Lecture Notes

LOGIC, LANGUAGE AND COMPUTATION

Volume 1

edited by
Jerry Seligman
&
Dag Westerståhl

CSLI Publications
CENTER FOR THE STUDY OF
LANGUAGE AND INFORMATION

CSLI
Lecture Notes
No. 58

LOGIC, LANGUAGE AND COMPUTATION

Volume 1

edited by
Jerry Seligman
&
Dag Westerståhl

CSLI Publications
CENTER FOR THE STUDY OF
LANGUAGE AND INFORMATION
STANFORD, CALIFORNIA

5

Logical Aspects of Combined Structures

PATRICK BLACKBURN AND MAARTEN DE RIJKE

Abstract

This is an exploratory paper about combining structures. Typically when one applies logic to such areas as computer science, artificial intelligence or linguistics, one encounters hybrid ontologies. The aim of this paper is to identify plausible strategies for coping with such ontological richness.

Introduction

This is an exploratory paper about combining structures. The need for various such combinations has come up in many areas, including computer science (Aceto 1992, Montanari et al. 1993), artificial intelligence (Hobbs 1985), linguistics (Blackburn et al. 1993, 1994a), philosophy (Seligman and Barwise 1993) and logic itself (Kracht and Wolter 1991, de Rijke 1993).

The aim of this paper is to identify the issue of combining structures (and of combining logics and theories, for that matter) as a new research line. We do this as follows. We first present a list of examples in Section 1. In Section 2 we introduce a very simple framework for combining structures using so-called *trios*; briefly, a trio is a triple consisting of a two classes of structures and a collection of links between them. We give examples of theories of specific trios, and we discuss how properties of structures that are combined into trios, transfer — or don't transfer — to the trio. Section 3 concludes the paper with a discussion of further questions.

A final introductory remark: this paper is a preliminary report of ongoing work; a fuller account will be given in (Blackburn and de Rijke 1994).

Logic, Language and Computation.
Jerry Seligmann and Dag Westerståhl, eds.

5.1 Examples

In this section we present examples. We focus on combining structures, but much of what we will say below can be couched in terms of combining logics or theories. We start with two simple examples of what we call *refinement semantics* in which one ontology is given additional structure at the atomic level by other ontologies; we then move on to the richer *classification semantics* in which one structure classifies the elements of another structure by inducing an equivalence relation on it. Finally we consider *fully interacting structures*, where there is no restriction on the relation between the structures being combined.

Generalized Phrase Structure Grammar

Our first example of a refinement semantics stems from generative linguistics: the Generalized Phrase Structure Grammar of Gazdar *et al.* (1985) views linguistic structure as a combined ontology, namely finite trees fibered over finite feature structures, that is: finite trees such that to every node in the tree is associated a finite labeled transition system in which every transition relation $\xrightarrow{a}$ is a partial function.

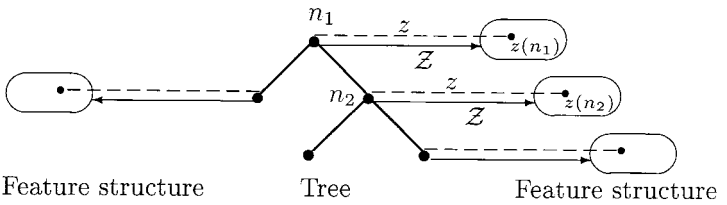

FIGURE 1 Linguistic structure in GPSG

In GPSG the feature structures are used to refine the notion of grammatical category. In contrast to the usual practice in formal language theory where the nodes of parse trees are decorated with 'indivisible' information about categories (for example NP for Noun Phrase or VP for Verb Phrase), GPSG splits the atom: an NP is now a structured object, a feature structure, that contains information about various subatomic features and values.

Finite trees fibered over finite feature structures provide a semantics for two distinct languages: a tree language $\mathcal{L}^T$ that moves us around the tree, and a feature language $\mathcal{L}^F$ that allows us access to the inner structure of grammatical categories. The central ideas of GPSG can then be expressed in a mixture of the two languages called $\mathcal{L}^T(\mathcal{L}^F)$ — the language $\mathcal{L}^T$ *layered over* the language $\mathcal{L}^F$ — in which the $\mathcal{L}^F$ wffs are viewed as the atomic wffs of $\mathcal{L}^T$. A wff ϕ in the layered language $\mathcal{L}^T(\mathcal{L}^F)$ is evaluated as follows: in general ϕ contains $\mathcal{L}^T$ connectives that move us around the tree until we hit what used to be the atomic level; instead of invoking an assignment or

valuation at this stage we have further work to do: we *zoom in* from the tree node n to the associated feature structure $\mathcal{Z}(n)$ and start evaluating at $z(n)$.

Refinement is a very simple way of combining structures; the interaction between the components is limited — nodes in the feature structure, for example, simply don't have permission to access the tree structure. This restriction has a number of pleasant consequences; it's usually fairly straightforward to combine completeness and decidability results for the component logics into completeness and decidability results for the layered language (cf. Section 3 below).

Action Refinement in Process Theory

The previous example involving GPSG concerned refinement of states. In the present example we consider refinement of *actions* or *transitions*. In the top-down design of distributed systems one uses actions and states on an abstract level to represent complex processes on a more concrete level, leading naturally to refinement of states (as in the earlier GPSG example), and of actions (Aceto 1992).

Consider the design of an input device, repeatedly reading data and sending it off. A first, and highly abstract description is given in Figure 2.

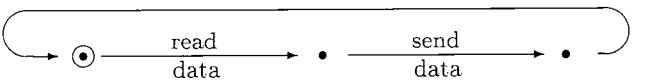

FIGURE 2 An input device

On a slightly less abstract level of description the action 'read data' decomposes into 'prepare reading' and 'carry out reading.' This corresponds to Figure 3:

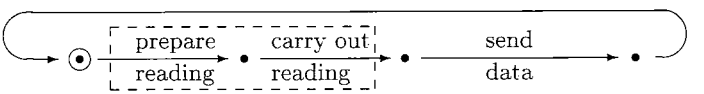

FIGURE 3 The input device refined

This is a very simple kind of refinement of actions: it just refines by a sequence of actions. In general more sophisticated types of refinement may be needed; one can think of refinement by parallel actions, or by infinite processes. This is best formulated as a form of substitution of structures in the following manner. Let r be a function from the (atomic) actions of a labeled transition system $\mathfrak{T}$ to rooted transition systems. The *refinement of* $\mathfrak{T}$ *by* r is the structure that is obtained as follows. For $s \xrightarrow{a} t$ an

edge in $\mathfrak{T}$ let $r(a)'$ be a new copy of $r(a)$; identify s with the root of $r(a)$, identify t with all end nodes of $r(a)$, and remove the edge $s \xrightarrow{a} t$. In other words, instead of making an a-transition at s, we now start at the root of $r(a)'$, traverse a terminating path through $r(a)'$, and then continue from t onwards.

In passing, it's quite natural to look systematically at the converse of refinement: *abstraction*. One could take a structure $\mathfrak{A}$ to be an abstraction of a structure $\mathfrak{B}$ if $\mathfrak{A}$ is the quotient of $\mathfrak{B}$ under an appropriate notion of morphism (see Hobbs (1985) for an example use of abstraction in AI). A general approach would allow for refinements/abstractions over any kind of item in ones structures simultaneously.

Lexical Functional Grammar

In many applications where structures or logics need to be combined, more complex interactions are required than refinements; *classifications* provide an important example of such combinations. To explain these, and to see how classifications are different from refinements, it's best to return to generative grammar; more specifically, we will look at Lexical Functional Grammar (LFG) (Kaplan and Bresnan 1982). Like GPSG, LFG views syntactic structure in terms of composite entities made from finite trees and finite feature structures, but it glues these together rather differently. The basic picture is the one given in Figure 4.

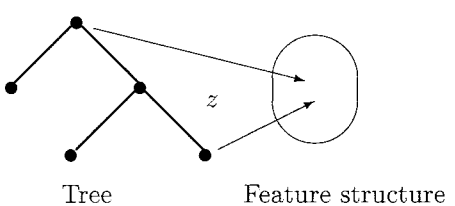

FIGURE 4 Linguistic structure in LFG

Here we have a single finite tree and a single finite feature structure linked by a partial function z. This feature structure induces a classification of tree nodes via z in the following sense. According to LFG, sentences embody two levels of structure: constituent structure, which is represented by a tree, and grammatical relations, represented by a feature structure. Then, two tree nodes are identified, or classified as 'being functionally the same,' if they are mapped onto the same point in the feature structure.

Note that this is *not* the same as refinement of atomic information, rather it's about ensuring that the internal structures of the two ontologies correctly 'match' each other. LFG enforces the required matching using

phrase structure rules annotated with equations. For example, $\uparrow = \downarrow$ means that if we move up the tree from a node t, and then zoom in to the feature structure, we arrive at the same point we would have reached by zooming in directly from t.

Channel Theory

Another area where the idea of using one structure to classify the objects of another is Situation Semantics. Situation semantics has long emphasized the importance of ontological diversity, and one branch where this is put forward very elegantly is the *channel theory* initiated by Jerry Seligman (1990).

As part of a general attempt to model laws or regularities, and information flow, a classification is defined as a triple $\mathbf{A} = \langle tok(\mathbf{A}), typ(\mathbf{A}), : \rangle$, where $tok(\mathbf{A})$ and $typ(\mathbf{A})$ are non-empty sets (of tokens and types, respectively), and : is the *classification relationship* between tokens and types (see Figure 5).

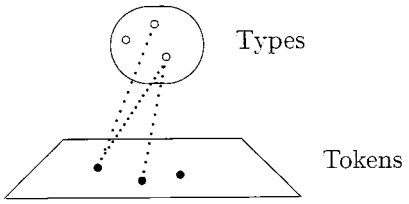

FIGURE 5 A classification

Here the types of $\mathbf{A}$ classify the tokens of $\mathbf{A}$, and the types induce a natural equivalence relation $\sim$ of indistinguishability on tokens: $a \sim b$ iff for all types α we have $a : \alpha$ iff $b : \alpha$. As with LFG and its annotated phrase structure rules, further restrictions may be imposed on the way types and tokens interact.

In channel theory classifications are not considered in isolation. A further 'stacking' of structures occurs when classifications are combined into so-called channels to model information flow. A channel is something which directs information flow between classifications. This is achieved as follows. First, a notion of information preserving morphisms between classifications $\mathbf{A}$ and $\mathbf{B}$ is defined as a certain kind of bi-function $f : \mathbf{A} \rightrightarrows \mathbf{B}$. Then, a channel $\mathbf{C} : \mathbf{A} \Longrightarrow \mathbf{B}$ is a classification $\mathbf{C}$ together with morphisms $\text{left}_C : \mathbf{C} \rightrightarrows \mathbf{A}$ and $\text{right}_C : \mathbf{C} \rightrightarrows \mathbf{B}$.

Roughly, the tokens of $\mathbf{C}$ are used to model connections between the tokens of $\mathbf{A}$ and the tokens of $\mathbf{B}$, and the types of $\mathbf{C}$ are used to express constraints between the types of $\mathbf{A}$ and the types of $\mathbf{B}$; and a connection is

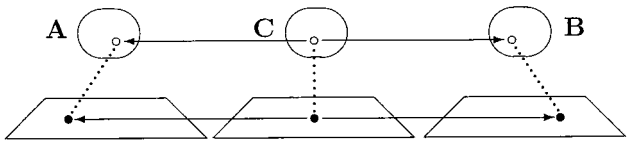

FIGURE 6 A channel

classified by a constraint just in case information flows along the connection in a way that conforms to the constraint.

Full Interaction: Fibering

In an essential way the four examples of combining structures or logics given so far all involve only *one way* traffic between structures: objects in one structure convey information about objects in another structure. In a number of recent talks and papers Dov Gabbay has advocated the idea of *fibering* two sets of semantic entities over each other (Gabbay 1991). Roughly, a fibered structure consists of two classes of models, each class with its own language, plus a function between the classes that tells you how to evaluate formulas belonging to the one language inside structures of the other.

To make this more concrete, here is an example: we fiber finite trees and finite equivalence relations; for the sake of this example we assume that we have two mono-modal languages, $\mathcal{L}_T$ for talking about trees, and $\mathcal{L}_E$ for talking about equivalence relations.

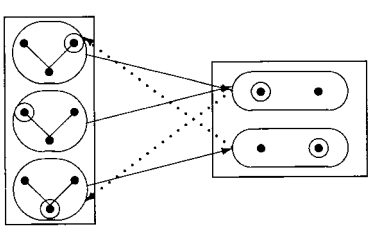

FIGURE 7 Fibering a tree and an equivalence relation

First of all, let a *model-state* pair be a pair $(\mathfrak{M}, s)$ where $\mathfrak{M}$ is a model based on a finite tree or on a finite equivalence relation, and s is an element of $\mathfrak{M}$. Second, let M_T, M_E be non-empty sets of model-state pairs whose first component is a finite tree or a finite equivalence relation, respectively, and such that if $(\mathfrak{M}, s) \in \mathsf{M}_T \cup \mathsf{M}_E$ and $s' \in \mathfrak{M}$, then $(\mathfrak{M}, s') \in \mathsf{M}_T \cup \mathsf{M}_E$. Now, for the fibering function, let F be a pair of functions (F_T, F_E) with $F_T : \mathsf{M}_T \to \mathsf{M}_E$ and $F_E : \mathsf{M}_E \to \mathsf{M}_T$ such that model-state pairs

that are mapped onto each other agree on all atomic symbols common to both languages; the fibering function regulates the interaction between the classes of structures M_T and M_E. Finally, for F a fibering function, the F-fibered structure over M_T and M_E is the triple (W_F, R_F, V_F) such that

- W_F is $\mathsf{M}_T \cup \mathsf{M}_E$,
- R_F is $\{ ((\mathfrak{M}_1, s_1), (\mathfrak{M}_2, s_2)) : \mathfrak{M}_1 = \mathfrak{M}_2 \text{ and } Rs_1s_2 \}$,
- V_F is simply the union of the component valuations.

As to the evaluation of complex formulas, tree formulas are interpreted in M_T as usual, and likewise for $\mathcal{L}_E$-formulas and M_E. If we hit a tree formula while evaluating in M_E, we apply the fibering function F to the current model-state pair, and continue evaluating in its associated model-state pair in M_T; a similar move is made when we hit an $\mathcal{L}_E$ subformula while evaluating in M_T.

We should point out that more involved definitions of fibering (or similar constructions) have been proposed in the literature (Eiben et al. 1992,Gabbay 1991, Goguen and Burstall 1984). For our purposes the definition given here suffices.

Much contemporary research in logic is strongly influenced by applications — and not merely the traditional applications in philosophy or mathematics. Instead, new interdisciplinary work in such areas as Cognitive Science, Artificial Intelligence and Theoretical and Computational Linguistics is the focus of attention. This broadening of the scope of applied logic forces the logician to take ontological diversity seriously, and emphasizes the need for investigations such as the present one.

5.2 Trios

In this section we present a first pass at a mathematical framework for combining structures.

Definition 1 Let A and B be two classes of structures, and let Z be a collection of relations between the elements of A and those of B. Then the triple $(\mathsf{A}, \mathsf{Z}, \mathsf{B})$ is called a *trio*. The classes A and B are called the *left* and *right continents*, respectively, of the trio, and Z is called its *bridge*.

As an example, the trios in GPSG style refinement consist of a single tree $\mathfrak{A}$ as their left continent, a right continent consisting of $|\mathfrak{A}|$ many structures $\{ \mathfrak{B}_a : a \text{ in } \mathfrak{A} \}$, and a bridge consisting of an injective function linking each point of $\mathfrak{A}$ to an element of the right continent.

Of course, the general notion of a trio will only lead to useful and interesting theorizing when we refine it. Such refinements can be pursued along at least two lines. First of all we can try to develop the systems theory of trios.

How do they combine? What kind of structure do they form? It seems that 2-categories (Street 1987) will provide a natural setting for understanding trios at a very abstract level. This kind of question is left to a separate paper.

Here we pursue a second line of questions that comes up in connection with trios: logical issues. We will touch upon logics of specific trios, the analysis of specific bridges, and the classification of bridges.

Logics of specific trios. Consider a *bisimilar* trio $(\mathfrak{A}, \leftrightarrows, \mathfrak{B})$ where $\mathfrak{A}$, $\mathfrak{B}$ are labeled transition systems with transition relations $\xrightarrow{a}$ and $\xrightarrow{b}$, respectively, and $\leftrightarrows$ is a bisimulation between $\mathfrak{A}$ and $\mathfrak{B}$; that is, $\leftrightarrows$ is a non-empty relation on $\mathfrak{A} \times \mathfrak{B}$ that only relates points with the same atomic information and that satisfies a back-and-forth condition: if $x, y \in \mathfrak{A}$, $x' \in \mathfrak{B}$, $x \xrightarrow{a} y$ and $x \leftrightarrows x'$, then there is a $y' \in \mathfrak{B}$ such that $x' \xrightarrow{b} b'$ and $y \leftrightarrows y'$ (and likewise in the opposite direction). We also assume that $\mathfrak{A}$ comes with a mono-modal language $\mathcal{L}(\langle a \rangle)$, and $\mathfrak{B}$ with a language $\mathcal{L}(\langle b \rangle)$.

A first decision we have to make is: what language do we use to talk about such bisimilar trios? Given that we have components $\mathfrak{A}$, $\mathfrak{B}$ and $\leftrightarrows$, the natural set-up has two constants `left` and `right` to denote $\mathfrak{A}$ and $\mathfrak{B}$ respectively, and four diamonds $\langle a \rangle$, $\langle b \rangle$ and $\langle z \rangle$, $\langle z^{-1} \rangle$, where $\langle z \rangle$, $\langle z^{-1} \rangle$ let us move back-and-forth between the two continents $\mathfrak{A}$ and $\mathfrak{B}$, that is: they are interpreted using the bisimulation relation $\leftrightarrows$.

An obvious question is: what is the logic of bisimilar trios? — We need at least the axioms and rules of inference of the minimal modal logic **K** for each of $\langle a \rangle$, $\langle b \rangle$, $\langle z \rangle$ and $\langle z^{-1} \rangle$. In addition the following axioms should be added:

- the well-known axioms from temporal logic stating that the interpretation of $\langle z \rangle$ is the converse of the interpretation of $\langle z^{-1} \rangle$;
- $\mathtt{left} \vee \mathtt{right}$ and $\neg(\mathtt{left} \wedge \mathtt{right})$ to force every point to live in exactly one continent;
- $\phi \leftrightarrow (\phi \wedge \mathtt{left})$ for all $\mathcal{L}(\langle a \rangle)$ formulas, and likewise with `right` and $\mathcal{L}(\langle b \rangle)$ formulas, to force the interpretation of `left` and `right` to be an $\mathcal{L}(\langle a \rangle)$ model and an $\mathcal{L}(\langle b \rangle)$ model, respectively;
- $\langle z \rangle \phi \rightarrow \mathtt{left} \wedge \langle z \rangle(\mathtt{right} \wedge \phi)$ and $\langle z^{-1} \rangle \phi \rightarrow \mathtt{right} \wedge \langle z^{-1} \rangle(\mathtt{left} \wedge \phi)$ to force the interpretation of $\langle z \rangle$ to be a subset of 'the interpretation of `left` $\times$ the interpretation of `right`';
- $\mathtt{left} \wedge p \rightarrow [z]p$ and $\mathtt{right} \wedge p \rightarrow [z^{-1}]p$ (*for p atomic*!), to force the condition on atomic information;
- $\langle a \rangle \phi \rightarrow [z]\langle b \rangle \langle z^{-1} \rangle \phi$ and $\langle b \rangle \phi \rightarrow [z^{-1}]\langle a \rangle \langle z \rangle \phi$, to force the back-and-forth conditions.

Theorem 1 *The above set of axioms and rules completely axiomatizes validity of bisimilar trios.*

The proof of the theorem is a canonical model construction; as the requirement that $\leftrightarrows$ be non-empty is not expressible we may have to tinker somewhat with the canonical model — but this can be done using standard techniques from modal logic.

Analyzing specific bridges. We now move on to a slightly more general *genre* of question. Fix a kind of bridge, and let Z be a bridge of that kind — what do we know about completeness, decidability, complexity ... of the trios $(\mathsf{A}, \mathsf{Z}, \mathsf{B})$, given that we have complete, decidable ... theories for the continents? Here are a few examples.

Finger and Gabbay (1992) prove some general transfer results for a special form of the notion of refinement that we discussed in Section 2. They show how to add a temporal dimension to a logic system, or in our terms: they take temporal logic with *Since* and *Until* over the natural numbers as the 'top language', and refine the atomic information of that language, using any other language as the 'bottom language'. The results Finger and Gabbay establish include that, provided the bottom language has a complete axiomatization, the combined language has one as well; and, provided the bottom language is decidable, so is the combined one. In the full paper we enhance and generalize these transfer results in a number of ways. First, we also consider transfer and non-transfer of complexity results. Second, we show that the Finger and Gabbay results remain valid when other (one-dimensional) top languages are used instead of *Since*, *Until* logic. And third, to capture phenomena such as Action Refinement in Process Theory (as discussed in Section 2), we consider transfer problems for top languages whose formulas are interpreted at semantic objects other than single states, including pairs, transitions, and sequences.

As a second example, following their introduction in the formal semantics of natural language, Shehtman (1978) considers the *Cartesian product* of two modal logics. For instance, the intended frames of the Cartesian product of the modal logics **S4** and **S5** consists of structures whose universe is a product $U_0 \times U_1$ with a pre-order on U_0 and an equivalence relation on U_1. An important question here is to determine in which cases $\mathbf{L}(\mathfrak{F}_1 \times \mathfrak{F}_2) = \mathbf{L}(\mathfrak{F}_1) \times \mathbf{L}(\mathfrak{F}_2)$, that is, when does the logic of the product coincide with the product of the component logics? Shehtman (1978) provides a partial answer. Another important example of a similar 'simple' combination of structures arises when we consider so-called independent joins of logics. For instance, the independent join of two mono-modal logics $\mathbf{L}_1$ and $\mathbf{L}_2$ with distinct modal operators $\langle a \rangle$ and $\langle b \rangle$, respectively, is simply the union of the two logics. On the level of structures this operation amounts to considering structures $(W, \xrightarrow{a}, \xrightarrow{b})$ that have reducts living in the language of $\mathbf{L}_1$ and in the language of $\mathbf{L}_2$. Kracht and Wolter (1991) show that the independent join of two complete or decidable logics is again

complete or decidable. And Spaan (1993) completely classifies the complexity of the independent join of two modal logics in terms of the complexity of the component logics; she also analyzes the much more difficult situation in which the component logics are not fully independent. In the full paper we present further results along these lines.

Classifying bridges. The final of type of question we want to mention is still more general; it can roughly be summarized as: what kind of bridges are there? Questions like this serve a dual purpose: on the one hand they spring from a desire to find sensible ways of cutting up the universe of all trios (in Section 2 we only considered three kinds of trios: refinements, classifications and full interactions); on the other hand having available a taxonomy of trios and bridges may help us in obtaining generalizations — and thus in gaining a better understanding — of the results obtained so far. For example, this may help us to understand why refinements and independent joins behave nicely. These issues are the focus of our ongoing technical investigations.

5.3 Discussion

At both the technical and conceptual level there is much obvious work to do. For example, the completeness result for bisimilar trios is just a pointer to further results; Blackburn and de Rijke (1994) axiomatize other logics of specific trios, and indeed it is possible to state and prove general completeness results for trios in the spirit of Sahlqvist's Theorem.

It seems hard to state general results and properties of combined structures without moving to a very abstract mathematical framework. As has already been mentioned, to understand the systems theory of trios at a general level, we feel that 2-categories may be useful. However, for particular kinds of trios dedicated system theories can be much more appropriate; channel theory as a theory of clasification structures provides an example.

To conclude the paper let us consider a very obvious weakness of the story we have told so far; we have acted as if combined ontologies are lifeless, *static* entities. This ignores the fact that for many applications it is precisely the *dynamic* aspects of combined ontologies that are of interest. To make matters more concrete, we revert to generative grammar. Consider Tree Adjoining Grammars (tags) (Joshi et al. 1975). Tag analyses are essentially dynamic; sentences are viewed as the result of merging trees together. To gain something of the flavour of tags in action, consider the operation known as *adjunction*. Let τ be a tree with an internal node labeled by the nonterminal symbol A. Let ρ be an auxiliary tree with root and foot node labeled by the same nonterminal symbol A. The tree τ' that results by adjoining ρ at the A-labeled node in τ is formed by removing the subtree of τ rooted at this node, inserting ρ in its place, and substituting it

at the foot node of ρ. Perhaps the most important thing to notice is the role played by the node labeled A. We began with an initial structure (namely τ) with a designated node (namely that labeled A); we then performed a computation step; and this created a larger structure with a new designated node, the site for further creation. Of course, all this *could* be described statically. But to do so does violence to the underlying intuitions. We need analyses which cope with the growth of structures rather than merely treating them as completed objects.[1] This idea brings us to territory already explored by much of the literature on feature logic (Carpenter 1992), on evolving algebras (Gurevich 1991) and on specification languages (Groenboom and Renardel de Lavalette 1994). Ultimately this seems to require investigations of 'imperative logics', that is, logics that write to models rather than treating them as read-only structures; see Blackburn, de Rijke and Seligman (1994b) for some preliminary investigations.

Acknowledgments

Both authors would like to thank the Netherlands Organization for Scientific Research (NWO), project NF 102/62-356 'Structural and Semantic Parallels between Natural Languages and Programming Languages' for financial support.

References

Aceto, L. 1992. *Action Refinement in Process Algebras.* Cambridge: Cambridge University Press.

Blackburn, P., and M. de Rijke. 1994. Zooming In, Zooming Out. Technical report. CWI, Amsterdam. To appear in *Journal of Logic, Language and Information.*

Blackburn, P., M. de Rijke, and C. Gardent. 1994a. Back and Forth through Time and Events. In *Temporal Logic (ICTL '94)*, ed. D.M. Gabbay and H.J. Ohlbach. 225–237. LNAI 827. Berlin: Springer.

Blackburn, P., M. de Rijke, and J. Seligman. 1994b. Modal Logics with Memory. Unpublished manuscript.

Blackburn, P., C. Gardent, and W. Meyer Viol. 1993. Talking about Trees. In *Proceedings 6th EACL.* 21–29. Utrecht.

Carpenter, B. 1992. *The Logic of Typed Feature Structures.* Cambridge: Cambridge University Press.

de Rijke, M. 1993. What is Modal Logic? In *Arrow Logics and Multi-Modal Logics*, ed. L. Pólós, M. Masuch, and M. Marx. Oxford: Oxford University Press. To appear.

[1] Although this is precisely what domain theory sets out to do, it remains to be seen whether domain theoretic tools are the most suitable setting for the applications considered in this paper.

Eiben, A., A. Jánossy, and Kurucz. Á. 1992. Combining Logics. Technical Report IR-319. Amsterdam: Department of Mathematics and Computer Science, Free University.

Finger, M., and D.M. Gabbay. 1992. Adding a Temporal Dimension to a Logic System. *Journal of Logic, Language and Information* 1:203–233.

Gabbay, D.M. 1991. Labelled Deductive Systems, 6th intermediate draft. Lecture Notes 3rd ESSLLI Summer School, Saarbrücken.

Gazdar, G., E. Klein, G. Pullum, and I. Sag. 1985. *Generalized Phrase Structure Grammar.* Oxford: Basil Blackwell.

Goguen, J.A., and R.M. Burstall. 1984. Some Fundamental Algebraic Tools for the Semantics of Computation (I). *Theoretical Computer Science* 31:175–209.

Groenboom, R., and G. Renardel de Lavalette. 1994. Reasoning about Dynamic Features in Specification Languages. Unpublished manuscript.

Gurevich, Y. 1991. Evolving algebras; a tutorial introduction. *Bulletin EATCS* 43:264–284.

Hobbs, J. 1985. Granularity. In *Proceedings 9th IJCAI. Vol. 1*, 432–435.

Joshi, A., L. Levy, and M. Takahashi. 1975. Tree Adjunct Grammars. *Journal of Computer and Systems Sciences* 10.

Kaplan, R., and J. Bresnan. 1982. Lexical-Functional Grammar: A Formal System for Grammatical Representation. In *The Mental Representation of Grammatical Relations*, ed. J. Bresnan. 173–281. Cambridge, MA: MIT Press.

Kracht, M., and F. Wolter. 1991. Properties of Independently Axiomatizable Bimodal Logics. *Journal of Symbolic Logic* 56:1469–1485.

Montanari, A., E. Ciapessoni, E. Corsetti, and P. San Pietro. 1993. Dealing with Time Granularity in Logical Specifications of Real-Time Systems. Technical report. Department of Mathematics and Computer Science, University of Udine.

Seligman, J. 1990. *Perspectives: a Relativistic Approach to the Theory of Information.* Doctoral dissertation, University of Edinburgh.

Seligman, J., and J. Barwise. 1993. Channel Theory: Toward a mathematics of imperfect information flow. Unpublished manuscript.

Shehtman, V. 1978. Two-Dimensional Modal Logic. *Matematicheskie Zametki* 23:759–772. (In Russian).

Spaan, E. 1993. *Complexity of Modal Logics.* Doctoral dissertation, ILLC, University of Amsterdam.

Street, R. 1987. The algebra of oriented simplexes. *Journal of Pure and Applied Algebra* 49:283–335.