. In *Constructivity in computer science (San Antonio, TX, 1991)*, pages 131–149. Springer, Berlin.

[MR1250671] This item will not be reviewed individually. For details of the collection in which this item appears see 94e:68007.

Tonino, H. and Fujita, K.-e. (1992). On the adequacy of representing higher order intuitionistic logic as a pure type system. *Ann. Pure Appl. Logic*, 57(3):251–276.

[MR93c:03011] The authors define a formalisation of higher-order many-sorted intuitionistic predicate calculus PRED^ω, in which there is a basic sort of propositions, further sorts to denote mathematical objects, and terms that are allowed to range over higher-order (though not polymorphic) types. They also define a type system $\lambda\mathrm{PRED}^\omega$ as a generalised type system in the style initiated by Barendregt with his cube, and subsequently formalised. They prove correspondence between proofs in the one system, and terms in the other, in some detail. This allows them to be admirably explicit about the limitations of the correspondence, notably as regards constants. Reviewed by Edmund Robinson.

Abramsky, S. (1993). Computational interpretations of linear logic. *Theoret. Comput. Sci.*, 111(1-2):3–57. 6th Workshop on the Mathematical Foundations of Programming Semantics (Kingston, ON, 1990).

[MR94e:03062] In this paper the author uses the mathematical framework of the well-known Curry-Howard isomorphism to apply Girard's linear logic to computation, and thus establish a concrete computational interpretation of the logic. In the case of intuitionistic linear logic, he refines the lambda calculus (which then allows greater control over computational behavior), gives an operational semantics for this language in the natural semantics style, applies it to static program analysis and optimization, gives a description of an implementation by a variant of the SECD machine, and then establishes some (new) basic theoretical properties of the calculus in its second-order propositional version. In the classical case, the author introduces a kind of generalized term calculus, a calculus of proof expressions replacing the calculus of Girard's proof nets, combining a term notion for proofs in the one-sided sequent calculus with a list of cuts occurring in the proof and the symmetry of classical linear logic with the usual conceptual apparatus for evaluating expressions in term calculi, this being one of the most important novel features of this paper. Then he presents a concurrent operational semantics in the style of G. Berry and G. Boudol's chemical abstract machine [Theoret. Comput. Sci. 96 (1992), no. 1, 217–248; MR 93e:68018]. In conclusion, he sketches a parallel implementation.

It is worth noting that in recent years in "the mathematization of algorithmics" J.-Y. Girard's geometrical approach [in Categories in computer science and logic (Boulder, CO, 1987), 69–108, Contemp. Math., 92, Amer. Math. Soc., Providence, RI, 1989; MR 90e:03072; in Logic Colloquium '88 (Padova, 1988), 221–260, North-Holland, Amsterdam, 1989; MR 91f:03054; in COLOG-88 (Tallinn, 1988), 76–93, Lecture Notes in Comput. Sci., 417, Springer, Berlin, 1990; MR 91k:03076] has been used. As the author suggests, further separate research establishing connections with Girard's "geometry of interaction" is required, while this paper may be enjoyed as an interesting complement to it. For the entire collection see MR 93k:68006. Reviewed by R. Gylys.

Altenkirch, T. (1993). A formalization of the strong normalization proof for system F in LEGO. In *Typed lambda calculi and applications (Utrecht, 1993)*, pages 13–28. Springer, Berlin.

[MR94g:03105] Summary: "We describe a complete formalization of a strong normalization proof for the Curry style presentation of system F in LEGO. The underlying type theory is the calculus of constructions enriched by inductive types. The proof follows that of J.-Y. Girard, P. Taylor and Y. Lafont [Proofs and types, Cambridge Univ. Press, Cambridge, 1989; MR 90g:03013], i.e., we use the notion of candidates of reducibility, but we make essential use of general inductive types to simplify the presentation. We discuss extensions and variations of the proof: the extraction of a normalization function, the use of saturated sets instead of candidates, and the extension to a Church style presentation. We conclude with some general observations about computer-aided formal reasoning." For the entire collection see MR 94b:68025.

Barbanera, F. and Berardi, S. (1993). Extracting constructive content from classical logic via control-like reductions. In *Typed lambda calculi and applications (Utrecht, 1993)*, pages 45–59. Springer, Berlin.

[MR94c:03019] Summary: "Recently there has been much interest in the problem of finding the computational content of classical reasoning. One of the most appealing directions for the computer scientist to tackle such a problem is the relation which has been established between classical logic and lambda calculi with control operators, such as M. Felleisen's control operator $\mathcal{C}$. In this paper we introduce a typed lambda calculus with the $\mathcal{C}$ operator corresponding to Peano arithmetic, and a set of reduction rules related to the ones of the usual control calculi with $\mathcal{C}$. We show how these rules, which are proved to be strongly normalizing, can be used to extract witnesses from proofs of Σ^0_1 sentences in Peano arithmetic." For the entire collection see MR 94b:68025.

Cook, S. and Urquhart, A. (1993). Functional interpretations of feasibly constructive arithmetic. *Ann. Pure Appl. Logic*, 63(2):103–200.

[MR94g:03107] The authors present many new results on weak fragments of arithmetic related to the system PV originally introduced by Cook in 1975 to formalize "polynomially verifiable number theory". They also offer a clear and well-motivated introduction to this research field. A notion of feasible function of finite type based on the typed lambda-calculus is introduced which generalizes the PTIME functions. Such feasible functions of higher type are defined by adjoining to the simply-typed lambda-calculus a set of PTIME initial functions and a "bounded recursor on binary notation". The motivation goes back to a theorem of A. Cobham which characterizes the PTIME functions as those functions which can be obtained from a small set of

initial functions by composition and bounded recursion on notation. Two theories are introduced to reason about such functions: an equational one PV^ω, and an intuitionistic one IPV^ω. It is shown that IPV^ω is a conservative extension of both PV and IPV (= PV plus intuitionistic logic), which are in turn conservative extensions of PV. Moreover, IPV is a conservative extension of Buss's system IS^1_2. Interpretations for IPV^ω are developed in the style of both Kreisel's modified realizability and Gödel's Dialectica interpretation. As an application, alternative proofs of Buss's results on IS^1_2 are given and some of his conjectures are settled. It is also proved that IPV^ω, and hence IS^1_2, cannot prove that extended Frege systems are not polynomially bounded. Reviewed by Alessandro Berarducci.

Corella, F. (1993). What holds in a context? *J. Automat. Reason.*, 10(1):79–93.

[MR94e:03008] The "substitution of equals for equals" at a certain place in a formula is an often-used aid in the derivation of a proof. Given $\Gamma \vdash A = B$, one desires to derive $\Gamma \vdash \mathcal{C}[\mathcal{A}] = \mathcal{C}[\mathcal{B}]$, with Γ a set of assumptions and $\mathcal{C}$ a so-called context. In practical situations, e.g. in an interactive theorem prover, one sometimes appeals to information in the context which does not follow from the assumptions. The author gives as an example that one writes $\int_0^1 |x|\,dx = \int_0^1 x\,dx$ because the equality $|x| = x$ holds in the context $\int_0^1 \cdots dx$ and therefore may be added to the assumptions (in this context).

The paper solves the problem of finding the set $\Gamma^{\mathcal{C}}$ of sentences which can be admitted as assumptions in the proof of $A = B$, given a set of assumptions Γ and a context $\mathcal{C}$. This set can be described in a closed form when the derivations are given in the simple theory of types as defined by Alonzo Church. An example shows that there can be correct substitutions other than the ones in $\Gamma^{\mathcal{C}}$, which is due to the incompleteness of the method of proving $\Gamma \vdash \mathcal{C}[\mathcal{A}] = \mathcal{C}[\mathcal{B}]$ by trying to justify $A = B$.

The paper also shows that the situation in a first-order language is not so pleasant as in a higher-order language like Church's system. Reviewed by R. P. Nederpelt.

de Groote, P. (1993). The conservation theorem revisited. In *Typed lambda calculi and applications (Utrecht, 1993)*, pages 163–178. Springer, Berlin.

[MR94c:03072] Summary: "This paper describes a method of proving strong normalization based on an extension of the conservation theorem. We introduce a structural notion of reduction that we call β_s, and we prove that any λ-term that has a $\beta_1\beta_s$-normal form is strongly β-normalizable. We show how to use this result to prove the strong normalization of some typed λ-calculi." For the entire collection see MR 94b:68025.

Gallier, J. (1993). Constructive logics. I. A tutorial on proof systems and typed λ-calculi. *Theoret. Comput. Sci.*, 110(2):249–339.

[MR94c:03076] The paper under review is Part I of a tutorial on proof systems consisting of two separate parts. In Part I the author covers some basic material on natural deduction, sequent calculus and typed lambda-calculus. The latter is introduced by the so-called "Curry-Howard isomorphism", which shows how terms in lambda-calculus constitute a convenient system of notation for natural deduction derivations. The careful treatment of the structural rules is aimed at introducing Girard's linear logic, which is dealt with in Part II (not under review). The author presents a large selection of proof systems both for classical and for intuitionistic logic, including some contraction-free sequent calculi (due to Kleene, Vorobev, Lincoln, and Dickhoff). The paper does not deal with Hilbert-style systems and the corresponding systems of typed combinatory logic. The purpose of the paper is to give a general overview, and the emphasis is placed more on comparing and motivating the different systems than on offering an in-depth treatment of any particular one. However, a full proof of the cut elimination theorem for Gentzen's sequent calculus is given. For the proofs of other classical results, such as normalization theorems in natural deduction and typed lambda-calculi, precise references to the literature are given. Various translations between different systems are given, such as the transformation of a sequent calculus derivation into a natural deduction derivation and vice versa. Four different translations of classical logic into intuitionistic logic are presented, namely the ones of Gentzen, Kolmogorov and Gödel, and a more recent one due to Girard. The plain style of presentation and the number of well-chosen examples make the paper suitable for use in a classroom or for self-study. Reviewed by Alessandro Berarducci.

Hyland, J. M. E. and Ong, C.-H. L. (1993). Modified realizability toposes and strong normalization proofs (extended abstract). In *Typed lambda calculi and applications (Utrecht, 1993)*, pages 179–194. Springer, Berlin.

[**MR1233148**] This item will not be reviewed individually. For details of the collection in which this item appears see 94b:68025.

Murthy, C. R. (1993). Finding the answers in classical proofs: a unifying framework. In *Logical environments (Edinburgh, 1991)*, pages 247–272. Cambridge Univ. Press, Cambridge.

[**MR1255118**] This item will not be reviewed individually. For details of the collection in which this item appears see 94f:03004.

Parigot, M. (1993). Strong normalization for second order classical natural deduction. In *Eighth Annual IEEE Symposium on Logic in Computer Science (Montreal, PQ, 1993)*, pages 39–46. IEEE Comput. Soc. Press, Los Alamitos, CA.

[**MR94h:03112**] Summary: "We prove the strong normalization theorem for second-order classical natural deduction. The method used is an adaptation of the method of reducibility candidates introduced by J.-Y. Girard ["Interpretation fonctionnelle et elimination des coupures de l'arithmetique d'ordre superieure", These d'Etat, Univ. Paris VII, Paris 1972] for second-order intuitionistic natural deduction. The extension to the classical case requires in particular a simplification of the notion of reducibility candidate." For the entire collection see MR 94d:68005.

Solitro, U. and Valentini, S. (1993). Local computation in linear logic. *Math.*

Logic Quart., 39(2):201–212.
[MR95h:03030] Summary: "This work deals with the exponential fragment of Girard's linear logic [J.-Y. Girard, Theoret. Comput. Sci. 50 (1987), no. 1, 101 pp.; MR 89m:03057] without the contraction rule, a logical system which has a natural relation with direct logic [H. Ono and Y. Komori, J. Symbolic Logic 50 (1985), no. 1, 169–201; MR 87a:03053; J. Ketonen and R. W. Weyhrauch, Theoret. Comput. Sci. 32 (1984), no. 3, 297–307; MR 86g:03018]. A new sequent calculus for this logic is presented in order to remove the weakening rule and recover its behavior via a special treatment of the propositional constants, so that the process of cut elimination can be performed using only 'local' reductions. Hence a typed calculus, which admits only local rewriting rules, can be introduced in a natural manner. Its main properties—normalizability and confluence—are investigated; moreover, this calculus is proved to satisfy a Curry-Howard isomorphism [W. A. Howard, in To H. B. Curry: essays on combinatory logic, lambda calculus and formalism, 480–490, Academic Press, London, 1980; MR 82g:03094] with respect to the logical system in question."

Wadler, P. (1993). A taste of linear logic. In *Mathematical foundations of computer science 1993 (Gdańsk, 1993)*, pages 185–210. Springer, Berlin.
[MR95d:03104] This expository paper starts with intuitionistic logic and outlines the procedure for reducing proofs to normal form and the Curry-Howard isomorphism. Linear logic is presented essentially as a fragment of Girard's unitary logic LU. Hence, there are "linear assumptions" written as $\langle A \rangle$, to be used exactly once, along with "intuitionistic assumptions" written as $[A]$, on which contraction and weakening are allowed. Although $[A]$ is equivalent to $\langle !A \rangle$ as an assumption, the distinction between them is required for a sound reduction of linear proofs. Also, a "linear lambda calculus" and the embedding of intuitionistic logic into linear logic are outlined. For the entire collection see MR 94i:68008. Reviewed by Aldo Ursini.

Duquesne, E. and Van de Wiele, J. (1994). Modèle cohérent des réseaux de preuve. *Arch. Math. Logic*, 33(2):131–158.
[MR95d:03098] This is a rather technical paper devoted to the introduction and study of a coherent space model of the set PN of so-called pure proof nets, syntactical objects that were introduced by V. Danos and L. Regnier in order to provide an interpretation of the untyped (or pure) λ-calculus, similar to that of the typed λ-calculus in Girard's proof nets for linear logic.

The model presented consists in a coherent space $\mathcal{C}$ satisfying $\mathcal{C} \supseteq (\mathcal{C} \implies \mathcal{C})$ (hence $\mathcal{C}$ is in fact a model of the untyped λ-calculus). For their interpretation in $\mathcal{C}$ of pure proof nets the authors prove results analogous to those of Girard for proof nets: compatibility theorems and preservation of the interpretation under cut elimination. They then go on to show that the interpretation of a λ-term in $\mathcal{C}$ is equal to the interpretation of its pure proof net translation.

The remainder of the paper introduces and studies an alternative syntax for PN that is suggested by the semantical interpretation. Reviewed by Harold Schellinx.

Mascari, G. F. and Pedicini, M. (1994). Head linear reduction and pure proof net extraction. *Theoret. Comput. Sci.*, 135(1):111–137. Selected papers of the Meeting on the Mathematical Foundations of Programming Semantics (MFPS '92), Part I (Oxford, 1992).

[MR96h:68117] Summary: "Proof net calculus, introduced by J.-Y. Girard [Theoret. Comput. Sci. 50 (1987), no. 1, 101 pp.; MR 89m:03057], has been extended in the paradigm of Scott's domains equation $D = D \to D$ which generates a logical point of view for pure λ-calculus by V. Danos ["Dynamic graphs, an alternative way to compute λ terms (summarized version)", in Third Italian Conference on Theoretical Computer Science (Mantova, 1990), World Sci. Publishing, Singapore, 1990; see MR 91m:68005]. Methodologically speaking, in this paper the proof-theoretic counterpart of C. Böhm's theorem ["Some properties of the $\beta\eta$ normal forms of the λK calculus" (Italian), No. 696, Ist. Naz. Appl. Calcolo, Rome, 1968] for pure λ-calculus is proposed as an extension of the Curry-Howard paradigm. Technically speaking, as the extraction of a subterm using the β-reduction is possible, so can subnet extraction be internalized by cut-elimination; using proof nets realizes managing and better understanding the procedure of extraction." For the entire collection see MR 95f:68007.

Altenkirch, T., Hofmann, M., and Streicher, T. (1995). Categorical reconstruction of a reduction free normalization proof. In *Category theory and computer science (Cambridge, 1995)*, pages 182–199. Springer, Berlin.

[MR1463721] This item will not be reviewed individually. For details of the collection in which this item appears see 98c:68008.

Barbanera, F. and Berardi, S. (1995). A strong normalization result for classical logic. *Ann. Pure Appl. Logic*, 76(2):99–116.

[MR97f:03073]

The authors prove strong normalization for a version of classical logic by using the formulas-as-types notion to prove strong normalization for a system of typed λ-calculus. The latter system, which the authors call $\lambda_{\mathcal{C}_\tau -}$, is not completely new, but is a version of the typed system with a control operator $\lambda_{\mathcal{C}}$ of Felleisen [M. Felleisen and D. P. Friedman, in Formal description of programming concepts, III (Ebberup, 1986), 193–219, North-Holland, Amsterdam, 1987; M. Felleisen et al., in Proceedings of the Symposium on Logic in Computer Science (Cambridge, MA, 1986), 131–141, IEEE, Washington D.C., 1986]. The reduction rules used by the authors are modifications of those that have been used in the literature: some are new and others are restrictions.

Previously, only weak normalization has been proved for such systems. In order to obtain the strong normalization theorem, the authors have to restrict the formulas (types) allowed in the system: they forbid proper subtypes of the form A and forbid occurrences of $\perp$ on the left-hand side of $\to$. Thus, the formulas corresponding to the rules "ex falso quodlibet" and "double negation elimination", namely $\perp \to A$ and $A \to A$, are not formulas of the system. The authors do not feel that this really limits the system since, as they put it, "in meaningful classical reasoning only the corresponding rules are used".

{Reviewer's remark: This paper, like the works it cites, treats of a system in which only $\to$ and $\perp$ are primitive. However, the same basic kinds of reduction rules will work when the other standard connectives (and first-order quantifiers) are taken as primitive. See a paper by the reviewer [Studia Logica 48 (1989), no. 2, 193–217;

MR 91g:03018], in which, however, only weak normalization was proved and the formulas-as-types notion was not used explicitly.} Reviewed by J. P. Seldin.

Borisavljević, M. (1995). Linear multicategories. In *Proceedings of the Mathematical Conference in Priština 1994*, pages 7–15. Univ. Priština, Priština.

[MR98h:03074] Linear multicategories $\mathcal{M}$ correspond to multiplicative fragments of linear intuitionistic logic. They are shown to satisfy a restricted form of the functional completeness theorem. For each $\mathcal{M}$, the internal language $\mathcal{L}(\mathcal{M})$ is a typed lambda calculus. Analogous to a paper by J. Lambek [in Categories in computer science and logic (Boulder, CO, 1987), 217–239, Contemp. Math., 92, Amer. Math. Soc., Providence, RI, 1989; MR 90e:18010], a linear multicategory is completely characterized by its internal language. For the entire collection see MR 98b:00030. Reviewed by Wim Ruitenburg.

Braüner, T. (1995). The Girard translation extended with recursion. In *Computer science logic (Kazimierz, 1994)*, pages 31–45. Springer, Berlin.

[MR98e:03016] Summary: "This paper extends Curry-Howard interpretations of intuitionistic logic and intuitionistic linear logic rules for recursion. The resulting term languages, the λ^{rec}-calculus and the linear λ^{rec}-calculus, respectively, are given sound categorical interpretations. The embedding of proofs in intuitionistic logic into proofs in intuitionistic linear logic given by the Girard translation is extended by rules for recursion such that an embedding of terms of the λ^{rec}-calculus into terms of the linear λ^{rec}-calculus is induced via the extended Curry-Howard isomorphisms. This embedding is shown to be sound with respect to the categorical interpretations." For the entire collection see MR 98d:68013.

Cantini, A. (1995). Levels of truth. *Notre Dame J. Formal Logic*, 36(2):185–213.

[MR96i:03055] This paper defines a formal theory of truth, TLR (= truth with levels and reflection), which extends the classical first-order theory of pure combinators with a primitive truth predicate T and a family of truth predicates T_i, indexed by a directed partial ordering. The motivation for the form of the theory can be summarized: (1) If i is a lower level than j, then the truth predicate T_i is related to T_j as follows: (a) whatever is declared true by T_i is declared true by T_j; (b) T_i is decidable with respect to T_j, i.e., T_jT_iA or T_jT_iA, for appropriate A. (2) Each local truth predicate T_i satisfies the closure principles of the general theory of partial truth of S. Kripke [J. Philos. 72 (1975), no. 19, 690–716]. (3) There is a level-free truth predicate T which is the "limit" of the local truth predicates; T itself has the self-referential abilities of the T_i. The author suggests that TLR might be considered as a sort of axiomatization for a version of a proposal by T. Burge [J. Philos. 76 (1979), 169–198; per bibl.]. Two of the applications of TLR described are a theory of partial classifications and type-free abstraction in the sense of S. Feferman [J. Symbolic Logic 49 (1984), no. 1, 75–111; MR 85i:03068], and a study of classical subsystems of second order arithmetic, in particular ATR_0 [H. Friedman, K. McAloon and S. G. Simpson, in Patras Logic Symposion (Patras, 1980), 197–230, North-Holland, Amsterdam, 1982; MR 84k:03138]. Reviewed by P. C. Gilmore.

Danos, V. and Regnier, L. (1995). Proof-nets and the Hilbert space. In *Advances in linear logic (Ithaca, NY, 1993)*, pages 307–328. Cambridge Univ. Press, Cambridge.
[MR97a:03063] Girard's geometry of interaction (GOI) program is based on analyzing the dynamics of cut-elimination by operators on certain C^* algebras. Roughly, Girard associates to proofs of sequents (or lambda terms or proof nets in linear logic) pairs of matrices (π, σ) whose coefficients are in algebras of operators on l^2, with σ a partial symmetry (roughly expressing a "feedback" interpretation of the cut rule). Girard's execution formula $\mathrm{EX}(\pi, \sigma) = (1-\sigma^2)\pi(1-\sigma\pi)^{-1}(1-\sigma^2)$ gives the interpretation of cut-elimination, where proofs are represented by π's. Termination of the process is interpreted as nilpotency of $\sigma\pi$.

The authors, in a series of papers, have analyzed the execution formula as a computation of various notions of "paths" in the graph representation of lambda terms (via proof nets). Indeed, in work with Asperti and Laneve, they showed the equivalence of several notions of "reduction" paths in lambda calculus (e.g. Levy's labelled reductions and Lamping's graph-reduction algorithm) with a particular notion arising from GOI: that of "regular" path. A regular path is one that has nonzero weight in Girard's dynamic algebra Λ^*. In this technically interesting paper, the authors prove the equivalence of regular paths with yet another notion, "persistent" paths. Roughly, a path is persistent if its residuals through any reductions are still connected. The equivalence involves interpreting regular paths via the space $B(H)$ of bounded linear operators on a Hilbert space H with countable basis. For the entire collection see MR 96d:03004. Reviewed by Philip J. Scott

de Groote, P. (1995). *The Curry-Howard isomorphism.* Academia-Erasme, Louvain.
[MR96h:03041] From the preface: "This eighth volume of the Cahiers du Centre de Logique is a collection of papers connected with the Curry-Howard isomorphism. This isomorphism, which is also known as the formulae-as-types principle, originates in the analogy between typed λ-calculi and natural deduction systems: types may be interpreted as propositions, λ-terms as proofs, and the notions of reduction between λ-terms as Prawitz-style operations that eliminate redundant inferences in proofs. The first two papers of the volume are historical. They reproduce the original contributions of Curry and Howard setting out the isomorphism."

Contents: H. B. Curry and R. Feys, The basic theory of functionality. Analogies with propositional algebra (9–13); W. A. Howard, The formulae-as-types notion of construction (15–26); N. G. de Bruijn, On the roles of types in mathematics (27–54); J. Gallier, On the correspondence between proofs and λ-terms (55–138); H. Geuvers, The calculus of constructions and higher order logic (139–191); J.-Y. Girard, Linear logic: a survey (193–255); J. Lipton, Realizability, set theory and term extraction (257–364).

The first paper is taken from Chapter 9 of Combinatory logic. Vol. I, by Curry and Feys [North-Holland, Amsterdam, 1958; MR 20 #817; second printing, 1968; MR 39 #5368]. The second paper has been reviewed [in To H. B. Curry: essays on combinatory logic, lambda calculus and formalism, 480–490, Academic Press, London-New York, 1980; MR 82g:03094]. The remaining papers are being reviewed individually.

Duquesne, E. and Van de Wiele, J. (1995). Proof nets of PN as graphs. *Arch. Math. Logic*, 34(1):1–20.
[MR96e:03071] From the text: "The notion of proof net in the linear logic of J.-Y. Girard [Theoret. Comput. Sci. 50 (1987), no. 1, 101 pp.; MR 89m:03057] is equivalent to the notion of natural deduction (NJ) in the intuitionistic logic of D. Prawitz [in Proceedings of the Second Scandinavian Logic Symposium (Oslo, 1970), 235–307, North-Holland, Amsterdam, 1971; MR 52 #7871] plus the subformula property. However, the former notion has nice properties only for the multiplicative fragment (PN_0 is the set of multiplicative proof nets). Indeed, in this fragment, one of the properties is the definition of proof net as a graph, so that we get a local definition of proof. In other parts of linear logic, we do not have such a property. We obtain analogous properties for a (isomorphic) subset of a quotient of the set PN of untyped proof nets of V. Danos and L. Regnier which is a translation of the lambda-calculus Λ [see V. Danos, "La logique lineaire appliquee a l'etude de divers processus de normalisation (principalement du λ-calcul)", These de Doctorat, Univ. Paris VII, Paris, 1990; per bibl.]. The subset is the set of untyped proof nets without any exponential-cut.

"The aims of the paper are to give the definition of the subset, which is also named PN, and to give its properties. The first property is that PN is closed under cut elimination (Part 1.B). The second one, which is discussed in Part 2.B, establishes an isomorphism between the subset of PN of axiom-cut free untyped proof nets (which is isomorphic to a quotient set of PN) and the lambda-calculus."

Gallier, J. (1995). On the correspondence between proofs and λ-terms. In *The Curry-Howard isomorphism*, pages 55–138. Academia, Louvain.
[MR97h:03017] Summary: "The correspondence between natural deduction proofs and λ-terms is presented and discussed. A variant of the reducibility method is presented, and a general theorem for establishing properties of typed (first-order) λ-terms is proved. As a corollary, we obtain a simple proof of the Church-Rosser property, and of the strong normalization property, for the typed λ-calculus associated with the system of (intuitionistic) first-order natural deduction, including all the connectors $\rightarrow$, $\times, +, \forall, \exists$, and $\perp$ (falsity) (with or without η-like rules)." For the entire collection see MR 96h:03041.

Girard, J.-Y. (1995). Light linear logic. In *Logic and computational complexity (Indianapolis, IN, 1994)*, pages 145–176. Springer, Berlin.
[MR1449660] This item will not be reviewed individually. For details of the collection in which this item appears see 97m:03004.

Pym, D. J. (1995). A note on the proof theory of the $\lambda\pi$-calculus. *Studia Logica*, 54(2):199–230.
[MR96c:03033] Summary: "The $\lambda\Pi$-calculus, a theory of first-order dependent function types in Curry-Howard-de Bruijn correspondence with a fragment of minimal first-order logic, is defined as a system of (linearized) natural deduction. In this paper, we present a Gentzen-style sequent calculus for the $\lambda\Pi$-calculus and prove the cut-elimination theorem.

"The cut-elimination result builds upon the existence of normal forms for the natural deduction system and can be considered to be analogous to a proof provided by Prawitz for first-order logic. The type-theoretic setting considered here elegantly illustrates the distinction between the processes of normalization in a natural deduction system and cut-elimination in a Gentzen-style sequent calculus.

"We consider an application of the cut-free calculus, via the subformula property, to proof-search in the $\lambda\Pi$-calculus. For this application, the normalization result for the natural deduction calculus alone is inadequate, a (cut-free) calculus with the subformula property being required."

Barbanera, F. and Berardi, S. (1996). Proof-irrelevance out of excluded-middle and choice in the calculus of constructions. *J. Funct. Programming*, 6(3):519–525.

[**MR97j:03061**] The authors show that if we add the principle of excluded middle and the axiom of choice to the Coquand-Huet calculus of constructions, then we have proof-irrelevance; namely Prop is necessarily interpreted as a two-element set and each type is either empty or has only one element. As the authors point out, proofs of similar results were previously obtained by Coquand and Pottinger through an interpretation of Girard's inconsistent system $\mathcal{U}$. The authors' approach gives a short and direct syntactic proof obtained by formalizing a version of Russell's paradox. This yields the equality of true and false with respect to Leibniz equality, a fact which implies proof-irrelevance. Reviewed by Alessandro Berarducci.

Hirokawa, S., Komori, Y., and Takeuti, I. (1996). A reduction rule for Peirce formula. *Studia Logica*, 56(3):419–426.

[**MR98b:03021**] The authors investigate the Curry-Howard isomorphism, typed λ-calculus (with a new constant p, where $p\colon ((\alpha \to \beta) \to \alpha) \to \alpha$ is the Peirce formula) and implicational classical logic. The paper has three sections: (1) A combinatory reduction rule for the Peirce formula; (2) Collapse of λp-terms of the same type; (3) Normalization of λp-terms.

It is shown that all terms with the same type are equivalent with respect to β-reduction augmented by the p-reduction rule; as a result, all the proofs of the same implicational formula are equivalent; strong normalization fails for βp-reduction; the weak normalization theorem holds for the simplified λp-terms with respect to βp-reduction. See also the preceding review [MR 98b:03020]. Reviewed by A. S. Kuzichev.

Andou, Y. (1997). A system of $\lambda\mu$-calculus proper to the implicational fragment of classical natural deduction with one conclusion. *Sūrikaisekikenkyūsho Kōkyūroku*, (976):70–76. Ordinal numbers in proof theory (Japanese) (Kyoto, 1996).

[**MR1488484**] This item will not be reviewed individually. For details of the collection in which this item appears see 98f:03001 .

Fujita, K.-e. (1997). Calculus of classical proofs. I. In *Advances in computing science—ASIAN'97 (Kathmandu, 1997)*, pages 321–335. Springer, Berlin.

[**MR99e:03037**] Summary: "We introduce a simple natural deduction system of classical propositional logic called λ^v_{exc}, and prove the computational properties of the system based on a call-by-value strategy. We show (1) a strict fragment of λ^v_{exc} that is complete with respect to classical provability, and the computational meaning of the existence of such a fragment; (2) a simple exit mechanism by the use of a proof of Peirce's law, and some examples using classical proofs as programs; (3) the Church-Rosser property; (4) the CPS-translation from λ^v_{exc} to $\lambda^{\rightarrow}$, and its correctness with respect to conversions; (5) a computational use of the logical inconsistency in λ^v_{exc}, extended with a certain signature." For the entire collection see MR 99b:68001.

Gilmore, P. C. (1997). NaDSyL and some applications. In *Computational logic and proof theory (Vienna, 1997)*, pages 153–166. Springer, Berlin.
[**MR98i:03074**] Summary: "NaDSyL, a natural deduction-based symbolic logic, and some of its applications are briefly described. The semantics of NaDSyL is based on the term models of the lambda calculus and is motivated by the belief that a confusion of use and mention is the source of the paradoxes. Proofs of soundness, completeness and the eliminability of cuts are sketched along with three applications: the foundations for recursive definitions of well-founded and non-well-founded predicates, classical and intuitionistic second-order arithmetic, and a study of Cantor's diagonal argument and paradox." Cantor's paradox is that if V is the universe of sets then $|V| < |\mathcal{P}(V)| \leq |V|$. For the entire collection see MR 98h:03004.

Hofmann, M. (1997). An application of category-theoretic semantics to the characterisation of complexity classes using higher-order function algebras. *Bull. Symbolic Logic*, 3(4):469–486.
[**MR99f:03050**] Summary: "We use the category of presheaves over PTIME-functions in order to show that Cook and Urquhart's higher-order function algebra PV^ω defines exactly the PTIME-functions. As a by-product we obtain a syntax-free generalisation of PTIME-computability to higher types.

"By restricting to sheaves for a suitable topology we obtain a model for intuitionistic predicate logic with Σ^b_1-induction over PV^ω and use this to re-establish that the provably total functions in this system are polynomial time computable. Finally, we apply the category-theoretic approach to a new higher-order extension of Bellantoni-Cook's system BC of safe recursion."

Joly, T. (1997). Un plongement de la logique classique du 2nd ordre dans FA_2. *C. R. Acad. Sci. Paris Sér. I Math.*, 325(1):1–4.
[**MR98g:03130**] Summary: "We embed second-order classical logic, expressed as a natural deduction system with the rule of reductio ad absurdum (RAA), into the same calculus without RAA, in other words into second-order functional arithmetic FA_2 [see J.-L. Krivine, Lambda-calculus, types and models, Translated from the 1990 French original by Rene Cori, Horwood, New York, 1993; MR 94h:03034 (Chapter 9)]. Using the fact that reductio ad absurdum, $A \vdash A$, can be intuitionistically proved from its restriction to the case in which A is atomic [see D. Prawitz, Natural deduction. A proof-theoretical study, Almqvist & Wiksell, Stockholm, 1965; MR 33 #1227 (Chapter 3, 1)], we merely relativize the formulae to the 'classical' predicates, i.e. to the predicates P satisfying $P \to P$. Like the $_0$-translation of Krivine [op. cit.,

Chapter 10], this embedding allows translation of a classical proof of function totality into an intuitionistic one.

"The normalization of the proof translations proves to be an analysis of the λ-calculus reduction [see M. Parigot, in Logic programming and automated reasoning (St. Petersburg, 1992), 190–201, Lecture Notes in Comput. Sci., 624, Springer, Berlin, 1992; MR 94g:03029; in Eighth Annual IEEE Symposium on Logic in Computer Science (Montreal, PQ, 1993), 39–46, IEEE Comput. Soc. Press, Los Alamitos, CA, 1993; MR 94h:03112; in Computational logic and proof theory (Brno, 1993), 263–276, Lecture Notes in Comput. Sci., 713, Springer, Berlin, 1993; MR 94m:68024], so that the strong normalization property of the second-order λ-calculus follows at once from that of system F. Moreover, in the λ-calculus with η-expansion, the translation preserves normal forms of proofs and even sets up a one-to-one correspondence between the closed normal λ-terms of type A and the normal (intuitionistic) proofs of the relativization A^* of A for sufficiently simple types A, in particular for the integral type N. Therefore, the normal proofs of N^* in FA_2 are exactly the classical integer representations of the λ-calculus."

Mints, G. (1997a). Three faces of natural deduction. In *Automated reasoning with analytic tableaux and related methods (Pont-à-Mousson, 1997)*, pages 16–30. Springer, Berlin.

[**MR99k:03011**] This paper is a survey of several results connecting natural deduction systems with methods of extracting programs from proofs. First, the author compares two methods of extracting programs from proofs in a fragment of intuitionistic sequent propositional logic, one based on the sequent system LJ and the other on the natural deduction system NJ. The principal result is that derivations in either system can readily be transformed into derivations in the other. Next, the author considers a "manageable" formulation of intuitionistic predicate logic with existential instantiation called NJi. It is shown how deductions in this system can be translated into NJ and thus that the system is sound, complete, and strongly normalizable. Next, the author describes the assignment of λ-terms to the !-free fragment of intuitionistic propositional logic, an extension of the Curry-Howard isomorphism. A unique normal form for natural deduction in this logic is obtained, demonstrating the possibility of treating linear logic in the framework of standard λ-calculus without adding new combinators. The author concludes with a method of transforming Gentzen-type rules of natural deduction systems for intuitionistic logic into a resolution calculus. For the entire collection see MR 98k:03004. Reviewed by Henry Africk.

Mints, G. (1997b). Three faces of natural deduction. In *Automated reasoning with analytic tableaux and related methods (Pont-à-Mousson, 1997)*, pages 16–30. Springer, Berlin.

[**MR1610818**] A review for this item is in process.

Ronchi Della Rocca, S. and Roversi, L. (1997). Lambda calculus and intuitionistic linear logic. *Studia Logica*, 59(3):417–448.

[**MR99a:03023**] Summary: "The introduction of linear logic extends the Curry-Howard isomorphism to intensional aspects of typed functional programming. In particular, every formula of linear logic tells whether the term it is a type for can be erased/duplicated or not, during a computation. So linear logic can be seen as a model of a computational environment with explicit control of the management of resources.

"This paper introduces a typed functional language $\Lambda_!$ and a categorical model for it. The terms of $\Lambda_!$ encode a version of natural deduction for intuitionistic linear logic such that linear and nonlinear assumptions are managed multiplicatively and additively, respectively. Correspondingly, the terms of $\Lambda_!$ are built out of two disjoint sets of variables. Moreover, the λ-abstractions of $\Lambda_!$ bind variables and patterns. The use of two different kinds of variables and the patterns allow a very compact definition of the one-step operational semantics of $\Lambda_!$, unlike all other extensions of the Curry-Howard isomorphism to intuitionistic linear logic. The language $\Lambda_!$ is Church-Rosser and enjoys both strong normalizability and subject reduction. The categorical model induces operational equivalences such as, for example, a set of extensional equivalences. The paper also presents an untyped version of $\Lambda_!$ and a type assignment for it, using formulas of linear logic as types. The type assignment inherits from $\Lambda_!$ all the good computational properties and also enjoys the principal-type property."

Seldin, J. P. (1997). On the proof theory of Coquand's calculus of constructions. *Ann. Pure Appl. Logic*, 83(1):23–101.

[**MR98e:03019**] Coquand's "calculus of constructions" is a system of typed λ-calculus in which the second-order polymorphic typed λ-calculus can be interpreted. It forms the core of several interesting "working" implementations of proof systems for mechanizing mathematics. In this paper the author performs an in-depth study of (the usual) metamathematical properties of some (re)formulations of Coquand's system. The arguments are presented in great detail, and all results are given with complete proofs. The paper can be recommended as material for thinking, reading and training to all who would like help in acquiring the technical skills that are likely to come in handy for doing successful research in the area. Reviewed by Harold Schellinx.

Weiermann, A. (1997). A proof of strongly uniform termination for Gödel's T by methods from local predicativity. *Arch. Math. Logic*, 36(6):445–460.

[**MR99c:03080**] Summary: "We estimate the derivation lengths of functionals in Gödel's system T of primitive recursive functionals of finite type by a purely recursion-theoretic analysis of Schutte's 1977 exposition of Howard's weak normalization proof for T. By using collapsing techniques from Pohlers' local predicativity approach to proof theory and the Buchholz-Cichon and Weiermann 1994 approach to subrecursive hierarchies, we define a collapsing function $\mathcal{D}\colon \mathrm{T} \to \omega$ such that for (closed) terms a, b of Gödel's T we have: If a reduces to b then $\omega > \mathcal{D}(a) > \mathcal{D}(b)$.

"By one uniform proof we obtain as corollaries: (a) A derivation lengths classification for functionals in T, hence a new proof of strongly uniform termination of T. (b) A new proof of Kreisel's classification of the number-theoretic functions which can be

defined in T, hence a classification of the provably total functions of Peano arithmetic. (c) A new proof of Tait's results on weak normalization for T. (d) A new proof of Troelstra's result on strong normalization for T.

"Additionally, a slow-growing analysis of Gödel's T is obtained via Girard's hierarchy comparison theorem. This analysis yields a contribution to two open problems posed by Girard in Part II of his book on proof theory. For the sake of completeness we also mention the Howard-Schutte bound on derivation lengths for the simple typed λ-calculus." Reviewed by John P. Helm.

Ishtiaq, S. S. and Pym, D. J. (1998). A relevant analysis of natural deduction. *J. Logic Comput.*, 8(6):809–838.
[MR1664165] A review for this item is in process.

Mints, G. (1998). Linear lambda-terms and natural deduction. *Studia Logica*, 60(1):209–231. Natural deduction.
[MR1620063] A review for this item is in process.

Shirahata, M. (1998). The strong normalization of intuitionistic set theory without equality. *Hiyoshi Rev. Natur. Sci.*, (24):1–13.
[MR1685281] A review for this item is in process.

Danner, N. (1999). Ordinals and ordinal functions representable in the simply typed lambda calculus. *Ann. Pure Appl. Logic*, 97(1-3):179–201.
[MR1682071] A review for this item is in process.

Schwichtenberg, H. (1999). Termination of permutative conversions in intuitionistic Gentzen calculi. *Theoret. Comput. Sci.*, 212(1-2):247–260. Gentzen (Rome, 1996).
[MR1672092] A review for this item is in process.

Stone, M. (1999). Representing scope in intuitionistic deductions. *Theoret. Comput. Sci.*, 211(1-2):129–188.
[MR1663881] A review for this item is in process.

1.12 Recursion theory and decidability

Ferrari, P. L. and Longo, G. (1978). Axiomatic theory of enumeration: a note on the axiom of extensionality. *Studia Logica*, 37(3):261–268.
[MR80k:03049] The authors study recursive function theory using the set-theoretic formalism of Grzegorczyk. They start with a "minimal" theory and show that, contrary to initial hopes, most reasonable extensions of this minimal theory contradict the axiom of extensionality. This seems to destroy all hope for finding useful, extensional models of substantial portions of recursion theory, and in particular, of the lambda-calculus and its variants. (Reviewer's remarks: It now seems quite clear that category theory, and in particular the theory of Cartesian closed categories,

furnishes the most natural, useful models of recursion theory and of the lambda-calculus. Compare with, in particular, the forthcoming article by J. Lambek (To H. B. Curry; Essays on combinatory logic, lambda calculus, and formalism, pp. 375 - 402, Academic Press, New York, 1980) as well as the article by D. Scott (ibid. pp. 403 - 450).) Reviewed by W. S. Hatcher.

Sanchis, L. E. (1979). Reducibilities in two models for combinatory logic. *J. Symbolic Logic*, 44(2):221–234.

[MR80k:03044] The author examines interesting generalizations of several reducibilities from recursion theory in the setting of models for combinatory logic. The two models each have domain $D = \mathcal{P}(N)$, the power set of the natural numbers. In the graph model (due originally to D. Scott) the application operation is $ZY = \{x \mid \exists y \ (\ \langle y, x\rangle \in Z \ \wedge \ D_y \subseteq Y \ )\}$ (where $\langle a, b\rangle$ encodes the ordered pair (a, b) and D_y is the finite set with canonical index y), while in the hypergraph model $ZY = \{x \mid \forall f \exists v \exists y \ (\ \langle f(v), y, x\rangle \in Z \ \wedge \ D_y \subseteq Y \ )\}$, where f is a course-of-values function for f. Thus the generalization of the graph model to the hypergraph model parallels the introduction of hyperenumeration reducibility in the author's earlier paper (Notre Dame J. Formal Logic 19 (1978), no. 3, 405 - 415; MR 58 # 10374). Reducibilities are generalized as follows. A countable subset $\alpha \subseteq D$ is called a reduction basis in the appropriate model if α contains all r.e. sets and is closed under the application operation of the model. With α fixed, define X reducible to Y to mean that $ZY = X$ for some $Z \in \alpha$. Of course, this generates a whole family of reducibilities dependent on the choice of α and on whether ZY is the application operation in the graph model or the hypergraph model. For the simplest case in the graph model, take α to be the class of all r.e. sets. The relation generated is simply enumeration reducibility. By considering $ZY_d = X_d$, where $X_d = X \times \{0\} \cup X' \times \{1\}$, Turing reducibility can be similarly generalized. Various standard theorems (for example, the existence of quasiminimal sets and of incomparable sets) are proved in this more general setting. In the graph model the proofs hinge on continuity arguments, since the operators representable in this model are precisely the continuous ones. A Thomason-style forcing argument is used to prove that for a fixed reduction basis α and $X \in D$, the set $\{Y \mid X \leq Y$ in the hypergraph model $\Rightarrow X \leq Y$ in the graph model$\}$ is comeager (in the Cantor set topology on $\mathcal{P}(N)$). Results of this type are used to lift the theorems proved for the graph model to the setting of the hypergraph model. Corrections: p. 227, l.12: "$x \in X \ \wedge \ y = 0. \ \wedge \ x \notin X \ \wedge \ y = 1$" should read "$x \in X \wedge y = 0. \vee x \notin X \wedge y = 1$"; p. 230, l.4: "Theorem 9" should be "Theorem 10"; p. 233, l.34: "Theorems 10 and 12" should be "Theorems 11 and 13". Reviewed by John W. Berry.

Noah, A. (1980). Predicate-functors and the limits of decidability in logic. *Notre Dame J. Formal Logic*, 21(4):701–707.

[MR82b:03042] The author discusses the decidability of Quine's restricted system of predicate-functor logic (RPF) (Proceedings of the Second Scandinavian Logic Symposium (Oslo, 1970), pp. 309 - 315, North-Holland, Amsterdam, 1971; MR 49

2370). The decidability of the extended system obtained from RPF by allowing nonhomogeneous conjunctions is still open. Reviewed by S. Comer.

Visser, A. (1980). Numerations, λ-calculus & arithmetic. In *To H. B. Curry: essays on combinatory logic, lambda calculus and formalism*, pages 259–284. Academic Press, London.

[MR83c:03041] From the introduction: "In this paper we use some of Ershov's concepts to prove results about structures such as term models of λ-calculus. These results and their proofs are 'coordinate free', i.e. they use (nearly) no specific properties of e.g. λ-calculus. As a consequence, theorems from arithmetic carry over to λ-calculus and vice versa. For example, there is a version of the Gödel-Rosser-Mostowski-Myhill-Kripke theorem for RE λ-theories." The concepts of Ershov used by the author are from his theory of numerations. A numeration γ is a pair (ν, S), where ν is a surjective function from N to S. If $\gamma_1 = (\nu_1, S_1)$ and $\gamma_2 = (\nu_2, S_2)$ are numerations, then μ is a morphism from γ_1 to γ_2 if μ is a function from S_1 to S_2 and there is a recursive function ψ such that $\nu_2 \circ \psi = \mu \circ \nu_1$. Numerations with their morphisms form the category of numerations. If $\gamma = (\nu, S)$ is a numeration, then $m \sim_\gamma n$ means $\nu(m) = \nu(n)$. A numeration γ is positive if $\sim_\gamma$ is an RE relation, precomplete if for every partial recursive function ϕ there is a total recursive function ψ such that for every n in the domain of ϕ, $\phi(n) \sim_\gamma \psi(n)$, and complete if there is an element a of S such that for every partial recursive function ϕ there is a total recursive function ψ such that for every n in the domain of ϕ, $\phi(n) \sim_\gamma \psi(n)$ and for every n not in the domain of ϕ, $\nu(\psi(n)) = a$. As an example of a numeration, take a consistent set T of identities between closed λ-terms which is closed under the rules of the $\lambda\beta$-calculus, and consider a primitive recursive Gödel numbering of the terms which is bijective. Let λ_T be the function which assigns to the Gödel number of a term the set of all terms equal to it under T, and let M_T be the set of all equivalence classes of terms determined by T. Then (λ_T, M_T) is a precomplete numeration. If we take for T the theory obtained from the $\lambda\beta$-calculus by identifying all unsolvable terms, we get a complete numeration. Other examples include the recursive natural numbers (0-place partial recursive functions), for which there is a complete numeration; the partial recursive functions, which have a complete numeration; the RE sets, which have a complete numeration; and the "intensional' partial recursive functions, which are based on extensions of PA and which have a precomplete numeration. A topology can be defined for a numeration by taking as a basis the set of all subsets S_0 of S for which $\nu^{-1}(S_0)$ is Π^0_1. (This differs from a topology introduced by Ershov based on Σ^0_1 sets and which does not have all the properties of this one.) Under this topology, morphisms are continuous and, for a precomplete numeration, any two nonempty open sets intersect. By using some simple topological facts, this last result, and some simple results for the precomplete numeration of the $\lambda\beta$-calculus, it is possible to prove the range theorem of the λ-calculus (if there are closed lambda-terms $N_1, \ldots, N_k$ such that for any closed term P, $MP =_{\lambda\beta} N_i$ for some i between 1 and k, where M is a given closed term, then there is a closed term N such that for all closed P, $MP =_{\lambda\beta} N$ and also that if, for any closed P, MP has a normal form (where M is closed) then there is a closed normal form N such that, for all closed P, $MP =_{\lambda\beta} N$. Ershov proved the following fixed point theorem: If γ is precomplete, then for each partial recursive ϕ there is an n (which can be determined effectively from an index of ϕ) such that if $\phi(n)$ has a value, then $\phi(n) \sim_\gamma n$. The author uses this to prove the Gödel-Rosser-Mostowski-Myhill-Kripke theorem, of which a corollary is the following: If T is a consistent RE extension of PA, then there is an infinite recursive sequence of Σ_1^0-sentences $B_0, B_1, \dots$, such that for any sequence $C_0, C_1, \dots$ of Σ_1^0-sentences, $T + (C_0 \leftrightarrow B_0) + (C_1 \leftrightarrow B_1) + \dots$ is consistent. A similar result can be proved for λ-theories: If T is an RE λ-theory which is consistent, then there are (closed) terms $\Omega_0, \Omega_1, \dots$, such that for every sequence $M_0, M_1, \dots$ of closed terms, $T + (\Omega_0 = M_0) + (\Omega_1 = M_1) + \cdots$ is consistent. It also follows that any countable partial order can be embedded in the partial order of RE λ-theories. The author concludes by trying to generalize to numerations in general the passage from the precomplete numeration of an arbitrary λ-theory to the complete numeration of the λ-theory obtained from the original one by identifying certain terms. He does not get a complete generalization, but he does obtain some partial results in that direction. As a corollary to the results he does have, he shows that the partial recursive functions can be represented in any RE λ-theory. (For the entire collection see MR 81j:03005.) Reviewed by J. P. Seldin.

Byerly, R. E. (1982). Recursion theory and the lambda-calculus. *J. Symbolic Logic*, 47(1):67–83.

[MR83i:03066] Two enumerations $\langle \psi_e : e \in \omega \rangle$ of the partial recursive functions determine (partial) structures $(\omega, \cdot)$ and $(\omega, *)$, where $e \cdot m \simeq \phi_{e(m)}$ and $e * m \simeq \psi_{e(m)}$. It is known that for acceptable enumerations $(\omega, \cdot)$ and $(\omega, *)$ are recursively isomorphic. The author describes a class of predicates P with the property that for any $a, b \in \omega$ such that $P(a)$ and $P(b)$, there is an isomorphism which maps a to b. Previously known examples of this were the predicates $P_{f(a)}$ (a is an index for the partial recursive function f) for certain f such as the empty and identity functions. The class of predicates is characterized by means of a semantics for the λ-calculus. An extension of the class is shown not to consist entirely of such predicates. Reviewed by Peter G. Hinman.

Barendregt, H. and Longo, G. (1983). Recursion theoretic operators and morphisms on numbered sets. *Fund. Math.*, 119(1):49–62.

[MR86b:03051] From the introduction: "There is a natural correspondence between classes of recursion-theoretic operators (enumeration operators, Turing operators, etc.) and morphisms on an appropriate numbered set (in the sense of Ershov), via the constructive part of the λ-calculus models $\mathcal{P}\omega^2$ and T^ω. The different classes of operators on $\mathcal{P}\omega$ are effective continuous maps obtained by embedding $\mathcal{P}\omega$ into $\mathcal{P}\omega^2$ or T^ω in two natural ways, giving $\mathcal{P}\omega$ either the Cantor or the Scott topology. In particular, Turing operators work on $\mathcal{P}\omega$ with the Cantor topology. This is implicit in A. Nerode's theorem relating tt-reducibility to total Turing operators [cf. H. Rogers, Jr., Theory of recursive functions and effective computability, see p. 154,

McGraw-Hill, New York, 1967; MR 37 # 61]. Also we give a different proof of a theorem [op. cit.; see p. 151] relating enumeration reducibility and Turing reducibility. Finally we prove an interpolation result, in the sense of algebra, for total Turing operators."

Engeler, E. (1984). Equations in combinatory algebras. In *Logics of programs (Pittsburgh, Pa., 1983)*, pages 193–205. Springer, Berlin.

[**MR778940**] This item will not be reviewed individually. For details of the collection in which this item appears see 86a:68004.

Longo, G. (1984). Limits, higher type computability and type-free languages. In *Mathematical foundations of computer science, 1984 (Prague, 1984)*, pages 96–114. Springer, Berlin.

[**MR87c:03094**] This paper is a nice, captivating survey of various abstract theories for programming languages and computability issues related to these, aimed at giving a unifying approach to partial and total functionals. In a very essential and intuitive way, the author discusses some of the work on limit spaces and filter spaces of J. M. E. Hyland [Ann. Math. Logic 16 (1979), no. 2, 101–143; MR 81e:03047] and then introduces domains as defined by D. S. Scott [Ordered sets (Banff, Alta., 1981), 677–718, Reidel, Dordrecht, 1982; MR 83j:06007]; he also gives one of the first accounts of the category-theoretic study of partial functions [see also A. Heller, Proceedings of the conference on mathematical logic, Vol. 2 (Italian) (Siena, 1983/1984), 339–344, Univ. Siena, Siena, 1985; see MR 87a:03008; G. Rosolini, "Continuity and effectiveness in topoi", Ph.D. Thesis, Univ. Oxford, Oxford, 1986; H.-J. Hoehnke, Universal algebra (Esztergom, 1977), 373–412, North-Holland, Amsterdam, 1982; MR 83j:18010]. The intuitive character is always well expressed; the reader should be aware, though, that some of the categorical notions appear in the literature under different names.

The discussion moves on to partial computable operators defined on subspaces of $P\omega$ as restrictions of effective operators. Finally, some constructions of models of type-free λ-calculus are sketched. See also the following review. For the entire collection see MR 86b:68004. Reviewed by Ana Pasztor.

Zashev, J. A. (1984a). Basic recursion theory in partially ordered models of some fragments of the combinatory logic. *C. R. Acad. Bulgare Sci.*, 37(5):561–564.

[**MR85j:03077**] The author describes some partially ordered models of subsets of the combinators and among other things claims to be able to prove versions of the recursion theorems for these. The reviewer was unable to ascertain the truth or the interest of these claims. Reviewed by Henk Barendregt.

Zashev, J. A. (1984b). ∇-combinatory algebras and abstract Plotkin's models. *C. R. Acad. Bulgare Sci.*, 37(6):711–714.

[**MR765739**] There will be no review of this item.

Longo, G. and Martini, S. (1985). Computability in higher types and the universal domain $P\omega$. In *Proceedings of the conference on mathematical logic, Vol. 2 (Siena, 1983/1984)*, pages 501–506, Siena. Univ. Siena.

[**MR825041**] This item will not be reviewed individually. For details of the collection in which this item appears see 87a:03008.

Soskov, I. (1985). The connection of prime computability with recursiveness in functional combinatory spaces. In *The mathematical theory of programming (Russian)*, pages 4–11, 169. Akad. Nauk SSSR Sibirsk. Otdel. Vychisl. Tsentr, Novosibirsk.

[**MR88f:03040**] From the introduction (translated from the Russian): "We investigate the connection between prime computability as defined by Y. N. Moschovakis [Trans. Amer. Math. Soc. 138 (1969), 427–464; MR 39 # 5362] and recursiveness in functional combinatory spaces, introduced by D. G. Skordev [Bull. Acad. Polon. Sci. Ser. Sci. Math. Astronom. Phys. 29 (1976), no. 1, 23–31; MR 54 # 4950; Combinatory spaces and recursiveness in them (Russian), Bulgar. Acad. Sci., Sofia, 1980; RZhMat 1981:12 A22 K; Mathematical theory and practice of software systems (Russian) (Sofia, 1980), 7–16, Novosibirsk, 1980; RZhMat 1982:10 V880]." For the entire collection see MR 88b:68002.

Zashev, Ĭ. (1985). Abstract Plotkin models. In *The mathematical theory of programming (Russian)*, pages 12–33, 169. Akad. Nauk SSSR Sibirsk. Otdel. Vychisl. Tsentr, Novosibirsk.

[**MR88h:03066**] Summary (translated from the Russian): "The Plotkin model $P\omega$ of combinatory logic was used by D. S. Scott [in Proceedings of the third Scandanavian logic symposium (Uppsala, 1973), 154–193, North-Holland, Amsterdam, 1975; MR 52 # 5372] to obtain an elegant algebraic treatment of the elementary theory of recursively enumerable sets. In this paper we consider an abstract treatment of this type along the lines of Skordev's theory of combinatory spaces. It is an exposition, though in slightly different terms and notation, of a part of the results briefly formulated in our earlier papers [C. R. Acad. Bulgare Sci. 37 (1984), no. 5, 561–564; MR 85j:03077; ibid. 37 (1984), no. 6, 711–714]. Here we use a method (specifically, the notion of a generating operator) similar to the method of L. L. Ivanov [Algebraic recursion theory, Horwood, Chichester, 1986; MR 88e:03068] in the theory of operator spaces." For the entire collection see MR 88b:68002.

Stanevichene, L. I. (1987). On the problem of the intersection emptiness of deterministic languages. *Kibernetika (Kiev)*, 1987(6):101, 135.

[**MR89c:68063**] Summary: "An algorithmic problem is formulated concerning deterministic and unambiguous languages which are approximations of programming languages. Unsolvability of the problem is proved by relating it to the Post correspondence problem."

Longo, G. (1988a). From numbered sets to type theories. *Rend. Sem. Mat. Univ. Politec. Torino*, (Special Issue):41–73. Conference on Logic and Computer Science: New Trends and Applications (Turin, 1986).

[**MR90f:03030**] From the introduction: "This introductory lecture stresses a recursion-theoretic and constructive perspective (in the sense of intuitionistic logic) as structures invented for the semantics of typed and type-free λ-calculus are looked

at within a constructive frame. In particular, we view Scott's constructive domains as countable (and numbered) sets, in the sense of Maltsev (1); then 'subcountable' and still effective sets are investigated (2). The formalization of these structures (3) leads us to higher-order type theories (4) and finally to their (constructive) models (5). Several issues sketched here will be revisited and expanded in another paper by the author (1987)."

The revisitation referred to above actually appeared somewhat later [Ann. Pure Appl. Logic 40 (1988), no. 2, 93–133]. For the entire collection see MR 89i:00024. Reviewed by Robert M. Baer.

Longo, G. (1988b). On Church's formal theory of functions and functionals. The λ-calculus: connections to higher type recursion theory, proof theory, category theory. *Ann. Pure Appl. Logic*, 40(2):93–133.

[MR91h:03022] This paper is a general survey of the relationships among λ-calculus, recursion theory, and category-theoretic models. The author begins with some very simple notions: numbers, pairings, and Gödel numberings. In terms of these, in order to obtain a notion of computable functional of higher type, he defines the hereditary partial effective functionals; at each type this definition requires a suitable coding of pairs. It is also desirable to generalize the s-m-n theorem to higher types. Maltsev's category EN of numbered sets (enumerations) is presented as the simplest generalization of recursion theory to an abstract setting. Its objects are pairs $A = (A, e_A)$, where A is a set and $e_A : \omega \to A$ is a surjective map, and f is a morphism from A to B if there is a recursive f' such that $f \circ e_A = e_B \circ f'$. Unfortunately, EN lacks some desirable closure properties; for example, exponentiations are desirable, but EN is not Cartesian closed. Two approaches to obtaining these closure properties are considered, one inside EN and the other outside.

Inside EN, it is possible to use topological ideas (as D. Scott did [in 4th Annual Princeton Conference on Information Systems Science (1970), 169–176; per revr.; SIAM J. Comput. 5 (1976), no. 3, 522–587; MR 55 #10262] and as Yu. L. Ershov did [in Logic Colloquium '76 (Oxford, 1976), 455–467, North-Holland, Amsterdam, 1977; MR 58 #21541]) to define the category CD of constructive domains. Its objects are constructive domains considered as numbered sets, and its morphisms are continuous and computable maps, partially ordered pointwise. It is Cartesian closed and is a full subcategory of EN.

Outside EN, the author defines the category M of modest sets, whose objects are the subsets of ω modulo an equivalence relation, and if $A = (A, \pi_A)$ and $B = (B, \pi_B)$, where π_A and π_B are numberings, then f is a morphism from A to B if and only if there is a primitive recursive f' such that $f \circ \pi_A = \pi_B \circ f'$. This category M is a CCC and includes EN as a full sub-Cartesian category.

The author then proceeds to the consideration of a formalized theory of functions based on the notions with which he began. The result is the typed $\lambda\beta\eta$-calculus with surjective pairing, whose models are exactly the (concrete) CCCs. By erasing types and dropping the surjective property of the pairing operator, he obtains the type-free $\lambda\beta\eta$-calculus. This is compared with combinatory logic, and models of both λ-calculus and combinatory logic are considered, especially in relation to M.

The author then extends this discussion to higher types by introducing λ_2, an impredicative variant of the system F of J.-Y. Girard ["Interpretation fonctionelle elimination des coupures dans l'arithmetique d'ordre superieur", These d'Etat, Univ. Paris VII, Paris, 1972] and, at the same time, an extension of the second-order polymorphic typed λ-calculus. He mentions Girard's results that strong normalization holds for λ_2, that the functions definable in λ_2 are the recursive functions provably total in second-order arithmetic, and that the normalization theorem for λ_2 cannot be proved in second-order arithmetic.

The author concludes by showing how CD and M can be extended to models of λ_2. He begins with CD, and from it constructs the domain CFP of constructive finitary projections. Beginning with an object U of CD such that $U^U < U$ (the existence of which can be guaranteed by a constructive version of Scott's D_∞ construction), define a projection to be essentially a retraction less than or equal to the identity. A projection φ in CD is said to be finitary if the range of φ is a domain. Then CFP_U is the set of morphisms of CD which are finitary projections. If U is clear from the context, one writes CFP. It is a full subCCC of CD which models λ_2.

In the case of M, the author sketches a method for turning HEO^2 into a model of λ_2. The model uses Eff, the effective topos of J. M. E. Hyland [in The L. E. J. Brouwer Centenary Symposium (Noordwijkerhout, 1981), 165–216, North-Holland, Amsterdam, 1982; MR 84m:03101]. The objects of Eff are pairs $(A, =_A)$, where $=_A : A \times A \to P\omega$ is partial; the set of morphisms of Eff from A_1 to A_2 is the set of "total functions" (total, single-valued, strict and substitutive relations) with respect to $=_1$ and $=_2$. Eff is a topos which models λ_2 and has M as a full subCCC and an internal category.

Reviewer's remarks: The paper is, on the whole, clearly written. It also has a lot of references. Unfortunately, the reference list was not proofread as carefully as it might have been; there are several places where different works by an author are cited with the same year, and some citations in the text do not match the list at the end. Reviewed by J. P. Seldin.

Saks, M. and Statman, R. (1988). An intersection problem for finite automata. *Discrete Appl. Math.*, 21(3):245–255.

[MR90f:68047] Summary: "A finite automaton with state space S and alphabet A can be thought of as a directed graph with vertex set S such that for each vertex $t \in S$ the edges leaving t are in one-to-one correspondence with A. Motivated by a problem in logic, we propose a problem about intersections of languages accepted by finite automata and give some partial results."

Trakhtenbrot, B. A. (1988). Comparing the Church and Turing approaches: two prophetical messages. In *The universal Turing machine: a half-century survey*, pages 603–630. Oxford Univ. Press, New York.

[MR1011492] This item will not be reviewed individually. For details of the collection in which this item appears see 91b:03004.

Abrusci, V. M. and Mascari, G. (1989). A logic of recursion. *Inform. and Comput.*, 81(2):168–226.

[MR90j:68022] Since the concept of recursion is basic in many structures and languages of computing science, a logic of recursion is of high value. The paper introduces so-called recursion structures based on term languages including typed λ-calculus and some operators related to the fixpoint operator. Then a logical framework is defined consisting of formulas constructed from the terms of those term languages. Finally, a logical calculus is given as a logic of recursion. The paper is extremely technical and very hard to read. Many definitions are given and most of the formulas and concepts are hardly motivated or explained. Also, the result is rather theoretical. The logic of recursion is based on quantifiers over ordinals used for iterators. Therefore the practical applicability of this logic in proofs about programs is not obvious. A more tractable logical calculus based on the ideas presented would be of high interest. Reviewed by Manfred Broy.

Goerdt, A. (1989). Characterizing complexity classes by general recursive definitions in higher types. In *CSL '88 (Duisburg, 1988)*, pages 99–117. Springer, Berlin.
[MR91d:68032] Summary: "General recursive definitions in higher (finite) types, an alternative notation for finitely typed λ-terms with if-then-else and fixpoints, can be classified into an infinite syntactic hierarchy: a definition is in the nth level of this hierarchy, and is called a rank-n-definition, if n is an upper bound on the levels of the types occurring in it.

"We restrict attention to definitions defining first-order functions, i.e., functions of type ind $x \cdots x$ind $\to$ ind, with ind for individuals; higher types occur only as detours in between." For the entire collection see MR 90i:68005.

Pino Perez, R. (1989). Decidability of the restriction equational theory in the partial lambda calculus. *Theoret. Comput. Sci.*, 67(1):129–139.
[MR90j:03028] The starting point of the partial λ-calculus is a theorem of Plotkin which links the notion of definability in a semantics of partial functions to the notion of termination in a calculus in which the normal forms are the values (i.e. constants, variables or terms starting with 'λ'). The partial λ-calculus has been extensively studied by E. Moggi ["The partial lambda-calculus", Ph.D. Thesis, Univ. Edinburgh, 1988; per bibl.]. The terms of the partial λ-calculus are defined as the terms of the (type-free) λ-calculus plus an operator $\lceil$ of "restriction". In particular, if M is a term and D is a set of terms, $M\lceil D$ is a term. Intuitively the meaning of $M\lceil D$ is M if all the terms in D are defined, undefined otherwise. Moggi introduced a set of rules to handle equivalence of terms with respect to restrictions (without considering β or η conversion) which define an equational theory $T_\lceil$ and he conjectured that $T_\lceil$ is decidable. In this paper the author proves Moggi's conjecture. Since Moggi has proved that the partial λ-calculus is confluent modulo $T_\lceil$, from the decidability of $T_\lceil$ it follows that equality in the partial λ-calculus is decidable for all strongly normalizable terms.

It does not seem possible to prove the decidability of $T_\lceil$ by finding a canonical form for a term of the partial λ-calculus by orienting the restriction equations (making them into a rewriting system) in such a way that the number of steps in the computation of the normal form is finite. This difficulty is overcome by defining a sort of parallel (binary) reduction which depends on the maximum of the sizes of M and N, where M and N are the terms to be compared. Reviewed by M. Coppo.

Statman, R. (1989). The word problem for Smullyan's lark combinator is decidable. *J. Symbolic Comput.*, 7(2):103–112.
[MR90e:03017] R. Smullyan's lark combinator L is characterized by the equation $Lxy = x(yy)$ [Smullyan, To mock a mockingbird, Knopf, New York, 1985]. The author presents a thorough analysis and a decision procedure for the theory of equality defined by this equation (i.e., the theory whose axiom-schemes are $Lxy = x(yy)$ and $x = x$, and whose rules are transitivity, replacement and symmetry). This theory is

shown to be limited but by no means trivial, and the author's work is a significant addition to the very small stock of known decision procedures for type-free systems in combinatory logic. Reviewed by R. Hindley.

Cook, S. A. and Kapron, B. M. (1990). Characterizations of the basic feasible functionals of finite type. In *Feasible mathematics (Ithaca, NY, 1989)*, pages 71–96. Birkhäuser Boston, Boston, MA.

[MR94g:03076] This paper is an effort to extend the notion of polynomial time computability to functionals. Type 0 functionals are natural numbers. Type 1 functionals are ordinary functions with natural numbers as arguments and which return a natural number. Type $t+1$ functionals are functions whose arguments are functionals of types $j \leq t$ and which return natural numbers.

The authors use an equational definition to define a class of "basic feasible functionals" and justify their name for this class by pointing out that the type 1 basic feasible functionals are exactly the polynomial time computable functions, and by presenting two different programming-style characterizations of the class. One of the characterizations, which develops a general Algol-style language devised for functionals, depends on the equational definition; the other, which is based on a notion of bounded type loop programs, is independent of the definition.

Suggesting a need for further study, the authors state that examples exist of functionals which are intuitively feasible but are not basic feasible. They also note that type 3 basic feasible functionals can be discontinuous and therefore, intuitively, not feasibly computable. For the entire collection see MR 94b:03005. Reviewed by John P. Helm.

Goerdt, A. and Seidl, H. (1990). Characterizing complexity classes by higher type primitive recursive definitions. II. In *Aspects and prospects of theoretical computer science (Smolenice, 1990)*, pages 148–158. Springer, Berlin.

[MR92f:03036] Summary: "Higher type primitive recursive definitions (also known as Gödel's system T) defining first-order functions (i.e., functions of type ind $\rightarrow \cdots \rightarrow$ ind $\rightarrow$ ind, ind for individuals, higher types occur in between) can be classified into an infinite syntactic hierarchy: A definition is in the nth stage of this hierarchy, a so-called rank-n-definition, if and only if n is an upper bound on the levels of the types occurring in it.

"We interpret these definitions over finite structures and show for $n \geq 1$: Rank-$(2n+2)$-definitions characterize (in the sense of Y. Gurevich [in 24th Annual Symposium on Foundations of Computer Science (Tucson, AZ, 1983), 210–214, IEEE, Silver Spring, MD, 1983; CCA 1984:21281]) the complexity class DTIME$(\exp_n(\text{poly}))$ while rank-$(2n+3)$-definitions characterize DSPACE$(\exp_n(\text{poly}))$ (here $\exp_0(x) = x$, $\exp_{n+1}(x) = 2^{\exp_n(x)}$). This extends the results that rank-1-definitions characterize LOGSPACE [op. cit.], rank-2-definitions characterize PTIME, rank-3-definitions characterize PSPACE, and rank-4-definitions characterize EXPTIME [Part I, Goerdt, in CSL '88 (Duisberg, 1988), 99–117, Lecture Notes in Comput. Sci., 385, Springer, Berlin, 1989; MR 91d:68032]." For the entire collection see MR 91h:68006.

Leivant, D. (1990). Subrecursion and lambda representation over free algebras (preliminary summary). In *Feasible mathematics (Ithaca, NY, 1989)*, pages 281–291. Birkhäuser Boston, Boston, MA.

[**MR1232934**] This item will not be reviewed individually. For details of the collection in which this item appears see 94b:03005.

Amadio, R. M. (1991). Recursion over realizability structures. *Inform. and Comput.*, 91(1):55–85.

[**MR92e:03066**] A realizability model for a language including J.-Y. Girard's system F [in Proceedings of the Second Scandinavian Logic Symposium (Oslo, 1970), 63–92, North-Holland, Amsterdam, 1971; MR 53 #12895] and an operator of recursion on types is given and some basic local properties are investigated. Reviewed by Cristian Calude.

Dowek, G. (1991). L'indécidabilité du filtrage du troisième ordre dans les calculs avec types dépendants ou constructeurs de types. *C. R. Acad. Sci. Paris Sér. I Math.*, 312(12):951–956.

[**MR92c:03013**] Summary: "We prove the undecidability of the third-order pattern matching problem in typed λ-calculi with dependent types and in those with type constructors, by reducing the second-order unification problem to them."

Luckhardt, H. (1991). Complexity versus the Church-Rosser property and confluence. *Z. Math. Logik Grundlag. Math.*, 37(1):85–92.

[**MR93e:03020**] The author considers two important properties of term rewriting systems, namely, confluence defined as follows: (C) $A > B$ &$A > C$ implies $B > D$ &$C > D$ for some D, and the Church-Rosser property in the following setting: (CR) $A = B$ &$A > A'$ &$B > B'$ implies $A' > C$ & $B' > C$ for some C. Obviously CR implies C, but not conversely. However, this approach is limited in practice by the complexity involved. The author illustrates this idea by applying it to propositional logic and in particular to intuitionistic propositional logic.

The results are as follows: Consider a computational theory T and a complexity class C which contains quadratic complexity and is closed under addition. If provability in T or in a T-fibre F is C-undecidable, then any reduction in T which is locally of complexity C is not C-well-founded (C-terminating) or not Church-Rosser on T (or on F).

The author studies conditions under which this becomes an equivalence and holds for confluence. Several interesting observations and a theorem concerning classes with exponential complexity are proved. Reviewed by P. Stepanek.

Mairson, H. G. (1992). A simple proof of a theorem of Statman. *Theoret. Comput. Sci.*, 103(2):387–394.

[**MR94a:03021**] In the paper under review the author gives a simplified and self-contained proof of a theorem of Statman that deciding the $\beta\eta$-equality of two first order typable λ-terms is not elementary recursive [see also R. Statman, Theoret. Comput. Sci. 9 (1979), no. 1, 73–81; MR 80m:68043; J. Symbolic Logic 47 (1982), no. 1, 17–26; MR 84e:03020]. Reviewed by Alessandro Berarducci.

Tiuryn, J. (1992). Solving equational constraints in polymorphic types. In *Logical foundations of computer science—Tver '92*, pages 480–492. Springer, Berlin.

[**MR94j:03039**] Summary: "We propose an algebra $\mathcal{F}$ as a framework for talking about polymorphic types. This framework may be useful for studying various type reconstruction problems that arise in type disciplines which employ polymorphic types. We prove that the problem of equational satisfiability in $\mathcal{F}$ is NP-complete. As an illustration of how this framework may work we show how to set up equational constraints in $\mathcal{F}$ for the problem of type reconstruction for the system of P. Giannini and S. Ronchi Della Rocca which was introduced in 1991. This gives another, purely algebraic, proof of their decidability result, with a clear upper bound for the complexity of this problem." For the entire collection see MR 94h:68009.

Constable, R. L. and Smith, S. F. (1993). Computational foundations of basic recursive function theory. *Theoret. Comput. Sci.*, 121(1-2):89–112. A collection of contributions in honour of Corrado Böhm on the occasion of his 70th birthday.

[MR95f:03061] Summary: "The theory of computability, or basic recursive function theory as it is often called, is usually motivated and developed using Church's thesis. Here we show that there is an alternative computability theory in which some of the basic results on unsolvability become more absolute, results on completeness become simpler, and many of the central concepts become more abstract. In this approach, computations are viewed as mathematical objects, and theorems in recursion theory may be classified according to which axioms of computation are needed to prove them.

"The theory is about typed functions over the natural numbers, and it includes theorems showing that there are unsolvable problems in this setting, independent of the existence of indexings. The unsolvability results are interpreted to show that the partial function concept, so important in computer science, serves to distinguish between classical and constructive type theories (in a different way than does the decidability concept as expressed in the law of excluded middle). The implications of these ideas for the logical foundations of computer science are discussed, particularly in the context of recent interest in using constructive type theory in programming." For the entire collection see MR 94e:68005.

Dowek, G. (1993). The undecidability of pattern matching in calculi where primitive recursive functions are representable. *Theoret. Comput. Sci.*, 107(2):349–356.

[MR94a:03019] The pattern matching problem asks whether a closed term is a substitution instance of another. The author shows that the answer is negative in Girard's system F and in Gödel's T. (For these respective sytems, refer, for instance, to two other works [J.-Y. Girard, P. Taylor and Y. Lafont, Proofs and types, Cambridge Univ. Press, Cambridge, 1989; MR 90g:03013; K. Gödel, Collected works. Vol. II, Oxford Univ. Press, New York, 1990; MR 91b:01093].) The general scheme of the proof is, as one expects, to represent primitive recursive functions in the system and to reduce the problem in question to that of the existence of a solution to $f(x) = 0$. The author indicates a number of other systems in which the answer is negative, and leaves the question open for weak systems such as the simply typed λ-calculus. Reviewed by M. Yasuhara.

Pfenning, F. (1993). On the undecidability of partial polymorphic type reconstruction. *Fund. Inform.*, 19(1-2):185–199.

[MR95h:03105] The author shows that, given a partially typed term in the pure polymorphic λ-calculus, it is undecidable whether the term can be fully typed. This important result was later strengthened by J. B. Wells [in Proceedings of the Symposium on Logic in Computer Science LICS '94 (Paris, 1994); per revr.], who proved that, given an untyped term, it is undecidable whether it can be typed in the pure polymorphic λ-calculus. The proof contained in the paper under review is obtained through a reduction to a second-order unification problem and extends previous results by H.-J. Boehm and Mitchell. The technique applies also to predicative fragments of the polymorphic λ-calculus, such as λ^{ML}. Therefore these calculi also have an undecidable partial type reconstruction problem. Reviewed by Alessandro Berarducci.

Sazonov, V. and Voronkov, A. (1993). A construction of typed lambda models related to feasible computability. In *Computational logic and proof theory (Brno, 1993)*, pages 301–312. Springer, Berlin.

[MR94k:03058] Various notions of computable functionals of finite type have been considered in the literature. Here the authors propose an abstract approach. They start with an arbitrary class of basic functions on one or more basic domains (the class is only assumed to be closed under explicit definitions). Then they define in a very natural way the corresponding functionals of higher types in such a manner that no new functions on the basic types are added. Such a class of functionals gives an extensional model of the simply typed lambda-calculus. It is observed that if one starts with the basic domain of the natural numbers and the PTIME functions, then the class of functionals thus obtained includes the feasible functionals of S. Cook and A. Urquhart [in Proceedings of the Twenty-first Annual Symposium on Theory of Computing (Seattle, WA, 1989), 107–112, ACM, New York, 1989]. For a related approach see a paper by G. Longo and E. Moggi [J. Symbolic Logic 49 (1984), no. 4, 1319–1332; MR 86h:03082]. For the entire collection see MR 94f:03003. Reviewed by Alessandro Berarducci.

Soloviev, S. V. (1993). Lambda terms with functional symbols and decidability in certain closed categories. In *Substructural logics (Tübingen, 1990)*, pages 309–325. Oxford Univ. Press, New York.

[MR95h:03031] This paper is a contribution to the investigation of the relations among closed categories, typed lambda-calculi and constructive (substructural) logics, to which a considerable amount of research is currently being devoted both in logic and in computer science. In particular, the author investigates the problem of obtaining a decision procedure for the equality of morphisms of the free symmetric monoidal closed category CL(V) over V, given the decidability of the same problem for V, where V is symmetric monoidal but not closed. In order to solve this problem the author devises a system of typed λ-terms which act as notations for the morphisms of CL(V) (including constants for morphisms of V), whose types are

interpreted as objects of CL(V). Equality between these λ-terms turns out to be decidable, by adapting a proof technique previously used in solving related problems by Mints and Babaev, and this entails the decidability of equality of morphisms in CL(V). For particular symmetric monoidal categories, notably a familiar one whose objects are finite ordinals, simplifications of the general construction and of the decision procedure are described. For the entire collection see MR 95a:03003. Reviewed by Felice Cardone.

Sprenger, M. and Wymann-Böni, M. (1993). How to decide the lark. *Theoret. Comput. Sci.*, 110(2):419–432.

[MR94d:03026] It is not hard to show that combinatory equality based on combinators that do not allow duplication (such as any subset of $\{\mathbf{B}, \mathbf{C}, \mathbf{I}, \mathbf{K}\}$) is decidable, while, for example, the system generated by $\{\mathbf{B}, \mathbf{C}, \mathbf{I}, \mathbf{W}\}$ is not. R. Statman has shown [J. Symbolic Comput. 7 (1989), no. 2, 103–112; MR 90e:03017] that combinatory equality based on Smullyan's lark combinator L (which is such that $\mathbf{L}xy \to x(yy)$, or $\mathbf{L} = \mathbf{BWB}$) is decidable. The present authors derive the same result, using a procedure based on a canonical rewriting system that has complexity which is only a polynomial of low degree. An additional advantage is that the method can be applied more generally, for example, to show that the logics based on $\mathbf{W}$ ($\mathbf{W}xy \to xyy$) and on $\mathbf{M}_1$ ($\mathbf{M}_1 xy \to xxy$) are also decidable. Reviewed by Martin W. Bunder.

Urzyczyn, P. (1993). Type reconstruction in F_ω is undecidable. In *Typed lambda calculi and applications (Utrecht, 1993)*, pages 418–432. Springer, Berlin.

[MR94c:03026] Summary: "We investigate Girard's calculus F_ω as a 'Curry style' type assignment system for pure lambda terms. We prove that the type-reconstruction problem for F_ω is undecidable (even with quantification restricted to constructor variables of rank 1). In addition, we show an example of a strongly normalizable pure lambda term that is untypable in F_ω." For the entire collection see MR 94b:68025.

David, R. (1994). The inf function in the system F. *Theoret. Comput. Sci.*, 135(2):423–431.

[MR96b:03021] Summary: "We give a λ term of type Nat, Nat$\to$Nat in the system F that computes the minimum of two Church numerals in time $O(\inf \log(\inf))$. This refutes a conjecture of the 'λ folklore'." Reviewed by Chantal Berline.

Dowek, G. (1994a). Erratum: "The undecidability of third-order pattern matching in calculi with dependent types or type constructors" [C. R. Acad. Sci. Paris Sér. I Math. **312** (1991), no. 12, 951–956; MR 92c:03013]. *C. R. Acad. Sci. Paris Sér. I Math.*, 318(9):873.

[MR95e:03046] The author reports that M. Bezem and J. Springintveld have pointed out an error in the proof of the second theorem (p. 956) and have obtained some related theorems.

Dowek, G. (1994b). Third order matching is decidable. *Ann. Pure Appl. Logic*, 69(2-3):135–155. Invited papers presented at the 1992 IEEE Symposium on Logic in Computer Science (Santa Cruz, CA).

[**MR95i:68074**] Summary: "The higher order matching problem is the problem of determining whether a term is an instance of another in the simply typed λ-calculus, i.e. of solving the equation $a = b$ where a and b are simply typed λ-terms and b is ground. The decidability of this problem is still open. We prove the decidability of the particular case in which the variables occurring in the problem are at most third order." For the entire collection see MR 95e:00034.

Henglein, F. and Mairson, H. G. (1994). The complexity of type inference for higher-order typed lambda calculi. *J. Funct. Programming*, 4(4):435–477.

[**MR95j:03024**] his paper is the first to give interesting lower bounds on the complexity of type inference for the system F_2 and its extensions up to F_ω. However, two researchers have subsequently proved that type inference for the systems F_2 and F_ω is undecidable [see P. Urzyczyn, in Typed lambda calculi and applications (Utrecht, 1993), 418–432, Lecture Notes in Comput. Sci., 664, Springer, Berlin, 1993; MR 94c:03026; J. B. Wells, "Typability and type checking in F_2 are equivalent and undecidable", in Proceedings of Logic in Computer Science Conference, IEEE, Comput. Soc. Press, Long Beach, CA, 1994; per revr.]. Reviewed by Eugenio Moggi.

Kobayashi, S. and Tatsuta, M. (1994). Realizability interpretation of generalized inductive definitions. *Theoret. Comput. Sci.*, 131(1):121–138.

[**MR95h:03109**] Summary: "Generalized inductive definitions give a way of defining a predicate as the least solution P of the equation $P \leftrightarrow A[P]$, where the predicate variable P may occur in the formula $A[P]$ positively. This paper gives a **q**-realizability interpretation of generalized inductive definitions and proves the soundness of the interpretation." Reviewed by Yu. G. Ventsov.

Kozen, D., Palsberg, J., and Schwartzbach, M. I. (1994). Efficient inference of partial types. *J. Comput. System Sci.*, 49(2):306–324.

[**MR95m:03032**] Partial types form a partially ordered set $(T, \leq)$ built up from the constant Ω using $\rightarrow$, where the partial types are ordered by (i) $t \leq \Omega$, for all t, (ii) $s \rightarrow t \leq s' \rightarrow t'$ if and only if $s' \leq s$ and $t \leq t'$. Note that $\leq$ is covariant in the second argument t and contravariant in the first argument s.

If A is a type environment, E a λ-term, and t is a partial type, then $A| \vdash E\colon t$ means that E has the partial type t in the environment A. Thatte gave four proof rules including the subsumption rule: from $A| \vdash E\colon s$, $s \leq t$, infer $A| \vdash E\colon t$. The type inference problem is the question, given a λ-term E, whether E is typable in the partial types. This problem was shown to have an exponential time solution by O'Keefe and Wand. The type constraints problem is the question, given a finite system of inequalities $s \leq t$ over the partial types T, whether the system has a solution in $(T, \leq)$, and if so, to give a solution.

In the paper under review, the authors give an $O(n^3)$ algorithm for the type inference problem by reducing this problem to the problem of solving a finite system of type constraints. This is done by associating partial types with binary trees, constraints

with constraint graphs, and then defining finite-state automata with transitions along the edges of the constraint graphs. By checking whether the finite automaton has a cycle with one non-ϵ transition from the initial state, the type constraints problem (even with recursive types) is shown to be solvable in $O(n^3)$ time. Reviewed by Peter Clote.

Pierce, B. C. (1994). Bounded quantification is undecidable. *Inform. and Comput.*, 112(1):131–165.

[**MR95h:03029**] The combination of polymorphic second-order λ-calculus with subtyping and bounded universal quantification was first introduced by L. Cardelli and P. Wegner [Computing Surveys 17 (1985), no. 4, 471–522]. In their original formulation the subtyping relation can easily be shown to be decidable. P.-L. Curien and G. Ghelli [in Theoretical aspects of computer software (Sendai, 1991), 731–749, Lecture Notes in Comput. Sci., 526, Springer, Berlin, 1991; see MR 92h:68009] proposed a stronger quantifier subtyping rule, in which subtyping of quantified types is contravariant in their bounds and covariant in their bodies. The question of the decidability of this last subtyping relation has been discussed by many researchers. The author gives here a negative answer: the subtyping is undecidable. This result is obtained by reducing the halting problem for two-counter Turing machines to the decision of this subtyping relation. As an intermediate step, rowing machines (consisting of registers that store rows) are introduced. The content of the present paper is indeed outstanding, both for its consequences for language design and for its deep mathematical development. The presentation is polished and makes pleasant reading. Reviewed by Mariangiola Dezani-Ciancaglini.

Smullyan, R. M. (1994). *Diagonalization and self-reference.* The Clarendon Press Oxford University Press, New York. Oxford Science Publications.

[**MR96c:03001**] This book is the third volume by the author [Gödel's incompleteness theorems, Oxford Univ. Press, New York, 1992; MR 93i:03002; Recursion theory for metamathematics, Oxford Univ. Press, New York, 1993; MR 94a:03066] on the relations between recursion theory and incompleteness. It is basically meant to present a unified theory of fixed point theorems for metamathematics.

The book is divided into four parts. The first part contains three introductory chapters devoted to problems on self-reference (and their solutions), and one chapter on classical results due to Gödel, Tarski, Rosser (plus an intriguing theorem of Parikh saying, essentially, that in PA there exist sentences which can be proved to be provable in far fewer steps than they can be proved). Self-reference in arithmetic (Chapter 5) precedes the development of recursion theory for formal systems (Chapters 6–8); the remainder of this part presents an effective version of incompleteness. The theory of fixed points is then presented, in a detailed form, in Parts 3 and 4.

The book is devoted to a rather difficult topic; its style, however, makes most of its parts digestible, sometimes even enjoyable. The major drawback pertains to the selection of the material: with the exception of some results on lambda calculus, the book concentrates on the literature up to the end of the 1960s. No reference is made to new and significant results, in particular to the information-theoretic version of incompletness due to G. J. Chaitin [see Information-theoretic incompleteness, World Sci. Publishing, River Edge, NJ, 1992; MR 94e:68001]. Reviewed by Cristian Calude.

David, R. and Nour, K. (1995). Storage operators and directed lambda-calculus. *J. Symbolic Logic*, 60(4):1054–1086.

[MR96k:03035] Summary: "Storage operators were introduced by J. L. Krivine [RAIRO Inform. Theor. Appl. 25 (1991), no. 1, 67–84; MR 93a:68084]; they are closed λ-terms which, for a data type, allow one to simulate a 'call by value' while using the 'call by name' strategy. In this paper, we introduce the directed λ-calculus and show that it has the usual properties of the ordinary λ-calculus. With this calculus, we get an equivalent—and simple—definition of the storage operators that allows some of their properties to be shown: the stability of the set of storage operators under β-equivalence (Theorem 5.1.1); the undecidability (and semidecidability) of the problem 'is a closed λ-term t a storage operator for a finite set of closed normal λ-terms?' (Theorems 5.2.2 and 5.2.3); the existence of storage operators for every finite set of closed normal λ-terms (Theorem 5.4.3); the computation time of the 'storage operation' (Theorem 5.5.2)."

Joy, M. S. and Rayward-Smith, V. J. (1995). NP-completeness of a combinator optimization problem. *Notre Dame J. Formal Logic*, 36(2):319–335.

[MR96e:03013] Summary: "We consider a deterministic rewrite system for combinatory logic over combinators S, K, I, B, C, S', B', and C'. Terms are represented by graphs so that reduction of a duplicator causes the duplicated expression to be 'shared' rather than copied. To each normalizing term we assign a weighting which is the number of reduction steps necessary to reduce the expression to normal form. A lambda-expression may be represented by several distinct expressions in combinatory logic, and two combinatory logic expressions are considered equivalent if they represent the same lambda-expression (up to β-η-equivalence). The problem of minimizing the number of reduction steps over equivalent combinator expressions (i.e., the problem of finding the 'fastest running' combinator representation for a specific lambda-expression) is proved to be NP-complete by reduction from the 'hitting set' problem."

Kurata, T. and Takahashi, M. (1995). Decidable properties of intersection type systems. In *Typed lambda calculi and applications (Edinburgh, 1995)*, pages 297–311. Springer, Berlin.

[MR1477990] This item will not be reviewed individually. For details of the collection in which this item appears see 98e:03020.

Schlüter, A. (1995). A theory of rules for enumerated classes of functions. *Arch. Math. Logic*, 34(1):47–63.

[MR95m:03033] Summary: "We define an applicative theory CL_2 similar to combinatory logic which can be interpreted in classes of functions possessing an enumerating function. In contrast to the models of classical combinatory logic, it is not necessarily assumed that the enumerating function itself belongs to that function class. Thereby, we get a variety of possible models including, for example, the classes of primitive recursive, recursive, elementary, polynomial-time computable or ξ_0-recursive functions. We show that in CL_2 a major part of the metatheory of enumerated classes of functions can be developed. Namely, a kind of λ-abstraction can be defined and abstract versions of the S^m_n- and (primitive) recursion theorems are proved. Thereby, a closer analysis of the phenomenon of the different recursion theorems is achieved. A theory closely related to CL_2 can be used to replace the applicative part of Feferman's theories for explicit mathematics. So this work can be seen as an answer to Feferman's question to formulate a theory for explicit mathematics in which operations can be interpreted as primitive recursive or even more feasible ones.

"Finally it is shown that the proof-theoretical strength of various theories for explicit mathematics is preserved when replacing the applicative part of the theories by our theory together with an operation for primitive recursion."

Bezem, M. and Springintveld, J. (1996). A simple proof of the undecidability of inhabitation in λP. *J. Funct. Programming*, 6(5):757–761.

[**MR97k:03015**] The theorem mentioned in the title was previously deduced from the essential undecidability of Robinson's arithmetic Q by embedding predicate logic in λP (the simplest pure type system with dependent types). The authors deduce the theorem more directly from the unsolvability of the halting problem for register machines by encoding register machine programs in λP, using only primitive notions of λP.

Hillebrand, G. and Kanellakis, P. (1996). On the expressive power of simply typed and let-polymorphic lambda calculi. In *11th Annual IEEE Symposium on Logic in Computer Science (New Brunswick, NJ, 1996)*, pages 253–263. IEEE Comput. Soc. Press, Los Alamitos, CA.

[**MR98f:68101**] Summary: "We present a functional framework for descriptive computational complexity, in which the regular, first-order, Ptime, Pspace, k-Exptime, k-Expspace ($k \geq 1$), and elementary sets have syntactic characterizations. In this framework, typed lambda terms represent inputs and outputs as well as programs. The lambda calculi describing the above computational complexity classes are simply or let-polymorphically typed with functionalities of fixed order. They consist of order 0 atomic constants, order 1 equality among these constants, variables, application, and abstraction. Increasing functionality order by one for these languages corresponds to increasing the computational complexity by one alternation. This exact correspondence is established using a semantic evaluation of languages for each fixed order, which is the primary technical contribution of this paper." For the entire collection see MR 98a:68006.

Hillebrand, G. G., Kanellakis, P. C., and Mairson, H. G. (1996). Database query languages embedded in the typed lambda calculus. *Inform. and Comput.*, 127(2):117–144. Special issue dedicated to Paris Christos Kanellakis.

[**MR97k:68043**] The authors investigate the expressive power of the simply typed λ-calculus when expressing computations over finite structures, i.e. databases. They prove that it is possible to efficiently embed in the typed λ-calculus the following database query languages: the relational algebra, inflationary Datalog with negation, and the complex object algebra. The embeddings are efficient in the sense that

PTIME-computable queries are expressible by λ-terms that, when applied to an input, reduce to an output in a PTIME-sequence of reduction steps according to a PTIME-reduction strategy (in most cases a call-by-value strategy). These results show that, despite some classical limitative results, the simply typed λ-calculus is a rich framework for expressing computations provided the right setting is chosen. To overcome some of the limitations of the simple types, the types of the input and the output are allowed to differ. As pointed out by the authors, the concern with computational issues is an important part of their results since it is implicit in the literature that, without considering efficient reduction strategies, all elementary functions are expressible. With a careful encoding it is possible to show that all PTIME queries are expressible in the order 5 fragment of the typed λ-calculus (order 4 if we admit an equality predicate as a primitive). For the entire collection see MR 97c:68014. Reviewed by Alessandro Berarducci.

Padovani, V. (1996). Decidability of all minimal models. In *Types for proofs and programs (Torino, 1995)*, pages 201–215. Springer, Berlin.
[MR98e:03018] Summary: "We consider a simply typed λ-calculus with constants of ground types, and assume that for one ground type o, there are finitely many constants of type o. By the minimal model we mean the quotient by observational equivalence of the set of all closed terms whose type has terminal subformula o. We show that this model is decidable: all classes of any given type are recursively representable, and observational equivalence on closed terms is a decidable relation. In particular, this result solves the question raised by R. Statman on the decidability of this model in the case of a unique ground type and two constants." For the entire collection see MR 98d:03006.

Andreev, A. and Soloviev, S. (1997). A deciding algorithm for linear isomorphism of types with complexity $O(n \log^2(n))$. In *Category theory and computer science (Santa Margherita Ligure, 1997)*, pages 197–209. Springer, Berlin.
[MR99e:03014] Summary: "It is well known that ordinary isomorphisms (associativity and commutativity of 'times', isomorphisms for 'times' unit and currying) provide a complete axiomatisation of isomorphism of types in multiplicative linear lambda calculus (isomorphism of objects in a free symmetric monoidal closed category). One of the reasons to consider linear isomorphism of types instead of ordinary isomorphism was that better complexity could be expected. Meanwhile, no upper bounds reasonably close to linear have been obtained. We describe an algorithm deciding whether two types are linearly isomorphic with complexity $O(n \log^2(n))$." For the entire collection see MR 99c:68004.

Małolepszy, J., Moczurad, M., and Zaionc, M. (1997). Schwichtenberg-style lambda definability is undecidable. In *Typed lambda calculi and applications (Nancy, 1997)*, pages 267–283. Springer, Berlin.
[MR98g:03045] Summary: "We consider the lambda definability problem over an arbitrary free algebra. There is natural notion of primitive recursive function in such

algebras and at the same time a natural notion of a lambda definable function. We show that the problem: 'For a given free algebra and a primitive recursive function within this algebra, decide whether this function is lambda definable' is undecidable if the algebra is infinite. The main part of the paper is dedicated to the algebra of numbers, in which lambda definability is described by the Schwichtenberg theorem. The result for an arbitrary infinite free algebra is obtained by a simple interpretation of numerical functions as recursive functions in this algebra. This result is a counterpart of the distinguished result of R. Loader in which lambda terms are evaluated in finite domains for which undecidability of lambda definabilty is proved." For the entire collection see MR 98f:68017.

Hirokawa, S. (1998). Infiniteness of proof(α) is polynomial-space complete. *Theoret. Comput. Sci.*, 206(1-2):331–339.
[MR1641659] A review for this item is in process.

Huet, G. (1998). Regular Böhm trees. *Math. Structures Comput. Sci.*, 8(6):671–680. Logic and models of computation (Marseille, 1996).
[MR1672404] A review for this item is in process.

Loader, R. (1998). Unary PCF is decidable. *Theoret. Comput. Sci.*, 206(1-2):317–329.
[MR1641655] A review for this item is in process.

1.13 Universal algebra and model theory (including categorical versions)

Plotkin, G. (1978). T^ω as a universal domain. *J. Comput. System Sci.*, 17(2):209–236.
[MR80d:68105] The author deals with the choice of the domain of interpretation for the mathematical semantics in the sense of Scott. Instead of a lattice, the author suggests considering the domain T^ω, where ω is a power of the three-element truth-value complete partial order T. The language LAMBDA is studied to define functions over T^ω. It is shown that LAMBDA definability coincides with computability for the functions. The category of domains which are retracts of T^ω is introduced. It is shown how constructions such as sum, product, and exponentiation are effected and how domain equations can be solved using least fixed points. A subcategory of algebraic domains usually used for giving semantics is examined. The topological aspect of T^ω is discussed. Reviewed by V. Yu. Meitus.

Hajnal, A. and Németi, I. (1979). Applications of universal algebra, model theory, and categories in computer science. *CL & CL—Comput. Linguist. Comput. Lang.*, 13:251–282. Survey and bibliography.
[MR81d:68012] From the introduction: "In the last eight years universal algebra and model theory together with their categorical versions have found ever-increasing applications in the field of computer science, more precisely, in the study of semantics

of programming languages and in the methodology of proving properties of programs. Some of the main directions of research are surveyed briefly below." (Over 400 items occur in the bibliography.)

Ricci, G. (1979). P-algebras and combinatory notation. *Riv. Mat. Univ. Parma (4)*, 5:577–589 (1980).

[MR83d:18004] Given a base set A, the author defines "P-algebras" on A to be certain structures made up of functions from $\mathcal{P}(A)$ into A. He defines homomorphisms, direct products, polynomials, varieties, etc. of P-algebras in a fairly obvious way, and proves some basic results. The role of combinators [H. B. Curry and R. Feys, Combinatory logic, Vol. I, North-Holland, Amsterdam, 1958; MR 20 #817] is emphasised. This paper seems to be an independent discovery of ideas very close to the theory of Cartesian closed categories, in which the role of combinators was described in a much deeper paper by J. Lambek [Toposes, algebraic geometry and logic (Halifax, N.S., 1971), pp. 57 - 82, Lecture Notes in Math., 274, Springer, Berlin, 1972; MR 50 #1850]. The present paper's emphasis and treatment are different from standard category theory, however, and if it had appeared some years earlier, it would have been an interesting beginning. Reviewed by R. Hindley.

Bracho, F. (1980). Continuously generated fixed points in $P\omega$. *Fund. Inform. (4)*, 3(4):477–489.

[MR82c:06015] D. S. Scott (SIAM J. Comput. 5 (1976), no. 3, 522 - 587; MR 55 # 10262) has shown that $\mathcal{P}\omega$, the power set of ω, is a domain within which can be modelled the set of all continuous (=directed-join preserving) functions $\mathcal{P}\omega \to \mathcal{P}\omega$, the point being that such functions are already determined by their values on finite subsets of ω. Thus $\mathcal{P}\omega$ has a subspace FUN which is order-isomorphic to the set of continuous functions $\mathcal{P}\omega \to \mathcal{P}\omega$. Further, the "least-fixed-point operator" $FUN \to \mathcal{P}\omega$ which takes each continuous function $f \in FUN$ to its least fixed point $\bigcup f^n(\emptyset)$ $(n \geq 1)$ is again a continuous function. In the present paper, the author considers arbitrary fixed point operators, that is, continuous functions $F : FUN \to \mathcal{P}\omega$ such that $f(F(f)) = F(f)$ for each $f \in FUN$. A subset $x \in \mathcal{P}\omega$ is called a continuously generated fixed point of f if there exists a fixed point operator F with $F(f) = x$. The author introduces a continuous function $Cgn : FUN \times \mathcal{P}\omega \to \mathcal{P}\omega$ which generates all continuously generated fixed points, i.e., for any $f \in FUN$, the continuously generated fixed points of f are precisely the subsets of the form $Cgn(f,x)$ for $x \in \mathcal{P}\omega$. The author also discusses continuously generated retracts and fixed point operators in general. Reviewed by Evelyn Nelson.

Lawson, J. D. (1980). Algebraic conditions leading to continuous lattices. *Proc. Amer. Math. Soc.*, 78(4):477–481.

[MR81g:06002] Let L be a complete lattice. For $x, y \in L$, $x \ll y$ if for all up-directed sets D the relation $y \leq supD$ implies there exists a $d \in D$ such that $x \leq d$. A lattice L is (Scott) continuous if for all $y \in L$, $y = sup\{x : x \ll y\}$. A monograph by G. Gierz, K. H. Hofmann, K. Keimel, the author, M. Mislove and D. Scott (A compendium of continuous lattices, Springer, New York, 1980) surveys the recent activity on continuous lattices and their applications. (For a different use of the term "continuous lattice" see the book by P. Crawley and R. P. Dilworth (The algebraic theory of lattices, Prentice-Hall, Englewood Cliffs, N.J., 1973).) In this paper the author gives several algebraic conditions which force a complete lattice that is a compact topological semilattice with respect to one of its operations to be a continuous lattice with respect to the other operation. For example, if join is the (topologically) continuous operation and each element of L is the meet of the primes (p is prime if $a \wedge b \leq p$ implies $a \leq p$ or $b \leq p$) above it, then L is a (Scott) continuous lattice with respect to meet. A very important result is that a complete lattice of finite breadth is (Scott) continuous if and only if it is meet-continuous. Reviewed by James W. Lea, Jr.

Sanchis, L. E. (1980). Reflexive domains. In *To H. B. Curry: essays on combinatory logic, lambda calculus and formalism*, pages 339–361. Academic Press, London.

[MR83c:03012] From the introduction: "We study here reflexive structures in which the basic domains are complete lattices, and all continuous operations are representable. This works as a model only if the representation itself is continuous. Models of this form were discovered by D. Scott, and following J. E. Stoy we call them reflexive domains. "We are primarily interested in reflexive domains as models of some languages containing lambda abstraction and combinators. But to some extent the study can be pursued quite independently of the particular language. The reason is that the representable operations are independent of the language. "We study a number of language-independent properties of domains. We propose a classification and consider several constructions, in particular the inverse limit model, that we generalize by using representations in place of projections." (For the entire collection see MR 81j:03005.)

Engeler, E. (1981). Algebras and combinators. *Algebra Universalis*, 13(3):389–392.

[MR83f:03012] Given a set A, let $B \supseteq A$ be the least set such that if $\beta \subseteq B$, β finite, $b \in B$, then $(\beta, b) \in B$. Define on the power set $\mathcal{P}B$ the operation $*$: $X * Y = \{b : \exists \beta \subseteq Y \ (\beta, b) \in X\}$. Let $D_A = (\mathcal{P}B, *)$. Theorem: (i) D_A is a combinatory algebra. (ii) D_A is universal in the sense that every applicative structure $(A, \cdot)$ can be isomorphically embedded into D_A. (Reviewer's remark: Although D_A is related to the graph model, recent work of G. Longo shows that it has interesting different properties.) Reviewed by Henk Barendregt.

Obtułowicz, A. and Wiweger, A. (1982). Categorical, functorial and algebraic aspects of the type-free lambda calculus. In *Universal algebra and applications (Warsaw, 1978)*, pages 399–422. PWN, Warsaw.

[MR86f:03026] This is a development of the category-theoretic treatment of the syntax and semantics of λ-calculus begun by Obtulowicz [Fundamentals of computation theory (Poznan-Kornik, 1977), 302–307, Springer, Berlin, 1977; MR 58 #16178]. The authors define further basic λ-calculus concepts in category language, and prove the key lemmas. Interpretations in an arbitrary Cartesian closed category are also defined and discussed, not just the category of sets. This paper is based on work first presented in 1977–1978, and is hence one of the earliest general formulations of λ-calculus semantics. But today, eight years later, it appears rather clumsy. A reader would have to be very well trained in category language before the paper seemed easy to read, and even then, it suffers by comparison with the neat account in C.P.J. Koymans' paper [Inform. and Control 52 (1982), no. 3, 306–332; MR 85b:03020], where just enough category-language was introduced to distinguish between different kinds of models in a very simple way. For the entire collection see MR 85d:00024. Reviewed by R. Hindley

Seely, R. A. G. (1982). Locally Cartesian closed categories and type theory. *C. R. Math. Rep. Acad. Sci. Canada*, 4(5):271–275.

[MR83m:03076] From the summary and text: "A modified version ML of P. Martin - Löf's type theory [Logic Colloquium '73 (Bristol, 1973), pp. 73 - 118, North-Holland, Amsterdam, 1975; MR 52 #7856] is presented which characterises the categorical structure of a locally Cartesian closed category. The details are available from the author as a publication (with the same title as this note) in the McGill lecture notes series."

Baker-Finch, C. A. (1984). Algebraic, operational and denotational semantics of the lambda calculus. *Austral. Comput. J.*, 16(3):96–101.

[MR86h:68104] Author summary: "This paper presents a proof of the congruence of an operational and a denotational semantics of the lambda calculus. We adopt an algebraic notion of congruence and use an algebraic style of semantics to unify the two more traditional definitions."

Barendregt, H. (1984). Lambda calculus and its models. In *Logic colloquium '82 (Florence, 1982)*, pages 209–239. North-Holland, Amsterdam.

[MR86f:03025] This is a thorough revision of the introduction to the concept of model in Chapter 5 of the first edition of the author's book The lambda calculus [North-Holland, Amsterdam, 1981; MR 83b:03016]. The author attributes part of his material to a paper by C. P. J. Koymans [Inform. and Control 52 (1982), no. 3, 306–332; MR 85b:03020], and the review of Koymans' paper serves well as a summary of the present one.

The present paper has now been incorporated, after further changes, in the revised edition of the author's book The lambda calculus [1984; MR 86a:03012] as a new Chapter 5; in fact this is the main change in the revision. The revised treatment brings the exposition in Chapter 5 up to the high standard of the rest of the book, and the category-theoretic description of the different kinds of model is a superb example of category language at its most illuminating.

The definition of "λ-algebra" in the paper is wrong, and should be replaced by the definition in the revised edition of the book; i.e., a λ-algebra is a combinatory algebra that satisfies all equations $X = Y$ such that X_λ is $\lambda\beta$-convertible to Y_λ. For the entire collection see MR 85g:03006. Reviewed by R. Hindley.

Bunder, M. W. (1984). Category theory based on combinatory logic. *Arch. Math. Logik Grundlag.*, 24(1-2):1–16.

[**MR86b:18001**] This paper is concerned with the foundations of category theory. The problems in this area arise from the desire to form categories by definition principles which are excluded from any kind of present set theory (e.g., the formation of classes of classes of classes). The observation that a category is a structure talking about "composition of morphisms" suggests the use of an applicative structure and its relation to combinatory logic in order to find more adequate definition principles. This approach is carried out systematically by the author. Comparisons with other methods (like the one by Herrlich and Strecker which is less general) are given. Reviewed by M. M. Richter.

Wiweger, A. (1984). Pre-adjunctions and lambda-algebraic theories. *Colloq. Math.*, 48(2):153–165.

[**MR86d:18001**] A left pre-adjunction (F_0, φ, ψ) of a functor $G\colon \mathcal{A} \to \mathcal{X}$ consists of functions $F_0\colon \mathrm{Ob}\mathcal{X} \to \mathrm{Ob}\mathcal{A}$ and

$$\mathcal{A}(F_0X, A) \underset{\psi_{X,A}}{\overset{\varphi_{X,A}}{\rightleftarrows}} \mathcal{X}(X, GA)$$

such that, for all X, $\varphi_{X,A}$ is natural in A. In addition the author considers the following conditions: (α_1) for all X, $\psi_{X,A}$ is natural in A; (α_2) $\varphi_{X,A} \circ \psi_{X,A} = \mathrm{id}$ for all X and A; (α_3) $\psi_{X,A} \circ \varphi_{X,A}{=}\mathrm{id}$ for all X and A. $(\alpha_2) \wedge (\alpha_3)$ is equivalent to right adjointness of G and implies (α_1), but no other two conditions imply the third. The author discusses this in terms of examples with particular interest in algebraic theories. Reviewed by Walter Tholen.

Curien, P.-L. (1985a). Categorical combinatory logic. In *Automata, languages and programming (Nafplion, 1985)*, pages 130–139. Springer, Berlin.

[**MR87c:68050**] Summary: "We present the connection between λ-calculus and Cartesian closed categories both in an untyped and in a purely syntactic setting. More specifically, we establish a syntactic equivalence theorem between what we call categorical combinatory logic and λ-calculus with explicit products and projections, with β- and η-rules as well as with surjective pairing. 'Combinatory logic' is of course inspired by Curry's combinatory logic, based on the well-known S, K, I. Compiling λ-calculus happens to be natural and leads to only $n \log n$ code expansion (which is achieved only with infinitely many S, K-like combinators). Moreover, categorical combinatory logic is entirely faithful to β-reduction, whereas combinatory logic needs additional, rather complex and unnatural axioms. This paper is intended as a mathematical foundation for developing implementations of functional programming languages based on a 'categorical abstract machine', which is developed in forthcoming papers." For the entire collection see MR 87a:68002.

Curien, P.-L. (1985b). Typed categorical combinatory logic. In *Mathematical foundations of software development, Vol. 1 (Berlin, 1985)*, pages 157–172. Springer, Berlin.

[**MR87k:03011**] Summary: "The subject of the paper is the connection between the typed λ-calculus and Cartesian closed categories, pointed out by several authors.

Three languages and their theories, defined by equations, are shown to be equivalent: the typed λc-calculus (i.e. the λ-calculus with explicit products and projections) λc_K, the free Cartesian closed category CCC_K, and a third intermediary language, the typed categorical combinatory logic CCL_K, introduced here. In contrast to CCC_K, CCL_K has the same types as λc_K, and roughly the terminal object in CCC_K is replaced by the application and pairing operators in CCL_K. In CCL_K, β-reductions as well as evaluations with respect to environments (the basis of most practical implementations of λ-calculus based languages) may be simulated in the well-known framework of a term rewriting system. Finally the introduction of CCL_K allowed us to understand the untyped underlying calculus, investigated in a companion paper. Another companion paper describes a general setting for equivalences between equational theories and their induced semantic equivalences, of which the equivalence between CCL_K and CCC_K is an instance." For the entire collection see MR 86j:68005.

Hayashi, S. (1985). Adjunction of semifunctors: categorical structures in nonextensional lambda calculus. *Theoret. Comput. Sci.*, 41(1):95–104.
[MR87i:18005] The motivating example for this paper comes from a book by D. S. Scott [To H. B. Curry: essays on combinatory logic, lambda calculus and formalism, 403–450, Academic Press, London, 1980; MR 82d:03025] where it was shown how to use the Karoubi envelope to represent canonically a model of the $\lambda\beta$-calculus as an object U of an appropriate Cartesian closed category equipped with a retraction pair $U^U \xrightarrow{i} U \xrightarrow{j} U^U$, $ji = \mathrm{id}$. The author of the present paper explores the category-theoretic power of these ideas and hence unifies the various generalisations which appeared after Scott's original observation. Recall that the Karoubi envelope of a category A is the category $\tilde{A}$ whose objects are idempotent maps $e = e \circ e$ of A and whose maps $f\, e \to d$ are maps in A such that $f = d \circ f \circ e$. Given categories A and B, define a semifunctor $F\, A \to B$ between them to be an assignment of objects of B to objects of A and of arrows of B to arrows of A which preserves composition, but not necessarily identities. If $F, G\, A \to B$ are semifunctors, a family $(\alpha_A\, FA \to GA)_{A \in \mathrm{ob}\, A}$ is called a normal natural transformation between F and G if, for every $f\, A \to B$, one has $Gf \circ \alpha_A = \alpha_B \circ Ff$ and $\alpha_A = G\,\mathrm{id}_A \circ \alpha_A \circ F\,\mathrm{id}_A$. These data define a 2-category SEM whose objects are categories, 1-arrows are semifunctors, and whose 2-arrows are the normal natural transformations.

The author recalls the folklore result that the 2-functor induced by ˜ between SEM and the 2-category of functors and natural transformations on the categories with splitting idempotents is a 2-equivalence. He uses this equivalence to express the notion of a Cartesian closed structure on $\tilde{A}$ in the language of semifunctors and normal natural transformations on A. Reviewed by G. Rosolini.

Barendregt, H. and van Mill, J. (1986). Are there countable topological combinatory algebras? *Nederl. Akad. Wetensch. Indag. Math.*, 48(3):233–241.
[MR88a:03030] The reader of this paper is not told what is meant by a (topological) combinatory algebra, but he is referred to the fundamental (and voluminous) book of Barendregt [The lambda calculus, revised edition, North-Holland, Amsterdam, 1984; MR 86a:03012]. However, familiarity with combinatory algebras is not absolutely necessary for the present note since, as the authors point out, "the question in the title remains open", while in fact two theorems are proved which have

their own interest and do not require any knowledge of combinatory algebras. The first theorem concerns "complete" partially ordered sets (cpo's), i.e. posets in which every chain has a supremum, and it states that for any infinite cpo D the function space $[D \to D]$ is always uncountable. This result is based on the well-known fact that any infinite poset contains an infinite antichain or an infinite ascending or descending sequence. In all three cases, the corresponding subposet (with one or two extra points added, if necessary) turns out to have uncountably many continuous monotone self-maps, which may be extended to the whole poset, provided one is dealing with a cpo.

The second result is of a purely topological nature: A countable space is presented in which only the identity and the constant functions are continuous. This space is obtained as the direct limit of a sequence of compact, connected and locally connected T_1-spaces that are constructed with the help of a countable family of pairwise orthogonal spaces with the same compactness and connectedness properties. Here two spaces X and Y are said to be orthogonal provided that for every open subset U of X, all continuous functions from U to Y are constant, and vice versa. For the construction of such a countable family of pairwise orthogonal countable spaces the authors use a nontrivial result due to K. Kunen [Trans. Amer. Math. Soc. 172 (1972), 299–306; MR 47 #3170] stating that the Stone-Cech compactification of $\mathbf{N}$ contains an uncountable antichain with respect to the Rudin-Keisler preorder on $\beta\mathbf{N}$, defined by $p \leq q \Leftrightarrow p = \{B \subseteq \mathbf{N} \colon \exists A \in q(f[A] \subseteq B)\}$ for some $f \in \mathbf{N}^{\mathbf{N}}$. Reviewed by Marcel Erne.

Blass, A. (1986). Functions on universal algebras. *J. Pure Appl. Algebra*, 42(1):25–28.

[MR87i:03134] Suppose V is a variety of algebras, $\mathcal{E}$ the classifying topos for V-algebras and U the universal V-algebra in $\mathcal{E}$. The main result is that the algebra of all functions from U^n into U is isomorphic to the algebra of polynomials in n variables over A (so U is functionally complete). The result is applied to the variety V of combinatorial algebras, i.e., algebras with one binary operation and constants S and K such that the laws $(Kx)y = x$ and $((Sx)y)z = (xz)(yz)$ hold. In this case, functional completeness implies that U, under a suitable interpretation, serves as a model for the untyped lambda calculus. Reviewed by S. Comer.

Curien, P.-L. (1986). Categorical combinators. *Inform. and Control*, 69(1-3):188–254.

[MR88f:03012] Summary: "Our main aim is to present the connection between λ-calculus and Cartesian closed categories in both an untyped and a purely syntactic setting. More specifically, we establish a syntactic equivalence theorem between what we call categorical combinatory logic and λ-calculus with explicit products and projections, with β- and η-rules as well as with surjective pairing. 'Combinatory logic' is of course inspired by Curry's combinatory logic, based on the well-known S, K, I. Our combinatory logic is 'categorical' because its combinators and rules are obtained by extracting untyped information from Cartesian closed categories (looking at arrows only, thus forgetting about objects). Compiling λ-calculus into these combinators happens to be natural and provokes only $n \log n$ code expansion. Moreover, categor-

ical combinatory logic is entirely faithful to β-reduction, whereas combinatory logic needs additional rather complex and unnatural axioms to be so. The connection easily extends to the corresponding typed calculi, where typed categorical combinatory logic is a free Cartesian closed category in which the notion of terminal object is replaced by the explicit manipulation of applying (a function to its argument) and coupling (arguments to build data in products). Our syntactic equivalences induce equivalences at the model level. The paper is intended as a mathematical foundation for developing implementations of functional programming languages based on a 'categorical abstract machine', as developed in a companion paper [G. Cousineau, the author and M. Mauny, Sci. Comput. Programming 8 (1987), no. 2, 173–202]."

Huet, G. (1986). Cartesian closed categories and lambda-calculus. In *Combinators and functional programming languages (Val d'Ajol, 1985)*, pages 123–135. Springer, Berlin.

[MR88b:03024] The author sketches a procedure to translate a categorical statement (usually a conditional statement about commuting diagrams that can be checked by diagram chasing) into equational statements. The procedure is illustrated by giving an equational axiomatization for Cartesian closed categories. The close relationship between the resulting axiomatization and intuitionistic sequent calculus is pointed out. The connection with the λ-calculus is described in detail. For the entire collection see MR 87j:03004. Reviewed by S. Comer.

Jankowski, A. W. (1986). Some modifications of Scott's theorem on injective spaces. *Studia Logica*, 45(2):155–166.

[MR88h:06012] Summary: "In a paper on the mathematical models for the Church-Curry λ-calculus, D. S. Scott [in Toposes, algebraic geometry and logic (Halifax, N.S., 1971), 97–136, Lecture Notes in Math., 274, Springer, Berlin, 1972; MR 53 #7879] proved the following theorem: A topological space X is an absolute extensor for the category of all topological spaces if and only if a contraction of X is a topological space of 'Scott's open sets' in a continuous lattice.

"In this paper we prove a generalization of this theorem for the category of $\langle\alpha,\delta\rangle$-closure spaces. The main theorem says that, for some cardinal numbers α,δ, absolute extensors for the category of $\langle\alpha,\delta\rangle$-closure spaces are exactly $\langle\alpha,\delta\rangle$-closure spaces of $\langle\alpha,\delta\rangle$-filters in $\langle\alpha,\delta\rangle$-semidistributive lattices (Theorem 3.5).

"If $\alpha=\omega$ and $\delta=\infty$ we obtain Scott's theorem (Corollary 2.1). If $\alpha=0$ and $\delta=\omega$ we obtain a characterization of closure spaces of filters in a complete Heyting lattice (Corollary 3.4). If $\alpha=0$ and $\delta=\infty$ we obtain a characterization of the closure space of all principal filters in a completely distributive complete lattice (Corollary 3.3)."

Lambek, J. (1986). Cartesian closed categories and typed λ-calculi. In *Combinators and functional programming languages (Val d'Ajol, 1985)*, pages 136–175. Springer, Berlin.

[MR88j:03010] From the introduction: "This article consists of two parts. Sections 1 to 4 give an exposition for computer scientists of one version of the relation between typed λ-calculi with surjective pairing and Cartesian closed categories. There is an emphasis on the logical point of view; categories are introduced as deductive systems with equations between proofs and, in the spirit of Haskell Curry, functional completeness is viewed as a jazzed-up version of the deduction theorem. Sections 5

and 6 are concerned with a natural numbers object in a Cartesian closed category and the question of how such an object can be presented equationally in the sense of Albert Burroni."

Reviewer's remarks: The controversial remarks about possible uses of Occam's razor at the end of Section 4 also arise in connection with computer implementation of typed λ-calculi, where the attempt to reduce the number of primitive operations often has high costs in memory and running time. This is especially the case with higher-order typed λ-calculi, where it is theoretically possible to define product types, disjoint sum types, existential types, the natural number type, etc., in terms of a dependent universal type; it would be too inefficient in practice to use these definitions in implementation. On the other hand, for purposes of using proof theory to prove the consistency of the system, which is desirable if the system is to be used as the basis of a theorem-prover, the theoretical possibility of using these definitions simplifies things enormously. Thus, whether or not Occam's razor should be used in this way depends on one's purpose. For the entire collection see MR 87j:03004.

Lambek, J. and Scott, P. J. (1986). *Introduction to higher order categorical logic.* Cambridge University Press, Cambridge.

[MR88a:03032] This concise book gives a careful presentation of the remarkable and recently developed connection between category theory (topoi) and logic (proof theory, type theory, and the λ-calculus). It thus neatly complements the recent book by M. Barr and C. Wells [Toposes, triples, and theories, Springer, New York, 1985; MR 86f:18001], which handled its topics well but did not present the connections of topoi with logic.

As Scott wrote the reviewer two years ago, the key plan of the present book is to present adjoint functors between (categories of) languages and the corresponding (categories of) structured categories so that commutative diagrams in a category correspond to linguistic equations between terms in the internal language of the category. Thus objects are logistic types, arrows are (equivalence classes of) proofs.

To prepare the ground, Part 0 of this book is a clear 40-page introduction to category theory, with emphasis on some of Lawvere's "slogans". Then Part I establishes the equivalence between the (category of) typed λ-calculi and the (category of) Cartesian closed categories equipped with a natural numbers object (NNO). This is a clear formalization of the more familiar informal observation that the λ-operation provides the adjunction between product and exponential required for Cartesian closure. The formal result requires that the Cartesian closed category have canonical products, against the usual custom, but so as to match the conjunction operation in the calculus, while the λ-calculus can have "unexpected" identities between types; that is, it need not be a "pure" λ-calculus. Technically, it is a typed $\lambda\eta$-calculus with product types and iteration, while morphisms of λ-calculi are suitable "translations". Here and elsewhere, the book adjudicates well between two otherwise isolated specialities.

On the way to these results, it is shown that the usual deduction theorem leads to the functional completeness of Cartesian closed categories: Every arrow $\varphi(x)\,1 \to B$ in an indeterminate $x\ 1 \to A$ is a composite $f \circ x$ for some f independent of x, much as in the corresponding result of combinatory logic. These arrows $\varphi(x)$ live in a "polynomial category" which can be regarded as the Kleisli category of a suitable cotriple.

The coherence problem for Cartesian closed categories (with NNO) is solved using the Church-Rosser theorem, essentially equivalent to Gentzen's cut elimination. This use of cut elimination was pioneered by Lambek, but subsequently confused in papers of M. E. Szabo.

The extended λ-calculus is also considered; the category of such calculi is isomorphic to a category of C-monoids, defined algebraically. One of Dana Scott's models for this untyped λ-calculus is presented: a category with an object $U \neq 1$ such that $U \cong U^U \cong U \times U$; thus every $f \in U$ is a function defined on the whole universe U. This idea, currently popular in computer science, has been with us for 50 years; the reviewer's long-standing impression that the idea is a fantasy was not altered by the presentation here.

Part II constructs the internal (Mitchell-Benabou) language of a topos, shows that it is an intuitionistic type theory and provides an adjunction between the category of such type theories and the category of (elementary) topoi with canonical subobjects; as before, these topoi have canonical products, these requirements being necessary for the adjunction to work. Sections on the "internal language at work" show how the language can be used to prove theorems about topoi without drawing elaborate commuting diagrams. (Objects of a category do not have elements, but one can pretend that they do, and argue with them intuitionistically.) This important idea, probably due to Joyal, might have been well illustrated here by a comparison of one of these linguistic proofs with the corresponding diagrammatic proof, but at this point the authors eschew diagrams—however, they do wisely emphasize the extraordinary fact that geometric ideas about topoi led unexpectedly to intuitionistic logic.

The free topos (corresponding to pure type theory) is constructed and emphasized, and is proposed as the "ideal universe" of a moderate intuitionist. It is remarked that this reconciles three philosophies of mathematics (intuitionism, Platonism, and formalism); the reviewer thinks that such philosophical issues are more subtle. Type theory is proposed as an appropriate foundation of mathematics. The reviewer notes that type theory is heavily linguistic, while the competing foundation provided by Zermelo-Fraenkel set theory uses only first-order logic and restricts the linguistic aspects to the replacement axiom (as in the comprehension axiom: there is a set consisting of all $x \in A$ with property $P(x)$, P given by a linguistic formula). This may account for the common preference of set theory over type theory; as W. S. Hatcher has remarked [The logical foundations of mathematics, Pergamon, Oxford, 1982; MR 84g:03003], "A basic criticism of type theory is its preoccupation with purely logical and linguistic questions."

Part II also includes topos semantics, sheaf semantics and the Henkin completeness theorem for higher-order logic, using the proof by P. H. G. Aczel [Contributions to mathematical logic (Hannover, 1966), 1–11, North-Holland, Amsterdam, 1968; MR 38 #5587], as well as a fascinating deduction of several basic intuitionist principles (e.g., if $\vdash p \vee q$, then $\vdash p$ or $\vdash q$); the proof is an elegant use of the Freyd cover of a topos T (with objects (X, f, U), X a set, U an object of T and f a function $f\, X \to \mathrm{Hom}(1, U)$.

Part III covers the representation of numerical functions (of several numerical variables) in a topos and in the corresponding type theory; for example (Gödel), f is numeralwise representable in pure classical type theory if and only if it is recursive. For the pure typed λ-calculus, not every recursive function is so representable, but the primitive ones and the Ackermann function are (using methods of W. W. Tait [J. Symbolic Logic 32 (1967), 198–212; MR 36 #2483]).

This wealth of material is supplemented by informative historical notes—not an easy task, given that a number of categorists have been negligent about publishing. There is an index, but the reviewer would have also welcomed a glossary of symbols used (there are many). The whole book is carefully designed to appeal both to logicians

and to categorists—even to the extent that the preface gives equal billing to two prior books: the difficult but definitive book by P. T. Johnstone [Topos theory, Academic Press, London, 1977; MR 57 #9791] and the easy but superficial and inadequate one by R. Goldblatt [Topoi. The categorical analysis of logic, North-Holland, Amsterdam, 1979; MR 81a:03063; op. cit., second edition, 1984; MR 85m:03002]. Reviewed by Saunders Mac Lane.

Moggi, E. (1986). Categories of partial morphisms and the λ_p-calculus (extended abstract). In *Category theory and computer programming (Guildford, 1985)*, pages 242–251. Springer, Berlin.

[MR875692] This item will not be reviewed individually. For details of the collection in which this item appears see 87j:68004.

Mulmuley, K. (1986). Fully abstract submodels of typed lambda calculi. *J. Comput. System Sci.*, 33(1):2–46. Twenty-fifth annual symposium on foundations of computer science (Singer Island, Fla., 1984).

[MR88f:68098] Let $\mathcal{L}$ be the usual language constructed for a typed lambda calculus. The classical model of $\mathcal{L}$ is $(D^\tau, , A)$, where, for each ground type κ, D^κ is isomorphic to the given ground domain, $D^{\alpha\to\beta}$ is the domain of continuous functions from D^α to D^β, is the usual function application and A is the standard interpretation of the terms of the language. This model is faithful or adequate in the sense that it "keeps the values" of the terms of the language that can be reduced to a base constant but it is not fully abstract in the sense that it does not reflect the "operational power" of the elements of the model.

The basic aim of this paper is to construct an extensional, fully abstract and algebraic model of typed lambda calculus which is a homomorphic retraction of the classical model, if the latter is based on complete lattices. The intuitive idea of the construction is to take for each element of $\mathcal{L}$ the maximal extension of D which includes its image and does not change the meaning of the model: although the idea is simple, the technical details turn out to be rather complicated.

The paper presents the relation of the constructed model with the model of R. Milner [Theoret. Comput. Sci. 4 (1977), no. 1, 1–22; MR 58 #16238] and the extension of the construction for the case of reflexive types. Reviewed by M. L. Laplaza.

Poigné, A. (1986). On specifications, theories, and models with higher types. *Inform. and Control*, 68(1-3):1–46.

[MR87m:68059] The theory of algebraic specifications of abstract data types is well developed and widely accepted as a tool for the development of software systems. However, its wider applicability is restricted by the lack of machinery to deal with functions of higher type. This paper is concerned with the mathematical foundations of specifications with higher types as well as with finding a suitable notion of a language. This paper (especially the second section) is an elaboration of ideas of Lambek (1980) and Scott (1980) on typed λ-calculus, Cartesian closed categories and on their equivalence. Typed λ-calculus is extended by a product structure and a binding structure to model composition of (formal) functions, resulting in the so-called Λ-calculus. The main results of the second section are the equivalence of the theory of the Λ-calculus and the theory of Cartesian closed categories and the equivalence of the categories of small Cartesian closed categories and Λ-theories. Cartesian closed categories, as well as the typed λ-calculus with finite products, are defined using two-level specifications, which are defined in Section 1 by an extension of the standard algebraic specifications.

In the third section, the author defines specifications and theories with higher types. He shows that the equivalence of the languages of Cartesian closed categories and Λ-calculus can be extended to languages which include the axioms of the respective higher type specifications.

The fourth section of the paper is concerned with the model theory of a given theory with higher types, in particular with questions about the existence of free models and completeness. Since the answers to these questions are positive for so-called intensional models (while unclear for the so-called extensional ones), the author suggests that intensional models be used for data type specification. He also suggests that, despite the equivalence of the theories of Cartesian closed categories and the Λ-calculus, the Λ-calculus is better suited for many practical purposes. Reviewed by Ana Pasztor.

Seely, R. A. G. (1986a). Higher order polymorphic lambda calculus and categories. *C. R. Math. Rep. Acad. Sci. Canada*, 8(2):135–139.

[MR88e:03018] On the basis of the polymorphic lambda calculus [J.-Y. Girard, "Interpretation fonctionelle et elimination des coupures de l'arithmetique d'ordre superieur", Ph.D. Thesis, Univ. Paris VII, Paris, 1972; per bibl.; J. C. Reynolds, Programming symposium (Paris, 1974), 408–425, Springer, Berlin, 1974; MR 56 #17187], the notion of a PL theory is introduced. A natural categorical semantics for these theories is given in the form of so-called PL categories. These consist of a pair (G, S) where S is a category, G is a category indexed by objects of S, and G, S satisfy some conditions. The categories of PL theories and PL categories are shown to be equivalent. The closure model of PL [N. J. McCracken, "An investigation of a programming language with a polymorphic type structure", Ph.D. Thesis, Syracuse Univ., Syracuse, N.Y., 1979; per bibl.] inside the graph model $P\omega$ [D. S. Scott, SIAM J. Comput. 5 (1976), no. 3, 522–587; MR 55 #10262] is described as a PL category. For part II see the following review. Reviewed by Henk Barendregt.

Seely, R. A. G. (1986b). Higher order polymorphic lambda calculus and categories. II. *C. R. Math. Rep. Acad. Sci. Canada*, 8(3):197–201.

[MR88e:03019] In this paper the PL theories of part I [same journal 8 (1986), no. 2, 135–139; see the preceding review] are extended with subtypes and equality types Eaa' (with intended interpretation $\{ba(b) = a'(b)\}$). The PL categories are extended accordingly. Again the two resulting notions of PLS theory and PLS category can be considered as equivalent categories. The closure model viewed as a PL category is expanded to a PLS category.

{Reviewer's remark: As shown by J.-Y. Girard ["Interpretation fonctionelle et elimination des coupures de l'arithmetique d'ordre superieur", Ph.D. Thesis, Univ. Paris VII, Paris, 1972; per bibl.], some extensions of the polymorphic lambda calculus are inconsistent. The author remarks that the PLS theories are consistent because there are PLS categories as models. This is slightly misleading. Even for the "inconsistent" theories mentioned there are closure models. The inconsistency refers to the fact that the intended interpretation (propositions as types) is violated: all types are inhabited and therefore by this interpretation all propositions are provable.} Reviewed by Henk Barendregt.

Taylor, P. (1986). Internal completeness of categories of domains. In *Category theory and computer programming (Guildford, 1985)*, pages 449–465. Springer, Berlin.

[MR875703] This item will not be reviewed individually. For details of the collection in which this item appears see 87j:68004.

Yokouchi, H. (1986). Retraction map categories and their applications to the construction of lambda calculus models. *Inform. and Control*, 71(1-2):33–86.

[MR88e:03020] Intending to generalize Scott's D_∞ construction, the author introduces retraction map categories (partial orders in an order enriched c.c.c. (Cartesian closed category) induced by D_∞-like embedding-projection pairs) and ε-categories by a subsequent completion-by-ideals process, which ensures that ε-categories are order enriched c.c.c.'s as well. Under certain conditions, examined in the main part of the paper, ε-categories will have reflexive objects, i.e. they will define λ-algebras, and enough points, i.e. they will define λ-models. The final sections reconstruct D_∞ and P_ω within this framework. Reviewed by Dean Rosenzweig.

Mosses, P. D. and Plotkin, G. D. (1987). On proving limiting completeness. *SIAM J. Comput.*, 16(1):179–194.

[MR88f:68097] The term limiting completeness means the property of a denotational semantics (of some programming language) to be consistent with an operational semantics also with respect to diverging computations. In this case the outputs given by partial finite computations are to converge to the full output specified by the denotational semantics. This notion was first introduced by C. P. Wadsworth. In his well-known approximation theorem for the pure lambda-calculus [same journal 5 (1976), no. 3, 488–521; MR 58 #21493], he showed that, in a standard semantics on a domain $\mathbf{D}$ satisfying $\mathbf{D} \cong [\mathbf{D} \to \mathbf{D}]$, the denotations of the approximate normal forms of a term do converge to the denotations of the term itself, proving the completeness (in this limiting sense) of the β reduction rule.

Wadsworth's method for proving this result required the introduction of the notions of labelled term and labelled β reduction which, although interesting in themselves, required the development of many syntactic details (in the proof of limiting completeness, for languages which allow self-application, direct methods, like structural induction of fixpoint induction, fail).

In this paper the authors give two alternative proofs of the approximation theorem for the pure lambda-calculus using two different approaches which do not require any particular syntactic apparatus.

In the first approach they introduce, in addition to the domain $\mathbf{D}$, a new domain $\mathbf{D}'$ defined as the initial solution of the equation $\mathbf{D}' \cong [\,[\mathbf{1}^\infty \to \mathbf{D}'] \to \mathbf{D}']$ (where $\mathbf{1}^\infty$ is the c.p.o. of natural numbers $0 \sqsubseteq 1 \sqsubseteq \cdots \sqsubseteq \infty$) which includes the same initial domain $\mathbf{D}_0$ used to start the construction of $\mathbf{D}$. The interpretation of a term in $\mathbf{D}'$, given by a function $\mathcal{V}'[\![\,]\!]\lceil\wr\sqcup]\!]$ $\mathbf{Exp} \to \mathbf{1}^\infty \to \mathbf{Env}' \to \mathbf{D}'$ (where $\mathbf{Exp}$ is the set of terms and $\mathbf{Env}' = \mathbf{Var} \to [\mathbf{1}^\infty \to \mathbf{D}'])$ can be seen as an intermediate semantics which takes also as argument a value k in $\mathbf{1}^\infty$. In some sense, $\mathcal{V}'[\![\mathcal{M}]\!](||)$ corresponds to a partial output produced after at most k steps of a reduction algorithm specified by the operational semantics. A sort of approximation theorem for $\mathbf{D}'$ can be proved in a relatively easy way. Lastly, the authors define a relation between $\mathbf{D}$ and $\mathbf{D}'$, relating the standard denotation of a term in $\mathbf{D}$ to its intermediate denotation at ∞, which allows one to establish the approximation theorem for the standard semantics.

In the second approach, which seems technically simpler, no intermediate semantics is required. A relation $\sim_\rho$ between **D** and **Exp** is defined (for all $\rho \in \mathbf{Env} = \mathbf{Var} \to \mathbf{D}$) such that $d \sim_\rho M$ if and only if $d \sqsubseteq \bigsqcup_{n \geq 0} \mathcal{V}[\![\mathcal{S}^{\backslash}(\mathcal{M})]\!]\rho$ (where $\mathcal{S}^{\backslash}(\mathcal{M})$ is the nth level of approximation of M and $\mathcal{V}[\![\rfloor\lceil\wr\sqcup]\!]$ is the standard interpretation function) and, for all $e \in \mathbf{D}$ and $N \in \mathbf{Exp}$, $e \sim_\rho N \Rightarrow d(e) \sim_\rho (MN)$. The relation $d \sim_\rho M$ can be interpreted roughly as "d approximates to $\mathcal{V}[\![\mathcal{M}]\!]$". To prove the approximation theorem it is then enough to show, by structural induction on M, that $\mathcal{V}[\![\mathcal{M}]\!] \sim_\rho \mathcal{M}$ (the induction hypothesis used in the paper is indeed a little stronger). The relation $\sim_\rho$ is defined inductively following the construction of the domain **D**.

The paper is very well organized and can be read easily; the most technical proofs appear in the appendix. Reviewed by M. Coppo.

Negri, M. (1987a). On a modified version of the ultraproduct and reduced product construction. *Boll. Un. Mat. Ital. A (7)*, 1(1):127–133.

[MR88e:03049] Summary (translated from the Italian): "We present two modifications of the ultraproduct and reduced product constructions that, under fairly general conditions, make a product of nontransitive relations transitive. We then obtain models of combinatory logic in which the terms and their reductions coexist on the same plane."

Negri, M. (1987b). Products of term models. In *Proceedings of the conferences on mathematical logic, Vol. 3 (Italian) (Siena, 1985; Padova, 1985; and Siena, 1986)*, pages 409–412, Siena. Univ. Siena.

[MR904920] This item will not be reviewed.

Pitts, A. M. (1987). Polymorphism is set theoretic, constructively. In *Category theory and computer science (Edinburgh, 1987)*, pages 12–39. Springer, Berlin.

[MR89c:03022] From the introduction: "The title of this paper ought really to be 'Polymorphism can be set theoretic, constructively', but the obvious reference to J. C. Reynolds' paper, 'Polymorphism is not set theoretic' [in Semantics of data types (Valbonne, 1984), 145–156, Lecture Notes in Comput. Sci., 173, Springer, Berlin, 1984; MR 86e:03016] was too tempting. The purpose of the paper is to prove particular kinds of completeness and full embedding theorems for the polymorphic lambda calculus. In order to explain the theorems we need to review some of the recent history of type polymorphism." For the entire collection see MR 88j:68002.

Proietti, M. (1987). Connections between partial maps categories and tripos theory. In *Category theory and computer science (Edinburgh, 1987)*, pages 254–269. Springer, Berlin.

[MR89a:18007] The author defines a doctrine of functional relations to be a category equipped with the following structures that make it resemble the category of sets and partial functions: Every two objects have a "product" $A \otimes B$, not a categorical product but one analogous to the Cartesian product of sets viewed in the category of sets and partial functions; also, there is a "domain classifier" T, playing the role of a one-element set. For every two objects, there is an "object of partial maps" ($A \rightharpoonup B$). In addition to morphisms subject to equations that describe the intended meaning of these constructs, there are morphisms $A \otimes A \to A$ analogous to the (partial) inverse of the diagonal inclusion and morphisms $(A \rightharpoonup T) \to A$ analogous to the partial function assigning to each singleton its element; these too are subject to suitable equations. From any doctrine of functional relations, the author constructs a topos whose objects are the domains in the doctrine, i.e., the morphisms $A \to T$, and whose morphisms are those of the doctrine but with suitably redefined sources and targets. He shows that this construction can (up to equivalence of categories) be decomposed in two steps: first, using the doctrine to build a tripos, and then converting the tripos into a topos by the method of J. M. E. Hyland, P. T. Johnstone and A. M. Pitts [Math. Proc. Cambridge Philos. Soc. 88 (1980), no. 2, 205–232; MR 81i:03102]. For the entire collection see MR 88j:68002. Reviewed by Andreas Blass.

Sanchis, L. E. (1987). Completeness of transfinite evaluation in an extension of the lambda calculus. *J. Symbolic Logic*, 52(1):243–275.
[MR88d:03021] Scott's D_∞-model for the type-free λ-calculus is based on continuous functions over complete lattices. The construction of D_∞ can also be done for complete partial orders (CPOs) rather than complete lattices [see, e.g., H. P. Barendregt, The lambda calculus, North-Holland, Amsterdam, 1984; MR 86a:03012]. In the present paper it is shown that in CPOs monotonic functions can do the job instead of continuous functions, provided that transfinite recursion is permitted. In order to obtain this result, the author defines θ-continuous functions, θ being a limit ordinal, as follows. Let K and H be domains and $f\ K \to H$ a monotonic function; f is termed θ-continuous if whenever $\{x\}_{<\theta}$ is a θ-chain in K, then $f(\sup_K\{x\}_{<\theta}) = \sup_H\{f(x)\}_{<\theta}$ (a θ-chain $\{x\}_{<\theta}$ is an indexed subset $\{x\ <\theta\}$ of a partially ordered set such that $x \subseteq x_{\prime}$ if $\ \leq {}'$). The set of θ-continuous functions is shown to coincide with the set of monotonic functions, for sufficiently high ordinals θ. A second, minor extension to Scott's theory is the incorporation of a conditional operator in the calculus.

The paper, which is self-contained, gives three different evaluations (semantics) for a proposed formal language of λ-like terms. The first, called basic evaluation, is a mapping from this language into a structure $\mathcal{H}^\theta$ of domains and function domains, θ being a fixed limit ordinal. The author claims that this is the natural semantics for the language. The other evaluations are mappings into a larger structure $\mathcal{D}$ which is "reflexive", i.e. isomorphic to the set of all θ-continuous functions from $\mathcal{D}$ into itself. $\mathcal{D}$ is constructed by means of transfinite recursion, using domains and projections in the style of Scott. The author uses the more complicated evaluations to show that the basic evaluation is "complete", meaning that when a term X evaluates to a function f and a term Y to an object x in the domain of f, then XY (or rather XY', with Y' a variant of Y) evaluates to $f(x)$. Reviewed by R. P. Nederpelt.

Scott, P. J. (1987). Computable functions in categories. In *Mathematical logic and theoretical computer science (College Park, Md., 1984–1985)*, pages 349–372. Dekker, New York.
[MR89a:03124] From the introduction: "One of the fundamental observations of categorical logic is that certain formal logics are really 'the same thing' as well-known categories. More precisely, in the case of higher-order theories considered here, we have the following correspondences between logics and categories: untyped

λ-calculi $\leftrightarrow$ C-monoids; typed λ-calculi $\leftrightarrow$ Cartesian closed categories; type theories $\leftrightarrow$ toposes. The correspondence in each case is mediated by a pair of functors setting up an equivalence or adjointness.

"We survey the general question of representing number-theoretic functions in various free categories. In so far as category theory delineates 'natural' notions, the recursion theories of such free categories are good candidates for 'natural recursion theories'. Some of this material is based on Part III of our book with J. Lambek [Introduction to higher order categorical logic, Cambridge Univ. Press, Cambridge, 1986; MR 88a:03032], although we have also included some recent work on categories with equalizers, natural numbers objects, and additional structures." For the entire collection see MR 88i:03005.

Carboni, A., Freyd, P. J., and Scedrov, A. (1988). A categorical approach to realizability and polymorphic types. In *Mathematical foundations of programming language semantics (New Orleans, LA, 1987)*, pages 23–42. Springer, Berlin.

[**MR89g:68043**] The realizability universe has become an important construct in higher-order logic, domain theory, and the study of polymorphic types.

The authors use a categorical calculus of relations, previously introduced and developed by themselves (with R. F. C. Walters) in prior works, in which not only realizability but also the higher-order lambda calculus interpretation of polymorphic types find an interesting theoretical setting. In this context both known and some new interpretations of polymorphism are obtained. The work begins with the consideration of a set K of partial endofunctions on the set of natural numbers $\mathbf{N}$, to which a regular category A of "assemblies" with initial object is associated. This category has not quotients of all equivalence relations: thus, the authors proceed to adjoin quotients by using their general calculus of relations. This is done by assigning to the mentioned category A, another regular category R in which all equivalence relations are "effective".

The next step in the work is the description of what happens when R has a higher-order logical structure. It is shown that if R is a topos, every partial recursive function is a restriction of a partial endofunction in D to an r.e. set. Conversely, if K is the collection of partial recursive functions, then the conditions given in the first definition of K are met, and R is a topos. This R is the "realizability universe".

This universe is carefully explored in the last part of the paper. It happens that in R, for every discrete object D there exists a subobject X of the natural number object N and an equivalence relation E on X such that D is isomorphic to X/E: moreover, quotients of discrete objects are also discrete. Thus all discrete objects in R are subobjects of a single object.

After introducing the definition of "modest object" as an object which is both discrete and separated (all definitions are given in the paper), the authors arrive at the result that in any topos, the full subcategory of discrete, and hence modest, objects is a reflective, Cartesian closed, internally complete subcategory. Moreover, in the particular case of R, as said above, all discrete, and hence modest, objects are subobjects of a single one. All this allows these subcategories of R to be the ones which provide the announced theoretical setting of polymorphism. The paper is well worked out and seems to be of importance; it is quite technical and requires background knowledge which is given in a good bibliography (some of the items in the bibilography are still preprints or to appear). Some explanatory remarks in the paper should probably have been given more expansion. For the entire collection see MR 89a:68005. Reviewed by Luis M. Laita.

Engeler, E. (1988). Representation of varieties in combinatory algebras. *Algebra Universalis*, 25(1):85–95.

[MR89f:08014] The author begins by giving a method of obtaining by strictly algebraic means an analogue of the notion of inner model in an algebraic structure with one binary operation. These models are called inner algebras. He then turns his attention to his model of combinatory algebra, $\mathbf{D}_A$ [see the author, same journal 13 (1981), no. 3, 389–392; MR 83f:03012]. He shows that an arbitrary algebraic structure $\mathbf{H}_0$ can be represented in $\mathbf{D}_A$ in such a way that a completion of $\mathbf{H}_0$ can be defined which is isomorphically representable as an inner algebra in $\mathbf{D}_A$ (provided that the cardinality of $\mathbf{H}_0$ does not exceed that of A). He then proceeds to consider a variety of algebraic structures which is given by a set Eq of equations, and shows how to characterize the corresponding completion by a set of equations Eq$'$ over $\mathbf{D}_A$. He concludes by showing that Eq$'$ is always equivalent to a single equation over $\mathbf{D}_A$ of the form $AX = BX$. Reviewed by J. P. Seldin.

Lambek, J. and Scott, P. J. (1988). *Introduction to higher order categorical logic.* Cambridge University Press, Cambridge. Reprint of the 1986 original.

[MR89a:03024] The original was reviewed [1986; MR 88a:03032]. Some typographical errors have been corrected.

Robinson, E. and Rosolini, G. (1988). Categories of partial maps. *Inform. and Comput.*, 79(2):95–130.

[MR90b:18010a] Joint review with (Curien and Obtułowicz, 1989). Categories as theories for (total or complete) algebras became powerful through the work of F. W. Lawvere. A similar research programme in the case of partial algebras caused many difficulties because of the lack of the Yoneda lemma and the impossibility of reducing properties of partial algebras (with signature) to corresponding purely functorial properties. Several steps having been carried out, the state is rather at the beginning (reaching its full complexity in the encoding of terms [and formulas], validity, and logical inference in an elementary topos, and in the passage from descriptive to structural classification of categories, which will occur as theories [resp., languages]).

Now the theories carry an enriched structure, which is much more complex and informative than in the total case, and the problem of characterizing them (equational-logically complete) by their approximability relative to the concrete category Par of sets and partial mappings has been given several different, although widely overlapping, solutions. Not noted by the authors is the early and later related research in the theory of semigroups, inductive groupoids and categories (e.g. of A. H. Clifford, Ch. Ehresmann, G. B. Preston, B. M. Schein, V. V. Vagner, and S. Armstrong, J. B. Fountain, M. V. Lawson, K. S. S. Nambooripad, respectively), as well as the more closely related research done by Y. Kawahara (1973), E. Schwab (1973) and J. Kastl (1975, 1979).

Both of the papers under review contain a comparative study of the abstract classes of categories which arise in this way. A typical representative is the notion of a partial Cartesian category in the sense of the second paper (C.-O.), also called pre-dht-symmetric ("pre" to that of the reviewer) and p-category with a one-element object in the first paper (R.-R.). It is a 9-tuple $(\underline{C}, \otimes, I, a, r, l, c, d, t)$, where the part $(\underline{C}, \otimes, I, a, r, l, c)$ is symmetric monoidal, $d\colon (-) \to (- \otimes -)$ a natural transformation ("diagonal") and $t\colon (-) \to I$ a lax natural transformation ("lax terminal") subject to certain axioms. Variants of it are partial categories (C.-O., carrying a partial order), dominical pointed categories (R. di Paola and A. Heller, having a bi-functor $\otimes$ and distinguished zero morphisms), and bicategories of partial maps (A. Carboni). For a related comparing axiomatic study cf. a paper by J. Schreckenberger [Beitrage Algebra Geom. No. 24 (1987), 83–98; MR 88m:18012].

In both papers an embedding of a category $\underline{A}$ with an admissible class $\mathcal{M}$ of monics of $\underline{A}$ into a certain derived category $\mathcal{M}$-Part $\underline{A}$ is constructed as a base for a representation theory. The close connection to the calculus of fractions (J.-L. Verdier), omitted here, is drawn in a forthcoming paper of the reviewer ["On certain classes of categories and monoids constructed from abstract Malcev clones, I", in Proceedings of the Vth Universal Algebra Symposium (Turawa, 1988), World Sci. Publishing, Singapore, to appear]. In (C.-O.) this construction is carried over to partial Cartesian closed categories and to partial toposes, which are proved to be equationally definable (using e.g. an internal equality $(\equiv)\colon A \otimes A \to A$ (to be called diagonal inversion according to B. Schepull (1976))). Reviewed by H.-J. Hoehnke.

Curien, P.-L. and Obtułowicz, A. (1989). Partiality, Cartesian closedness, and toposes. *Inform. and Comput.*, 80(1):50–95.
[MR90b:18010b] See (Robinson and Rosolini, 1988).

Freyd, P. (1989). Combinators. In *Categories in computer science and logic (Boulder, CO, 1987)*, pages 63–66. Amer. Math. Soc., Providence, RI.
[MR90m:03029] This paper is an introduction to combinatory logic from an algebraic point of view. The combinators and their reduction rules are given an algebraic motivation. The paper also contains an algebraic proof that the extensional principle (if ψ and ψ' are x-free terms such that $\psi x = \psi' x$ then $\psi = \psi'$) follows from a finite set of equations.

Reviewer's remark: Although the author expresses doubts about the newness of the proof of the extensional principle, the reviewer has never seen it and finds it interesting: if the proof is not new, it is an interesting presentation of an old proof. For the entire collection see MR 89m:18001. Reviewed by J. P. Seldin.

Gray, J. W. (1989). The integration of logical and algebraic types. In *Categorical methods in computer science (Berlin, 1988)*, pages 16–35. Springer, Berlin.
[MR91f:68135] Summary: "Three versions of the lambda calculus are discussed: typed lambda calculus, untyped lambda calculus, and polymorphically typed lambda calculus. It is shown that the syntax for these calculi can be described as initial algebras for suitable endofunctors. Models of the first two calculi are described as in J. Lambek and P. J. Scott's Introduction to higher order categorical logic [Cambridge Univ. Press, Cambridge, 1986; MR 88a:03032] and the PER model of the third is recalled. Algebraic types are described in terms of sketches. A main contribution

of the paper is the internalization of the theory of sketches in the category PER. Sketches themselves, models of sketches, categories of models of sketches and the category of all models of all sketches can all be realized in this setting. Next, dinatural transformations are reviewed in the context of the category PER and the relation between certain end formulas and initial algebras for finite signatures is discussed. The point of the theory of algebraic semantics in PER is to provide a setting for thinking about algebraic types and algebraic operations in conjunction with logical types and polymorphic operations. Two specific examples are discussed and a general principle for combining algebraic and polymorphic operations is proposed." For the entire collection see MR 90k:68003.

Hardin, T. (1989). Confluence results for the pure strong categorical logic CCL. λ-calculi as subsystems of CCL. *Theoret. Comput. Sci.*, 65(3):291–342.

[MR91h:03021] In the past few years a substantial amount of research has been devoted to a correct understanding of the relations between typed λ-calculi and Cartesian closed categories. The original motivating remark, apparently due to F. W. Lawvere , consisted in observing that the adjunction defining Cartesian closure for a category $\mathcal{C}$, i.e. $\hom_{\mathcal{C}}(A \times B, C) \cong \hom_{\mathcal{C}}(A, C^B)$, expresses the reduction of functions of finite arity to unary functions which have functions as values, which lies at the heart of λ-calculus and combinatory logic.

The first formal treatment of the equivalence between a simply typed λ-calculus with pairing and projection operators and Cartesian closed categories is to be found in a paper by J. Lambek [Ann. of Math. Logic 6 (1973/74), 259–292; MR 49 #5121].

From the adjunctions defining Cartesian closure it is possible, as usual, to extract an equational characterization of Cartesian closed categories which turns out to be interesting even when one forgets types (i.e., objects) in the equations, obtaining a formulation of what is called "strong categorical combinatory logic" (CCL $\beta\eta$ SP). It can be proved that CCL $\beta\eta$ SP is equivalent to the extension of type-free λ-calculus Λ_c obtained from ordinary λ-calculus by adding term-forming operations (where M and N are terms of Λ_c): $\langle M, N\rangle$, $\mathrm{fst}(M)$, $\mathrm{snd}(M)$, with reduction rules: (1) $\mathrm{fst}(\langle x, y\rangle) \to x$; (2) $\mathrm{snd}(\langle x, y\rangle) \to y$; (3) $\langle \mathrm{fst}(x), \mathrm{snd}(x)\rangle \to x$. An advantage of the categorical formulation is that there are no bound variables, as in combinatory logic, and it is well known that a correct treatment of bound variables is difficult to obtain in implementations.

However, a classical counterexample due to J. W. Klop (a new counterexample is given in the paper under review) shows that Λ_c does not have the Church-Rosser property (also called the confluence property), and this raises the problem of finding subsystems of categorical combinatory logic which are "isomorphic" to pure λ-calculus.

The main results obtained by the author are: (i) There is a confluent subsystem of CCL $\beta\eta$ SP which simulates the (meta-linguistic) operation of substitution of ordinary λ-calculus; (ii) it is possible to find a set $\mathcal{D}$ of terms in the signature of CCL $\beta\eta$ SP and subsystems of CCL $\beta\eta$ SP which can be used to simulate ordinary $\beta\eta$-reduction [resp., $\beta\eta$-reduction plus the rules (1) and (2) for the pairing operation] on a subset of $\mathcal{D}$ which is isomorphic to the set of terms of pure λ-calculus [resp., the set of terms of Λ_c].

The paper is of a rather technical nature, but the results are quite interesting. A new method, of interest for the general theory of term rewriting, is introduced in order to prove the confluence results in the paper. Reviewed by F. Cardone.

Jacobs, B. (1989). On the semantics of second order lambda calculus: from Bruce-Meyer-Mitchell models to hyperdoctrine models and vice-versa. In *Category theory and computer science (Manchester, 1989)*, pages 198–212. Springer, Berlin.

[MR91b:03029] This paper contains a study of ties between different types of models for second-order polymorphic λ-calculus. Models of the first type were constructed by K. B. Bruce, A. R. Meyer and J. C. Mitchell on the basis of set theory [cf. Inform. and Comput. 85 (1990), no. 1, 76–134]. They are called BMM-models. Models of the second type were constructed by Seely and are category-theoretic, based on hyperdoctrines, i.e., the so-called PL-categories are considered as models of higher-order polymorphic calculus [cf. R. A. G. Seely, J. Symbolic Logic 52 (1987), no. 4, 969–989; MR 89g:03019].

In the paper under review, by changing the conditions defining PL-categories the author constructs (categorical) λ_2-models and gives an interpretation with valuation functions of second-order λ-calculus in these models. Any such hyperdoctrine λ_2-model gives rise to a BMM-model. In the other direction, the author shows that any BMM-model gives rise to a categorical λ_2-model. If that λ_2-model is transformed into a BMM-model again by the same rules, the result is essentially the same as the initial BMM-model. For the entire collection see MR 90m:68002. Reviewed by M. S. Burgin.

Lamarche, F. (1989). A simple model of the theory of constructions. In *Categories in computer science and logic (Boulder, CO, 1987)*, pages 201–216. Amer. Math. Soc., Providence, RI.

[MR91b:03030] Introduction: "The theory of semigranular categories and aggregates [the author, C. R. Math. Rep. Acad. Sci. Canada 10 (1988), no. 2, 89–94; MR 89j:68100; "Modelling polymorphism with categories", Ph.D. Thesis, McGill Univ., Montreal, PQ, 1988; per bibl.] has been developed as a tool to provide models for polymorphic type theories, following the lead of J.-Y. Girard [Theoret. Comput. Sci. 45 (1986), no. 2, 159–192; MR 88j:68105]. In this paper we present a simple model (meaning that one can calculate simple types) for the theory of constructions [T. Coquand, "Une theorie des constructions", These de 3eme cycle, Univ. Paris VII, Paris, 1985; per bibl.] which uses only the restriction of the theory to posets (the formal system therein is not amenable to a semantical treatment; it has been modified for this purpose by more than one author). We do not give the proofs in full detail, but we feel we provide enough information for the dedicated reader to fill in the gaps without too much trouble." For the entire collection see MR 89m:18001.

Lambek, J. (1989). Multicategories revisited. In *Categories in computer science and logic (Boulder, CO, 1987)*, pages 217–239. Amer. Math. Soc., Providence, RI.

[MR90e:18010] From the summary: "This is an account of the origin of multicategories and their relation to algebra, logic and linguistics. It is shown that a multicategory is completely characterized by its internal language and how this language may be exploited to obtain properties of the tensor product and internal hom.

There is a discussion of what happens when some or all of Gentzen's structural rules are imposed." For the entire collection see MR 89m:18001. Reviewed by R. H. Street.

Meseguer, J. (1989). General logics. In *Logic Colloquium '87 (Granada, 1987)*, pages 275–329. North-Holland, Amsterdam.

[MR90j:03112] The present paper takes new steps towards a general theory of logics based on category theory which uses categorical structures as logics, while mappings interpret different logics or proof calculi. Papers by J. A. Goguen and R. M. Burstall [in Logics of programs (Pittsburgh, PA, 1983), 221–256, Lecture Notes in Comput. Sci., 164, Springer, Berlin, 1984; see MR 86a:68004] and F. W. Lawvere [Proc. Nat. Acad. Sci. U.S.A. 50 (1963), 869–872; MR 28 #2143] must be mentioned as fundamental for the spirit of the present approach. The notions of entailment system, institution, logic, proof calculus (associating with each theory a proof-theoretic structure), and logical system (consisting of a logic together with an associated proof calculus) are defined. Within this context it follows that mapping of logics means the mapping of their underlying entailment and institution components. The same is true for proof calculi and logical systems. Section 4 ends with a complex commutative diagram, relating a number of categories of entailment systems, institutions, logics, proof calculi, and logical systems. Section 5 deals with categorical logics, the formal concept allowing a great model-theoretic flexibility. Many important particular cases are included: equational logics, the logic of typed lambda calculus, higher-order intuitionistic logic, the logic of Martin-Löf type theory, logics of Girard-Reynolds polymorphic lambda calculus, Girard's linear logic, and others.

Section 6 offers a categorical axiomatization of the strong view of logic programming (LP). In the weak view of LP, the computation in the machine and the deduction in logic are made equivalent. In the strong view of LP, this equivalence is extended to satisfaction in the standard model. The strong LP categorical axiomatization includes the weak one, for which the models are proof-theoretic structures. Mathematical semantics, query completeness, and operational semantics for LP languages are defined. First- and higher-order sublanguages of LP are discussed. Many substantial and illuminating examples are presented, and many fundamental results and relations come naturally. For example, (a) functional languages can be understood as LP languages in the strong sense of the axiomatization; (b) Martin-Löf type theory can be viewed as a categorical logic on the category of locally Cartesian closed categories; (c) type theory and concurrency are specific applications in the present approach. For the entire collection see MR 90d:03004. Reviewed by Neculai Curteanu.

Mitchell, J. C. and Scott, P. J. (1989). Typed lambda models and Cartesian closed categories (preliminary version). In *Categories in computer science and logic (Boulder, CO, 1987)*, pages 301–316. Amer. Math. Soc., Providence, RI.

[MR90f:03031] This paper considers lambda calculi with types generated from a type 1 and given ground types using $\times$, $\to$ and the projection functions. Postulates include those of $\lambda\eta$-equality and the theory allows types such as $a \to b$ and $a \times b$ to be identified.

The authors review the interpretation of such lambda calculi in set theoretic structures called Henkin models and in Cartesian closed categories (ccc's). The authors see the latter interpretations as a natural generalization of the former.

The primary points of comparison are the interpretation functions and soundness and completeness theorems associated with each kind of structure. The "meaning function" for ccc's and the status of extensionality over ccc's are discussed. For the entire collection see MR 89m:18001. Reviewed by Martin W. Bunder.

Sato, H. (1989). E-ccc: between ccc and topos. *Theoret. Comput. Sci.*, 64(1):55–66.

[MR90e:68068] The author defines an E-ccc to be a Cartesian closed category together with a subclass E of its objects such that any parallel pair of arrows with codomain in E has an equalizer, and if both the domain and codomain are in E then so is the equalizer. So E-ccc's lie between Cartesian closed categories and Cartesian closed categories with all finite limits. They are not close to toposes.

In computer science terms the idea is to allow data types defined by some kinds of equations, but not by all.

The author introduces an extension of typed lambda calculus, allowing some equational subtype constructors, and shows that E-ccc's relate to this extension as Cartesian closed categories relate to typed lambda calculus. The paper refers to J. Lambek and P. J. Scott's book Introduction to higher order categorical logic [Cambridge Univ. Press, Cambridge, 1986; MR 88a:03032] and is very much in the spirit of that book. Reviewed by Colin McLarty.

Scott, P. J. (1989). Functorial polymorphism and semantic parametricity. *Diagrammes*, 22:77–90. Journées E.L.I.T. (Esquisses, Logique et Informatique Théorique), Vol. 1 (Paris, 1988).

[MR91g:03037] Summary (translated from the French): "We describe a functorial interpretation of the parametric polymorphism suggested by Girard's system F. In this framework, types are multivariant functors, terms are dinatural transformations, and quantifiers are ends of functors, all defined over some complete (in the appropriate sense) Cartesian closed category. Attempts to find a notion of composition of dinatural transformations lead us to some interesting semantic criteria for parametricity. We also present a compositional model in the effective topos."

Streicher, T. (1989). Independence results for calculi of dependent types. In *Category theory and computer science (Manchester, 1989)*, pages 141–154. Springer, Berlin.

[MR91a:68184] Summary: "On the basis of categorical semantics for impredicative calculi of dependent types we prove several independence results. In particular, we prove that there exists a model where all syntactical concepts can be interpreted with one exception: in the model the strong sum of a family of propositions indexed over a proposition need not be a proposition. The method of proof consists of restricting the set of propositions in the well-known ω-Set model due to E. Moggi." For the entire collection see MR 90m:68002.

Abramsky, S. (1990). Abstract interpretation, logical relations, and Kan extensions. *J. Logic Comput.*, 1(1):5–40.

[MR93i:68125] This paper presents a general approach to abstract interpretation for typed lambda calculus with recursion (fixed-point operators) that is based on so-called logical relations. These are hereditarily defined relations that preserve function application and are closed under lambda abstraction. Although the introduction and some of the discussion are accessible to a wider audience, this is essentially a technical paper. Familiarity with both domain theory and certain aspects of category theory is assumed.

The main worked examples of abstract interpretation are strictness analysis (whether a function uses its arguments) and termination. For these examples, the method gives more direct proofs than the traditional approach using continuous functions instead of relations. Section 6, which constitutes half the paper (by length), considers abstraction functions that pick out the best approximation (in a finite domain) to a given element (of an infinite domain). The construction of abstraction functions is closely related to Kan extensions from category theory, hence the title of the paper. Reviewed by John C. Mitchell.

Bainbridge, E. S., Freyd, P. J., Scedrov, A., and Scott, P. J. (1990a). Corrigendum: "Functorial polymorphism". *Theoret. Comput. Sci.*, 71(3):431.

[MR91g:03129b] See MR 91g:03129a. The corrigendum concerns two diagrams on page 40 and three lines on page 52 of the paper.

Bainbridge, E. S., Freyd, P. J., Scedrov, A., and Scott, P. J. (1990b). Functorial polymorphism. *Theoret. Comput. Sci.*, 70(1):35–64. Fourth Workshop on Mathematical Foundations of Programming Semantics (Boulder, CO, 1988).

[MR91g:03129a] This is an interesting and significant paper on the theory of polymorphic type systems for programming languages. The main topic is the semantics of the polymorphic (or second-order) typed lambda calculus, developed independently by J.-Y. Girard [in Proceedings of the Second Scandinavian Logic Symposium (Oslo, 1970), 63–92, North-Holland, Amsterdam, 1971; MR 53 #12895] and J. C. Reynolds [in Programming Symposium (Paris, 1974), Lecture Notes in Comput. Sci., 19, Springer, Berlin, 1974; MR 56 #17187]. As shown by Reynolds in more recent work, this calculus does not have classical set-theoretic models of the expected form. Therefore, it is a well-motivated problem to investigate the semantics of this language in alternative frameworks. This paper studies interpretations in which types denote multi-argument functors (possibly covariant in some arguments and contravariant in others) over some Cartesian closed category, and terms denote the appropriate form of "dinatural" transformation. These interpretations are intuitively satisfying in that they clearly point out a certain uniformity to the functions definable in the calculus, traditionally referred to in the related literature as "parametricity".

While dinatural transformations exhibit the pleasing uniformity property just mentioned, there is an annoying technical inconvenience: in general, dinatural transformations do not compose to form dinatural transformations. This problem is circumvented (but not solved) by working within various categories, i.e., by choosing not to consider all functors and transformations, but only those that are represented internally within some sufficiently rich category (and possibly satisfying additional conditions). A particularly significant category for the theory of polymorphism, and one used extensively in the paper, is the category PER of partial equivalence relations over the natural numbers. The interpretation of the syntax relative to the category PER exhibits some significant uniformity properties which are explored in detail in the paper. There are also some surprising results about the representation of recursively defined types in this structure. Specifically, a type A defined by the equation $A = F(A)$ for some functor F is traditionally interpreted as an initial F-algebra. The authors show that for any F represented in PER in a certain way (and this includes a very wide class of functors), the initial F-algebra in PER is defined by the nonrecursive polymorphic type expression $\forall\alpha((F\alpha \Rightarrow \alpha) \Rightarrow \alpha)$. Reviewed by John C. Mitchell.

Gray, J. W. (1990). Initial algebra semantics for lambda calculi. In *Mathematical foundations of programming semantics (New Orleans, LA, 1989)*, pages 418–439. Springer, Berlin.

[MR91h:68114] Summary: "Ordinary and polymorphic typed lambda calculi are constructed as initial algebras for suitable endofunctors. The semantics is realized as the unique morphism from the initial algebra to an appropriate semantic algebra. In the case of the polymorphic lambda calculus, this semantic algebra is constructed from the category **bPER**˙ of pointed partial equivalence relations." For the entire collection see MR 91e:68008.

Hyland, J. M. E., Robinson, E. P., and Rosolini, G. (1990). Algebraic types in PER models. In *Mathematical foundations of programming semantics (New Orleans, LA, 1989)*, pages 333–350. Springer, Berlin.

[MR91h:68128] Summary: "Huet has conjectured that the interpretations of a class of types (the 'algebraic types') in the PER model on the natural numbers for the second-order lambda calculus are in a certain sense the initial algebras. In this paper we examine several different PER models, and show that Huet's conjecture holds in each." For the entire collection see MR 91e:68008.

Longo, G. and Moggi, E. (1990). A category-theoretic characterization of functional completeness. *Theoret. Comput. Sci.*, 70(2):193–211.

[MR91g:03036] In this paper, the authors discuss category-theoretic modelling of various kinds of combinatory algebras, thought of as algebraic counterparts to abstract functional languages. A key ingredient in their study is the generalization to Cartesian closed categories (CCCs) of two results from classical recursion theory: Kleene's S^m_n theorem and Rogers' notion of acceptable Gödel numbering; this is summarized in their notion of principal morphism and its variants. These morphisms are then used to characterize (among applicative structure objects $\mathcal{A}$ in CCCs) such notions as lambda and combinatory algebras, lambda models, etc. in the familiar retract-pair style (e.g., if $\mathcal{A}^{\mathcal{A}}$ is a retract of $\mathcal{A}$ in an appropriate CCC, then $\mathcal{A}$ is a model of the $\lambda\beta$ calculus).

The authors discuss an especially important class of ambient CCCs in the characterization theorems above: the category PER($\mathcal{A}$) of partial equivalence relations on the applicative structure $\mathcal{A}$. In particular, they discuss the characterization of lambda and combinatory models as objects in this category. For example, for the case of $\mathcal{A}$ = the natural numbers under Kleene brackets (as a partial applicative structure), the fact that PER($\mathcal{A}$) is a CCC is essentially Kleene's S^m_n theorem. Finally the authors discuss various higher-order partial recursive functionals (e.g., the Kleene-Kreisel countable functionals) over certain CCCs of Scott domains, and relate these to principal morphisms and the Myhill-Shepherdson theorem. Reviewed by Philip J. Scott.

Mulry, P. S. (1990). Categorical fixed point semantics. *Theoret. Comput. Sci.*, 70(1):85–97. Fourth Workshop on Mathematical Foundations of

Programming Semantics (Boulder, CO, 1988).

[MR91c:03049] The author studies fixed point combinators using techniques from category theory. In Section 1, he discusses fixed point combinators $Y_D\colon D^D \longrightarrow D$ in various categories (e.g., the effective topos, effective Scott domains, etc.) from a purely Cartesian-closed-category viewpoint. This part is technically related to an early paper of Lawvere on categorical diagonal arguments; in the case of effective Scott domains, it yields Kleene's first and second recursion theorems.

In later sections, the author studies algebraic properties of the operators Y_D. Since recursion involves solving "domain equations" similar to $D^D \simeq D$, $D \not\simeq 1$, problems of co- and contravariance in the variable D are usually resolved by moving to categories of embedding-projection pairs. An alternative is to meet contravariance head-on by using multivariant functors and multivariant (dinatural) transformations. This general approach (as a semantics of polymorphism) has been discussed in a paper by E. S. Bainbridge, P. J. Freyd, A. Scedrov, and the reviewer [same journal 70 (1990), no. 1, 35–64]. In this setting, least fixed point combinators (for domains) yield dinatural transformations $(\,)^{(\,)} \longrightarrow \mathrm{id}$. This phenomenon was studied in the above-mentioned paper [op. cit., p. 60] and, independently of that paper, is treated in detail in Sections 2–4 of the paper under review (for various kinds of domains). For example, the author shows that in certain order-enriched categories (e.g., domains enriched over ω-CPOs) we need not have natural numbers objects (NNOs) but we do have ordered NNOs. He then proves that, in this enriched-category setting, once again least fixed point combinators yield dinatural transformations $(\,)^{(\,)} \longrightarrow \mathrm{id}$. The last section verifies that the phenomena above actually hold in the category of effective Scott domains. Reviewed by Philip J. Scott.

Obtułowicz, A. (1990). The idea of polymorphism: an algebraic approach. *Bull. Polish Acad. Sci. Math.*, 38(1-12):199–207.

[MR93g:68079] Summary: "A notion of a polymorphic structure is introduced to provide an algebraic characterization of models of polymorphic lambda calculi."

Seely, R. A. G. (1990). Polymorphic linear logic and topos models. *C. R. Math. Rep. Acad. Sci. Canada*, 12(1):21–26.

[MR90m:03109] Summary: "We give a definition of a 'linear fibration', which is a hyperdoctrine model of polymorphic linear logic, and show how to internalise the fibration, generating topos models. This gives a constructive set theoretical context for the logic of Petri nets, as recently developed by N. Marti-Oliet and J. Meseguer. Also, we sketch how this can be further extended to include the exponential operator !. In this context, the topos model we construct can be embedded in the model constructed by A. M. Pitts."

Asperti, A. and Longo, G. (1991). *Categories, types, and structures.* MIT Press, Cambridge, MA. An introduction to category theory for the working computer scientist.

[MR93i:18001] Category theory is playing an increasingly influential role in computer science, notably in programming language theory and design. Indeed, categories and categorical logic provide a uniform, powerful mathematical framework to deal with many difficult problems in the foundations of computing. Conversely, for good or ill, computer science is itself heavily influencing the work of many categorists.

The book under review provides a rapid introduction to aspects of category theory relevant to the logical theory of functional languages: lambda calculus, type theory, proof theory, and denotational semantics. The neophyte reader will come across a dizzying array of recent models and constructions: e.g., categorical semantics of Girard's linear logic, indexed category models for polymorphic (second-order) lambda calculus, internal partial equivalence relation (per) models, etc.

Part I (Categories and structures) is a rapid course in category theory, covering many nontrivial topics, ending with indexed and internal categories (simplified by ignoring categorical coherence problems, and defining concepts up to equality rather than up to isomorphism). Examples in Part I are related to computer science and logic: there are rapid surveys of Cartesian closed categories (= ccc's), partial ccc's, Scott and Girard domains, pers, monoidal closed and $*$-autonomous categories, T-algebras, and data types. Categorical properties of Girard's linear logic are discussed in detail.

Part II (Types as objects) consists of five advanced topics: Chapter 8 has ccc-semantics for typed lambda calculus, Gentzen's cut-elimination theorem, and fixed-point combinators. Chapter 9 continues with semantics of untyped lambda calculus, pers, and an interesting complete axiomatization of the valid isomorphisms in arbitrary ccc's (the proof uses a highly nontrivial characterization of invertible untyped lambda terms, due to Dezani). There is a discussion of implementing the lambda calculus using CAM (the categorical abstract machine), an abstract environment machine which efficiently compiles using categorical combinators. Chapter 9 ends on higher recursion theory and Ershov's enumerated sets; unfortunately no mention is made of relevant work of P. Mulry.

Chapter 10 sketches the Plotkin-Smyth method of solving recursive domain equations (constructing fixed points of functors) by passing to enriched categories of embedding-projection pairs.

Chapters 11 and 12 are a very formal treatment of categorical models of Girard's second-order lambda calculus, and a detailed comparison of external Seely models (using indexed categories) versus the somewhat mysterious internal approach (in which the notion of internally complete ccc is used). Although in Sets, the only internally complete small ccc's are posets (i.e. trivial categories), this need not be so in a topos (e.g. Hyland's realizability topos), thus leading to interesting conceptual frameworks for polymorphism.

Reviewer's remarks: In my opinion, the Mac Lane-inspired subtitle of the book: An introduction to category theory for the working computer scientist, is definitely a misnomer. Although the topics covered are important and exciting, the authors seem to assume "working computer scientists" are mostly logicians and type theorists, which is rather dubious.

Applications of category theory to computer science span a large—and growing—field, a lot of it not explicitly logic-oriented. These aspects are too briefly hinted at in the authors' bibliography. For example, over the past twenty years, there have been fundamental applications to (i) Algol-like languages, (ii) categorical programming languages, (iii) flowchart and additive semantics, (iv) algebraic theories, as well as a large literature on categories in automata, control theory, and concurrency theory.

Unfortunately, the book sometimes has an air of being hastily written: some discussions are cursory, while some exercises occur before the appropriate definitions. The sections on internal and indexed categories are quite syntactical and may not please

some categorists. On the other hand, many proofs are purely equational, reinforcing the authors' computational orientation.

Nonetheless, this is a useful reference book. The authors manage to cover an enormous range of current topics. The exercises are quite challenging; moreover, some material (particularly on linear logic, polymorphism, and domain theory) is presented here for the first time in book form. Some committed categorists may not always agree with the authors' viewpoints, but a diligent reader of this book will be able to rapidly progress to the research literature. Reviewed by Philip J. Scott.

Berardi, S. (1991). Retractions on dI-domains as a model for Type:Type. *Inform. and Comput.*, 94(2):204–231.

[MR92h:03018] The assumption that a type is a type (the Type:Type assumption) is a very common one in the typed lambda-calculus. However, the problem arises of finding models for such a calculus. In this paper, the $\lambda\beta p$-calculus is considered and a class of models is determined. To do this, only dI-domains are taken into account. Here, below each compact element there are finitely many elements. By means of the characterization of stable functions by traces it is shown that the range of a stable retraction on a dI-domain is still a dI-domain. This allows one to construct a special stable retraction the range of which consists of all retractions. This stable retraction is used as an interpretation of the constant p in the $\lambda\beta p$-calculus. Since any type can be interpreted as a retraction, this approach models the Type:Type assumption in a general way. Reviewed by Peter Bachmann.

Bethke, I. (1991). Finite type structures within combinatory algebras. *Ann. Pure Appl. Logic*, 55(2):101–123.

[MR92m:03014] Inside a combinatory algebra (ca), the author defines certain intensional and extensional type structures over ω. (Note: in the definition of ET_{n+1} subscript 0 is missing from =.) If these classes coincide the algebra is called ft-extensional. Criteria for ft-extensionality are given and some well-known ca's, including D_A, P_ω, T^ω, H_ω and certain D_∞-models, are shown to be ft-extensional. Examples of ca's without this property are given. Finally, ft-extensionality is compared to and shown to be independent of extensionality and weak extensionality. Reviewed by Martin W. Bunder.

Goguen, J. A. (1991). Types as theories. In *Topology and category theory in computer science (Oxford, 1989)*, pages 357–390. Oxford Univ. Press, New York.

[MR1145782] This item will not be reviewed individually. For details of the collection in which this item appears see 93h:68004.

Gray, J. W. (1991a). Order-enriched sketches for typed lambda calculi. In *Category theory (Como, 1990)*, pages 105–130. Springer, Berlin.

[MR93h:68091] A sketch consists of a directed graph together with some formal cones and diagrams. In a model of a sketch, these formal cones are interpreted as universal cones and the formal diagrams as commuting diagrams. Originally such sketches (in somewhat different formulation) were introduced by Ehresmann as presentations of theories. More recently [see, e.g., M. Barr and C. F. Wells, Category theory for computing science, Prentice Hall, New York, 1990; MR 92g:18001], they have been put forward as presentations of datatypes. It is in this setting that the author seeks to describe three typed lambda calculi: the simply typed and polymorphic typed calculus and PCF.

In order to model reductions, the notion of sketch is extended to include "formal 2-cells" and "formal quasi-limit cones". These can be modelled in an arbitrary 2-category, but only the order-enriched case is of interest here. The above calculi are then described by long series of formal quasi-cones forming such order-enriched sketches. For the entire collection see MR 93b:18001. Reviewed by Bart Jacobs.

Gray, J. W. (1991b). Products in PER: an elementary treatment of the semantics of the polymorphic lambda calculus. In *Category theory at work (Bremen, 1990)*, pages 325–340. Heldermann, Berlin.
[MR93b:03017] This paper is an exposition of the interpretation of the polymorphic lambda-calculus in the category PER of partial equivalence relations on the natural numbers (the morphisms being (equivalence classes of indices for) partial recursive functions that respect the equivalence relations). The author begins by defining the syntax of the polymorphic lambda-calculus. Then he defines PER and the larger category of ω-sets. He proves their most important properties, particularly the fact that PER is closed under internal products indexed by PER itself. (The precise meaning of this closure assertion is carefully explained.) Finally, he presents the PER semantics of the polymorphic lambda-calculus. For the entire collection see MR 92i:18001. Reviewed by Andreas Blass.

Hatcher, W. S. and Tonga, M. (1991). Pairings on lambda algebras. *Z. Math. Logik Grundlag. Math.*, 37(4):343–352.
[MR93c:03015] It is well known that, in the lambda calculus, a pairing operation with the properties $p_1[a,b]=a$ and $p_2[a,b]=b$ can be defined, but that the addition of the surjective property $[p_1a, p_2a]=a$ is, in general, inconsistent. This paper introduces λ-π-algebras, which have pairing operators with the three properties above as primitives. These algebras can be formulated as monoids. The authors claim significant methodological and practical advantages, in that many results on λ-monoids can be transferred to the "more practically useful" λ-π-algebras. Not mentioned is the fact that one of the more useful properties of the λ-calculus, the Church-Rosser theorem, does not apply in λ-π-algebras. Reviewed by Martin W. Bunder.

Hoofman, R. and Schellinx, H. (1991). Collapsing graph models by preorders. In *Category theory and computer science (Paris, 1991)*, pages 53–73. Springer, Berlin.
[MR1135332] This item will not be reviewed individually. For details of the collection in which this item appears see 92g:68005.

Jacobs, B. (1991). Semantics of the second order lambda calculus. *Math. Structures Comput. Sci.*, 1(3):327–360.
[MR93c:68060] Summary: "In the literature there are two main notions of model for the second order λ-calculus: one by Bruce, Meyer and Mitchell (the BMM-model,

for short) in a set-theoretical formulation, and one category-theoretical by Seely. Here we generalise Seely's notion, using semifunctors and semi-adjunctions from Hayashi, and introduce $\lambda 2$-algebras, $\lambda\eta 2$-algebras, $\lambda 2$-models and $\lambda\eta 2$-models, similar in a way to the untyped λ-calculus. Nonextensional abstraction of both term and type variables is described by semi-adjunctions (essentially as in Martini's thesis). We also show that for second order sums, the β- (and commutation!) conversions correspond to semi-functoriality and that the (additional) η-conversion corresponds to ordinary functoriality.

"In the above framework various examples—well-known ones and variations—are described. Also, we determine the place of the BMM-models; an earlier version of the latter has been reported by us."

Jacobs, B., Moggi, E., and Streicher, T. (1991). Relating models of impredicative type theories. In *Category theory and computer science (Paris, 1991)*, pages 197–218. Springer, Berlin.

[MR1135341] This item will not be reviewed individually. For details of the collection in which this item appears see 92g:68005.

Mitchell, J. C. and Moggi, E. (1991). Kripke-style models for typed lambda calculus. *Ann. Pure Appl. Logic*, 51(1-2):99–124. Second Annual IEEE Symposium on Logic in Computer Science (Ithaca, NY, 1987).

[MR92a:03017] One of the long-standing bugbears for model theory has been the formal problems that arise when certain of the carriers for the model are allowed to be empty. This paper provides a solution for the problem in the case of theories of the simply typed lambda calculus. The approach adopted is to use a model in which types are interpreted over many different possible worlds, as in Kripke's semantics for intuitionistic and modal logics. Thus a type can be empty at one possible world, and inhabited at another. A good formal system is given, and soundness and completeness results proved. The authors also explain the close connections between their theory and results from categorical logic, of which much of the paper can be regarded as an attractively worked, concrete, and useful example. For the entire collection see MR 91k:03004. Reviewed by Edmund Robinson.

Moggi, E. (1991). Notions of computation and monads. *Inform. and Comput.*, 93(1):55–92. Selections from the 1989 IEEE Symposium on Logic in Computer Science.

[MR92j:03011] This is an interesting and important paper which has already spawned a good deal of related research investigating the strengths, and of course the weaknesses, of the ideas presented definitively here.

The heart of the author's approach to the semantics of programming languages is a distinction between the interpretation of the set of values of a given type, and the set of computations of that type. He argues that certain relationships hold between these sets, so that they form a Kleisli triple, and hence give rise to a monad.

He gives a metalanguage for manipulating these semantics. This is based on a call-by-value form of the typed lambda calculus, in which there are terms to denote the trivial computation representing a value of a given type, and a "let" construct to handle composition of operations. It is intended that the semantics of an expression of type $A \to B$ in call-by-value language should be given as a function $[\![A]\!] \to T[\![B]\!]$, where $T[\![B]\!]$ represents computations of type B. In a call-by-name regime, the same expression would be given semantics of the form $T[\![A]\!] \to T[\![B]\!]$.

The author shows that the metalanguage can be interpreted in a Cartesian closed category with a strong monad (the strength of the monad being related to substitution laws). Indeed, interpretations of the metalanguage correspond naturally to such structures. This is important because it allows him to deal with conventional domain-theoretic models in the same formal setting, as well as (potentially) to handle the sheaf-theoretic models aimed at dealing with local variables and declarations. For the entire collection see MR 92a:68003. Reviewed by Edmund Robinson.

Phoa, W. (1991a). From term models to domains. In *Theoretical aspects of computer software (Sendai, 1991)*, pages 88–111. Springer, Berlin.

[MR92j:03012] Summary: "Let B be the closed term model of the λ-calculus in which terms with the same Böhm tree are identified. We investigate which partial equivalence relations (PERs) on B can be regarded as predomains or domains. When we work inside the realizability topos on B, such PERs can be regarded simply as sets in a particular model of constructive set theory.

"No well-behaved partial order has been identified for any class of PERs; but it is still possible to isolate those PERs which have 'suprema of chains' in a certain sense, and all maps between such PERs in the model preserve such suprema of chains. One can also define what it means for such a PER to have a 'bottom'; partial function spaces provide an example. For these PERs, fixed points of arbitrary endofunctions exist and are computed by the fixed point combinator y. There is also a notion of meet-closure for which all maps are stable.

"The categories of predomains are closed under the formation of total and partial function spaces, polymorphic types and convex powerdomains. (Subtyping and bounded quantification can also be modelled.) They in fact form reflective subcategories of the realizability topos; and in this set-theoretic context, these constructions are very simple to describe." For the entire collection see MR 92h:68009.

Phoa, W. (1991b). Two results on set-theoretic polymorphism. In *Category theory and computer science (Paris, 1991)*, pages 219–235. Springer, Berlin.

[MR1135342] This item will not be reviewed individually. For details of the collection in which this item appears see 92g:68005.

Schellinx, H. (1991). Isomorphisms and nonisomorphisms of graph models. *J. Symbolic Logic*, 56(1):227–249.

[MR93c:03016] Among the set-theoretical models of the untyped lambda-calculus, the graph models are particularly easy to describe. Graph models are defined as pairs $(\mathcal{P}(X), \cdot)$, where X is an infinite set and $\cdot$ is a binary operation over $\mathcal{P}(X)$ defined via an embedding $(\text{–},\text{–})$: $\mathcal{P}_f(X) \times X \to X$ ($\mathcal{P}_f(X)$ denotes the set of all finite subsets of X). There are two main classes of graph models: the $\mathcal{P}(\mathbf{N})$-structures (introduced by Plotkin and Scott), in which X is the set N of natural numbers and $(\text{–},\text{–})$ is defined via an encoding of pairs of integers and of finite subsets of integers, and the D_A-models (introduced by Engeler), in which X is defined from a set A of "atoms" by an inductive definition and $(\text{–},\text{–})$ is trivially given by the construction of X. A standard coding for $\mathcal{P}(N)$-structures has been proposed in the literature (the well-known $\mathcal{P}(\omega)$ model), while the properties of the $\mathcal{P}(N)$-structures are, in general, strongly dependent on the coding used.

The only well-known fact about the structure of graph models is that D_A-models are never isomorphic to $\mathcal{P}(N)$-structures defined through a surjective coding of pairs. In this paper the author presents a number of results about the global structure of graph models, mainly concerning isomorphisms, automorphisms and elementary equivalence (with respect to a standard first-order language L to reason about applicative structures). It is shown, for instance, that two D_A-models are isomorphic if and only if their atom sets have the same cardinality, while this is not enough for $\mathcal{P}(\mathbf{N})$-structures (where the atoms are defined as the integers which are not in the range of the coding function). To characterize isomorphism of $\mathcal{P}(N)$-structures (over a wide class of basic codings) the more general notion of frame is needed, which takes into account also the integers which can "reproduce" themselves by application (like 0 in $\mathcal{P}(\omega)$). Other interesting results in the paper concern the elementary equivalence of graph models, the L-definability of the lattice structure of a graph model, the fact that the second-order theory of the first-order definable elements of a graph model is first-order expressible in the model and the existence of uncountably many nonisomorphic and non-elementarily equivalent $\mathcal{P}(N)$ structures. A last interesting remark is that there are non-elementarily equivalent categorical lambda-models which have the same equational theory. Reviewed by M. Coppo.

Asperti, A. (1992). A categorical understanding of environment machines. *J. Funct. Programming*, 2(1):23–59.

[MR94d:68065] This paper employs Cartesian closed categories to provide semantics for a variety of lambda calculi. The author's stated goal is to show how such an algebraic approach "can provide a direct hint towards the implementation [of the language on abstract machines]". This goal is successfully accomplished for at least the comparatively simple languages and abstract machines studied in the paper.

The paper consists of three parts. In all three parts a key role is played by the concept of categorical soundness: the existence of a mapping $\mathcal{F}$ that maps each machine state to its interpretation in the categorical model (a Cartesian closed category) and is such that if the machine goes from state S to state S' then $\mathcal{F}(\mathcal{S}') = \mathcal{F}(\mathcal{S})$. The pursuit of categorical soundness both enforces and facilitates the forging of the required link between the denotational semantics provided by the Cartesian closed category and the operational semantics provided by the abstract machine.

The first part takes a new look at three known abstract machines for the λ-calculus. The first is Curien's categorical abstract machine (the CAM), which provides an implementation based on call-by-value. The second is Mauny's lazy version of the CAM employing call-by-name. The third is Krivine's machine (previously unpublished). The categorical approach to the semantics successfully explicates the relationship between the latter two machines.

The second part of the paper introduces a variant of the CAM that implements a λ-calculus with both call-by-value and call-by-names parameter passing. Again the categorical semantics guides the implementation.

The final part of the paper presents two new categorical machines for strong reduction of λ-terms (i.e., reduction up to normal form). The impact of the categorical approach is especially evident in this part of the paper. As the author notes, "since the result of the evaluation is a λ-term, and not just a closure, the categorical 'soundness' of the machine immediately yields a 'correctness' result, showing the full power of this approach".

While the author claims that the paper is "essentially self-contained", she clearly assumes that the reader has a knowledge of λ-calculus, Cartesian closed categories and such concepts as lazy evaluation. However, given those prerequisites, the paper reads very nicely. Reviewed by Eric G. Wagner.

Asperti, A. and Martini, S. (1992). Categorical models of polymorphism. *Inform. and Comput.*, 99(1):1–79.

[MR93i:03017] This paper is primarily a survey of the use of category-theoretic techniques in the modelling of the second-order lambda calculus and some of its closer relatives (such as Girard's F_ω). The category-theoretic models come in two forms, as indexed categories (PL-categories) and as internal categories. The authors define both these forms, and explore the relationship between them. The paper concludes with a discussion of various models viewed as internal categories.

The main use of the paper is probably as a tutorial. The results are presented in some detail, enough for the paper to be readable, albeit with effort, by somebody who has an acquaintance with elementary category theory, up to about the level of the definition of adjunction and the Yoneda lemma. Knowledge of indexed and internal category theory is not needed, but an acquaintance with the connection between Cartesian closed categories and the simply typed lambda calculus (as in, e.g., the book by J. Lambek and P. J. Scott [Introduction to higher order categorical logic, Cambridge Univ. Press, Cambridge, 1986; MR 88a:03032]) would be a definite help.

There are two approaches to the presentation of internal and indexed category theory, the algebraic and the logical. The logical approach is to prove results by reasoning in the internal logic of the appropriate form of category. We can say, for example, that Theorem 3.7 holds because it is the internalised form of a conventional result about adjunctions, and because the proof of that result can be formalised in lim-logic. This cuts down on the apparent complexity of internal category theory (and indexed category theory, since similar techniques are applicable there), by providing a clear link with a more conventional body of theory. The disadvantage is that one must be quite familiar not just with the results but with the proofs of conventional category theory, to the point at which one can spot the fragment of logic needed for their proof. One must also be familiar enough with the interpretation of different logical systems in categories. These requirements represent a fair degree of overhead, largely avoided in a purely algebraic approach, as followed in the present work. This enables the authors to present their proofs in a relatively self-contained fashion, but with the disadvantage that arguments tend to involve a fair amount of rather uninformative symbol-pushing.

Since the paper is likely to be chiefly of interest to those who have already seen something of the categorical modelling of polymorphism, and who wish to understand it in more detail, this review will not attempt to describe the two different approaches compared here. In broad outline, the authors show how an internal model gives rise to an indexed category, and how an indexed category can be used to give internal models. There are some novel aspects of their treatment, notably a rather simpler production of an indexed model from an internal model than has appeared heretofore. They conclude the paper with a quite clear discussion of internal categories defined from the PER model, and from retracts of a model of an extended lambda calculus. They also give a slightly less clear account of the relationship between their externalisation and that produced by Pitts.

Reviewer's remarks: Although the paper gives a good clear introduction to this material, it is perhaps not suitable as a sole reference, since there are points where

the exposition seems questionable. One example is when the authors require that the base category of a PL-category be Cartesian closed. This sacrifices a close connection with the type theory in favour of ease of one of their desired proofs. Another is in the comparison between their work and that of Pitts. Their construction is simpler than that of Pitts, but one reason for this is that Pitts obtains more from his. He gets a "full internal Cartesian closed subcategory" and not just an "internal Cartesian closed category". Reviewed by Edmund Robinson.

Berline, C. (1992). Rétractions et interprétation interne du polymorphisme: le problème de la rétraction universelle. *RAIRO Inform. Théor. Appl.*, 26(1):59–91.

[MR92m:68069] Summary: "We give a condensed and uniform presentation of all known results (some of them unpublished) concerning the existence of 'universal' retractions in interesting subclasses of the set of retractions of a Scott domain. This purely algebraic problem is linked to the modelisation of polymorphism. Berardi's solution provides, as a by-product, nontrivial models to a non-Church-Rosser extension of λ-calculus."

Cardone, F. (1992). An algebraic approach to the interpretation of recursive types. In *CAAP '92 (Rennes, 1992)*, pages 66–85. Springer, Berlin.

[MR94i:68169] Summary: "We present an algebraic framework for the interpretation of type systems including type recursion, where types are interpreted as partial equivalence relations over a Scott domain. We use the notion of iterative algebra, introduced by J. Tiuryn as a counterpart to the categorical notion of iterative algebraic theory given by C. C. Elgot. We show that a suitable collection of partial equivalence relations is closed under type constructors and forms an iterative algebra. The existence of type interpretations follows from the initiality, in the class of iterative algebras, of the algebra of regular infinite trees obtained by infinitely unfolding recursive types." For the entire collection see MR 94f:68009.

Freyd, P. J., Robinson, E. P., and Rosolini, G. (1992). Dinaturality for free. In *Applications of categories in computer science (Durham, 1991)*, pages 107–118. Cambridge Univ. Press, Cambridge.

[MR94a:68084] A category C is considered as a category of types and a family of functors $(C^{\mathrm{op}} \times C)^n \to C$ and dinatural transformations between them.

The question asked is the following one: If this model is interpreted in the internal logic of effective topos, is the internal category of modest sets an inherently parametric model for polymorphism? In a special case, the problem of dinaturality is reduced to checking commutativity of the hexagon for category PER (partial equivalence relation). Properties of internal functors are considered.

Reviewer's remark: To read this paper, one should have sufficient knowledge of works by Freyd and other authors concerning polymorphisms listed in the references. For the entire collection see MR 93d:68001. Reviewed by V. Yu. Meitus.

Gamatie, B. (1992). Intensionally stable functions. In *Logical foundations of computer science—Tver '92*, pages 163–173. Springer, Berlin.

[**MR94j:03032**] Summary: "Our principal concern in this paper is to further pave the way towards a semantical version of Milner's fully abstract model of typed λ-calculi. Relying on Milner's results that show that the function-based model of PCF is unique (up to isomorphism) and is Scott-ordered, and taking into account the fact that the underlying domains of the model cannot be distributive, we introduce SK-domains and define SI-functions which generalize, by departing from the distributivity requirement, the stable functions already introduced by Berry. We then show that SK-domains and SI-functions constitute a Cartesian closed category which can be order enriched with Scott's ordering, i.e., can be used to provide order extensional models for λ-calculus-based languages." For the entire collection see MR 94h:68009.

Honsell, F. and Ronchi Della Rocca, S. (1992). An approximation theorem for topological lambda models and the topological incompleteness of lambda calculus. *J. Comput. System Sci.*, 45(1):49–75.

[**MR93h:03014**] It is well known that a reflexive object in the Cartesian closed category of complete partial orders and Scott-continuous functions is a model of λ-calculus. Such "topological" models provide, through the interpretation function, a geometrical means of defining a λ-theory. As a means of analysing the λ-theory induced by such a model the authors derive an "approximation theorem". This theorem is used to study two topological models which induce nonsensible theories, i.e., ones in which not all unsolvable terms are equated. The notion of a contextual theory is introduced and it is shown that every λ-theory has a contextual definition. This gives an operational meaning to the notion of λ-theory. Finally it is proved that there is no consistently complete topological model of the λ-calculus which induces the contextual theory which strictly includes the nonsensible theory proposed by D. Park. Reviewed by Martin W. Bunder.

Jacobs, B. (1992). Simply typed and untyped lambda calculus revisited. In *Applications of categories in computer science (Durham, 1991)*, pages 119–142. Cambridge Univ. Press, Cambridge.

[**MR94b:03027**] It is standard practice to assume as the categorical formulation of models for simply typed λ-calculus the notion of Cartesian closed category, even when Cartesian products of types and unit types are not included in the rules of type formation. The paper under review proposes a refinement to this view, by distinguishing product types from products of contexts in which types are formed. Although the former can be missing from a specific type theory, the latter are always present. By treating contexts as indexes for collections of types and using some of the machinery of indexed category theory, they can be used to define exponent types. Technically, this is realized in the paper by the notions of a CT-structure acting as a category of contexts, and of a $\lambda 1$-category as the appropriate categorical notion of model for simply typed λ-calculus. When these notions are restricted to monoids (categories with one object), it can be shown that they subsume the case of untyped λ-calculus (typed λ-calculus with one, reflexive type). For the entire collection see MR 93d:68001. Reviewed by Felice Cardone.

Jiang, Y. (1992). Le sémantique continue du lambda calcul est incompatible avec l'existence d'un rétraction universelle. *C. R. Acad. Sci. Paris Sér. I Math.*, 314(10):779–782.

[MR93c:68061] Summary: "We prove that no Scott model of the type-free lambda calculus can have a universal retraction. This result was known up to now for $D = P\omega$ only (Ershov, 1975), and is strongly linked to the interpretation of polymorphism and also to the consistency of a certain extension of lambda calculus which does not enjoy Church-Rosser's property. In view of a recent theorem of Rothe, it is sufficient to prove that every algebraic and reflexive dcpo contains a dense chain. More generally, if there is an infinite and strictly increasing sequence of compact elements in a dcpo D, then $[[D \to D] \to D]$ contains a dense chain."

Knobel, A. (1992). Constructive λ-models. *J. Pure Appl. Algebra*, 79(3):271–279.

[MR93b:03018] Summary: "We prove in this paper that there is a partial order P such that for every (classical) extensional countable λ-model $\mathcal{M}$ there exist an object U in $\mathrm{Set}^{P^{\mathrm{op}}}$ and a bijection $f\colon U \twoheadrightarrow U^U$ such that the λ-model defined from U and f has the same equational theory as $\mathcal{M}$."

Lamarche, F. (1992). Quantitative domains and infinitary algebras. *Theoret. Comput. Sci.*, 94(1):37–62.

[MR93d:03022] The author describes a way of producing categories analogous to the conventional continuous (a la Scott) and stable (a la Berry) categories of domains as algebras for a semiring with infinitary addition (and an infinitary distributive law). This seems to yield both a real simplification over conventional treatments, and an attractive link with some more or less conventional algebra. However, some of the claims made in the abstract and the introduction seem slightly misleading. For instance, although the general construction goes through as stated for the class of semirings described in the introduction, the reader should not take this to mean that all the results in the paper are valid in this generality. Indeed, different sections place different requirements on the semiring. Moreover, the author is interested in finding categories with properties similar to those of conventional domain theory, rather than reconstructions of the exact categories, and no precise theorem relating the categories produced here to their conventional analogues is proved, despite a couple of teasing hints about conventional constructions being the analogues of constructions here "with the repetitions removed". Reviewed by Edmund Robinson.

Martini, S. (1992). Categorical models for nonextensional λ-calculi and combinatory logic. *Math. Structures Comput. Sci.*, 2(3):327–357.

[MR93m:03021] Exponents in Cartesian closed categories (CCCs) give a model of extensional λ-abstraction; that is, both the β-rule and the η-rule are valid. In the article under review one finds the notion of a weak CCC, in which one has the β-equation plus a naturality rule, and the notion of a very weak CCC, in which one has only β. A weak CCC is simply a category with finite products in which the ordinary adjunctions defining ordinary exponents are replaced by semi-adjunctions [see S. Hayashi, Theoret. Comput. Sci. 41 (1985), no. 1, 95–104; MR 87i:18005].

These semi-adjunctions are adjunctions in an appropriate 2-category of categories, semi-functors and semi-natural transformations [see R. Hoofman, Math. Structures Comput. Sci. 3 (1993), no. 1, 93–128]. An interesting question is whether there is a comparable 2-category in which the adjunctions describe the very weak exponents. There is no answer in the paper. The author does describe "quasifunctors", but these do not do the job. The paper contains some examples of (very) weak CCCs, and gives the relation between weak CCCs and nonextensional λ-calculus and between very weak CCCs and combinatory logic. The nonextensional approach is extended to the second order λ-calculus. There is not much which is new. Reviewed by Bart Jacobs.

Pavlović, D. (1992). On the structure of paradoxes. *Arch. Math. Logic*, 31(6):397–406.
[MR93i:03021] In the spirit of F. W. Lawvere [in Category theory, homology theory and their applications, II (Seattle, WA, 1968), 134–145, Lecture Notes in Math., 92, Springer, Berlin, 1969; MR 39 #4075], the author studies paradoxes in the setting of Cartesian closed categories. A notion of "paradoxical structure" on an object Ω is described in such categories; it yields fixed points for all endomorphisms $\Omega \to \Omega$. In terms of such paradoxical structures, Russell's paradox, Cantor's diagonalization and Gödel's incompleteness theorem are redescribed.

J. C. Reynolds [in Semantics of data types (Valbonne, 1984), 145–156, Lecture Notes in Comput. Sci., 173, Springer, Berlin, 1984; MR 86e:03016] showed that the second-order polymorphic lambda calculus has no set-theoretic models, by establishing that for each inhabited type B there is a type A together with an isomorphism $(A \to B) \to B \cong A$. Reynolds suggested that such fixed points exist for all type functors involving only exponentiation, $\to$. It is shown that by assuming this to be true, one can construct a paradoxical structure and thus a paradox. Hence what was suggested does not hold.

The reviewer fails to understand Gödel's incompleteness theorem in this setting. Reviewed by Bart Jacobs.

Phoa, W. (1992). Building domains from graph models. *Math. Structures Comput. Sci.*, 2(3):277–299.
[MR93j:68125] In "synthetic domain theory" one does domain theory within constructive set theory so that domains are just sets, and functions between domains are automatically continuous. In particular, such a set-theoretic setting can be provided by a realizability topos which is constructed from a partial combinatory algebra. In the paper under review, the author concentrates on the realizability topos arising from the recursively enumerable graph model. Various subcategories (given by replete objects, link-ordered objects or uniformly complete modest sets) are studied with respect to closure under lift (which gives a notion of partial map) and under limits (which gives the possibility to model polymorphism, datatypes and power constructions).

Superficial reading of the paper gives a nice idea of the "synthetic domain theory" project. A detailed study, however, requires more background knowledge of the project and access to the author's (as yet) unpublished work. Reviewed by Bart Jacobs.

Ricci, G. (1992). Universal eigenvalue equations. *Pure Math. Appl. Ser. B*, 3(2-4):231–288 (1993).
[MR95a:08002]

From the summary: "A part of nonlinear mathematics on vector spaces is made universal here, as was done for the linear case by means of universal algebra. In order to bridge from linearity to nonlinearity we consider eigenvalue equations (which have an intermediate nature). The main facts we universalize about such equations are: the eigenspaces of a matrix form a complete lattice; null eigenvalues characterize singular matrices; eigenvalues and eigenvectors define the general integral of certain difference (and differential) equations; the Cayley-Hamilton theorem holds; each matrix has a (single) geometric dimension and a null determinant implies a matrix singularity."

Set-ary operations are used in the paper instead of n-ary operations, and algebras are treated as families of operations over an index set. The universalization of "eigenvector" splits into two concepts, eigenspaces and innerspaces, whose definitions are roughly as follows. For an algebra α on A, a set X, and a "matrix" $M\colon X \to A$, if there is a unique endomorphism h of α such that $M = hg$ for some $g\colon X \to A$, then h is called the endomorphism associated with M. An eigenmorphism of such an h is an endomorphism m of a quotient of α such that $ph = mp$, where p is the partition function associated with the quotient. The quotient is an eigenspace of M. Also, an endomorphism of a subalgebra δ of α is an innermorphism for h if it is a restriction of h. Such subalgebras are the innerspaces of M.

A non-vector space example is given to illustrate the universal theory using Boolean algebras with the "matrices" as binary clocks from switching theory. Reviewed by S. Comer.

Weibel, T. (1992). Extension of combinatory logic to a theory of combinatory representation. *Theoret. Comput. Sci.*, 97(1):157–173.

[MR93g:03014] Summary: "In theoretical computer science and mathematics, the models of combinatory logic are of significance in various ways. In particular within the discipline of algebra, varieties are represented in the graph models of Engeler using some set of equations. In this paper, such representations of algebraic classes are presented entirely within the theory of combinatory logic. Under formal conditions this is achieved in a uniform manner by equations composed only of a few basic combinators. Some conditions are first-order logic formulae and yield the semi-universal combinatory models that provide for varieties using equations. The other, second-order logic axioms are added to the former conditions, yielding the unviersal combinatory models. In those models, varieties may also be presented as solutions of the equations. These results are extended to universal and algorithmic classes."

Crole, R. L. (1993). *Categories for types.* Cambridge University Press, Cambridge.

[MR95h:18007] This book is a well-organized introduction to categorical models of λ-calculus—simply typed, polymorphic and higher-order polymorphic. It starts with the basic concepts of domain theory and category theory and proceeds to algebraic theories, as a good introduction to the concepts and notation used in later chapters. Those algebraic theories are nothing more than a many-sorted version of Lawvere's algebraic theories, without saying so (not even in "pointers to the literature"). Next comes simply typed λ-calculus with pairing, here called λx-theory, in a treatment somewhat similar to that of A. Asperti and G. Longo [Categories, types, and struc-

tures, MIT Press, Cambridge, MA, 1991; MR 93i:18001]. The book of J. Lambek and P. J. Scott [Introduction to higher order categorical logic, Cambridge Univ. Press, Cambridge, 1986; MR 88a:03032] is not even included in the bibliography(!), but this seems to be an unintentional mistake, however quite painful. A discussion of second-order polymorphisms, called $2\lambda x$-theory, follows, including Per-model and model in Scott-domains of T. Coquand, C. A. Gunter and G. Winskel [Inform. and Comput. 81 (1989), no. 2, 123–167; MR 91d:68079]. The final, briefly addressed topic is that of higher-order polymorphism. The individual topics are treated in a uniform, balanced style from introduction, syntax, to categorical models and categorical-type theory correspondence. The book is a useful introduction to a dynamic topic of interest to computer scientists and category theorists. Reviewed by Jiri Adamek.

Di Gianantonio, P. and Honsell, F. (1993). An abstract notion of application. In *Typed lambda calculi and applications (Utrecht, 1993)*, pages 124–138. Springer, Berlin.

[MR94c:03020] Summary: "Many concrete notions of function application, suitable for interpreting typed lambda calculi with recursive types, have been introduced in the literature. These arise in different fields such as set theory, multiset theory, type theory and functor theory and are apparently unrelated. In this paper we introduce the general concept of applicative exponential structure and show that it subsumes all these notions. Our approach is based on a generalization of the notion of intersection type. We construe all these structures in a finitary way, so as to be able to utilize uniformly a general form of type assignment system for defining the interpretation function. Applicative exponential structures are just combinatory algebras, in general. Our approach suggests a wide variety of entirely new concrete notions of function application; e.g., in connection with Boolean sets. Applicative exponential structures can be used for modeling various forms of nondeterministic operators." For the entire collection see MR 94b:68025.

Freyd, P. (1993). Structural polymorphism. *Theoret. Comput. Sci.*, 115(1):107–129. 4th Summer Conference on Category Theory and Computer Science (Paris, 1991).

[MR94k:68126] From the introduction: "In a previous attempt to formalize parametricity in polymorphism, a subject called 'functorial polymorphism' was introduced by Bainbridge, Freyd, Scedrov and Scott. The basic idea was to interpret variable types as functors and terms as 'dinatural transformations'. This semantics successfully suggested further equations for higher-order lambda calculus but did not capture all that seems to be involved in the notion of parametricity. Several attempts have been made to strengthen the notion of dinatural transformation. The point of departure in this work is to generalize the notion of functor. We arrive at a new subject, called 'the theory of structors'. We show (i) that both covariant and contravariant functors may be construed as structors, (ii) that dinatural transformations between bifunctors may be construed as transformations between structors, and (iii) that the exponentiation of structors is naturally defined. We use these facts to interpret polymorphism in a sufficiently parametric way to allow intrinsic definitions of the standard basic data types." For the entire collection see MR 94b:68007. Reviewed by Ana Pasztor.

Hoofman, R. (1993). The theory of semi-functors. *Math. Structures Comput. Sci.*, 3(1):93–128.

[**MR94e:18001**] Summary: "The notion of semi-functor was introduced by Hayashi in order to make possible a category-theoretical characterization of models of the nonextensional typed lambda calculus. Motivated by the further use of semi-functors independently by Martini, Jacobs, and us, we consider the general theory of semi-functors in this paper. It turns out that the notion of semi-natural transformation plays an important part in this theory, and that various categorical notions involving semi-functors can be viewed as 2-categorical notions in the 2-category of categories, semi-functors and semi-natural transformations. In particular, we find that the notion of normal semi-adjunction as defined by Hayashi is the canonical generalization of the notion of adjunction to the world of semi-functors. Further topics covered in this paper are the relation between semi-functors and splittings, the Karoubi envelope construction, semi-comonads, and a semi-adjoint functor theorem."

Jacobs, B. (1993). Semantics of λ-I and of other substructure lambda calculi. In *Typed lambda calculi and applications (Utrecht, 1993)*, pages 195–208. Springer, Berlin.

[**MR94d:03023**] Summary: "The ordinary untyped λ-calculus will be denoted here by λK. Church originally introduced the λI-calculus, which can be understood as the λK-calculus without weakening: one cannot throw away variables. Similarly, there is an affine calculus λA without contraction: there, one cannot duplicate variables. There is also a linear calculus λL in which one has neither weakening nor contraction. In λL, variables occur precisely once.

"We give a systematic description of the semantics of these four calculi. It starts with two sorts of domain-theoretic models: graph models and filter models (of intersection types) are constructed for each of these calculi. Later on, we describe an appropriate categorical way to capture such structures in terms of monoidal categories (with diagonals or projections)." For the entire collection see MR 94b:68025.

Jung, A. and Tiuryn, J. (1993). A new characterization of lambda definability. In *Typed lambda calculi and applications (Utrecht, 1993)*, pages 245–257. Springer, Berlin.

[**MR94c:03023**] Summary: "We give a new characterization of lambda definability in Henkin models using logical relations defined over ordered sets with varying arity. The advantage of this over earlier approaches by Plotkin and Statman is its simplicity and universality. Decidability of lambda definability for hereditarily finite Henkin models remains an open problem. But if the variable set allowed in terms is also restricted to be finite then our techniques lead to a decision procedure." For the entire collection see MR 94b:68025.

Lambek, J. (1993). Least fixpoints of endofunctors of Cartesian closed categories. *Math. Structures Comput. Sci.*, 3(2):229–257.

[**MR94h:18001**] For Cartesian closed categories satisfying a weak completeness hypothesis a fixed-point theorem is proved: every endofunctor which is a finite coproduct of "definable" functors (a natural concept of a kind of polynomial endofunctor) has a least fixed point. The result is applied to the category of partial equivalence relations PER.

Mitchell, J. C. and Scedrov, A. (1993). Notes on sconing and relators. In *Computer science logic (San Miniato, 1992)*, pages 352–378. Springer, Berlin.
[**MR94h:68133**] Summary: "This paper describes a semantics of typed lambda calculi based on relations. The main mathematical tool is a category-theoretic method of sconing, also called glueing or Freyd covers. Its correspondence to logical relations is also examined." For the entire collection see MR 94f:68011.

Nielson, F. and Nielson, H. R. (1993). Layered predicates. In *Semantics: foundations and applications (Beekbergen, 1992)*, pages 425–456. Springer, Berlin.
[**MR94j:03035**] Summary: "We review the concept of logical relations and how they interact with structural induction, giving examples of their use; of particular interest is the combination with the PER-idea (partial equivalence relations). This is then generalized to Kripke-logical relations; the major application is to show that in combination with the PER-idea this solves the problem of establishing a substitution property in a manner conducive to structural induction. Finally we introduce the concept of Kripke-layered predicates; this allows a modular definition of predicates and supports a methodology of 'proofs in stages' where each stage focuses on only one aspect and thus is more manageable. All of these techniques have been tested and refined in 'realistic applications' that have been documented elsewhere." For the entire collection see MR 94h:68011.

Pigozzi, D. and Salibra, A. (1993a). An introduction to lambda abstraction algebras. In *Proceedings of the IX Latin American Symposium on Mathematical Logic, Part 1 (Bahía Blanca, 1992)*, pages 93–112, Bahía Blanca. Univ. Nac. del Sur.
[**MR96e:03015**] A lambda abstraction algebra (LAA) is an algebraic, equational version of the untyped lambda calculus, which forms a variety in the sense of universal algebra. From a lambda calculus point of view, the "new ideas" in an LAA are that the set of variables I ("the dimension") may be finite and the set of variables free in a term a ("the dimension of a") may be infinite. An LAA is "locally finite" if it has infinite dimension but the dimensions of all its terms are finite. Roughly an LAA is "functional" if every function can be represented by lambda abstraction.

The paper includes, besides much work on substitution in LAAs, the result that every locally finite LAA is isomorphic to a functional LAA. Lambda calculus is of course both locally finite and functional. All known models of lambda calculus are LAAs; it would have been interesting to have some nontrivial examples of LAAs that are not. There is only a suggestion of these in the final section of the paper. For the entire collection see MR 96a:03006. Reviewed by Martin W. Bunder.

Pigozzi, D. and Salibra, A. (1993b). A representation theorem for lambda abstraction algebras. In *Mathematical foundations of computer science 1993 (Gdańsk, 1993)*, pages 629–639. Springer, Berlin.

[MR94k:03017] Summary: "The concept of a lambda abstraction algebra (LAA) is designed to algebraize the untyped lambda calculus in the same way cylindric and polyadic algebras algebraize the first-order predicate logic. Like cylindric and polyadic algebras, LAAs can be defined by true identities, and thus they form a variety in the sense of universal algebra. They provide a distinctly algebraic alternative to the highly combinatorial lambda calculus. A characteristic feature of LAAs is the algebraic reformulation of (β)-conversion as the definition of abstract substitution. The equational axioms of LAAs reflect (α)-conversion and Curry's recursive axiomatization of substitution in the lambda calculus. Functional LAAs arise from environment models or lambda models, the natural models of the lambda calculus. The main result of this paper is a stronger version of the functional representation theorem for locally finite LAAs, the algebraic analogue of the completeness theorem of lambda calculus." For the entire collection see MR 94i:68008.

Reynolds, J. C. and Plotkin, G. D. (1993). On functors expressible in the polymorphic typed lambda calculus. *Inform. and Comput.*, 105(1):1–29.

[MR94h:03036] In 1984, Reynolds showed that there were no models of the polymorphic lambda calculus in which types were interpreted as arbitrary sets and the function space was interpreted as the full set of functions. Reynolds' proof, however, was somewhat obscure, and this fact stimulated a number of attempts to clarify and generalise it, as well as to prove analogues in other settings. The current paper is central to this body of work, in that it presents a clear abstract reformulation of Reynolds' original argument. This refinement leads to a substantial increase in the power of the results, most notably allowing their application to certain concrete categories, such as categories of cpo's.

The paper is concerned with applied forms of (extensions of) the second-order lambda calculus. This means that as well as types in the pure calculus, we may have additional constant types and type constructors, together with associated operations. Given a model of such a calculus, let $\mathcal{K}_{\prime}$ be the carrier for the collection of (closed) types. Note that not every element of $\mathcal{K}_{\prime}$ need be denotable by a type expression. Note also that $\mathcal{K}_{\prime}$ forms the set of objects of a Cartesian-closed category $\mathcal{K}$. The authors are concerned with functors $T\colon \mathcal{K} \to \mathcal{K}$ which are definable by expressions in the calculus with constants from $\mathcal{K}_{\prime}$. (Hence their slightly odd definition of what constitutes a model, which amounts to an environment model at the level of types, but a Cartesian-closed category model at the level of values.)

The authors show that any functor which is so expressible has a weak initial algebra, and that if $\mathcal{K}$ has enough equalisers then it has an initial algebra. This gives a type A such that $T(A) \cong A$. By choosing a suitable T (e.g., $T(X) = (X \to c) \to c$) it is possible to show the nonexistence of certain forms of models. Reviewed by Edmund Robinson.

Di Gianantonio, P., Honsell, F., Liani, S., and Plotkin, G. D. (1994). Countable non-determinism and uncountable limits. In *CONCUR '94: concurrency theory (Uppsala, 1994)*, pages 130–145. Springer, Berlin.

[**MR96b:68107**] Summary: "In this paper we address the problem of solving recursive domain equations using uncountable limits of domains. These arise for instance, when dealing with the ω_1-continuous function-space constructor and are used in the (generative) denotational semantics of programming languages which feature nondeterministic fair constructs implemented with unbounded choice. It is well known that such features cannot be modeled using just Scott-continuous functions; they can be modeled conveniently, instead, using functions which satisfy the weaker continuity property of ω_1-continuity. Surprisingly, the category of cpo's and ω_1-continuous embeddings is not ω_0-cocomplete. Hence the standard technique of Adamek and Koubek for solving reflexive domain equations involving these constructs fails. This phenomenon was not noticed before in the literature. We discuss two alternative methods that can be used to solve the required domain equations. The first one is an application of the method of Adamek and Koubek to a different category. The second one, instead, is a genuine extension of their method and amounts to considering cones instead of limits. We put both methods on solid categorical grounds and discuss some applications. Finally we utilise the second method in order to give a model for the untyped λ-calculus whose theory is precisely the $\lambda\beta\eta$ theory. Thus we show that ω_1-lambda models are complete for the $\lambda\beta\eta$-calculus." For the entire collection see MR 95j:68006.

Hasegawa, R. (1994a). Categorical data types in parametric polymorphism. *Math. Structures Comput. Sci.*, 4(1):71–109.

[**MR95c:68162**] Summary: "The categorical data types in models of second-order lambda calculus are studied. We prove that Reynolds parametricity is a necessary and sufficient condition for the categorical data types to satisfy the universal properties." Reviewed by Zhenyu Qian.

Hasegawa, R. (1994b). Relational limits in general polymorphism. *Publ. Res. Inst. Math. Sci.*, 30(4):535–576.

[**MR96j:68113**] Summary: "Parametric models of polymorphic lambda calculus have the structure of enriched categories with cotensors and ends in some generalized sense, and thus have many categorical data types induced by them. The ω-order minimum model is a parametric model."

Phoa, W. (1994). From term models to domains. *Inform. and Comput.*, 109(1-2):211–255. International Conference on Theoretical Aspects of Computer Software (TACS '91) (Sendai, 1991).

[**MR95f:68148**] Summary: "Let B be the closed term model of the λ-calculus in which terms with the same Böhm tree are identified. We investigate which partial equivalence relations (PERs) on B can be regarded as predomains or domains. Inside the realizability topos on B, such PERs can be regarded simply as sets in a particular model of constructive set theory. No well-behaved partial order has been identified for any class of PERs; but it is still possible to isolate those PERs which have 'suprema of chains' in a certain sense, and all maps between such PERs in the model preserve such suprema of chains. One can also define what it means for such a PER to have a 'bottom'; partial function spaces provide an example. For these PERs, fixed points of arbitrary endofunctions exist and are computed by the fixed point combinator y. The categories of predomains are closed under the formation of total and partial function spaces, polymorphic types, and convex powerdomains. They in fact form reflective subcategories of the realizability topos; and in this set-theoretic context, these constructions are very simple to describe. We illustrate the theory by discussing an interpretation of PCF, and proving a computational adequacy theorem. None of the usual counterexamples to full abstraction are applicable to our model." For the entire collection see MR 94j:68004. Reviewed by E. Fehr.

Ricci, G. (1994). Errata to: "Universal eigenvalue equations" [Pure Math. Appl. Ser. B **3** (1992), no. 2-4, 231–288; MR 95a:08002]. *Pure Math. Appl.*, 5(2):241–243.

[MR1327453] There will be no review of this item.

Ritter, E. (1994). Categorical abstract machines for higher-order typed λ-calculi. *Theoret. Comput. Sci.*, 136(1):125–162. Selected papers of the Meeting on the Mathematical Foundations of Programming Semantics (MFPS '92), Part II (Oxford, 1992).

[MR96d:68140] The author proposes particular indexed categories (split D-categories) as an appropriate categorical framework for abstract machines, not just for the simply typed λ-calculus, but also for the higher-order typed λ-calculi. The author concentrates on the calculus of constructions. Split D-categories permit the separation of terms and environments because terms correspond to morphisms in the fibres and environments to morphisms in the base category. This automatically leads to different combinators for substitution (which is modelled by a reindexing functor) and a function application (which corresponds to composition in the fibre).

The D-categories also model Cartesian products and contexts differently: the former are handled by left adjoints to weakening and the latter by a right adjoint to the terminal object functor. There are two groups of abstract machines based on these D-categories. The first group uses eager environments that contain only canonical combinators, whereas in the second group, an environment may contain arbitrary expressions.

Further design decisions yield machines that, if restricted to the combinators corresponding to the simply typed λ-calculus, can be transformed directly into the categorical abstract machine (eager case) or Krivine's machine (lazy case). These abstract machines have a modular structure. Both types of machines can be easily implemented in programming languages. The author gives some examples. For the entire collection see MR 95h:68002. Reviewed by V. Yu. Meitus.

Sergeraert, F. (1994). The computability problem in algebraic topology. *Adv. Math.*, 104(1):1–29.

[MR95c:55017] The problem of computability in algebraic topology is of clear theoretical and practical interest. Practical approaches have included Adams, Eilenberg-Moore and similar spectral sequences, Quillen's homotopical algebra, simplicial sets, and the Sullivan program. On the negative side, any approach is limited by Novikov's result that the word problem for finitely presented groups is not effectively solvable (by an algorithm in the sense of Church and Turing).

In the present paper the author defines a "universal effective homology theory" and argues that it is effectively (machine) computable for simply connected spaces. Although many details are unfortunately omitted, the argument appears convincing and offers the promise of future software for computing invariants of topological interest. Unfortunately the author offers little help in this direction, that is, for both actually computing effective homology and for deriving other invariants from it. If such results are in fact forthcoming, then the author has made a promising start on foundations in the present paper. Reviewed by Harold Hastings.

Stoltenberg-Hansen, V., Lindström, I., and Griffor, E. R. (1994). *Mathematical theory of domains.* Cambridge University Press, Cambridge. **[MR96d:68142]** The book under review is an introduction to the mathematical theory of Scott-Ershov domains. The initial and fundamental contributions to the theory are due to D. S. Scott and Y. L. Ershov, in that order. A domain is a mathematical structure modelling the notion of computation and approximation. As an initial motivation the authors consider Kleene's fixed point theorem for continuous functionals. This theorem is the basis for a denotational semantics of computer programs. An abstract formulation of the fixed point theorem can be given in the category cpo of complete partial orders and continuous mappings (actually, ω-complete partial orders suffice). These structures may be considered as a minimal model for a computation theory, but the important concept of finiteness is ignored. An abstract analogue of "finite" is provided by the notion of compact element of a cpo. An algebraic cpo is defined as a cpo in which every element is the supremum of its compact approximations. With this definition one obtains an abstract version of the fact that the value of a computable functional F applied to a function f depends only on finitely many values of f. Algebraic cpo's are adequate for analysing many aspects of computability. However, they have the drawback of not forming a Cartesian closed category. Therefore one considers the smaller category of domains and continuous mappings. A domain is an algebraic cpo in which every consistent set (i.e. a set with an upper bound) has a supremum. After proving that the category of domains is Cartesian closed the authors treat the important problem of constructing domains with prescribed properties expressed in the form of "domain equations". In 1972 Scott constructed a model of λ-calculus by solving the domain equation $D = [D \to D]$, i.e. by constructing a domain isomorphic to its own function space. Beginning with Scott's construction, domain equations are nowadays considered a basic tool in the semantics of computer programs. It turns out that rather than solving domain equations directly, it is often more manageable to work with representations of domains rather than with the domains themselves. The starting observation is that a domain can be represented by the subposet of its compact elements. These posets are "conditionally complete upper semi-lattices" (CUSLs), and they form a category equivalent to the category of domains (with CUSL-embeddings and domain-embeddings, respectively). The proof that CUSL is Cartesian closed is rather involved since the construction of the function space requires taking a suitable quotient to identify certain elements. To avoid this difficulty it is better to work with preordered structures rather than ordered structures, obtaining the category pre-CUSL. A pre-CUSL can yield a rather concrete representation of a domain, and it can be shown to be essentially equivalent to Scott's notion of "information system". The class of all pre-CUSLs (with a selected bottom element) ordered by pre-CUSL inclusions is a large cpo, and therefore the fixed point theorem can be applied. Since solving domain equations corresponds to finding a fixed point of certain continuous functions on this large cpo, we obtain in this way the solvability of domain equations. A topological approach to the theory of domains is also given. Scott's domains admit the so-called "Scott topology", and the order can be recovered from the topology. A topological representation of a domain can be obtained through the notion of "formal space". The solutions of domain equations and the theory of domain representations conclude the first part of the book. The second part deals with more special topics, such as ultrametric spaces, effective domains, power domains and models of λ-calculus. It is a pity that the authors do not discuss in depth the relationship between domain theory and type theory in the λ-calculus. In conclusion, this is a valuable introductory book which will be useful as a textbook for students in computer science or logic, or as a general reference. Reviewed by Alessandro Berarducci.

Alimohamed, M. (1995). A characterization of lambda definability in categorical models of implicit polymorphism. *Theoret. Comput. Sci.*, 146(1-2):5–23.

[MR96h:03031] Summary: "Lambda definability is characterized in categorical models of simply typed lambda calculus with type variables. A category-theoretic framework known as glueing or sconing is used to extend the Jung-Tiuryn characterization of lambda definability [A. Jung and J. Tiuryn, in Typed lambda calculi and applications (Utrecht, 1993), 245–257, Lecture Notes in Comput. Sci., 664, Springer, Berlin, 1993; MR 94c:03023] first to ccc models, and then to categorical models of the calculus with type variables." Reviewed by Chantal Berline.

Asperti, A. (1995). Linear logic, comonads and optimal reductions. *Fund. Inform.*, 22(1-2):3–22.

[MR96c:03027] The author presents work on optimal graph reduction techniques for the λ-calculus in a category-theoretical setting, relating the control operators of G. Gonthier, M. Abadi and J.-J. Levy [in Proceedings of the 19th Annual ACM Symposium on Principles of Programming Languages (Albequerque, NM, 1992), 15–26, ACM, New York, 1992] to the natural transformations $\epsilon\colon ! \to 1$ and $\delta\colon ! \to !!$ associated with linear logic's comonad ! and explaining their rewriting rules as a "local implementation" of naturality laws. Reviewed by Harold Schellinx.

Di Gianantonio, P., Honsell, F., and Plotkin, G. (1995). Uncountable limits and the lambda calculus. *Nordic J. Comput.*, 2(2):126–145.

[MR96j:68108] In 1982 Plotkin proposed a variant of the category ω_0-cpos to model fairness and uncountable nondeterminism. Contrary to a claim in Plotkin's original paper, in this category one cannot use the technique for solving recursive domain equations based on colimits of ω-chains of embeddings.

This paper corrects the mistake and gives two methods for solving recursive domain equations. Both methods are applicable to this category, and are general enough for the kind of languages one wants to model in it. One of the methods is also used to prove that in this category there is a reflexive domain whose theory is exactly $\lambda\beta\eta$. Reviewed by Eugenio Moggi.

Hofmann, K. H. and Mislove, M. W. (1995). All compact Hausdorff lambda models are degenerate. *Fund. Inform.*, 22(1-2):23–52.

[MR96g:03020] A model of the untyped lambda calculus can be described as an object X in some Cartesian closed category for which the exponent X^X is a retract of X. This reflexivity is needed because an untyped lambda term can be both a function and an argument. Dana Scott constructed the first such model D_∞ in a category of algebraic lattices. Since then, lambda models have been described in various other categories of ordered sets. An open question is whether there are lambda models in Cartesian closed categories of topological spaces. In the paper this question is answered negatively for compact models in the category of Hausdorff k-spaces and continuous functions. The outcome follows from a degeneracy result for compact topological semigroups. Reviewed by Bart Jacobs.

Mori, A. and Matsumoto, Y. (1995). Deriving categorical combinators from adjoint functors. *Trans. Inform. Process. Soc. Japan*, 36(10):2422–2432.

[MR96h:03038] Summary: "Categorical combinators can be seen as variable-free translations of lambda-calculus and have been used in the implementation of functional languages based on their categorical interpretations. We show that if we define categorical structures by adjoint functors we can derive categorical combinators and their equations uniformly from the basic concepts of category theory. Categorical combinators appear as the units and counits of adjoint functors, and their equations come directly from the definitions of categories, functors and natural transformations, and from the triangular identities for adjoint functors. We first show the derivation for Cartesian closed categories and illustrate the construction of free categories in terms of categorical combinators. We next discuss the derivation of nonextensional categorical combinators from semi-adjoint functors. Finally we consider general limit objects and recursive objects with regard to the duality between left and right adjoint functors."

Pigozzi, D. and Salibra, A. (1995a). The abstract variable-binding calculus. *Studia Logica*, 55(1):129–179.

[MR97b:03085]

This paper has grown out of successive attempts of the authors to develop a general algebraic theory of variable-binding and variable-substitution. This theory, called the VB-calculus, is modeled after the λ-calculus: just as in the λ-calculus, all axioms of the VB-calculus are in the form of equations; nevertheless, the calculus is not an equational theory because of restricted substitution for variables. To overcome the problem, the authors follow the example of the first-order predicate logic, where a similar problem was solved by introducing operation symbols for substitutions, and led to the theory of polyadic algebras. So the authors introduce (a variety of) generalized polyadic algebras which becomes their principal tool. The notion of local finiteness is also borrowed from the classical theory.

The first section of the paper begins with an exposition of the authors' position and includes the description of the syntax and semantics of a simplified version of the VB-calculus. Also indicated are connections with the general theory of algebras with quantifiers [A. Salibra, in Algebraic logic (Budapest, 1988), 573–620, Colloq. Math. Soc. Janos Bolyai, 54, North-Holland, Amsterdam, 1991; MR 93h:03084], and the so-called abstract algebraic logic actively developed during the last decade by H. Andreka, I. Nemeti and I. Sain [in Algebraic methodology and software technology (AMAST '93) (Enschede, 1993), 5–26, Springer, Berlin, 1994], and (in a somewhat different form) by W. J. Blok and the first author [Mem. Amer. Math. Soc. 77 (1989), no. 396, vi+78 pp.; MR 90d:03140]. In further sections the authors (1) show that the λ-calculus, classical quantification theory, various formal systems with generalized quantifiers, and a kind of abstract theory of integration can all be be formalized as VB-theories, (2) present in detail the general VB-calculus, and (3) discuss polyadic VB-algebras, the pure equational form of the calculus. The central result is the Functional Representation Theorem, which states that every locally finite polyadic VB-algebra of infinite dimension is isomorphic to a functional VB-algebra. Of course, this is a direct generalization of the Halmos functional representation theorem for polyadic Boolean algebras [P. R. Halmos, Fund. Math. 43 (1956), 255–325; MR 19, 112d]. As a corollary, a strong representation theorem for the VB-calculus is obtained. It specializes to the above-mentioned particular VB-theories. Reviewed by Janis Cirulis.

Pigozzi, D. and Salibra, A. (1995b). Lambda abstraction algebras: representation theorems. *Theoret. Comput. Sci.*, 140(1):5–52. Selected papers of AMAST '93 (Enschede, 1993).

[MR96b:68123] Summary: "Lambda abstraction algebras (LAAs) are designed to algebraize the untyped lambda calculus in the same way cylindric and polyadic algebras algebraize the first-order predicate logic. Like combinatory algebras they can be defined by true identities and thus form a variety in the sense of universal algebra, but they differ from combinatory algebras in several important respects. The most natural LAAs are obtained by coordinatizing environment models of the lambda calculus. This gives rise to two classes of LAAs of functions of finite arity: functional LAAs (FLA) and point-relativized functional LAAs (RFA). It is shown that RFA is a variety and is the smallest variety including FLA.

"Dimension-complemented LAAs constitute the widest class of LAAs that can be represented as an algebra of functions and are known to have a natural intrinsic characterization. We prove that every dimension-complemented LAA is isomorphic to a member of RFA. This is the crucial step in showing that RFA is a variety." For the entire collection see MR 95j:68023. Reviewed by Martin W. Bunder.

Révész, G. E. (1995). Categorical combinators with explicit products. *Fund. Inform.*, 22(1-2):153–166.

[MR96g:03022] Adding surjective pairing to the untyped lambda calculus has the undesirable effect that the confluence (or Church-Rosser) property no longer holds.

It is not known whether adding surjective pairing to combinatory logic has the same effect. To tackle this problem various (weaker) subsystems of combinatory logic with surjective pairing are studied in the literature—in which it is hoped that confluence can be proved. In the paper under review two such subsystems (called "weak categorical combinatory logic" CCL and CCL $\beta\eta$P by Curien) are related via a finite set of complicated axioms involving encoded abstractions. Reviewed by Bart Jacobs.

Zalamea, F. (1995). Recursion in categories. *Rev. Colombiana Mat.*, 29(2):127–144.

[MR97i:18003] Summary: "The article splits in two parts. First, a survey of work done around recursion theory, in several categorical settings, is presented. Natural number objects in topoi, particularly in the recursive and effective topoi, Freyd's allegories and his reconstructions of the free and effective topoi, the representation of numerical functions in Cartesian categories, are examined. Second, an axiomatic study of categorical versions of enumeration and parametrization (s-m-n) is presented in the general framework of partial Cartesian categories, leading to a residual characterization, previously unnoticed in classical work."

Drakengren, T. (1996). Uniqueness of Scott's reflexive domain in $P\omega$. *Theoret. Comput. Sci.*, 155(1):267–276.

[MR97c:03046] Summary: "The extensional model in $P\omega$ for the pure λ-calculus, found by D. S. Scott in 1976, does not seem to have been much used, although it is more concrete than the D_∞ models found by Scott in 1969. The main results presented in this paper are that the model in $P\omega$ is actually isomorphic to one of the simplest D_∞ models, and that this domain is unique, in the sense that it is the only nontrivial solution to the retract equation $d = d \circ\!\!\rightarrow d$, using the specific codings of the $P\omega$ model." Reviewed by Guo-Qiang Zhang.

Hofmann, K. H. and Mislove, M. W. (1996). Principles underlying the degeneracy of topological models of the untyped lambda calculus. In *Semigroup theory and its applications (New Orleans, LA, 1994)*, pages 123–155. Cambridge Univ. Press, Cambridge.

[MR99c:03014] A reflexive object X in a Cartesian closed category C is an object such that X^X is a retract of X, so providing a natural model of the untyped lambda calculus. Reflexive objects are, however, degenerate in many well-known categories, such as the category of sets and functions, the category of posets and monotone maps, and the category of complete ultrametric spaces and contractions. A Cartesian closed category possessing nondegenerate reflexive objects was first discovered by Dana Scott: ALG, the category of algebraic lattices and Scott continuous maps.

This paper considers the question of which Cartesian closed categories of Hausdorff spaces and continuous maps have a nondegenerate reflexive object. It extends the authors' earlier results [Fund. Inform. 22 (1995), no. 1-2, 23–52; MR 96g:03020] showing that in the category of Hausdorff k-spaces and continuous maps all compact reflexive objects are degenerate, and provides five "principles" that lie at the core of eliminating topological spaces as nondegenerate lambda models. These are the cardinality principle, based on Cantor's lemma, the degeneracy principle, which eliminates models where the fixed-point operator Y is injective, the compactness principle, which eliminates compact spaces, the retraction principle, showing that any retract of a

model must be a model, and the normality principle, which eliminates normal spaces containing an arc.

The paper includes an extensive appendix covering background material on Cartesian closed categories. For the entire collection see MR 97g:20002. Reviewed by Vladimiro Sassone.

Hoofman, R. (1996). Comparing models of the intensional typed λ-calculus. *Theoret. Comput. Sci.*, 166(1-2):83–99.

[**MR97m:03029**] This paper compares "powerset models" of the simply typed λ-calculus, which are those models such that types are interpreted by powersets and the type constructor "$\rightarrow$" by a semi-exponent "$\Rightarrow$" in the category POW of powersets and continuous functions.

The powerset $PA \Rightarrow PB$ is an approximation of the continuous function space $PA \rightarrow_c PB$ in the sense that the latter is a continuous retract of the former. The most relevant example is given by $PA \Rightarrow_m PB =_{\mathrm{def}} P\{(X,b)\colon X \subseteq A, X$ finite, $b \in B\}$, which clearly allows us to encode continuous functions by their traces.

It is shown in the paper that $\Rightarrow_m$ is the best approximation of $\rightarrow_c$ in POW, in the sense that it can be embedded in any other semi-exponent (which belongs to a class $\mathcal{E}$ which also contains all the other usual semi-exponents). As a consequence the theory induced by $\Rightarrow_m$ is maximal among the theories induced by elements of $\mathcal{E}$, and λ_β is complete for the theory of any λ-model $\mathcal{M}$ based on an infinite set and an arbitrary exponent in $\mathcal{M}$: for any simply typed terms t, u one has $t =_\beta u$ iff $\mathcal{M} \models t = u$. Reviewed by Chantal Berline.

Jiang, Y. (1996a). Consistency and axiomatization of a natural extensional combinatory logic. *Sci. China Ser. A*, 39(9):897–908.

[**MR99c:03015**] Summary: "In the light of a question of J.-L. Krivine about the consistency of an extensional λ-theory, an extensional combinatory logic ECL + $U(G) + \mathrm{RU}_\infty + \mathcal{F}$ is established, its consistency is proved model-theoretically, and it is shown that it is not equivalent to any system of universal axioms. It is expressed by the theory in first-order logic that, for every given group G of order n, there simultaneously exist infinitely many universal retractions and a surjective n-tuple notion such that each element of G acts as a permutation of the components of the n-tuple, and as an Ap-automorphism of the model; further, each of the universal retractions is invariant under the action of the Ap-automorphisms induced by G. The difference between this theory and that of Krivine is that G need not be a symmetric group."

Jiang, Y. (1996b). Isomorphisms between the coherent models of the lambda-calculus. *Sci. China Ser. A*, 39(8):785–798.

[**MR98d:03023**] In this paper the author studies definability in a class of well-founded (extensional) coherent models of untyped lambda calculus and proves that, as in the case of, e.g., Engeler's graph models, such models are determined, up to isomorphism, by the cardinality of their sets of atoms—thus adding to the apparent canon in the mathematics of set-theoretic models of untyped λ-calculus and combinatory logic.

See also papers by R. Kerth [C. R. Acad. Sci. Paris Ser. I Math. 318 (1994), no. 7, 685–688; MR 95i:68077], G. D. Plotkin [Theoret. Comput. Sci. 121 (1993), no.

1-2, 351–409; MR 95b:03014] and the reviewer [J. Symbolic Logic 56 (1991), no. 1, 227–249; MR 93c:03016]. Reviewed by Harold Schellinx.

Maietti, M. E. (1996). Some aspects of the categorical semantics for the polymorphic λ-calculus. In *Logic and algebra (Pontignano, 1994)*, pages 589–601. Dekker, New York.

[MR99a:03022] This paper investigates a new model of second-order lambda calculus, consisting of a categorical presentation of an algebraic semantics, namely Girard's coherence spaces. The key notion is the definition of PLF-categories as a generalization of Seely's PL-categories, since the Cartesian closed categories are indexed by a Cartesian category rather than a Cartesian closed one. The author proves that these generalized structures can provide a complete semantics of polymorphic lambda calculus. The most interesting point is the presentation of a concrete example of a PLF-category based on coherence spaces, thus correlating algebraic and categorical semantics of polymorphism.

The paper is neat and pleasant to read. For the entire collection see MR 97c:00047. Reviewed by Betti Venneri.

Révész, G. (1996). Categorical combinators with explicit products. *Publ. Math. Debrecen*, 48(3-4, suppl.):389–403. Automata and formal languages, VII (Salgótarján, 1993).

[MR97e:03015] A revised and extended version of this paper has been reviewed [Fund. Inform. 22 (1995), no. 1-2, 153–166; MR 96g:03022]. For the entire collection see MR 96m:68003.

Alessi, F., Dezani-Ciancaglini, M., and de'Liguoro, U. (1997). A convex powerdomain over lattices: its logic and λ-calculus. *Fund. Inform.*, 32(3-4):193–250.

[MR1657310] A review for this item is in process.

Coquand, T. and Dybjer, P. (1997). Intuitionistic model constructions and normalization proofs. *Math. Structures Comput. Sci.*, 7(1):75–94.

[MR98b:03022] Summary: "The traditional notions of strong and weak normalization refer to properties of a binary reduction relation. In this paper we explore an alternative approach to normalization, in which we bypass the reduction relation and instead focus on the normalization function, that is, the function that maps a term to its normal form. We work in an intuitionistic metalanguage, and characterize a normalization function as an algorithm that picks a canonical representative from the equivalence class of convertible terms. This means that we also get a decision algorithm for convertibility.

"Such a normalization function can be constructed by building an appropriate model and a quote function, which inverts the interpretation function. The normalization function is then obtained by composing the quote function with the interpretation function. We also discuss how to get a simple proof of the property that constructors are one-to-one, which is usually obtained as a corollary of the Church-Rosser property and normalization in the traditional sense.

"We illustrate this approach by showing how a glueing model (closely related to the glueing construction used in category theory) gives rise to a normalization algorithm for a combinatory formulation of Gödel's system T. We then show how the method extends in a straightforward way when we add Cartesian products and disjoint unions (full intuitionistic propositional logic under a Curry-Howard interpretation) and transfinite inductive types such as the Brouwer ordinals."

Corrêa, M. D. S. and Haeusler, E. H. (1997). A concrete categorical model for the Lambek syntactic calculus. *Math. Logic Quart.*, 43(1):49–59.
[MR97m:03028] Summary: "We present a categorical/denotational semantics for the Lambek Syntactic Calculus (LSC), indeed for a λ-typed version Curry-Howard isomorphic to it. The main novelty of our approach is an abstract noncommutative construction with right and left adjoints, called the sequential product. It is defined in terms of a hierarchical structure of categories reflecting the implicit permission to sequence expressions and the inductive construction of compound expressions. We claim that Lambek's noncommutative product corresponds to a noncommutative bi-endofunctor into a category which includes all categories of such hierarchical structure. A soundness theorem for LSC is shown with respect to this semantical framework."

Fu, Y. (1997). Categorical properties of logical frameworks. *Math. Structures Comput. Sci.*, 7(1):1–47.
[MR98g:03150] This paper presents a model-theoretic account of logical frameworks. The latter are special type theories designed so that other type theories or logics can be encoded in them (see notably Martin-Löf's logical framework [B. Nordstrom, K. Petersson and J. M. Smith, Programming in Martin-Löf's type theory, Oxford Univ. Press, New York, 1990; MR 94g:68020] and the Edinburgh logical framework [R. Harper, F. Honsell and G. D. Plotkin, J. Assoc. Comput. Mach. 40 (1993), no. 1, 143–184; MR 94d:68071]). The author presents his own logical framework λ_{TT}. It distinguishes itself in particular by a "lifting" operation $t_U(-)$ for every universe U of an encoded system. It takes elements $x\colon U$ of the universe to types in the framework. In the term model of the framework, this lifting operation gives rise to a context projection $(x\colon U, y\colon t_U(x)) \to (x\colon U)$, which in turn gives rise in a standard way to an internal category capturing (the U-part of) the system that is encoded. This is the key aspect of the author's view that what internal categories are to an ambient category, object languages are to a logical framework. This (elegant) view is elaborated in detail, notably via the notion of "internal definability" of one fibration/type theory in another. Various examples are given in which adequacy of an encoding plays a special role. Reviewed by Bart Jacobs.

Goldblatt, R. (1997). The functional lambda abstraction algebras form a variety. In *Combinatorics, complexity, & logic (Auckland, 1996)*, pages 226–243. Springer, Singapore.
[MR1647312] A review for this item is in process.

Hasegawa, M. (1997). Recursion from cyclic sharing: traced monoidal categories and models of cyclic lambda calculi. In *Typed lambda calculi and applications (Nancy, 1997)*, pages 196–213. Springer, Berlin.
[MR98g:03043] Summary: "Cyclic sharing (cyclic graph rewriting) has been

used as a practical technique for implementing recursive computation efficiently. To capture its semantic nature, we introduce categorical models for lambda calculi with cyclic sharing (cyclic lambda graphs), using the notions of computation of Moggi/Power and E. P. Robinson and the traced monoidal categories of Joyal, Street and Verity. The former is used for representing the notion of sharing, and the latter is used for cyclic data structures. Our new models provide a semantic framework for understanding recursion created from cyclic sharing, which includes traditional models for recursion created from fixed points as special cases. Our cyclic lambda calculus serves as a uniform language for this wider range of models of recursive computation." For the entire collection see MR 98f:68017.

Pigozzi, D. and Salibra, A. (1997). Lambda abstraction algebras: coordinatizing models of lambda calculus. *Fund. Inform.*, 32(1):47–90.
[MR99c:03016] Summary: "Lambda abstraction algebras are designed to algebraize the untyped lambda calculus in the same way cylindric and polyadic algebras algebraize first-order logic; they are intended as an alternative to combinatory algebras in this regard. Like combinatory algebras they can be defined by true identities and thus form a variety in the sense of universal algebra. One feature of lambda abstraction algebras that sets them apart from combinatory algebras is the way variables in the lambda calculus are abstracted; this provides each lambda abstraction algebra with an implicit coordinate system. Another peculiar feature is the algebraic reformulation of (β)-conversion as the definition of abstract substitution. Functional lambda abstraction algebras arise as the 'coordinatizations' of environment models or lambda models, the natural combinatory models of the lambda calculus. As in the case of cylindric and polyadic algebras, questions of the functional representation of various subclasses of lambda abstraction algebras are an important part of the theory. The main result of the paper is a stronger version of the functional representation theorem for locally finite lambda abstraction algebras, the algebraic analogue of the completeness theorem of lambda calculus. This result is used to study the connection between the combinatory models of the lambda calculus and lambda abstraction algebras. Two significant results of this kind are the existence of a strong categorical equivalence between lambda algebras and locally finite lambda abstraction algebras, and between lambda models and rich, locally finite lambda abstraction algebras."

Power, J. and Thielecke, H. (1997). Environments, continuation semantics and indexed categories. In *Theoretical aspects of computer software (Sendai, 1997)*, pages 391–414. Springer, Berlin.
[MR1608903] This item will not be reviewed individually. For details of the collection in which this item appears see 98i:68019.

Statman, R. (1997). On Cartesian monoids. In *Computer science logic (Utrecht, 1996)*, pages 446–459. Springer, Berlin.
[MR98k:03041] From the summary: "A Cartesian monoid is a structure $(M, *, I, L, R, \langle\rangle)$ where $(M, *, I)$ is a monoid with L and R members of M and $\langle\rangle\colon M^2 \to M$ such that $L * \langle x, y\rangle = x, R * \langle x, y\rangle = y, \langle x, y\rangle * z = \langle x * z, y * z\rangle$, and $\langle L, R\rangle = I$. Cartesian monoids are easy to come by. Any surjective pairing function

lifts pointwise to a $\langle\rangle$ on the monoid M of all functions under composition with L and R the lifted projections. Cartesian monoids were first introduced by Dana Scott and independently by J. Lambek.

"It is the purpose of this paper to collect in one place our observations on Cartesian monoids, especially the free Cartesian monoid, which are useful for applications to lambda calculus." For the entire collection see MR 98k:03005.

Čubrić, D. (1998). Embedding of a free Cartesian-closed category into the category of sets. *J. Pure Appl. Algebra*, 126(1-3):121–147.
[MR99d:18004] Summary: "We show that every free Cartesian closed category can be faithfully mapped to the category of sets. For that we use a Church-Rosser property of the appropriate typed lambda calculus." Reviewed by Maria E. Maietti.

Čubrić, D., Dybjer, P., and Scott, P. (1998). Normalization and the Yoneda embedding. *Math. Structures Comput. Sci.*, 8(2):153–192.
[MR99d:03012] Summary: "We show how to solve the word problem for the simply typed $\lambda\beta\eta$-calculus by using a few well-known facts about categories of presheaves and the Yoneda embedding. The formal setting for these results is $\mathcal{P}$-category theory, a version of ordinary category theory where each hom-set is equipped with a partial equivalence relation. The part of $\mathcal{P}$-category theory we develop here is constructive and thus permits extraction of programs from proofs. It is important to stress that in our method we make no use of traditional proof-theoretic or rewriting techniques. To show the robustness of our method, we give an extended treatment for more general λ-theories in the Appendix." Reviewed by Maria E. Maietti.

Dezani-Ciancaglini, M., de'Liguoro, U., and Piperno, A. (1998). A filter model for concurrent λ-calculus. *SIAM J. Comput.*, 27(5):1376–1419 (electronic).
[MR99f:68025] Summary: "Type-free lambda-calculus is enriched with angelic parallelism and demonic nondeterminism. Call-by-name and call-by-value abstractions are considered and the operational semantics is stated in terms of a must convergence predicate. We introduce a type assignment system with intersection and union types, and we prove that the induced logical semantics is fully abstract."

One can note here that full abstraction comes from the fact that all compact elements of the dual filter model are definable (are interpretations of terms).

Besides giving a "convincing treatment" of all the above programming features within a unique system, the paper contains a very comprehensive comparison with related works. Reviewed by Chantal Berline.

Drakengren, T. (1998). A decidable canonical representation of the compact elements in Scott's reflexive domain in $P\omega$. *Theoret. Comput. Sci.*, 193(1-2):181–195.
[MR98m:06014] Summary: "The extensional model in $P\omega$ for the pure λ-calculus, found by Scott in 1976, is provided with a canonical, decidable representation of its compact elements, with the interesting property that it does not at all refer to the retract that defines the domain, but only to the codings of the $P\omega$ model." Reviewed by Guo-Qiang Zhang.

Goldblatt, R. (1998). Enlargements of functional algebras for the lambda calculus. *Theoret. Comput. Sci.*, 198(1-2):177–200.

[MR99b:03027] The author proves that an RFA, relativized functional algebra, is isomorphic to an FLA, (full) functional lambda algebra, by using a tool from nonstandard analysis, namely a comprehensive enlargement with uniformity conditions. As a corollary of this and results of D. L. Pigozzi and A. Salibra [Theoret. Comput. Sci. 140 (1995), no. 1, 5–52; MR 96b:68123], he shows that the exhibited equations characterize the class of FLAs. The proof naturally hinges on the fact that a "finite set" in an enlargement can include an infinite set of interest, and thus a full algebra can be viewed as relativized. The paper begins with succinct summaries of relevant facts from λ-calculus and nonstandard analysis. The existence of an enlargement of the above nature is justified by an ultrapower construction; the exposition is very clear. The author talks about the potential of this approach for other parts of mathematics. He also quotes, from Robinson's 1966 book, the statement that nonstandard analysis might offer new deductive procedures. But he neither discusses nor cites E. Nelson's 1977 paper "Internal set theory" [Bull. Amer. Math. Soc. 83 (1977), no. 6, 1165–1198; MR 57 # 9544], or subsequent works by him and by others, that carry out Robinson's idea. Reviewed by M. Yasuhara.

Kerth, R. (1998). Isomorphism and equational equivalence of continuous λ-models. *Studia Logica*, 61(3):403–415.

[MR1657121] A review for this item is in process.

Jacobs, B. (1999). *Categorical logic and type theory.* North-Holland Publishing Co., Amsterdam.

[MR1674451] A review for this item is in process.

Kerth, R. (1999). Forcing in stable models of untyped λ-calculus. *Indag. Math. (N.S.)*, 10(1):59–71.

[MR1691459] A review for this item is in process.

Salibra, A. and Goldblatt, R. (1999). A finite equational axiomatization of the functional algebras for the lambda calculus. *Inform. and Comput.*, 148(1):71–130.

[MR1659467] A review for this item is in process.

2 Applications

2.1 Computing

Astesiano, E. and Costa, G. (1979). Sharing in nondeterminism. In *Automata, languages and programming (Sixth Colloq., Graz, 1979)*, pages 1–15. Springer, Berlin.

[MR82b:68006] Typed lambda-calculus with if-then-else and fixed-point functionals is extended by an or-function for ground types with reductions: $or\, a_1 a_2 \to a_i$, i=1, 2, so the expression can have several different values. This reflects nondeterminism. Execution of substitutions resulting from β-reductions $\lambda x M.N \to M(N/x)$ is postponed as long as possible, so that reductions in N are made once for several occurrences of x in M, especially in *or* $a(x)b(x)$. This is called sharing. Operational semantics is defined via outermost reduction. The denotational semantics defined here uses many-one functions via power-set models. The equivalence of both types of semantics is proved and the possibility of constructing fully abstract models is mentioned. (For the entire collection see MR 81c:68002.) Reviewed by G. E. Mints.

Coppo, M., Dezani-Ciancaglini, M., and Salle, P. (1979). Functional characterization of some semantic equalities inside λ-calculus. In *Automata, languages and programming (Sixth Colloq., Graz, 1979)*, pages 133–146. Springer, Berlin.

[MR81d:68043] Authors' summary: "Both (operational or denotational) semantics and type theories for lambda -calculus induce in a natural way equivalence relations between terms. The aim of the present paper is to show that in some cases the semantic and functional equivalences coincide." (For the entire collection see MR 81c:68002.)

Pettorossi, A. (1979a). A hierarchy of infinite computations in type free subtree replacement systems. In *Les arbres en algèbre et en programmation (4ème Colloq., Lille, 1979)*, pages 181–195. Univ. Lille I, Lille.

[MR81b:68008] Author's summary: " A hierarchy of infinite computations is introduced. Although, in general, the association of a computation with the corresponding position in the hierarchy is not decidable, it turns out that such a classification is useful for studying nontermination properties for subrecursive classes of algorithms." (For the entire collection see MR 80j:68004.)

Pettorossi, A. (1979b). On the definition of hierarchies of infinite sequential computations. In *Fundamentals of computation theory (Proc. Conf. Algebraic, Arith. and Categorical Methods in Comput. Theory, Berlin/Wendisch-Rietz, 1979)*, pages 335–341. Akademie-Verlag, Berlin.

[MR81j:68021] Author's summary: "A method for defining hierarchies of infinite sequential computations is introduced. We apply the method to computations defined in weak combinatory logic and we show that it is useful for studying termination and nontermination properties of subrecursive classes of algorithms." (For the entire collection see MR 80m:68007.)

Robinet, B. (1979). Logique combinatoire et programmation: une autre approche de la théorie de la programmation. *Rend. Sem. Mat. Univ. Politec. Torino*, 36:35–37.

[MR80b:03017] This is a brief summary of a lecture in which the author discussed the possibility of translating programs into expressions of the lambda-calculus following the work of P. Landin, C. Böhm and J. McCarthy. No details of the translation are given. There is a list of 29 references.

Astesiano, E. and Costa, G. (1980). Languages with reducing reflexive types. In *Automata, languages and programming (Proc. Seventh Internat. Col-*

loq., Noordwijkerhout, 1980), pages 38–50. Springer, Berlin.

[MR81j:68082] Authors' summary: "We consider a class of languages of typed lambda-expressions with first-order primitives and reflexive types and investigate which types it is useful and consistent to allow, when the meaning of a term is defined by its behavior as a part of programs. We define the class of reducing types and show that it is irrelevant to allow the nonreducing ones in our languages. Moreover we prove that, for any language with reducing types and a sufficiently rich set of primitives, there is a fully abstract model, in the sense of Milner." (For the entire collection see MR 81i:68004.)

Hennessy, M. C. B. and Ashcroft, E. A. (1980). A mathematical semantics for a nondeterministic typed λ-calculus. *Theoret. Comput. Sci.*, 11(3):227–245.

[MR82a:68049] Authors' summary: "This paper is concerned with the problems encountered in defining the semantics of nondeterministic algorithms. A nondeterministic control structure is added to a typed lambda-calculus and the usual operational semantics for the deterministic language is generalized to take into account the more complex behaviour of nondeterministic algorithms. A mathematical model is then given for the language and the relationship between the denotational and operational semantics is explored." Reviewed by Gianni Aguzzi.

Perl, J. (1980). Reachability homomorphisms on graphs. In *Discrete structures and algorithms (Proc. Conf., Berlin, 1979)*, pages 105–113. Hanser, Munich.

[MR82h:68093] A reachability homomorphism is defined on directed graphs. Specifically, an $arc(y, z)$ can be contracted (fused) if z is the only successor of y or y is the only predecessor of z. This homomorphism is shown to have the finite Church-Rosser property. (For the entire collection see MR 81m:83045.) Reviewed by Marianne Gardner.

Robinet, B. (1980). Les langages de Backus sont des systèmes de manipulation d'arbres. In *Trees in algebra and programming (Proc. 5th Lille Colloq., Lille, 1980)*, pages 83–94. Univ. Lille I, Lille.

[MR82g:68018] From the introduction: "We present a model of Backus languages constructed according to combinatory logic; we show that these languages are in fact special systems of tree manipulation, and we conclude with a consistency theorem similar to that of Church and Rosser." (For the entire collection see MR 82e:68004.)

Salle, P. (1980a). Une généralisation de la théorie des types en λ-calcul. I. *RAIRO Inform. Théor.*, 14(2):143–167.

[MR82f:03012a] (Joint review with (Salle, 1980b).) Computer scientists have become interested in the lambda-calculus in recent years because it has been discovered that it is a good formalism for expressing the semantics of programming languages. Any program can be translated automatically into a lambda-term which, in a sense, represents its important properties. One of the more interesting properties is whether

a particular computation terminates; this is equivalent to the question of whether the corresponding lambda-term has a normal form. The typed lambda-calculus is of interest in this context because every term with a type has a normal form. However, the classical typed lambda-calculus is quite limited since it is possible that if P represents a program and X some data, then PX may have a normal form without P (or X) having a normal form, and in this case PX will have no type. Furthermore, some terms related to programming do not have types: (1) if F is a function then $F(F(x))$ has a type only if the value of F has the same type as its argument, so that if $F(x)$ is the integer part of x, where x is a real number, then $F(F(x))$ has no type; (2) the fixed point combinator, which is useful for expressing the semantics of iterative and recursive calculations, has no normal form and hence no type; (3) $IF\,\mathrm{condition}\,THEN\,A\,ELSE\,B$ has a type only if A and B correspond to calculations which terminate. It is therefore desirable to extend the usual typed λ-calculus so that more of the terms which represent programs have types. M. Coppo and M. Dezani-Ciancaglini [Arch. Math. Logik Grundlag. 19 (1978/79), no. 3-4, 139- 156; MR 81d:03014] proposed extending the usual assignment by introducing two new basic types and two new relations on types. The new basic types, 0 and 1, are to denote, respectively, terms having a normal form and terms having the property that if any of them is applied to any number of terms each having a normal form, the result has a normal form. To assign these types to the appropriate terms, it is sufficient to define an equivalence relation and a partial ordering relation on the set of types. (Here types are built up in the usual way from the basic types by forming, from types σ and τ, the type $(\sigma)\tau$ of functions whose arguments are of type σ and whose values are of type τ.) The equivalence relation is defined by any finite number of replacements of one side of a defining equation by the other, where the defining equations are $1 = (0)1$ and $0 = (1)0$. The partial order is specified by $0 \sqsubseteq 1$ and $(\sigma_1)\tau_1 \sqsubseteq (\sigma_2)\tau_2$ if and only if $\sigma_2 \sqsubseteq \sigma_1$ and $\tau_1 \sqsubseteq \tau_2$. It is easy to show that for any type τ, $0 \sqsubseteq \tau \sqsubseteq 1$. (This remains true if there are additional basic types θ provided that they all satisfy $0 \sqsubseteq \theta \sqsubseteq 1$.) To the usual natural deduction rules Fe (from $(\sigma)\tau X$ and σY to deduce $\tau(XY)$) and Fi (from $\mathcal{B}, \sigma x \vdash \tau X$, where x does not occur (free) in $\mathcal{B}$, to deduce $\mathcal{B} \vdash (\sigma)\tau(\lambda x.X)$) of type assignment are added rules permitting an inference from σX to τX when either $\sigma = \tau$ or $\sigma\tau$. It then follows that any term which is in normal form is assigned type 0 provided that each of its free variables is assigned type 1; for any term in normal form it is decidable whether or not it possesses a type; if for any terms N and M in normal form there is an assignment to the free variables which results in the assignment of type 1 to N, then MN and NM have normal forms; and every term with a type has a normal form. In the first paper, the author extends this system of Coppo and Dezani- Ciancaglini by adding a new basic type, ω, which is to be assigned to every term (and is thus the same as Curry's type E). This property of ω is obtained by adding the defining equation $\omega = (\tau)\omega$ for each type τ and by specifying that $\omega \sqsubseteq 0$. The author then proves that all of the properties of the system of Coppo and Dezani-Ciancaglini carry over to his system provided that the last one given in the previous paragraph is changed to: every term which is assigned a type $\tau 0$ has a normal form which also has type τ. He also proves that every term with a type distinct from ω has a head normal form. He concludes with a counterexample to the converses of the last two results by giving a term which has a normal form (and thus a head normal form) but whose only type is ω. This counterexample shows that the theory presented so far is still too limited. In the second paper, the author uses another idea of Coppo and Dezani-Ciancaglini [Notre Dame J. Formal Logic 21 (1980), no. 4, 685- 693; MR 82d:03023], that of using finite sequences of types as individual types. The operation for forming compound types now forms, from types $\sigma_1, \ldots, \sigma_n, \tau$ the type $[\sigma_1, \ldots, \sigma_n]\tau$. The equivalence relation between types is now generated by $0 = [1]0$, $1 = [0]1$, and $\omega = [\tau]\omega$, where τ is an abbreviation for $\tau_1, \ldots, \tau_n$. The partial order is generated by $\omega \sqsubseteq 0 \sqsubseteq 1$ and the rule $[\tau_1, \ldots, \tau_m]\tau_(m+1) \sqsubseteq [\sigma_1, \ldots, \sigma_n]\sigma_(n+1)$ if and only if $\tau_(m+1) \sqsubseteq \sigma_(n+1)$ and for each i, $1 \leq i \leq n$, there is a j, $1 \leq j \leq m$, such that $\sigma_i \sqsubseteq \tau_j$. The rule Fe is modified as follows: from $[\sigma_1, \ldots, \sigma_n]\tau X$ and $\sigma_i Y$ for each $i = 1, \ldots, n$ to deduce $\tau(XY)$. And the rule Fi is modified to: from $\mathcal{B}, \sigma_1 x, \ldots, \sigma_n x \vdash \tau X$ to deduce $\mathcal{B} \vdash [sigma_1, \ldots, \sigma_n]\tau(lambdax.X)$. In this system, the term mentioned in the last paragraph (whose only type in the original system is ω) can be proved to have type 0. The author goes on to prove that two terms which are $\beta\eta$-convertible have the same types. He finishes by proving that each term with a type $\tau 0$ [respectively, distinct from ω] has a normal form [respectively, head normal form], thus completing the proof that a term has a normal form [respectively, head normal form] if and only if it has a type 0 [respectively, distinct from ω]. (Unfortunately, these last proofs have gaps which the reviewer cannot fill. In the proofs of Lemmas 11 and 13 (top of p. 306 and p. 308, respectively), the author assumes that if a term in head normal form [respectively, normal form] has type 0 or 1, then so does each of its subterms (provided that each of these subterms which has a type distinct from ω [respectively, 0] is also in head normal form [respectively, normal form]). This is true only if the terms are constants, and in the context in which the author uses this assumption, there are free variables present. And in another part of the proof of Lemma 13 (middle of p. 307, Case 1), the author's assertion that $|\Psi| < |\sigma|$, which is necessary for the application of the induction hypothesis, will be false if none of the ν_i contain any sequences (with more than one element). (The reviewer has the following additional remarks: (1) At the bottom of p. 147, in the statement of the meaning of axiom A_0, 'XN' should be 'NX'. (2) On p. 151, Example 5, the first line of the proof should read as follows: '$(\sigma)\tau x \vdash (\sigma)\tau x$, $(\sigma)\tau x \vdash \sigma x$'. (3) In the proof of Proposition 2 on p. 153, the author has not proved that if $\mathcal{B}, \sigma x \vdash \tau F$ and $\mathcal{B} \vdash \sigma G$ then $\mathcal{B} \vdash \tau([G/x]F)$; this proof may involve complications because of the binding of variables by Rule Fi. (4) In the proof of Proposition 3 on p. 153, it might be worth mentioning in the case for $|N| = 1$ (where N is a variable, x), that if $\mathcal{B} = (\tau x)$ for any type τ we have $\mathcal{B} \vdash \omega x$ since $\omega \sqsubseteq \tau$ (by rule R_i); the proof then shows that for any term X and any assignment of types to its free variables, X is assigned type ω . (5) The remark at the bottom of p. 154 is unclear, for ω is, in fact, assigned to every β-redex (as to every other term), but we cannot from this fact alone determine whether any given redex is assigned any other types, and determining all types assigned to a given β-redex appears to be necessary to decide whether a term containing such a redex has a type

(even if the type is given). (6) In the proof of Lemma 7, p. 159, it should be noted that either F or X has type 1; if F has type 0, then $(\sigma)\tau = 0$, which implies that σ, the type of X, equals 1. (7) On p. 310, just before Lemma A, it should probably be mentioned that $\Omega = \Delta\Delta$.) Reviewed by J. P. Seldin.

Salle, P. (1980b). Une généralisation de la théorie des types en λ-calcul. II. *RAIRO Inform. Théor.*, 14(3):301–314.
[MR82f:03012b] See (Salle, 1980a).

van der Poel, W. L., Schaap, C. E., and van der Mey, G. (1980). New arithmetical operators in the theory of combinators. I, II, III. *Nederl. Akad. Wetensch. Indag. Math.*, 42(3):271–286, 287–302, 303–325.
[MR82a:68163]

The authors make some proposals for defining a modified calculus of combinators and give several schemes for representing integers and integer operations within that calculus. Extensive documentation is given of a LISP implementation of the calculus, and of the performance of that implementation. The paper reads as if it were written by programmers who, discovering that very small calculi of combinators can be regarded as rudimentary programming languages, display their programming skill by giving algorithms which run faster, or in less space, or which have a more amusing code than the conventional ones. This effort seems to the reviewer mostly misconceived. The conventional algorithms were developed for theoretical reasons and were never intended to be judged by such criteria. If a calculus of combinators were to be developed as a practical programming language, it would undoubtedly be provided with efficient primitive representations of (some) integers and integer operations. The work under review might have more value if recast to give schemes for extended-precision arithmetic over an arbitrary given initial segment of integers. The title's reference to "the" theory of combinators is misleading, since the authors consider a calculus of combinators which has seldom if ever been considered before. For that reason, any value which the work may have for the theory of standard calculi of combinators remains hidden. It is unfortunate that the authors restrict their notion of reduction (evaluation) of combinators so that evaluation rules can be applied to at most one position in an expression – the leftmost position which matches the rule. This restriction reduces the interest of the work for theorists and makes it inapplicable to practical work on parallel processing. A bright spot is that the authors experiment with dynamic lambda-binding; that is, the use of a lambda symbol as an operator in its own right, which can be used during evaluation to bind the free occurrences of a nominated variable in a nominated expression. Such a device may have both theoretical and practical value and should be put on a sound basis. Reviewed by John Staples.

Ward, S. A. and Halstead, Robert H., J. (1980). A syntactic theory of message passing. *J. Assoc. Comput. Mach.*, 27(2):365–383.
[MR81b:68013] Authors' summary: "Recent developments by Hewitt and others have stimulated interest in message-passing constructs as an alternative to the more

conventional applicative semantics on which most current languages are based. The present work illuminates the distinction between applicative and message-passing semantics by means of the μ-calculus, a syntactic model of message-passing systems similar in mechanism to the lambda-calculus. Algorithms for the translation of expressions from the lambda- to the μ-calculus are presented, and differences between the two approaches are discussed. "Message-passing semantics seems particularly applicable to the study of multiprocessing. The μ-calculus, through the mechanism of conduits, provides a simple model for a limited but interesting class of parallel computations. Multiprocessing capabilities of the μ-calculus are illustrated, and multiple-processor implementations are discussed briefly."

Aiba, A. and Nakanishi, M. (1981). General calculator based on lambda conversion. *Keio Math. Sem. Rep.*, (6):1–9.
[MR83a:68029] The authors outline an algorithm for reducing lambda-terms, and state that it can be implemented on a computer. The authors seem to be unaware of the last ten years' work on this kind of problem in Europe and the U.S.A. They consider neither termination nor efficiency, for example. Also, their algorithm has one or two peculiarities: e.g., terms $\lambda v.E$ are not reduced at all, and in reducing terms YZ with Y a constant, Z is not reduced. These are not motivated. Reviewed by R. Hindley.

Berry, G. (1981). On the definition of lambda-calculus models. In *Formalization of programming concepts (Proc. Internat. Colloq., Pe níscola, 1981)*, pages 218–230. Springer, Berlin.
[MR82h:68044] Author's summary: "Most authors use a classical Tarski-like definition of lambda-calculus models. However, in earlier papers we constructed semantic 'models' which do not obey this definition. We propose a new definition of models which is strictly more general than the usual one, takes care of all known cases, but is slightly more complicated. We illustrate the need for this definition by various examples, and compare it to the usual definition." (For the entire collection see MR 82f:68005.)

Kozen, D. (1981). Semantics of probabilistic programs. *J. Comput. System Sci.*, 22(3):328–350. Special issue dedicated to Michael Machtey.
[MR82i:68011] Author's summary: "We present two complementary but equivalent semantics for a high level probabilistic programming language. One of these interprets programs as partial measurable functions on a measurable space. The other interprets programs as continuous linear operators on a Banach space of measures. It is shown how the ordered domains of Scott and others are embedded naturally into these spaces. We use the semantics to prove a general result about probabilistic programs, namely, that a program's behavior is completely determined by its action on fixed inputs."

Le Berre, F. and Nolin, L. (1981). L'existence d'espaces informatiques. *C. R. Acad. Sci. Paris Sér. I Math.*, 292(9):499–502.

[**MR83f:68023**] Authors' summary: "We show that sets $\mathcal{E}$ exist which contain, in addition to given elements, all normal functions from $\mathcal{E}$ to $\mathcal{E}$. The results generalize those of D. S. Scott [see, e.g., Logic, methodology and philosophy of science, IV (Bucharest, 1971), pp. 157 - 187, North-Holland, Amsterdam, 1973; MR 57 #15987] and C. P. Wadsworth ["Formal semantics of programming languages", Ph.D. Thesis, Oxford Univ., Oxford, 1971]."

Volfengagen, V. È. (1981). Applicative computing systems and representation of knowledge of frames in an intelligent data bank. *Tartu Riikl. Ül. Toimetised*, (594):18–42.

[**MR83m:68164**] Frames are defined here as sequents $\Gamma \to \Gamma$, Δ are finite sequences of terms of an untyped lambda-calculus with an implication connective Ξab similar to $\forall x\ ax \subset bx$. All reservations concerning attempts to base predicate logic on a system of this kind apply to the work under review. Reviewed by G. E. Mints.

Mirasyedioğlu, Ş. (1982). λ-functional exclusive-or and logical equivalence. *Karadeniz Univ. Math. J.*, 5(1):64–76. International Nazim Terzioğlu symposium on mathematics (Trabzon, 1982).

[**MR85f:03012**] Author's summary: "λ-functional structures for the logical operators AND, OR and NOT are known. In this paper, λ-functional structures that represent the exclusive-or operator and the logical equivalence operator are studied."

Astesiano, E. and Costa, G. (1983). The insensitivity theorem for nonreducing reflexive types. *J. Comput. System Sci.*, 27(3):389–399.

[**MR85j:03013**] This paper is concerned with distinguishing between useful and useless types in applicative languages with primitive functions of finite type. The key concept, defined originally in the authors' earlier paper [Automata, languages and programming (Noordwijkerhout, 1980), 38–50, Springer, Berlin, 1980; MR 81j:68082] is that of a reducing type. To a first approximation, a type σ is said to be a reducing type if it is of the form $\sigma = \sigma_1 \to \cdots \to \sigma_n \to \kappa$ where κ is of ground type; the complete definition further restricts the σ_i to have been generated in a particular natural way. The main result is the "insensitivity theorem" which, to quote the paper, shows that, "intuitively speaking, contexts of finite type and closed with respect to variables of reflexive type are insensitive to differences between terms of nonreducing type". The actual theorem is formally presented in terms of the concept of syntactic value in the sense of G. Berry [Automata, languages and programming (Udine,1978), 72–89, Springer, Berlin, 1978; MR 80d:68034] and J. J. Levy ["Correct and optimal reductions in the lambda-calculus" (French), These d'Etat, Univ. Paris VII, Paris, 1978]. This theorem is then employed to show that nonreducing types are useless whenever the semantics are fully abstract. Reviewed by E. G. Wagner.

Constable, R. L. (1983). Constructive mathematics as a programming logic. I. Some principles of theory. In *Foundations of computation theory (Borgholm, 1983)*, pages 64–77. Springer, Berlin.

[**MR85f:68060**] Author's summary: "The design of a programming system is guided by certain beliefs, principles, and practical constraints. These considerations are not always manifest from the rules defining the system. This paper discusses some of the principles which have guided the design of the programming logics built at Cornell in the last decade. Most of the necessarily brief discussion concerns type theory, with emphasis on the concepts of function space and quotient types." For the entire collection see MR 85e:68003.

Giovannetti, E. (1983). Une caractérisation des termes typés dans un langage applicatif. *C. R. Acad. Sci. Paris Sér. I Math.*, 296(1):97–99.

[**MR84b:03029**] Author's summary: "In a simple applicative language including disjoint sum and Cartesian product, a semantic characterization of the set of typed terms and a completeness result for a nontrivial subset are given."

Raoult, J.-C. and Sethi, R. (1983). Properties of a notation for combining functions. *J. Assoc. Comput. Mach.*, 30(3):595–611.

[**MR85a:68116**] Authors' summary: "A combinator called a pipe is proposed for combining functions in a linear order. Examples suggest that semantic rules using pipes are easy to read and understand, even for readers with little knowledge of semantics. The readability is a consequence of the operational intuition associated with pipes. The operational view is that each function connected by a pipe is handed a finite sequence of values. Each function takes zero or more arguments from the right end of the sequence, replaces the arguments by zero or more results, and passes the sequence to the next function. The new idea is that a function may skip over some number of values before picking up its arguments. This approach is suited to expressing the composition of operations on machine states in a programming language. Pipes allow continuation semantics to be written with direct operators: instead of the operator having to worry about its continuation, the second function in a pipe is essentially a continuation of the first. A connection is established between functions connected by pipes and more traditional continuation semantics."

Bruce, K. B. and Meyer, A. R. (1984). The semantics of second-order polymorphic lambda calculus. In *Semantics of data types (Valbonne, 1984)*, pages 131–144. Springer, Berlin.

[**MR784455**] This item will not be reviewed individually. For details of the collection in which this item appears see 85m:68003.

Bunder, M. W. (1984). Some problems and inadequacies in the van der Poel-Schaap-van der Mey theory of combinators. *Nederl. Akad. Wetensch. Indag. Math.*, 46(4):373–377.

[**MR87b:03027**] This article comments on a paper by W. L. van der Poel et al. [Nederl. Akad. Wetensch. Indag. Math. 42 (1980), no. 3, 271–325; MR 82a:68163]. Part I of that paper described axioms for an abstract theory of combinators, part II gave a new way of representing natural numbers by pure combinators, and part III displayed some computer programs for basic arithmetical operations.

The author points out that the axioms in part I do not describe the authors' practice in parts II and III. He shows that the axioms are too weak to allow the reductions which have actually been built into the programs in part III; and if the axioms were strengthened by adding rules which express these transformations in a natural way, then they would become inconsistent (i.e. all terms would reduce to all others). Reviewed by R. Hindley.

Hook, J. G. (1984). Understanding Russell—a first attempt. In *Semantics of data types (Valbonne, 1984)*, pages 69–85. Springer, Berlin.
[MR784452] This item will not be reviewed individually. For details of the collection in which this item appears see 85m:68003.

Longo, G. (1984). Continuous structures and analytic methods in computer science. In *Ninth colloquium on trees in algebra and programming (Bordeaux, 1984)*, pages 1–22. Cambridge Univ. Press, Cambridge.
[MR86k:68067] Author summary: "Most of the structures one deals with in computer science have a discrete and effective character: (finite) graphs, nets, trees in programming and formal languages, algorithms over finite strings or natural numbers, circuits, etc. By continuous structures we mean 'smooth' spaces, usually of cardinality not less than that of the continuum, where interesting topological or order properties give some information on, say, classes of functions over them. By discussing results from various areas of mathematical computer science, we emphasize the role of continuous structures as tools for proving results about discrete or even finite structures. In particular, we survey results concerning trees in lambda-calculus, functionals in computability theory, and Boolean circuits in complexity theory and relate the finitary/combinatorial nature of the problems with their continuous solutions. We focus mostly on the methodology, more than on the technical aspects of the results presented." For the entire collection see MR 86b:68001.

Breazu-Tannen, V. and Meyer, A. R. (1985). Lambda calculus with constrained types. In *Logics of programs (Brooklyn, N.Y., 1985)*, pages 23–40. Springer, Berlin.
[MR808787] This item will not be reviewed individually. For details of the collection in which this item appears see 86f:68007.

Heering, J. (1985). *Partial evaluation and ω-completeness of algebraic specifications.* Stichting Mathematisch Centrum Centrum voor Wiskunde en Informatica, Amsterdam.
[MR86j:68084] The author gives necessary conditions for an algebraic specification to be ω-complete; the meaning, different from the usual one, is the following: all equations valid in the initial model are equationally provable from the equations of the specification, hence are valid in all models. He provides such a specification for Boolean algebras. Reviewed by Irene Guessarian.

Hirokawa, S. (1985). Complexity of the combinator reduction machine. *Theoret. Comput. Sci.*, 41(2-3):289–303 (1986).
[MR88g:68033] Summary: "The complexity of the computation of recursive programs by the combinator reduction machine is studied. The two models of com-

putation are compared with respect to the number of reduction steps. The main theorem states that the time required by the reduction machine is linear in that of the program scheme. The coefficient of linearity is shown to be $O(n^2)$, where n is the maximal number of variables of the functions being used. For the analysis of the combinator codes, the notion of extended combinator code is introduced."

Noshita, K. (1985). Translation of Turner combinators in $O(n \log n)$ space. *Inform. Process. Lett.*, 20(2):71–74.

[MR86e:68051] Author summary: "A practical method for representing Turner combinators is presented, which needs only $O(n \log n)$ space in the worst case for translating lambda expressions of length n. No precomputation is necessary in our translation, which should be contrasted with Burton's proposal. The runtime system can be implemented with virtually no essential change to Turner's reduction machine."

Zaionc, M. (1985). Nondeterministic programs definable in typed λ-calculus. *Fund. Inform.*, 8(1):63–72.

[MR87k:68096] The setting is the simple typed λ-calculus with $N \equiv (0 \to 0) \to (0 \to 0)$ and $\underline{n} \equiv \lambda xyx(\cdots(xy)\cdots)$ (with n x's following λxy) the nth Church numeral. The author shows that the following collections of sets are identical: (1) the smallest collection containing the finite subsets of N and closed under all λ-definable number-theoretic functions, (2) the collection of all subsets of N which can be enumerated by term grammars and (3) the collection of subsets of N computable by nondeterministic, composition-free, regular programs.

The author has made a number of applications of term grammars to the λ-calculus. The interested reader should consult other work by the author ["Gramatyki termow wλ-rachunka z typami" ("Term grammars in typed λ-calculus"), Dissertation, Dept. Comput. Sci., Univ. Warsaw, Warsaw, 1984; per revr.; "The regular expression descriptions of unifier sets", Fund. Inform., to appear; "Word operations definable in the typed λ-calculus", manuscript, 1985; per revr.]. Reviewed by Richard Statman.

Aiba, A. (1986). Relative efficiencies of reduction machines. *Keio Sci. Tech. Rep.*, 39(1):24.

[MR89e:68041] Summary: "The relative efficiencies of the reduction of lambda-terms and that of combinatory expressions are compared when they represent recursive programs belonging to the linear branched recursion schema in a paper by H. R. Strong, Jr. [J. Comput. System Sci. 5 (1971), 245–285; MR 43 #4461]. The terms and the expressions are extended by introducing constants and some primitive operations on them, and all of them are reduced by the left-most reduction strategy. Reduction costs of the lambda-calculus are obtained theoretically, and those of the combinatory logic are obtained by experiments. The conclusions: (1) reduction with subexpression sharing is more efficient than that without sharing, and (2) the reduction in the lambda-calculus is more efficient than that in the combinatory logic if the number of parameters of the program is larger, and the converse is true when its recursion depth is larger."

Bracho, F. (1986). Towards a mathematical theory of computation: λ-calculus, models and fixed points. In *Research program of the XVIIIth national congress of the Mexican Mathematical Society (Spanish) (Mérida, 1984)*, pages 115–124. Soc. Mat. Mexicana, Mexico City.

[MR88d:03020] This paper provides definitions of the basic notions of lambda-calculus and statements of some basic properties. For the entire collection see MR 87m:00016. Reviewed by G. E. Mints.

Jiang, M. (1986). Functional abstraction of combinatory obs with respect to structured-object templates in combinatory logic. *Chinese J. Comput.*, 9(5):361–365.

[MR88a:03031] Summary: "We present a new practical algorithm for carrying out functional abstraction with respect to object templates (an object template may be a data structure with variables as its atomic elements) on combinatory obs. This algorithm can perform abstraction with respect to all specified variables in a single step, yield abstracts (combinatorics) in a concise form, and retain most, if not all, of the natural parallelism of the original combinatory ob in a straightforward form."

Lieu, C. T. (1986). Point-fixe sur un ensemble restreint. *RAIRO Inform. Théor. Appl.*, 20(4):383–394.

[MR88c:68064] Summary: "This article presents a constructive proof of the existence of normalizing fixed-point operators restricted to λ-term subclasses. Domains where fixed-point operators can be defined cover the usual areas of functional programming applications."

Bird, R. S. (1987). A formal development of an efficient supercombinator compiler. *Sci. Comput. Programming*, 8(2):113–137.

[MR896254] There will be no review of this item.

Castan, M., Durand, M.-H., and Lemaître, M. (1987). A set of combinators for abstraction in linear space. *Inform. Process. Lett.*, 24(3):183–188.

[MR88d:68004] Summary: "A set of simple indexed combinators and an associated linear abstraction algorithm are presented, allowing us to translate a combinator expression into a combinator code containing no variables, in such a way that the length of the generated code is less than twice the length of the original expression in the usual case. The key idea of this algorithm is to operate globally on all variables during the whole abstraction process. This set of combinators is intended for practical use as a reduction machine instruction set."

Kennaway, J. R. and Sleep, M. R. (1987). Variable abstraction in $O(n \log n)$ space. *Inform. Process. Lett.*, 24(5):343–349.

[MR88m:68022] D. A. Turner [Software: Practice & Experience 9 (1979), no. 1, 31–49; J. Symbolic Logic 44 (1979), no. 2, 267–270; MR 80f:03021] has shown that the variables of a lambda expression can be eliminated by translating the expression into one that involves combinators from some fixed finite set of combinators. In the worst case Turner's translation can produce a quadratic increase in the size of an

expression. In the present paper the authors describe a translation into "director strings" which requires $O(n \log n)$ space. A similar translation has been given by K. Noshita [Inform. Process. Lett. 20 (1985), no. 2, 71–74; MR 86e:68051]. However, both translations require $O(n^2)$ time. The authors combine known methods to show that this time bound can be reduced to $O(n \log n)$, thus yielding a translation algorithm that requires $O(n \log n)$ space and time. Reviewed by Dorothy Bollman.

Meijer, E. (1987). Petri net models for the λ-calculus. In *Advances in Petri nets 1987 (Oxford, 1986)*, pages 162–180. Springer, Berlin.
[MR911944] This item will not be reviewed individually. For details of the collection in which this item appears see 88f:68001.

Meira, S. R. d. L. (1987). Strict combinators. *Inform. Process. Lett.*, 24(4):255–258.
[MR882238] There will be no review of this item.

Zaionc, M. (1987). The regular expression descriptions of unifier set in the typed λ-calculus. *Fund. Inform.*, 10(3):309–322.
[MR89j:68105] Summary: "A λ-language over a simple type structure is considered. We investigate a unification problem such that the number of elementary substitutions in a potentially infinite matching tree introduced by G. Huet is finite [cf. Theoret. Comput. Sci. 1 (1975), no. 1, 27–57; MR 54 #6583]. The substitutions are generated using a term grammar technique. Such a grammar consists of productions which are of the form that a nonterminal variable of a certain type produces a term of the same type. The most interesting is the case when the matching tree is infinite and includes an infinite number of most general unifiers. We show an interesting property of the infinite unification trees from this class: for every infinite branch there is a node which occurs earlier in the tree. This means that there are some fragments which are repeated in the tree. Therefore the set of most general unifiers which are represented by terminal nodes can be described by a regular expression over a finite alphabet. Regular expressions are constructed as a solution of linear language equalities. For every problem from this class the set of unifiers is described by some language. This language can be approximated by regular languages."

Bellia, M. (1988). Logic and functional programming by retractions: operational semantics. *RAIRO Inform. Théor. Appl.*, 22(4):395–436.
[MR90g:68082] Summary: "Retractions are used to model predicates in such a way that with each predicate defined by a Horn theory we associate a retraction of a set theory built according to the structure of the Herbrand universe, HU_c. The set theory permits a set-theoretic interpretation of the Herbrand terms and supplies them with a combinatory formulation in terms of so-called constant expressions.

"In the present paper we discuss normal forms for constant expressions and define a system of rewriting rules which reduces constant expressions to normal forms. This set of rewriting rules, together with the rules for α, β, and Y reduction of lambda terms, also forms an operational semantics of our calculus with retractions. A reduction system is finally considered in order to investigate relations between inference on logic formulas and reduction on this kind of combinatory forms."

Boom, H. J. (1988). Lazy variable-renumbering makes substitution cheap. *Inform. Process. Lett.*, 29(5):229–232.

[MR89m:68083] In actual calculation with repeated substitutions of terms for variables, straightforward substitution algorithms have unreasonably high cost. The author suggests using the dynamic renaming of variables during substitution to reduce the space to a quadratic in the number of vertices of the expression dag (directed acyclic graph). He suggests that this technique is particularly useful for the "repeated complex substitutions that occur in type-theoretical calculations". Reviewed by David B. Benson.

Mishra, P. (1988). Strictness analysis of the untyped λ-calculus. *Inform. Process. Lett.*, 28(3):121–125.

[MR89h:68103] Summary: "We provide a precise characterization of the strictness properties of the untyped lambda-calculus. Using the notion of head evaluation to model lazy evaluation, we give a definition of strictness appropriate for the lambda-calculus. We establish that strictness is a necessary and sufficient condition for the 'eager' evaluation of function arguments. We describe an algorithm for computing strictness properties of convergent terms. We show that the natural classification of if-then-else as strict in its first and, jointly, strict in its second and third arguments is possible only in an applied lambda-calculus that either includes constants or has some form of type structure."

Révész, G. E. (1988). Rule-based semantics for an extended lambda-calculus. In *Mathematical foundations of programming language semantics (New Orleans, LA, 1987)*, pages 43–56. Springer, Berlin.

[MR89i:68092] The λ-calculus is extended to include reduction rules that manipulate lists. One of the consequences of this "vectorization" of the λ-calculus is that mutual recursion can be resolved in terms of an expression with a single occurrence of the Y combinator. The author introduces a "controlled" reduction in order to define an operational semantics and states that he has developed an interpreter program for his extended system by using a direct implemention of his reduction rules. For the entire collection see MR 89a:68005. Reviewed by Dorothy Bollman.

Schmidt, D. A. (1988). Detecting stack-based environments in denotational definitions. *Sci. Comput. Programming*, 11(2):107–131.

[MR90e:68069] The author presents a new syntactic criterion, called stack single-threading, that ensures that the arguments of a given type can be implemented as a global stack variable during the reduction of a lambda-expression. The intended application is to improve the performance of code produced by a compiler generated from the denotational definition of a programming language. The author describes how this optimisation may be used in conjunction with some known techniques, such as single-threading and annotations distinguishing run-time from compile-time domains. A (complex) proof is sketched in the appendix. Reviewed by Pierre-Yves Schobbens.

Cardone, F. (1989). Relational semantics for recursive types and bounded quantification. In *Automata, languages and programming (Stresa,* *1989)*, pages 164–178. Springer, Berlin.

[MR1037051] This item will not be reviewed individually. For details of the collection in which this item appears see 90m:68001.

Piperno, A. (1989). Abstraction problems in combinatory logic: a compositive approach. *Theoret. Comput. Sci.*, 66(1):27–43.

[MR91b:03031] The problem of defining the abstraction operator in combinatory logic has attracted attention from computer scientists in recent years because such a definition is necessary in order to implement λ-calculus in terms of combinators, and λ-calculus is the mathematical prototype of functional languages. Among those who have proposed abstraction algorithms more efficient than the ones traditional in combinatory logic are S. K. Abdali [J. Symbolic Logic 41 (1976), no. 1, 222–224; MR 53 #12888], D. A. Turner [ibid. 44 (1979), no. 2, 267–270; MR 80f:03021], M. Lemaitre et al. [in 1986 ACM Symposium on LISP and Functional Programming, 113–121; per bibl.], and R. Statman [in Proceedings of the Symposium on Logic in Computer Science (Cambridge, MA, 1986), 378–382, IEEE, Washington, DC, 1986; CCA 1987:39780].

The author proposes an abstraction algorithm for combinatory logic that can be directly implemented (although a potentially infinite basis of combinators is employed, each basic combinator depends on at most two parameters); gives compact code (it introduces a number of basic combinators which is proportional to the size of the expression to be abstracted and invariant for one- and multi-sweep abstraction techniques); gives the result in the form $\mathbf{RI}M_1M_2\cdots M_n$, where $\mathbf{R}$ is a regular combinator expressed as a composition of basic combinators, $\mathbf{I}$ is the identity combinator, and $M_1, M_2, \ldots, M_n$ are the constant terms appearing in the expression to be abstracted (thus giving the resulting code a "structured" look); and can be modified in a natural way to obtain a combinatory equivalent of R. J. M. Hughes' supercombinatorics [in 1982 ACM Symposium on LISP and Functional Programming (Pittsburgh, PA, 1982), 1–10, ACM, New York, 1982].

The author's algorithm is, in fact, a modification of the first definition of abstraction given by H. B. Curry [Ann. of Math. (2) 34 (1933), 381–404; Zbl 7, 194]. Reviewed by J. P. Seldin.

Renardel de Lavalette, G. R. (1989). Modularisation, parametrisation, interpolation. *J. Inform. Process. Cybernet.*, 25(5-6):283–292.

[MR1018762] There will be no review of this item.

Wraith, G. C. (1989). A note on categorical datatypes. In *Category theory and computer science (Manchester, 1989)*, pages 118–127. Springer, Berlin.

[MR91a:68196] Summary: "It is shown how T. Hagino's categorical datatypes can be expressed in the polymorphic typed λ-calculus. This gives a way of passing from a description of a datatype in terms of its universal properties to a representation in terms of λ-expressions." For the entire collection see MR 90m:68002.

Zaionc, M. (1989a). How to define functionals on free structures in typed λ calculus. In *Mathematical foundations of computer science 1989 (Porabka-Kozubnik, 1989)*, pages 566–576. Springer, Berlin.

[MR91f:03026] From the introduction: "The problem is to characterize λ-definable mappings in any free algebra. It is proved that the set of λ-definable operations is the minimal set containing the constant functions and projections, and closed under composition and limited recursion." For the entire collection see MR 90m:68009.

Zaionc, M. (1989b). On the λ definable higher order Boolean operations. *Fund. Inform.*, 12(2):181–189.
[MR91f:03027] The higher-order Boolean operations are defined thus: those of type 0 are the truth-values 0 and 1, and those of type $\tau \to$ are the functions which accept inputs of type τ and produce outputs of type . The author points out that for each τ there are only a finite number of Boolean operations f of type τ because there are only two of type 0; hence for each such f we can list its domain and define f by cases using IF-THEN-ELSE.

The author defines λ-definability in a typed λ-calculus and shows that the IF-THEN-ELSE operator on any finite domain with a λ-definable equality test is λ-definable. He then proves that all higher-order Boolean operators are λ-definable in typed λ-calculus. Reviewed by R. Hindley.

Cosmadakis, S. S., Meyer, A. R., and Riecke, J. G. (1990). Completeness for typed lazy inequalities (preliminary report). In *Fifth Annual IEEE Symposium on Logic in Computer Science (Philadelphia, PA, 1990)*, pages 312–320. IEEE Comput. Soc. Press, Los Alamitos, CA.
[MR1099184] This item will not be reviewed individually. For details of the collection in which this item appears see 92b:68007.

Guzmán, J. C. and Hudak, P. (1990). Single-threaded polymorphic lambda calculus. In *Fifth Annual IEEE Symposium on Logic in Computer Science (Philadelphia, PA, 1990)*, pages 333–343. IEEE Comput. Soc. Press, Los Alamitos, CA.
[MR1099186] This item will not be reviewed individually. For details of the collection in which this item appears see 92b:68007.

Holt, C. M. (1990). Intervals as time lattices. In *Real-time systems: theory and applications (York, 1989)*, pages 63–79. North-Holland, Amsterdam.
[MR1103838] This item will not be reviewed individually. For details of the collection in which this item appears see 91k:68018.

Milner, R. (1990). Functions as processes. In *Automata, languages and programming (Coventry, 1990)*, pages 167–180. Springer, New York.
[MR1076820] This item will not be reviewed individually. For details of the collection in which this item appears see 91e:68004.

Parigot, M. (1990). On the representation of data in lambda-calculus. In *CSL '89 (Kaiserslautern, 1989)*, pages 309–321. Springer, Berlin.
[MR1079086] This item will not be reviewed individually. For details of the collection in which this item appears see 91g:03006.

Robinson, E. and Rosolini, G. (1990). Polymorphism, set theory, and call-by-value. In *Fifth Annual IEEE Symposium on Logic in Computer Science (Philadelphia, PA, 1990)*, pages 12–18. IEEE Comput. Soc. Press, Los Alamitos, CA.

[MR1099156] This item will not be reviewed individually. For details of the collection in which this item appears see 92b:68007.

Wand, M. and Wang, Z. Y. (1990). Conditional lambda-theories and the verification of static properties of programs. In *Fifth Annual IEEE Symposium on Logic in Computer Science (Philadelphia, PA, 1990)*, pages 321–332. IEEE Comput. Soc. Press, Los Alamitos, CA.

[MR1099185] This item will not be reviewed individually. For details of the collection in which this item appears see 92b:68007.

Zaionc, M. (1990). On the λ-definable tree operations. In *Algebraic logic and universal algebra in computer science (Ames, IA, 1988)*, pages 279–292. Springer, Berlin.

[MR92d:03009] This paper answers the question of which functions on binary trees are definable by terms in the typed lambda calculus. The closed terms of the type $(0 \to (0 \to 0)) \to (0 \to 0)$ are isomorphic to the set T of (unlabelled) binary trees. The unique (up to $\beta\eta$ equivalence) term representing a tree t is denoted $\underline{t}$. A function $h\colon T^n \to T$ is represented by a closed term H if for all $t_1, \cdots, t_n \in T$, $H\underline{t_1}\cdots\underline{t_n} =_{\beta\eta} \underline{h(t_1, \cdots, t_n)}$. If such an H exists for a given h, then h is λ-definable. The first result on typed λ-definability is due to H. Schwichtenberg [Arch. Math. Logik Grundlag. 17 (1975/76), no. 3-4, 113–114; MR 54 #4928], for the natural number type $(0 \to 0) \to (0 \to 0)$. The author has previously characterized the λ-definable k-ary functions on strings (i.e., type $(0 \to 0) \to ((0 \to 0) \to (0 \to 0))$) [Theoret. Comput. Sci. 52 (1987), no. 1-2, 1–14; MR 89d:68016]. The well-written paper under review provides the answer for trees, where the class of λ-definable functions is the minimal class containing the joining of trees, the empty tree and the projection functions that is closed under composition and a restricted form of primitive recursion. For the entire collection see MR 91g:03005. Reviewed by John Pedersen.

Aczel, P., Carlisle, D. P., and Mendler, N. (1991). Two frameworks of theories and their implementation in Isabelle. In *Logical frameworks (Sophia-Antipolis, 1990)*, pages 3–39. Cambridge Univ. Press, Cambridge.

[MR1139779] This item will not be reviewed individually. For details of the collection in which this item appears see 92f:68003.

Amadio, R. M. (1991). Typed equivalence, type assignment, and type containment. In *Conditional and typed rewriting systems (Montreal, PQ, 1990)*, pages 372–382. Springer, Berlin.

[**MR1138504**] This item will not be reviewed individually. For details of the collection in which this item appears see 92g:68076.

Curien, P.-L. (1991). An abstract framework for environment machines. *Theoret. Comput. Sci.*, 82(2 Logic Semantics Theory Program.):389–402.

[**MR92c:68042**] Summary: "We present a nondeterministic calculus of closures for the evaluation of λ-calculus, which is based, not on symbolic evaluation (β-reduction), but on the paradigm of environments and closures, as in the old SECD machine, or in the more recent categorical abstract machine (CAM). This calculus stands as a suitable abstraction right above those devices: there is no commitment as to the order of evaluation, and no explicit handling of stacks; the calculus is expressed in the style of structural operational semantics, by a set of conditional rewrite rules. The CAM, and a very simple lazy abstract machine, due to J.-L. Krivine, arise naturally by first restricting the calculus to a specific strategy, and then implementing recursive calls by a stack. The Church-Rosser property and termination in the presence of types hold in the calculus of closures, and are easier to establish than in the λ-calculus. The calculus of closures has served as a starting point to a more powerful general calculus of substitutions at which we briefly hint, insisting on its category-theoretic significance."

de Bruijn, N. G. (1991). A plea for weaker frameworks. In *Logical frameworks (Sophia-Antipolis, 1990)*, pages 40–67. Cambridge Univ. Press, Cambridge.

[**MR1139780**] This item will not be reviewed individually. For details of the collection in which this item appears see 92f:68003.

de Groote, P. (1991). Nederpelt's calculus extended with a notion of context as a logical framework. In *Logical frameworks (Sophia-Antipolis, 1990)*, pages 69–86. Cambridge Univ. Press, Cambridge.

[**MR1139781**] This item will not be reviewed individually. For details of the collection in which this item appears see 92f:68003.

Egidi, L., Honsell, F., and Ronchi Della Rocca, S. (1991). The lazy call-by-value λ-calculus. In *Mathematical foundations of computer science, 1991 (Kazimierz Dolny, 1991)*, pages 161–169. Springer, Berlin.

[**MR1135357**] This item will not be reviewed individually. For details of the collection in which this item appears see 92g:68008.

Frandsen, G. S. and Sturtivant, C. (1991). What is an efficient implementation of the λ-calculus? In *Functional programming languages and computer architecture (Cambridge, MA, 1991)*, pages 289–312. Springer, Berlin.

[**MR92k:68053**] Summary: "We propose to measure the efficiency of any implementation of the λ-calculus as a function of a new parameter ν that is itself a function of any λ-expression. Complexity is expressed here as a function of ν just as runtime

is expressed as a function of the input size n in ordinary analysis of algorithms. This enables implementations to be compared for worst case efficiency. We argue that any implementation must have complexity $\Omega(\nu)$, i.e. a linear lower bound. Furthermore, we show that implementations based upon Turner combinators or Hughes super-combinators have complexities $2^{\Omega(\nu)}$, i.e. an exponential lower bound. It is open whether any implementation of polynomial complexity, $\nu^{O(1)}$, exists, although some implementations have been implicitly claimed to have this complexity." For the entire collection see MR 92h:68006.

Krivine, J.-L. (1991). Lambda-calcul, évaluation paresseuse et mise en mémoire. *RAIRO Inform. Théor. Appl.*, 25(1):67–84.

[MR93a:68084] In λ-calculus, leftmost-outermost (or head) reduction (which implements a call-by-name evaluation strategy) has good mathematical properties (in particular, it reaches a normal form whenever one exists) but leads to inefficient computations since each argument of a function must be re-evaluated each time it is used. A call-by-value evaluation strategy, on the other hand, is more efficient but it always evaluates (once) each argument of a function F, even if the argument is never used by F. This can also lead to nontermination in cases in which head reduction reaches a normal form.

The notion of storage operator introduced in this paper allows one to simulate a call-by-value evaluation strategy using head reduction. If θ is a term of a given particular data type D, a storage operator for D is a term T such that $(T\theta) > \lambda f.(f\tau)$ where τ is a standard representation of θ (not necessarily its normal form but in general a form in which θ has been essentially evaluated) and $>$ denotes head reduction. Then if F is a term which represents a function over D, $(T\theta F) > (F\tau)$, i.e., F is applied to an argument which is essentially evaluated, as in the call-by-value strategy. The trade-off of this simulation is that, if the evaluation of θ (in head reduction) does not terminate, the evaluation of $T\theta$ (and then that of $(T\theta F)$) could also not terminate.

Storage operators for the data types of Booleans and (Church) numerals are derived in the paper. It is also shown how to define storage operators for data types obtained as Cartesian products and disjoint sums from the storage operators of their components.

A general characterization of storage operators is given introducing a type inference system of λ-terms which corresponds, in the Curry-Howard isomorphism, to the intuitionistic second-order logic. In this system a data type can be naturally characterized by a formula $D[x]$ which corresponds to its inductive definition. It turns out that a storage operator for a data type $D[x]$ is a (closed) term of the type $\forall x.D^*[x] \to D[x]$ where $D^*[x]$ is the "Gödel translation" of the formula $D[x]$. This result is only conjectured in the paper (where it is shown for some particular cases) and has subsequently been proved, in a more general setting, by the author [91j:03011Arch. Math. Logic 30 (1990), no. 4, 241–267; MR 91j:03011]. This paper can be seen as an updated version of the present one. Reviewed by M. Coppo.

Postma, S. W. (1991). The $\sigma\lambda$-calculus and derived program forms. *Quaestiones Math.*, 14(2):137–159.

[MR92j:03013] Summary: "The $\Sigma\lambda$-calculus is an extension of an applied λ-calculus designed to abstract parallel and nondeterministic function application evaluation. The abstraction is extended to include P. J. Landin's closure, and constant

terms and a special form of the conditional allow the regular logics to be developed along the lines of the Hilbert-Ackermann approach. The regular logics are used to characterize derived conditionals in extensions to E. W. Dijkstra's weakest preconditions. It is shown that various styles of programming—top-down, bottom-up, recursive and iterative—are expressible in the $\Sigma\lambda$-calculus. Lists and streams are found to be useful in expressing time-dependent behaviour of computational systems. An approximate Church-Rosser theorem is given."

Qian, Z. (1991). Second-order unification in the presence of linear shallow algebraic equations (extended abstract). In *Conditional and typed rewriting systems (Montreal, PQ, 1990)*, pages 448–453. Springer, Berlin.
[**MR1138512**] This item will not be reviewed individually. For details of the collection in which this item appears see 92g:68076.

Sato, M. (1991). An abstraction mechanism for symbolic expressions. In *Artificial intelligence and mathematical theory of computation*, pages 381–391. Academic Press, Boston, MA.
[**MR1136058**] This item will not be reviewed individually. For details of the collection in which this item appears see 92f:68002.

Wand, M. and O'Keefe, P. M. (1991). Automatic dimensional inference. In *Computational logic*, pages 479–483. MIT Press, Cambridge, MA.
[**MR1132712**] This item will not be reviewed individually. For details of the collection in which this item appears see 92d:03002.

de'Liguoro, U. and Piperno, A. (1992). Must preorder in nondeterministic untyped λ-calculus. In *CAAP '92 (Rennes, 1992)*, pages 203–220. Springer, Berlin.
[**MR94h:03035**] Summary: "This paper studies the interplay between functional application and nondeterministic choice in the context of untyped λ-calculus. We introduce an operational semantics which is based on the idea of a 'must' preorder, coming from the theory of process algebras. To characterize this relation, we build a model using the classical inverse limit construction, and we prove that it is fully abstract by using a generalization of Böhm trees." For the entire collection see MR 94f:68009.

Meinke, K. (1992). Equational specification of abstract types and combinators. In *Computer science logic (Berne, 1991)*, pages 257–271. Springer, Berlin.
[**MR1232892**] This item will not be reviewed individually. For details of the collection in which this item appears see 94c:68001.

Milner, R. (1992). Functions as processes. *Math. Structures Comput. Sci.*, 2(2):119–141. International Conference on Symbolic Computation (Zurich, 1990).
[**MR93i:03020**]

The author's purpose in writing this paper is to show how two different reduction strategies in the λ-calculus can be simulated in his π-calculus, a calculus in which the terms represent processes which are allowed to communicate by passing the names of communication channels (and hence access to those channels) from one to another. Thus the paper provides two different ways of viewing functions as processes. In addition, the author demonstrates that for both translations, putting the λ-terms into concurrent contexts allows a strictly sharper form of observational distinction than is obtained by using only functional contexts.

In addition, the paper provides an excellent introduction to the π-calculus. This is likely to be particularly useful for readers who are not members of the concurrency community, if only because it is backed up by extended examples which, refreshingly, are not the usual communication protocols.

Reviewer's remarks: The material presented here has now been developed further. One direction has been the work of Sangiorgi, mentioned in the paper, on identifying the pre-orders on λ-terms induced by pre-orders on the π-calculus. Another has been the investigation of different translations. For example, the similarity between the transfer of a name in the π-calculus and the switching of a pointer has led Jeffrey to look at the implementation of graph reduction. For the entire collection see MR 93a:68007. Reviewed by Edmund Robinson.

Mirasyedioğlu, Ş. (1992). An efficient interpreter for the lambda-Boolean theory. *Doğa Mat.*, 16(3):218–229.

[MR94b:03028] This paper describes a translation from λB-terms (a subset of λ-terms representing Boolean functions) into lists. The author defines an interpreter for reducing λB-terms by working on their list representation. The overall impression is that this paper is mainly a programming exercise. Reviewed by Eugenio Moggi.

Mogensen, T. Æ. (1992). Efficient self-interpretation in lambda calculus. *J. Funct. Programming*, 2(3):345–363.

[MR93j:68121] Summary: "We start by giving a compact representation schema for λ-terms, and show how this leads to an exceedingly small and elegant self-interpreter. We then define the notion of a self-reducer, and show how this too can be written as a small λ-term. Both the self-interpreter and the self-reducer are proved correct. We finally give a constructive proof for the second fixed-point theorem for the representation schema. All the constructions have been implemented on a computer, and experiments verify their correctness. Timings show that the self-interpreter and self-reducer are quite efficient, being about 35 and 50 times slower than direct execution using a call-by-need reductions strategy."

Palsberg, J. and Schwartzbach, M. I. (1992). Safety analysis versus type inference for partial types. *Inform. Process. Lett.*, 43(4):175–180.

[MR93g:68080] Summary: "Safety analysis is an algorithm for determining if a term in an untyped lambda calculus with constants is safe, i.e., if it does not cause an error during evaluation. We prove that safety analysis accepts strictly more safe lambda terms than does type inference for Thatte's partial types."

Roversi, L. (1992). A compiler from Curry-typed-λ-terms to linear-λ-terms. In *Theoretical computer science (L'Aquila, 1992)*, pages 330–344. World Sci. Publishing, River Edge, NJ.

[**MR94j:03037**] Summary: "Given the λ-terms admitting Curry's types as source language and considering the implicative fragment of the linear-λ-terms as object language, we define a compiler from the first to the second language that is driven exclusively by the structure of the proof which gives a type to a source program. The object (linear) program obtained can be evaluated by an extensive application of an eager-evaluation strategy." For the entire collection see MR 94f:68034.

Aït-Kaci, H. and Garrigue, J. (1993). Label-selective λ-calculus syntax and confluence. In *Foundations of software technology and theoretical computer science (Bombay, 1993)*, pages 24–40. Springer, Berlin.

[**MR95e:03043**] Summary: "We introduce an extension of λ-calculus, called label-selective λ-calculus, in which arguments of functions are selected by labels. The set of labels includes numeric positions as well as symbolic keywords. While the latter enjoy free commutation, the former must comply with relative precedence in order to preserve currying. This extension of λ-calculus is conservative in the sense that when the set of labels is the singleton $\{1\}$, it coincides with λ-calculus. The main result of this paper is that the label-selective λ-calculus is confluent. In other words, argument selection and reduction commute." For the entire collection see MR 95d:68003.

Amadio, R. M. (1993). On the adequacy of per models. In *Mathematical foundations of computer science 1993 (Gdańsk, 1993)*, pages 222–231. Springer, Berlin.

[**MR94j:68173**] Summary: "We consider a fixed-point extension of the second-order lambda calculus equipped with a call-by-value evaluation mechanism. We interpret the language in a partial Cartesian closed category of 'directed complete' partial equivalence relations (pers) over a domain theoretic model of a type-free, call-by-value, lambda calculus. Our main result is that the notions of 'syntactic' and 'semantic' convergence coincide." For the entire collection see MR 94i:68008.

Berardi, S. (1993). An application of PER models to program extraction. *Math. Structures Comput. Sci.*, 3(3):309–331.

[**MR95b:68067**] The paper treats the extraction of programs out of a proof in some constructive type theory. The author addresses especially the task of optimizing the extracted program by deleting "purely logical code" and only preserving "code with computational meaning". He shows how a PER model semantics can be used to show extensional equality of the optimized program and the original program extracted from the proof. The usual correctness result for program optimization by using realizability semantics can be obtained as a corollary.

In the paper the author uses the typed λ-calculus CCΣN (calculus of constructions with predicative strong sums and primitive integers) but he claims that the obtained results are independent of the chosen λ-calculus. The paper is self-contained in the sense that there is no familiarity assumed with program optimization. However, the reader should be familiar with higher order typed λ-calculi, especially with the calculus of constructions by Coquand and Huet.

In the introduction the author explains the problem domain and outlines the paper. Section 2 is about CCΣN; Section 3 describes proof and program extraction in CCΣN. Section 4 addresses the formalization of "redundant" code and its erasure from a program. Section 5 gives a PER (partial equivalence relation) model for $F_\omega \Sigma N$ (ω-order λ-calculus), which is a subsystem of CCΣN, namely CCΣN without dependent types. The last section states the main theorem of the paper, namely, the extensional equivalence of the optimized program and the original one. In the appendix he proves a type soundness theorem for the PER semantics given in Section 5. Reviewed by Manfred Broy.

Berger, U. (1993). Program extraction from normalization proofs. In *Typed lambda calculi and applications (Utrecht, 1993)*, pages 91–106. Springer, Berlin.

[**MR94c:68026**] Summary: "From a constructive proof of strong normalization plus uniqueness of the long beta-normal form for the simply typed lambda calculus, a normalization program is extracted using Kreisel's modified realizability interpretation for intuitionistic logic. The proof—which uses Tait's computability predicates—is not completely formalized: Induction is done on the meta level, and therefore not a single program but a family of programs, one for each term, is obtained. Nevertheless this may be used to write a short and efficient normalization program in a type free programming functional programming language (e.g. LISP) which has the interesting feature that it first evaluates a term r of type ρ to a functional $|r|$ of type ρ and then collapses $|r|$ to the normal form of r." For the entire collection see MR 94b:68025.

Castagna, G., Ghelli, G., and Longo, G. (1993). A semantics for λ&-early: a calculus with overloading and early binding (extended abstract). In *Typed lambda calculi and applications (Utrecht, 1993)*, pages 107–123. Springer, Berlin.

[**MR1233143**] This item will not be reviewed individually. For details of the collection in which this item appears see 94b:68025.

de Bruijn, N. G. (1993). Algorithmic definition of lambda-typed lambda calculus. In *Logical environments (Edinburgh, 1991)*, pages 131–146. Cambridge Univ. Press, Cambridge.

[**MR1255112**] This item will not be reviewed individually. For details of the collection in which this item appears see 94f:03004.

Diskin, Z. and Beylin, I. (1993). Lambda substitution algebras. In *Mathematical foundations of computer science 1993 (Gdańsk, 1993)*, pages 423–432. Springer, Berlin.

[**MR1265080**] This item will not be reviewed.

Garrigue, J. (1993). The transformation calculus and its typing. *Sūrikaisekikenkyūsho Kōkyūroku*, (851):34–45. Type theory and its applications to computer systems (Japanese) (Kyoto, 1993).

[**MR1304523**] This item will not be reviewed individually. For details of the collection in which this item appears see 95f:68011.

Honda, K. (1993). Types for dyadic interaction. *Sūrikaisekikenkyūsho Kōkyūroku*, (851):61–77. Type theory and its applications to computer systems (Japanese) (Kyoto, 1993).

[**MR95i:68042**] Summary: "We formulate a typed formalism for concurrency where types denote freely composable structures of dyadic interaction in the symmetric scheme. The resulting calculus is a typed reconstruction of name passing process calculi. Systems with both the explicit and implicit typing disciplines, where types form a simple hierarchy of types, are presented, which are proved to be in accordance with each other. A typed variant of bisimilarity is formulated and it is shown that typed β-equality has a clean embedding in the bisimilarity. Name reference structure induced by the simple hierarchy of types is studied, which fully characterises the typable terms in the set of untyped terms. It turns out that the name reference structure results in the deadlock-free property for a subset of terms with a certain regular structure, showing behavioural significance of the simple type discipline." For the entire collection see MR 95f:68011.

Honsell, F. and Lenisa, M. (1993). Some results on the full abstraction problem for restricted lambda calculi. In *Mathematical foundations of computer science 1993 (Gdańsk, 1993)*, pages 84–104. Springer, Berlin. [**MR95c:03036**]

This paper deals with the full abstraction problem for Church's λI-calculus and Plotkin's λv-calculus with respect to their natural evaluation machines (left normalisation and SECD evaluation, respectively).

For the λI-calculus a simple solution of the domain equation $D \simeq [D \to_\perp D]$, where $[D \to_\perp D]$ is the space of strict continuous functions from D to D, is shown to be fully abstract; the proof of correctness, which is the difficult part, is done by showing equivalence among interpretability in D, typability in the associated type system S_D, and a realisability notion linked to the evaluation mechanism, here in Krivine's style. The exposition is especially clear and, as by-products, one gets new and semantic proofs that (i) in the λI-calculus weak and strong normalization coincide; (ii) the extensional λI-theory where all terms without normal form are equated is consistent.

For λv-calculus the natural domain equation is $D \simeq [D \to_\perp D]_\perp$, where the function space is lifted. Because of the "parallel or" there exist no "continuous" fully abstract models, and the authors investigate the "stable" semantics, where they are faced with two technical problems: first the nonstability of the natural "strict" operator on functions, and second the nowhere definability of the "lift" operator on coherent spaces. This is solved within a variant of the coherent semantics dealing with partial stable functions. Not surprisingly, the model is not fully abstract with respect to λv and SECD; but its correctness allows the authors to give a semantic proof of the SECD-operational equivalence of some λv- terms. For the entire collection see MR 94i:68008. Reviewed by Chantal Berline.

Matthews, S., Smaill, A., and Basin, D. (1993). Experience with FS_0 as a framework theory. In *Logical environments (Edinburgh, 1991)*, pages

61–82. Cambridge Univ. Press, Cambridge.
[**MR1255110**] This item will not be reviewed individually. For details of the collection in which this item appears see 94f:03004.

Milner, R. (1993). The polyadic π-calculus: a tutorial. In *Logic and algebra of specification (Marktoberdorf, 1991)*, pages 203–246. Springer, Berlin.
[**MR94j:68066**] Summary: "The π-calculus is a model of concurrent computation based upon the notion of naming. It is first presented in its simplest and original form, with the help of several illustrative applications. Then it is generalized from monadic to polydic form. Semantics is done in terms of both a reduction system and a version of labelled transitions called commitment; the known algebraic axiomatization of strong bisimilarity is given in the new setting, and so also is a characterization in modal logic. Some theorems about the replication operator are proved.

"Justification for the polyadic form is provided by the concepts of sort and sorting which it supports. Several illustrations of different sortings are given. One example is the presentation of data structures as processes which respect a particular sorting; another is the sorting for a known translation of the λ-calculus into π-calculus. For this translation, the equational validity of β-conversion is proved with the help of replication theorems. The paper ends with an extension of the π-calculus to ω-order processes, and a brief account of the demonstration by D. Sangiorgi [Ph.D. Thesis, Univ. Edinburgh, Edinburgh, 1992; per bibl.] that higher-order processes may be faithfully encoded at first order. This extends and strengthens the original result of this kind given by B. Thomsen ["Calculi for higher-order communicating stystems", Ph.D. Thesis, Imperial College, Univ. London, London 1990; per bibl.] for second-order processes." For the entire collection see MR 94c:68003.

Nour, K. (1993). Opérateurs propres de mise en mémoire. *C. R. Acad. Sci. Paris Sér. I Math.*, 317(1):1–6.
[**MR94c:68122**] Summary: "Storage operators were introduced two years ago by J.-L. Krivine, they are closed λ-terms which, for some fixed data type (for example integers), allow one to simulate call by value while using call by name. J.-L. Krivine showed such operators can be typed, in the type system $\mathcal{AF}2$, using Gödel's translation from classical to intuitionistic logic. In this note we study the problem of typing those storage operators which give a closed term as result (we call them proper storage operators). We give restrictive conditions on the type system $\mathcal{AF}2$ so as to obtain proper operators. This study will be restricted to the case of Church integers."

Palsberg, J. (1993). Correctness of binding-time analysis. *J. Funct. Programming*, 3(3):347–363.
[**MR94k:68133**] Summary: "A binding-time analysis is correct if it always produces consistent binding-time information. Consistency prevents partial evaluators from 'going wrong'. A sufficient and decidable condition for consistency, called well-annotatedness, was first presented by Gomard and Jones. In this paper we prove that a weaker condition implies consistency. Our condition is decidable, subsumes the one of Gomard and Jones, and was first studied by Schwartzbach and the present author.

Our result implies the correctness of the binding-time analysis of Mogensen, and it indicates the correctness of the core of the binding-time analyses of Bondorf and Consel. We also prove that all partial evaluators will on termination have eliminated all 'eliminable'-marked parts of an input which satisfies our condition. This generalizes a result of Gomard. Our development is for the pure λ-calculus with explicit binding-time annotations."

Parigot, M. and Rozière, P. (1993). Constant time reductions in λ-calculus. In *Mathematical foundations of computer science 1993 (Gdańsk, 1993)*, pages 608–617. Springer, Berlin.

[**MR95c:68071**] The paper under review presents incremental completeness results in pure lambda-calculus, with respect to different representations of data.

It is well known that all recursive functions are representable by pure λ-terms, and that there exist (extensional) completeness results with respect to various typed λ-calculi, but this paper is one of the first theoretical ones dealing with the intentional question of the existence of programs having an expected behaviour in terms of efficiency.

The authors analyse the inherent complexity of three basic representations of natural numbers: the inductive one (Church integers), the recursive one (a variant of Scott's numerals), and a mixed version. These three representations correspond to three natural ways of defining N in second order predicate calculus. A typical result is the following theorem (where $\hat{n}$ denotes the nth Church numeral). For any function f from N to $\mathbf{N}$, the following are equivalent: (1) There are m, a, b in N such that for all $n \geq m$, $f(n) = an + b$. (2) There is a term t which represents f and there is m in N such that the computation of $(t)\widehat{m}$ takes $< m$ reduction steps. (3) There is a term t which represents f and there is c such that for all n in N, the computation of $(t)\hat{n}$ takes $\leq c$ reduction steps.

The method of the paper can be applied to prove similar results for data types other than unary natural numbers; moreover the proofs are very elegant, even if some more technicalities are needed to get the complete result in the iterative case. For the entire collection see MR 94i:68008. Reviewed by Chantal Berline.

Paulin-Mohring, C. (1993). Inductive definitions in the system Coq; rules and properties. In *Typed lambda calculi and applications (Utrecht, 1993)*, pages 328–345. Springer, Berlin.

[**MR94e:68111**] Summary: "In the pure calculus of constructions, it is possible to represent data structures and predicates using higher-order quantification. However, this representation is not satisfactory from the point of view of both the efficiency of the underlying programs and the power of the logical system. For these reasons, the calculus was extended with a primitive notion of inductive definitions. This paper describes the rules for inductive definitions in the system Coq. They are general enough to be seen as one formulation of adding inductive definitions to a typed lambda-calculus. We prove strong normalization for a subsystem of Coq corresponding to the pure calculus of constructions plus inductive definitions with only weak eliminations." For the entire collection see MR 94b:68025.

Sangiorgi, D. (1993). From π-calculus to higher-order π-calculus—and back. In *TAPSOFT '93: theory and practice of software development (Orsay,* *1993)*, pages 151–166. Springer, Berlin.

[**MR94f:68136**] Summary: "We compare the first-order and the higher-order paradigms for the representation of mobility in process algebras. The prototypical calculus in the first-order paradigm is the π-calculus. By generalizing its sort mechanism we derive an ω-order extension, called higher-order π-calculus. We give examples of its use, including the encoding of λ-calculus. Surprisingly, we show that such an extension does not add expressiveness: Higher-order processes can be faithfully represented at first order. We conclude that the first-order paradigm, which enjoys a simpler and more intuitive theory, should be taken as basic. Nevertheless, the study of the λ-calculus encodings shows that a higher-order calculus can be very useful for reasoning at a more abstract level." For the entire collection see MR 94c:68008.

Schwichtenberg, H. (1993). Minimal logic for computable functions. In *Logic and algebra of specification (Marktoberdorf, 1991)*, pages 289–320. Springer, Berlin.

[**MR95a:68074**] This paper presents specification languages for higher-order functionals, and for computable functions; it also gives a framework for the formal verification of hardware and functional programs. The author considers partial continuous functions in the context of Scott's information systems. Noncontinuous functionals are also allowed, but such functionals can only be applied and can never be arguments; they are considered and called external functions. A formal system of higher-order arithmetic is described, using the domain of higher-order continuous functions as its intended model. A fragment of Gentzen's natural deduction is considered as a deduction system. The formal system of higher-order arithmetic is able to formalize constructive proofs; therefore a more explicit formulation of constructive mathematics is obtained. An implementation of this formal system is described; the implementation is written in SCHEME; the built-in evaluation mechanism of SCHEME is used to carry out the normalisation of proofs. Proofs could be used as programs. For the entire collection see MR 94c:68003. Reviewed by Gabriel Ciobanu.

Vasconcelos, V. T. (1993). A predicative polymorphic type system for a calculus of objects. *Sūrikaisekikenkyūsho Kōkyūroku*, (851):78–87. Type theory and its applications to computer systems (Japanese) (Kyoto, 1993).

[**MR1304526**] This item will not be reviewed individually. For details of the collection in which this item appears see 95f:68011.

Wand, M. (1993). Specifying the correctness of binding-time analysis. *J. Funct. Programming*, 3(3):365–387.

[**MR94j:68190**] Summary: "Mogensen has exhibited a very compact partial evaluator for the pure lambda calculus, using binding-time analysis followed by specialization. We give a correctness criterion for this partial evaluator and prove its correctness relative to this specification. We show that the conventional properties of partial evaluators, such as the Futamura projections, are consequences of this specification. By considering both a flow analysis and the transformation it justifies together, this proof suggests a framework for incorporating flow analyses into verified compilers."

Wang, M. Q. (1993). An extension of the typed λ-calculus. *J. Fuzhou Univ. Nat. Sci. Ed.*, 21(3):8–15.

[**MR1282135**] There will be no review of this item.

Barbanera, F. and Berardi, S. (1994). A symmetric lambda calculus for "classical" program extraction. In *Theoretical aspects of computer software (Sendai, 1994)*, pages 495–515. Springer, Berlin.

[**MR1296689**] This item will not be reviewed individually. For details of the collection in which this item appears see 95d:68006.

Bergstra, J. A., Bethke, I., and Ponse, A. (1994). Process algebra with combinators. In *Computer science logic (Swansea, 1993)*, pages 36–65. Springer, Berlin.

[**MR96b:68046**] Summary: "We introduce typed combinatory process algebra, a system combining process algebra with types and combinators. We describe its syntax and semantics, and by way of example, verify within this framework the simple alternating bit protocol." For the entire collection see MR 95j:68009. Reviewed by Wieslaw Zielonka.

Boudol, G. (1994). Some chemical abstract machines. In *A decade of concurrency (Noordwijkerhout, 1993)*, pages 92–123. Springer, Berlin.

[**MR95i:68040**] Summary: "This paper explains by a series of examples the use of the notion of a chemical abstract machine (CHAM). This is a framework for parallel computations which builds upon the chemical metaphor of Banatre and Le Metayer. We first illustrate the CHAM style by describing the operational semantics of process calculi constructs. Then we apply our approach to the λ-calculus, designing a CHAM for the weak β-reduction with sharing, and specializing it into a CHAM for the call-by-need reduction strategy. Our last example is the π-calculus. We discuss the notions of 'asynchronous' and 'fully asynchronous' reductions, and some π-calculus encodings of the λ-calculus."

Contents: 1. Introduction: The chemical metaphor; 2. The chemical abstract machine; 3. Semantics of some process calculi constructs; 4. The weak λ-calculus with sharing; 5. The asynchronous π-calculus; 6. Conclusion. For the entire collection see MR 95d:68002.

Ciobanu, G. (1994). On the I-models of the $\lambda\beta$-calculus. In *Romanian Symposium on Computer Science (Iaşi, 1993)*, pages 80–93. "A. I. Cuza" Univ. Iasi, Iaşi.

[**MR96h:03033**] Summary: "We described a model of λ-calculus in a previous article [An. Stiint. Univ. Al. I. Cuza Iasi Inform. (N.S.) 1 (1992), 57–67; MR 95e:68127], where we used it to give a trace semantics of dataflow computation. This model is based on the ideals of a complete partial order; we call it the I-model. In this paper we study this model. The axioms and rules of inference of λ-calculus (in fact, of

$\lambda\beta$-calculus) are sound when they are interpreted by I-models; every I-model yields an associated $\lambda\beta$-theory. The consistency of $\lambda\beta$-calculus (a purely syntactic notion) is shown to follow from the existence of a nontrivial model. An important result is a completeness theorem expressing that every $\lambda\beta$-theory is the theory associated with an I-model." For the entire collection see MR 96f:68014.

Coquand, C. (1994). From semantics to rules: a machine assisted analysis. In *Computer science logic (Swansea, 1993)*, pages 91–105. Springer, Berlin.
[MR1316901] This item will not be reviewed individually. For details of the collection in which this item appears see 95j:68009.

de Groote, P. (1994). On the relation between the $\lambda\mu$-calculus and the syntactic theory of sequential control. In *Logic programming and automated reasoning (Kiev, 1994)*, pages 31–43. Springer, Berlin.
[MR96a:03028] Summary: "We construct a translation of first-order λ-calculus [M. Parigot, in Logic programming and automated reasoning (St. Petersburg, 1992), 190–201, Lecture Notes in Comput. Sci., 624, Springer, Berlin, 1992; MR 94g:03029] into a subtheory of Felleisen's λ_c-calculus [M. Felleisen et al., Theoret. Comput. Sci. 52 (1987), no. 3, 205–237; MR 89j:68097; M. Felleisen and R. Hieb, Theoret. Comput. Sci. 103 (1992), no. 2, 235–271; MR 94e:68107]. This translation preserves typing and reduction. Then, by constructing the inverse translation, we show that the two calculi are actually isomorphic." For the entire collection see MR 95g:68002.

Dezani-Ciancaglini, M., de'Liguoro, U., and Piperno, A. (1994). Fully abstract semantics for concurrent λ-calculus (extended abstract). In *Theoretical aspects of computer software (Sendai, 1994)*, pages 16–35. Springer, Berlin.
[MR1296671] This item will not be reviewed.

Hagiya, M. and Toda, Y. (1994). On implicit arguments. In *Logic, language and computation*, pages 10–30. Springer, Berlin.
[MR95b:03011] Summary: "A typechecker for a typed λ-calculus with implicit arguments is presented. The typechecker works in such a way that the uniqueness of implicit arguments is always preserved during reduction. Consequently, when it compares two terms by reduction, it can reduce them without inferring implicit arguments. Before describing the typechecker, we analyze various situations where the uniqueness of implicit arguments is not preserved by naively defined reduction." For the entire collection see MR 95b:68015.

Krivine, J.-L. (1994). A general storage theorem for integers in call-by-name λ-calculus. *Theoret. Comput. Sci.*, 129(1):79–94.
[MR95j:68106] Summary: "The notion of storage operator which we introduced in earlier papers appears to be an important tool in the study of data types in second-order λ-calculus. These operators are λ-terms which simulate call-by-value in the call-by-name strategy, and they can be used in order to model assignment instructions.

The main result about storage operators is that there is a very simple second-order type for them, using Gödel's 'not-not translation' of classical into intuitionistic logic.

"We give here a new and simpler proof of a strengthened version of this theorem, which contains previous results of ours in intuitionistic and in classical logic (Krivine, 1990, 1992), and gives rise to new 'storage theorems'. Moreover, this result has a simple and intuitive meaning in terms of realizability." Reviewed by Zhenyu Qian.

Laneve, C. (1994). Distributive evaluations of λ-calculus. *Fund. Inform.*, 20(4):333–352.
[MR96a:68070] Summary: "In this paper we address the problem of encoding evaluation strategies for the λ-calculus into prime event structures. In order for this to be possible the derivation spaces yielded by the evaluation mechanism must be prime algebraic cpo's. This requirement is not met by permutation equivalence (the standard concurrent semantics with which λ-calculus is equipped) since the derivation spaces it yields are upper semi-lattices. We solve this problem by taking the coarsest congruence contained in permutation equivalence such that permutations of disjoint reductions are equated and the downward closure of every derivation is a distributive lattice. This equivalence, called distributive permutation equivalence, is characterized directly by restricting permutations of redexes to those sets U which are distributive, i.e. for every $u \in U$, the development of every $V \subseteq (U \setminus \{u\})$ does not duplicate or delete u.

"A simple consequence of our results is that the derivation spaces of the call-by-value λ-calculus are distributive lattices. Finally, we show that a sequential evaluation mechanism cannot, in general, be effectively transformed into a maximally distributive one." Reviewed by Chantal Berline.

Milner, R. (1994). Higher-order action calculi. In *Computer science logic (Swansea, 1993)*, pages 238–260. Springer, Berlin.
[MR95m:68112] Summary: "Action calculi are a broad class of algebraic structures, including a formulation of Petri nets as well as a formulation of the π-calculus. Each action calculus $\mathrm{HAC}(\mathcal{K})$ is generated by a particular set $\mathcal{K}$ of operators called controls. The purpose of this paper is to extend action calculi in a uniform manner to higher order. A special case is essentially the extension of the π-calculus to higher order by Sangiorgi. To establish a link between the interactive and functional paradigms of computation, a variety of the λ-calculus is obtained as the extension of the smallest action calculus $\mathrm{HAC}(\varnothing)$.

"The dynamics of higher-order action calculi is presented, blending communication—for example in process calculi—with reduction as in the λ-calculus. Strong normalisation is obtained for reduction. A set of equational axioms is given for higher-order action calculi. Taking the quotient of $\mathrm{HAC}(\varnothing)$ by a single extra axiom η, a Cartesian-closed category is obtained.

"An ultimate goal of the paper is to combine process calculi and functional calculi, both in their formulation and in their semantics." For the entire collection see MR 95j:68009.

Monteiro, E. (1994). Linear logic as CSP. *J. Logic Comput.*, 4(4):405–421.
[MR95f:68146] Summary: "The cut elimination/normalization process in a logical calculus may simulate computations. This amounts to designing a programming language on a logical calculus. Typed λ-calculus is a familiar example. Linear logic

has been used for simulating both a 'resource sensitive' computation and parallel computations. We investigate how to let linear logic simulate parallel computations. The focus is on the multiplicative fragment in the proof net formalization of linear logic. A translation from such proofs into a corresponding CSP process is offered. It is shown that the translation yields a step-by-step correspondence between the cut elimination process and the CSP execution. Generalizations and related work are discussed. We are concerned with the handling of the failure of the Church-Rosser property. We point out the two possible strategies to adopt: either restrict attention to the cases where the property does hold, or try to give an interpretation where the ordering matters."

Qian, Z. and Nipkow, T. (1994). Reduction and unification in lambda calculi with a general notion of subtype. *J. Automat. Reason.*, 12(3):389–406.
[**MR96h:03039**] Summary: "Reduction, equality, and unification are studied for a family of simply typed λ-calculi with subtypes. The subtype relation is required to relate base types only to base types and to satisfy some order-theoretic conditions. Constants are required to have a least type, that is, 'no overloading'. We define the usual β- and a subtype-dependent η-reduction. These are related to a typed equality relation and shown to be confluent in a certain sense. We present a generic algorithm for preunification modulo $\beta\eta$-conversion and an arbitrary subtype relation. Furthermore, it is shown that unification with respect to any subtype relation is universal."

Raffalli, C. (1994). Machine deduction. In *Types for proofs and programs (Nijmegen, 1993)*, pages 333–351. Springer, Berlin.
[**MR95b:68070**] Summary: "Machine deduction is a type system designed to extract code for an abstract machine from proofs. This paper presents the basic definitions and results, and shows how we can replace a compilation of typed lambda terms by a proof translation into our system." For the entire collection see MR 95b:68023.

Rehof, N. J. and Sørensen, M. H. (1994). The λ_δ-calculus. In *Theoretical aspects of computer software (Sendai, 1994)*, pages 516–542. Springer, Berlin.
[**MR1296690**] This item will not be reviewed individually. For details of the collection in which this item appears see 95d:68006.

Sangiorgi, D. (1994a). An investigation into functions as processes. In *Mathematical foundations of programming semantics (New Orleans, LA, 1993)*, pages 143–159. Springer, Berlin.
[**MR96c:68116**] Summary: "Milner examined the encoding of the λ-calculus into the π-calculus. The former is the universally accepted basis for computations with functions, the latter aims at being its counterpart for computations with processes. The primary goal of this paper is to continue the study of Milner's encodings. We focus mainly on the lazy λ-calculus. We show that its encoding gives rise to a λ-model, in which a weak form of extensionality holds. However the model is not fully

abstract: to obtain full abstraction, we examine both the restrictive approach, in which the semantic domain of processes is cut down, and the expansive approach, in which λ-calculus is enriched with constants to obtain a direct characterisation of the equivalence on λ-terms induced, via the encoding, by the behavioural equivalence adopted on the processes. Our results are derived exploiting an intermediate representation of Milner's encodings into the higher-order π-calculus, an ω-order extension of π-calculus where agents may be transmitted. For this, essential use is made of the fully abstract compilation from the higher-order π-calculus to the π-calculus studied in an earlier paper of ours ["Expressing mobility in process algebras: first-order and higher-order paradigms", CST-99-93/ECS-LFCS-93-266, Univ. Edinburgh, Edinburgh, 1992; per bibl.]." For the entire collection see MR 95i:68003.

Sangiorgi, D. (1994b). The lazy lambda calculus in a concurrency scenario. *Inform. and Comput.*, 111(1):120–153. 1992 IEEE Symposium on Logic in Computer Science.

[MR95g:03015] This interesting paper, which ends with several open problems, deals with the discrimination of pure λ-terms by means of operational considerations with respect to programming languages extending lazy λ-calculus and including possibly parallel or nondeterministic features. This is linked to the full abstraction problem, i.e., the complete adequation between the operational indiscernibility of terms and the equality of their interpretations in a given mathematical model.

Starting from lazy λ-calculus (left reduction which stops when reaching an abstraction), the full abstraction problem was solved, with respect to well-chosen "canonical domains", in a paper by S. Abramsky and C.-H. L. Ong [Inform. and Comput. 105 (1993), no. 2, 159–267; MR 94h:68124] by adding a parallel convergence testing operator, and in a paper by G. Boudol [Inform. and Comput. 108 (1994), no. 1, 51–127; MR 95d:03010] by adding general computational methods, namely call-by-value and parallel reduction mechanisms.

The aim of the author is to present solutions that are closer to common programming practice, and to make it clear whether the induced equivalences on pure terms are sensitive to the addition of more operators.

Two approaches are proposed, both of which embed λ-calculus in richer settings, and, though apparently of very different kinds, they turn out to be equivalent for discriminating pure terms. First, λ-calculus is studied within Milner's process π-calculus. Several simple characterizations of the congruence equ induced on pure terms by Milner's embedding are given, from which the full abstraction with respect to Levy-Longo trees is proved.

Second, λ-calculus is augmented by any family of well formed operators. It is shown that the discrimination between pure terms (in terms of Abramski's applicative bisimulation) is maximal when all operators are used, coincides with equ, and is already induced by the mere presence of simple nondeterministic operators, for example the unary operator which, when applied to a term, outputs this term or the always divergent term Ω. This (kind of) nondeterminism is actually necessary, in the precise sense that the same discriminating power cannot be recovered by adding only Church-Rosser operators, such as parallel convergence testing. In particular, equ is more discriminating than the bisimulations considered by Abramsky and Boudol; in other words, their canonical models are not sound for equ. For the entire collection see MR 94m:68004. Reviewed by Chantal Berline.

Stoughton, A. (1994). Mechanizing logical relations. In *Mathematical foundations of programming semantics (New Orleans, LA, 1993)*, pages 359–377. Springer, Berlin.

[**MR95j:03027**] Summary: "We give an algorithm for deciding whether there exists a definable element of a finite model of an applied typed lambda calculus that passes certain tests, in the special case when all the constants and test arguments are of order at most one. When there is such an element, the algorithm outputs a term that passes the tests; otherwise, the algorithm outputs a logical relation that demonstrates the nonexistence of such an element. Several example applications of the C implementation of this algorithm are considered." For the entire collection see MR 95i:68003.

Tsuiki, H. (1994). A normalizing calculus with overloading and subtyping. In *Theoretical aspects of computer software (Sendai, 1994)*, pages 273–295. Springer, Berlin.

[**MR1296680**] This item will not be reviewed individually. For details of the collection in which this item appears see 95d:68006.

Wand, M. and Wang, Z. Y. (1994). Conditional lambda-theories and the verification of static properties of programs. *Inform. and Comput.*, 113(2):253–277.

[**MR95f:68159**] Summary: "We present a proof that a simple compiler correctly uses the static properties in its symbol table. We do this by regarding the target code produced by the compiler as a syntactic variant of a λ-term. In general, this λ-term C may not be equal to the semantics S of the source program: they need be equal only when information in the symbol table is valid. We formulate this relation as a conditional λ-judgment $\overline{\Gamma} \supset S = C$, where $\overline{\Gamma}$ is a formula that represents the invariants implicit in the symbol table Γ. We present rules of inference for conditional λ-judgements and prove their soundness. We then use these rules to prove the correctness of a simple compiler that relies on a symbol table. The form of the proof suggests that such proofs may be largely mechanizable."

de'Liguoro, U. and Piperno, A. (1995). Nondeterministic extensions of untyped λ-calculus. *Inform. and Comput.*, 122(2):149–177.

[**MR97a:68104**] The present paper contains an analysis of the operational and denotational semantics of the nondeterministic λ-calculus, which is the classical (untyped) calculus endowed with a choice operator $\oplus$ and two reduction rules $M \oplus N \to M$ and $M \oplus N \to N$.

This adds a new kind of head-redexes, and head-reduction is here a (finitely) branching procedure. The must-operational (or observational) semantics is based on the must-convergency predicate $\downarrow_{\mathrm{must}}$, where $M{\downarrow_{\mathrm{must}}}$ means that M is "strongly head-normalizable": its head-reduction tree is finite.

A fully-abstract model is provided, using the inverse limit construction over non-deterministic algebras, and its full abstraction is proved via a generalization of Böhm trees. Finally, the model is also suitable for Sharma's call-time choice calculus, which proves its consistency. This strengthens Sharma's consistency result, which concerned only a Church-Rosser subsystem, and replaces its very technical syntactic proof by a more attractive semantic one. Reviewed by Chantal Berline.

Engeler, E. (1995). *The combinatory programme.* Birkhäuser Boston Inc., Boston, MA. In collaboration with K. Aberer, B. Amrhein, O. Gloor, M. von Mohrenschildt, D. Otth, G. Schwärzler and T. Weibel.

[MR96k:68125] From the publisher's description: "The programme in combinatory logic, developed at the ETH in Zurich, takes its philosophical basis primarily from the work of two mathematicians of widely separate eras. The first was Richard Dedekind, who was probably the first to base the development of mathematics on pure thought. The second was Haskell Curry, who, as a young man, took on the task of creating a formal basis for the foundation of all mathematics. Probably neither of them would have forseen the extension of their ideas to a profound influence on a discipline that did not even exist in their time. For the purpose of the programme is no less than to rework the mathematical foundations of computer science on such a theory of pure thought. It begins from the idea that, if logic is to be the science of correctly dealing with thought-objects, the underlying theory must be in some sense a part of, or at least a preliminary to, its structure; i.e., a protologic. From this idea, a combinatory algebra is constructed, using a programmatic mixture of the classical axiomatic and set-theoretic approaches.

"This monograph is the result of a sustained effort on the part of the author and the group of students who agreed to work with him to put combinatory algebra in the center of the foundational structure of computer science and related mathematics. It shows that sufficiently rich combinatory algebras can indeed serve as a platform from which the algorithmic aspects of many areas of computer science, mathematics and their applications can be developed. The book will provide a rich experience for scholars and students in such topics as universal algebra, logic, computer algebra, recursion theory, and many other topics that stand at the heart of theoretical computer science.

"The book is divided into the following sections: I. Historical and philosophical background. II. An algebraization of universal algebra. III. An algebraization of algorithmics. IV. Relations to logical computer-algebraic calculi."

Hofmann, M. (1995). Sound and complete axiomatisations of call-by-value control operators. *Math. Structures Comput. Sci.*, 5(4):461–482. Fifth Biennial Meeting on Category Theory and Computer Science (Amsterdam, 1993).

[MR96m:68107] Summary: "We formulate a typed version of call-by-value λ-calculus containing variants of Felleisen's control operators $\mathcal{A}$ and $\mathcal{C}$ that provide explicit access to continuations and logically extend the propositions-as-types correspondence to classical propositional logic. We give an equational theory for this calculus, which is shown to be sound and complete with respect to a class of categorical models based on continuation-passing-style semantics." For the entire collection see MR 96h:68007.

Honda, K. and Yoshida, N. (1995). On reduction-based process semantics. *Theoret. Comput. Sci.*, 151(2):437–486. Thirteenth Conference on Foundations of Software Technology and Theoretical Computer Science (Bombay, 1993).

[MR97f:68044] Various semantics for a very simple process algebra with name

passing are discussed. The language considered, the ν-calculus, is a simplification of the π-calculus introduced by Milner, Parrow and Walker. A number of reduction semantics for it are considered and it is shown how minor variations thereof can capture synchronous and asynchronous behaviours.

It is then shown how these semantics can be characterized, in the style of the chemical abstract machine, via sets of structural congruence laws. The key notion of the contribution is a set of reduction closed properties that are reminiscent of the convertibility (beta-reduction) of lambda calculus.

Via a series of examples on different calculi, the authors then show how a number of bisimulation-based observational equivalences can be recovered within this axiom-based framework. The challenge is left to the reader of generalizing the approach to coarser semantics based on decorated trace equivalences. For the entire collection see MR 96f:68004. Reviewed by Andrea Maggiolo-Schettini.

Jensen, T. P. (1995). Conjunctive type systems and abstract interpretation of higher-order functional programs. *J. Logic Comput.*, 5(4):397–421.

[MR96g:68076] Summary: "We establish an equivalence between two techniques for analysing higher-order functional programs: abstract interpretation and non-standard type inference. The equivalence is based on an axiomatic presentation of the lattices used in abstract interpretation. This axiomatization forms the basis of a program logic for deducing properties of expressions in a simply typed lambda calculus enriched with fixed points and constants. The main result of the paper is that the strictness logic is sound and complete with respect to the abstract interpretation thus proving that strictness analysis by type inference and by abstract interpretation are equally powerful techniques. We then show how a similar result can be obtained for binding-time analysis."

Palsberg, J. and Schwartzbach, M. I. (1995). Safety analysis versus type inference. *Inform. and Comput.*, 118(1):128–141.

[MR96c:68115] Summary: "Safety analysis is an algorithm for determining whether a term in an untyped lambda calculus with constants is safe, i.e., whether it does not cause an error during evaluation. This ambition is also shared by algorithms for type inference. Safety analysis and type inference are based on rather different perspectives, however. Safety analysis is global in that it can analyze only a complete program. In contrast, type inference is local in that it can analyze pieces of a program in isolation. In this paper we prove that safety analysis is sound, relative to both a strict and a lazy operational semantics. We also prove that safety analysis accepts strictly more safe lambda terms than does type inference for simple types. The latter result demonstrates that global program analyses can be more precise than local ones."

Tronci, E. (1995). Defining data structures via Böhm-out. *J. Funct. Programming*, 5(1):51–64.

[MR96c:68127] From the text: "If D is the data structure determined by a free algebra, then an element d in D can be represented in a natural way by a lambda term $\overline{d}$. This paper gives a representation of the data structure itself by constructing a

single lambda term $\langle\langle D\rangle\rangle$ such that M represents an element of the data structure D iff M is a solution of the following two equations: $\langle\langle D\rangle\rangle \mathrm{I} M = M$, $\langle\langle D\rangle\rangle (Kz) M = z$. This follows from Corollary 4.4. Statman (1989) had already proved that every recursively enumerable set of closed lambda terms which is closed under β-conversion is the solution set of a combinatory equation. However, for the given data structures the solution above is without coding."

Benton, N. and Wadler, P. (1996). Linear logic, monads and the lambda calculus. In *11th Annual IEEE Symposium on Logic in Computer Science (New Brunswick, NJ, 1996)*, pages 420–431. IEEE Comput. Soc. Press, Los Alamitos, CA.

[**MR1461853**] This item will not be reviewed individually. For details of the collection in which this item appears see 98a:68006.

Boudol, G. and Laneve, C. (1996). The discriminating power of multiplicities in the λ-calculus. *Inform. and Comput.*, 126(1):83–102.

[**MR97m:68131**] Various refinements or extensions of λ-calculus have been proposed for capturing "resource consciousness" (e.g. linear logic, π-calculus). The refinement proposed by the authors consists of an explicit multiplicity-of-arguments construct and a lazy reduction strategy. With the resulting calculus it is possible to distinguish deadlock and infinite divergence. Contexts with multiplicities are used to observe λ-terms by means of the "observational preorder" $\preceq_{\mathrm{m}}$. The result obtained by the authors is a characterization of such a preorder by intensional preorder. Reviewed by Gianfranco Mascari.

Boudol, G. and Lavatelli, C. (1996). Full abstraction for lambda calculus with resources and convergence testing. In *Trees in algebra and programming—CAAP '96 (Linköping, 1996)*, pages 302–316. Springer, Berlin.

[**MR97e:68072**] Summary: "The calculus with resources is a non-deterministic refinement of lazy lambda calculus which provides explicit means to control the number of times an argument can be used, and introduces the possibility of raising deadlocks during evaluation. It arose in connection with Milner's encoding of lazy lambda calculus into π calculus and proved to be the correct extension to study the semantics induced by π-calculus over pure lambda terms. In this paper, we study a functionality theory in the style of Coppo et al.'s intersection type system [see, e.g., M. Coppo and M. Dezani-Ciancaglini, Notre Dame J. Formal Logic 21 (1980), no. 4, 685–693; MR 82d:03023; M. Coppo, M. Dezani-Ciancaglini and B. Venneri, Z. Math. Logik Grundlag. Math. 27 (1981), no. 1, 45–58; MR 82j:03011] for the calculus of resources extended with convergence testing. The interpretation of terms in this typing system gives rise to a fully abstract semantics of the calculus. This is shown following the definability approach. We also prove that this semantics is not fully abstract for the calculus without convergence testing." For the entire collection see MR 97d:68007.

Crossley, J. N., Mandel, L., and Wirsing, M. (1996). First-order constrained lambda calculus. In *Frontiers of combining systems (Munich, 1996)*, pages 339–356. Kluwer Acad. Publ., Dordrecht.

[MR98b:03023] The authors extend an earlier system of constrained lambda calculus by adding existential quantifiers to both the lambda terms and the constraints. They give some interesting practical examples of mathematical problems, expressed as constrained lambda terms, that can be solved by the extended reduction rules and a "constraint solver" (that can solve certain systems of equations). They provide a denotational semantics for the calculus and prove the Church-Rosser property for the extended form of reduction. For the entire collection see MR 98a:03004. Reviewed by Martin W. Bunder.

Dam, M. (1996). Model checking mobile processes. *Inform. and Comput.*, 129(1):35–51.

[MR97i:68132] The author provides a temporal logic for the polyadic π-calculus by extending the modal μ-calculus with name-parametrisation and dependent sum and product. Following an approach described by M. C. B. Hennessy and H. Lin [Theoret. Comput. Sci. 138 (1995), no. 2, 353–389; MR 96g:68031], the author uses relativised correctness assertions to conditions on names. A proof system for these relativised correctness assertions is presented. The proofs of soundness, completeness, and decidability of the proof system are based on the corresponding proofs for the modal μ-calculus given by C. Stirling and D. J. Walker [Theoret. Comput. Sci. 89 (1991), no. 1, 161–177; MR 93a:03029]. The paper presents the notion of a finite control agent in the π-calculus. Finally, a proof system and a tableau based model checking algorithm are obtained for arbitrary π-calculus processes with finite control. Reviewed by Gabriel Ciobanu.

Davies, R. (1996). A temporal-logic approach to binding-time analysis. In *11th Annual IEEE Symposium on Logic in Computer Science (New Brunswick, NJ, 1996)*, pages 184–195. IEEE Comput. Soc. Press, Los Alamitos, CA.

[MR1461833] This item will not be reviewed individually. For details of the collection in which this item appears see 98a:68006.

Dezani-Ciancaglini, M., de'Liguoro, U., and Piperno, A. (1996). Filter models for conjunctive-disjunctive λ-calculi. *Theoret. Comput. Sci.*, 170(1-2):83–128.

[MR98c:68126] Summary: "The distinction between the conjunctive nature of non-determinism as opposed to the disjunctive character of parallelism constitutes the motivation and the starting point of the present work. λ-calculus is extended with both a non-deterministic choice and a parallel operator; a notion of reduction is introduced, extending β-reduction of the classical calculus. We study type assignment systems for this calculus, together with a denotational semantics which is initially defined constructing a set semimodel via simple types. We enrich the type system with intersection and union types, dually reflecting the disjunctive and conjunctive behaviour of the operators, and we build a filter model. The theory of this model is compared both with a Morris-style operational semantics and with a semantics based on a notion of capabilities."

Ehrhard, T. (1996). Projecting sequential algorithms on strongly stable functions. *Ann. Pure Appl. Logic*, 77(3):201–244.

[MR97f:68109] Summary: "We relate two sequential models of PCF: the sequential algorithm model due to Berry and Curien and the strongly stable model due to Bucciarelli and Ehrhard. More precisely, we show that all the morphisms arising in the strongly stable model of PCF are sequential in the sense that they are the 'extensional projections' of some sequential algorithms. We define a model of PCF where morphisms are 'extensional' sequential algorithms and prove that any equation between PCF terms which holds in this model also holds in the strongly stable model." Reviewed by E. Fehr.

Jim, T. and Meyer, A. R. (1996). Full abstraction and the context lemma. *SIAM J. Comput.*, 25(3):663–696.

[MR97h:68075] From the text: "It is impossible to add a combinator to PCF to achieve full abstraction for models such as Berry's stable domains in a way analogous to the addition of the 'parallel-or' combinator that achieves full abstraction for the familiar complete partial order (cpo) model. In particular, we define a general notion of rewriting system of the kind used for evaluating simply typed λ-terms in Scott's PCF. Any simply typed λ-calculus with such a 'PCF-like' rewriting semantics is shown necessarily to satisfy Milner's context lemma (Recall that Milner's context lemma asserts that to see whether two terms are operationally equivalent it is enough to check that they have the same applicative behaviour.) A simple argument demonstrates that any denotational semantics that is adequate for PCF, and in which certain simple Boolean functionals exist, cannot be fully abstract for any extension of PCF satisfying the context lemma. An immediate corollary is that stable domains cannot be fully abstract for any extension of PCF definable by PCF-like rules.

"Finally the paper gathers some examples of Boolean functionals which allow the separation of operationally equivalent terms in given models of PCF, and a proof that the stable, the bi-stable, and the strongly stable semantics are not closer to full abstraction than the continuous one." Reviewed by Chantal Berline.

Kiwata, K. and Arikawa, S. (1996). Introducing types into elementary formal systems. *Bull. Inform. Cybernet.*, 28(1):79–89.

[MR97d:03007] Summary: "The elementary formal systems (EFSs) work as a theoretical basis in various fields of computer science, such as formal language theory, programming semantics, and machine learning. In this paper, we first define proof figures which illustrate proofs of EFSs and show standard proof figures for ground atoms which are distinctive in the usage of inference rules. Then we formalize the notion of a typed EFS, which is a natural extension of that of an EFS obtained by replacing variables by typed variables, and show some of its basic properties as a logic programming language. Finally, we illustrate first-order logic in terms of typed EFSs." Reviewed by Ronald E. Prather.

Sangiorgi, D. (1996). π-calculus, internal mobility, and agent-passing calculi. *Theoret. Comput. Sci.*, 167(1-2):235–274. TAPSOFT '95 (Aarhus, 1995).

[MR98f:68088] In this paper the author divides the mobility mechanism of the π-calculus into internal and external mobility and studies the subclass πI of the calculus which uses only the internal mobility. It is claimed that internal mobility is responsible for much of the expressiveness of the π-calculus, whereas the external mobility is responsible for many of the semantic complications. Results about encoding λ-calculus, defining a hierarchy of calculi based on internal mobility, and comparisons between agent-passing and name-passing calculi are presented. For the entire collection see MR 97e:68004. Reviewed by Gheorghe Stefanescu.

Tiuryn, J. and Wand, M. (1996). Untyped lambda-calculus with input-output. In *Trees in algebra and programming—CAAP '96 (Linköping, 1996)*, pages 317–329. Springer, Berlin.

[MR97e:68080] Summary: "We introduce an untyped λ-calculus with input-output, based on Gordon's continuation-passing model of input-output. This calculus is intended to allow the classification of possibly infinite input-output behaviors, such as those required for servers or distributed systems. We define two terms to be operationally approximate iff they have similar behaviors in any context. We then define a notion of applicative approximation and show that it coincides with operational approximation for these new behaviors. Finally, we consider the theory of pure λ-terms under this notion of operational equivalence." For the entire collection see MR 97d:68007.

Boudol, G. (1997). Typing the use of resources in a concurrent calculus (extended abstract). In *Advances in computing science—ASIAN'97 (Kathmandu, 1997)*, pages 239–253. Springer, Berlin.

[MR1638564] This item will not be reviewed individually. For details of the collection in which this item appears see 99b:68001.

Fu, Y. (1997). A proof theoretical approach to communication. In *Automata, languages and programming (Bologna, 1997)*, pages 325–335. Springer, Berlin.

[MR1616195] This item will not be reviewed individually. For details of the collection in which this item appears see 98k:68002.

Herbelin, H. (1997). Games and weak-head reduction for classical PCF. In *Typed lambda calculi and applications (Nancy, 1997)*, pages 214–230. Springer, Berlin.

[MR98g:03044] Summary: "We present a game model for classical PCF, a finite version of PCF extended by a catch/throw mechanism. This model is built from E-dialogues, a kind of two-player game defined by P. Lorenzen. In the E-dialogues for classical PCF, the strategies of the first player are isomorphic to the Böhm trees of the language.

"We define an interaction in E-dialogues and show that it models the weak-head reduction in classical PCF. The interaction is a variant of Coquand's debate and the weak-head reduction is a variant of the reduction in Krivine's Abstract Machine. We then extend E-dialogues to a kind of games similar to Hyland-Ong games. Interaction in these games also models weak-head reduction. In the intuitionistic case (i.e. without the catch/throw mechanism), the extended E-dialogues are Hyland-Ong games for which the innocence condition on strategies is now a rule.

"Our model for classical PCF is different from Ong's model of Parigot's lambda-mu calculus. His model works by adding new moves to the intuitionistic case, whereas ours works by relaxing the game rules." For the entire collection see MR 98f:68017.

Lafont, Y. (1997). Interaction combinators. *Inform. and Comput.*, 137(1):69–101.

[MR99c:68079] Summary: "It is shown that a very simple system of interaction combinators, with only three symbols and six rules, is a universal model of distributed computation, in a sense that we make precise. This paper is a continuation of the author's work on interaction nets, inspired by Girard's proof nets for linear logic, but no preliminary knowledge of these topics is required for its reading."

Lenisa, M. (1997). Semantic techniques for deriving coinductive characterizations of observational equivalences for λ-calculi. In *Typed lambda calculi and applications (Nancy, 1997)*, pages 248–266. Springer, Berlin.

[MR99a:03021] Summary: "Coinductive (applicative) characterizations of various observational congruences which arise in the semantics of λ-calculus, for various reduction strategies, are discussed. Two semantic techniques for establishing the coincidence of the applicative and the contextual equivalences are analyzed. The first is based on intersection types; the second is based on a mixed induction-coinduction principle." For the entire collection see MR 98f:68017.

Milner, R. (1997). Strong normalisation in higher-order action calculi. In *Theoretical aspects of computer software (Sendai, 1997)*, pages 1–19. Springer, Berlin.

[MR99a:68068] The author has proposed action calculi as a framework to accomodate a variety of disciplines of interaction and computation [see Acta Inform. 33 (1996), no. 8, 707–737; MR 97j:68028]. In such a framework a specific action calculus $\mathrm{AC}(\mathcal{K})$ is determined by a set $\mathcal{K}$ of control operations and associated reaction rules. Examples of specific action calculi are the λ-calculus and the π-calculus.

In a previous paper the author has shown that any action calculus can be extended in a conservative manner to higher order, thus allowing its actions to be encapsulated and treated as data. The dynamics of each higher-order calculus includes β-reduction, analogous to the case of the λ-calculus.

The result of the present paper shows that under an assumption analogous to the assumption of simple types in the λ-calculus, β-reduction in higher-order action calculi is strongly normalising and confluent. For the entire collection see MR 98i:68019. Reviewed by Andrea Maggiolo-Schettini.

Sewell, P. (1997). Nonaxiomatisability of equivalences over finite state processes. *Ann. Pure Appl. Logic*, 90(1-3):163–191.

[MR99e:68104] The author considers certain notions of behavioural equivalences

over a calculus of finite state processes and investigates the existence of finite "equational" axiomatisations of them. Here, nondeterministic finite state machines are taken as the basis for modelling computational phenomena, and the labelled transition system, presented in a simple syntactic form (that of the -expressions of Milner) is the formalisation adopted to represent them. As a consequence, the paper finds a place in the extensive literature concerned with the axiomatisation of behavioural equivalences over the -expressions (and other simple process calculi). In that literature, a number of complete axiomatic systems can be found that contain an impure Horn clause (together with a finite set of equational axioms). In this paper, the author confirms the intuition that the use of implication is essential, showing that there is no finite equational axiomatisation for any of a wide class of equivalences over -expressions.

The calculus is embedded in a simply typed lambda calculus, allowing the properties involved to be expressed as equations between terms containing first-order variables. Moreover, intermediate results are applied to give two nonaxiomatisability results over calculi of regular expressions. Reviewed by A. Aiello.

van Eijck, J. (1997). Typed logics with states. *Log. J. IGPL*, 5(5):623–645 (electronic).

[MR98j:03017] A typed logic with states (TLS) is considered. TLS arises from a standard typed lambda calculus by adding a special function which is called a function for register update.

An equational calculus for TLS, extending the familiar rules for equalities by new axioms for register lookup, register update and register extensionality is given.

It is shown that universal validity of an equality is decidable for this case. A variant of the completeness theorem is proved. Reviewed by Feodor Alexandrovich Murzin.

Amadio, R. M. and Curien, P.-L. (1998). *Domains and lambda-calculi.* Cambridge University Press, Cambridge.

[MR1694538] A review for this item is in process.

Lafont, Y. (1998). Erratum: "Interaction combinators" [Inform. and Comput. **137** (1997), no. 1, 69–101; 1468878]. *Inform. and Comput.*, 142(2):237.

[MR1613887] There will be no review of this item.

Nour, K. (1998). $\underline{S}$-storage operators. *Math. Logic Quart.*, 44(1):99–108.

[MR99g:03014] Storage operators were introduced in pure lambda calculus by J. L. Krivine in 1990 to avoid the drawback of the left reduction strategy: a function computes its arguments every time it uses it. The class of $\underline{S}$-storage operators, for an arbitrary successor $\underline{S}$, is introduced in the paper: this class strictly includes the class of storage operators and offers the same improvement with respect to left reduction. The paper is rather technical, but well written. A semantical characterisation of $\underline{S}$-storage operators could be of some interest. Reviewed by Gianfranco Mascari.

Sangiorgi, D. (1998). An interpretation of typed objects into typed π-calculus. *Inform. and Comput.*, 143(1):34–73.

[MR99k:68064] Summary: "An interpretation of M. Abadi and L. Cardelli's first-order function object calculus into a typed π-calculus is presented. The interpretation validates the subtyping relation and the typing judgements of the object calculus and is computationally adequate. This is the first interpretation of a typed object-oriented language into a process calculus. The study intends to offer a contribution to understanding, on the one hand, the relationship between π-calculus types and conventional types of programming languages, and, on the other hand, the usefulness of the π-calculus as a metalanguage for the semantics of typed object-oriented languages.

"The type language for the π-calculus has Pierce and Sangiorgi's I/O annotations to separate the capabilities of reading and writing on a channel and variant types. Technical contributions of the paper are the presentation of variant types for the π-calculus and their typing and subtyping properties, and an analysis of behavioural equivalences in a π-calculus with variant types."

Dezani-Ciancaglini, M., Tiuryn, J., and Urzyczyn, P. (1999). Discrimination by parallel observers: the algorithm. *Inform. and Comput.*, 150(2):153–186.

[**MR1687116**] A review for this item is in process.

2.2 Proof-checking

de Bruijn, N. G. (1980). A survey of the project AUTOMATH. In *To H. B. Curry: essays on combinatory logic, lambda calculus and formalism*, pages 579–606. Academic Press, London.

[**MR81m:03017**] The AUTOMATH project is an attempt to write machine-checkable proofs for substantial portions of mathematics, such as E. Landau's book on the development of real numbers from natural numbers (Grundlagen der Analysis, Akademie-Verlag, Leipzig, 1930; Jbuch 56, 191). The author gives a brief survey of this project, which has been going on at Eindhoven for over a decade. (Reviewer's remarks: One disconcerting aspect of the AUTOMATH language is its assembler code-like format. Another is the fact that it seems not to make any use of the improved automatic theorem proving techniques developed during the last decade (e.g., R. S. Boyer and J. S. Moore, A computational logic, Academic Press, New York, 1979; MR 81d:68127). Nevertheless, the AUTOMATH project remains a significant and important attempt at formalizing the results of reasoning in mathematics.) (For the entire collection see MR 81j:03005.) Reviewed by Frank Malloy Brown.

Barendregt, H. and Rezus, A. (1983). Semantics for classical AUTOMATH and related systems. *Inform. and Control*, 59(1-3):127–147.

[**MR86g:03100**] The task of the AUTOMATH project is the formalization of (present day) mathematics so that proofs can be formally checked (e.g., by a computer). Although the languages of the AUTOMATH family are in some sense essentially typed lambda-calculi their structure is quite complicated; the publications in this area in general deal with the syntactic aspects. In this paper a formal semantics for AUTOMATH is provided. The domains of interpretation are models of the type-free lambda-calculus; hence the type distinctions of AUTOMATH have to be translated into a type-free setting. This leads to a notion of truth-evaluation similar to the one in first-order or second-order logic, and it is proved that the syntactic consequence operator is correct with respect to this semantics. Reviewed by M. M. Richter.

Rezus, A. (1983). *Abstract AUTOMATH.* Mathematisch Centrum, Amsterdam.

[MR84j:03030] It is an old idea to represent mathematics in a pure formal system. Such attempts are connected with names like Leibniz, Peano, Whitehead and Russell, Hilbert and others. The formal language required for this purpose has to reflect many aspects: the theorems, the proofs, the ideas of the proofs, the formal and informal background, etc. All historical approaches have failed insofar as they did not represent a completely formalized substantial part of mathematics without any gaps. In the mid- sixties N. G. de Bruijn initiated a far-reaching enterprise for the formalization of mathematics: AUTOMATH. In the meantime the project has been continuously developed and by now quite a number of AUTOMATH-languages exist for different purposes. The idea of this book is to represent the syntax and structure of some of these languages in some detail. The main parts are: Well-formedness, correctness, and its main aspects ("reference order", "complexity", "economy of means"). From the intention it is quite clear that the structure of such a "generalized typed lambda-calculus" is somewhat complicated. The reader is carefully guided and often motivated. Nevertheless, in order to get a good feeling for the system one will probably actually have to work with it and to use it in practice. Reviewed by M. M. Richter.

Coquand, T. and Huet, G. (1985). Constructions: a higher order proof system for mechanizing mathematics. In *EUROCAL '85, Vol. 1 (Linz, 1985)*, pages 151–184. Springer, Berlin.

[MR826550] This item will not be reviewed individually. For details of the collection in which this item appears see 87a:68006.

Zaionc, M. (1985). The set of unifiers in typed λ-calculus as regular expression. In *Rewriting techniques and applications (Dijon, 1985)*, pages 430–440. Springer, Berlin.

[MR88f:68130] Introduction: "The main problem for mechanization of theorem proving in the typed λ-calculus is the problem of determining whether two terms of a fixed type with free variables have a common instance. This unification problem is undecidable for orders ≥ 2 [see G. Huet, Inform. and Control 22 (1973), 257–267; MR 54 #2442; W. Goldfarb, Theoret. Comput. Sci. 13 (1981), no. 2, 225–230; MR 82c:03067]. A paper by Huet [ibid. 1 (1975), no. 1, 27–57; MR 54 #6583] contains the description of a semidecision algorithm which, in the case of success, returns a substitution called the most general unifier. This algorithm produces the matching tree, in which the most general unifiers are represented by the final branches. The most interesting is the case when the matching tree is infinite and consists of an infinite number of most general unifiers. Note that some infinite matching trees have an interesting property: for every infinite branch there is a node which occurs earlier in this tree. This means that there are some fragments which are repeated in this tree.

"In this paper we consider a class of unification problems. For a fixed problem from

this class the set of unifiers can be represented by the regular expression built up on the basis of a finite alphabet. This alphabet consists of symbols called elementary substitutions. Every word is interpreted as a composition of the elementary substitutions." For the entire collection see MR 87a:68017.

Glickfeld, B. and Overbeek, R. (1986). A foray into combinatory logic. *J. Automat. Reasoning*, 2(4):419–431.

[MR88h:03016] The authors have run itp, the Argonne equational theorem prover, on some problems in R. Smullyan's book on combinatory logic, To mock a mockingbird [Knopf, New York, 1985], and they report their experience. The system solved the early problems quickly; the paper reports in detail its solution to a problem combinatory logicians would formulate as follows: Produce a fixed-point expression in terms of B and WI ($Bxyz = x(yz)$, $(WI)x = xx$; WI is called M by Smullyan, as it is, in his birds-in-a-forest metaphor, a mockingbird, which responds to any bird x as x itself would); itp produced $\mathrm{Fix}(x) = M(BxM)$ within seconds. The bulk of the paper reports how itp improved over Smullyan, and how the authors, supported by itp, improved over itp, on the following problem: Express an "egocentric bird" (E such that $EE = E$) in terms of W alone ($Wxy = x(yy)$; it was called L, a "lark", by Smullyan). Smullyan was unable to find an expression shorter than 12 L's; itp produced a 10-L expression in less than 2 minutes, and the authors then guided it to an 8-L expression, and were unable to do better or to show optimality; in particular, neither the authors nor itp could decide whether $(LL)(LL)$ is egocentric. It seems from the examples that itp could not use any form of the extensionality rule (infer $A = B$ from $Ax = Bx$ for x not occurring in A, B), which the authors obviously did, but the point is not discussed. Reviewed by Dean Rosenzweig.

Andrews, P. B. (1987). Typed λ-calculus and automated mathematics. In *Mathematical logic and theoretical computer science (College Park, Md., 1984–1985)*, pages 1–14. Dekker, New York.

[MR89h:03019] From the summary: "It would be useful for researchers on automating mathematics to arrive at a consensus concerning which languages are most appropriate for such endeavors, since this would facilitate valuable cooperation in this field. Probably the best known language for formalizing mathematics is axiomatic set theory, but there are certain advantages to using type theory (also known as higher-order logic), especially in a formulation due to Alonzo Church which incorporates λ-notation. Higher-order logic needs to be more widely known, so it is described and discussed here, with special attention to typed λ-calculus and issues relevant to the automation of mathematics." For the entire collection see MR 88i:03005.

Coquand, T. and Huet, G. (1987). Concepts mathématiques et informatiques formalisés dans le calcul des constructions. In *Logic colloquium '85 (Orsay, 1985)*, pages 123–146. North-Holland, Amsterdam.

[MR88j:68149] The "calculus of constructions", developed by the authors, is based on a typed lambda calculus. It can be used as a formal language for expressing notions from mathematics and computer science; the reduction mechanism of lambda calculus, together with the syntactic properties of the type relation, give it the power of a useful formal calculus, suited for a concise treatment of reasoning—in the style of natural deduction—or programming.

Terms in the language are divided into three levels, corresponding to objects (or proofs), types (or propositions) and the class of all types. In this respect, and in a number of other aspects, the calculus is an adaptation of a version of de Bruijn's Automath, as mentioned explicitly by the authors. A minor difference in the definitions is that types in the calculus of constructions are required to be in normal form. The authors add a metalanguage based on ML, with explicit polymorphism, in order to facilitate the generation of constructions.

The paper gives the syntactic basis of the calculus, followed by a number of worked-out examples. These examples—from logic, set theory, universal algebra and the theory of ordinal numbers—are well chosen and instructive. In their conclusion, the authors state that the formalism is powerful enough to be considered as a high-order programming language, albeit general recursion is not incorporated and efficient machine execution is not yet possible. For the entire collection see MR 88b:03003. Reviewed by R. P. Nederpelt.

de Bruijn, N. G. (1987). Generalizing Automath by means of lambda-typed lambda calculus. In *Mathematical logic and theoretical computer science (College Park, Md., 1984–1985)*, pages 71–92. Dekker, New York.
[MR89b:03026] Summary: "The calculus developed in this paper is claimed to be able to embrace all the essential aspects of Automath, apart from the feature of type inclusion, which are not considered in this paper. The calculus deals with a correctness notion for lambda-typed lambda formulas (which are presented in the form of what are called lambda trees). To an Automath book there corresponds a single lambda tree, and the correctness of the book is equivalent to the correctness of the tree. The algorithmic definition of correctness of lambda trees corresponds to an efficient checking algorithm for Automath books." For the entire collection see MR 88i:03005.

Huet, G. (1987). Deduction and computation. In *Logic of programming and calculi of discrete design (Marktoberdorf, 1986)*, pages 305–342. Springer, Berlin.
[MR89i:68029] These course notes give a concise overview of the syntactic notions underlying the theory of deduction and computation, in a general and uniform setting. A preliminary Section 1 gives an abstract syntax for strings and tree-shaped terms. Section 2 discusses inference systems, including such subjects as combinatory logic, polymorphism and natural deduction. Moreover, connections with the "propositions as types" notion and with Prolog-strategies are illustrated. In Section 3 the computational aspects of inference rules and rewriting are explained, with a short excursion to the categorical viewpoint. Section 4 deals with the relation between Gentzen-type natural deduction and lambda-calculus. It also discusses recursion and an incorporation of polymorphism into the language. This leads to an ML-like formalism. A further extension yields Girard's second-order typed lambda calculus. In the final Section 5 the theory is briefly compared with Martin-Löf's intuitionistic theory

of types, with de Bruijn's AUTOMATH and with Coquand's theory of constructions. For the entire collection see MR 88g:68006. Reviewed by R. P. Nederpelt.

Li, D. F. (1987). Simple proofs of soundness and completeness for ET-proofs. *J. Tsinghua Univ.*, 27(6):49–56.

[MR89j:03009] This paper is about automatic theorem proving in higher-order logic. P. B. Andrews and G. P. Huet, respectively, extended J. A. Robinson's resolution principle and unification algorithm for first-order logic to higher-order logic. In the proof for expansion trees, the concept of a shallow formula was introduced in D. A. Miller's thesis ["Proofs in higher-order logic", Ph.D. Thesis, Carnegie-Mellon Univ., Pittsburgh, PA, 1983], so that the top-level instantiated nodes could be eliminated. But in this paper the author suggests eliminating the bottom-level instantiated nodes. The advantages are that the intuition is improved and the proof procedure is shortened. In addition, some errors in Miller's thesis are corrected. Reviewed by Mao Kang Wu.

McCune, W. and Wos, L. (1987). A case study in automated theorem proving: finding sages in combinatory logic. *J. Automat. Reasoning*, 3(1):91–107.

[MR88d:03017] Summary: "We present in this article an application of automated theorem proving to a study of a theorem in combinatory logic. The theorem states that the strong fixed point property is satisfied in a system that contains the B and W combinators. The theorem can be stated in terms of R. M. Smullyan's forests of birds: a sage exists in a forest that contains a bluebird and a warbler [Smullyan, To mock a mockingbird, Knopf, New York, 1985]. Proofs of the theorem construct sages from B and W. Prior to the study, one sage, discovered by Statman, was known to exist. During the study, with much assistance from two automated theorem-proving programs, four new sages were discovered. The study was conducted from a syntactic point of view because the authors know very little about combinatory logic. The uses of the automated theorem-proving programs are described in detail."

Coquand, T. and Huet, G. (1988). The calculus of constructions. *Inform. and Comput.*, 76(2-3):95–120.

[MR89j:68096] The "calculus of constructions", part of which was the subject of Coquand's thesis ["Une theorie des constructions", These de 3^{e} cycle, Univ. Paris VII, Paris, 1985; per bibl.], is a formalism for expressing constructive proofs in a natural deduction manner. It is based on a typed lambda-calculus and it is in a certain sense an attractive mixture of systems like de Bruijn's Automath, Martin-Löf's intuitionistic theory of types and Girard's system F.

Proofs of propositions can be expressed in such typed lambda-calculi as lambda-expressions which are "members" ("inhabitants") of the corresponding proof type, following the well-known propositions-as-types notion of Curry and Howard. The computational aspects of a proof are incorporated in the "stripped" version of the lambda-expression formalizing the proof, i.e. in the associated pure, nontyped term. This enables one to regard such calculi as high-level functional programming languages.

The system presented in this paper encompasses polymorphism, has the property that typable terms have an effectively computable type and has the strong normalization property, guaranteeing termination of calculations. The authors give both the formal syntax and the backgrounds of the calculus of constructions. They also discuss its interpretation and its use, claiming it to be appropriate for correct program development. Reviewed by R. P. Nederpelt.

McCune, W. and Wos, L. (1988). Errata: "A case study in automated theorem proving: finding sages in combinatory logic" [J. Automat. Reasoning **3** (1987), no. 1, 91–107; MR 88d:03017]. *J. Automat. Reason.*, 4(1):110–111.

[MR89d:03010] The authors correct the statement of the axiom of extensionality and note some consequent corrections that are necessary.

Shankar, N. (1988). A mechanical proof of the Church-Rosser theorem. *J. Assoc. Comput. Mach.*, 35(3):475–522.

[MR90j:03023] Summary: "The Church-Rosser theorem is a celebrated metamathematical result on the lambda calculus. We describe a formalization and proof of the Church-Rosser theorem that was carried out with the Boyer-Moore theorem prover. The proof presented in this paper is based on that of Tait and Martin-Löf. The mechanical proof illustrates the effective use of the Boyer-Moore theorem prover in proof-checking difficult metamathematical proofs."

Staples, J. (1988). Delaying unification algorithms for lambda calculi. *Theoret. Comput. Sci.*, 56(3):277–288.

[MR90b:03015] Summary: "Huet proposed delaying some unifications of typed lambda terms, as part of a higher-order resolution method. This paper abstracts that idea from the resolution context. Unification delays are formalised independently of any context, by defining a lambda calculus where a unification constraint forms an integral part of each term. This calculus is shown to support an unusually simple unification theory, where most general unifiers trivially always exist. Attention shifts from the existence of unifiers to the simplification of expressions for most general unifiers. The approach is convenient for discussing the unification of untyped lambda terms and has promise for discussing schematic unification of term schemes. It may also be a convenient format for unification algorithms in rewriting systems that use lazy evaluation to compute with notations for infinite structures."

Wos, L. and McCune, W. (1988). Challenge problems focusing on equality and combinatory logic evaluating automated theorem-proving programs. In *9th International Conference on Automated Deduction (Argonne, IL, 1988)*, pages 714–729. Springer, Berlin.

[MR956121] This item will not be reviewed individually. For details of the collection in which this item appears see 89c:68004.

Zaionc, M. (1988). Mechanical procedure for proof construction via closed terms in typed λ calculus. *J. Automat. Reason.*, 4(2):173–190.

[**MR89i:03030**] There are many proof search procedures for propositional intuitionistic implicational logic, the most sophisticated of which is perhaps the one implemented in the PRIZ programming system [the reviewer and E. Tyugu, Sci. Comput. Programming 2 (1982), no. 3, 215–240; MR 85e:68060]. Most of these procedures are based on Gentzen L-type systems with introduction into the antecedent and succedent. A straightforward, and probably the most efficient, way to obtain natural deduction is to apply Prawitz' translation to L-type derivations. The author presents a proof search algorithm working directly for natural deductions written via typed lambda-terms (formulas-as-types). One formula can have infinitely many different (up to normal form) derivations, and the author, following Huet, constructs a simple finite grammar generating such derivations. Implementation is not mentioned. Reviewed by G. E. Mints.

Avron, A., Honsell, F., and Mason, I. A. (1989). An overview of the Edinburgh logical framework. In *Current trends in hardware verification and automated theorem proving*, pages 323–340. Springer, New York.
[**MR90f:03022**] Summary: "In recent years there has been a growing interest in using computers as an aid for correctly manipulating logical systems, as well as using formal systems for correctly designing computers. However, implementing a proof environment for a specific logical system is both complex and time-consuming; this—together with the proliferation of logics—suggests that a uniform and reliable alternative is desirable. One such alternative is the Edinburgh logical framework (LF), currently under development at the Laboratory for Foundations of Computer Science. The LF is a logic-independent tool which, given a specification for a logical system, synthesizes a proof editor and checker for that system. Its specification language is based on a general theory of logics, which makes it possible to capture uniformities and idiosyncrasies of a large class of logics without sacrificing generality for tractability. Peculiarities (such as side conditions on rule application, variable occurrence or formula formation) are expressed at the level of the specification.

"A paper by R. Harper, Honsell and G. Plotkin [in Symposium on Logic in Computer Science (Ithaca, NY, 1987), 194–204, IEEE Comput. Soc. Press, Washington, DC, 1987; see MR 88k:68012] described the basic features of the LF, while another paper of ours ["Using typed lambda calculus to implement formal systems on a machine", Tech. Rep., No. ECS-LFCS-87-31, Lab. Foundations Comput. Sci., Edinburgh Univ., Edinburgh, 1987; per bibl.] provided a broader illustration of its applicability and discussed to what extent it is successful. The present paper serves as an introduction to the LF and summarizes the main points made in our 1987 paper." For the entire collection see MR 90c:68002.

Elliott, C. M. (1989). Higher-order unification with dependent function types. In *Rewriting techniques and applications (Chapel Hill, NC, 1989)*, pages 121–136. Springer, Berlin.
[**MR1070372**] This item will not be reviewed individually. For details of the collection in which this item appears see 91f:68006.

Fribourg, L. (1989). Proofs by combinatory induction on recursively reducible expressions. In *Resolution of equations in algebraic structures, Vol. 2*, pages 117–140. Academic Press, Boston, MA.
[**MR92e:68171**]

There have been several approaches to inductive theorem proving in equational number theory. One of them is based on term rewriting and completion [see, e.g., J. P. Jonannaud and H. Kirchner, EATCS Bull. (1985), 49–55; per revr.]. In this paper, the author presents a method which is based on combinatory logic.

The basic terms are numbers, where succ and zero are two constant symbols and the number n is represented by succ$(\cdots$(succ zero)$\cdots)$. The inductive terms are of the form $M =_{\text{weak}} Fx\overline{v}\,\overline{w}$, where $F(n+1)\overline{y}\,\overline{z} = Pn(Fn(Q\overline{ny})(Q'\overline{z}))$ (F, P are constants for functions, $\overline{Q}, \overline{Q}'$ vectors of function constants). In order to deal with the word problem for such inductive terms, combinatorial rules are introduced. These rules are based on the two combinators: (1) recursion combinator: $R'XYZ0 = X$, $R'XYZ(n+1) = Y(Zn)(R'XYZn)$; (2) iteration combinator: $JXY0 = X$, $JXY(n+1) = Y(JXYn)$.

The central rule for combinatory transformation is the rule of combinatory recursive reducibility; this rule transforms an equation (representing a recursion) into a combinatory equation containing R' and J. From this rule, the rules of combinatory recursive inclusion and iterative embedding are derived. To prove an equation $s = t$, the terms s and t have to be transformed to a term u (not necessarily in normal form) in the equational theory $=$ r; $=$ r is defined by extensional equality + the rule of combinatory recursive reducibility. The method is demonstrated by several examples (one of these was to prove associativity of $*$, $*$ being defined by $x*(y+1) = x*y+x$).

The method is interesting, but still needs improvements to become computationally relevant. So it would be necessary to find a reduction relation for $=$ r such that symmetry of $=$ r can be avoided. A comparison with "induction-free" induction methods in term-rewriting theory would be useful too (they are not even mentioned in the references). For the entire collection see MR 91b:68010. Reviewed by Alexander Leitsch.

Snyder, W. and Gallier, J. (1989). Higher-order unification revisited: complete sets of transformations. *J. Symbolic Comput.*, 8(1-2):101–140.
[MR90j:03025] The authors discuss the problem of general higher-order unification and develop an approach based on the method of transformations on systems of terms which was used by A. Martelli and U. Montanari in the context of first-order unification [ACM Trans. Programming Languages Systems 4 (1982), no. 2, 258–282; CCA 1982:30336]. This method consists of applying simple transformations until a system in solved form is obtained whose solution is obvious. After a good introduction to this field, the full method is demonstrated in the first-order case. The fundamental concepts of higher-order unification are presented. A set of transformations for higher-order unification is given and the method mentioned above is extended to this case. The soundness and the completeness of the given set of transformations are proved. Finally, the authors show how Huet's method for pre-unification can be described as a special case of this set of transformations. This paper can also be recommended to readers unfamiliar with this field. Reviewed by Hans-Joachim Goltz.

Sarni, D. (1990). Lambda-calculus on occurrence. The theory and its combinatorial counterpart. *J. Inform. Process. Cybernet.*, 26(3):169–187.

[**MR91e:68043**] A new lambda calculus notation, called lambda calculus on occurrence, is introduced. This system differs from AUTOMATH, but is similar in spirit in that it has no variables and is concerned with machine implementation of the lambda calculus. For instance, the ordinary λ-calculus term $\lambda x(x(\lambda y(xy)))$ is represented in the λ-calculus on occurrence by $\lambda(2,6)(x(\lambda(7)(xy)))$, where sequences of integers indicate the position of bound occurrences and x, y are considered as basic terms. A combinatory theory related to the λ-calculus on occurrence is introduced. The author argues the usefulness of λ-calculus on occurrence for automatic manipulation of λ-calculus expressions. Reviewed by Peter Clote.

Snyder, W. (1990). Higher order E-unification. In *10th International Conference on Automated Deduction (Kaiserslautern, 1990)*, pages 573–587. Springer, Berlin.

[**MR1077024**] This item will not be reviewed individually. For details of the collection in which this item appears see 91g:68004 .

Snyder, W. and Gallier, J. (1990). Higher-order unification revisited: complete sets of transformations [MR 90j:03025]. In *Unification*, pages 565–604. Academic Press, London.

[**MR1090388**] This item will not be reviewed individually. For details of the collection in which this item appears see 91j:68006.

Statman, R. (1990). Some models of Scott's theory LCF based on a notion of rate of convergence. In *Logic and computation (Pittsburgh, PA, 1987)*, pages 263–280. Amer. Math. Soc., Providence, RI.

[**MR91i:03034**] Since the introduction of Dana Scott's LCF (logic of computable functionals), several theorem-proving systems based on LCF have been developed. It is important to study the significance of such formal systems. However, these problems are difficult. In this paper, the author provides a new approach to establishing the significance of some formal fragments of LCF. Four systems, FP, LFP, OWHY and PPλ, are presented. The author constructs models $\mathcal{A}_\infty$ which are dependent on an algebra $\mathcal{F}$. The author proves that if $\mathcal{F}$ consists of H-functions, where H is, respectively, h.m.l., h.m.p.r., h.m.m.r. or h.m.P.r, then $\mathcal{A}_\infty$ satisfies the universal Horn theorems of J, where J is, respectively, FP, LFP, OWHY or PPλ. In the proof, the method of rate of convergence is used. The models constructed here are applied to a question of Scott; the author gives the answer that the universal Horn theorems of OWHY and PP cannot be finitely axiomatized by universal Horn sentences. In addition, he shows that finitely many least fixed point operators are not sufficient to define every least fixed point operator. For the entire collection see MR 91a:03002.

Wos, L., Winker, S., McCune, W., and et al. (1990). Automated reasoning contributes to mathematics and logic. In *10th International Conference on Automated Deduction (Kaiserslautern, 1990)*, pages 485–499. Springer, Berlin.

[**MR1077019**] This item will not be reviewed individually. For details of the collection in which this item appears see 91g:68004.

Dowek, G. (1991). *Démonstration automatique dans le calcul des constructions.* Institut National de Recherche en Informatique et en Automatique (INRIA), Rocquencourt. Thèse, Université de Paris VII, Paris, 1991.

[**MR95i:68116**] Automated deduction is a challenging subject even for classical (propositional or first-order) logics. The main concern of this thesis is to treat the mentioned theme in higher-order logics ("Dans cette these, nous avons decrit une methode de semi-decision pour les systemes de types"). "Le calcul des constructions"—as a logical proof-system—is in fact a typed λ-calculus for which the Curry-Howard correspondence is used: "To declare an axiom P simple means to declare a variable of type P and proving a formula P simple means to provide a λ-term of type P." The book is divided into three chapters. The first one is mainly introductory (Logique du premier ordre; Logique d'ordre superieur; Systemes de types; Bonne fondation de la recursion simultanee sur le type et les sous-termes stricts d'un terme). Chapter two ("Demonstration automatique") is the "core" of this work, its sections being self-explanatory (Contextes quantifies, Constraints, Substitutions: Une methode complete de synthese de preuve; Une restriction incomplete). The last chapter is dedicated to the particular case of resolution in equational logics and has three sections (Unification; Equations decidables; Equations indecidables). We think that the thesis contains many valuable orginal results and conjectures and will be a source of fruitful future research. Reviewed by C. Masalagiu.

Helmink, L. and Ahn, R. (1991). Goal directed proof construction in type theory. In *Logical frameworks (Sophia-Antipolis, 1990)*, pages 120–148. Cambridge Univ. Press, Cambridge.

[**MR1139783**] This item will not be reviewed individually. For details of the collection in which this item appears see 92f:68003.

Huet, G. and Plotkin, G., editors (1991). *Logical frameworks*, Cambridge. Cambridge University Press.

[**MR92f:68003**] Contents: Peter Aczel, David P. Carlisle and Nax Mendler, Two frameworks of theories and their implementation in Isabelle (pp. 3–39); N. G. de Bruijn, A plea for weaker frameworks (pp. 40-67); Philippe de Groote, Nederpelt's calculus extended with a notion of context as a logical framework (pp. 69–86); David Basin and Matt Kaufmann, The Boyer-Moore prover and Nuprl: an experimental comparison (pp. 89–119); Leen Helmink and Rene Ahn, Goal directed proof construction in type theory (pp. 120–148); Frank Pfenning, Logic programming in the LF logical framework (pp. 149–181); Rod Burstall and Furio Honsell, Operational semantics in a natural deduction setting (pp. 185–214); Amy Felty, Encoding dependent types in an intuitionistic logic (pp. 215–251); Thierry Coquand, An algorithm for testing conversion in type theory (pp. 255–279); Peter Dybjer, Inductive sets and families in Martin-Löf's type theory and their set-theoretic semantics (pp.

280–306); David Pym and Lincoln Wallen, Proof-search in the $\lambda\Pi$-calculus (pp. 309–340); Robert Constable and Chet Murthy, Finding computational content in classical proofs (pp. 341–362); Lars Hallnas, Models of partial inductive definitions (pp. 365–384); Peter Schroeder-Heister, Structural frameworks, substructural logics, and the role of elimination inferences (pp. 385–403). The papers will not be reviewed individually.

Lassez, J.-L. and Plotkin, G., editors (1991). *Computational logic.* MIT Press, Cambridge, MA. Essays in honor of Alan Robinson.
[**MR92d:03002**]

Contents: Larry Wos, Ross Overbeek and Ewing Lusk, Subsumption, a sometimes undervalued procedure (pp. 3–40); Hans Jurgen Ohlbach and Jorg H. Siekmann, The Markgraf Karl refutation procedure (pp. 41–112); Melvin Fitting, Modal logic should say more than it does (pp. 113–135); W. W. Bledsoe, Interactive proof presentation (pp. 136–165); Maurice Bruynooghe, Intelligent backtracking revisited (pp. 166–177); Alan Bundy, A science of reasoning (pp. 178–198); Ehud Shapiro, Inductive inference of theories from facts (pp. 199–254); Jean-Pierre Jouannaud and Claude Kirchner, Solving equations in abstract algebras: a rule-based survey of unification (pp. 257–321); Hubert Comon, Disunification: a survey (pp. 322–359); Deepak Kapur and Hantao Zhang, A case study of the completion procedure: proving ring commutativity problems (pp. 360–394); Gerard Huet and Jean-Jacques Levy, Computations in orthogonal rewriting systems. I, II (pp. 395–414, 415–443); Paris C. Kanellakis, Harry G. Mairson and John C. Mitchell, Unification and ML-type reconstruction (pp. 444–478); Mitchell Wand and Patrick M. O'Keefe, Automatic dimensional inference (pp. 479–483); Keith L. Clark, Logic-programming schemes and their implementations (pp. 487–541); Donald W. Loveland and David W. Reed, A near-Horn Prolog for compilation (pp. 542–564); P. A. Gardner and J. C. Shepherdson, Unfold/fold transformations of logic programs (pp. 565–583); Andrea Corradini and Ugo Montanari, An algebraic representation of logic-program computations (pp. 584–612); Jack Minker, Arcot Rajasekar and Jorge Lobo, Theory of disjunctive logic programs (pp. 613–639); Jeffrey F. Naughton and Raghu Ramakrishnan, Bottom-up evaluation of logic programs (pp. 640–700); E. W. Elcock, Absys, the first logic-programming language: a view of the inevitability of logic programming (pp. 701–721). The papers will not be reviewed individually.

Pym, D. and Wallen, L. (1991). Proof-search in the $\lambda\pi$-calculus. In *Logical frameworks (Sophia-Antipolis, 1990)*, pages 309–340. Cambridge Univ. Press, Cambridge.
[**MR1139789**] This item will not be reviewed individually. For details of the collection in which this item appears see 92f:68003.

Avron, A., Honsell, F., Mason, I. A., and Pollack, R. (1992). Using typed lambda calculus to implement formal systems on a machine. *J. Automat. Reason.*, 9(3):309–354.
[**MR93j:68171**] Summary: "Much research has been devoted to building computer

systems for checking proofs or for developing interactively correct proofs in specific logical systems. However, implementing a proof environment for a specific logical system is both complex and time-consuming; this—together with the proliferation of logics—suggests that a uniform and reliable alternative is desirable. One such alternative is the Edinburgh logical framework (LF), developed in the late eighties at the LFCS (Laboratory for Foundations of Computer Science). The LF is a logic-independent tool which, given a specification for a logical system, synthesizes a proof editor and checker for that system. Its specification language is based on a general theory of logics, which enables one to capture uniformities and idiosyncrasies of a large class of logics without sacrificing generality for tractability. Peculiarities (such as side conditions on rule application, variable occurrence or formula formation) are expressed at the level of the specification. In this paper we provide a broad illustration of its applicability and discuss to what extent it is successful. The analysis (of the formal presentation) of a system carried out through encoding often illuminates the system itself. This paper also deals with this phenomenon."

Bittel, O. (1992). Tableau-based theorem proving and synthesis of λ-terms in the intuitionistic logic. In *Logics in AI (Berlin, 1992)*, pages 262–278. Springer, Berlin.

[MR94f:03013] Summary: "Because of its constructive aspect, intuitionistic logic plays an important role in the context of the programming paradigm 'programming by proving'. Programs are expressed by λ-terms which can be seen as compact representations of natural deduction proofs. We present a tableau calculus for the first-order intuitionistic logic which allows us to synthesize λ-terms. The calculus is obtained from the tableau calculus for the classical logic by extending its rules by λ-terms. In each rule application and in closing of tableau branches, λ-terms are synthesized by unification. In particular, a new λ-term construct (implicit case analysis) is introduced for the disjunction rules.

"In contrast to existing approaches based on the natural deduction calculus, our calculus is very appropriate for automatic reasoning. We have implemented the calculus in Prolog. A strategy which is similar to model elimination has been built in. Several formulas (including program synthesis problems) have been proven automatically." For the entire collection see MR 94c:68005.

Kamareddine, F. (1992). A system at the cross-roads of functional and logic programming. *Sci. Comput. Programming*, 19(3):239–279.

[MR93k:68064] Summary: "The type-free λ-calculus is powerful enough to contain all the polymorphic and higher-order nature of functional programming and, furthermore, types could be constructed inside it. However, mixing the type-free λ-calculus with logic is not very straightforward (see work of Aczel and Scott). In this paper, a system that combines polymorphism and higher-order functions with logic is presented. The system is suitable for both the functional and the logical paradigms of programming, as from the functional paradigm's point of view, the system enables one to have all the polymorphism and higher order that exist in functional languages and much more. In fact, even the fixed-point operator Y which is defined as $\lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$ can be type-checked to $((\alpha \to \alpha) \to \alpha))$, where α is a

variable type. $(\lambda x.xx)(\lambda x.xx)$ can be type-checked also, something not allowed in functional languages. From the point of view of theorem proving, the system is expressive enough to allow self-referential sentences and those sentences that lead to Russell's and Curry's paradoxes. However, the paradoxes do not hold, due to the notion of circular types which contain the type of propositions. In fact, both sentences $\lambda x.xx$ and $\lambda x.xx \to\perp$ are ill-typed according to the system, because their resulting types are circular. Hence the application of either sentence to itself will not result in a proposition. The system is implemented in Milner's ML and can be seen as extending ML in two important ways. First, it extends the part related to the functional paradigm, in that it can type terms that could not be typed in ML, namely, the terms that contain self-application such as the term Y above. Second, our system extends ML by adding logic to it in a consistent way."

Miller, D. (1992). Unification under a mixed prefix. *J. Symbolic Comput.*, 14(4):321–358.

[MR93g:68121] Summary: "Unification problems are identified with conjunctions of equations between simply typed λ-terms where free variables in the equations can be universally or existentially quantified. Two schemes for simplifying quantifier alternation, called Skolemization and raising (a dual of Skolemization), are presented. In this setting, where variables of functional type can be quantified and not all types contain closed terms, the naive generalization of first-order Skolemization has several technical problems that are addressed. The method of searching for pre-unifiers described by Huet is easily extended to be mixed prefix setting, although solving flexible-flexible unification problems is undecidable since types may be empty. Unification problems may have numerous incomparable unifiers. Occasionally, unifiers share common factors and several of these are presented. Various optimizations on the general unification search problem are also discussed."

Nipkow, T. and Qian, Z. (1992). Reduction and unification in lambda calculi with subtypes. In *Automated deduction—CADE-11 (Saratoga Springs, NY, 1992)*, pages 66–78. Springer, Berlin.

[MR94i:03044] Summary: "Reduction, equality and unification are studied for a family of simply typed λ-calculi with subtypes. The subtype relation is required to relate base types only to base types and to satisfy some order-theoretic conditions. Constants are required to have a least type, i.e., 'no overloading'. We define the usual β-reduction and a subtype-dependent η-reduction. These are related to a typed equality relation and shown to be confluent in a certain sense. A generic algorithm for pre-unification modulo $\beta\eta$-conversion and an arbitrary subtype relation is presented. Furthermore it is shown that unification with respect to any subtype relation is universal." For the entire collection see MR 94f:68008.

Pym, D. (1992). A unification algorithm for the $\lambda\pi$-calculus. *Internat. J. Found. Comput. Sci.*, 3(3):333–378.

[MR94d:03025] Summary: "We present a unification algorithm for the $\lambda\Pi$-calculus, the lambda calculus with first-order dependent function types. Our algorithm is an extension of Huet's algorithm for the simply-typed lambda calculus to first-order dependent function types.

"In the simply-typed lambda calculus one attempts to unify a pair of terms of the same type; in the $\lambda\Pi$-calculus types are dependent on terms, so one must unify not only terms, but their types as well. Accordingly, we identify a suitable notion of similarity of types, and only attempt to unify a pair of terms of similar type; if the types are not similar then they are not unifiable. Since Huet's algorithm does not enumerate all of the unifiers of a given pair of terms, the strategy of first unifying pairs of types—by identifying suitable pairs of subterms for unification—is not complete. Accordingly, we must unify terms and their types simultaneously, taking care to maintain all of the conditions that are necessary to ensure the well-formedness of the resulting calculated substitution.

"Our notion of substitution is an algebraic one: substitutions are morphisms of $\lambda\Pi$-contexts. However, in order to define our algorithm we must work with psubstitutions and pcontexts—substitutions and contexts, respectively, in which variables are replaced by terms of similar, not $\beta(\eta)$-equal, type."

Qian, Z. and Wang, K. (1992). Higher-order E-unification for arbitrary theories. In *Logic programming (Washington, DC, 1992)*, pages 52–66. MIT Press, Cambridge, MA.

[MR94e:03012] Summary: "We present an algorithm consisting of three transformation rules for preunification of simply typed λ-terms with respect to α, β and η conversions and an arbitrary first-order equational theory E. The algorithm is parametrized by E-unification algorithms that admit free function symbols. It is proved that the algorithm is complete if the given E-unification algorithm is complete. This result is relevant to implementations of higher-order logic programming languages and higher-order proof systems." For the entire collection see MR 94b:68014.

Roorda, D. (1992). Proof nets for Lambek calculus. *J. Logic Comput.*, 2(2):211–231.

[MR94b:68103] Summary: "The proof nets of linear logic are adapted to the noncommutative Lambek calculus. A different criterion for soundness of proof nets is given, which gives rise to new algorithms for proof search. The order sensitivity of the Lambek calculus is reflected by the planarity condition on proof nets; this gives rise to a new nonprovability check: balance."

Wos, L. and McCune, W. (1992). The application of automated reasoning to questions in mathematics and logic. *Ann. Math. Artificial Intelligence*, 5(2-4):321–369. Artificial intelligence and mathematics, I (Fort Lauderdale, FL, 1990).

[MR96e:03012] Summary: "Sharply different from the well-known use of various computer programs for the numerical aspects of mathematics and logic is the newer and less familiar use of a program to address the logical reasoning aspects. In this article, we focus on such a program—the automated reasoning program OTTER—which is portable, available electronically by anonymous FTP, and usable on a wide variety of computers, even on personal computers. We discuss the types of assistance provided by OTTER, including proof finding, conjecture formulation, and object construction. With OTTER's assistance, we answer a number of open questions taken from a variety of fields. We focus on such questions from combinatory logic, equivalential calculus, Robbins algebra, and finite semigroup theory. For those who enjoy a challenge, we also offer ten questions, including some that are still open, and an open question that eluded even Tarski." For the entire collection see MR 96d:68001.

Curien, R. (1993). Second order E-matching as a tool for automated theorem proving. In *Progress in artificial intelligence (Porto, 1993)*, pages 242–257. Springer, Berlin.

[**MR95g:68102**] Summary: "We describe a second order matching algorithm which takes into account algebraic properties of functions. Higher order allows us the use of function variables and is relevant in many applications concerning functional or logical schemes. Adding algebraic properties significantly increases the expressiveness of the matching process and extends the set of problems having solutions. This higher order E-matching works for a set of algebraic theories E which possess certain properties. It has been implemented for the associative commutative case and adapted for similar theories such as AC1, which do not fulfill the previous requirements." For the entire collection see MR 95c:68010.

Hagiya, M. (1993). A typed λ-calculus for proving-by-example and bottom-up generalization procedure. In *Algorithmic learning theory (Tokyo, 1993)*, pages 73–86. Springer, Berlin.
[**MR95m:03022**]

It is often the case that the best way to present a proof by mathematical induction to a human reader is to prove a few steps of the induction concretely. The author presents a procedure enabling a machine to generate induction proofs from an expanded proof of the first few steps of the induction. The technique is further refined to recognize nested inductions by reconstructing inductions from examples "from the bottom up" (from inner subterms to outer subterms).

The work is carried out in an extension of logical frameworks, a system in which proofs are represented by terms in a typed λ-calculus. The typed λ-calculus is augmented with arithmetical terms, related λ-term and product term constructions, and a recursion operator. Range restrictions on arithmetical variables are expressed in the sequent calculus used. It is asserted that the extended system has desirable properties including the Church-Rosser property and strong normalization.

The algorithm for constructing inductions from examples is clearly described in the paper. In outline, it proceeds by first looking for a term which can be expressed as a recursion term (standing for a proof which has the correct form to be an expansion of the first few cases of an induction proof); then all terms which can be recognized as examples of recursions of the same general form as the one just detected are converted to recursion terms. This process is then repeated. Nested inductions can be detected when the process is repeated.

The author concludes with an example of an induction proof which can be reconstructed by his algorithm.

The paper is clearly written and interesting to read, and the results appear useful. The reviewer wonders whether there might be some fruitful interaction between this work and the work of R. S. Boyer and J. S. Moore [A computational logic, Academic Press, New York, 1979; MR 81d:68127] on automatic generation of induction proofs. For the entire collection see MR 95d:68001. Reviewed by M. Randall Holmes.

Harper, R., Honsell, F., and Plotkin, G. (1993). A framework for defining logics. *J. Assoc. Comput. Mach.*, 40(1):143–184.

[MR94d:68071] Summary: "The Edinburgh logical framework (LF) provides a means of defining (or presenting) logics. It is based on a general treatment of syntax, rules, and proofs by means of a typed λ-calculus with dependent types. Syntax is treated in a style similar to, but more general than, Martin-Löf's system of arities. The treatment of rules and proofs focuses on his notion of a judgement. Logics are represented in LF via a new principle, the judgements as types principle, whereby each judgement is identified with the type of its proofs. This allows for a smooth treatment of discharge and variable occurrence conditions and leads to a uniform treatment of rules and proofs whereby rules are viewed as proofs of higher-order judgments and proof checking is reduced to type checking. The practical benefit of our treatment of formal systems is that logic-independent tools, such as proof editors and proof checkers, can be constructed."

Kohlhase, M. and Pfenning, F. (1993). Unification in a λ-calculus with intersection types. In *Logic programming (Vancouver, BC, 1993)*, pages 488–505. MIT Press, Cambridge, MA.

[MR96d:03026] Summary: "We propose related algorithms for unification and constraint simplification in $\lambda^{\rightarrow\&}$, a refinement of the simply-typed λ-calculus with subtypes and bounded intersection types. $\lambda^{\rightarrow\&}$ is intended as the basis of a logical framework in order to achieve more succinct and declarative axiomatizations of deductive systems than possible with the simply-typed λ-calculus. The unification and constraint simplification algorithms described here lay the groundwork for a mechanization of such frameworks as constraint logic programming languages and theorem provers." For the entire collection see MR 95e:68006.

Parigot, M. (1993). Classical proofs as programs. In *Computational logic and proof theory (Brno, 1993)*, pages 263–276. Springer, Berlin.

[MR94m:68024] Summary: "We present an extension of the correspondence between intuitionistic proofs and functional programs to classical proofs, and more precisely to second order classical proofs. The advantage of classical logic in this context is that it allows one to model imperative features of programming languages as well. But there is an intrinsic difficulty with classical logic which lies in a certain nondeterminism of its computational interpretations. The use of a natural deduction system removes a part of this nondeterminism by fixing the inputs to the left of the sequents. However a conflict remains between the confluence of the computation mechanism and the uniqueness of the representation of data (for instance the uniqueness of the representation of the natural number 1). In this paper we develop the

solution to this problem proposed in another paper of ours [in Logic programming and automated reasoning (St. Petersburg, 1992), 190–201, Lecture Notes in Comput. Sci., 624, Springer, Berlin, 1992; MR 94g:03029]: we show how to extract the intuitionistic representation of data from a classical one using an 'output' operator, while keeping a confluent computation mechanism. This result allows us to extend in a sound way the proofs-as-programs paradigm to classical proofs in a framework where all the usual theoretical properties of intuitionistic proofs still hold." For the entire collection see MR 94f:03003.

Andrews, P. B., Bishop, M., Issar, S., Nesmith, D., Pfenning, F., and Xi, H. W. (1994). TPS: an interactive and automatic tool for proving theorems of type theory. In *Higher order logic theorem proving and its applications (Vancouver, BC, 1993)*, pages 366–370. Springer, Berlin.
[MR1283843] This item will not be reviewed individually. For details of the collection in which this item appears see 95b:68012.

Coquand, T. and Herbelin, H. (1994). A-translation and looping combinators in pure type systems. *J. Funct. Programming*, 4(1):77–88.
[MR95j:68092] From the text: "Consider a pure type system (PTS) extending $\lambda 2$ [see H. P. Barendregt, J. Funct. Programming 1 (1991), no. 2, 125–154; MR 93e:03019]. Under mild conditions concerning the PTS in question the paper gives an algorithm that transforms any proof M of a contradiction (i.e. such that $\vdash M : \forall\alpha.\alpha$) into a looping combinator, i.e. a term Y' such that $\vdash Y' : \forall\alpha.(\alpha \to \alpha) \to \alpha$ and $\mathrm{BT}(Y') = \mathrm{BT}(Y)$, where Y is the usual fixed point combinator, and $\mathrm{BT}(P)$ is the Böhm tree of lambda term P. This extends a previous result [D. J. Howe, in Proceedings of the Symposium on Logic in Computer Science (Ithaca, NY, 1987), 205–214, IEEE Comput. Soc. Press, Washington, DC, 1987; CCA 1987:58946] in which a concrete proof of a contradiction in a particular PTS is translated with the help of a computer into a looping combinator."

de Bruijn, N. G. (1994a). AUT-QE without type inclusion. In *Selected papers on Automath*, pages 289–297. North-Holland, Amsterdam.
[MR1429408] This item will not be reviewed individually. For details of the collection in which this item appears see 98g:03038.

de Bruijn, N. G. (1994b). AUT-SL, a single line version of Automath. In *Selected papers on Automath*, pages 275–281. North-Holland, Amsterdam.
[MR1429406] This item will not be reviewed individually. For details of the collection in which this item appears see 98g:03038.

de Bruijn, N. G. (1994c). Computer program semantics in space and time. In *Selected papers on Automath*, pages 947–972. North-Holland, Amsterdam.
[MR1429431] This item will not be reviewed individually. For details of the collection in which this item appears see 98g:03038.

de Bruijn, N. G. (1994d). Example of a text written in Automath. In *Selected papers on Automath*, pages 687–700. North-Holland, Amsterdam.
[MR1429419] This item will not be reviewed individually. For details of the collection in which this item appears see 98g:03038.

de Bruijn, N. G. (1994e). Generalizing Automath by means of a lambda-typed lambda calculus [MR 89b:03026]. In *Selected papers on Automath*, pages 313–337. North-Holland, Amsterdam.
[MR1429411] This item will not be reviewed individually. For details of the collection in which this item appears see 98g:03038.

de Bruijn, N. G. (1994f). Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the Church-Rosser theorem [MR **48** #71]. In *Selected papers on Automath*, pages 375–388. North-Holland, Amsterdam.
[MR1429414] This item will not be reviewed individually. For details of the collection in which this item appears see 98g:03038.

de Bruijn, N. G. (1994g). The mathematical language Automath, its usage, and some of its extensions [MR **42** #9094]. In *Selected papers on Automath*, pages 73–100. North-Holland, Amsterdam.
[MR1429398] This item will not be reviewed individually. For details of the collection in which this item appears see 98g:03038.

de Bruijn, N. G. (1994h). Reflections on Automath. In *Selected papers on Automath*, pages 201–228. North-Holland, Amsterdam.
[MR1429403] This item will not be reviewed individually. For details of the collection in which this item appears see 98g:03038.

de Bruijn, N. G. (1994i). Some extensions of Automath: the AUT-4 family. In *Selected papers on Automath*, pages 283–288. North-Holland, Amsterdam.
[MR1429407] This item will not be reviewed individually. For details of the collection in which this item appears see 98g:03038.

de Bruijn, N. G. (1994j). A survey of the project Automath [MR 81m:03017]. In *Selected papers on Automath*, pages 141–161. North-Holland, Amsterdam.
[MR1429401] This item will not be reviewed individually. For details of the collection in which this item appears see 98g:03038.

Eisenbiegler, D., Schneider, K., and Kumar, R. (1994). A functional approach for formalizing regular hardware structures. In *Higher order logic theorem proving and its applications (Vancouver, BC, 1993)*, pages 101–114. Springer, Berlin.
[MR1283829] This item will not be reviewed individually. For details of the collection in which this item appears see 95b:68012.

Gordon, A. D. (1994). A mechanisation of name-carrying syntax up to alpha-conversion. In *Higher order logic theorem proving and its applications (Vancouver, BC, 1993)*, pages 413–425. Springer, Berlin.
[**MR1283846**] This item will not be reviewed individually. For details of the collection in which this item appears see 95b:68012.

Goubault, J. (1994). Higher-order rigid E-unification. In *Logic programming and automated reasoning (Kiev, 1994)*, pages 129–143. Springer, Berlin.
[**MR95j:03020**] Summary: "Higher-order E-unification, i.e. the problem of finding substitutions that make two simply typed λ-terms equal modulo β- or $\beta\eta$-equivalence and a given equational theory, is undecidable. We propose to rigidify it, to get a resource-bounded decidable unification problem (with arbitrarily high bounds), providing a complete higher-order E-unification procedure. The techniques are inspired by Gallier's rigid E-unification and by Dougherty and Johann's use of combinatory logic to solve higher-order E-unification problems. We improve their results by using general equational theories, and by defining optimizations such as higher-order rigid E-preunification, where flexible terms are used, gaining much efficiency, as in the non-equational case due to Huet." For the entire collection see MR 95g:68002.

Huet, G. (1994). Residual theory in λ-calculus: a formal development. *J. Funct. Programming*, 4(3):371–394.
[**MR96f:03011**] Summary: "We present the complete development, in Gallina, of the residual theory of β-reduction in pure λ-calculus. The main result is the prism theorem, and its corollary Levy's cube lemma, a strong form of the parallel-moves lemma, itself a key step towards the confluence theorem and its usual corollaries (Church-Rosser, uniqueness of normal forms). Gallina is the specification language of the Coq proof assistant. It is a specific concrete syntax for its abstract framework, the calculus of inductive constructions [C. Paulin-Mohring, in Typed lambda calculi and applications (Utrecht, 1993), 328–345, Lecture Notes in Comput. Sci., 664, Springer, Berlin, 1993; MR 94e:68111]. It may be thought of as a smooth mixture of higher-order predicate calculus with recursive definitions, inductively defined data types and inductive predicate definitions reminiscent of logic programming. The development presented here was fully checked in the current distribution version Coq V5.8. We just state the lemmas in the order in which they are proved, omitting the proof justifications. The full transcript is available as a standard library in the distribution of Coq."

Johann, P. and Kohlhase, M. (1994). Unification in an extensional lambda calculus with ordered function sorts and constant overloading. In *Automated deduction—CADE-12 (Nancy, 1994)*, pages 620–634. Springer, Berlin.
[**MR96d:68186**] Summary: "We develop an order-sorted higher-order calculus suitable for automatic theorem proving applications by extending the extensional simply typed lambda calculus with a higher-order ordered sort concept and constant overloading. Huet's well-known techniques for unifying simply typed lambda terms are

generalized to arrive at a complete transformation-based unification algorithm for this sorted calculus. Consideration of an order-sorted logic with functional base sorts and arbitrary term declarations was originally proposed by the second author in a 1991 paper; we give here a corrected calculus which supports constant rather than arbitrary term declarations, as well as a corrected unification algorithm, and prove in this setting results corresponding to those claimed there." For the entire collection see MR 95m:68004.

Kesner, D. (1994). Reasoning about layered, wildcard and product patterns. In *Algebraic and logic programming (Madrid, 1994)*, pages 253–268. Springer, Berlin.

[**MR95k:68126**] Summary: "We study the extensional version of the simply typed λ-calculus with product types and fixpoints enriched with layered, wildcard and product patterns. Extensionality is expressed by the surjective pairing axiom and a generalization of the η-conversion to patterns. We obtain a confluent reduction system by turning the extensional axioms into expansion rules, and then adding some restrictions to these expansions in order to avoid reduction loops. Confluence is proved by composition of modular properties of the extensional and non-extensional subsystems of the reduction calculus." For the entire collection see MR 95j:68002.

Lugiez, D. (1994). Higher order disunification: some decidable cases. In *Constraints in computational logics (Munich, 1994)*, pages 121–135. Springer, Berlin.

[**MR97g:03014**] Summary: "This paper deals with higher-order disunification, i.e., solving constraints on simply typed lambda-terms involving the $\eta\beta$ equality, negation, the logical connectives $\wedge$ and $\vee$, and universal and existential quantification. Contrary to the first-order case, higher-order disunification is undecidable, and not even semidecidable since it contains higher-order unification and its negation. Therefore we merely give general rules to simplify the basic constraints called equational problems and we use these rules to obtain decision procedures in some useful cases. First we prove that the second-order top-linear complement problem and other special cases of complement problems are decidable. Then we consider the case when all the terms involved in the constraints are patterns, i.e., when the arguments of a free variable are distinct bound variables. In this case, we can devise new rules to extend the initial set of rules, and we are able to prove the decidability of the general case of disunification constraints, involving mixed quantification." For the entire collection see MR 97f:68003.

McIsaac, A. (1994). A formalization of abstraction in LAMBDA. In *Higher order logic theorem proving and its applications (Vancouver, BC, 1993)*, pages 227–238. Springer, Berlin.

[**MR95c:68156**] Summary: "In a mixed approach to system verification using theorem provers with an interface to specialized model-checking tools, it may be necessary to simplify models by considering abstract versions of them. We report on work in progress that aims to develop support within LAMBDA for a systematic approach to abstraction. We give a formalization in LAMBDA of a notion of abstraction for transition systems; the abstract systems have two sorts of transitions, and are related to specifications in modal process logic. We prove that formulae in the modal mu-calculus are satisfied in an abstract version of a model only if they are satisfied in the model itself. We illustrate how the proof of an inductive step in the verification of a satisfaction relation for an infinite model can be reduced to the verification of a satisfaction relation for a very small finite model." For the entire collection see MR 95b:68012.

Nederpelt, R. P. and Geuvers, J. H. (1994). Twenty-five years of Automath research. In *Selected papers on Automath*, pages 3–54. North-Holland, Amsterdam.

[**MR1429396**] This item will not be reviewed individually. For details of the collection in which this item appears see 98g:03038.

Nederpelt, R. P., Geuvers, J. H., and de Vrijer, R., editors (1994). *Selected papers on Automath*. North-Holland Publishing Co., Amsterdam.

[**MR98g:03038**] Contents: R. P. Nederpelt and J. H. Geuvers, Twenty-five years of Automath research (3–54); N. G. de Bruijn, Verification of mathematical proofs by a computer: a preparatory study for a project Automath (57–72); N. G. de Bruijn, The mathematical language Automath, its usage, and some of its extensions [MR 42 #9094] (73–100); D. T. van Daalen, A description of Automath and some aspects of its language theory (101–126); J. Zucker [Jeffery Zucker], Formalization of classical mathematics in Automath (127–139); N. G. de Bruijn, A survey of the project Automath [MR 81m:03017] (141–161); D. T. van Daalen, The language theory of Automath: Chapter I, Sections 1–5 (Introduction) [MR 81h:68092] (163–200); N. G. de Bruijn, Reflections on Automath (201–228); R. P. Nederpelt, Type systems—basic ideas and applications (229–247).

L. S. van Benthem Jutting, Description of AUT-68 (251–273); N. G. de Bruijn, AUT-SL, a single line version of Automath (275–281); N. G. de Bruijn, Some extensions of Automath: the AUT-4 family (283–288); N. G. de Bruijn, AUT-QE without type inclusion (289–297); L. S. van Benthem Jutting, Checking Landau's "Grundlagen" in the Automath system: Appendix 9 (AUT-SYNT) [MR 58 #32124ab] (299–301); D. T. van Daalen, The language theory of Automath: Chapter VIII, 1 and 2 (AUT-Π) [MR 81h:68092] (303–312); N. G. de Bruijn, Generalizing Automath by means of a lambda-typed lambda calculus [MR 89b:03026] (313–337); H. Balsters, Lambda calculus extended with segments: Chapter 1, Sections 1.1 and 1.2 (Introduction) (339–367).

L. S. van Benthem Jutting, A normal form theorem in a λ-calculus with types (371–374); N. G. de Bruijn, Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the Church-Rosser theorem [MR 48 #71] (375–388); R. P. Nederpelt, Strong normalization in a typed lambda calculus with lambda structured types (389–468); R. C. de Vrijer, Big trees in a λ-calculus with λ-expressions as types [MR 58 #5085] (469–492); D. T. van Daalen, The language theory of Automath: parts of Chapters II, IV, V–VIII [MR 81h:68092] (493–653); L. S. van Benthem Jutting, The language theory of Λ_∞, a typed λ-calculus where terms are types (655–683); N. G. de Bruijn, Example of a text written in Automath (687–700); L. S. van Benthem Jutting, Checking Landau's "Grundlagen" in the Automath system: parts of Chapters 0, 1 and 2 (Introduction, Preparation, Translation) [MR 58 #32124ab] (701–720); L. S. van Benthem Jutting, Checking Landau's "Grundlagen" in the Automath system: Chapter 4 (Conclusions) [MR 58

#32124ab] (721–732); L. S. van Benthem Jutting and R. C. de Vrijer, A text fragment from Zucker's "Real analysis" (733–761); L. S. van Benthem Jutting, Checking Landau's "Grundlagen" in the Automath system: Appendices 3 and 4 (The PN-lines; Excerpt for [from] "Satz 27") [MR 58 #32124ab] (763–780).

I. Zandleven, A verifying program for Automath (783–804); L. S. van Benthem Jutting, Checking Landau's "Grundlagen" in the Automath system: parts of Chapter 3 (Verification) [MR 58 #32124ab] (805–808); L. S. van Benthem Jutting, An implementation of substitution in a λ-calculus with dependent types (809–837); N. G. de Bruijn, Set theory with type restrictions [MR 53 #2615] (841–847); N. G. de Bruijn, Formalization of constructivity in Automath (849–864); N. G. de Bruijn, The mathematical vernacular, a language for mathematics with typed sets (865–935); R. M. A. Wieringa, Relational semantics in an integrated system (937–946); N. G. de Bruijn, Computer program semantics in space and time (947–972).

This book is a selection of papers on the Automath project, which was started by N. G. de Bruijn about 1967 and officially ended in 1976.

The Automath project belongs to the general area of automated reasoning (in which it was one of the earlier projects). It is distinctive in that its product was not a piece of software but a language. Automath is a formal language designed to facilitate the completely formal expression of mathematics; Automath texts can be and have been verified by computer programs; however, the effort in the Automath project was not lavished on software, but on the formal system(s) in which reasoning was to be expressed.

Automath as a formal system has a distinctive flavor. Typed λ-calculus is used to represent both the objects that mathematics is about and the proofs of mathematical propositions. De Bruijn independently discovered the "propositions as types" (he prefers to say "proofs as objects") paradigm usually called the "Curry-Howard isomorphism" between typed λ-calculus and proofs in intuitionistic propositional and predicate calculus. Automath is not committed to constructive logic (as the related system Coq is, for example) because a distinction is maintained between types of objects and types of proofs. De Bruijn formally enunciated a principle of "irrelevance of proofs" which allows classical mathematical reasoning to be carried on in Automath: the important thing about a type of proofs is whether it is inhabited, and for purposes of classical reasoning it is convenient to postulate that any two proofs of the same proposition are equal.

A mathematical text in Automath has a very definite structure. The Automath text is called a "book" and is made up of "lines", each of which has the same four components: a context, an identifier, an expression and a type.

The context will either be empty (0) or the identifier of a previous line. The identifier of each line is (in principle) unique to that line. The expression tells us how the identifier is to be understood. The type tells us what kind of object the expression is.

There are three kinds of lines. There are lines which introduce primitive notions, in which the expression is PN. There are lines which introduce "arbitrary" objects (as in "let x be a real number"); such lines add material to the context, and they have the expression —. Finally, there are lines which are effectively definitions of their identifiers; these have expressions which are mathematical expressions in the normal sense, and the mathematical content to check in such lines is that their type is appropriate.

We describe a very small Automath book. We will separate the four components of an Automath line by semicolons. The first part of our book is

$0; a; -$;real

$a; b; -$;real

b; prod;PN;real.

This book translates into English as "Let a be a real number. Let b be a real number. We introduce a primitive operation prod such that prod(a, b) is a real number." The reason that we can tell that prod depends on a and b is that its context includes a and b (line b is its given context, which in turn has line a as its context).

Another very short Automath book illustrates an application of the proofs as types paradigm:

$0; p; -$;prop

$p; q; -$;prop

q;imp;PN;prop

$q; a; -; p$

$a; b; -$;imp(p, q)

b;mp;$PN; q$

In English translation, this reads "Let p be a proposition. Let q be a proposition. Then there is a primitive operation imp (implication) yielding a proposition imp(p, q). Let a be an object of type p (a proof of the proposition p). Let b be an object of type imp(p, q) (a proof of the proposition 'p implies q'). A primitive operation mp (modus ponens) gives us a proof mp(p, q, a, b) of q."

An easy proof follows.

$0; p; -$; prop

$p; q; -$; prop

$q; r; -$; prop

$r; a; -; p$

$a; b; -$; imp(p, q)

$b; c; -$; imp(q, r)

$c; A$; mp$(p, q, a, b); q$

$d; B$; mp$(q, r, A, c); r$

The English translation reads: "Let p, q, r be propositions. Let a, b, c be proofs of p, imp(p, q), and imp(q, r), respectively. $A =$ mp(p, q, a, b) will be a proof of q; $B =$ mp(q, r, A, c) will be a proof of r." An Automath type checker would be able to verify that the expressions A and B are well-formed and have the indicated types.

Notice that the introductions of propositions and the definition of imp are actually type definitions (because a proposition is a type inhabited by proofs): prop is a "supertype" (type of types). The other basic supertype in Automath is the "type" of types of mathematical objects, called 'type': 0; real; PN; type would be a line introducing the type of real numbers. Most Automath systems admit just three "degrees" of object: objects and proofs are degree 3, types and propositions are degree 2, and type and prop (and possibly some other supertypes) are degree 1. Only objects of degrees 3 and 2 can be introduced or defined by lines.

The device of context allows the introduction of functions, as was seen in our examples; functions can also be introduced explicitly in λ-notation. Functions may have dependent types: the type of the output of a function may depend on its argument. This is needed, for example, in the typing of proofs of statements with universal quantifiers: a proof of $(\forall x: \tau.P(x))$ is a function which takes each object x of type τ to a proof of $P(x)$; recall that the proposition $P(x)$ is the type of its proofs. A peculiarity of the notation of Automath is that superficially identical notation is used for a function $[x: \sigma]t(x)$, where $t(x)$ may have type $\tau(x)$ depending on the argument x, and the type of this function, which is written $[x: \sigma]\tau(x)$; more usual notation would have $(\lambda x: \sigma.t(x))$ for the function and $(\Pi x: \sigma.t(x))$ for the type. Another peculiarity of Automath notation is that the order of function application is reversed: $\langle x\rangle f$ is preferred to the usual $f(x)$.

The possibility of translation of classical mathematical texts into Automath has been demonstrated in practice; a substantial monograph on analysis by Landau was translated into Automath and mechanically checked. A telling point is that the compression factor (the ratio between the length of Landau's text and the length of corresponding parts of the Automath translation) was constant throughout the book; the translation did not have to become more verbose as the level of difficulty of the material in the book progressed. An original text in analysis was later developed by Zucker entirely in Automath. These developments, and their practical impact on Automath text format and checkers, are discussed in this volume.

One reason why the "product" of the Automath project is a formal language rather than a piece of software is that the project was not devoted to "automated theorem proving", but to the facilitation of automated checking of proofs produced by mathematicians. Another reason why the elaboration of software was not the focus was that the language incorporated very little strength; even the definitions of such concepts as logical negation and conjunction are carried out in "books" rather than built into the system. The logic of Automath is the minimal constructive logic of implication and universal quantification, when viewed from the standpoint of the Curry-Howard isomorphism; the only type construction which is built in is dependent typing of functions (of which the usual "arrow types" of functions are a special case). The development of mathematics on this foundation is rather different from the usual development in terms of ZFC, and has theoretical interest independently of computer-related applications.

Many other things are to be found in this book. One may find the original reference for "de Bruijn indices", the device for replacing bound variables in λ-notation with numerical "nameless dummies" now used in computer science. There are formal language definitions and proofs of the Church-Rosser property and strong normalization for versions of Automath and other Automath-like systems of typed λ-calculus (the bulk of the book is devoted to such technical mathematical issues). One may find good expositions of the difference between approaches to foundations based on set theory and approaches based on type theory, and plausible exposition of the position that approaches based on type theory of functions are closer to the actual practice of mathematicians than the "officially" favored foundations using set theory. There are references to the application of Automath or Automath-like systems to mathematics pedagogy, which excited the curiosity of the reviewer. There is a description of a computer program for checking the validity of Automath texts. The reviewer is certain that he has not exhausted the contents with this description.

The book should be of interest to any worker in automated reasoning; it should also be of interest to any logician with technical interests in type systems and λ-calculi. Some parts of it should be interesting to the philosopher of mathematics.

The critical apparatus of the volume is good; the editors have supplied a detailed discussion of what is to be found in each paper in the volume in a lengthy "survey of the contents", which the reviewer found useful as a guide. There is also a discussion of the similarities and differences between Automath and other projects in automated

reasoning, and of how it has influenced subsequent developments. Reviewed by M. Randall Holmes.

Petersen, K. D. (1994). Graph model of LAMBDA in higher order logic (progress report). In *Higher order logic theorem proving and its applications (Vancouver, BC, 1993)*, pages 16–28. Springer, Berlin.
[MR1283823] This item will not be reviewed individually. For details of the collection in which this item appears see 95b:68012.

Prehofer, C. (1994). Decidable higher-order unification problems. In *Automated deduction—CADE-12 (Nancy, 1994)*, pages 635–649. Springer, Berlin.
[MR96d:68189] Summary: "Second-order unification is undecidable in general. Miller showed that unification of so-called higher-order patterns is decidable and unitary. We show that the unification of a linear higher-order pattern s with an arbitary second-order term that shares no variables with s is decidable and finitary. A few extensions of this unification problem are still decidable: unifying two second-order terms, where one term is linear, is undecidable if the terms contain bound variables but decidable if they do not." For the entire collection see MR 95m:68004.

Shankar, N. (1994). *Metamathematics, machines, and Gödel's proof.* Cambridge University Press, Cambridge.
[MR95i:03002] This book is about how computers can be used to construct and verify the correctness of mathematical proofs. More specifically, the book describes the use of a specific computer program, the Boyer-Moore theorem prover, in giving a full verification of the proofs of Gödel's celebrated first incompleteness theorem (for a theory of hereditarily finite sets essentially equivalent to Peano arithmetic), and of the Church-Rosser theorem of λ-calculus (based on the proof of Tait and Martin-Löf). These proofs were constructed through an interaction with the theorem prover in which the user supplied the machine with a well-chosen series of lemmas and definitions (over 2 000 lemmas and definitions in the case of Gödel's incompleteness theorem) and the automated theorem prover was able to prove a large number of nontrivial lemmas. The Boyer-Moore theorem prover is one of the best-known, non-resolution based, heuristic theorem provers. It can be employed to prove properties of recursively defined functions (presented in the formalism of pure LISP) by automatically discovering the induction scheme required for a proof or the sublemmas which might turn out to be useful for a given goal. The theorem prover generates as output a description of its attempted proof, which makes it possible to uncover errors in the formulation or to formulate hints for the proof. The experiments reported in this book are quite convincing in demonstrating that proofs of difficult metamathematical results can be formalized up to the point of making a machine verification possible, without an enormous increase in the size of the proof. In the case of the Church-Rosser theorem the automated proof is particularly efficient: it takes only 4 624 words, as compared with the 3 476 words of the informal proof. These estimates also include the basic definitions of λ-calculus. It is argued

that experiments with metamathematical results might have important implications for automated reasoning, due to the possibility of making an efficient use of derived inference rules to enhance the deductive ability of the program. The book ends with a qualitative analysis of the limitations of the theorem prover. One of these is the absence of quantifiers in its underlying logic. The theorem prover can only prove conjectures expressed in a quantifier-free way. Statements of the form $\forall x \, \exists y \, A(x, y)$, such as the Church-Rosser theorem, must be reformulated in the form $A(x, f(x))$, where f is a recursively defined function supplied by the user. Reviewed by Alessandro Berarducci.

van Benthem Jutting, L. S. (1994a). *Checking Landau's "Grundlagen" in the Automath system*: Appendix 9 (AUT-SYNT) [MR **58** #32124ab]. In *Selected papers on Automath*, pages 299–301. North-Holland, Amsterdam.
[**MR1429409**] This item will not be reviewed individually. For details of the collection in which this item appears see 98g:03038.

van Benthem Jutting, L. S. (1994b). Description of AUT-68. In *Selected papers on Automath*, pages 251–273. North-Holland, Amsterdam.
[**MR1429405**] This item will not be reviewed individually. For details of the collection in which this item appears see 98g:03038.

van Benthem Jutting, L. S. (1994c). An implementation of substitution in a λ-calculus with dependent types. In *Selected papers on Automath*, pages 809–837. North-Holland, Amsterdam.
[**MR1429426**] This item will not be reviewed individually. For details of the collection in which this item appears see 98g:03038.

van Daalen, D. T. (1994a). *The language theory of Automath*: Chapter I, Sections 1–5 (Introduction) [MR 81h:68092]. In *Selected papers on Automath*, pages 163–200. North-Holland, Amsterdam.
[**MR1429402**] This item will not be reviewed individually. For details of the collection in which this item appears see 98g:03038.

van Daalen, D. T. (1994b). *The language theory of Automath*: Chapter VIII, 1 and 2 (AUT-π) [MR 81h:68092]. In *Selected papers on Automath*, pages 303–312. North-Holland, Amsterdam.
[**MR1429410**] This item will not be reviewed individually. For details of the collection in which this item appears see 98g:03038.

van Daalen, D. T. (1994c). *The language theory of Automath*: parts of Chapters II, IV, V–VIII [MR 81h:68092]. In *Selected papers on Automath*, pages 493–653. North-Holland, Amsterdam.
[**MR1429417**] This item will not be reviewed individually. For details of the collection in which this item appears see 98g:03038.

Wieringa, R. M. A. (1994). Relational semantics in an integrated system. In *Selected papers on Automath*, pages 937–946. North-Holland, Amsterdam.

[**MR1429430**] This item will not be reviewed individually. For details of the collection in which this item appears see 98g:03038.

Coscoy, Y., Kahn, G., and Théry, L. (1995). Extracting text from proofs. In *Typed lambda calculi and applications (Edinburgh, 1995)*, pages 109–123. Springer, Berlin.

[**MR1477978**] This item will not be reviewed individually. For details of the collection in which this item appears see 98e:03020.

Despeyroux, J., Felty, A., and Hirschowitz, A. (1995). Higher-order abstract syntax in Coq. In *Typed lambda calculi and applications (Edinburgh, 1995)*, pages 124–138. Springer, Berlin.

[**MR1477979**] This item will not be reviewed individually. For details of the collection in which this item appears see 98e:03020.

Dezani-Ciancaglini, M. and Plotkin, G., editors (1995). *Typed lambda calculi and applications*, Berlin. Springer-Verlag.

[**MR98e:03020**] Contents: Andrea Asperti and Cosimo Laneve, Comparing λ-calculus translations in sharing graphs (1–15); Gilles Barthe, Extensions of pure type systems (16–31); Roberto Bellucci, Martin Abadi and Pierre-Louis Curien, A model for formal parametric polymorphism: a PER interpretation for system $\mathcal{R}$ (32–46); Stefano Berardi, Marc Bezem and Thierry Coquand, A realization of the negative interpretation of the axiom of choice (47–62); S. Berardi and L. Boerio, Using subtyping in program optimization (63–77); G. M. Bierman, What is a categorical model of intuitionistic linear logic? (78–93); Daniel Briaud, An explicit Eta rewrite rule (94–108); Yann Coscoy, Gilles Kahn and Laurent Thery, Extracting text from proofs (109–123); Joelle Despeyroux, Amy Felty and Andre Hirschowitz, Higher-order abstract syntax in Coq (124–138); Roberto Di Cosmo and Adolfo Piperno, Expanding extensional polymorphism (139–153); Gilles Dowek, Lambda-calculus, combinators and the comprehension scheme (154–170); Neil Ghani, $\beta\eta$-equality for coproducts (171–185); Healfdene Goguen, Typed operational semantics (186–200); Philippe de Groote, A simple calculus of exception handling (201–215); Martin Hofmann, A simple model for quotient types (216–234); M. Randall Holmes, Untyped λ-calculus with relative typing (235–248); Furio Honsell and Marina Lenisa, Final semantics for untyped λ-calculus (249–265); Antonius J. C. Hurkens, A simplification of Girard's paradox (266–278); Hidetaka Kondoh, Basic properties of data types with inequational refinements (extended abstract) (279–296); Toshihiko Kurata and Masako Takahashi, Decidable properties of intersection type systems (297–311); Francois Leclerc, Termination proof of term rewriting system with the multiset path ordering. A complete development in the system Coq (312–327); Paul-Andre Mellies, Typed λ-calculi with explicit substitutions may not terminate (328–334); Vincent Padovani, On equivalence classes of interpolation equations (335–349); Jaco van de Pol and

Helmut Schwichtenberg, Strict functionals for termination proofs (350–364); Robert Pollack, A verified typechecker (365–380); A. Pravato, S. Ronchi della Rocca and L. Roversi, Categorical semantics of the call-by-value λ-calculus (381–396); Eike Ritter and Andrew M. Pitts, A fully abstract translation between a λ-calculus with reference types and standard ML (397–413); Alex K. Simpson, Categorical completeness results for the simply-typed lambda-calculus (414–427); Jan Springintveld, Third-order matching in the presence of type constructors (extended abstract) (428–442). Some of the papers are being reviewed individually.

Hagiya, M. (1995). A typed λ-calculus for proving-by-example and bottom-up generalization procedure. *Theoret. Comput. Sci.*, 137(1):3–23. Special issue on algorithmic learning theory (Tokyo, 1993).
[MR95m:03023] A previous version of this paper has been reviewed [in Algorithmic learning theory (Tokyo, 1993), 73–86, Lecture Notes in Comput. Sci., 744, Springer, Berlin, 1993; see the preceding review]. For the entire collection see MR 95m:68007.

Jech, T. (1995). OTTER experiments in a system of combinatory logic. *J. Automat. Reason.*, 14(3):413–426.
[MR96c:03032] Summary: "This paper describes some experiments involving the automated theorem-proving program OTTER in the system TRC of illative combinatory logic. We show how OTTER can be steered to find a contradiction in an inconsistent variant of TRC, and present some experimentally discovered identities in TRC."

Manoury, P. (1995). A user's friendly syntax to define recursive functions as typed λ-terms. In *Types for proofs and programs (Båstad, 1994)*, pages 83–100. Springer, Berlin.
[MR1456760] This item will not be reviewed individually. For details of the collection in which this item appears see 97m:68008.

Padovani, V. (1995). On equivalence classes of interpolation equations. In *Typed lambda calculi and applications (Edinburgh, 1995)*, pages 335–349. Springer, Berlin.
[MR1477993] This item will not be reviewed individually. For details of the collection in which this item appears see 98e:03020.

Pollack, R. (1995). A verified typechecker. In *Typed lambda calculi and applications (Edinburgh, 1995)*, pages 365–380. Springer, Berlin.
[MR1477995] This item will not be reviewed individually. For details of the collection in which this item appears see 98e:03020.

Saïbi, A. (1995). Formalization of a λ-calculus with explicit substitutions in Coq. In *Types for proofs and programs (Båstad, 1994)*, pages 183–202. Springer, Berlin.
[MR1456765] This item will not be reviewed individually. For details of the collection in which this item appears see 97m:68008.

Yamada, S. (1995). Computer-aided synthetic reasoning in smooth infinitesimal calculus. In *Proceedings of the 5th International Colloquium on Differential Equations, Vol. 2 (Plovdiv, 1994)*, pages 258–267. Sci. Cult. Technol. Publ., Singapore.

[MR1652232] A review for this item is in process.

Brown, J. and Wallen, L. A. (1996). Representing unification in a logical framework. In *Computer science logic (Paderborn, 1995)*, pages 106–129. Springer, Berlin.

[MR1461872] This item will not be reviewed individually. For details of the collection in which this item appears see 98a:68002.

Gordon, A. D. and Melham, T. (1996). Five axioms of alpha-conversion. In *Theorem proving in higher order logics (Turku, 1996)*, pages 173–190. Springer, Berlin.

[MR1470250] This item will not be reviewed individually. For details of the collection in which this item appears see 98c:68024.

Qian, Z. (1996). Unification of higher-order patterns in linear time and space. *J. Logic Comput.*, 6(3):315–341.

[MR98h:03017] Summary: "Higher-order patterns are simply typed λ-terms in long β-normal form where the arguments of a free variable are always η-equal to distinct bound variables. It has been proved that unification of higher-order patterns modulo α, β and η reductions in the simply typed λ-calculus is decidable and that unifiable higher-order patterns have a most general unifier. In this paper a unification algorithm for higher-order patterns is presented, whose time and space complexities are proved to be linear in the size of the input."

Elbers, H. J. (1998). *Connecting informal and formal mathematics.* Technische Universiteit Eindhoven, Eindhoven. Dissertation, Technische Universiteit Eindhoven, Eindhoven, 1998.

[MR1631531] From the introduction: "The main contribution of this thesis is the description of a formal mathematical language, which we call $\lambda\mathrm{HOL}_\pi$, based on type theory in which executable functions can be defined by pattern matching. The pattern matching formalism of this formal language can be used for reasoning by cases, and enables the development of certified proving procedures. We implement a program called LEGO with pattern matching that can verify the correctness of expressions in this formal language.

"In Chapter 2 we describe how we can formalize computations. First we introduce a very general model for computations on objects whose structure is unknown, called abstract reduction systems. Then we present term rewriting systems as a general model for computations in functional programming languages. A term rewriting system is an abstract reduction system that rewrites objects with a term structure.

"In Chapter 3 we describe how we can represent logic in type theory. First we introduce Higher Order Logic, a typed version of λ-calculus, and illustrate how mathematics can be formalized in this language. Then we describe the fundamental properties of this formal language that make it suited for automated verification.

"In Chapter 4 we discuss the role of computations in formalizing mathematics. First we present several methods for using external computations (that are not done in the

formal language) to make the construction of formal proofs easier. Then we discuss two formalisms for representing computations in the formal language.

"In Chapter 5 we illustrate the elegance of the pattern matching formalism of functional programming languages for defining functions. Then we introduce constructor systems as term rewriting systems with two kinds of function symbols: constructor symbols (without a computational meaning) and defined symbols (with a computational meaning). This separation of defined symbols and constructor symbols gives constructor systems a clear semantics that is simpler than the semantics of (the more general) term rewriting systems. Next we introduce priority constructor systems that use an order on rules to determine which rule may be applied. Priority rewriting can be used to disambiguate term rewrite systems with overlapping rules and to specify an exhaustive function by a few rules. Finally we present an algorithm that transforms priority constructor systems to equivalent constructor systems.

"In Chapter 6 we describe the problems that are involved in the extension of the formal language of Higher Order Logic with pattern matching and how these problems can be solved. After giving a formal definition of this extension we prove several fundamental properties of this type system. For instance, we show that the result of the application of a function defined by pattern matching to its arguments is unique. Next we analyse the relation between pattern matching and inductive types and present an algorithm that maps functions defined by pattern matching to their representations based on inductive types. Using this algorithm we show that (a sequential version of) priority rewriting is terminating for (legal) types. Finally we present a type synthesis algorithm for the extending type system and show that it has decidable type checking.

"In Chapter 7 we give a global description of our extension of the proof development system LEGO with priority rewriting. We explain how the new typing rules are verified, and we describe its facilities for defining functions by pattern matching. Finally we describe the formalization of a serious example to test our prototype."

2.3 Programming

Damm, W., Fehr, E., and Indermark, K. (1978). Higher type recursion and self-application as control structures. In *Formal description of programming concepts (Proc. IFIP Working Conf., St. Andrews, N.B., 1977)*, pages 461–489. North-Holland, Amsterdam.

[MR80e:68014] Authors' summary: "It is well known from computability theory that recursion equations and the lambda-calculus can be considered as equivalent algorithmic languages since both serve for defining the class of computable arithmetical functions. However, as control structures for programming languages, recursion and self-application are no longer equivalent. We demonstrate that self-application is in fact computationally more powerful than ordinary recursion." (For the entire collection see MR 80d:68004.)

Berry, G. and Lévy, J.-J. (1979). A survey of some syntactic results in the λ-calculus. In *Mathematical foundations of computer science, 1979 (Proc. Eighth Sympos., Olomouc, 1979)*, pages 552–566. Springer, Berlin.

[MR81f:68013] Authors' introduction: "In this paper, we consider the lambda-calculus as a formal setting for studying procedure calls in programming languages.

These languages (as in LISP) allow procedures to take procedures as arguments and also to deliver procedures as results. The lambda-calculus could be typed or untyped. In the first case, it is more realistic to add a recursion operator as in LCF. In the untyped case, the recursion is already in the language. (Take $Y = \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$.) In both cases, the lambda-calculus has constants which correspond to the basic function symbols of terms in recursive program schemes. Therefore, one can say that the lambda-calculus framework is a generalisation of recursive program schemes to any functional order. "Here we look at syntactic results. We only consider the untyped lambda-calculus, because definitions are simpler in it. Of course, we do not claim that programming languages should be untyped. But the pure lambda -calculus already has enough properties which also exist in other formalisms. As usual, the survey will be very partial. For a full treatment of the syntax and the semantics of the lambda-calculus, we strongly recommend a book by H. Barendregt (The lambda-calculus, its syntax and semantics, North-Holland, Amsterdam, to appear)." (For the entire collection see MR 81c:68004.)

Donahue, J. (1979). On the semantics of "data type". *SIAM J. Comput.*, 8(4):546–560.

[MR82f:68015] Author's summary: "We consider the general problem of specifying the meaning of programming languages that include 'data type definition facilities'. The fundamental question posed in attempting to define such languages is: 'What meaning should be given to a data type definition', or more simply, 'what does 'data type' mean?' We describe a new approach to defining the meaning of data types, treating data type as values, and give its application to the definition of a typed lambda calculus extension. We also prove a theorem stating that our lambda calculus definition is 'strongly typed'."

Aiello, L. and Prini, G. (1981). An efficient interpreter for the lambda-calculus. *J. Comput. System Sci.*, 23(3):383–424.

[MR82m:68061] Authors' summary: "A call-by-need reduction algorithm for the lambda-calculus is presented. Call-by-need is as efficient as call-by-value and is equivalent to call-by-name in languages without side effects. The algorithm, which is the core of a running system, is presented by illustrating successive transformations of a straightforward implementation of the classical definition of reduction in the lambda-calculus. All notions and algorithms are introduced in LISP code."

Backus, J. (1981). The algebra of functional programs: function level reasoning, linear equations, and extended definitions. In *Formalization of programming concepts (Proc. Internat. Colloq., Pe níscola, 1981)*, pages 1–43. Springer, Berlin.

[MR82j:68003] From the introduction: "We introduced [Comm. ACM 21 (1978), no. 8, 613 - 641; MR 58 #25059] a functional style of programming in which variable-free programs are built from a set of primitive programs by a small set of combining forms or functionals and by recursive definitions. The programs of this 'FP' style are all strict functions (of a single argument) that map a set of 'objects' into itself. This set of objects is the closure under Cartesian products of some given set of atoms. The earlier paper suggested that FP systems have the following properties: (a) extremely simple semantics, (b) existence of closed-form, nonrecursive expressions for many functions that are normally recursively defined, (c) programs exhibit a clear hierarchical structure in which high level programs can be combined to form higher level programs, (d) the principal FP combining forms are the operations of a powerful algebra of programs that can be used to transform programs and to solve equations for recursively defined programs. "The present paper is concerned with three topics about FP systems, the first of which relates to (c) above and the other two relate to (d). After a brief review of the basics of FP systems and the algebra of programs, the first main section discusses the relationship between the FP style and the 'lambda style' of programming and of reasoning about programs, where the lambda style is that used in lambda-calculus-based languages such as LISP and ISWIM [P. J. Landin, Comput. J. 6 (1964), no. 4, 308 - 320; CR 1964:6677]. "The second main section of the paper generalizes and greatly simplifies our earlier work (op. cit.) on solving equations for recursively defined functions. It introduces the notion of 'forms', the program schemas of FP, and of 'linear' forms. "The third main section proposes a new form for defining FP functions that corrects a major problem in the proper definitions (op. cit.): the difficulty of reading variable-free function definitions." (For the entire collection see MR 82f:68005.)

Stoy, J. E. (1981). *Denotational semantics: the Scott-Strachey approach to programming language theory.* MIT Press, Cambridge, Mass. Reprint of the 1977 original, With a foreword by Dana S. Scott.

[**MR82m:68003**] The original hardbound edition has been reviewed [1977; MR 58 #8460].

Williams, J. H. (1981). Formal representations for recursively defined functional programs. In *Formalization of programming concepts (Proc. Internat. Colloq., Peníscola, 1981)*, pages 460–470. Springer, Berlin.

[**MR82j:68009**] From the introduction: "J. W. Backus [Comm. ACM 21 (1978), no. 8, 613 - 641; MR 58 # 25059] showed how a formal system for functional programming (FFP) can be used to represent all of the objects, primitive functions, and combining forms contained in his informal system of functional programming (FP). In this note we give a general algorithm for producing FFP representatives for arbitrary, recursively defined FP functions. In doing so, we demonstrate the universality of FFP, since all partial recursive functions can be defined in FP." For the entire collection see MR 82f:68005.

Berkling, K. J. and Fehr, E. (1982a). A consistent extension of the lambda-calculus as a base for functional programming languages. *Inform. and Control*, 55(1-3):89–101.

[**MR86g:03027**] Author summary: "Church's lambda-calculus is modified by introducing a new mechanism, the lambda-bar operator, which neutralizes the effect of one preceding lambda binding. This operator can be used in such a way that renaming of bound variables in any reduction sequence can be avoided, with the effect that efficient interpreters with comparatively simple machine organization can be designed. It is shown that any semantic model of the pure λ-calculus also serves as a model of this modified reduction calculus, which guarantees smooth semantic theories. The Berkling reduction language (BRL) is a new functional programming language based upon this modification."

Berkling, K. J. and Fehr, E. (1982b). A modification of the λ-calculus as a base for functional programming languages. In *Automata, languages and programming (Aarhus, 1982)*, pages 35–47. Springer, Berlin.
[MR83m:68013] Authors' summary: "Church's lambda-calculus is modified by introducing a new mechanism, the lambda-bar operator '#', which neutralizes the effect of one preceding lambda-binding. This operator can be used in such a way that renaming of bound variables in any reduction sequence can be avoided, with the effect that efficient interpreters with comparatively simple machine organization can be designed. Any semantic model of the pure lambda-calculus also serves as a model for this modified reduction calculus, which guarantees smooth semantical theories. The Berkling reduction language (BRL) is a new functional programming language based upon this modification." (For the entire collection see MR 83k:68002.)

Berry, G. and Curien, P. L. (1982). Sequential algorithms on concrete data structures. *Theoret. Comput. Sci.*, 20(3):265–321.
[MR83h:68040] Authors' summary: "We provide a sequential denotational semantics for sequential programming languages, based on a new notion of sequential algorithm on the Kahn - Plotkin concrete data structures. Intuitively an algorithm may be seen either as a concrete object – a 'program' in a simple output-driven language – or as an abstract object – the pair of a sequential function and of a computation strategy for it. The concrete and abstract presentations are equivalent, as shown by a representation theorem. The algorithms form a Cartesian closed category with straightforward solutions to recursive domain equations. Hence they may replace functions in the denotational semantics of any sequential language. An applicative programming language based on sequential algorithms is presented in a companion paper."

Burton, F. W. (1982). A linear space translation of functional programs to Turner combinators. *Inform. Process. Lett.*, 14(5):201–204.
[MR84b:03027] H. B. Curry's original algorithm to translate a lambda-expression into a term of combinatory logic can lead to an exponential growth in the length of the expression. Many authors have presented more efficient translation algorithms. In particular, D. A. Turner [J. Symbolic Logic 44 (1979), no. 2, 267 - 270; MR 80f:03021] introduced an improvement of Curry's algorithm which leads to a growth of at most O(n_2) in the length of the expression. In this paper the author presents an improvement of Turner's algorithm which guarantees only a linear growth in size when it is applied to those lambda-expressions (called autonomous functions) in which no subexpression A can be applied to a variable free in A. Any lambda-

expression can be translated into an autonomous function (this, in the worst case, can lead to a growth of $O(n_2)$ in length). Reviewed by M. Coppo.

Scott, D. S. (1982). Lectures on a mathematical theory of computation. In *Theoretical foundations of programming methodology (Munich, 1981)*, pages 145–292. Reidel, Dordrecht.

[MR85g:68043] From the introduction: "These notes were originally written for lectures on the semantics of programming languages delivered at Oxford during the Michaelmas term of 1980. The purpose of the course was to provide the foundations needed for the method of denotational semantics; in particular, we wanted to make the connections with recursive function theory more definite and to show how to obtain explicit, effectively given solutions to domain equations. Roughly, these chapters cover the first half of the book by J. E. Stoy [Denotational semantics: the Scott-Strachey approach to programming language theory, MIT Press, Cambridge, Mass., 1977; MR 58 # 8460], and he was able to continue the lectures the next term discussing semantical concepts following his text."

Lecture I: Domains given by neighbourhoods; Lecture II: Mapping between domains; Lecture III: Domain constructs; Lecture IV: Fixed points and recursion; Lecture V: Typed λ-calculus; Lecture VI: Introduction to domain equations; Lecture VII: Computability in effectively given domains; Lecture VIII: Retracts of the universal domain. For the entire collection see MR 84c:68004.

Coppo, M. (1983). On the semantics of polymorphism. *Acta Inform.*, 20(2):159–170.

[MR85k:03008] The author adds to λ-calculus two extra term-forming operations: (if e_1 then e_2 else e_3) and (fix $x.e_1$), where e_i are arbitrary terms and x is a variable (fix $x.e_1$ corresponds to Curry's $Y(\lambda x.e_1)$). The resulting terms are a core fragment of the language ML of M. Gordon , R. Milner and C. Wadsworth [Edinburgh LCF, Springer, Berlin, 1979]. The author states natural-deduction rules for assigning types to these terms. He also describes Milner's very natural semantics for terms, types, and type-assignment statements, in a c.p.o. D which contains its own space of continuous functions. He gives examples to show that the type-assignment rules are not complete with respect to this semantics. (And no complete rules are possible; he has mentioned in private correspondence that the semantic derivability relation is not r.e.).

But if the set of assumptions satisfies a reasonable compatibility restriction, the relation becomes r.e. and the author proves his rules complete. For type-schemes (called polymorphic types), a similarly restricted completeness theorem is proved, together with a principal-type-scheme theorem.

{Reviewer's remark: The author's interpretation of terms in this paper is not a λ-model, nor even a λ-algebra. But its interior (the interpretations of closed terms) is a λ-model. If the interpretation of application were changed to

$$v_1^* v_2 = v_1 E[D \to D] \to v_1(v_2), ?$$

(corresponding to a call-by-name approach instead of a call-by-value), we would get a λ-model and the other theorems in the paper would still hold.} Reviewed by R. Hindley.

Coppo, M., Dezani, M., and Longo, G. (1983). Applicative information systems. In *CAAP '83 (L'Aquila, 1983)*, pages 35–64. Springer, Berlin.

[MR86k:68062] This paper deals with information systems in the sense of D. S.

Scott [Automata, languages and programming (Aarhus, 1982), 527–613, Springer, Berlin, 1982; MR 83m:68029]. These structures were introduced in order to provide a tool for studying the denotational semantics of programming languages. The authors extend the concept to capture self-application directly. Thus, they define applicative information systems as those in which one can get naturally elements of the form $x.y$, where x and y are elements of the domain $|A|$ determined by the information system A. In this context, the set of representable functions from $|A|$ to $|A|$ is built by functions $f\colon |A| \to |A|$ such that $f(y) = x.y$ for all y in $|A|$ and some x in $|A|$. The main result here is a characterization of the representable functions.

Using these results, the authors solve some fixpoint equations that arise in denotational semantics; special attention is devoted to λ-calculus models. In the last section they study computability on these abstract structures from the point of view of computer science. For the entire collection see MR 85d:68006. Reviewed by A. Bahamonde. For errata and/or addenda to the original MR item see MR 1986 Errata and Addenda in the paper version.

Coppo, M. and Giovannetti, E. (1983). Completeness results for a polymorphic type system. In *CAAP '83 (L'Aquila, 1983)*, pages 179–190. Springer, Berlin.

[MR86a:68064] Authors' summary: "An interesting notion of polymorphism is the one introduced in the language ML [M. J. Gordon, A. J. Miller and C. P. Wadsworth, Edinburgh LCF: a mechanized logic of computation, Springer, Berlin, 1979]. Its soundness has been proved by R. Milner [J. Comput. System Sci. 17 (1978), no. 3, 348–375; MR 81g:68022] for a subset of ML based on the λ-calculus plus constants. A partial completeness result for the same language has been given by Coppo [Acta Inform. 20 (1983), no. 2, 159–170; MR 85k:03008]. The aim of the present paper is to extend the above results to a language including also the Cartesian product and the disjoint sum. The extension is not trivial, owing to difficulties introduced mainly by the disjoint sum. Moreover, a semantic characterization of typed terms is given."

Wand, M. (1983). Loops in combinator-based compilers. *Inform. and Control*, 57(2-3):148–164.

[MR85j:68072] This is one of a series of papers by the author exploring the creation of compilers from denotational semantic definitions of programming languages via the use of combinators. In previous papers he has shown how to convert a denotational definition expressed in its usual form (i.e., in λ-expressions) to one using combinators, suggested ways of choosing appropriate combinators, and shown how to rotate the resulting combinator trees to a right-linear form which corresponds in an obvious and aesthetic way to traditional assembly language code. Loops, however, were handled in an unsatisfactory way: the start-of-loop address was pushed onto the runtime stack and retrieved from it at the end of the loop, i.e. loops were handled as if they were subroutines.

In this paper the author improves on his earlier approach by showing how to turn combinator trees representing loops into combinator graphs which explicitly represent the loops and hence can cause a "jump" to be generated for the loop closure, rather than a "call" and "return". He does so by introducing an extension of the Y-combinator called "label" which binds variables. He then shows that occurrences of label can be eliminated in favor of cycles in the combinator graph which, in turn, correspond naturally to "jump" instructions. Reviewed by Steven S. Muchnick.

Longo, G. and Moggi, E. (1984). Gödel numberings, principal morphisms, combinatory algebras. In *Mathematical foundations of computer science, 1984 (Prague, 1984)*, pages 397–406. Springer, Berlin.
[MR87d:03034] Summary: "Functional languages are based on the notion of application: programs may be applied to data or programs. By application one may define algebraic functions, and a programming language is functionally complete when any algebraic function $f(x_1, \cdots, x_n)$ is representable (i.e., there is a constant a such that $f(x_1, \cdots, x_n) = ax_1 \ldots x_n$). Combinatory logic (CL) is the simplest type-free language which is functionally complete.

"In a sound category-theoretic framework the constant a above may be considered an 'abstract Gödel number' for f, as Gödel numberings are generalized to 'principal morphisms'. In this way, models of CL are categorically characterized and their relation to λ-calculus models within Cartesian closed categories is given. Finally, the partial recursive functionals in any finite higher type are shown to yield models of CL." For the entire collection see MR 86b:68004.

Révész, G. (1984). An extension of lambda-calculus for functional programming. *J. Logic Programming*, 1(3):241–251.
[MR86g:68111] Author summary: "An implementation-oriented modification of lambda-calculus is presented together with some additional conversion rules for list manipulations. The resulting set of axioms is very simple and provides for a theoretical foundation of the semantics of functional programming. At the same time it can be used directly for the implementation of a powerful graph-reduction algorithm."

Stark, W. R. (1984). A glimpse into the paradise of combinatory algebra. *Internat. J. Comput. Inform. Sci.*, 13(3):219–236.
[MR86d:68048] Author summary: "This expository paper examines combinatory algebra and the lambda calculus as foundations for functional languages (LISP, FP, KRC, etc.). The basic notions are defined and numerous examples are given. Historical comments and references are provided."

Böhm, C. and Berarducci, A. (1985). Automatic synthesis of typed λ-programs on term algebras. *Theoret. Comput. Sci.*, 39(2-3):135–154.
[MR87h:68012] The authors introduce the notion of iteratively defined functions on heterogeneous term algebras as a syntactic restriction of primitive recursive functions, without any loss of expressive power. They show that there exists a formal definition of a compiler from iteratively defined functions into the class of second-order typed lambda-calculus. They establish a completeness theorem for lambda-terms with type of degree at most two. They also study a new congruence relation for such lambda-terms, which is stronger than lambda-convertibility. Reviewed by E. Fehr.

de Bakker, J. W. and Kok, J. N. (1985). Towards a uniform topological treatment of streams and functions on streams. In *Automata, languages and*

programming (Nafplion, 1985), pages 140–148. Springer, Berlin.
[**MR87d:68070**] Summary: "We study the semantics of functional languages on streams such as Turner's SASL or KRC. The basis of these languages is recursive equations for (functions on) finite or infinite sequences. The paper presents a start towards a mathematical (denotational) description of such languages using tools from metric topology. The description is based on the Banach fixed point theorem and a restricted version of a typed lambda calculus. With a system of recursive stream (function) declarations we associate a system of functions in an appropriate topological domain. These functions have to be contracting in certain arguments and non-distance-increasing in others; a syntax is designed which ensures the right interplay between these conditions. Nondeterminism is handled by considering compact sets of streams, and preservation of compactness is another important technical issue. Not all concepts in a language such as KRC are covered, and some indications on possible extensions of the framework are provided." For the entire collection see MR 87a:68002.

Goerdt, A. (1985). A Hoare calculus for functions defined by recursion on higher types. In *Logics of programs (Brooklyn, N.Y., 1985)*, pages 106–117. Springer, Berlin.
[**MR808792**] This item will not be reviewed individually. For details of the collection in which this item appears see 86f:68007.

Kantor, R. and Sontacchi, G. (1985). Un interprète LISP de la programmation fonctionnelle réalisé par des combinateurs. *RAIRO Inform. Théor.*, 19(1):33–41.
[**MR795770**] There will be no review of this item.

Meyer, A. R. and Wand, M. (1985). Continuation semantics in typed lambda-calculi. In *Logics of programs (Brooklyn, N.Y., 1985)*, pages 219–224. Springer, Berlin.
[**MR808800**] This item will not be reviewed individually. For details of the collection in which this item appears see 86f:68007.

Mitchell, J. C. and Meyer, A. R. (1985). Second-order logical relations. In *Logics of programs (Brooklyn, N.Y., 1985)*, pages 225–236. Springer, Berlin.
[**MR808801**] This item will not be reviewed individually. For details of the collection in which this item appears see 86f:68007.

Abramsky, S. (1986). Strictness analysis and polymorphic invariance (extended abstract). In *Programs as data objects (Copenhagen, 1985)*, pages 1–23. Springer, Berlin.
[**MR846658**] This item will not be reviewed individually. For details of the collection in which this item appears see 87b:68002.

Allison, L. (1986). *A practical introduction to denotational semantics*. Cambridge University Press, Cambridge.

[**MR88f:68091**] This is a well-written and readable textbook on denotational semantics (for further references see papers by D. S. Scott [Toposes, algebraic geometry and logic (Halifax, N.S., 1971), 97–136, Lecture Notes in Math., 274, Springer, Berlin, 1972; MR 53 #7879], R. D. Tennent [Comm. ACM 19 (1976), no. 8, 437–453; MR 55 #1791], and J. E. Stoy [Denotational semantics: the Scott-Strachey approach to programming languages theory, MIT Press, Cambridge, Mass., 1977; MR 58 #8460]).

After some motivating examples, the author introduces the λ-notation (i.e., typed or polymorphic λ-calculus) as the meta-language of denotational semantics. The mathematical foundations of the subject are "covered in sufficient detail to illustrate the fundamental problems in semantics theory that were solved by Scott" (from the Preface).

Although the semantic definition of programming languages can be applied to some programs to show what they denote, this is not the aim of denotational semantics. The emphasis is placed on practical work. In this sense, the semantic definitions of programming languages are used to build interpreters translating them into languages like Pascal or Algol-68.

A wide range of programming language features are defined in a denotational semantics style, including side-effects, storage allocation, procedure, functions, data structures and type checking. Of special interest is the ninth chapter where the author gives a denotational semantics of a subset of Prolog (not including the cut) which is later translated into Algol-68 to form an interpreter. Reviewed by A. Bahamonde.

Cousineau, G., Curien, P.-L., Mauny, M., and Suárez, A. (1986a). Combinateurs catégoriques et implémentation des langages fonctionnels. In *Combinators and functional programming languages (Val d'Ajol, 1985)*, pages 85–103. Springer, Berlin.

[**MR88k:03027**] Summary (translated from the French): "We present a technique for implementing functional programming languages that is based on the use of categorical combinators. Unlike other implementation techniques using combinators, ours does not lead to a reduction machine but rather to a von Neumann-type machine whose terms formed by categorical combinators are, as it were, the code. Our machine, called the categorical abstract machine (CAM), implements call-by-value in a natural way but can also do lazy evaluations very efficiently. This version of our presentation of the CAM [see Cousineau, Curien and Mauny, Sci. Comput. Programming 8 (1987), no. 2, 173–202] includes some recent optimizations [Mauny and Suarez, "Implementing functional languages in the categorical abstract machine", in Proceedings of the ACM conference on Lisp and functional programming, 1986, ACM, New York; to appear]." For the entire collection see MR 87j:03004.

Cousineau, G., Curien, P.-L., and Robinet, B., editors (1986b). *Combinators and functional programming languages*, Berlin. Springer-Verlag.

[**MR87j:03004**] Contents: P. Bellot and B. Robinet, Streams are not dreams (pp. 1–20); Luca Cardelli, Amber (pp. 21–47); Luca Cardelli, The Amber machine (pp. 48–70); Thierry Coquand, Sur l'analogie entre les propositions et les types [On the analogy between propositions and types] (pp. 71–84); Guy Cousineau, Pierre-Louis

Curien, Michel Mauny and Ascander Suarez, Combinateurs categoriques et implementation des langages fonctionnels [Categorical combinators and implementation of functional languages] (pp. 85–103); J. Roger Hindley [Roger Hindley], Combinators and lambda-calculus. A short outline (pp. 104–122); Gerard Huet, Cartesian closed categories and lambda-calculus (pp. 123–135); J. Lambek, Cartesian closed categories and typed λ-calculi (pp. 136–175); Simon L. Peyton Jones, An introduction to fully-lazy supercombinators (pp. 176–208). The papers of mathematical interest are being reviewed individually.

Curien, P.-L. (1986). *Categorical combinators, sequential algorithms and functional programming.* Pitman Publishing Ltd., London.
[MR87k:68002] This monograph gives a concrete approach to the semantics of sequential programming languages. After a presentation of categorical combinators and lambda-calculus, the notion of sequential algorithms as developed by Berry is introduced. These concrete mathematical objects are of a syntactic nature and lead naturally to the design of a programming language called CDSO. The motivation behind these constructions is to ensure a fully abstract semantics which means that two program pieces have the same meaning as soon as they behave the same way on a computer.

This book is intended for theoretical computer scientists with a profound knowledge in category theory, universal algebra and mathematical logic. It is a comprehensive discussion of sequential algorithms, their theory and their implementation, which previously was scattered over a number of independent publications. Reviewed by E. Fehr.

Hindley, J. R. (1986). Combinators and lambda-calculus. A short outline. In *Combinators and functional programming languages (Val d'Ajol, 1985)*, pages 104–122. Springer, Berlin.
[MR88c:03016] Over the last several years the interest of computer scientists in λ-calculus and combinatory logic has steadily grown. The present paper is an introduction to these subjects oriented to their applications in programming theory. Both typed and untyped languages are presented and their most important features are discussed. The clarity of exposition and the richness of examples, together with a constant rigour, make this paper outstandingly useful. For the entire collection see MR 87j:03004. Reviewed by M. Dezani-Ciancaglini.

Ida, T. and Konagaya, A. (1986). Comparison of closure reduction and combinatory reduction schemes. In *RIMS symposia on software science and engineering, II (Kyoto, 1983/1984)*, pages 261–291. Springer, Berlin.
[MR87j:68021]

Summary: "We analyze the efficiencies of closure reduction and combinatory reduction schemes by introducing a labelled tree representing a λ-term. Translation of a λ-term into combinatory terms, i.e., bracket abstraction, can be viewed as attaching labels S, B, C, K, I to each node. Similarly, a node of a tree representing a λ-term can be labelled depending upon the presence of free variables in the subtrees. Resulting labelled trees which represent a λ-term and the translated combinatory term are similar, i.e., their underlying trees are the same. We can then make performance comparisons in terms of the cost involved in traversing the labelled trees by machine models reflecting the essential behavior of Turner's combinatory reducer and a closure reducer. Our work is an elaboration of Turner's and Peyton Jones's experiments of combinatory reductions. However, our approach is not to resort to actual runs of programs, but is more theoretical, based on abstract machine models working on labelled trees.

"Our conclusion from the performance comparisons is that a closure reducer is in most cases more efficient than combinatory reducers in terms of storage consumption, which is a dominant factor in determining the overall performance of the reducers. Furthermore, we show that the two reducers, which seem quite different at first sight, are in fact very similar and with small modifications the two schemes become essentially the same." For the entire collection see MR 87f:68003.

Jiang, M. D. (1986). A discussion of the combinatory logical nature of the functions ρ and μ in FFP. *Kexue Tongbao (Chinese)*, 31(21):1626–1629.
[MR885842] There will be no review of this item.

Nerode, A. (1986). Applied logic. In *The merging of disciplines: new directions in pure, applied, and computational mathematics (Laramie, Wyo., 1985)*, pages 127–163. Springer, New York.
[MR88b:03042] This paper, which has a survey character, is devoted to the application of mathematical logic in computer science and is not addressed to the specialist in either logic or computer science. The main topics of this interesting paper are first-order logic, automated reasoning, PROLOG, knowledge bases, program correctness, modal and dynamic logic, lambda calculus, universal Turing machines, and Post production systems. The author gives historical dates in the development of mathematical logic and its applications. A long and useful list of references to indicate the scope of the applications of logic in computer science is also included. For the entire collection see MR 87i:00020. Reviewed by Hans-Joachim Goltz.

Peyton Jones, S. L. (1986). An introduction to fully-lazy supercombinators. In *Combinators and functional programming languages (Val d'Ajol, 1985)*, pages 176–208. Springer, Berlin.
[MR881861] This item will not be reviewed individually. For details of the collection in which this item appears see 87j:03004.

Shultis, J. (1986). What is a model? A consumer's perspective on semantic theory. In *Mathematical foundations of programming semantics (Manhattan, Kan., 1985)*, pages 296–307. Springer, Berlin.
[MR874389] This item will not be reviewed individually. For details of the collection in which this item appears see 87j:68005.

Bellot, P. and Jay, V. (1987). A theory for natural modelisation and implementation of functions with variable arity. In *Functional programming languages and computer architecture (Portland, Ore., 1987)*, pages 212–233. Springer, Berlin.

[**MR914980**] This item will not be reviewed individually. For details of the collection in which this item appears see 88f:68005.

Coppo, M. (1987). Types and polymorphism in programming languages. In *Proceedings of the conferences on mathematical logic, Vol. 3 (Italian) (Siena, 1985; Padova, 1985; and Siena, 1986)*, pages 233–281, Siena. Univ. Siena.

[**MR88m:68009**] This survey aims at highlighting the relationships between mathematical logic and type structures in programming languages. It covers functional languages, presenting the pure λ-calculus, its typed version ($\lambda\eta t$) and the second-order typed calculus ($\lambda t2$). The expressive power, termination properties and semantic models of each system are reviewed. Proofs are omitted, but a sketch of Girard's proof of the strong normalization theorem for $\lambda t2$ is reported. The "formulae-as-types" analogy between typed systems and intuitionistic calculi is also discussed and illustrated by examples. Polymorphism, i.e., the ability to apply a function to values of different types consistently, which is attained in the second-order system and is appealing from a computer science perspective, can be also attained by type assignment rules, as in the programming language ML; the paper reports the rules and discusses their adequacy and completeness. Several extensions are also discussed, e.g., the introduction of a fixed-point operator and of recursive types. For the entire collection see MR 88e:03002. Reviewed by Carlo Montangero.

Cousineau, G., Curien, P.-L., and Mauny, M. (1987). The categorical abstract machine. *Sci. Comput. Programming*, 8(2):173–202.

[**MR89a:68147**] The lambda-calculus is one of the classical tools in programming language semantics. It can itself be considered as a programming language, and it serves as a metalanguage in programming language descriptions. Recently, several researchers have established strong connexions between the lambda-calculus and the theory of Cartesian closed categories. The authors show that this last theory is not only suited to semantical developments, but can be seen as defining a very simple and efficient abstract machine that can be easily implemented on actual physical processors. They call it the Categorical Abstract Machine (CAM). The authors present their machine in detail, compare it with Landin's classical SECD machine, prove the correctness of the machine with respect to beta-reductions in the lambda-calculus, and show how to compile the ML functional language. The CAM actually serves as the execution machine of the CAML system developed at INRIA (France). Reviewed by G. Berry.

Felleisen, M. and Friedman, D. P. (1987). A reduction semantics for imperative higher-order languages. In *PARLE. Parallel architectures and languages Europe, Vol. II (Eindhoven, 1987)*, pages 206–223. Springer, Berlin.

[**MR910308**] This item will not be reviewed individually. For details of the collection in which this item appears see 88e:68002.

Felleisen, M., Friedman, D. P., Kohlbecker, E., and Duba, B. (1987). A syntactic theory of sequential control. *Theoret. Comput. Sci.*, 52(3):205–237.

[MR89j:68097] The λ-calculus considered as a programming language has no control structures, which can lead to inefficient function (program) evaluation. Some λ-calculus-based languages seek to remedy this problem by providing nonfunctional control operators that give programs access to current continuations. However, proofs of program properties in such extended languages can no longer be carried out in syntactic domains, but must rely on semantic interpretations. The authors of the present paper, in an extended revision of two earlier reports [Felleisen and Friedman, in Formal description of programming concepts, III, 193–217, North-Holland, Amsterdam, 1986; per bibl.; the authors, in Symposium on Logic in Computer Science (Cambridge, MA, 1986), 131–141, IEEE, Washington, DC, 1986], propose an extended calculus that characterizes nonfunctional control in a completely syntactic way.

The authors introduce a variant of Scheme's control operator call/cc into the λ-calculus term set. The semantics of the extended language is defined in terms of an abstract machine which in turn derives a reduction system, the λ_c-calculus, which is an algebraic extension of the λ-value-calculus of G. D. Plotkin [Theoret. Comput. Sci. 1 (1975), no. 2, 125–159; MR 55 #2514]. They show that λ_c is correct, in the sense of Plotkin, with respect to the machine semantics, which in turn is equivalent to the operational rewriting semantics. Intermediate results are based in large part on ideas and methods of Plotkin.

The authors argue that their calculus together with the machine semantics provides a convenient tool for reasoning about programs with nonfunctional control operators. Reviewed by Dorothy Bollman.

Fokkinga, M. M. (1987). Programming language concepts—the lambda calculus approach. In *Essays on concepts, formalisms, and tools*, pages 129–162. Math. Centrum Centrum Wisk. Inform., Amsterdam.

[MR88i:68061] Summary: "The lambda calculus is a formal system, originally intended as a tool in the foundation of mathematics, but mainly used to study the concepts of algorithm and effective computability. Recently, the lambda calculus and related systems received attention in computer science for another reason: several important programming language concepts can be explained elegantly and can be studied successfully in the framework of the lambda calculi. We show this mainly by means of examples. We address ourselves to interested computer scientists who have no prior knowledge of the lambda calculus. The concepts discussed include: parameterization, definitions, recursion, elementary and composite data types, typing, abstract types, control of visibility and life-time, and modules." For the entire collection see MR 88g:68005.

Gibert, J. (1987). Functional programming with combinators. *J. Symbolic Comput.*, 4(3):269–293.

[MR89a:68019] Summary: "We show that combinators provide a very suitable basis for implementations of functional and symbolic computation in computer architecture. A powerful combinator reduction system is developed which meets programmer and machine requirements for (i) efficiency of representation and execution of symbolic algorithms, and (ii) availability of algebraic manipulation needed to analyse symbolic computations. An algebraic model is constructed to provide rigorous semantics for the system. The reduction language of the system aims at exposing efficient flows of data and fine-grain parallelism, and a computer architecture, which is proposed to run the system, utilizes both sequential and parallel processing modes in order to achieve maximum efficiency of symbolic computation. Finally, an implementation of the interpreter and functional simulator for the architecture is described."

Lins, R. D. (1987). Categorical multicombinators. In *Functional programming languages and computer architecture (Portland, Ore., 1987)*, pages 60–79. Springer, Berlin.

[MR914979] This item will not be reviewed individually. For details of the collection in which this item appears see 88f:68005.

Poigné, A. and Voss, J. (1987). On the implementation of abstract data types by programming language constructs. *J. Comput. System Sci.*, 34(2-3):340–376.

[MR89g:68048] Summary: "Implementations of abstract data types are defined via enrichments of a target type. We propose to use an extended typed λ-calculus for enrichments in order to meet the conceptual requirement that an implementation has to bring us closer to a (functional) program. Composability of implementations is investigated, the main result being that composition of correct implementations is correct if terminating programs are implemented by terminating programs. Moreover, we provide syntactical criteria to guarantee correctness of composition. The proof is based on strong normalization and Church-Rosser results for the extended λ-calculus which seem to be of interest in their own right."

Trakhtenbrot, B. A. (1987). On "logical relations" in program semantics. In *Mathematical logic and its applications (Druzhba, 1986)*, pages 213–229. Plenum, New York.

[MR89e:68083] Logical relations, as a conceptual and technical tool, entail interesting applications in the theory of programming, properties of typed programming languages which intuitively amount to representation independence. The paper illustrates the impact of logical relations, revising some previous results of both the author and others.

Section 2 considers functional frames, models and language construction, using as tools arbitrary infinite Böhm trees (as the natural extension of finite λ-terms augmented with fixed point operators). Section 3 presents a survey on logical relations and their fundamental properties. Section 4 deals with the characterization of schema-topological functionals via logical relations. The formalization is in terms of rank 2 functionals that are invariant with respect to specific logical relations (partial isomorphisms), for complete partially ordered models. In Section 5, the observational equivalence of two models with respect to a given programming language exhibits a typical situation when the usual logical relations do not provide the expected characterization, and more powerful (Statman's class) S-relations are needed. Section 6 is devoted to the problem of invariance with respect to locations for Algol-like language semantics: some memory locations are not relevant for the meaning of a given programming construction. The formal definitions use specific logical relations in reasoning about this type of program. Each section deals with specific open questions. This is a paper in which the technical aspects of program semantics are wonderfully governed by fruitful ideas, and an open-ended discussion containing hints for further valuable research directions is included. For the entire collection see MR 89b:03004. Reviewed by Neculai Curteanu.

Bellot, P. and Sarni, D. (1988). Proposal for a natural formalization of functional programming concepts. *RAIRO Inform. Théor. Appl.*, 22(3):341–360.
[**MR91a:68175**] Summary: "Graal is the name of a variable-free functional programming (FP) language based on principles arising from FP systems and combinatory logic. This article proposes a formalization of the concepts of Graal using a formal theory T_G where the notions of uncurryfied combinator and polyadic application are included. This allows us to give a clear semantics of the Graal language, because T_G becomes its elementary model. T_G appears as a new theoretical basis for the study of applicative programming languages. T_G has been conceived as a theory of intensional functions; that is to say, T_G is a new formalization of computability well suited for computer science."

Borzyszkowski, A., Kubiak, R., and Sokołowski, S. (1988). A set-theoretic model for a typed polymorphic lambda calculus. A contribution to MetaSoft. In *VDM '88. VDM—the way ahead (Dublin, 1988)*, pages 267–298. Springer, Berlin.
[**MR995062**] This item will not be reviewed individually. For details of the collection in which this item appears see 89m:68093.

Bruce, K. B. and Riecke, J. G. (1988). The semantics of Miranda's algebraic types. In *Mathematical foundations of programming language semantics (New Orleans, LA, 1987)*, pages 455–473. Springer, Berlin.
[**MR89h:68098**] Summary: "Miranda has two interesting features in its typing system: implicit polymorphism (also known as ML-style polymorphism) and algebraic types. Algebraic types create new types from old and can operate on arbitrary types. This paper argues that functions on types, or type constructors, best represent the meaning of algebraic types. Building upon this idea, we develop a denotational semantics for algebraic types. We first define a typed lambda calculus that specifies constructors. A semantic model of type constructors is then built, using the ideal model as a basis. (The ideal model gives the most natural semantics for Miranda's implicit polymorphism.) The model is shown to be sound with respect to this lambda calculus. Finally, we demonstrate how to use the model to interpret algebraic types, and prove that the translation produces elements in the model." For the entire collection see MR 89a:68005.

Constable, R. L. (1988). Themes in the development of programming logics circa 1963–1987. In *Annual review of computer science, Vol. 3*, pages 147–165. Annual Reviews, Palo Alto, CA.

[MR91g:03064] This paper begins with some general historical comments on the development of program logics and then turns to a simple, worked example. A logic of typed functions, essentially Church's typed lambda calculus with fixed-point operators added, is introduced and used to give semantics to imperative programs with assignment, conditional and iterative loops. Floyd-Hoare assertions are introduced for these programs, and shown to be sound with respect to the logic-based semantics. The paper concludes with more general comments about alternative approaches, such as Dijkstra's predicate transformers, and a summary of the author's own Nuprl typed program logic. The paper is written in an accessible, informal style. For the entire collection see MR 90a:68001. Reviewed by John C. Mitchell.

Engeler, E. (1988). Cumulative logic programs and modelling. In *Logic colloquium '86 (Hull, 1986)*, pages 83–93. North-Holland, Amsterdam.

[MR89g:03016] Summary: "The formation process of pure logic programs over a first-order language is iterated to give rise to cumulative logic programs. Such programs turn out to be objects in a combinatory model and are therefore amenable to algebraic manipulation, including equation solving. It is pointed out how this fact can be employed in a discipline of modelling for cooperative processes. Other results concern the representability of equational classes, universal classes and algorithmic classes as solution sets of a set of combinatory equations." For the entire collection see MR 88g:03009.

Jiang, M. D. (1988). On combinatory-logic essence of functions ρ and μ in ffp. *Kexue Tongbao (English Ed.)*, 33(3):246–250.

[MR89m:68087] J. Backus, in a famous paper [Comm. ACM 21 (1978), no. 8, 613–641; MR 58 #25059], introduced a class of applicative programming systems (FP systems), and a formal language, FFP, for dealing with this style of programming. Moreover, he gave the semantics of FFP, by a translation into FP. This translation is achieved by means of two functions, ρ and , the environment and the semantic function, respectively.

In two earlier papers, the author has investigated the combinatory logic version of FP. Here the author completes the combinatory logic analogue of Backus' work, by giving the combinators corresponding to ρ and . In particular, the triple $\langle \rho, , \text{apply} \rangle$, where apply is the application function, turns out to be the solution of a least fixpoint equation in combinatory logic. As a consequence, this gives an interpreter of FFP based on combinatory logic. The paper is interesting, but quite difficult to read since it is not self-contained. The author refers to definitions given in earlier papers, written in Chinese; it is, however, possible to figure them out with a little effort. Reviewed by Simona Ronchi Della Rocca.

Lafont, Y. (1988a). Corrigenda: "The linear abstract machine". *Theoret. Comput. Sci.*, 62(3):327–328.

[MR90j:68027b] See (Lafont, 1988b).

Lafont, Y. (1988b). The linear abstract machine. *Theoret. Comput. Sci.*, 59(1-2):157–180. International Joint Conference on Theory and Practice of Software Development (Pisa, 1987).

[MR90j:68027a] In this paper (which is an extended version of his joint work with J.-Y. Girard [in TAPSOFT '87, Vol. 2 (Pisa, 1987), 52–66, Lecture Notes in Comput. Sci., 250, Springer, Berlin, 1987; MR 88h:68010]), the author discusses the intuitionistic linear logic of Girard [Theoret. Comput. Sci. 50 (1987), no. 1; MR 89m:03057], the corresponding categorical linear combinatory logic, linear lambda calculus, a natural execution process, which is proved to terminate, and its sequential implementation—a linear abstract machine (a cousin of the categorical abstract machine [G. Cousineau, P. L. Curien and M. Mauny, in Functional programming languages and computer architecture (Nancy, 1985), 50–64, Lecture Notes in Comput. Sci., 201, Springer, Berlin, 1985; Zbl 592:68045]). The main idea of linear logic consists in omitting contraction and weakening rules in Gentzen's sequent calculus and hence in splitting (rules for) the connectives $\wedge$ and $\top$, respectively, into $\otimes$, & and 1,t so that, e.g., in this logic, $A \,\&\, B \to A$ is provable, but not $A \otimes B \to A$ and not $A \to A \otimes A$. In this logic the antecedent Γ of a sequent $\Gamma \to A$ is treated as tensor product $\bigotimes \Gamma$ (not as the ordinary conjunction $\&\Gamma$). The corresponding categories are symmetric monoidal (with tensor product $\otimes$) and closed (with exponentiation or implication $\multimap$), with finite products (or conjunctions &). Linear (typed) lambda calculus is presented as the ordinary lambda language (no axioms and rules for this lambda calculus are given in the paper) with the pairing construct corresponding to the tensor product $\otimes$ (instead of the direct product) and with lambda prefixes over tuples of variables (built via pairing in any order) with the exception that the following are not allowed in the language: (a) projection constructs corresponding to pairing, (b) sharing free variables in lambda terms under application or pairing, and (c) fictitious variables in lambda prefixes.

These calculi are extended by a unary connective (type constructor) $!A$, "of course A", which is recursively defined as $!A = A \,\&\, (1 \,\&\, (!A \otimes !A))$ (so that, e.g., $!A \to !A \otimes !A$ is provable) and which allows conservative embedding of the ordinary intuitionistic logic into these calculi. In categorical terms $!A$ is represented as the co-free co-monad co-generated by A. It is stated that the corresponding co-Kleisli category is Cartesian closed.

The author presents some motivation of these considerations as a refinement of functional programming with the following features: a synthesis of strict and lazy evaluation, a clean semantics of side effects, and no garbage collector. In the corrigenda some minor misprints are corrected. Reviewed by Vladimir Sazonov.

Nielson, F. (1988). Strictness analysis and denotational abstract interpretation. *Inform. and Comput.*, 76(1):29–92.

[MR90f:68111] Summary: "A theory of abstract interpretation [P. Cousot and R. Cousot, "Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints", in Conference Record, 4th ACM Symposium on Principles of Programming Languages, 1977; per bibl.] is developed for a typed λ-calculus. The typed λ-calculus may be viewed as the 'static' part of a two-level denotational metalanguage for which abstract interpretation was developed by the author ["Abstract interpretation using domain theory", Ph.D. Thesis, Univ. Edinburgh, Edinburgh, 1984; per bibl.; in STACS 86 (Orsay, 1986), 1–20, Lecture Notes in Comput. Sci., 210, Springer, Berlin, 1986; see MR 87a:68018]. The present development relaxes a condition imposed there and this suffices to make the

framework applicable to strictness analysis for the λ-calculus. This shows the possibility of a general theory for the analysis of functional programs and it gives more insight into the relative precision of the various analyses. In particular, it is shown that a collecting (static [Cousot and Cousot, in Conference Record, 6th ACM Symposium on Principles of Programming Languages, 1979; per bibl.]) semantics exists, thus answering a problem left open by G. L. Burn, C. L. Hankin and S. Abramsky [Sci. Comput. Programming 7 (1986), no. 3, 249–278; MR 88b:68010]."

Nielson, H. R. and Nielson, F. (1988). Automatic binding time analysis for a typed λ-calculus. *Sci. Comput. Programming*, 10(2):139–176.
[MR89d:68014] Summary: "A binding time analysis imposes a distinction between the computations to be performed early (e.g., at compile-time) and those to be performed late (e.g., at run-time). For the λ-calculus this distinction is formalized by a two-level λ-calculus. We present an algorithm for static analysis of the binding times of a typed λ-calculus with products, sums, lists and general recursive types. Given partial information about the binding times of some of the subexpressions it will complete that information so that (i) early bindings may be turned into late bindings, but not vice versa, (ii) the resulting two-level λ-expression reflects our intuition about binding times, e.g., that early bindings are performed before late bindings, and (iii) as few changes as possible have been made compared with the initial binding information. The results can be applied in the implementation of functional languages and in semantics-directed compiling."

Ramsay, A. (1988). *Formal methods in artificial intelligence.* Cambridge University Press, Cambridge.
[MR92b:68053] From the preface: "The aim of this book is to bring together the major useful applications of logic in AI in one place, with as consistent a notation as possible and with no prejudices for or against particular schools of thought. Since we are not, in this book, arguing for any specific theory in which we have invested a great deal of time and effort, we hope to report objectively on the most widely used theories. It is almost certain that some of our own views will creep in, particularly in the specific framework used to try to unify the various theories, but at least we are not currently arguing for one language at the expense of another. Our intention is to enable readers to conduct the arguments for themselves, not to tell them the answer.

"The material is fairly dense. It is also organised in a way that reflects its origins in a two-term course for graduate AI students. The book starts with a broad overview of what logic is about, and of what it is that makes a language a logic. The next two chapters cover classical logic, with comparatively little discussion of its relevance to AI. The aim of these two chapters is to establish the specific notation used in the remainder of the book, and to present the major meta-results about soundness, completeness and decidability. Some of the proofs of metatheorems in this part of the book are rather intricate. It is quite possible to read the rest of the book despite skipping these proofs, particularly the ones towards the end of Chapter 3. Of course if one decides to skip them, one will not really be in a position to undertand some of the debate about the merits or otherwise of particular languages, but it will not mean that one cannot follow the rest of the book at all. Chapter 4 deals with theorem proving for classical logic. It is just about possible to read Chapter 4 without reading any of what comes before or after it, and to read what comes before and after without

reading Chapter 4. Chapters 5, 6, 7 and 8 discuss ways in which classical logic is inadequate for describing the phenomena which AI systems have to deal with. The topics covered in these chapters are all concerned with ways of extending classical logic so that it is possible to describe new phenomena, or make new distinctions. These chapters do build on one another, and it would not be a good idea to try to read them in isolation, or in some other order. Finally, Chapter 9 deals with two formalisms which are predicated on the assumption that classical logic is just plain wrong."

Contents: Preface; (1) Logic: proofs and models; (2) Propositional calculus; (3) Predicate calculus; (4) Theorem proving; (5) Modal logic; (6) Temporal reasoning; (7) Nonmonotonic reasoning; (8) Properties; (9) Alternative views; Reflections; Bibliography; Index.

Revesz, G. E. (1988). *Lambda-calculus, combinators, and functional programming.* Cambridge University Press, Cambridge.
[MR90m:68089]

This is an introductory book about λ-calculus and combinators, from a computer scientist's point of view. Its aim is to give a reader, familiar with programming, a theoretical background on functional languages, through the analysis of the fundamental mathematical properties of the λ-calculus. In fact, the λ-calculus is introduced in the book as a functional language; the syntax of the pure λ-calculus is extended with some primitive combinators and constants, which allow one to deal more directly with recursion and list operations. As a consequence of this choice, some classical topics are developed in an original and interesting way. For example, the substitution (of a term for a variable) is no longer viewed as a primitive operation, and so the fundamental reduction rule, the β-rule, splits into five more elementary rules. This permits the construction of a graph-reduction machine in a very natural way.

The section devoted to semantics is not clear enough. The author follows the idea, introduced by Landin, of using λ-calculus, once a semantics for it has been given, as a metalanguage for defining the semantics of programming languages. Thus, he gives the semantics of the system FP of Backus and of a functional subset of the language Miranda, through a translation into λ-calculus. In order to give the semantics of the λ-calculus itself, the author does not utilize a standard notion. He introduces a sort of nondeterministic "operational semantics", which unfortunately he does not discuss in relation to the ordinary approach which uses reduction strategies. In view of the introductory nature of the book, this semantics should be explored more deeply. In particular, a reader could be quite puzzled in trying to understand the meaning of terms not belonging to the λ-I-calculus.

The book is written in a tutorial way. Many examples are given and several informal and interesting discussions are carried out. However, the book can be somewhat difficult to read, especially in the part devoted to the pure λ-calculus (Chapters 2 and 3), because of the ponderous style used in the definitions. The order of presentation sometimes is not carefully arranged. Reviewed by Simona Ronchi Della Rocca.

Filinski, A. (1989). Declarative continuations: an investigation of duality in programming language semantics. In *Category theory and computer science (Manchester, 1989)*, pages 224–249. Springer, Berlin.
[MR90m:68087] Summary: "This paper presents a formalism for including first-class continuations in a programming language as a declarative concept, rather than

an imperative one. A symmetric extension of the typed λ-calculus is introduced, where values and continuations play dual roles, permitting mirror-image syntax for dual categorical concepts such as products and coproducts. An implementable semantic description and a static type system for this calculus are presented. We also give a categorical description of the language, by presenting a correspondence with a system of combinatory logic, similar to a Cartesian closed category, but with a completely symmetrical set of axioms." For the entire collection see MR 90m:68002.

Josephs, M. B. (1989). The semantics of lazy functional languages. *Theoret. Comput. Sci.*, 68(1):105–111.

[MR90k:68107] Summary: "A denotational semantics for the λ-calculus is described. The semantics is continuation-based, and so reflects the order in which expressions are evaluated. It provides a means by which lazy functional languages can be better understood."

Barendregt, H. and Hemerik, K. (1990). Types in lambda calculi and programming languages. In *ESOP '90 (Copenhagen, 1990)*, pages 1–35. Springer, Berlin.

[MR1062898] This item will not be reviewed individually. For details of the collection in which this item appears see 91a:68003.

Barendregt, H. P. (1990). Functional programming and lambda calculus. In *Handbook of theoretical computer science, Vol. B*, pages 321–363. Elsevier, Amsterdam.

[MR1127192] This item will not be reviewed individually. For details of the collection in which this item appears see 92d:68002.

Bloom, B. (1990). Can LCF be topped? Flat lattice models of typed λ-calculus. *Inform. and Comput.*, 87(1-2):264–301.
[MR91h:68110]

This paper continues investigations begun by Plotkin on the connections between the denotational and operational semantics of a simple programming language based on LCF, Dana Scott's language for computable functions [G. D. Plotkin, Theoret. Comput. Sci. 5 (1977/78), no. 3, 223–255; MR 81e:68016]. The programming language, PCF, is a typed lambda calculus with integers and Booleans as ground types, supplemented with arithmetic, logical, and fixed-point operators to make it computationally viable. Executable programs correspond to closed terms; programs, together with their inputs, match closed ground terms; observed outputs correspond to numerals.

The author considers carrying out Plotkin's program for PCF using Scott models built from flat lattices rather than flat continuous partial orderings (cpo's). PCF-style languages, together with their denotational and operational semantics, are considered with respect to the presence or absence of three natural and desirable properties: computational adequacy, full abstraction, and denotational universality. Intuitively, a language is computationally adequate if any two terms having the same denotation are behaviorally interchangeable in any computing environment. Conversely, a language is fully abstract if any two terms which are behaviorally interchangeable have the same denotation. A language is denotationally universal if every computable semantic value is definable.

For models based on flat cpo's, Plotkin had shown that extending PCF with a parallel conditional resulted in a fully abstract language, and that further extending PCF with a continuous approximation to an existential quantifier resulted in denotational universality. The author shows that Plotkin's program cannot be carried out for models based on flat lattices without the loss of essential properties usually associated with functional programming languages. In particular, it is shown that any language which is fully abstract is not operationally extensional, and that no single-valued extension of PCF is denotationally universal for flat lattice models.

"In summary, this paper is the converse to Plotkin's [op. cit.]. Plotkin showed how a well-chosen denotational semantics can match an operational semantics, and demonstrated that everything can work right. The results of this paper show that a poorly-chosen denotational semantics can fail to match any good operational semantics (although it can be tantalizingly close to matching), and demonstrates that the success of Plotkin's program is far from automatic."

While building on the work of Plotkin, the paper is essentially self-contained, and provides a convenient review of PCF and Scott domains which should enable the reader to follow all the constructions and proofs. Reviewed by E. Ray Lucas.

Böhm, C. and Intrigila, B. (1990). Multisets and polynomials: an algebraic approach to functional programming. *Rend. Mat. Appl. (7)*, 10(4):867–898 (1991).

[MR92m:68011] Summary: "Combinatory algebras are equationally defined and they have the peculiar feature, not shared by usual algebras, that every element of their supports represents a mapping of the supports into themselves. Such algebras, considered as formal systems, are very powerful functional programming languages, and such formal systems are, at the same time, confluent reduction systems. A combinatory algebra $\mathcal{C}$ is presented in the most natural way, i.e. as an extension of an algebra based on $\mathbf{N}$ (the natural integers), with the adjunction of two new constants C^* (for the exponentiation) and A (for the addition). Multisets over $\mathbf{N}$ and polynomials over $\mathbf{N}$ are presented, like $\mathbf{N}$ itself, as instances of semirings. In order to represent the algorithms for the reduction of multisets and/or polynomials to a canonical form, such algebraic structures are mapped into subsets of $\mathcal{C}$ where all algebraic equalities of the source algebras are satisfied. As reduction in such subsets is also Noetherian, there is no need to construct an algorithm to bring the objects to a canonical form, since the reduction rules of $\mathcal{C}$ perform this task automatically."

Cheng, M. H. M., van Emden, M. H., and Richards, B. E. (1990). On Warren's method for functional programming in logic. In *Logic programming (Jerusalem, 1990)*, pages 546–560. MIT Press, Cambridge, MA.

[MR1130659] This item will not be reviewed individually. For details of the collection in which this item appears see 92d:68014.

Galmiche, D. (1990). Constructive system for automatic program synthesis. *Theoret. Comput. Sci.*, 71(2):227–239.

[MR91d:68083] Summary: "This paper describes a constructive system based on a particular typed λ-calculus with constants and λ-expressions as types. This theory

is principally based on intuitionistic-type theory and on the general idea of programming proofs, which means extracting programs from constructive proofs. It can be considered as a basis of a system for automated program synthesis by developing adapted techniques of programming. It allows one partially to automate the construction of programs from different specifications by a type-driven strategy. Moreover, the corresponding logical theory appears to be a good basis for a higher-order calculus."

Krivine, J.-L. (1990). Opérateurs de mise en mémoire et traduction de Gödel. *Arch. Math. Logic*, 30(4):241–267.
[**MR91j:03011**] The author begins this paper by observing that the strategy of left-most reduction has the unfortunate property that arguments to functions get evaluated each time they are used. The solution he proposes is to define the notion of "storage operator for a given data type". If T is a storage operator appropriate for the type of τ then evaluation of $\varphi\tau$ is replaced by evaluation of $T\tau\varphi$. The main result is to show that if $D[x]$ is a type in a given extension of the second-order lambda calculus, then we can find an inhabited type $\forall x\{D^*[x] \to {}_{00}D[x]\}$, each member of which is a storage operator for terms of types D (here D^* is the Gödel translation of D, and ${}_0(-)=(-)\to 0$).

Unfortunately it is not clear from the paper whether the author regards these results as being of essentially theoretical interest, or whether he believes them to have practical implications for the implementation of programming languages. It is a shame that there is no attempt to relate this work to the techniques used to avoid the multiple evaluation of expressions in existing implementations of lazy languages (such as the structure sharing implicit in graph reduction). It is also sad that the paper also lacks an account of how this work relates to other work on the simulation of a call-by-value evaluation strategy using call-by-name. Reviewed by Edmund Robinson.

Krivine, J.-L. and Parigot, M. (1990). Programming with proofs. *J. Inform. Process. Cybernet.*, 26(3):149–167.
[**MR91j:68086**] Summary: "We present a logical framework for constructing a programming language in which one programs by proofs, in order to ensure program correctness. The general programming methodology, based on second-order intuitionistic logic, is the following: (i) the data types are defined by second-order formulas from which a representation of the data in lambda-calculus can be extracted automatically; (ii) the specifications of the programs are expressed by equations; (iii) a correct program in lambda-calculus is automatically extracted from a proof of the fact that a function satisfying the equations has the intended type."

Li, W. (1990). A type-theoretic approach to program development. *J. Comput. Sci. Tech. (English Ed.)*, 5(3):209–224.
[**MR93c:68062**] The author proposes a new type theory called ALT in which to perform development and verification in the style of the Edinburgh logical framework. Reviewed by Edmund Robinson.

Nielson, H. R. and Nielson, F. (1990). Functional completeness of the mixed λ-calculus and combinatory logic. *Theoret. Comput. Sci.*, 70(1):99–126.

Fourth Workshop on Mathematical Foundations of Programming Semantics (Boulder, CO, 1988).

[**MR91g:68109**] Summary: "Functional completeness of combinatory logic means that every λ-expression may be translated into an equivalent combinator expression; this is the theoretical basis for the implementation of functional languages on combinator-based abstract machines. To obtain efficient implementations it is important to distinguish between early and late binding times, i.e., to distinguish between compile-time and run-time computations. We therefore introduce a two-level version of the λ-calculus where this distinction is made in an explicit way. Turning to combinatory logic, we wish only to generate combinator code for the run-time computations. The two-level version of combinatory logic therefore will be a mixed λ-calculus and combinatory logic. A previous paper has shown that (a natural formulation of) the mixed λ-calculus and combinatory logic is not functionally complete, but only corresponds to a strict subset of the two-level λ-calculus. In this paper we extend the mixed λ-calculus and combinatory logic with a new combinator, Ψ, and show that this suffices for the mixed λ-calculus and combinatory logic to be functionally complete. However, the new combinator may not always be implementable, and we therefore discuss conditions under which it can be dispensed with."

Piperno, A. and Tronci, E. (1990). Regular systems of equations in λ-calculus. *Internat. J. Found. Comput. Sci.*, 1(3):325–339. Third Italian Conference on Theoretical Computer Science (Mantova, 1990).

[**MR92a:03018**] Section 1 of this paper surveys what was known about the solvability of systems of equations in lambda calculus. Section 2 introduces a new class of "regular systems" and shows that for these systems solvability is decidable. Section 3 gives some interesting examples of problems that can be solved by means of regular systems and Section 4 gives examples of some undecidable systems.

The authors claim that regular systems have enough expressive power to be used as the core of an equational programming language in which the compiler is the algorithm constructing the system's solution. Also, they claim that regular systems are not easily generalisable, in meaningful ways, without running into undecidable solvability problems. For the entire collection see MR 91m:68005. Reviewed by Martin W. Bunder.

Stoughton, A. (1990). Equationally fully abstract models of PCF. In *Mathematical foundations of programming semantics (New Orleans, LA, 1989)*, pages 271–283. Springer, Berlin.

[**MR91m:68124**] A term of Plotkin's applied typed calculus PCF is said to be operationally less defined than another term if whenever the first converges to a constant in a ground context, then the second converges to the same constant in that context. R. Milner's [Theoret. Comput. Sci. 4 (1977), no. 1, 1–22; MR 58 #16238] order-extensional model of PCF is inequationally fully abstract, which means that a term is operationally less defined than another whenever the meaning of the first is less than that of the second. Two terms are said to be operationally equivalent if each is operationally less defined than the other. A model is said to be equationally fully abstract if any two operationally equivalent terms are mapped to the same semantic

value. Milner's model is equationally fully abstract. The question arises if there is an equationally fully abstract model that is not inequationally fully abstract. The author studies the category of extensional equationally fully abstract models and shows that the answer to this question is affirmative. For the entire collection see MR 91e:68008. Reviewed by Dorothy Bollman.

Breazu-Tannen, V., Coquand, T., Gunter, C. A., and Scedrov, A. (1991). Inheritance as implicit coercion. *Inform. and Comput.*, 93(1):172–221. Selections from the 1989 IEEE Symposium on Logic in Computer Science.

[MR92c:68115] Summary: "We present a method for providing semantic interpretations for languages with a type system featuring inheritance polymorphism. Our approach is illustrated on an extension of the language Fun of Cardelli and Wegner, which we interpret via a translation into an extended polymorphic lambda calculus. Our goal is to interpret inheritances in Fun via coercion functions which are definable in the target of the translation. Existing techniques in the theory of semantic domains can then be used to interpret the extended polymorphic lambda calculus, thus providing many models for the original language. This technique makes it possible to model a rich type discipline which includes parametric polymorphism and recursive types as well as inheritance. A central difficulty in providing interpretations for explicit type disciplines featuring inheritance in the sense discussed in this paper arises from the fact that programs can type-check in more than one way. Since interpretations follow the type-checking derivations, coherence theorems are required: that is, one must prove that the meaning of a program does not depend on the way it was type-checked. Proofs of such theorems for our proposed interpretation are the basic technical results of this paper. Interestingly, proving coherence in the presence of recursive types, variants, and abstract types forced us to reexamine fundamental equational properties that arise in proof theory (in the form of commutative reductions) and domain theory (in the form of strict vs. nonstrict functions)." For the entire collection see MR 92a:68003.

Cardone, F. (1991). Recursive types for FUN. *Theoret. Comput. Sci.*, 83(1, Logic Semantics Theory Program.):29–56. 16th International Colloquium on Automata, Languages, and Programming (Stresa, 1989).

[MR92i:68091] The language FUN, introduced by Cardelli and Wegner in 1985, is a typed polymorphic λ-calculus with record and variant types, a notion of subtyping, and quantifiers ranging over subtypes of a given type (in the present paper only universal quantifiers are considered). The author shows that it is consistent to allow recursive type definitions in FUN. The methodology followed here is a refinement of one introduced by Coppo. The author builds, by means of the classical colimit construction, a Scott domain D in which the underlying language of untyped terms of FUN is interpreted; then he interprets types as "profinite" partial equivalence relations (pers), where "profinite" means here that the per is the limit of a denumerable sequence of approximate relations built following the structure of D. A model for the (strongly typed) language is obtained by interpreting a typed term as the equivalence class of its untyped meaning modulo the per which interprets the type. For the entire collection see MR 92d:68004. Reviewed by Chantal Berline.

Chen, W. and Warren, D. S. (1991a). Compilation of predicate abstractions in higher-order logic programming. In *Programming language implementation and logic programming (Passau, 1991)*, pages 287–298. Springer, Berlin.
[MR1136820] This item will not be reviewed individually. For details of the collection in which this item appears see 92g:68009.

Chen, W. and Warren, D. S. (1991b). An intensional logic of (multi-arity) set abstractions. In *Logic programming (Paris, 1991)*, pages 97–110. MIT Press, Cambridge, MA.
[MR94c:03037] Summary: "In higher-order logic programming, set abstractions are often used for auxiliary predicates that do not have to be named. This paper presents a simple logic for reasoning about set abstractions. The semantics is more tractable than that of other systems in that the equality of terms corresponds to a α-equivalence. A strongly restricted variant of β-reduction, called top β-reduction, is introduced for evaluating set memberships as atomic formulas. This enables us to incorporate set abstractions into a logic without the complications of higher-order functions." For the entire collection see MR 94b:68013.

Gomard, C. K. and Jones, N. D. (1991a). Erratum: "A partial evaluator for the untyped lambda-calculus". *J. Funct. Programming*, 1(3):373.
[MR92h:68054b] See (Gomard and Jones, 1991b).

Gomard, C. K. and Jones, N. D. (1991b). A partial evaluator for the untyped lambda-calculus. *J. Funct. Programming*, 1(1):21–69.
[MR92h:68054a] From the summary: "This article describes theoretical and practical aspects of an implemented self-applicable partial evaluator for the untyped lambda-calculus with constants and a fixed point operator. To the best of our knowledge, it is the first partial evaluator that is simultaneously higher-order, nontrivial, and self-applicable.

"Partial evaluation produces a residual program from a source program and some of its input data. When given the remaining input data the residual program yields the same result that the source program would when given all its input data. Our partial evaluator produces a residual lambda-expression given a source lambda-expression and the values of some of its free variables. By self-application, the partial evaluator can be used to compile and to generate stand-alone compilers from a denotational or interpretive specification of a programming language."

Jouannaud, J.-P. (1991). Executable higher-order algebraic specifications. In *STACS 91 (Hamburg, 1991)*, pages 16–25. Springer, Berlin.
[MR92f:68115] Summary: "Conventional algebraic specifications are first-order. Using higher-order equations in combination with first-order ones raises several fundamental model-theoretic and proof-theoretic questions. The model theory of higher-

order equations is well understood (see work of M. Wirsing [Handbook of theoretical computer science, Vol. B, see pp. 675–780, MIT Press, Cambridge, MA, 1990] for a survey of algebraic specifications). The proof theory of higher-order equations is equally well understood; it requires higher-order matching and higher-order rewriting, therefore allowing a simple execution model. Higher-order variables may be instantiated by functions described by λ-expressions, bringing in λ-calculus, whose execution model is again rewriting (β-redexes). Hence rewriting is at the heart of all three execution models, which makes their combination quite simple on the operational side. The main question reviewed in this paper is whether the Church-Rosser and termination properties of these three execution models are preserved within their combination. We see that the answer is to a large extent positive." For the entire collection see MR 92b:68019.

Kanellakis, P. C., Mairson, H. G., and Mitchell, J. C. (1991). Unification and ML-type reconstruction. In *Computational logic*, pages 444–478. MIT Press, Cambridge, MA.

[MR1132711] This item will not be reviewed individually. For details of the collection in which this item appears see 92d:03002.

Leiss, H. (1991). Polymorphic constructs in natural and programming languages. In *Logics in AI (Amsterdam, 1990)*, pages 348–365. Springer, Berlin.

[MR1099639] This item will not be reviewed individually. For details of the collection in which this item appears see 91k:68012.

Mairson, H. G. (1991). Outline of a proof theory of parametricity. In *Functional programming languages and computer architecture (Cambridge, MA, 1991)*, pages 313–327. Springer, Berlin.

[MR93g:03013] A polymorphic function is one that is applicable to data of various types. Intuitively, a polymorphic function is parametric if its behavior is "uniform" over all possible types. This paper develops a theory of parametricity, previously explored by J. C. Reynolds [in Information processing 83 (Paris, 1983), 513–523, North-Holland, Amsterdam, 1983; CCA 1984:44017] in a syntactic way. This gives a straightforward method for demonstrating certain equational properties of polymorphic functions written in the polymorphic lambda calculus. For the entire collection see MR 92h:68006. Reviewed by John C. Mitchell.

Miller, D. (1991a). A logic programming language with lambda-abstraction, function variables, and simple unification. *J. Logic Comput.*, 1(4):497–536.

[MR93b:68011] Summary: "It has been argued elsewhere that a logic programming language with function variables and λ-abstractions within terms makes a good meta-programming language, especially when an object-language contains notions of bound variables and scope. The λ Prolog logic programming language and the related Elf and Isabelle systems provide meta-programs with both function variables and λ-abstractions by containing implementations of higher-order unification. This paper presents a logic programming language, called L_λ, that also contains both function variables and λ-abstractions, although certain restrictions are placed on occurrences of function variables. As a result of these restrictions, an implementation of L_λ does not need to implement full higher-order unification. Instead, an extension to first-order unification that respects bound variable names and scopes is all that is required. Such unification problems are shown to be decidable and to possess most general unifiers when unifiers exist. A unification algorithm and logic programming interpreter are described and proved correct. Several examples of the use of L_λ as a meta-programming language are presented."

Miller, D. (1991b). A logic programming language with lambda-abstraction, function variables, and simple unification. In *Extensions of logic programming (Tübingen, 1989)*, pages 253–281. Springer, Berlin.
[MR1101608] This item will not be reviewed individually. For details of the collection in which this item appears see 91k:68007.

Miller, D. (1991c). Unification of simply typed lambda-terms as logic programming. In *Logic programming (Paris, 1991)*, pages 255–269. MIT Press, Cambridge, MA.
[MR94f:68058] Summary: "The unification of simply typed λ-terms modulo the rules of β- and η-conversions is often called 'higher-order' unification because of the possible presence of variables of functional type. This kind of unification is undecidable in general and if unifiers exist, most general unifiers may not exist. In this paper, we show that such unification problems can be coded as a query of the logic programming language L_λ in a natural and clear fashion. In a sense, the translation only involves explicitly axiomatizing in L_λ the notions of equality and substitution of the simply typed λ-calculus: the rest of the unification process can be viewed as simply an interpreter of L_λ searching for proofs using those axioms." For the entire collection see MR 94b:68013.

Miller, D., Nadathur, G., Pfenning, F., and Scedrov, A. (1991). Uniform proofs as a foundation for logic programming. *Ann. Pure Appl. Logic*, 51(1-2):125–157. Second Annual IEEE Symposium on Logic in Computer Science (Ithaca, NY, 1987).
[MR93b:03044] This paper continues earlier efforts by the authors to extend the logic programming paradigm by strengthening its logical foundations. In practice, the dominant paradigm has been Prolog, with its language based on first-order Horn clauses, and its interpreter based on SLD-resolution and unification. The need to extend this paradigm stems from the widely recognized fact that first-order Horn clauses offer no direct and natural support for several desirable forms of abstraction found in other programming languages. For example, first-order Horn clauses do not support modules, abstract data types or higher-order functions. Thus the paper operates within the highly active area of research into synthesizing systems combining aspects of logic programming and functional programming. The authors also follow the currently promising direction of using typed lambda calculus and intuitionistic proof-theoretic methods in the realization of abstraction mechanisms in logic programming.

The authors distinguish three broad approaches to supplying the missing features in languages like Prolog: mixing constructs from other languages, modifying existing interpreters to include various nonlogical features (such as call, univ and functor in Prolog), and extending the logic programming paradigm to include richer logics than first-order Horn clauses (thus providing a purely logical foundation for the missing abstraction mechanisms). The authors prefer the third approach since the first two "have the disadvantage that they clutter up the semantics of the language, obscure the declarative reading of programs and move the language far from its basis in logic". Thus, in implementing features that move away from the current Prolog state of the art, the authors take what H. Gallaire has called the "high road" in contrast to the "low road" of the first two approaches [H. Gallaire, in Symposium on logic programming (Boston, MA, 1985), 88–96, IEEE Comput. Soc. Press, Washington, DC, 1985].

Extending the logic programming paradigm by strengthening its logical foundations requires a choice from a broad range of logical systems with Horn clauses at one extreme and full quantificational logic with a generalized theorem prover at the other extreme. The success of this approach also involves stating a criterion for determining when a logical theory is an adequate basis for a logic programming language. The guiding principle of the authors in finding such a criterion has been that a certain success/failure semantics for the logical connectives lies at the heart of logic programming. "The basic observation here is that logic programs are intended to specify a search behavior and that this fact is just as central to logic programming as the fact that it makes use of symbolic logic for some of its syntax and metatheory."

The authors attempt to formalize the search behavior of the logical connectives within a formal framework that uses simply typed lambda calculus for term construction and cut-free sequent calculi for proof construction. Classical, intuitionistic, and minimal provability relations are distinguished in the usual way. Given a formal logic that is to serve as a foundation for a logic programming language, we can consider the set of possible program clauses and the set of possible goals or queries. Given a program, a goal, and a provability relation, we could require that an interpreter for the language succeeds in this situation if and only if the goal is provable from the program. A key contention of the paper is that such a general unrestricted notion of success is unsatisfactory as a foundation for logic programming.

In particular, the notion of provability must be restricted so that the meaning of the logical connectives supports an interpretation involving goal-directed search. Also, many provability relations do not satisfy the existential property which requires constructive instantiation of terms for goals involving existential quantifiers. To provide a satisfactory provability relation for logic programs, the authors offer the notion of a uniform proof, which is an intuitionistic proof in which each occurrence of a sequent whose succedent contains a nonatomic formula is the lower sequent of the inference figure that introduces its top-level connective.

To motivate the notion of a satisfactory provability relation for logic programming, a simple abstract interpreter (theorem prover) for logic programs is described, in which uniform proofs reflect the search instructions associated with the logical connectives. The operational semantics is formalized by requiring that the interpreter succeed with a given program and a specific goal if and only if there is a uniform proof of the sequent with the program as antecedent and the goal as succedent. An abstract logic programming language is then defined as a triple consisting of the set of possible program clauses, the set of possible goals or queries, and a provability relation, such that for all programs and goals, a goal is provable from a program if and only if the interpreter succeeds with the program and goal.

The authors present four abstract programming languages: first-order and higher-order positive Horn clauses, and first-order and higher-order versions of a new class of formulas called hereditary Harrop formulas. The authors also indicate several logical systems which fail to be abstract logic programming languages. A detailed proof is given that higher-order Harrop formulas interpreted using either intuitionistic or minimal logic form an abstract programming language. Possible applications to the problem of supplying the missing abstraction mechanisms are then considered. Definitions of the four languages are collected in an appendix, together with observations on the strength of their provability relations, their Herbrand universes, and the containment relations (or lack thereof) among them.

The authors note that this paper corrects Theorem 3 of a preliminary version of this paper [Miller, Nadathur and Scedrov, in Symposium on logic in computer science, 98–105, IEEE Comput. Soc. Press, Washington, DC, 1987; see MR 88k:68012]. Another paper by Miller is a useful supplement for more details on the formal logic and examples addressing the missing abstraction mechanisms [in Logic and computer science, 329–359, Academic Press, London, 1990; per revr.]. For the entire collection see MR 91k:03004. Reviewed by E. Ray Lucas.

Nielson, H. R. and Nielson, F. (1991). Using transformations in the implementation of higher-order functions. *J. Funct. Programming*, 1(4):459–494.
[MR92i:68024]

Summary: "Traditional functional languages do not have an explicit distinction between binding times. It arises implicitly, however, as one typically instantiates a higher-order function with the arguments that are known, whereas the unknown arguments remain to be taken as parameters. The distinction between 'known' and 'unknown' is closely related to the distinction between binding times such as 'compile-time' and 'run-time'. This leads to the use of a combination of polymorphic type inference and binding time analysis for obtaining the required information about which arguments are known.

"Following the current trend in the implementation of functional languages we then transform the run-time level of the program (not the compile-time level) into categorical combinators. At this stage we have a natural distinction between two kinds of program transformations: partial evaluation, which involves the compile-time level of our notation, and algebraic transformations (i.e., the application of algebraic laws), which involves the run-time level of our notation.

"By reinterpreting the combinators in suitable ways we obtain specifications of abstract interpretations (or data flow analyses). In particular, the use of combinators makes it possible to use a general framework for specifying both forward and backward analyses. The results of these analyses will be used to permit program transformations that are not applicable under all circumstances.

"Finally, the combinators can also be reinterpreted to specify code generation for various (abstract) machines. To improve the efficiency of the code generated, one may apply abstract interpretations to collect extra information for guiding the code generation. Peephole optimizations may be used to improve the code at the lowest level."

Pfenning, F. (1991). Logic programming in the LF logical framework. In *Logical frameworks (Sophia-Antipolis, 1990)*, pages 149–181. Cambridge Univ. Press, Cambridge.

[**MR1139784**] This item will not be reviewed individually. For details of the collection in which this item appears see 92f:68003.

Pfenning, F. and Lee, P. (1991). Metacircularity in the polymorphic λ-calculus. *Theoret. Comput. Sci.*, 89(1):137–159. 1989 International Joint Conference on Theory and Practice of Software Development (TAPSOFT '89) (Barcelona, 1989).

[**MR93a:03015**] Summary: "We consider the question of whether a useful notion of metacircularity exists for the polymorphic λ-calculus. Even though complete metacircularity seems to be impossible, we obtain a close approximation to a metacircular interpreter. We begin by presenting an encoding for the Girard-Reynolds second-order polymorphic λ-calculus in the third-order polymorphic λ-calculus. The encoding makes use of representations in which abstractions are represented by abstractions, thus eliminating the need for the explicit representation of environments. We then extend this construction to encompass all of the ω-order polymorphic λ-calculus (F_ω). The representation has the property that evaluation is definable, and furthermore that only well-typed terms can be represented and thus type inference does not have to be explicitly defined. Unfortunately, this metacircularity result seems to fall short of providing a useful framework for typed metaprogramming. We speculate on the reasons for this failure and the prospects for overcoming it in the future. In addition, we briefly describe our efforts in designing a practical programming language based on F_ω." For the entire collection see MR 92g:68006. Reviewed by John C. Mitchell.

Wand, M. (1991). Type inference for record concatenation and multiple inheritance. *Inform. and Comput.*, 93(1):1–15. Selections from the 1989 IEEE Symposium on Logic in Computer Science.

[**MR92d:03008**] Summary: "We show that the type inference problem for a lambda calculus with records, including a record concatenation operator, is decidable. We show that this calculus does not have principal types, but does have finite complete sets of types: that is, for any term M in the calculus, there exists an effectively generable finite set of type schemes such that every typing for M is an instance of one of the schemes in the set. We show how a simple model of object-oriented programming, including hidden instance variables and multiple inheritance, may be coded in this calculus. We conclude that type inference is decidable for object-oriented programs, even with multiple inheritance and classes as first-class values." For the entire collection see MR 92a:68003.

Zimmer, R. M. (1991). *Zur Pragmatik eines operationalisierten λ-Kalküls als Basis für interaktive Reduktionssysteme*. Gesellschaft für Mathematik und Datenverarbeitung mbH, St. Augustin.
[**MR93e:68001**]

From the summary (translated from the German): "In the first part of this book we discuss languages, in particular programming languages, and their logical foundations, and also schematic operation with figures in formal calculi as a simple well-founded basis for mathematics and computer science. We identify fundamental properties of programming languages and attempt a rough classification of languages. In this classification reduction languages are those languages with a very simple syntax whose semantics are defined in the language itself by an inherent nonsequential procedure with the Church-Rosser property.

"In the second part we introduce mechanisms, concepts and realizations for the use of nonsequential programming for reduction systems and we show that reduction systems can be employed for the simple organization of a multiprocessor system.

"In the third part we introduce concepts of types and discuss them using the example of OREL/2. A static type of inference method is represented as a formal type calculus in which OREL/2 programs can be transformed into OREL/2 programs. This type system is more flexible than the standard Damas-Milner type system which is used in almost all modern functional languages. The practical aspect of the introduction of recursive types and local type connections needs more intensive research. For compiling systems, such type systems are probably too complicated in practice. However, in an interactive reduction system substantially more complex type systems can be treated consistently and meaningfully ('second-order typed λ-calculus', universally and existentially quantified types, data abstractions and module concepts, 'constructions', etc.). The implementation of the dynamic type calculus in OREL/2 with 'type:type' still needs completion, but will permit the investigation of very powerful type concepts and their use in a practically usable programming system."

Bai, M. and Blair, H. A. (1992). General model-theoretic semantics for higher-order Horn logic programming. In *Logic programming and automated reasoning (St. Petersburg, 1992)*, pages 320–331. Springer, Berlin.

[MR94j:68032] Summary: "We introduce model-theoretic semantics for a higher-order Horn logic programming language. One advantage of logic programs over conventional nonlogic programs has been that the least fixpoint is equal to the least model; therefore it is associated with logical consequences and has a meaningful declarative interpretation. In the simple theory of types on which the higher-order Horn logic programming language is based, a domain is dependent on interpretation. To define the $T_{\mathcal{P}}$ operator for a logic program $\mathcal{P}$, we need a fixed domain without regard to interpretation, which is usually taken to be a set of atomic propositions. We build a semantics where we can fix a domain while changing interpretations. We also develop a fixpoint semantics based on our model, and show that we can get the least fixpoint, which is the least model. Using this fixpoint we prove the completeness of the interpreter of our language given in a paper by D. A. Miller et al. [Ann. Pure Appl. Logic 51 (1991), no. 1-2, 125–157; MR 93b:03044]." For the entire collection see MR 94c:68004.

Danvy, O. and Filinski, A. (1992). Representing control: a study of the CPS transformation. *Math. Structures Comput. Sci.*, 2(4):361–391. 1990 ACM Conference on Lisp and Functional Programming (Nice, 1990).

[MR94f:68121] The applicative order α-calculus that forms the core of languages such as Scheme and Standard ML is considered. The usual presentation of the continuation-passing style (CPS) transformation (Plotkin, 1975) is concise and simple, but tends to yield unreasonably large terms containing a lot of "administrative redexes".

In this paper the stepwise derivation of a one-pass CPS transformer from Plotkin's two-pass equational specification is described. It is shown that the one-pass transformer computes a result $\beta\eta$-equivalent to the original Fischer/Plotkin transformation, and is easily extended to conditional expressions, recursive definitions and similar constructs. The reduction properties of the transformer are investigated. A theorem that captures Plotkin's "indifference" and "simulation" theorems for the original CPS translation is proved. Two new control operators, shift and reset, which are similar to, but more general than, Scheme's call/cc and allow the additional power of general CPS to be exploited in direct style programs, are introduced. For the entire collection see MR 93h:68002. Reviewed by Victor N. Kas'yanov.

Feferman, S. (1992). Logics for termination and correctness of functional programs. In *Logic from computer science (Berkeley, CA, 1989)*, pages 95–127. Springer, New York.
[**MR1236007**] This item will not be reviewed individually. For details of the collection in which this item appears see 94c:03004.

Gateley, J. and Duba, B. F. (1992). Call-by-value combinatory logic and the lambda-value calculus. In *Mathematical foundations of programming semantics (Pittsburgh, PA, 1991)*, pages 41–53. Springer, Berlin.
[**MR1251308**] This item will not be reviewed.

Gunter, C. A. (1992). *Semantics of programming languages.* MIT Press, Cambridge, MA. Structures and techniques.
[**MR94f:68125**] This book will probably become a standard text and reference for those interested in semantics of languages with higher order functions. It is very well written and contains most of the topics of current interest. Many details usually omitted from textbooks and papers are treated here with care. (One class of questions which is ignored concerns decidability of various properties.) The topics covered include: the simply typed λ-calculus with recursive definitions, using environment and categorical models; the fixed-point semantics of recursive definitions, domain theory and its application to semantics, types and their relationship to the evaluation of computer programs; universal domains and how they can be used to model types; subtype polymorphism; fixed points of functors, bifinite domains, powerdomains; parametric polymorphism. Although some category theory is used, it is well motivated by the applications; similarly, many proofs are either omitted or simplified by the omission of routine details. Thus, the reader is able to move quickly through a section in order to grasp the main points. Further, most proofs have a very constructive character.

One of the best chapters is the introduction. A nice discussion of the difference between semantics of natural languages and semantics of formal (computer) languages is given, and various approaches to formal semantics are described. In particular, the defects of giving an "informal" semantics using English, say, are spelled out via the example of ALGOL 60. The approach to giving a semantics by specifying a transition relation on program fragments is discussed, and applied to a simple imperative language, as well as to a minimal untyped λ-calculus. The deficiency of this kind of

semantics, for both the imperative language and the λ-calculus, is also made clear by considering the objectives of "compositionality" and meaningful "denotations". If the meaning of, say, $[\![\text{while } B \text{ do } C]\!]$ is a partial function on memories (where a memory is a function from the set of identifiers to the set of values), then this function satisfies the equation

$$[\![\text{while } B \text{ do } C]\!](m) = \begin{cases} m & \text{if } [\![B]\!](m) = \text{false} \\ [\![\text{while } B \text{ do } C]\!](C(m)) & \text{otherwise.} \end{cases}$$

Thus this function is a fixed point for a function $[M \rightsquigarrow M] \to [M \rightsquigarrow M]$, leading to the topic of fixed points and domain theory. One natural question is: What mathematical objects are denoted by programs? In the simple imperative language, it seems that a satisfactory answer is: partial functions on memories. But the transition semantics given for the λ-calculus gives no useful answer. Landin's SECD-machine for the evaluation of λ-terms is described. The need for a simplified "natural" semantics becomes apparent.

The detailed discussion of semantics begins in Chapter 2 with the simply typed λ-calculus. The syntax of the λ-expressions is given in considerable detail along with three typing rules. A list of 10 equational rewrite rules is given and a completeness theorem is proved using the class of "standard models", in which types are interpreted by nonempty sets and λ-terms by set-theoretic functions.

In the following chapter, Cartesian closed categories are introduced and categorical models of the λ-calculus are defined. Three justifications of categories as a tool in semantics are given at the end of this chapter. A good example of the use of categories to clarify syntactical substitution can be seen by considering the difference in clarity of the two sides of the conclusion of the "substitution lemma" 3.9 (page 82).

$$[\![H \triangleright [N/x]M : t]\!] = f\langle \mathrm{id}_{A}, g\rangle : A \to C$$

In Chapter 4 a thorough discussion is given of an order-theoretic fixed point semantics for Scott's language PCF. The mathematical facts concerning the existence of least fixed points for continuous functions on cpo's are presented clearly and concisely, and a mathematical justification for the use of least fixed points (vs. greatest, or others) is given: the least fixed point operator is the only uniform fixed point operator. A soundness and adequacy theorem is proved, showing the equivalence of the fixed point semantics to a transition-based one.

In Chapter 5, two theories of finite approximation are introduced: the dI-domains of Beery, and the bounded-complete domains. The dI-domains determine a semantics which is used in the following chapter to show that the standard cpo-semantics of PCF is not "fully abstract".

In Chapter 7, the λ-calculus is extended to include assignment statements. This necessitates the introduction of reference types into the syntax and locations into the semantics. An operational semantics is given relative to some given function "new", from memories to locations. (There is a potentially confusing typographical error here: memories are functions from locations to values, not from variables to values.)

Models of the untyped λ-calculus based on information systems are discussed in Chapter 8. Such models are used to find a nontrivial solution of the domain equation $D = [D \to D]$. Other methods to solve circular domain equations are considered as well, using retractions and continuous functors.

Chapter 9 deals with subtypes and subtype polymorphism. The last two chapters 10 and 11 deal with some of the mathematics of topics in domain theory, such as powerdomains, and parametric polymorphism.

As noted in the preface, there is too much here for even a two-semester course: Chapters 1, 2 and 4 are recommended for a one-semester course. And despite the author's efforts, unless one has been reading from page 1, it is difficult to discover the meaning of all the symbolism. For example, $\rho[x \mapsto d]$ occurs in the list of notations, but $\theta[l \mapsto V]$ does not, although it means something slightly different. Further, it takes a couple of passes to find the page number of all-important tables. Nevertheless, the book rewards the patient reader by providing a wealth of information and a good deal of insight into the many problems that occur in giving meaning to programming languages.

Errata. Everywhere the author writes "if, and only if," giving the comma manufacturers plenty of work. Page 32: "appl" should be "app". Page 49, line 13: $d \notin [\![t]\!]$ should be $d \in [\![t]\!]$. Page 179, line 9: the first occurrence of "N" should be "M". Page 291, line 3b: subscripts missing from M's. Page 341, line 6: Space between **C** and "and". Page 369, rule for Π-Introduction: the symbol $\vdash_{tp:\,tp}$ should be tp. Reviewed by Stephen L. Bloom.

Hannan, J. (1992). Implementing λ-calculus reduction strategies in extended logic programming languages. In *Extensions of logic programming (Stockholm, 1991)*, pages 193–219. Springer, Berlin.
[MR1238102] This item will not be reviewed individually. For details of the collection in which this item appears see 94d:68002.

Mairson, H. G. (1992). Quantifier elimination and parametric polymorphism in programming languages. *J. Funct. Programming*, 2(2):213–226.
[MR93g:68078] Summary: "We present a simple and easy-to-understand explanation of ML type inference and parametric polymorphism within the framework of type monomorphism, as in the first order typed lambda calculus. We prove the equivalence of this system with the standard interpretation using type polymorphism, and extend the equivalence to include polymorphic fixpoints. The monomorphic interpretation gives a purely combinatorial understanding of the type inference problem, and is a classic instance of quantifier elimination, as well as an example of Gentzen-style cut elimination in the framework of the Curry-Howard propositions-as-types analogy."

Révész, G. E. (1992). A list-oriented extension of the lambda-calculus satisfying the Church-Rosser theorem. *Theoret. Comput. Sci.*, 93(1):75–89.
[MR93d:03023] In this paper the author analyses his earlier variant of λ-calculus which uses five elementary β-rules without the substitution operation instead of the equivalent standard β-rule. Though lists and list-manipulating functions can be "encoded" (by various means) in pure λ-calculus, their direct representation is more efficient in practice because of the complexity of the computations. The syntax of λ-calculus is here extended by explicit lists as primitive notions, and two γ-rules of reduction are added to the β-rules. The first of these γ-rules sets up the application of the λ-operator to a list and the second describes a componentwise application of a list to a λ-term—a "unique applicative property" which is closely related to the combining form of construction in the programming system FP of J. W. Backus. This property does not exist in any other functional language and is not true of the usual encoding of list structures in the standard type-free λ-calculus. This axiomatic

system is a vectorization of λ-calculus. At the same time it is quite simple and more general than the polymorphic typing of lists in the Miranda language.

The main result of the paper under review is a detailed proof of the Church-Rosser theorem for the system described above (i.e., for $\beta\gamma$-reduction) from which its consistency follows. This theorem is proved by showing that reductions induced by the union of the notion of β-reduction with both notions of γ-reduction satisfy the diamond property. To show this, the fundamental lemma of Hindley-Rosen and a lemma of the author about three binary relations commuting are used.

Finally, the author describes the further extension of λ-calculus by arithmetic, relational, Boolean and list operators, the predicates ZERO, NULL and the combinators TRUE, FALSE, Y. For each of these constants, the characteristic δ-reduction rule (in Church terminology) is postulated. The resulting computation-oriented extension of the λ-calculus is a very powerful high-order functional language in which it is easy to write computational and artificial intelligence programs. On the other hand, these rules of δ-reduction represent an operational semantics for many other functional languages. The author formulates, without proof, the proposition that the notion of $\beta\gamma\delta$-reduction satisfies the Church-Rosser property.

Unfortunately, the paper does not contain an analysis of the author's very interesting notion of controlled reduction. This reduction was introduced in the author's previous works and uses two versions of the fixed point combinator, Y and Y′, to obtain the solutions of recursive equations for functions in normal form, which is essential for infinite lists. It is interesting to investigate various minimal complete sets of basic list operators and atomic combinators and especially their relationship. This is important because many problems connected with variables and substitution in the practical implementation of λ-calculus may be avoided when the equivalent theory of Schönfinkel-Curry-Turner combinators is used. Such programming languages as SASL, KRC, HOPE, and FQL have extensively employed this idea. However, we have not come across strict formulations and a mathematical analysis of axiomatic systems with lists based on the theory of combinators. Therefore, the author's results for his variant of λ-calculus (for which he has earlier developed an interpreter program and its parallel implementation) are very timely and interesting for both mathematicians and computer science specialists. Reviewed by A. S. Kuzichev and V. P. Shumikhin.

Sieber, K. (1992). Reasoning about sequential functions via logical relations. In *Applications of categories in computer science (Durham, 1991)*, pages 258–269. Cambridge Univ. Press, Cambridge.

[MR94d:68070] Introduction: "In his seminal paper G. D. Plotkin [Theoret. Comput. Sci. 5 (1977/78), no. 3, 223–255; MR 81e:68016] introduced the functional language PCF ('programming language for computable functions') together with the standard denotational model of cpo's and continuous functions. He proved that this model is computationally adequate for PCF, but not fully abstract.

"In order to obtain full abstraction he extended PCF by a parallel conditional operator. The problem with this operator is that it changes the nature of the language. All computations in the original language PCF can be executed sequentially, but the new operator requires expressions to be evaluated in parallel.

"Here we address the problem of full abstraction for the original sequential language PCF, i.e. instead of extending the language we try to improve the model. Our approach is to cut down the standard model with the aid of certain logical relations, which we call sequentiality relations. We give a semantic characterization of these relations and illustrate how they can be used to reason about sequential (i.e. PCF-definable) functions. Finally we prove that this style of reasoning is 'complete' for proving observational congruences between closed PCF-expressions of order ≤ 3. Technically, this completeness can be expressed as a full abstraction result for the sublanguage which consists of these expressions." For the entire collection see MR 93d:68001.

Talpin, J.-P. and Jouvelot, P. (1992). Polymorphic type, region and effect inference. *J. Funct. Programming*, 2(3):245–271.
[MR94b:68084] Summary: "We present a new static system which reconstructs the types, regions and effects of expressions in an implicitly typed functional language that supports imperative operations on reference values. Just as types structurally abstract collections of concrete values, regions represent sets of possibly aliased reference values and effects represent approximations of the imperative behaviour on regions.

"We introduce a static semantics for inferring types, regions and effects, and prove that it is consistent with respect to the dynamic semantics of the language. We present a reconstruction algorithm that computes the types and effects of expressions, and assigns regions to reference values. We prove the correctness of the reconstruction algorithm with respect to the static semantics. Finally, we discuss potential applications of our system to automatic stack allocation and parallel code generation."

Abramsky, S. and Ong, C.-H. L. (1993). Full abstraction in the lazy lambda calculus. *Inform. and Comput.*, 105(2):159–267.
[MR94h:68124]

Lazy call-by-name λ-calculus, which is called λl in the paper, is the head-reduction strategy on pure closed λ-terms, with the proviso that all reductions terminate upon reaching abstractions. It is the basis of many functional languages, and the present paper is devoted to its study, in particular to questions of full-abstraction (fully abstract models of a reduction strategy are denotational models where two λ-terms have the same interpretation iff they have the same operational behaviour).

A theory of lazy λ-models is developed, under a number of more or less equivalent facets which correspond, roughly speaking, to the usual formalisms for λ-models enriched with a well-behaved convergence predicate (where terms whose λl-reduction terminates have to be interpreted).

The canonical model of lazy λ-calculus is the initial solution of the domain equation $D \simeq [D \to D]_\perp$ which expresses that D is isomorphic to the lifted space of its continuous functions. D is presented and studied by means of the usual limit construction. Then the "domain logic" associated to D is introduced; this corresponds to the view of D as a "filter model", or as one of the particularly clear cases of the general "Stone duality" theory developed by Abramsky in previous work.

Now D is also a model for λl_c, λl_p and λl_ω, which are extensions of λl obtained by adding respectively the constants c (convergence testing), p (parallel convergence testing), or $\pi_n/n \in \omega$ (projections), with the usual reduction rules that make p work as an (untyped) "parallel or" and make the π_n be the syntactic counterpart of the projections of D on its finite retracts (so that λl_ω is a labelled version of λl).

The main results of the paper are that (1) D is fully abstract for λl_p but not for λl_c and λl_ω, (2) one can construct retracts of D which are fully abstract with respect to λl_c or λl_ω, (3) the full abstraction problem for λl reduces to a conservativity result of λl_ω over λl. The first set of results and proofs is in natural affiliation with those in G. D. Plotkin's paper [Theoret. Comput. Sci. 5 (1977/78), no. 3, 223–255; MR 81e:68016], which, however, concerned the typed language PCF, and of Wadsworth's work relating termination of the (complete) head reduction strategy on pure λ-terms and their denotation in Scott's model D_∞, which is a minimal solution of $D \simeq [D \to D]$.

The paper is self-contained, proofs are complete and detailed, and one finds here some nice expository parts at a general level; but the length of the paper is mainly due to the fact that the authors enjoy presenting all abstract notions under a number of equivalent formalisms and variations, and use them in an interactive way; this does not help the reader to have a clear insight of what is really happening at the essential (mathematical) level. However, if the reader wishes to become familiar with all aspects of denotational semantics, (s)he will appreciate having at hand this fully complete presentation (of the lazy versions).

The semantic study of the lazy call-by-value λ-calculus, for which the canonical model is the initial solution of $D \simeq [D \to_\perp D]_\perp$ (where only strict continuous functions are considered), was done by L. Egidi, F. Honsell and S. Ronchi Della Rocca [Fund. Inform. 16 (1992), no. 2, 149–169; MR 93i:68036], and also by G. Boudol and R. Pino-Perez in papers which are cited there. Reviewed by Chantal Berline.

Castagna, G. (1993). A meta-language for typed object-oriented languages. In *Foundations of software technology and theoretical computer science (Bombay, 1993)*, pages 52–71. Springer, Berlin.
[MR95f:03023] Summary: "In an earlier paper we defined the $\lambda\&$-calculus, a simple extension of the typed λ-calculus to model typed object-oriented languages. To develop a formal study of type systems for object-oriented languages we define, in this paper, a metalanguage based on $\lambda\&$ and we show by a practical example how to use it to prove properties of a language. For this purpose we define a toy object-oriented language and its type-checking algorithm; then we translate this toy language into our metalanguage. The translation gives the semantics of the toy language, and a theorem on the translation of well-typed programs proves the correctness of the type-checker of the toy language.

"As an aside we also illustrate the expressivity of the $\lambda\&$-based model by showing how to translate existing features such as multiple inheritance and multiple dispatch, but also by integrating in the toy language new features directly suggested by the model, such as first-class messages, a generalization of the use of super and the use of explicit coercions." For the entire collection see MR 95d:68003.

Curien, P.-L. (1993). *Categorical combinators, sequential algorithms, and functional programming.* Birkhäuser Boston Inc., Boston, MA, second edition.
[MR94i:68171] This book is the second and revised edition of a monograph which appeared under the same title in 1986 [Pitman, London, 1986; MR 87k:68002]. Besides improvements of typography, English and presentation, the main novelty is the integration of some new material. For example, the following topics should be mentioned. The account of categorical combinatory logic has been augmented by

a description of the confluence properties of rewriting systems of categorical combinators, and of the newly developed calculi of explicit substitutions. A new full abstraction result concerning PCF is included. The model of sequential algorithms is fully abstract with respect to an extension of PCF with a control operator. There are a prologue and a picture to help convey the conceptual similarity between the operational semantics of sequential algorithms and the geometry of interaction, developed for linear logic. Reviewed by Heinrich Seidel.

Dowek, G. (1993). The undecidability of typability in the lambda-pi-calculus. In *Typed lambda calculi and applications (Utrecht, 1993)*, pages 139–145. Springer, Berlin.
[MR94c:03021] Summary: "The set of pure terms which are typable in the $\lambda\Pi$-calculus in a given context is not recursive. So there is no general type inference algorithm for the programming language Elf and, in some cases, some type information has to be mentioned by the programmer." For the entire collection see MR 94b:68025.

Dybkjær, H. and Melton, A. (1993). Comparing Hagino's categorical programming language and typed lambda-calculi. *Theoret. Comput. Sci.*, 111(1-2):145–189. 6th Workshop on the Mathematical Foundations of Programming Semantics (Kingston, ON, 1990).
[MR94f:68122] Summary: "T. Hagino ["A categorical programming language", Ph.D. Thesis, Univ. Edinburgh, Edinburgh, 1987; per bibl.] developed CPL, a categorical programming language based on dialgebras, which include algebras, coalgebras, products, sums and exponentials. We give an introduction to dialgebras and CPL. Working from the well-known correspondence between Cartesian closed categories (CCCs) and λ-calculi [J. Lambek, in To H. B. Curry, 375–402, Academic Press, London, 1980; MR 81m:03020; P.-L. Curien, Categorical combinators, sequential algorithms and functional programming, Pitman, London, 1986; MR 87k:68002], we study the relationship between CPL and F_1, Church's simply typed λ-calculus. We show that the reduction rules of CPL correspond to β-reduction in first-order contexts. Iteration over inductive types may be added to F_1, giving F_1^i [B. Pierce, S. Dietzen and S. Michaylov, "Programming in higher-order typed lambda-calculi", Tech. Rep. CMU-CS-89-111, Carnegie-Mellon Univ., Pittsburgh, PA, 1989; per bibl.]. We show how to represent F_1^i inductive types in CPL. Thus Ackermann's function is in CPL. We argue that all natural number functions $f\colon N \to N$ that are provably total in first-order arithmetic can be expressed in CPL." For the entire collection see MR 93k:68006.

Hannan, J. (1993). Extended natural semantics. *J. Funct. Programming*, 3(2):123–152.
[MR94h:68132] Summary: "We extend the definition of natural semantics to include simply typed λ-terms, instead of first-order terms, for representing programs, and to include inference rules for the introduction and discharge of hypotheses and eigenvariables. This extension, which we call extended natural semantics, affords a higher-level notion of abstract syntax for representing programs and suitable mech-

anisms for manipulating this syntax. We present several examples of semantic specifications for a simple functional programming language and demonstrate how we achieve simple and elegant manipulations of bound variables in functional programs. All the examples have been implemented and tested in λ-Prolog, a higher-order logic programming language that supports all of the features of extended natural semantics."

Hillebrand, G. G., Kanellakis, P. C., and Mairson, H. G. (1993). Database query languages embedded in the typed lambda calculus. In *Eighth Annual IEEE Symposium on Logic in Computer Science (Montreal, PQ, 1993)*, pages 332–343. IEEE Comput. Soc. Press, Los Alamitos, CA.

[**MR1238295**] This item will not be reviewed individually. For details of the collection in which this item appears see 94d:68005.

Honsell, F., Mason, I. A., Smith, S., and Talcott, C. (1993). A theory of classes for a functional language with effects. In *Computer science logic (San Miniato, 1992)*, pages 309–326. Springer, Berlin.

[**MR95c:03035**] This paper presents the logic VTLoE (variable typed logic of effects), designed for specifying and reasoning about programs with side effects. The terms of VTLoE are the terms of the λ-calculus, extended by reference primitives to create, update, and retrieve memory cells. The semantics of terms is given by a reduction relation. The atomic formulas of the language assert the operational equivalence of two expressions. In addition to the usual first-order constructions, contextual assertions are introduced in the VTLoE: if Φ is a formula and U is a certain type of context with a hole in it, then $U[\Phi]$ is a formula. This formula expresses the fact that whenever the point in the program text U denoted by the hole is reached, the assertion Φ is true. Precise definitions of the VTLoE semantics are given, and several useful theorems providing basic principles for reasoning about contextual assertions are proved.

The language VTLoE is then extended to include a general theory of classification. Class terms are either class variables, class constants, or comprehension terms $\{x \mid \Phi\}$. The class membership relation and quantification over classes are allowed in formulas.

The resulting language can express a wide variety of useful notions, including the equality and sharing of references (aliasing), side effects, functionality, and state-based reasoning. VTLoE thus generalizes traditional programming logics, going well beyond Hoare's logic and dynamic logics. Possible extensions of VTLoE, by typing judgments and by a structural induction principle, are sketched. For the entire collection see MR 94f:68011. Reviewed by Sergey G. Vorobyov.

Jung, A. and Stoughton, A. (1993). Studying the fully abstract model of PCF within its continuous function model. In *Typed lambda calculi and applications (Utrecht, 1993)*, pages 230–244. Springer, Berlin.

[**MR94c:03022**] Summary: "We give a concrete presentation of the inequationally fully abstract model of PCF as a continuous projection of the inductively reachable subalgebra of PCF's continuous function model." For the entire collection see MR 94b:68025.

Minamide, Y. (1993). Linear types and calculi with explicit sharing. *Sūrikaisekikenkyūsho Kōkyūroku*, (851):88–101. Type theory and its applications to computer systems (Japanese) (Kyoto, 1993).

[MR95g:68072] Summary: "We study the relation between Wadler's steadfast linear type system and calculi with explicit sharing. The calculi studied in this paper can be considered as an operational semantics of call-by-value functional programming languages. The soundness of the type system is proved and ensures that destructive operations in a well-typed program are performed safely." For the entire collection see MR 95f:68011.

Pierce, B. C., Rémy, D., and Turner, D. N. (1993). A typed higher-order programming language based on the pi-calculus. *Sūrikaisekikenkyūsho Kōkyūroku*, (851):46–60. Type theory and its applications to computer systems (Japanese) (Kyoto, 1993).

[MR1304524] This item will not be reviewed individually. For details of the collection in which this item appears see 95f:68011.

Qian, Z. (1993). Linear unification of higher-order patterns. In *TAPSOFT '93: theory and practice of software development (Orsay, 1993)*, pages 391–405. Springer, Berlin.

[MR94i:68174] Summary: "Higher-order patterns are simply typed λ-terms in η-long form in which the free variables F occur only in the form $F(x_1, \cdots, x_k)$ with $x_1, \cdots, x_k$ being distinct bound variables. It has been proved [D. A. Miller, J. Logic Comput. 1 (1991), no. 4, 497–536; MR 93b:68011] that in the simply typed λ-calculus unification of higher-order patterns modulo α, β and η reductions is decidable and unifiable higher-order patterns have a most general unifier.

"In this paper a unification algorithm for higher-order patterns is presented, whose time and space complexities are proved to be linear in the size of input." For the entire collection see MR 94c:68008.

Scott, D. S. (1993). A type-theoretical alternative to ISWIM, CUCH, OWHY. *Theoret. Comput. Sci.*, 121(1-2):411–440. A collection of contributions in honour of Corrado Böhm on the occasion of his 70th birthday.

[MR95d:03014]

In a book by C. A. Gunter [Semantics of programming languages, MIT Press, Cambridge, MA, 1992; MR 94f:68125] this paper was called the most well-known (and, we may add, the most interesting) unpublished manuscript in programming language theory. The main part of the text of this paper was written in 1969, but it was published only in 1993 with a new preface and many comments.

The aim of the paper is the elaboration of the mathematical semantics for programming languages. To do this, an axiomatic system for the hereditary monotone and continuous functionals is constructed in such a way that the functionals are generated from the integers and the Booleans (plus "undefined" elements). This approach is essentially based on the theory of types. According to the author, no matter how much thinking we do, the theory of types is here to stay. Even for such systems as Zermelo-Fraenkel set theory, the conceptions behind them are based on type theory. These are the concept of an arbitrary subset of a given domain and the idea that the collection of all subsets of a given domain can form a new domain (of the next

type). Starting with a domain of individuals (possibly empty), this process of forming subsets may then be iterated into the transfinite. Thus, each set has a type (or rank) given by an ordinal number. In this way, sets and their elements are named by the types and we come to the concept of a named set [M. S. Burgin, "Theory of named sets as a foundational basis of mathematics", in Structures in mathematical theories (San Sebastian, 1990), Univ. Pais Vasco-Euskal Herriko Unib, Bilbao, 1990; per revr.]. Named sets appear naturally in theories where it is necessary to be able to sort data into different types (categories) and to have a sufficient supply of derived types. The theory of domains satisfies the first condition, and the theory of types provides useful possibilities for discovering the proof principles for recursively defined domains. That is why the type-theoretical approach has been taken over and absorbed into the applications of category theory to semantics and computer theory. But categories are the systems generated from specific named sets [M. S. Burgin, Abstracts Amer. Math. Soc. 9 (1988), no. 1, 217, Abstract 88T-18-23]. All this demonstrates that in problems of semantics and computer theory, as well as in many other fields, named sets are quite unavoidable. The theory of types utilized by the author elaborates useful tools for hierarchical derivation of names of the objects considered, making a distinction between logical types and data types. As a consequence, the system in the paper is formulated as a typed system of combinators (or as a typed λ-calculus) with a recursion operator (the least fixed-point operator), and its proof rules are contrasted to a certain extent with those of the untyped λ-calculus. Problems of completeness of this system are also studied and their connections with computer theory are discussed. For the entire collection see MR 94e:68005. Reviewed by M. S. Burgin.

Sekiguchi, T. and Yonezawa, A. (1993). A complete type inference system for subtyped recursive types. *Sūrikaisekikenkyūsho Kōkyūroku*, (851):21–33. Type theory and its applications to computer systems (Japanese) (Kyoto, 1993).

[**MR95k:68130**] Summary: "Since record polymorphism is one of the essential factors for object-oriented languages, various approaches to incorporate record polymorphism into type systems have been made to lay the foundations for object-oriented languages. Recursive types, which are essentially types of lists or trees, are major programming tools. In object-oriented languages, a pseudo variable 'self' has a recursive type, which requires that type systems be able to treat recursive types. The purpose of this paper is to provide a type system and its inference system which can handle both subtype polymorphism and recursive types without any kind of type declaration or unnatural restrictions. Our system integrates subtyping and parametric polymorphism into Damas and Milner's type system and preserves several properties such as existence of principle types and syntactic completeness of the type checking algorithm. It can also be considered as an added let construct [see L. Cardelli, Inform. and Comput. 76 (1988), no. 2-3, 138–164; MR 89k:68093]. We give the type inference algorithm and prove its correctness. The basic idea is that we consider a type as a regular tree. Though our target language in this paper is a functional lan-

guage, we show the way to extend it to imperative languages, too." For the entire collection see MR 95f:68011.

Sreedhar, V. and Taghva, K. (1993). Capturing strong reduction in director string calculus. *Theoret. Comput. Sci.*, 107(2):333–347.
[MR94c:68029] Turner (1979) first showed that the basic combinators can be used to execute functional programs (or lambda expressions, theoretically). A director string is a sequence of the basic combinators S, B and C, and the director string calculus of Kennaway and Sleep (1988) is an elegant formulation of Turner's method. A limitation of this calculus is its weak reduction property, i.e., an inner subexpression cannot be reduced. The modified director string calculus (MDSC) of this paper remedies this: A director string is attached to each of the inner subexpressions so that now the calculus has the strong reduction property. The calculus MDSC is confluent. Translations between MDSC and the usual lambda calculus are given, showing that they are equivalent in their applicative structure.

Reviewed by Teruo Hikita

Takahashi, M. (1993). λ-calculi with conditional rules. In *Typed lambda calculi and applications (Utrecht, 1993)*, pages 406–417. Springer, Berlin.
[MR94c:03025] Summary: "A variety of typed/untyped λ-calculi and related reduction systems have been proposed in order to study various aspects of programs, some of which contain rules subject to side conditions. As a framework to study fundamental properties of such reduction systems, we first introduce the notion of conditional λ-calculus. Then we give a sufficient condition for them to be confluent (Church-Rosser) as well as to have a normalizing strategy a la Gross. The proof, being a generalization of the proof by Tait and Martin-Löf for the confluence of $\lambda\beta$, is inductive and simple." For the entire collection see MR 94b:68025.

Abadi, M. (1994). A semantics for static type inference in a nondeterministic language. *Inform. and Comput.*, 109(1-2):300–306. International Conference on Theoretical Aspects of Computer Software (TACS '91) (Sendai, 1991).
[MR95f:68137] The article under review is published as a short note immediately following a paper by G. D. Plotkin [Inform. and Comput. 109 (1994), no. 1-2, 256–299] in the same volume. It uses Plotkin's models of reduction—where reduction $M \twoheadrightarrow N$ is modelled as an inequality $[\![M]\!] \leq [\![N]\!]$—in a nondeterministic language; the choice operation $+$ with reductions $M + N \to M$ and $M + N \to N$ is modelled as an operation $\wedge$ in a poset, which satisfies $a \wedge b \leq a$ and $a \wedge b \leq b$. If one also has a reduction $M \to M + M$, then $\wedge$ must be a meet. Soundness and completeness results are obtained, for reduction and for static type inference. For the entire collection see MR 94j:68004. Reviewed by Bart Jacobs.

Blaaberg, B. and Clausen, C. (1994). Adequacy for a lazy functional language with recursive and polymorphic types. *Theoret. Comput. Sci.*, 136(1):243–275. Selected papers of the Meeting on the Mathematical Foundations of Programming Semantics (MFPS '92), Part II (Oxford, 1992).

[**MR95j:68087**] Summary: "We consider a lazy functional language with both recursive and polymorphic types. For this language we give an operational semantics and a denotational semantics based on information systems and establish adequacy, i.e. that a term converges denotationally if and only if it converges operationally. In the presence of universal types the usual approach taken is to apply the technique of reducibility candidates. But at this point we take a different approach: The concreteness of our model allows us to define a notion of size of tokens of information systems. Based on this notion, an appropriate logical relation can be defined by good old well-founded recursion. The proof of adequacy then follows by standard techniques." For the entire collection see MR 95h:68002.

Boerio, L. (1994). Extending pruning techniques to polymorphic second order λ-calculus. In *Programming languages and systems—ESOP '94 (Edinburgh, 1994)*, pages 120–134. Springer, Berlin.

[**MR95f:03021**] Summary: "Type theory and constructive logics allow us to extract, from a proof of a formula, a program that satisfies the specification expressed by this formula. Normally, programs extracted in this way are inefficient. Optimization algorithms for these programs have been developed. In this paper we give an algorithm to optimize programs represented by second-order typed λ-terms. We also prove that the simplified programs are observationally equivalent to the original ones." For the entire collection see MR 95e:68012.

Boudol, G. (1994). Lambda-calculi for (strict) parallel functions. *Inform. and Comput.*, 108(1):51–127.

[**MR95d:03010**] The author considers two calculi λ_j^{nv} and λ_j^{v}, which are, respectively, the call-by-value and the call-by-name λ-calculus extended with a parallel operator $\|$. The intuition is that $M_1\|M_2$ "terminates" as soon as one of the M_i has produced a value. The evaluation and reduction of both calculi are defined and investigated.

The main result is full abstraction of the standard domains (indeed, lattices) $D_s = (D_s \to_\perp D_s)_\perp$ and $D_\star = (D_\star \to D_\star)_\perp$, when $\|$ is interpreted as sup (in the lattices). The proof relies on a careful analysis of the models, following Abramsky's theory of "domains in logical forms".

To interpret the two calculi, it is crucial that the domains D_s and $D_\star$ be lattices. However, this property may fail for simple variants of the domain equations, e.g., $D = A \oplus (D \to D)_\perp$, where A is a flat domain and $\oplus$ is the coalesced sum. Reviewed by Eugenio Moggi.

Broome, P. and Lipton, J. (1994). Combinatory logic programming: computing in relation calculi. In *Logic programming*, pages 269–285. MIT Press, Cambridge, MA.

[**MR95i:68016**] Summary: "We define a combinatory relational calculus based on a variant of the relation algebra formalism (developed by Tarski and Givant, and in more general form by Freyd and others) which extends traditional database query formalisms in a number of ways. It is a variable-free calculus for rewriting relation expressions built up from ground relations using union, intersection, complementation, composition and a fixed point operator. The language extends conventional

Horn-clause programming with constraints (e.g. constructive negation) by permitting queries and outputs to be relation expressions. We also define an associated relational programming language RAlog which combines rewriting in the calculus with a quantifier-elimination algorithm due to Maltsev. The algorithm is used to translate reduced relation expressions in the calculus into two-quantifier formulas, which constitute the printed output of the language." For the entire collection see MR 95f:68005.

Curien, P.-L. and Ghelli, G. (1994). Decidability and confluence of $\beta\eta$ $\mathrm{Top}_{\leq}$ reduction in $F_{\leq}$. *Inform. and Comput.*, 109(1-2):57–114. International Conference on Theoretical Aspects of Computer Software (TACS '91) (Sendai, 1991).

[**MR96g:68068**] Summary: "We contribute to the syntactic study of $F_{\leq}$, a variant of second-order λ-calculus F which appears as a paradigmatic kernel language for polymorphism and subtyping. The type system of $F_{\leq}$ has a maximum type Top and bounded quantification. We endow this language with the β-rules (for terms and types), to which we add η-rules (for terms and types) and a rule which equates all terms of type Top. These rules are suggested by the axiomatization of Cartesian closed categories. We exhibit a weakly normalizing and confluent reduction system for this theory $\beta\eta$ $\mathrm{top}_{\leq}$, and show that it is decidable. It is also confluent, but decidability does not follow from confluence, since reduction is not effective. Our proofs rely on the confluence and decidability of a corresponding system on F_1 (the extension of F with a terminal type)."

One may recall that $F_{\leq}$ is essentially the language FUN of Cardelli and Wegner, and the kernel of Cardelli's language Quest. One can note also that an interesting variant of $F_{\leq}$ is studied in a paper by L. Cardelli et al. [Inform. and Comput. 109 (1994), no. 1-2, 4–56; MR 96d:68135]. For the entire collection see MR 94j:68004. Reviewed by Chantal Berline.

Fisher, K., Honsell, F., and Mitchell, J. C. (1994). A lambda calculus of objects and method specialization. *Nordic J. Comput.*, 1(1):3–37.

[**MR95m:68109**] Summary: "This paper presents an untyped lambda calculus, extended with object primitives that reflect the capabilities of so-called delegation-based object-oriented languages. A type inference system allows static detection of errors, such as 'message not understood', while at the same time allowing the type of an inherited method to be specialized to the type of the inheriting object. Type soundness is proved using operational semantics and examples illustrating the expressiveness of the pure calculus are presented."

Gunter, C. A. (1994). The semantics of types in programming languages. In *Handbook of logic in computer science, Vol. 3*, pages 395–475. Oxford Univ. Press, New York.

[**MR96j:68111**] The main aim of this handbook chapter is to describe "the use of types as a conceptual tool for classifying programs in a way that permits a more abstract understanding of their meanings".

Of course, one cannot expect full coverage of such a subject in a handbook chapter; nevertheless the author presents most of the main ideas and tools, and gives many pointers to the relevant literature. The 80 pages of the chapter are very readable, and probably even a newcomer can go through it in one try.

On the negative side, one should point out at least two omissions: the use of types for classifying computational aspects (e.g., monads and effect systems), and the use of types in binding-time analysis (e.g., 2-level lambda calculus). For the entire collection see MR 96f:68008. Reviewed by Eugenio Moggi.

Hall, C., Hammond, K., Peyton Jones, S., and Wadler, P. (1994). Type classes in Haskell. In *Programming languages and systems—ESOP '94 (Edinburgh, 1994)*, pages 241–256. Springer, Berlin.

[MR95f:68141] Summary: "This paper defines a set of type inference rules for resolving overloading introduced by type classes. Programs including type classes are transformed into ones which may be typed by the Hindley-Milner inference rules. In contrast to other work on type classes, the rules presented here relate directly to user programs. An innovative aspect of this work is the use of second-order lambda calculus to record type information in the program." For the entire collection see MR 95e:68012.

Hankin, C. (1994). *Lambda calculi.* The Clarendon Press Oxford University Press, New York. A guide for computer scientists, Oxford Science Publications.

[MR96j:68001] The book is an introduction to lambda calculus and related formalisms from a computer science perspective. It is intended not as a systematic treatise, but rather as a brief presentation, within the framework of lambda calculus, of a variety of notions which are becoming increasingly important in current research on programming language design, semantics and implementation. It can be profitably used as a textbook for final year undergraduate or first year graduate students in computer science. The first part of the book (Chapters 1–6) covers the basic notions of untyped lambda calculus and combinatory logic together with the relationships to computability theory. The second part (Chapters 7–10) links this classical material with more recent topics such as Milner-style polymorphism, second-order lambda calculus, intersection types, reduction machines, needed reductions, strictness analysis, lazy and concurrent calculi, calculi with explicit substitutions. Due to the size of the book the treatment is necessarily brief. The author succeeds in whetting the reader's appetite for more specialized readings, a guide to which is given in the last chapter. A distinctive feature of the book is the emphasis on the link between theoretical results and practical issues on the implementation of (functional) programming languages. Reviewed by Alessandro Berarducci.

Henglein, F. (1994). Dynamic typing: syntax and proof theory. *Sci. Comput. Programming*, 22(3):197–230. Selected papers of the Fourth European Symposium on Programming (Rennes, 1992).
[MR96b:68109]

Summary: "We present the dynamically typed λ-calculus, an extension of the statically typed λ-calculus with a special type Dyn and explicit dynamic type coercions corresponding to run-time type tagging and type check-and-untag operations. Programs in run-time typed languages can be interpreted in the dynamically typed λ-calculus via a nondeterministic completion process that inserts explicit coercions and type declarations such that a well-typed term results.

"We characterize when two different completions of the same run-time typed program are coherent with an equational theory that is independent of an underlying λ-theory. This theory is refined by orienting some equations to define safety and minimality of completions. Intuitively, a safe completion is one that does not produce an error at run-time which another completion would have avoided, and a minimal completion is a safe completion that executes the fewest tagging and check-and-untag operations amongst all safe completions.

"We show that every untyped λ-term has a safe completion at any type and that it is unique modulo a suitable congruence relation. Furthermore, we present a rewriting system for generating minimal completions. Assuming strong normalization of this rewriting system we show that every λI-term has a minimal completion at any type, which is furthermore unique modulo equality in the dynamically typed λ-calculus." For the entire collection see MR 95b:68020.

Krivine, J.-L. (1994a). Classical logic, storage operators and second-order lambda-calculus. *Ann. Pure Appl. Logic*, 68(1):53–78.
[MR95i:03029] The author discusses the connections between classical logic and storage operators in the context of the second-order lambda-calculus. In the lambda-calculus, a term T is a storage operator for the set of terms A if for all $t \in A$ there exists an α such that for all θ with $\theta =_\beta t$, $Tf\theta \to_h^* f\alpha'$, where α' is some substitution instance of α and f is a variable. Here, $\to_h^*$ denotes the multiple-step head-reduction. The interest of storage operators lies in the fact that they can be used to simulate call-by-value evaluation by call-by-name evaluation (head-reduction). Moreover, storage operators can be typed in the second-order lambda-calculus. For example, if T is a closed term of type $\forall x(\mathrm{Int}[x] \to \mathrm{Int}^*[x])$, then T is a storage operator for the Church numerals. (Here, $\mathrm{Int}[x]$ is the second-order formula that states that x is an inductive natural number, namely, $\mathrm{Int}[x] = \forall X(\forall y(Xy \to X(sy)) \wedge X0 \to Xx)$.) The operation * is defined as the Gödel translation on formulas, which maps classically provable formulas to intuitionistically provable formulas.

The interest of this article lies in the extension of the result on storage operators to the classical second-order lambda-calculus (in which there is a special constant $c\colon \forall X(X \to X))$. The main result is that, if a term T is of type $\forall x(\mathrm{Int}[x] \to \mathrm{Int}^*[x])$, then it is also a storage operator for a term θ of type $\mathrm{Int}[s^n 0]$ in the classical second-order lambda-calculus. This theorem is proved in detail, using semantic methods that were developed by the author and M. Parigot in previous articles. The author also gives motivations for the claim that the second-order lambda-calculus with head reduction (call-by-name) is a model for imperative programming languages. Reviewed by Herman Geuvers.

Krivine, J.-L. (1994b). Fonctions, programmes et démonstration. *Gaz. Math.*, (60):63–73.
[MR95e:03095] In a popular exposition containing no details, the author hints at some fruitful parallelisms between programs and formal proofs, and sketches some relations with Church's theory of λ-conversion. Reviewed by E. Mendelson.

Mück, A., Streicher, T., and Lock, H. C. R. (1994). A tiny constraint functional logic language and its continuation semantics. In *Programming languages and systems—ESOP '94 (Edinburgh, 1994)*, pages 439–453.

Springer, Berlin.

[**MR95g:03014**] Summary: "We present an extension of λ-calculus by logical features and constraints, which yields a minimal core language for constraint functional logic programming. We define a denotational semantics based on continuation passing style. The operational semantics of our language is given as a set of reduction rules. We prove soundness of the operational semantics with respect to the continuation semantics. Finally, we show how pure functional logic programs can be translated to this core language in a sound way." For the entire collection see MR 95e:68012.

Nishizaki, S.-y. (1994a). ML with first-class environments and its type inference algorithm. In *Logic, language and computation*, pages 95–116. Springer, Berlin.

[**MR96a:03029**] Summary: "We present a typed λ-calculus which enables us to handle first-class environments. The syntax and the reduction are obtained by applying the idea of Curien's 'explicit substitution'. The type system has ML-polymorphism and a type inference algorithm which is sound and terminates." For the entire collection see MR 95b:68015.

Nishizaki, S.-y. (1994b). Simply typed lambda calculus with first-class environments. *Publ. Res. Inst. Math. Sci.*, 30(6):1055–1121.

[**MR96h:68120**] Summary: "We propose a lambda calculus $\lambda_{\mathrm{env}}^{\rightarrow}$ in which it is possible to handle first-class environments. This calculus is based on the idea of explicit substitution, that is, on $\lambda\sigma$-calculus. The syntax of $\lambda_{\mathrm{env}}^{\rightarrow}$ is obtained by merging the class of terms and that of substitutions. Reduction is made from the weak reduction of $\lambda\sigma$-calculus. Its type system also originates in the one of $\lambda\sigma$-calculus. Confluence of $\lambda_{\mathrm{env}}^{\rightarrow}$ is proved by Hardin's interpretation method which is originally used for proving confluence of $\lambda\sigma$-calculus. We prove strong normalizability of $\lambda_{\mathrm{env}}^{\rightarrow}$ by reducing it to strong normalizability of a simply typed record calculus. Finally, we propose a type inference algorithm which produces a principal typing for each typable term."

Nour, K. (1994). Preuve syntaxique d'un théorème de J.-L. Krivine sur les opérateurs de mise en mémoire. *C. R. Acad. Sci. Paris Sér. I Math.*, 318(3):201–204.

[**MR94j:03036**] Summary: "Storage operators are closed λ-terms which, for the integer type, allow one to simulate call by value while using call by name. J.-L. Krivine showed, using a semantic method, that such operators can be characterized by a type of the $\mathcal{AF}\in$ type system. In this note we give a syntactic proof of this theorem. The proof is based on the properties of the $\mathcal{AF}\in$ type system and the notion of directed lambda-calculus. By using the same techniques, we can generalize this result for the class of types where the universal second-order quantifier appears positively."

Poirriez, V. (1994). MLOG: a strongly typed confluent functional language with logical variables. *Theoret. Comput. Sci.*, 122(1-2):201–223. Fifth

Generation Computer Systems Conference (Tokyo, 1992).
[**MR95c:68037**] Summary: "A new programming language called MLOG is introduced. MLOG is a conservative extension of ML with logical variables. To validate our concepts, a compiler named CAML light FLUO was implemented. Numerous examples are presented to illustrate the possibilities of MLOG. The pattern matching of ML is kept for λ-calculus bindings and a unification primitive is introduced for the bindings of logical variables. A suspension mechanism allows cohabitation of pattern-matching and logical variables. Although the evaluation strategy for the application is fixed, the order for the evaluation of the parts of pairs and application remains free. MLOG programs can be evaluated in parallel with the same result obtained irrespective of the particular order of evaluation. This is guaranteed by the Church-Rosser property possessed by the evaluation rules. As a corollary, a strict λ-calculus with explicit substitutions on named variables is shown to be confluent. A completely formal operational semantics of MLOG is given in this paper." For the entire collection see MR 94f:68013.

Reddy, U. S. (1994). Higher-order aspects of logic programming. In *Extensions of logic programming (St. Andrews, 1993)*, pages 301–321. Springer, Berlin.
[**MR95f:68042**] Summary: "Are higher-order extensions to logic programming needed? We suggest a negative answer by showing that higher-order features are already available in pure logic programming. It is demonstrated that higher-order lambda calculus-based languages can be compositionally embedded in logic programming languages, preserving their semantics and abstraction facilities. Further, we show that such higher-order techniques correspond to programming techniques often employed in logic programming." For the entire collection see MR 95c:68007.

Streicher, T. (1994). A universality theorem for PCF with recursive types, parallel-or and $\exists$. *Math. Structures Comput. Sci.*, 4(1):111–115.
[**MR94m:68121**] Summary: "In a PCF-like call-by-name typed λ-calculus with a minimal fixpoint operator, 'parallel-or', Plotkin's 'continuous existential quantifier' $\exists$ and recursive types together with constructors and destructors, all computable objects can be denoted by terms of the programming language. According to A. Meyer's terminology, such a programming language is called universal in the sense that any extension of it must be conservative, as all computable objects can already be expressed by program terms.

"As a byproduct, we obtain that, in principle, recursive types could be totally avoided, as they appear as syntactically expressible retracts of the nonrecursive type $\mathcal{N} \to \mathcal{B}$, where $\mathcal{N}$ and $\mathcal{B}$ are the flat domains of natural numbers and Boolean values, respectively."

Wolfram, D. A. (1994). A semantics for λProlog. *Theoret. Comput. Sci.*, 136(1):277–289. Selected papers of the Meeting on the Mathematical Foundations of Programming Semantics (MFPS '92), Part II (Oxford, 1992).
[**MR95j:68101**] Summary: "We give a model-theoretic semantics for the logic of

higher-order Horn clauses, the basis of a form of the λProlog higher-order logic programming language. We define certain intensional general models and show that higher-order Horn clause logic is sound and complete with respect to them." For the entire collection see MR 95h:68002.

Aït-Kaci, H. and Garrigue, J. (1995). Label-selective λ-calculus syntax and confluence. *Theoret. Comput. Sci.*, 151(2):353–383. Thirteenth Conference on Foundations of Software Technology and Theoretical Computer Science (Bombay, 1993).
[MR97f:03018] Summary: "We introduce an extension of λ-calculus, called label-selective λ-calculus, in which arguments of functions are selected by labels. The set of labels combines symbolic keywords with numeric positions. While the former enjoy free commutation, the latter and relative renumbering are needed to extend commutation to conflictuous names and allow full currying. This extension of λ-calculus is conservative, in the sense that when we restrict ourselves to using only one label, it coincides with λ-calculus. The main result of this paper is the proof that the label-selective λ-calculus is confluent. In other words, argument selection and reduction commute." For the entire collection see MR 96f:68004.

Amadio, R. M., Leth, L., and Thomsen, B. (1995). From a concurrent λ-calculus to the π-calculus. In *Fundamentals of computation theory (Dresden, 1995)*, pages 106–115. Springer, Berlin.
[MR1459170] This item will not be reviewed individually. For details of the collection in which this item appears see 98a:68005.

Berardi, S. and Boerio, L. (1995). Using subtyping in program optimization. In *Typed lambda calculi and applications (Edinburgh, 1995)*, pages 63–77. Springer, Berlin.
[MR1477975] This item will not be reviewed individually. For details of the collection in which this item appears see 98e:03020.

Boudol, G. and Laneve, C. (1995). Termination, deadlock and divergence in the λ-calculus with multiplicities. In *Mathematical foundations of programming semantics (New Orleans, LA, 1995)*, page 14 pp. (electronic). Elsevier, Amsterdam.
[MR1486843] This item will not be reviewed individually. For details of the collection in which this item appears see 98g:68006.

Castagna, G. (1995). A meta-language for typed object-oriented languages. *Theoret. Comput. Sci.*, 151(2):297–352. Thirteenth Conference on Foundations of Software Technology and Theoretical Computer Science (Bombay, 1993).
[MR97i:68125] Summary: "In [G. Castagna, G. Ghelli and G. Longo, Inform. and Comput. 117 (1995), no. 1, 115–135; MR 95m:68108] we defined the $\lambda\&$-calculus, a simple extension of the typed λ-calculus to model typed object-oriented languages. This paper is the continuation or, rather, the companion of the above-cited paper since it analyzes the practical counterpart of the theoretical issues introduced there.

Indeed, to develop a formal study of type systems for object-oriented languages we define a meta-language based on $\lambda\&$ and we show, by a practical example, how it can be used to prove properties of a language. To this purpose, we define a toy object-oriented language and its type-checking algorithm; then we translate this toy language into our meta-language. The translation gives the semantics of the toy language and a theorem on the translation of well-typed programs proves the correctness of the type-checker of the toy language.

"As an aside we also illustrate the expressivity of the $\lambda\&$-based model by showing how to translate existing features like multiple inheritance and multiple dispatch, but also by integrating into the toy language new features directly suggested by the model, such as first-class messages, a generalization of the use of super and the use of explicit coercions. An important novelty with respect to previous systems is that we show how to model multiple dispatch also in the presence of a notion of receiver (i.e. of a privileged argument to which the message is passed), a notion that is absent in languages such as CLOS." For the entire collection see MR 96f:68004.

Di Cosmo, R. (1995). *Isomorphisms of types: from λ-calculus to information retrieval and language design.* Birkhäuser Boston Inc., Boston, MA. [**MR96m:68002**]

As stated in the preface, "Isomorphisms of types is a research topic that has its roots both in the common use of types in programming languages and in branches of mathematical logic like the theory of types and the lambda-calculus. Their study started from very theoretical motivations in category theory and the theory of λ-calculus and turned out to provide a basis for both the design of new type systems for programming languages and the development of natural tools for information retrieval in software libraries."

The present monograph is based on the author's Ph.D. dissertation. "A new introduction to the topic makes it accessible to a wide spectrum of readers, from the theoretician to the practitioner interested in implementation details." The reviewer, who belongs to the first kind of readers, enjoyed the presentation of these practical motivations, as well as many enlightening remarks.

The intended practical use is the retrieval of functions (in ML-like functional language libraries) from their types, up to isomorphism (to add the necessary flexibility). The mathematical and logical work consists of providing a finite axiomatisation of the set of all type equations which are provable in a given type system and searching for decidability results; here $A = B$ is provable if there are two inverse typed terms of type $A \to B$ and $B \to A$ respectively.

All the type systems considered here are subsystems of the second-order typed calculus with surjective pairing and terminal object, which corresponds, via the Curry-Howard isomorphism, to (second-order) intuitionistic propositional logic based on $(\forall, \to, \wedge, \mathrm{true})$. This is not exactly the case of ML, which is only a type assignment system, and the author is led to the development of a parallel setting and study. Reviewed by Chantal Berline.

Garrigue, J. (1995). The transformation calculus. In *Foundations of software technology and theoretical computer science (Bangalore, 1995)*, pages 151–165. Springer, Berlin.

[**MR1458076**] This item will not be reviewed individually. For details of the collection in which this item appears see 98a:68004.

Hasegawa, M. (1995). Decomposing typed lambda calculus into a couple of categorical programming languages. In *Category theory and computer science (Cambridge, 1995)*, pages 200–219. Springer, Berlin.
[**MR1463722**] This item will not be reviewed individually. For details of the collection in which this item appears see 98c:68008.

Kamareddine, F. (1995). Important issues in foundational formalisms. *Bull. IGPL*, 3(2-3):291–317. Deduction and language.
[**MR96d:03047**] Summary: "This article discusses my work in the last few years on logical formalisms which have been shown to be useful to various aspects of natural and programming languages and for foundational formalisms. In this period, I have been involved in two extensive programs: (1) The first program concerns languages which exhibit various ways of combining expressiveness with logic. While I do not propose that any of these languages is ideal, I believe that they illustrate the fruitfulness of bringing together ideas from distinct disciplines. Central to the program will be logic, λ-calculus and type theory, which have played an important role not only in foundational discussions, but also in applied formal semantics; specifically, the semantics of natural language and of programming languages. The general goal here has been to find expressive and unifying theories which keep the earlier advantages but bring about new dimensions. This goal, moreover, extends to finding a general framework which can be used to compare earlier theories and to carry results from one theory to another without duplication of work. Issues that play a great role in the general framework include full expressiveness and logic, intensionality versus extensionality, polymorphism, internal definability of determiners and quantifiers, fixed point operators and self-application, avoidance of Russell's and Curry's paradoxes, and property and truth theories. This program comes from my ongoing work at and collaboration with Edinburgh where the stimulating environment of logic, foundation and language has been invaluable. (2) From the point of view of notation and language, I aimed at studying typed and type-free formalisms and at investigating a λ-notation which can be used to generalise the various existing type theories, and to improve the computational power of the λ-calculus by making substitution explicit, by refining reduction, by introducing definitions to the syntax and by rewriting terms so that their reduction can be made more efficient. This work comes from my ongoing collaboration with Eindhoven, the centre of AUTOMATH, the theorem prover which has inspired much research on language and formalisms. I hope that by studying general formalisms from the outside and by going inside and unpacking the notation, one can gain much further insights than have been achieved so far."
For the entire collection see MR 96b:03002.

Lester, D. and Mintchev, S. (1995). Towards machine-checked compiler correctness for higher-order pure functional languages. In *Computer science logic (Kazimierz, 1994)*, pages 369–381. Springer, Berlin.

[**MR1471240**] This item will not be reviewed individually. For details of the collection in which this item appears see 98d:68013.

Maraist, J., Odersky, M., Turner, D. N., and Wadler, P. (1995). Call-by-name, call-by-value, call-by-need, and the linear lambda calculus. In *Mathematical foundations of programming semantics (New Orleans, LA, 1995)*, page 23 pp. (electronic). Elsevier, Amsterdam.

[**MR98m:03030**] Summary: "J.-Y. Girard described two translations of intuitionistic logic into linear logic, one where $A \to B$ maps to $(!A) \multimap B$, and another where it maps to $!(A \multimap B)$. We detail the action of these translations on terms, and show that the first corresponds to a call-by-name calculus, while the second corresponds to call-by-value. We further show that if the target of the translation is taken to be an affine calculus, where ! controls contraction but weakening is allowed everywhere, then the second translation corresponds to a call-by-need calculus, as recently defined by Ariola, Felleisen, Maraist, Odersky and Wadler. Thus the different calling mechanisms can be explained in terms of logical translations, bringing them into the scope of the Curry-Howard isomorphism." For the entire collection see MR 98g:68006.

Mellies, P.-A. (1995). Typed λ-calculi with explicit substitutions may not terminate. In *Typed lambda calculi and applications (Edinburgh, 1995)*, pages 328–334. Springer, Berlin.

[**MR1477992**] This item will not be reviewed individually. For details of the collection in which this item appears see 98e:03020.

Nour, K. (1995). Strong storage operators and data types. *Arch. Math. Logic*, 34(1):65–78.

[**MR96b:03024**] Summary: "The storage operators were introduced by J.-L. Krivine [RAIRO Inform. Theor. Appl. 25 (1991), no. 1, 67–84; MR 93a:68084]; they are closed λ-terms which, for some fixed data type (the integers for example), allow the simulation of 'call by value' while using 'call by name'. Krivine showed that such operators can be typed, in the AF_2 type system, using Gödel's translation from classical to intuitionistic logic [J.-L. Krivine, Arch. Math. Logic 30 (1990), no. 4, 241–267; MR 91j:03011].

"This paper studies the existence of storage operators which give a normal form as result (strong storage operators) for recursive and iterative representation of data in λ-calculus. We obtain the following result: We can find typed strong storage operators for the recursive representations of a data type, but that is not the case for the iterative representations of an infinite data type. We give the proof of this result in the case of integers."

Odersky, M. (1995). Polarized name passing. In *Foundations of software technology and theoretical computer science (Bangalore, 1995)*, pages 324–337. Springer, Berlin.

[**MR1458086**] This item will not be reviewed individually. For details of the collection in which this item appears see 98a:68004.

Ong, C.-H. L. (1995). Correspondence between operational and denotational semantics: the full abstraction problem for PCF. In *Handbook of logic* *in computer science, Vol. 4*, pages 269–356. Oxford Univ. Press, New York.

[**MR96m:68108**] A chapter within a handbook always has the aim of giving a concise overview of the problem considered and of informing the reader about the state of the art. The article under review does this in a perfect way.

At the beginning of the introduction, the basic concepts of operational and denotational semantics are described informally. It becomes clear why both semantic styles are needed and why it is necessary to study how one is related to the other. Next, the author explains what full abstract models are, and the full abstraction problem is introduced as follows: Give an abstract, synthetic account of the unique order-extensional, continuous, inequationally fully abstract model of PCF as identified by Milner. The rest of the introduction is dedicated to the discussion of what this sentence means in detail and how the problem it describes has been tackled.

Section 2 concerns the syntactic and operational properties of PCF, especially stability and sequentiality. A reduction system, equipped with a normal order (i.e. leftmost) reduction strategy, is used to define PCF. This is closely related to a typed λ-calculus. The Church-Rosser property, standardization theorem, sequentiality, and stability are proved. An overview of some results on the full abstraction problem is given in Section 3. Observationally equivalent program fragments are defined by program contexts. The context lemma states that observationally inequivalent program parts may be distinguished by contexts. Denotational models in connection with the least fixed-point semantics are then described. Domains, especially Scott domains, are shown to be the order-extensional, continuous models of PCF. However, all these results are fully abstract only if the language PCF is extended by a parallel-or construct.

Finally, Chapter 4 deals with the full abstraction problem itself. Here, the difficulty lies in a precise definition of sequentiality of higher order functions. Some attempts, such as Kahn and Plotkin's concrete data structure, Berry and Curien's sequential algorithms and Berry's stable functions, are discussed.

Altogether, combined with the cited papers, the article can be warmly recommended to readers who wish to get a deep insight into the mentioned problems. For the entire collection see MR 96h:68006. Reviewed by Peter Bachmann.

Pravato, A., Ronchi della Rocca, S., and Roversi, L. (1995). Categorical semantics of the call-by-value λ-calculus. In *Typed lambda calculi and applications (Edinburgh, 1995)*, pages 381–396. Springer, Berlin.

[**MR98f:03014**] Summary: "The denotational semantics of the call-by-value λ-calculus in a categorical setting is given. Furthermore, a particular model based on coherence domains is studied." For the entire collection see MR 98e:03020.

Raja, N. and Shyamasundar, R. K. (1995). Combinatory formulations of concurrent languages. In *Algorithms, concurrency and knowledge (Pathumthani, 1995)*, pages 156–170. Springer, Berlin.

[**MR97d:68066**] Summary: "We design a system with six basic combinators, and prove that it is powerful enough to embed the full asynchronous π-calculus, including replication. Our theory for constructing combinatory versions of concurrent languages is based on a method, used by Quine and Bernays, for the general elimination of variables in linguistic formalisms. Our combinators are designed to eliminate the requirement of names that are bound by an input prefix. They also eliminate the need for input prefix, output prefix, and the accompanying mechanism of substitution. We define a notion of bisimulation for the combinatory version, and show that the combinatory version preserves the semantics of the original calculus. One of the distinctive features of this approach is that it can be used to rework many more process algebras in order to derive their equivalent combinatory versions." For the entire collection see MR 97b:68003.

Ritter, E. and Pitts, A. M. (1995). A fully abstract translation between a λ-calculus with reference types and standard ML. In *Typed lambda calculi and applications (Edinburgh, 1995)*, pages 397–413. Springer, Berlin.
[MR1477997] This item will not be reviewed individually. For details of the collection in which this item appears see 98e:03020.

Springintveld, J. (1995). Third-order matching in the presence of type constructors (extended abstract). In *Typed lambda calculi and applications (Edinburgh, 1995)*, pages 428–442. Springer, Berlin.
[MR1477999] This item will not be reviewed individually. For details of the collection in which this item appears see 98e:03020.

Walker, D. (1995). Objects in the π-calculus. *Inform. and Comput.*, 116(2):253–271.
[MR95k:68132] Summary: "Two semantics for a parallel object-oriented programming language are presented. One is a two-level transitional semantics in which the global behaviour of a system is derived directly from the possible actions of its constituent objects. The other is by translation into the π-calculus. A close correspondence between the two semantics is established."

Arvind, Maessen, J.-W., Nikhil, R. S., and Stoy, J. (1996). A lambda calculus with letrecs and barriers. In *Foundations of software technology and theoretical computer science (Hyderabad, 1996)*, pages 19–36. Springer, Berlin.
[MR99a:68023] From the introduction: "In this paper we describe some extensions which make the pure lambda calculus much more useful, both for discussing the semantic essence of 'real' languages, and also as an intermediate language for a compiler.

"We begin with a quick overview of the pure lambda calculus extended with constants. This is followed by a discussion of the inadequacy of this system to address some pragmatic aspects of even purely functional languages (Section 3). In Section 4 we describe λ_{let}, a lambda calculus with recursive let blocks (i.e., letrecs), and discuss why it is more appropriate as a kernel language; we also discuss the complications letrecs add. In Section 5, we discuss λ_B, which is λ_{let} extended with sequentializing constructs called barriers. We then mention some more extensions we have studied, which continue to be the target of active research." For the entire collection see MR 98i:68006.

Barbanera, F. and Berardi, S. (1996). A symmetric lambda calculus for classical program extraction. *Inform. and Comput.*, 125(2):103–117.

Theoretical aspects of computer software (Sendai, 1994).

[MR97i:68123] The paper introduces a symmetric lambda calculus where functional types are replaced by binary conjunction, its dual (i.e. binary disjunction) and an involutive operation of negation (defined by induction on the structure of types). The symmetric application $P \star Q$ has type $\perp$ provided P has type $A^{\perp}$ and Q has type A.

The main result says that the reduction is strongly normalizing (SN). However, the reduction is not confluent (CR), thus I do not see why such calculus should be of computational interest. This problem is avoided by Parigot's λ-calculus, which is both SN and CR.

However, in a paper to be presented at TACS '97 ("Theoretical aspects of computer software") the authors propose a possible computational reading of non-confluence in terms of non-determinism. For the entire collection see MR 97b:68011. Reviewed by Eugenio Moggi.

Berardi, S. (1996). Pruning simply typed λ-terms. *J. Logic Comput.*, 6(5):663–681.

[MR98g:03039] Summary: "We say that a simply typed λ-term is a 'pruning' of another one if the former is obtained from the latter by replacing some subterms with dummy constants. We prove that 'pruning' preserves observational behaviour of a simply typed λ-term if it does not modify the type or the context (assignment of types to free variables) of the term. This result is used to define a map $\mathrm{FI} : \{\text{simply typed}\lambda\text{-terms}\} \to \{\text{simply typed } \lambda\text{-terms}\}$ removing redundant code in functional programs. In the rest of the paper we prove some properties of FI that are interesting from a computational viewpoint. An algorithm to compute FI is included in the Appendix."

Bono, V., Bugliesi, M., and Liquori, L. (1996). A lambda calculus of incomplete objects. In *Mathematical foundations of computer science 1996 (Cracow)*, pages 218–229. Springer, Berlin.

[MR1478962] This item will not be reviewed individually. For details of the collection in which this item appears see 98f:68011.

Brock, S. (1996). Compile-time pointer reversal. *J. Funct. Logic Programming*, 1996:Approx. 33 pp. (electronic).

[MR97h:68009] Summary: "This paper introduces an alternative representation for λ-terms which has the notable property that the search for the leftmost outermost redex is restricted to two steps. This is important in the implementation of a lazy functional programming language, as this search consumes time and space. The representation introduced is similar to that resulting from the implementation technique of reversing pointers in the left spine of a term while traversing it, except that here the pointers in the left spine are reversed before reduction. This paper completely develops this new representation, including rigourous proofs of the correctness as a representation for λ-terms and a number of properties, such as the restriction on the search for the next redex. It is shown that the representation can be used with graphs, hence it can be used as the basis for an implementation of a lazy functional language. An implementation is introduced and its performance is compared with a conventional implementation, showing a notable increase in efficiency."

Di Cosmo, R. and Kesner, D. (1996). Combining algebraic rewriting, extensional lambda calculi, and fixpoints. *Theoret. Comput. Sci.*, 169(2):201–220. ICALP '94 (Jerusalem, 1994).
[MR98g:03041] The present paper gives an elegant and natural solution to the problem of incorporating extensional equalities and fixpoint combinators in higher-order programming languages with algebraic data types.

It is well known that confluence fails when combining algebraic rewriting systems with simply typed λ-calculus in the presence of the usual contraction rule for η or of recursion together with the usual contraction rule for surjective pairing.

The authors first show how the combination of first-order algebraic rewriting systems with the typed λ-calculus preserves strong normalization and confluence with expansion rules for η and surjective pairing. Then they prove that fixpoints can be added, preserving confluence, to any left-linear rewriting system, possibly containing expansion rules. For the entire collection see MR 97i:68005. Reviewed by Mariangiola Dezani-Ciancaglini.

Gandhe, M., Venkatesh, G., and Sanyal, A. (1996). Correcting type errors in the Curry system. In *Foundations of software technology and theoretical computer science (Hyderabad, 1996)*, pages 347–358. Springer, Berlin.
[MR98m:68025] Summary: "We address the problem of suggesting corrections to terms that are ill-typed in the Curry system. For this, we convert the term into a set of constraints over type expressions, and use a method due to P. T. Cox to compute the maximal consistent subset of this set. The main result of this paper is an interesting correspondence between the automaton constructed in Cox's method and a graphical representation of the term, called a sharing graph, which allows us to read back the term from the maximal consistent subset. We also characterise ill-typed terms using paths in the sharing graph." For the entire collection see MR 98i:68006.

Krivine, J.-L. (1996). About classical logic and imperative programming. *Ann. Math. Artificial Intelligence*, 16(1-4):405–414.
[MR97a:68103] Summary: "We consider a λ-calculus system in second-order classical logic, in which execution of λ-terms is weak head reduction (call-by-name). By means of some simple examples, we show how it can model some aspects of imperative programming languages, such as assignment and escape instructions. A crucial tool is the notion of storage operator, for which a simple type is given."

Lavatelli, C. (1996). Algebraic interpretation of lambda calculus with resources. In *CONCUR '96: concurrency theory (Pisa)*, pages 422–437. Springer, Berlin.
[MR99a:03020] Summary: "Lambda calculus with resources λ_r is a nondeterministic refinement of lazy lambda calculus which allows the control of argument availability and introduces the possibility of raising deadlocks during evaluation. We

apply Levy's program [J.-J. Levy, Theoret. Comput. Sci. 2 (1976), no. 1, 97–114; MR 53 #12891] to λ_r: we give an algebraic interpretation of this calculus in terms of approximants which is adequate with respect to the observational semantics of λ_r." For the entire collection see MR 98e:68002.

Miller, D. (1996). Forum a multiple-conclusion specification logic. *Theoret. Comput. Sci.*, 165(1):201–232. Fourth International Conference on Algebraic and Logic Programming (Madrid, 1994).

[MR97k:68033] Summary: "The theory of cut-free sequent proofs has been used to motivate and justify the design of a number of logic programming languages. Two such languages, λProlog and its linear logic refinement, Lolli [J. S. Hodas and D. A. Miller, Inform. and Comput. 110 (1994), no. 2, 327–365; MR 95c:68044], provide for various forms of abstraction (modules, abstract data types, and higher-order programming) but lack primitives for concurrency. The logic programming language, LO (linear objects) [J.-M. Andreoli and R. Pareschi, in Logic programming (Jerusalem, 1990), 495–510, MIT Press, Cambridge, MA, 1990; see MR 92d:68014], provides some primitives for concurrency but lacks abstraction mechanisms. In this paper we present Forum, a logic programming presentation of all of linear logic that modularly extends λProlog, Lolli and LO. Forum, therefore, allows specifications to incorporate both abstractions and concurrency. To illustrate the new expressive strengths of Forum, we specify in it a sequent calculus proof system and the operational semantics of a programming language that incorporates references and concurrency. We also show that the metatheory of linear logic can be used to prove properties of the object-languages specified in Forum." For the entire collection see MR 97d:68005.

Nour, K. (1996). Opérateurs de mise en mémoire et types ∀-positifs. *RAIRO Inform. Théor. Appl.*, 30(3):261–293.

[MR98h:03020] This paper gives one of the two parts of the proof of Krivine's storage operator theorem.

A term T of the λ-calculus is called a storage operator for a set of closed λ-terms D if it simulates the call-by-value reduction in weak-head-reduction for the terms in D, that is (the exact definition is slightly more complex):

$$(\forall t \in D)(\exists t_0 \sim_\beta t)(\forall t' \sim_\beta t)(T)\, f\, t' \succ (f)\, t_0.$$

Let us define the negation by $A = A \to O$, where O is a propositional variable. Let A^* be the Gödel translation of A (using this negation). J. L. Krivine's storage operator theorem states that any term T of type $D \to D^*$ in system $\mathcal{F}$ is a storage operator for any term of type D. This theorem holds for various Gödel translations when D is an ∀-positive type (a type with only positive occurrences of the second-order universal quantifier). Therefore, it holds when D defines a data type such as naturals or lists. This result and its generalization to classical logic have many applications [see, e.g., J.-L. Krivine, Ann. Math. Artificial Intelligence 16 (1996), no. 1-4, 405–414; MR 97a:68103].

Krivine's original proof [Arch. Math. Logic 30 (1990), no. 4, 241–267; MR 91j:03011] was semantical and it was published only for the type of Church numerals (his proof for all ∀-positive types was very intricate). The author gives a purely syntactic proof for all ∀-positive types and a large variety of Gödel translations. There is a new

semantical proof [C. Raffelli, Ann. Pure Appl. Logic 91 (1998), no. 1, 17–31], which also extends a weaker result to any type.

The proof in this paper is in two parts. The first part, which is not included in this paper but appears elsewhere [R. David and K. Nour, J. Symbolic Logic 60 (1995), no. 4, 1054–1086; MR 96k:03035], gives a characterization of storage operators in pure λ-calculus using the directed λ-calculus. This tool was introduced to avoid the use of an infinite number of variables as in Krivine's proof. This characterization allows one to test whether T is a storage operator for t by checking the reduction $(T)\, f\, [t]_0$ for only one directed λ-term $[t]_0$.

The second part uses two trivial facts on typing: (i) The typing of $\forall$-positive types needs only a restriction of the second-order $\forall$ quantifier to variables: $\Gamma \vdash t\colon \forall X\, A$ implies $\Gamma \vdash t\colon A[X/Y]$. (ii) The hypotheses ending with O (O is the right-most variable) in a context are not used if the conclusion and the other hypotheses do not use O.

From (i) we get that if $\vdash t\colon D$ then $\vdash t\colon D^*$. Then we have $f\colon D \vdash (T)\, f\, [t]_0\colon O$. By carefully examining the typed reduction of this directed λ-term (this is the technical part of the paper) and using (ii), we get just what is needed to prove that T is a storage operator using the characterization from the first part of the proof. Reviewed by Christophe Raffalli.

Pravato, A. and Roversi, L. (1996). $\lambda_!$ considered both as a paradigmatic language and as a meta-language. In *Theoretical computer science (Ravello, 1995)*, pages 146–161. World Sci. Publishing, River Edge, NJ.

[MR98b:03049] Summary: "Intuitionistic linear logic (ILL) is a resource-conscious logic. The Curry-Howard isomorphism (CHI), applied to ILL, generates typed functional-like languages that have primitive constants by means of which the amount of resources (terms) used during a computation is made explicit. $\Lambda_!^-$ is an untyped functional-like language inspired by a typed language connected with ILL by CHI. We want to use the resource-aware language $\Lambda_!^-$ both as a paradigmatic programming language and as a metalanguage for implementing a fragment of the untyped λ-calculus λ_β. For the former use we give an algorithm for automatically assigning formulas of ILL as types to terms of $\Lambda_!^-$. Concerning the second kind of use, we introduce a one-step translation Tr from the fragment Λ^C of λ_β that can be typed a la Curry to the typable fragment of $\Lambda_!^-$ in ILL. Tr preserves the linear-behaved terms of Λ^C and is both correct and complete, in a reasonable sense, with respect to the β-reductions on λ_β." For the entire collection see MR 97k:68013.

Qian, Z. and Wang, K. (1996). Modular higher-order equational preunification. *J. Symbolic Comput.*, 22(4):401–424.

[MR98g:68021] Summary: "Preunification of simple typed λ-terms with respect to the equivalence relation induced by α-, β- and η-conversion and an arbitrary first-order equational theory is useful in higher-order proof and logic programming systems. In this paper we present a procedure for such preunification, which is based on three transformations and parameterized by a first-order equational unification procedure that admits free function symbols. The procedure is proved to be sound and complete, provided that the parameterizing procedure is."

Tronci, E. (1996). Equational programming in λ-calculus via SL-systems. I, II. *Theoret. Comput. Sci.*, 160(1-2):145–184, 185–216.

[MR97j:68085] From the summaries of Parts I and II: "A system of equations in the λ-calculus is a set of formulas of Λ (the equations) together with a finite set of variables of Λ (the unknowns). A system $\mathcal{S}$ is said to be β-solvable [$\beta\eta$-solvable] iff there exists a simultaneous substitution with closed λ-terms for the unknowns that makes the equations of $\mathcal{S}$ theorems in the theory $\beta[\beta\eta]$. A system $\mathcal{S}$ can be viewed as a set of specifications (the equations) for a finite set of programs (the unknowns), whereas a solution for $\mathcal{S}$ yields executable codes for such programs.

"A class $\mathfrak{S}$ of systems for which the solvability problem is effectively decidable defines an equational programming language and a system solving algorithm for $\mathfrak{S}$ defines a compiler for such a language.

"This leads us to consider separation-like systems (SL-systems), i.e. systems with equations having the form $x\vec{M} = z$, where x is an unknown and z is a free variable which is not an unknown.

"In Part I, we show that the β-solvability [$\beta\eta$-solvability] problem for SL-systems is undecidable. On the other hand we are able to define a class of SL-systems (regular SL-systems) for which the β-solvability problem is decidable in polynomial time. Such a class yields an equational programming language in which self-application is handled, constraints on executable code to be generated by the compiler can be specified by the user and (properties of) data structures can be described in an abstract way."

"In Part II, we show that there is a class of SL-systems (NP-regular SL-systems) for which the β-solvability problem is NP-complete. Moreover, we show that any SL-system $\mathcal{S}$ can be transformed into an NP-regular SL-system $\mathcal{S}'$. This transformation consists of adding abstractions to the LHS occurrences of the RHS variables of $\mathcal{S}$. In this sense NP-regular SL-systems isolate the source of undecidability for SL-systems, namely: a shortage of abstractions on the LHS occurrences of the RHS variables.

"NP-regular SL-systems yield an equational programming language in which unrestrained self-application is handled. However, existence of executable code satisfying a specification in such language is an NP-complete problem. This is the price we have to pay for allowing unrestrained self-application in our language." Reviewed by Martin W. Bunder.

Ariola, Z. M. and Blom, S. (1997). Cyclic lambda calculi. In *Theoretical aspects of computer software (Sendai, 1997)*, pages 77–106. Springer, Berlin.

[MR1608938] This item will not be reviewed individually. For details of the collection in which this item appears see 98i:68019.

Ariola, Z. M. and Felleisen, M. (1997). The call-by-need lambda calculus. *J. Funct. Programming*, 7(3):265–301.

[MR99g:68024]

Summary: "G. D. Plotkin showed that the lambda calculus is a good model of the evaluation process for call-by-name functional programs. Reducing programs to constants or lambda abstractions according to the leftmost-outermost strategy exactly mirrors execution on an abstract machine like Landin's SECD machine. The machine-based evaluator returns a constant or the token 'closure' if and only if the standard

reduction sequence starting at the same program will end in the same constant or in some lambda abstraction. However, the calculus does not capture the sharing of the evaluation of arguments that lazy implementations use to speed up the execution. More precisely, a lazy implementation evaluates procedure arguments only when needed and then only once. All other references to the formal procedure parameter re-use the value of the first argument evaluation. The mismatch between the operational semantics of the lambda calculus and the actual behavior of the prototypical implementation is a major obstacle for compiler writers. Unlike implementors of the leftmost-outermost strategy or of a call-by-value language, implementors of lazy systems cannot easily explain the behavior of their evaluator in terms of source level syntax. Hence, they often cannot explain why a certain syntactic transformation 'works' and why another doesn't. In this paper we develop an equational characterization of the most popular lazy implementation technique—traditionally called 'call-by-need'—and prove it correct with respect to the original lambda calculus. The theory is a strictly smaller theory than Plotkin's call-by-name lambda calculus. Immediate applications of the theory concern the correctness proofs of a number of implementation strategies, e.g. the call-by-need continuation passing transformation and the realization of sharing via assignments. Some of this material first appeared in a paper presented at the 1995 ACM Conference on the Principles of Programming Languages. That paper was a joint effort with J. Maraist, M. Odersky and P. Wadler, who had independently developed a different equational characterization of call-by-need. We contrast our work with theirs in the body of this paper where appropriate."

Ariola, Z. M. and Klop, J. W. (1997). Lambda calculus with explicit recursion. *Inform. and Comput.*, 139(2):154–233.
[MR99c:68031]

This paper studies lambda graphs that may contain cycles, also called lambda calculus with explicit recursion. Such lambda graphs or cyclic lambda terms are represented by systems of recursion equations $\langle M \mid E\rangle \equiv \langle M \mid x_1 = M_1, \cdots, x_n = M_n\rangle$ which can be seen as another notation for the letrec construct. These systems can be nested. The initial part of the paper studies some peculiarities of this set-up, having to do with the notion of bound variable (bound by an abstractor λx), and cyclic 'empty' references of the recursion variables that may arise due to application of collapsing rules (e.g. $\langle M \mid x_1 = x_2,\ x_2 = x_1\rangle$). The paper then presents first a "naive" rewrite system for such cyclic lambda terms, involving, in addition to beta-reduction, also the basic term graph transformation of "copying" (used, e.g., in unwinding a cycle), and garbage collection (removal of inaccessible parts). Also substitution for some occurrence of a recursion variable x_i of the right-hand side of its defining equation $x_i = M_i$ is allowed. It is shown that this rewrite system is not confluent. This non-confluence is analysed with the help of infinitary lambda calculus, employing lambda trees of infinite depth instead of the usual finite lambda terms (lambda trees). These trees of infinite depth arise from the unwinding of cyclic lambda graphs. Some more counterexamples to confluence of the naive rewrite sys-

tem for cyclic lambda calculus are studied, one of which employs non-regular lambda trees. Next, the authors study some restricted calculi for cyclic lambda rewriting, proving confluence for them. One of these calculi is able to embed the lambda calculus extended with the -operator, with the rewrite rule $x.t(x) \to t(x.t(x))$. A lengthy appendix discusses cyclic interaction nets for the lambda calculus where sharing is obtained by fan-in nodes following Lamping. In this framework the counterexamples to confluence for cyclic lambda terms disappear, and the paper extensively analyses why this is the case. Reviewed by Detlef Plump.

Asperti, A. and Chroboczek, J. (1997). Safe operators: brackets closed forever. Optimizing optimal λ-calculus implementations. *Appl. Algebra Engrg. Comm. Comput.*, 8(6):437–468.

[MR99b:68025] Summary: "Considerations from category theory described by Asperti [Fund. Inform. 22 (1995), no. 1-2, 3–22; MR 96c:03027] have permitted the addition of new rewriting rules for optimal reductions of the λ-calculus. These rules produce an impressive improvement in the performance of the reduction system, and provide a first step towards the solution of the well-known and crucial problem of the accumulation of control operators. In this paper, after an introduction to optimal reductions, we exhibit the aforementioned problem and prove the correctness of the new rules."

Barbanera, F., Berardi, S., and Schivalocchi, M. (1997). "Classical" programming-with-proofs in $\lambda^{\mathrm{sym}}_{\mathrm{pa}}$: an analysis of non-confluence. In *Theoretical aspects of computer software (Sendai, 1997)*, pages 365–390. Springer, Berlin.

[MR1608899] This item will not be reviewed individually. For details of the collection in which this item appears see 98i:68019.

Barthe, G., Kamareddine, F., and Ríos, A. (1997). Explicit substitutions for the $\lambda\delta$-calculus. In *Algebraic and logic programming (Southampton, 1997)*, pages 209–223. Springer, Berlin.

[MR1639637] This item will not be reviewed individually. For details of the collection in which this item appears see 99b:68002.

Bono, V. and Bugliesi, M. (1997). Matching constraints for the lambda calculus of objects. In *Typed lambda calculi and applications (Nancy, 1997)*, pages 46–62. Springer, Berlin.

[MR98i:68039] Summary: "We present a new type system for the lambda calculus of objects [K. Fisher, F. Honsell and J. C. Mitchell, Nordic J. Comput. 1 (1994), no. 1, 3–37; MR 95m:68109], based on matching. The new system retains the property of type safety of the original system, while using implicit match-bounded quantification over type variables instead of implicit quantification over row schemes (as in the above-cited paper) to capture Mytype polymorphic types for methods. Type soundness for the new system is proved as a direct corollary of subject reduction. We also carry out a study of the relative expressive power of the two systems that

shows that the new system is as powerful as the original one on derivations of closed-object typing judgements. Finally, we present an extension of the new system that makes provision for a class-based calculus, where primitives such as creation of class instances and method update are rendered in terms of delegation." For the entire collection see MR 98f:68017.

Castagna, G. (1997a). *Object-oriented programming.* Birkhäuser Boston Inc., Boston, MA. A unified foundation, With a foreword by Luca Cardelli.

[MR98h:68025] From the text: "Simula is the precursor of object-oriented languages. Its style of programming was followed by all class-based object-oriented languages till the appearance of the so-called multiple-dispatching languages, such as CLOS, where a different style of object-oriented programming was introduced.

"Although much work has been done to define a theory of object-oriented languages, it has mostly concentrated on languages that follow Simula's style, ignoring the multiple-dispatching languages. This book fills the gap, by developing a theory that accounts for both classes of languages, thus providing a more comprehensive understanding of object-oriented concepts.

"The theory gives a unified view of two worlds, that of Simula and of CLOS, which were hitherto disconnected. We concretely show that this unified view can be integrated in a unique programming language in which both styles of programming can be used and even intermingled. The theory also suggests a possible solution to some longstanding problems, the interpretation and improvement of existing constructions, and the definition of new ones.

"The book is self-contained. Although an acquaintance with type theory would help, all the standard notions used in this book (type system, type algorithm, subtyping, soundness, lambda-notation, etc.) are introduced and explained in the first chapter.

"We use part of this book as notes for a 20-hour graduate level course in subtyping and object-oriented programming. We also use Chapter 1 as notes for an 8-hour graduate-level-course of introduction to type-theory and type-checking.

"This book is a revised and extended version of the author's Ph.D. thesis, and it describes the results of more than four years of work. Many of the results in this book have already been published in journals or conference proceedings. More precisely, Chapters 4 and 6 are based on an article by the author, G. Ghelli and G. Longo [Inform. and Comput. 117 (1995), no. 1, 115–135; MR 95m:68108]. The third and eighth chapters are partially based on (and strongly improve) the contents of another article by the author [Theoret. Comput. Sci. 151 (1995), no. 2, 297–352; MR 97i:68125]. Part of Chapter 5 appeared in [G. Castagna, ACM Trans. Program. Lang. Systems 17 (1995), no. 3, 431–447]. Part of Section 7.3 is contained in a note in [G. Castagna, Theoret. Comput. Sci. 176 (1997), no. 1-2, 337–345; MR 97m:68022]. The extended abstract of Chapter 10 can be found in [G. Castagna, G. Ghelli and G. Longo, in Typed lambda calculi and applications (Utrecht, 1993), 107–123, Lecture Notes in Comput. Sci., 664, Springer, Berlin, 1993; see MR 94b:68025CMP 1 233 143]. Chapter 11 and part of Chapter 12 are contained in [G. Castagna, Formal Aspects Comput. 8 (1996), no. 3, 247–293; Zbl 852.68011]."

Contents: I. Introduction. 1. Background and notation. 2. A quick overview. II. Simple typing. 3. Object-oriented programming. 4. The $\lambda\&$-calculus. 5. Covariance and contravariance: conflict without a cause. 6. Strong normalization. 7. Three variations on the theme. 8. Interpretation of object-oriented languages. 9. Imperative features and other widgets. 10. Semantics. III. Second order. 11. Introduction to Part III.

12. Second order overloading. 13. Second order overloading and object-oriented programming. 14. Conclusion. IV. Appendixes. A. Specification of KOOL. B. Formal definition of the translations.

Castagna, G. (1997b). Unifying overloading and λ-abstraction: $\lambda^{\{\}}$. *Theoret. Comput. Sci.*, 176(1-2):337–345.

[**MR97m:68022**] Summary: "In this short note, we present a minimal system to deal with overloaded functions with late binding, in which λ-abstractions are seen as a special case of overloaded functions with just one code. We prove some relevant properties that this system enjoys, and show its connection with the $\lambda\&$-calculus. We end by showing the practical interest of this system, in particular in modeling object-oriented languages."

Felleisen, M. and Weeks, S. (1997). On the orthogonality of assignments and procedures in ALGOL. In *ALGOL-like languages, Vol. 1*, pages 101–124. Birkhäuser Boston, Boston, MA.

[**MR98d:68036**] Summary: "According to folklore, ALGOL is an 'orthogonal' extension of a simple imperative programming language with a call-by-name functional language. The former contains assignments, branching constructs, and compound statements; the latter is based on the typed λ-calculus. To formalize ALGOL's notion of 'orthogonality', we define an extended λ-calculus that models ALGOL. The calculus includes the full β-reduction rule and reduction rules for assignment statements and commands. It has the usual Church-Rosser property and its recursion-free subset is strongly normalizing.

"In support of the 'orthogonality' claim, we show that the proofs of the Church-Rosser and strong normalization theorems are combinations of separate theorems for each sublanguage. In addition, the calculus satisfies a postponement theorem which operationally formalizes the idea that the evaluation of an ALGOL program has two distinct phases which correspond to the two sublanguages. The first phase corresponds to an unrolling of the program according to the usual β and fixpoint reductions. The result of this phase is essentially an imperative program. The second phase corresponds to the execution of an imperative program on an ordinary stack machine." For the entire collection see MR 98c:68003a.

Hatcliff, J. and Danvy, O. (1997). Thunks and the λ-calculus. *J. Funct. Programming*, 7(3):303–319.

[**MR98i:68040**] Summary: "Thirty-five years ago, thunks were used to simulate call-by-name under call-by-value in Algol 60. Twenty years ago, Plotkin presented continuation-based simulations of call-by-name under call-by-value and vice versa in the λ-calculus. We connect all three of these classical simulations by factoring the continuation-based call-by-name simulation $\mathcal{C}_n$ as a thunk-based call-by-name simulation $\mathcal{T}$ followed by the continuation-based call-by-value simulation $\mathcal{C}_v$ extended to thunks. We show that $\mathcal{T}$ actually satisfies all of Plotkin's correctness criteria for $\mathcal{C}_n$ (i.e. his indifference, simulation and translation theorems). Furthermore, most of the correctness theorems for $\mathcal{C}_n$ can now be seen as simple corollaries of the corresponding theorems for $\mathcal{C}_v$ and $\mathcal{T}$."

Kesner, D. (1997). Reasoning about redundant patterns. *J. Funct. Logic Programming*, 1997:Article 4, 48 pp. (electronic).
[MR1692468] A review for this item is in process.

Kuang, H., Sun, Y., and Lin, K. (1997). A partial evaluator for a parallel lambda language. *J. Comput. Sci. Tech.*, 12(5):441–457.
[MR1474176] There will be no review of this item.

Lawall, J. L. and Thiemann, P. (1997). Sound specialization in the presence of computational effects. In *Theoretical aspects of computer software (Sendai, 1997)*, pages 165–190. Springer, Berlin.
[MR1608950] This item will not be reviewed individually. For details of the collection in which this item appears see 98i:68019.

Maćoš, D., Budimac, Z., and Ivanović, M. (1997). An interpreter of extended λ-calculus using monads. In *PRIM '96 (Budva)*, pages 175–180. Univ. Novi Sad, Novi Sad.
[MR1492232] This item will not be reviewed individually. For details of the collection in which this item appears see 98h:00033.

Mezghiche, M. (1997). $c\beta$-machine with $\lambda\beta$-reduction. *Theoret. Comput. Sci.*, 189(1-2):221–228.
[MR98m:68027] Summary: "In this paper we describe a $c\beta$-machine which implements in a simple way the $c\beta$-reduction. This reduction is a combinatory reduction which is equivalent to the $\lambda\beta$-reduction and was defined by Mezghiche [Theoret. Comput. Sci. 31 (1984), no. 1-2, 151–163; MR 86j:03009; Theoret. Comput. Sci. 66 (1989), no. 3, 323–331; MR 91e:03018]. Some definitions and properties of $c\beta$-reduction are recalled before giving the definition of the $c\beta$-machine."

Muñoz H., C. A. (1997). A left-linear variant of $\lambda\sigma$. In *Algebraic and logic programming (Southampton, 1997)*, pages 224–239. Springer, Berlin.
[MR1639638] This item will not be reviewed individually. For details of the collection in which this item appears see 99b:68002.

Ørbæk, P. and Palsberg, J. (1997). Trust in the λ-calculus. *J. Funct. Programming*, 7(6):557–591.
[MR98i:68042] Summary: "This paper introduces trust analysis for higher-order languages. Trust analysis encourages the programmer to make explicit the trustworthiness of data, and in return it can guarantee that no mistakes with respect to trust will be made at run-time. We present a confluent λ-calculus with explicit trust operations, and we equip it with a trust-type system which has the subject reduction property. Trust information is presented as annotations of the underlying Curry types, and type inference is computable in $O(n^3)$ time."

Pitts, A. M. (1997). A note on logical relations between semantics and syntax. *Log. J. IGPL*, 5(4):589–601 (electronic).

[**MR98e:68022**] Summary: "This note gives a new proof of the 'operational extensionality' property of Abramsky's lazy lambda calculus—namely the coincidence of contextual equivalence with a co-inductively defined notion of 'applicative bisimilarity'. This purely syntactic result is here proved using a logical relation (due to Plotkin) between the syntax and its denotational semantics. The proof exploits a mixed inductive/co-inductive characterisation of the logical relation recently discovered by the author."

Roversi, L. (1997). A type-free resource-aware λ-calculus. In *Computer science logic (Utrecht, 1996)*, pages 399–413. Springer, Berlin.
[**MR98k:03039**] Summary: "We introduce and study a functional language λ_{R}, having two main features. λ_{R} has the same computational power as the λ-calculus, and it enjoys the resource-awareness of the typed/typable functional languages which encode intuitionistic linear logic." For the entire collection see MR 98k:03005.

Schwichtenberg, H. (1997). Proofs, lambda terms and control operators. In *Logic of computation (Marktoberdorf, 1995)*, pages 309–347. Springer, Berlin.
[**MR1492467**] This item will not be reviewed individually. For details of the collection in which this item appears see 98i:68008.

Stark, I. (1997). Names, equations, relations: practical ways to reason about new. In *Typed lambda calculi and applications (Nancy, 1997)*, pages 336–353. Springer, Berlin.
[**MR98g:03046**] Summary: "The nu-calculus of A. M. Pitts and I. Stark is a typed lambda-calculus, extended with states in the form of dynamically generated names. These names can be created locally, passed around, and compared with one another. Through the interaction between names and functions, the language can capture notions of scope, visibility and sharing. Originally motivated by the study of references in standard ML, the nu-calculus has connections to other kinds of local declaration, and to the mobile processes of the π-calculus.

"This paper introduces a logic of equations and relations which allows one to reason about expressions of the nu-calculus: this uses a simple representation of the private and public scope of names, and allows straightforward proofs of contextual equaivalence (also known as observational, or observable, equivalence). The logic is based on earlier operational techniques, providing the same power but in a much more accessible form. In particular it allows intuitive and direct proofs of all contextual equivalences between first-order functions with local names." For the entire collection see MR 98f:68017.

Stärk, R. F. (1997). Call-by-value, call-by-name and the logic of values. In *Computer science logic (Utrecht, 1996)*, pages 431–445. Springer, Berlin.
[**MR1611513**] This item will not be reviewed individually. For details of the collection in which this item appears see 98k:03005.

Banerjee, A. and Schmidt, D. A. (1998). Stackability in the simply-typed call-by-value lambda calculus. *Sci. Comput. Programming*, 31(1):47–73.

SAS'94 (Namur).
[**MR1634534**] Summary: "We address two issues: (1) what it means for a higher-order, eager functional language to be implemented with a single, global, stack-based environment and (2) how the existence of such an environment can be predicted statically.

"The central theme is the use of the λ-abstraction to control the extent or lifetime of bindings. All programs in a higher-order, call-by-name language can be implemented with a stack environment. The reason: soundness of η-expansion and decurrying for call-by-name. However, η-expansion is unsound for call-by-value. Hence, we must identify a subset of the simply typed, call-by-value λ-calculus in which λ-abstraction can serve as the block construct for a stack implementation.

"The essence of environment stackability is that the shape of the environment remains the same before and after the execution of an expression. Thus, if a closure is returned as a value, the environment trapped in it must be a subenvironment of the global environment. This yields a dynamic criterion for stackability—indeed, it is the downwards funargs criterion of the LISP community. A safe static criterion can now be found via closure analysis." For the entire collection see MR 99a:68013. Reviewed by E. Fehr.

Bierman, G. M. (1998). A computational interpretation of the $\lambda\mu$-calculus. In *Mathematical foundations of computer science, 1998 (Brno)*, pages 336–345. Springer, Berlin.
[**MR1684077**] A review for this item is in process.

Curien, P.-L. (1998). Abstract Böhm trees. *Math. Structures Comput. Sci.*, 8(6):559–591. Logic and models of computation (Marseille, 1996).
[**MR1672401**] A review for this item is in process.

Curien, P.-L. and Herbelin, H. (1998). Computing with abstract Böhm trees. In *Third Fuji International Symposium on Functional and Logic Programming (Kyoto, 1998)*, pages 20–39. World Sci. Publishing, River Edge, NJ.
[**MR1661812**] A review for this item is in process.

de Groote, P. (1998). An environment machine for the $\lambda\mu$-calculus. *Math. Structures Comput. Sci.*, 8(6):637–669. Logic and models of computation (Marseille, 1996).
[**MR1672403**] Summary: "We introduce a natural deduction-like formalisation of Parigot's λ-calculus. From this, we derive an environment machine that allows any well-typed λ-term to be reduced to a weak head normal form. The soundness and completeness of the machine is proved." For the entire collection see MR 99h:03001.

Denney, E. (1998). Simply-typed underdeterminism. In *Proceedings of the International Summer School and Workshop on Formal Models of Programming and their Applications (Beijing, 1997)*, volume 13, pages 491–508.
[**MR1677503**] A review for this item is in process.

Duggan, D. (1998). Unification with extended patterns. *Theoret. Comput. Sci.*, 206(1-2):1–50.

[MR1641694] A review for this item is in process.

Hartonas, C. and Hennessy, M. (1998). Full abstractness for a functional/concurrent language with higher-order value-passing. *Inform. and Comput.*, 145(1):64–106.
[MR99g:68027]

Models for concurrent systems and programming languages are the subject of intensive research, aiming at abstract description of existing systems, and devising design principles for new programming languages.

The present paper is concerned with higher-order languages, namely systems which are able to treat nonelementary data types as first class objects. This amounts to having a value passing mechanism which allows for transmission of functions and ("delayed", namely abstracted) processes. It is based on a simply typed λ-calculus with a special type for processes. The syntax of process terms is a CCS dialect, with a nonstandard hiding operator.

Operational semantics is provided via a transition system, which is inherently nonconfluent. To compare processes the authors define a notion of observable property (either to reduce to a process term capable of transmitting on a channel, or, equivalently, to evaluate to a process sending a transmittable value) and the relative observational preorder, which is, essentially, a "may-testing" preorder.

Following the lines of previous work by one of the authors, a denotational model is then constructed in the category of prime algebraic lattices. It is known that such domains admit a finitary description in terms of compact primes, and this is used to define a programming logic and a type assignment system, such that types describe compact elements in the domain, and the set of types that can be deduced for (closed) terms completely determines the term denotation. Then one has an almost fixed suite of theorems to prove: completeness of the type system, adequacy and full abstraction with respect to the observational preorder.

The paper is carefully written, and provides a good example of application of established techniques to a problem which is still challenging: to provide abstract semantics and proof tools for reasoning about concurrent systems, which also exhibit some features essential to higher-order languages. The technical treatment is centered on a particular system, but concepts and techniques are general enough, and could be adapted, with a moderate effort, to variations of the main theme.

The authors point out limitations and open problems of their work. My personal criticism is rather that the theoretical framework used shows here, more dramatically than elsewhere, its natural limits. Indeed, to model interleaving of input/output actions among processes by a sort of abstract interleaving among concretions and functions is not that radical a departure from syntax, and suffers from similar problems and intricacies. In any case the solution adopted by the authors is a sophisticated and elegant one, improving on existing proposals. Reviewed by Ugo de'Liguoro.

Hashimoto, M. and Ohori, A. (1998). A typed context calculus. *Sūrikaisekikenkyūsho Kōkyūroku*, (1023):76–91. Type theory and its application to computer systems (Japanese) (Kyoto, 1997).

[MR99g:68028] Summary: "This paper develops a typed calculus for contexts,

that is, lambda terms with 'holes'. In addition to ordinary lambda terms, the calculus contains labeled holes, hole abstraction and context application for manipulating first-class contexts. The primary operation for contexts is hole-filling, which captures free variables. This operation conflicts with β-substitution of the lambda calculus. Development of a confluent calculus that combines β-substitution and hole-filling constitutes a non-trivial technical challenge. We solve this problem by defining a type system that precisely specifies the variable-capturing nature of contexts, and by developing a proper mechanism for renaming variables to reconcile capture-avoiding substitution and variable-capturing hole-filling. The resulting calculus is Church-Rosser and the type system has the subject reduction property. We believe that the context calculus will serve as a basis for developing a programming language with advanced features that call for manipulation of open terms." For the entire collection see MR 99d:68014.

Kamareddine, F., Ríos, A., and Wells, J. B. (1998). Calculi of generalized β-reduction and explicit substitutions: the type-free and simply typed versions. *J. Funct. Logic Programming*, 1998:Article 5, 44 pp. (electronic).

[MR1679610] A review for this item is in process.

Loader, R. (1998). Unary PCF is decidable. *Theoret. Comput. Sci.*, 206(1-2):317–329.

[MR99i:03013] Plotkin's PCF is a typed lambda calculus with natural numbers, basic numerical operations and recursion at arbitrary types. The observational preorders on the terms of PCF and those of many simpler subsystems have been shown to be undecidable. The author shows that unary PCF, which has only one ground type 0, constants $\top, \bot : 0$ and a binary operator $\wedge : 0 \Rightarrow 0 \Rightarrow 0$ with β reduction, η expansion and two simple reductions for $\wedge$, is decidable.

A later paper by M. Schmidt-Schauss [Theoret. Comput. Sci. 216 (1999), no. 1-2, 363–373CMP 1 676 276] has a simpler proof of this result. Reviewed by Martin W. Bunder.

Maraist, J., Odersky, M., and Wadler, P. (1998). The call-by-need lambda calculus. *J. Funct. Programming*, 8(3):275–317.

[MR1652623] A review for this item is in process.

Nadathur, G. and Wilson, D. S. (1998). A notation for lambda terms. A generalization of environments. *Theoret. Comput. Sci.*, 198(1-2):49–98. **[MR99j:68010]**

This paper could have been subtitled "How to make substitution easier and lazier". Certainly, it is wiser to postpone substitution until the opportune moment rather than to do it right away. To carry out this idea and to have "a fine-grained control" over the substitution process, the authors introduce a new kind of "λ-terms", called suspension terms, and incorporate environments into the terms themselves.

From the summary: "A category of terms is provided that can encode other terms together with substitutions to be performed on them. The notion of an environment is used to realize this 'delaying' of substitutions. However, the precise environment mechanism employed here is more complex than the usual one because the ability to examine subterms embedded under abstractions has to be supported. The representation presented permits a β-contraction to be realized via an atomic step that generates a substitution and associated steps that percolate this substitution over the structure of a term. Operations on terms are provided that allow for the combination and hence the simultaneous performance of substitutions. Our notation eventually provides a basis for efficient realizations of β-reduction and also serves as a means for interleaving the steps inherent in this operation with steps in other operations such as higher-order unification. Manipulations on our terms are described through a system of rewrite rules whose correspondence to the usual notion of β-reduction is exhibited and exploited in establishing confluence and other similar properties. Our notation is similar in spirit to recent proposals deriving from the categorical combinators of Curien, and the relationship to the latter is discussed." Reviewed by M. Yasuhara.

Raffalli, C. (1998). A semantical storage operator theorem for all types. *Ann. Pure Appl. Logic*, 91(1):17–31.
[MR99c:03095] Summary: "Storage operators are λ-terms which simulate call-by-value in call-by-name for a given set of terms. Krivine's storage operator theorem shows that any term of type $D \to D^*$, where D^* is the Gödel translation of D, is a storage operator for the terms of type D when D is a data-type or a formula with only positive second-order quantifiers. We prove that a new semantical version of Krivine's theorem is valid for every type. This also gives a simpler proof of Krivine's theorem using the properties of data-types."

Sakurai, T. (1998). Categorical model construction for proving syntactic properties. In *Third Fuji International Symposium on Functional and Logic Programming (Kyoto, 1998)*, pages 187–206. World Sci. Publishing, River Edge, NJ.
[MR1661815] A review for this item is in process.

Streicher, T. and Reus, B. (1998). Classical logic, continuation semantics and abstract machines. *J. Funct. Programming*, 8(6):543–572.
[MR1678817] A review for this item is in process.

Takahashi, M. (1998). λ-representable functions over free structures revisited. In *Third Fuji International Symposium on Functional and Logic Programming (Kyoto, 1998)*, pages 1–19. World Sci. Publishing, River Edge, NJ.
[MR1661811] A review for this item is in process.

Bono, V. and Bugliesi, M. (1999). Matching for the lambda calculus of objects. *Theoret. Comput. Sci.*, 212(1-2):101–140. Gentzen (Rome, 1996).
[MR1672068] A review for this item is in process.

Kfoury, A. J. (1999). Beta-reduction as unification. In *Logic, algebra, and computer science (Warsaw, 1996)*, pages 137–158. Polish Acad. Sci., Warsaw.

[**MR1694741**] A review for this item is in process.

Nadathur, G. (1999). A fine-grained notation for lambda terms and its use in intensional operations. *J. Funct. Logic Programming*, pages Article 2, 62 pp. (electronic).

[**MR1686143**] A review for this item is in process.

Schmidt-Schauß, M. (1999). Decidability of behavioural equivalence in unary PCF. *Theoret. Comput. Sci.*, 216(1-2):363–373.

[**MR1676276**] Summary: "Unary PCF is a fragment of the simply typed lambda-calculus PCF. We present a short proof that behavioral equivalence in unary PCF is decidable. An algorithm is described that enumerates all behaviorally equivalent expressions by increasing types, where types are ordered by a nested multiset ordering."

Spreen, D. (1999). On functions preserving levels of approximation: a refined model construction for various lambda calculi. *Theoret. Comput. Sci.*, 212(1-2):261–303. Gentzen (Rome, 1996).

[**MR1672096**] A review for this item is in process.

2.4 Grammars

Mouloud, N. (1979). Quelques hypothèses sur la porte des notions opératoires dans les langues naturelles et techniques. *Publ. Dép. Math. (Lyon)*, 16(3-4):1–11.

[**MR82m:03041**] From the conclusion: "In this article we have traced some developments in operational analysis that may help to clarify the foundations of language construction. The conditions thus highlighted affect the laws of syntax as well as the relations of syntax to categories of meaning and to the typology of references. It is possible to apply this analysis to different levels of discourse: natural, formal, empirical. This, in the eyes of the technician, the logician or the linguist, should justify new hypotheses on the operational code of languages." The author, by starting with the problem of determining which expressions of a natural language are well formed, calls attention to the problem of the operative basis (base operatoire) of a language. He does not take up in this paper the entire problem, but only the part dealing with the use of combinators to normalize syntax. In this connection he discusses both pure combinatory logic and what Curry calls illative combinatory logic. He concludes the paper with some remarks on referential languages. (The reviewer finds the paper somewhat obscure. In Section I, the reviewer cannot determine whether the "operative clauses" (clauses operatives) are the conditions which determine well-formedness or the clauses which are well formed. The reviewer is unable to find on page 56 of the article by H. B. Curry [J. Symbolic Logic 7 (1942), 49 - 64; MR 3, 289] the remarks attributed to him by the author on p. 4 (Section II). The reviewer cannot determine the meaning of the displayed formula "$[\lambda x.\Xi P\phi x \subset \Psi x]\phi Y \vdash \Psi Y$" on line 3 of page 7. On page 8 the author talks of applying $B(BCW)B^4W$ (he evidently means $B(B^2CW)B^4W$) to the clause $\mathcal{C}p\mathcal{C}qp$ to get $\mathcal{CCC}pqpp$, but this application is not the ordinary application of combinatory logic, which would be $(B(B^2CW)B^4W)(\mathcal{C}p\mathcal{C}qp)$ and which does not produce the desired result. However, if we apply the combinator $B(B^2CW)B^4W$ to the five arguments which are, in order, $\mathcal{C}, p, \mathcal{C}, q$, and p, then we do get $\mathcal{CCC}pqpp$. This work would appear to have use for more explanation.) Reviewed by J. P. Seldin.

Zielonka, W. (1979). On the equivalence of Lambek's syntactic calculus and categorial calculi. In *Fundamentals of computation theory (Proc. Conf. Algebraic, Arith. and Categorical Methods in Comput. Theory, Berlin/Wendisch-Rietz, 1979)*, pages 537–541. Akademie-Verlag, Berlin.

[MR84d:03023] The aim of this paper is to solve two problems: (i) Does there exist a finite categorial calculus in which all theorems of product-free Lambek calculus of the form $X \to s$ (s primitive) are provable? (ii) Does there exist a finite categorial calculus equivalent to product-free Lambek calculus? In both cases the answer is negative. (For the entire collection see MR 80m:68007.) Reviewed by Boleslaw Mikolajczak.

Friedman, J. and Warren, D. S. (1980). λ-normal forms in an intensional logic for English. *Studia Logica*, 39(2-3):311–324.

[MR82k:03030] R. Montague's article "The proper treatment of quantification in ordinary English" [Approaches to natural language (Stanford, Calif., 1970), pp. 221 - 242, Reidel, Dordrecht, 1973; reprinted in Formal philosophy: selected papers of R. Montague, pp. 247 - 270, Yale Univ. Press, New Haven, Conn., 1974] provided a grammar for an interesting fragment of English and a precise way of interpreting sentences of the fragment by means of formulas of a system of intensional logic called IL. IL is a version of the typed λ-calculus extended by adding operators for necessity, tense, extension, intension, and the usual quantifiers and connectives. Montague's work (op. cit.) also contains a semantical account of IL involving a set of indices, which are to be thought of as ordered pairs consisting of possible worlds and instants of time. The intension of a formula in an interpretation (relative to an assignment of values to variables) is taken to be the function defined on the set of indices which, for a given index, has as its value the extension of the formula at that index. Thus, Montague (op. cit.) also provided a precise semantical treatment of the sentences of the fragment of English defined by the grammar mentioned above. The authors define a notion of reducibility for formulas of IL by means of (1) a restricted version of the usual rule of β contraction for the λ-calculus and (2) a rule which says that a formula which arises by first applying the intension operator and then the extension operator to a given formula contracts to the formula with which one began. The restriction imposed on contracting formulas of the form $[\lambda x A](B)$ is that either (a) no free occurrence of x in A falls within the scope of the operators for necessity, tense, and intension or (b) B is modally closed. From the point of view of Montague's semantics for IL, it can be seen that modally closed formulas have constant functions as their intensions (relative to every assignment of values to variables) in every interpretation. Because variables are modally closed, the formula which is the result of replacing the free occurrences of x in A by occurrences of B need not have the same extension as $[\lambda x A](B)$ (relative to an assignment of values to variables) in a given interpretation. In particular, where $^\wedge$ is the intension operator and c is a constant of the same type as x, in a given interpretation $[\lambda x[^\wedge x]](c)$ will have as its extension the constant function having as its value the extension c has at the index at which the formula is being evaluated, but $^\wedge c$ will have the intension of c, which need not be a constant function, as its extension. This shows that some restriction on β contraction must be imposed. The authors' restriction is sufficient to overcome this difficulty, but, as they point out and as will be explained below, it is too strong to provide a satisfactory theory of reducibility for IL. However, it does provide a basis for proving the main results of the paper, which have to do with characterizing the set of formulas of IL which are actually used in interpreting sentences of the fragment of English defined by the grammar of Montague (op. cit.). The formula A of IL is taken to be a direct translation of a given English sentence if and only if it is used in Montague's work (op. cit.) as an interpretation of the sentence in question. If A is reducible to B, then B is termed a translation of any sentence of which A is a direct translation. A is in reduced form if and only if neither contraction rule can be applied to a part of A. A is fully reduced if and only if A is in reduced form and no part of A has the form $[\lambda x C](D)$. The authors show that if A is a translation, then there is a fully reduced B, unique up to alphabetic change of bound variables, such that A is reducible to B. This is obtained as a corollary of the following results. First, if A is a direct translation and $C(D)$ is a part of A, then D is modally closed. Second, the same thing is true of translations. (Actually, more is true. In translations D is either a variable or of the form $[^\wedge D_1]$, as the authors also show.) Third, if every D such that $C(D)$ is a part of A is modally closed, then there is a fully reduced B, unique up to alphabetic change of bound variables, such that A is reducible to B. On the other hand, the authors exhibit A, B_1 and B_2 such that A is reducible to B_1 and B_2, B_1 and B_2 are in reduced form but are not fully reduced, and B_1 and B_2 are not alphabetic variants of each other. Taken together with the results stated in the preceding paragraph, this certainly does provide interesting information about the sets of formulas of IL which are translations. But it can also be shown that the B_1 and B_2 in question will have the same extension (relative to every assignment of values to variables) in every interpretation (as the authors note), so the reducibility relation considered in the paper is not satisfactory from the point of view of abstract logical theory. The authors suggest that it might be possible to remedy this situation by extending the reducibility relation in the light of axiom schemata AS4.1 - AS4.7 and AS6 of D. Gallin [Intensional and higher-order modal logic, see pp. 19 - 20, North-Holland,Amsterdam, 1957; MR 58 #21470]. Reviewed by Garrel Pottinger.

Ballmer, T. T. (1983). Fuzzy grammatical categories. In *Approaching vagueness*, pages 261–292. North-Holland, Amsterdam.

[MR86m:03038] Author summary: "Sentences and other linguistic expressions stand in close relationship to prosodic properties. This requires that prosodic elements be adequately represented in formal grammars. Prosodic elements are taken into account by specific grammatical entities called punctuation signs. Punctuation based grammars (language reconstruction systems) are dealt with in this paper and three independent arguments are given for allowing for a continuum of grammatical categories. The fuzzification of λ-calculus provides the formal means of coping with the problems of fuzzy grammatical categories on a par with fuzzy logic." For the entire collection see MR 85j:03003.

Turner, R. (1983). Montague semantics, nominalization and Scott's domains. *Linguist. Philos.*, 6(2):259–288.

[MR87g:03030a] The aim of this paper is to provide an explanation for the phenomenon of nominalization (e.g., "Walking hurts") within the framework of Montague grammar. The notion of semantic domains (SDs) is introduced: A semantic domain is defined by a set of equations and contains entities of different types. A given SD may contain entities of a given type as well as partial functions which take entities of that type as arguments. An application is provided for three cases of nominalization. The formal theory of SDs is then expounded. Reviewed by Francois Lepage.

Desclés, J.-P. (1988). Théorème de Church-Rosser et structuration des langues naturelles. *Math. Sci. Humaines*, (103):67–92, 115.

[MR89k:68125] Summary (translated from the French): "Curry's combinatory logic is an applicative system, based on the constitutive operation of application and equipped with a preordering of reduction generated by elementary reductions associated canonically with abstract operators called combinators. We show the usefulness of combinators for the construction of 'complex predicates' and the formulation of 'linguistic laws'. These two concepts are used to formalize the grammatical category of voice. The linguistic sense of reduction is a paraphrastic relation between sentences. A normal form is an expression that cannot be reduced; the class of sentences that are reduced to the same normal form constitutes a semantic field of sentences. A normal form thus gives the invariant meaning of all sentences that are reducible to it. The Church-Rosser theorem affirms that when a normal form exists, it is unique. This theorem gives a solid foundation to different linguistic models of paraphrases: the Harris model, and the Shaumyan model of universal applicative grammar."

Peregrin, J. (1990). On a logical formalization of natural language. *Kybernetika (Prague)*, 26(4):327–341.

[MR91h:68148] Summary: "We investigate the possibilities and limitations of a formalization of natural language by means of the common logical systems. The applicability of the traditional Tarskian model theory to natural language is discussed.

The direction which the formalizing approach to natural language should take in order to achieve an adequate account of its semantic aspect is indicated."

Kamareddine, F. and Klein, E. (1993). Nominalization, prediction and type containment. *J. Logic Lang. Inform.*, 2(3):171–215.
[MR95e:03087] Summary: "In an attempt to accommodate natural language phenomena involving nominalization and self-application, various researchers in formal semantics have proposed abandoning the hierarchical type system which Montague inherited from Russell, in favour of more flexible type regimes. We briefly review the main extant proposals, and then develop a new approach, based semantically on Aczel's notion of Frege structure, which implements a version of subsumption polymorphism. Nominalization is achieved by virtue of the fact that the types of predicative and propositional complements are contained in the type of individuals. Russell's paradox is avoided by placing a type constraint on lambda-abstraction, rather than by restricting comprehension." Reviewed by Marcus Kracht.

Damas, L. and Moreira, N. (1995). Constraint categorial grammars. In *Progress in artificial intelligence (Funchal, 1995)*, pages 347–358. Springer, Berlin.
[MR97a:68133] Summary: "Although unification can be used to implement a weak form of β-reduction, several linguistic phenomena are better handled by using some form of λ-calculus. In this paper we present a higher-order feature description calculus based on a typed λ-calculus. We show how the techniques used in $\mathcal{CLG}$ for resolving complex feature constraints can be efficiently extended. $\mathcal{CCLG}$ is a simple formalism, based on categorial grammars, designed to test the practical feasibility of such a calculus." For the entire collection see MR 97a:68005.

Desclés, J.-P. and Biskri, I. (1995). Logique combinatoire et linguistique: grammaire catégorielle combinatoire applicative. *Math. Inform. Sci. Humaines*, (132):39–68 (1996).
[MR1393631] There will be no review of this item.

Hintikka, J. and Sandu, G. (1995). The fallacies of the new theory of reference. *Synthese*, 104(2):245–283.
[MR96h:03069] Summary: "The so-called new theory of reference (Marcus, Kripke etc.) is inspired by the insight that in modal and intensional contexts quantifiers presuppose nondescriptive unanalyzable identity criteria which do not reduce to any descriptive conditions. From this valid insight the new theorists fallaciously move to the idea that free singular terms can exhibit a built-in direct reference and that there is even a special class of singular terms (proper names) necessarily exhibiting direct reference. This fallacious move has been encouraged by a mistaken belief in the substitutional interpretation of quantifiers, by the myth of the de re reference, and a mistaken assimilation of 'direct reference' to ostensive (perspectival) identification. The de dicto vs. de re contrast does not involve direct reference, being merely a matter of rule-ordering ('scope').

"The new theorists' thesis of the necessity of identities of directly referred-to individuals is a consequence of an unmotivated and arbitrary restriction they tacitly impose on the identification of individuals."

Nepomuceno Fernández, Á. (1996). Intensional Fregean models. In *Truth: logic, representation and the world (Spanish) (Santiago de Compostela, 1996)*, pages 201–209. Univ. Santiago de Compostela, Santiago de Compostela.

[MR97m:03017] Summary: "Extensional Fregean models are a very interesting way to study some linguistic structures. In this paper an intensional version of them is given. First, we define a second-order language with λ-abstraction. Next, we define an intensional frame by means of a family of domains (nonempty domains with predicates and relations defined over them), a base for a Post algebra, a set of indices (contexts of use), a set of functions (from that set of indices to the family of domains) and a special function (from the family of domains to the first domain). Then an intensional Fregean model is viewed as an intensional frame and an interpretation of the parameters and relators of the language is given. Finally, some propositions are stated." For the entire collection see MR 97k:03005.

Carpenter, B. (1997). *Type-logical semantics.* MIT Press, Cambridge, MA. A Bradford Book.

[MR1638879] A review for this item is in process.

Desclés, J.-P. (1997). Logique combinatoire, types, preuves et langage naturel. In *Introduction aux logiques non classiques*, pages 91–160. Univ. Neuchâtel, Neuchâtel.

[MR1611240] This item will not be reviewed individually. For details of the collection in which this item appears see 98k:03007.

3 Index of authors

X

Y

Z

4 Index of titles