Curriculum Vitae of C.A. Middelburg

Personal data

name:	Cornelis Adam Middelburg;
date of birth:	22 February 1948;
place of birth:	Rotterdam, the Netherlands;
nationality:	Netherlands.

Professional background

Education:

M.Sc. in physics, Delft University of Technology, 1971,
thesis title: *Hybrid Simulation of Queueing Problems*;
Ph.D. in mathematics & computer science, University of Amsterdam, 1990,
thesis title: *Syntax and Semantics of VVSL.*

Experience:

research and development in computer science, mainly in the following areas: systems software, relational databases, programming languages, compiler construction, formal methods, concurrency theory, theory of programs, theory of number systems;

teaching in computer science;

management.

Positions:

scientific staff member at KPN Research, December 1971 – December 1995;
part-time full professor of applied logic at Utrecht University, June 1992 – May 2001;
senior research fellow at UNU/IIST, January 1996 – December 1997;
scientific staff member at KPN Research, January 1998 – April 1998;
research fellow at CWI, May 1998 – September 1998;
research fellow at Eindhoven University of Technology, October 1998 – February 2003;
research fellow at Utrecht University, June 2001 – February 2003;
part-time full professor of computer science at Eindhoven University of Technology, April 2003 – March 2007;
senior researcher at the University of Amsterdam, December 2005 – November 2009;
guest researcher at the University of Amsterdam, December 2009 – . . . .

Particulars of my main past research and development activities

The following are particulars about my main research and development activities in the past:

Relational databases I have designed an experimental relational database management system for the PDP-11 mini computers. This design has much in common with the design of the well-known System R which was independently developed at about the same time for IBM main frames. I have also developed an extremely space efficient algorithm for the evaluation of database queries expressed in the relational calculus, which was, about 45 years ago, my first formal software development activity. I have specified the query evaluation problem and verified the correctness of the algorithm using classical first-order logic and elementary set theory.

Programming languages and compiler construction I have been involved in the design of the programming language CHILL and the construction of a compiler for it. One of my main contributions to the compiler construction have been the detailed design of the translation phase of the compiler by means of VDM. I have devised a new approach where code generation is driven by simulation of program execution on an abstract machine. The starting-point of the rigorous design of the translation phase was the formal definition of CHILL to which I have contributed as well. The design turned out to be highly reusable when the compiler had to be adapted to new target machines.

Formal methods (VVSL) I have designed a specification language, called VVSL, which extends the language used in the software development method VDM with features for specifying interfering operations and modular structuring of specifications. The approach to the specification of interfering operations encompasses various other approaches. VVSL is probably the only language for writing modularly structured VDM specifications with a sound mathematical basis for its modular structuring features. I have also used VVSL to formalize the relational database model and a model of transaction management in database systems. Many formerly unmentioned assumptions in most work on databases have been made explicit in these formalizations.

Formal methods (SDL) I have worked on various topics aimed at achieving advances in the area of analysis of systems described using SDL, a specification language widely used in telecommunications. My work on a new semantics of a fragment of SDL, based on a process algebra with discrete relative timing, is most known. Unlike in related work, the chosen fragment covers all behavioural aspects of SDL and the proposed semantics agrees with the official semantics as far as possible.

Concurrency theory I have made a systematic study of issues relevant to dealing with timing in process algebra. This work has, first of all, resulted in a coherent collection of four process algebras, each dealing with timing in a different way. Another result of my work on process algebra with timing is a process algebra for hybrid systems which has considerably more potential with regard to description and analysis than the ones proposed earlier. My model-theoretic investigation of first-order extensions of process algebras sets an interesting new direction for future research in the field of process algebra. I have also developed an imperative process algebra and used it to formalize versions of existing models of computation based on (sequential or parallel) random access machines and time and work complexity measures for those models.

Theory of programs I have investigated issues concerning the following subjects from the viewpoint that a program primarily represents an instruction sequence: programming language semantics, programming language expressiveness, computability, computational complexity, algorithm efficiency, algorithmic equivalence of programs, program performance, program compactness, program parallelization, and program verification. This work demonstrates that the notion of instruction sequence is relevant to diverse subjects from computer science.

Summary of my other past research and development activities

Most of my other research and development activities in the past are summarized below.

- In the area of *systems software*:
  - the development of a macro processor for the macro language TRAC;
  - the development of a run-time system for the programming language BCPL.
- In the area of *software reusability*:
  - a study of ways to formally specify general-purpose software components and to build up and interrogate libraries of formally specified components.
- In the area of *programming techniques*:
  - a study of the merits of parallel programming, functional programming, and logic programming.

- In the area of *formal methods*:
  - a comparison of the three-valued logic of partial functions LPF used with VDM and the two-valued logic of partial functions MPL_ω used with COLD;
  - the development of a complete proof system for the typed version of LPF;
  - the development of a common semantic model for SDL, LOTOS and other specification languages used for the development of communications software;
  - the development of a simple variant of action-based computation tree logic, meant for expressing properties of telecommunication services that can be checked with commercially available SDL tools;
  - the development of an approach to explain issues concerning programs from the viewpoint that a program is code that is capable of controlling the behaviour of some machine;
  - a study of characterizing properties of an expansion of the paraconsistent logic LP with an implication connective for which the standard deduction theorem holds and a falsity connective;
  - the development of a natural deduction proof system for the first-order version of this paraconsistent logic, with a logical justification by means of an embedding into first-order classical logic;
  - the development of a sequent calculus proof system for the first-order version of this paraconsistent logic, with an application to consistent query answering in inconsistent databases;
  - a study of various properties of an expansion of first-order Belnap-Dunn logic with an implication connective for which the standard deduction theorem holds and a falsity connective;
  - the development of a minor variation of this logic that can deal with terms with an indeterminate value, with an application to consistent query answering in inconsistent databases with null values.
- In the area of *concurrency theory*:
  - the development of algebras of synchronous and asynchronous dataflow networks with models based on process algebra;
  - the development of an algebra of timed frames, the kind of transition systems used for the operational semantics of process algebra with discrete relative timing;
  - an extension of the customary approach to structural operational semantics using transition system specifications that takes variable binding operators into account;
  - the development of a process algebra in which processes have an implicit computational capital;
  - the development of an algebra of interacting process components that comprise an interface and a process as considered in process algebra;
  - the development of thread algebra, a specialized process algebra dealing with threads, as found in programming languages such as Java and C#, and interleaving strategies for threads;
  - an extension of thread algebra to distributed multi-threaded programs that are subjected to load balancing;
  - an extension of thread algebra with probabilistic features;
  - the adaptation of a process algebra in which processes have their state to some extent visible by means of propositions to tolerance of self-contradictory states;
  - the development of a probabilistic process algebra that supports both arbitrary interleaving and interleaving according to some process-scheduling policy;
  - a study of the use of Hoare logic in a process algebra setting;
  - the development of an imperative process algebra with abstraction, with an example of a possible use of this imperative process algebra in the area of information-flow security analysis;

  - a study of the role that this imperative process algebra can play in describing models of parallel computation and complexity measures for them;
  - the elaboration of a variant of the standard notion of branching bisimilarity for processes with discrete relative timing which is coarser than the standard notion;
  - the development of a process algebra that concerns the timed behaviour of systems with a known spatial distribution and that provides a truly algebraic mechanism for asynchronous communication in space-time.

- In the area of *theory of programs*:

  - the expression of the secure hash algorithm SHA-256 and the Karatsuba multiplication algorithm by finite single-pass instruction sequences;
  - a study of instruction sequence size bounded functional completeness of instruction sets for Boolean registers;
  - the development of an axiom system for behavioural congruence of single-pass instruction sequences;
  - a study of the complexity of the problem of deciding whether a single-pass instruction sequence correctly implements the non-zeroness test;
  - a study of instruction sequences by which Turing machines can be simulated;
  - a study of instruction sequences by which random access machines can be simulated, with a proposal for a semi-realistic version of the random access machine model of computation.

- In the area of *dynamic data structures*:

  - an extension of thread algebra with features of data linkage dynamics, a simple model of computation which bears on the use of dynamic data structures in programming;
  - the explanation of data linkage dynamics by means of a term rewriting system with rule priorities;
  - a study of automatic removal of links in dynamic data structures as soon as they will not be used once again by the program, in the setting of data linkage dynamics.

- In the area of *processor architecture*:

  - the modelling of micro-architectures with pipelined instruction processing;
  - an study of the merits of program parallelization for speeding up instruction processing;
  - a study of how the transformations on the states of the main memory of a load/store architecture that can be achieved depend on the operating unit size.

- In the area of *operating systems*:

  - a study of how a definition of a theoretical concept of an operating system, suitable to be incorporated in a mathematical theory of operating systems, could look like.

- In the area of *theory of number systems*:

  - a study of meadows, which are roughly fields in which the multiplicative inverse of zero is zero, their partial variants, and logics that may be used when working with partial meadows;
  - a study of non-involutive meadows, which are meadows in which the multiplicative inverse of zero is a number different from zero;
  - a study of which meadows admit transformation of fractions into fractions of which neither the numerator nor the denominator contains a fraction.

- In the area of *quantitative finance*:

  - the formalization of a cumulative interest compliant conservation requirement for pure financial products in timed tuplix calculus, a specialized process algebra for financial behaviours;

  – a preparatory exploration of issues concerning the form of finance that the avoidance of interest gives rise to in Islamic finance.

Current research interests

The following are my current research interests:

- The description and analysis of hybrid systems. In continuation of my previous work on a process algebra for hybrid systems and its connections with the formalism of hybrid automata, I am among other things interested in efficient proof techniques for that process algebra, effective procedures for restricted versions of the process algebra to decide whether an equation is derivable, and model checking tools for restricted versions of the process algebra to verify properties expressed in a suitable temporal logic.
- The description and analysis of real-time reactive systems that are networks of dynamically reconfiguring components in which communication between the components is usually in whole or in part coordinated from outside the components. Such systems become dominating among the systems that shape our society. My special interest is in dealing with all aspects relevant to their behaviour, which includes coordination, reconfiguration and real-time reaction, in a single framework.
- The development of more theory in subject areas such as computability, computational complexity, algorithm efficiency, program performance, program compactness, program parallelization, probabilistic computation, and program verification from the viewpoint that a program primarily represents an instruction sequence. For a growing number of developments, including developments with respect to high-performance program execution on classical or non-classical computers and estimation of execution times of hard real-time systems, it becomes increasingly more important that programs are considered at the level of instruction sequence.
- In the past, I was for a long time concerned with formal techniques, and tools in support of them, to complement industrial approaches to software development for more rigorous approaches to software development in industry. I am still interested in this line of research.

Teaching

I have given the following courses and trainings:

- numerous ad hoc internal courses and trainings at KPN on systematic design of database systems, advanced programming techniques, and formal techniques for specification and design in the period 1972 to 1992;
- a course *Formal Specification* for students at Utrecht University in the academic years 1993–1994 and 1994–1995;
- a course *Beyond SDL* for graduates at numerous universities in developing countries, on industrial specification techniques such as SDL and MSC as well as formalisms from the academic community such as process algebra with timing and duration calculus, in the years 1996 and 1997;
- research training for graduates at universities in Indonesia, Pakistan, Brasil and South Africa, using topics aimed at achieving advances in the area of analysis of telecommunication services, in the year 1997;
- a course *Process Algebra with Timing* for graduates at UNU/IIST in September 2001;
- a course *Process Algebra with Timing* for students at Eindhoven University of Technology in the academic years 2002–2003, 2003–2004 and 2004–2005.

I have also prepared the course material of a new course *Process Theory* for first-year students at Eindhoven University of Technology in the academic year 2002–2003.

I have supervised the following M.Sc. students:

- T.H.P.F. Bullens, M.Sc. student at Utrecht University, thesis: *Implementing the Intelligent Network Model on the ToolBus* (1994);
- R.P.G. Brouwer, M.Sc. student at Eindhoven University of Technology, thesis: *Measuring Formal Tools and Methods* (2002).

I have supervised the following Ph.D. students:

- E. Kwast, Ph.D. student at Utrecht University, as daily supervisor, thesis: *Protocol Data Dependencies* (1997);
- S. Andova, Ph.D. student at Eindhoven University of Technology, as promotor, thesis: *Probabilistic Process Algebra* (2002);
- T.A.C. Willemse, Ph.D. student at Eindhoven University of Technology, as daily supervisor, thesis: *Semantics and Verification in Process Algebras with Data and Timing* (2003).

Moreover, I have supervised at UNU/IIST in the academic year 1996–1997 three graduates (R. Şoricuţ, Y.S. Usenko and B. Warinschi) doing research for nine months on topics aimed at achieving advances in the area of analysis of telecommunication services.

I have been member of the examination committee of 5 M.Sc. students and 26 Ph.D. students.

Management

My main managerial tasks are summarized below:

- at KPN Research, I have been the deputy manager of the former Programming Research and Development Group from 1978 till 1983;
- at UNU/IIST, I have managed the Research Group and assisted the director during the years 1996 and 1997.

Miscellaneous matters

Some remaining activities are summarized below:

- consultancy for software development projects within KPN;
- advice for the making of strategic policy of KPN with respect to new developments in software technology;
- participation in the ISO working group concerning the standardization of VDM and Z;
- organization of a workshop on Semantics of Specification Languages;
- drafting proposals for research projects within the framework of European and Dutch research programmes;
- giving invited talks as part of colloquia and workshops organized by professional associations and universities;
- guest editing special issues of journals;
- participation in advisory committees, evaluation committees, program committees, etc.;
- membership of the editorial board of the journal Science of Computer Programming.

From the early eighties till the beginning of 1994, my main duties at KPN Research have been those of a senior scientist:

- doing research;
- motivating younger colleagues;

- initiating new research and development activities;
- watching over the scientific character of the research in the group.

Notice that I have worked at KPN Research mostly in an environment where external publication was not a priority. Actually, there was only room for external publication in 1989 and 1990, when I was writing my Ph.D. thesis. Of course, there was some room for external publication since my appointment, for 20% of a full-time professorship, at Utrecht University in 1992.

Best publications

Among my best publications are the following:

A typed logic of partial functions reconstructed classically
At the time, this paper was among the best papers about proof-theoretical issues concerning VDM. The paper dealt with virtually all outstanding proof-theoretical issues. The complete proof system for typed LPF, the induction rules for recursively defined functions and types, and the meta-rules about induction rules for base types, types constructed by means of type formers and subtypes were all new. The logical justification of all inference rules was new as well, and provides an illuminating explanation of LPF in classical logic.

Discrete time process algebra and the semantics of SDL
At the time, this paper was among the best papers about the semantics of SDL, a specification language widely used in telecommunications. The paper was the only paper on this subject in which the chosen fragment of SDL covers all behavioural aspects of SDL and the proposed semantics agrees with the official semantics wherever the latter gives a definite answer. The main virtue of the proposed semantics is that it brought practically useful advanced tools and techniques for analysis of systems described using SDL within reach.

Process algebra with timing
At the time, this book was among the best books about timing in process algebra. To the best of my knowledge, this book is still the only book on this subject that deals thoroughly with relative and absolute timing on both a discrete time scale and a continuous time scale, and the relations between the process algebras dealing with these different ways of timing. Those relations are of both theoretical and practical interest.

Instruction sequences for computer science
This book is the only book that demonstrates that the concept of an instruction sequence, a forgotten basic concept of computer science, offers a useful viewpoint on issues relating to diverse subjects in computer science. Selected issues relating to well-known subjects from the theory of computation and the area of computer architecture are rigorously investigated in this book thinking in terms of instruction sequences. The book presents the first theory of instruction sequences and provides among other things a new perspective on non-uniform computational complexity and the halting problem.

On the complexity of the correctness problem for non-zeroness test instruction sequences
This paper appears to be the only paper that investigates, within the context of some programming language, under what restrictions it can be efficiently determined whether an arbitrary program solves a particular problem correctly. A very simple assembler-like programming language and a very simple problem concerning sequences of binary digits are considered. The rather deep results of the presented work indicate that, contrary to popular believe, the weakest restriction under which such correctness problems can be efficiently determined is a very strong one.

Imperative process algebra and models of parallel computation

This paper presents, within the setting of an imperative process algebra, formalizations of versions of models of computation based on (sequential) random access machines, asynchronous parallel random access machines, and synchronous parallel random access machines and also formalizations of time and work complexity measures for those models. This paper appears to be the only paper that presents formalizations of versions of models of computation other than models based on Turing machines.

Publications

The abstracts of the published papers and recent technical reports, and the prefaces of the books, are available from http://staff.fnwi.uva.nl/c.a.middelburg/. The published papers and technical reports themselves can easily be found via http://staff.fnwi.uva.nl/c.a.middelburg/.

Books

1. C.A. Middelburg. *Logic and Specification*. Computer Science: Research and Practice, Vol. 1, Chapman & Hall, 1993.
2. L.M.G. Feijs, H.B.M. Jonkers and C.A. Middelburg. *Notations for Software Design*. FACIT Series, Springer-Verlag, 1994.
3. J.C.M. Baeten and C.A. Middelburg. *Process Algebra with Timing*. EATCS Monograph Series, Springer-Verlag, 2002.
4. J.A. Bergstra and C.A. Middelburg. *Instruction Sequences for Computer Science*. Atlantis Studies in Computing, Vol. 2, Atlantis Press, 2012.

Handbook chapters

5. J.C.M. Baeten and C.A. Middelburg. Process algebra with timing: real time and discrete time. In J.A. Bergstra, A. Ponse and S.A. Smolka, editors, *Handbook of Process Algebra*, pages 627–684, Elsevier, 2001.
6. J.A. Bergstra, C.A. Middelburg and Y.S. Usenko. Discrete time process algebra and the semantics of SDL. In J.A. Bergstra, A. Ponse and S.A. Smolka, editors, *Handbook of Process Algebra*, pages 1209–1268, Elsevier, 2001.

Journal papers

7. C.A. Middelburg. VVSL: A language for structured VDM specifications. *Formal Aspects of Computing*, 1(1):115–135, 1989.
8. C.A. Middelburg. Modular structuring of VDM specifications in VVSL. *Formal Aspects of Computing*, 4(1):13–47, 1992.
9. C.A. Middelburg. Specification of interfering programs based on inter-conditions. *Software Engineering Journal*, 7(3):205–217, 1992.
10. C.B. Jones and C.A. Middelburg. A typed logic of partial functions reconstructed classically. *Acta Informatica*, 31(5):399–430, 1994.
11. J.A. Bergstra, W.J. Fokkink and C.A. Middelburg. Algebra of timed frames. *Int. Journal of Computer Mathematics*, 61:227–255, 1996.
12. J.A. Bergstra, C.A. Middelburg and Gh. Ştefănescu. Network algebra for asynchronous dataflow. *Int. Journal of Computer Mathematics*, 65:57–88, 1997.
13. C.A. Middelburg. Truth of duration calculus formulae in timed frames. *Fundamenta Informaticae*, 36(2/3):235–263, 1998.
14. C.A. Middelburg. Variable binding operators in transition system specifications. *Journal of Logic and Algebraic Programming*, 47(1):15–45, 2001.
15. J.C.M. Baeten and C.A. Middelburg. Real time process algebra with time-dependent conditions. *Journal of Logic and Algebraic Programming*, 48(1):1–37, 2001.
16. C.A. Middelburg. Process algebra with nonstandard timing. *Fundamenta Informaticae*, 53(1):55–77, 2002.
17. C.A. Middelburg. Revisiting timing in process algebra. *Journal of Logic and Algebraic Programming*, 54(1/2):109–127, 2003.

18. C.A. Middelburg. An alternative formulation of operational conservativity with binding terms. *Journal of Logic and Algebraic Programming*, 55(1/2):1–19, 2003.
19. J.A. Bergstra and C.A. Middelburg. Located actions in process algebra with timing. *Fundamenta Informaticae*, 61(3/4):183–211, 2004.
20. J.A. Bergstra and C.A. Middelburg. Process algebra for hybrid systems. *Theoretical Computer Science*, 335(2/3):215–280, 2005.
21. J.A. Bergstra and C.A. Middelburg. Continuity controlled hybrid automata. *Journal of Logic and Algebraic Programming*, 68(1/2):5–53, 2006.
22. J.A. Bergstra and C.A. Middelburg. Thread algebra with multi-level strategies. *Fundamenta Informaticae*, 71(2/3):153–182, 2006.
23. C.A. Middelburg. Conditionals in algebraic process calculi. *Electronic Notes in Theoretical Computer Science*, 162:237–241, 2006.
24. J.A. Bergstra and C.A. Middelburg. Splitting bisimulations and retrospective conditions. *Information and Computation*, 204(7):1083–1138, 2006.
25. J.A. Bergstra and C.A. Middelburg. Preferential choice and coordination conditions. *Journal of Logic and Algebraic Programming*, 70(2):172–200, 2007.
26. J.A. Bergstra and C.A. Middelburg. Thread algebra for strategic interleaving. *Formal Aspects of Computing*, 19(4):445–474, 2007.
27. J.A. Bergstra and C.A. Middelburg. A thread algebra with multi-level strategic interleaving. *Theory of Computing Systems*, 41(1):3–32, 2007.
28. J.A. Bergstra and C.A. Middelburg. Maurer computers with single-thread control. *Fundamenta Informaticae*, 80(4):333–362, 2007.
29. J.A. Bergstra and C.A. Middelburg. Synchronous cooperation for explicit multi-threading. *Acta Informatica*, 44(7/8):525–569, 2007.
30. J.A. Bergstra and C.A. Middelburg. Instruction sequences with indirect jumps. *Scientific Annals of Computer Science*, 17:19–46, 2007.
31. J.A. Bergstra and C.A. Middelburg. Programming an interpreter using molecular dynamics. *Scientific Annals of Computer Science*, 17:47–81, 2007.
32. J.A. Bergstra and C.A. Middelburg. Simulating Turing machines on Maurer machines. *Journal of Applied Logic*, 6(1):1–23, 2008.
33. J.A. Bergstra and C.A. Middelburg. Maurer computers for pipelined instruction processing. *Mathematical Structures in Computer Science*, 18(2):373–409, 2008.
34. J.A. Bergstra and C.A. Middelburg. Distributed strategic interleaving with load balancing. *Future Generation Computer Systems*, 24(6):530–548, 2008.
35. J.A. Bergstra and C.A. Middelburg. Parallel processes with implicit computational capital. *Electronic Notes in Theoretical Computer Science*, 209:55–81, 2008.
36. J.A. Bergstra and C.A. Middelburg. Program algebra with a jump-shift instruction. *Journal of Applied Logic*, 6(4):553–563, 2008.
37. J.A. Bergstra and C.A. Middelburg. Instruction sequences with dynamically instantiated instructions. *Fundamenta Informaticae*, 96(1–2):27–48, 2009.
38. J.A. Bergstra and C.A. Middelburg. Machine structure oriented control code logic. *Acta Informatica*, 46(5):375–401, 2009.
39. J.A. Bergstra and C.A. Middelburg. A thread calculus with molecular dynamics. *Information and Computation*, 208(7):817–844, 2010.
40. J.A. Bergstra and C.A. Middelburg. On the operating unit size of load/store architectures. *Mathematical Structures in Computer Science*, 20(3):395–417, 2010.

41. J.A. Bergstra and C.A. Middelburg. An interface group for process components. *Fundamenta Informaticae*, 99(4):355–382, 2010.
42. J.A. Bergstra and C.A. Middelburg. Data linkage dynamics with shedding. *Fundamenta Informaticae*, 103(1–4):31–52, 2010.
43. J.A. Bergstra and C.A. Middelburg. Thread algebra for poly-threading. *Formal Aspects of Computing*, 23(4):567–583, 2011.
44. J.A. Bergstra and C.A. Middelburg. Inversive meadows and divisive meadows. *Journal of Applied Logic*, 9(3):203–220, 2011.
45. J.A. Bergstra and C.A. Middelburg. Thread extraction for polyadic instruction sequences. *Scientific Annals of Computer Science*, 21(2):283–310, 2011.
46. J.A. Bergstra and C.A. Middelburg. Preliminaries to an investigation of reduced product set finance. *Journal of King Abdulaziz University: Islamic Economics*, 24(1):175–210, 2011.
47. J.A. Bergstra and C.A. Middelburg. On the expressiveness of single-pass instruction sequences. *Theory of Computing Systems*, 50(2):313–328, 2012.
48. J.A. Bergstra and C.A. Middelburg. Instruction sequence processing operators. *Acta Informatica*, 49(3):139–172, 2012.
49. J.A. Bergstra and C.A. Middelburg. On the behaviours produced by instruction sequences under execution. *Fundamenta Informaticae*, 120(2):111–144, 2012.
50. J.A. Bergstra and C.A. Middelburg. Indirect jumps improve instruction sequence performance. *Scientific Annals of Computer Science*, 22(2):253–265, 2012.
51. J.A. Bergstra and C.A. Middelburg. A process calculus with finitary comprehended terms. *Theory of Computing Systems*, 53(4):645–668, 2013.
52. J.A. Bergstra and C.A. Middelburg. Data linkage algebra, data linkage dynamics, and priority rewriting. *Fundamenta Informaticae*, 128(4):367–412, 2013.
53. J.A. Bergstra and C.A. Middelburg. Timed tuplix calculus and the Wesseling and van den Berg equation. *Scientific Annals of Computer Science*, 23(2):169–190, 2013.
54. J.A. Bergstra and C.A. Middelburg. Instruction sequence based non-uniform complexity classes. *Scientific Annals of Computer Science*, 24(1):47–89, 2014.
55. J.A. Bergstra and C.A. Middelburg. Division by zero in non-involutive meadows. *Journal of Applied Logic*, 13(1):1–12, 2015.
56. J.A. Bergstra and C.A. Middelburg. On algorithmic equivalence of instruction sequences for computing bit string functions. *Fundamenta Informaticae*, 138(4):411–434, 2015.
57. J.A. Bergstra and C.A. Middelburg. Probabilistic thread algebra. *Scientific Annals of Computer Science*, 25(2):211–243, 2015.
58. J.A. Bergstra and C.A. Middelburg. Transformation of fractions into simple fractions in divisive meadows. *Journal of Applied Logic*, 16:92–110, 2016.
59. J.A. Bergstra and C.A. Middelburg. Instruction sequence size complexity of parity. *Fundamenta Informaticae*, 149(3):297–309, 2016.
60. J.A. Bergstra and C.A. Middelburg. On instruction sets for Boolean registers in program algebra. *Scientific Annals of Computer Science*, 26(1):1–26, 2016.
61. J.A. Bergstra and C.A. Middelburg. A Hoare-like logic of asserted single-pass instruction sequences. *Scientific Annals of Computer Science*, 26(2):125–156, 2016.
62. J.A. Bergstra and C.A. Middelburg. Contradiction-tolerant process algebra with propositional signals. *Fundamenta Informaticae*, 153(1–2):29–55, 2017.
63. J.A. Bergstra and C.A. Middelburg. Axioms for behavioural congruence of single-pass instruction sequences. *Scientific Annals of Computer Science*, 27(2):111–135, 2017.

64. J.A. Bergstra and C.A. Middelburg. Instruction sequences expressing multiplication algorithms. *Scientific Annals of Computer Science*, 28(1):39–66, 2018.
65. J.A. Bergstra and C.A. Middelburg. A short introduction to program algebra with instructions for Boolean registers. *Computer Science Journal of Moldova*, 26(3):199–232, 2018.
66. J.A. Bergstra and C.A. Middelburg. Program algebra for Turing-machine programs. *Scientific Annals of Computer Science*, 29(2):113–139, 2019.
67. J.A. Bergstra and C.A. Middelburg. Process algebra with strategic interleaving. *Theory of Computing Systems*, 63(3):488–505, 2019.
68. J.A. Bergstra and C.A. Middelburg. On the complexity of the correctness problem for non-zeroness test instruction sequences. *Theoretical Computer Science*, 802:1–18, 2020.
69. C.A. Middelburg. Probabilistic process algebra and strategic interleaving. *Scientific Annals of Computer Science*, 30(2):205–243, 2020.
70. C.A. Middelburg. On the strongest three-valued paraconsistent logic contained in classical logic and its dual. *Journal of Logic and Computation*, 31(2):597–611, 2021.
71. J.A. Bergstra and C.A. Middelburg. Using Hoare logic in a process algebra setting. *Fundamenta Informaticae*, 179(4):321–344, 2021.
72. C.A. Middelburg. Imperative process algebra with abstraction. *Scientific Annals of Computer Science*, 32(1):137–179, 2022.
73. C.A. Middelburg. Program algebra for random access machine programs. *Scientific Annals of Computer Science*, 32(2):285–319, 2022.
74. C.A. Middelburg. Belnap-Dunn logic and query answering in inconsistent databases with null values. *Scientific Annals of Computer Science*, 33(2):159–192, 2023.
75. C.A. Middelburg. Paraconsistent logic and query answering in inconsistent databases. *Journal of Applied Non-Classical Logics*, 34(1):133–154, 2024.
76. C.A. Middelburg. Imperative process algebra and models of parallel computation. *Theory of Computing Systems*, 68(3):529–570, 2024. doi:10.1007/s00224-024-10164-0
77. C.A. Middelburg. Dormancy-aware timed branching bisimilarity, with an application to communication protocol analysis. *Theoretical Computer Science*, 1008: Article 114681, 2024. doi:10.1016/j.tcs.2024.114681

Conference papers

78. C.A. Middelburg. The effect of the PDP11 architecture on code generation for CHILL. In *Proceedings ACM Symposium on Architectural Support for Programming Languages and Operating Systems*, pages 149–157, 1982.
79. C.A. Middelburg. The VIP VDM specification language. In R. Bloomfield, L. Marshall and R. Jones, editors, *VDM '88*, pages 187–201. LNCS 328, Springer-Verlag, 1988.
80. C.A. Middelburg. Experiences with combining formalisms in VVSL. In J.A. Bergstra and L.M.G. Feijs, editors, *Algebraic Methods II: Theory, Tools and Applications*, pages 83–103. LNCS 490, Springer-Verlag, 1991.
81. C.A. Middelburg and G.R. Renardel de Lavalette. LPF and MPL_ω – A logical comparison of VDM-SL and COLD-K. In S. Prehn and W.J. Toetenel, editors, *VDM '91*, pages 279–308. LNCS 551, Springer-Verlag, 1991.
82. C.A. Middelburg. VVSL specification of a transaction-oriented access handler. In D.J. Harper and M.C. Norrie, editors, *Specifications of Database Systems*, pages 188–212. Workshops in Computing Series, Springer-Verlag, 1992.

83. L.G. Bouma, W.G. Levelt, A.A.J. Melisse, C.A. Middelburg and L. Verhaard. Formalisation of properties for feature interaction detection: experience in a real-life situation. In H.-J. Kugler, A. Mullery and N. Niebert, editors, *IS&N '94*, pages 393–405. LNCS 851, Springer-Verlag, 1994.
84. J.A. Bergstra and C.A. Middelburg. Process algebra semantics of φSDL. In A. Ponse, C. Verhoef and S.F.M. Vlijmen, editors, *ACP '95*, pages 309–346. Report 95-14, Eindhoven University of Technology, Department of Computing Science, 1995.
85. Yuan Zhaorui, Yang Fangchun and C.A. Middelburg. Detection of non-determinacy feature interaction in ACP$^\tau$. In *Communications 1999,* volume 2, pages 1231–1235. Publishing House BUPT, Beijing, China, 1999.
86. Yuan Zhaorui, Yang Fangchun and C.A. Middelburg. Detection of livelock feature interaction in ACP$^\tau$, In *Future Telecommunication Forum '99.* Publishing House BUPT, Beijing, China, 1999.
87. J.A. Bergstra and C.A. Middelburg. A thread algebra with multi-level strategic interleaving. In S.B. Cooper, B. Löwe and L. Torenvliet, editors, *CiE 2005*, pages 35–48. LNCS 3526, Springer-Verlag, 2005.
88. J.A. Bergstra and C.A. Middelburg. Strong splitting bisimulation equivalence. In J.L. Fiadeiro, N. Harman, M. Roggenbach and J. Rutten, editors, *CALCO 2005*, pages 85–99. LNCS 3629, Springer-Verlag, 2005.
89. J.A. Bergstra and C.A. Middelburg. Model theory for process algebra. In A. Middeldorp, V. van Oostrom, F. van Raamsdonk and R. C. de Vrijer, editors, *Processes, Terms and Cycles: Steps on the Road to Infinity*, pages 445–495, LNCS 3838, Springer-Verlag, 2005.
90. J.A. Bergstra and C.A. Middelburg. Transmission protocols for instruction streams. In M. Leucker and C. Morgan, editors, *ICTAC 2009*, pages 127–139, LNCS 5684, Springer-Verlag, 2009.
91. C.A. Middelburg. On the formalization of the notion of an algorithm. In A. Cavalcanti and J. Baxter, editors, *The Practice of Formal Methods*, pages 23–44, LNCS 14781, Springer-Verlag, 2024.

Preprints

92. C.A. Middelburg. A simple language for expressing properties of telecommunication services and features. Publication 94-PU-356, PTT Research, October 1994.
93. J.A. Bergstra, W.J. Fokkink and C.A. Middelburg. A logic for signal inserted timed frames. Logic Group Preprint Series 155, Utrecht University, Department of Philosophy, January 1996.
94. J.C.M. Baeten, C.A. Middelburg and M.A. Reniers. A new equivalence for processes with timing. Computing Science Report 02-10, Eindhoven University of Technology, Department of Mathematics and Computing Science, October 2002.
95. J.A. Bergstra and C.A. Middelburg. Process algebra with conditionals in the presence of epsilon. Computing Science Report 05-15, Eindhoven University of Technology, Department of Mathematics and Computing Science, May 2005.
96. J.A. Bergstra and C.A. Middelburg. Instruction sequences and non-uniform complexity theory. Electronic Report PRG0812, University of Amsterdam, Programming Research Group, September 2008.
97. J.A. Bergstra and C.A. Middelburg. Instruction sequences for the production of processes. Electronic Report PRG0814, University of Amsterdam, Programming Research Group, November 2008.

98. J.A. Bergstra and C.A. Middelburg. Instruction sequence notations with probabilistic instructions. Electronic Report PRG0906, University of Amsterdam, Programming Research Group, June 2009.
99. J.A. Bergstra and C.A. Middelburg. On the definition of a theoretical concept of an operating system. `arXiv:1006.0813v1 [cs.OS]`, June 2010.
100. C.A. Middelburg. A survey of paraconsistent logics. `arXiv:1103.4324v3 [cs.LO]`, March 2011.
101. J.A. Bergstra and C.A. Middelburg. An application specific informal logic for interest prohibition theory. `arXiv:1104.0308v1 [q-fin.GN]`, April 2011.
102. J.A. Bergstra and C.A. Middelburg. Turing impossibility properties for stack machine programming. `arXiv:1201.6028v1 [cs.LO]`, January 2012.
103. J.A. Bergstra, C.A. Middelburg, and Gh. Ştefănescu. Network algebra for synchronous dataflow. `arXiv:1303.0382v1 [cs.LO]`, March 2013.
104. J.A. Bergstra and C.A. Middelburg. Instruction sequence expressions for the secure hash algorithm SHA-256. `arXiv:1308.0219v7 [cs.PL]`, August 2013.
105. C.A. Middelburg. Process algebra, process scheduling, and mutual exclusion. `arXiv:2003.00473v3 [cs.LO]`, March 2020.
106. R.J. van Glabbeek and C.A. Middelburg. On infinite guarded recursive specifications in process algebra. `arXiv:2005.00746v1 [cs.LO]`, May 2020.
107. C.A. Middelburg. A classical-logic view of a paraconsistent logic. `arXiv:2008.07292v7 [cs.LO]`, August 2020.
108. C.A. Middelburg. A conventional expansion of first-order Belnap-Dunn logic. `arXiv:2301.10555v6 [cs.LO]`, January 2023.
109. C.A. Middelburg. The interdefinability of expansions of Belnap-Dunn logic. `arXiv:2403.04641v2 [cs.LO]`, March 2024.
110. C.A. Middelburg. Formalizing the notions of non-interactive and interactive algorithms. `arXiv:2405.19037v3 [cs.CC]`, May 2024.
111. J.A. Bergstra and C.A. Middelburg. Space-time process algebra with asynchronous communication. `arXiv:2409.15120v2 [cs.LO]`, September 2024.
112. C.A. Middelburg. Complementing an imperative process algebra with a rely/guarantee logic. `arXiv:2502.03320v1 [cs.LO]`, February 2025.

Other publications

113. C.A. Middelburg. Het relationele model gezien in het licht van de Codasyl voorstellen. *Informatie*, 17(2):67–76, 1975 (in Dutch).
114. C.A. Middelburg. Een beschouwing over informatiesystemen en databasemanagement. *Het PTT-Bedrijf*, 20(4):221–224, 1977 (in Dutch).
115. C.A. Middelburg and J. Mendrik. Een ingenieurskijk op Teletekst en Viewdata: Technische achtergronden. *De Ingenieur*, 90(48):927–932, 1978 (in Dutch).
116. C.A. Middelburg. A formal definition based design of the translation phase of a CHILL compiler. In *CHILL Implementors/Users Meeting*, Technical University of Denmark, Lyngby, 1980.
117. C.A. Middelburg. Prolog: Programmeren in logica? *Informatie*, 26(11):866–870, 1984 (in Dutch).
118. B.T. Denvir, J.E.P. Fienieg, W.T. Harwood, C.A. Middelburg and P.M. Taylor. Methods of defining, cataloguing and retrieving specifications of abstract data types. Final deliverable from the CEC-ADT Study, Dr. Neher Laboratorium and Standard Telecommunication Laboratories, 1984.

119. C.A. Middelburg. Programmeren in logica. In J.A.A.M. Poirters and G.J. Schoenmakers, editors, *Colloquium Programmeertalen*, pages 39–60. Academic Service, 1986 (in Dutch).
120. SPECS Project Team. A feasible IBC software specification environment. Final deliverable from RACE project 2093: Specification Environment for Communication Software (SPECS), SPECS Consortium, 1987.
121. J. Bruijning and C.A. Middelburg. VDM extensions: Final report. Final deliverable from ESPRIT project 1283: VDM for Interfaces of the PCTE (VIP), Praxis, Dr. Neher Laboratorium, CWI, and Océ, 1988.
122. C.A. Middelburg. Combining VDM and temporal logic. In *Workshop on Specification of Concurrent Systems*, Philips Research Laboratories, Eindhoven, 1989.
123. C.A. Middelburg. Preliminary remarks on Bear's model of BSI/VDM modules. BSI IST/5/50 Document N-133, BSI, 1989.
124. C.A. Middelburg. *Syntax and Semantics of VVSL*. Ph.D. thesis, University of Amsterdam, 1990.
125. C.A. Middelburg. Evaluation report from the ISO/VDM Review Board. Document I-10, ISO/IEC JTC1/SC22/WG19, 1992.
126. C.A. Middelburg. A framework for defining modular structuring facilities. In *Modules Meeting*, National Physical Laboratory, Teddington, 1992.
127. C.A. Middelburg. Is programmatuurontwikkeling met formele methoden zinvol. *Informatie*, 34(3):148–158, 1992 (in Dutch).
128. C.A. Middelburg. *Programmatuur zonder Logica*. Inaugural lecture, Utrecht University, Department of Philosophy, 1993 (in Dutch).
129. C.A. Middelburg. Design calculi in software development: Theory and practice. In *Proceedings of EU-China High Tech Conference*, pages 307–311. SSTCC and EU DG XII, 1996.
130. C.A. Middelburg. Beyond SDL. Lecture notes of UNU/IIST Course *DesCaRTeS*, United Nations University, International Institute for Software Technology, 1997.
131. C.A. Middelburg and M.A. Reniers. Introduction to process theory. Lecture notes of TUE Course *Process Theory*, Eindhoven University of Technology, Department of Mathematics and Computing Science, 2003.
132. C.A. Middelburg. Searching publications on operating systems. `arXiv:1003.5525v1 [cs.OS]`, March 2010.
133. C.A. Middelburg. Searching publications on software testing. `arXiv:1008.2647v1 [cs.SE]`, August 2010.
134. J.A. Bergstra and C.A. Middelburg. Interest prohibition and financial product innovation. In A. Escurat, editor, *Finance Islamique: Regard(s) sur une Finance Alternative*, pages 274–284. Mazars Hadj Ali, Algiers, 2011.
135. C.A. Middelburg. A short introduction to process theory. `arXiv:1610.01412v1 [cs.LO]`, October 2016.

Unpublished documents

136. C.A. Middelburg. Towards modular structuring facilities in VDM-SL. Unpublished note, invited by ISO/IEC JTC1/SC22/WG19, 1993.
137. C.A. Middelburg. Computer-aided formal software development. Unpublished note, invited by KPN Research, 1993.