Bibliography

Abramsky S. (1990). The lazy lambda calculus. In *Research Topics in Functional Programming, Austin Year of Programming Series* (Turner D., ed.), Austin TX, pp. 65–117. Reading MA: Addison-Wesley.

Achten P.M., van Groningen J.H.G. and Plasmeijer M.J. (1993). High-level specification of I/O in functional languages. In *Proc. 1992 International Workshop on Functional Languages,* Glasgow. Workshop Notes in Computer Science. Berlin: Springer-Verlag., in press.

Aho A.V., Sethi R. and Ullman J.D. (1986). *Compilers – Principles, Techniques and Tools*. Reading MA: Addison-Wesley.

Annot J.K. and Twist R.A.H. van (1987). A novel deadlock free and starvation free packet switching communication processor. In *Proc. Parallel Architectures and Languages Europe (PARLE '87),* Eindhoven (Bakker J.W. de, Nijman A.J. and Treleaven P.C., eds.), *LNCS*, **258**, pp. 68–85. Berlin: Springer-Verlag.

Augusteijn L. (1985). *The Warshall shortest path algorithm in POOL–T*. Internal Report Esprit Project 415 A, Doc. 0105. Eindhoven: Philips Research Laboratories.

Augustsson L.A. (1984). A compiler for lazy ML. In *Proc. ACM Symposium on LISP and Functional Programming,* Austin TX, pp. 218–227. ACM Press.

Backus J. (1978). Can programming be liberated from the Von Neumann style? A functional style and its algebra of programs. *Comm ACM,* **21**, pp. 613–641.

Baeten J.C.M., Bergstra J.A. and Klop J.W. (1987). Term rewriting systems with priorities. In *Proc. Conference on Rewriting Techniques and Applications,* Bordeaux (Lescanne P., ed.), *LNCS*, **256**, pp. 83–94. Berlin: Springer-Verlag.

Bakel S.J. van, Brock S. and Smetsers J.E.W. (1992). Partial type assignment in left-linear term rewriting systems. In *Proc. 17th Colloquium on Trees in Algebra and Programming (CAAP '92),* Rennes (Raoult J.C., ed.), *LNCS*, **581**, pp. 300–321. Berlin: Springer-Verlag.

Baker H.G. (1978). List processing in real time on a serial computer. *Comm ACM,* **4**, (21), pp. 280–294.

Barendregt H.P. (1984). *The Lambda-Calculus, its Syntax and Semantics*. Amsterdam: North-Holland.

Barendregt H.P., Eekelen M.C.J.D. van, Glauert J.R.W., Kennaway J.R., Plasmeijer M.J. and Sleep M.R. (1987a). Term graph rewriting. In *Proc. Parallel Architectures and Languages Europe (PARLE '87),* Eindhoven (Bakker J.W. de, Nijman A.J. and Treleaven P.C., eds.), *LNCS*, **259**, II, pp. 141–158. Berlin: Springer-Verlag.

Barendregt H.P., Eekelen M.C.J.D. van, Glauert J.R.W., Kennaway J.R., Plasmeijer M.J. and Sleep M.R. (1987b). Towards an intermediate language based on graph rewriting. In *Proc. Parallel Architectures and Languages Europe (PARLE '87),* Eindhoven (Bakker J.W. de, Nijman A.J. and Treleaven P.C., eds.), *LNCS*, **259**, II, pp. 159–175. Berlin: Springer-Verlag.

Barendregt H.P., Eekelen M.C.J.D. van, Glauert J.R.W., Kennaway J.R., Plasmeijer M.J. and Sleep M.R. (1988). LEAN: An intermediate language based on graph rewriting. *Parallel Computing,* **9**, pp. 163–177.

Barendsen E. and Smetsers J.E.W. (1992). *Graph rewriting and copying*. Technical Report 92-20, University of Nijmegen.

Berkling K.J. and Fehr E. (1982). A consistent extension of λ-calculus as a base for functional programming languages. *Information and control,* **55**, (1–3), pp. 89–101.

Bevan D.I. (1987). Distributed garbage collection using reference counting. In *Proc. Parallel Architectures and Languages Europe (PARLE '87),* Eindhoven (Bakker J.W. de, Nijman A.J. and Treleaven P.C., eds.), *LNCS*, **259**, pp. 176–187. Berlin: Springer-Verlag.

Bird R.S. and Wadler P. (1988). *Introduction to Functional Programming*. Englewood Cliffs NY: Prentice-Hall.

Bruijn N.G. de (1972). Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation. *Indagationes Mathematicae,* **34**, (2), pp. 381–39.

Brus T., Eekelen M.C.J.D. van, Leer M. van, Plasmeijer M.J. and Barendregt H.P. (1987). Clean – a language for functional graph rewriting. In *Proc. Conference on Functional Programming Languages and Computer Architecture (FPCA '87),* Portland OR (Kahn G., ed.), *LNCS*, **274**, pp. 364–384. Berlin: Springer-Verlag.

Burks A.W., Goldstine H.H. and Neumann J. von (1946). Preliminary discussion of the logical design of an electronic computing instrument. In *John von Neumann, Collected Works*, **5**, pp. 35–79.

Burn G.L., Hankin C.L. and Abramsky S. (1985). The theory of strictness analysis for higher order functions. In *Proc. Workshop on Programs as Data Objects,* Copenhagen (Ganzinger H. and Jones N.D., eds.), *LNCS*, **217**, pp. 42–62. Berlin: Springer-Verlag.

Burstall R.M., MacQueen D.B. and Sanella D.T. (1980). Hope: an experimental applicative language. In *Proc. ACM Symposium on LISP and Functional Programming,* Stanford CA, pp. 136–143. University of Stanford.

Burton F.W. (1987). Functional programming for concurrent and distributed computing. *Computer J,* **30**, (5), pp. 437–450.

Burton F.W. and Sleep M.R. (1982). Executing functional programs on a virtual tree of processors. In *Proc. Conf. Functional Programming Languages and Computer Architecture (FPCA '82),* Portsmouth, New Hampshire, pp. 187–194.

Church A. (1932). A set of postulates for the foundation of logic. *Annals of Mathematics,* **2**, (33), pp. 346–366.

Church A. (1933). A set of postulates for the foundation of logic. *Annals of Mathematics,* **2**, (34), pp. 839–864.

Church A. (1936). An unsolvable problem of elementary number theory. *American J Mathematics,* **58**, pp. 354–363.

Church A. and Rosser J.B. (1936). Some properties of conversion. *Trans American Mathematical Society,* **39**, pp. 472–482.

Cohen J. (1981). Garbage collection of linked data structures. *Computing Surveys,* **13**, (3), pp. 341–367.

Curien P.-L. (1986). *Categorical Combinators, Sequential Algorithms and Functional Programming*. New York: Pitman/Wiley.

Curry H. B. (1930). Grundlagen der Kombinatorischen Logik. *American J of Mathematics,* **52**, pp. 509–536, 789–834.

Curry H.B. and Feys R. (1958). *Combinatory Logic*. **1**. Amsterdam: North-Holland.

Darlington J., Field A.J., Harrison P.G., Harpe D., Jouret G.K., Kelly P.L., Sephton K.M. and Sharp D.W. (1991). Structured parallel functional programming. In *Proc. 3rd International Workshop on the Implementation of Functional Languages on Parallel Architectures,* Southampton (Glaser H. and Hartel P., eds.). Technical Report 91-07, pp. 31–51, University of Southampton.

Dershowitz N. and Jouannaud J.P. (1990). Rewrite systems. In *Handbook of Theoretical Computer Science* (van Leeuwen J., ed.), pp. 52–66. Amsterdam: North-Holland.

Dijkstra E.W. (1968). GOTO statement considered harmful. *Comm ACM,* **11**, pp. 341–346.

Dijkstra E. W., Lamport L., Martin A.J., Scholten C.S. and Steffens E.F.M. (1978). On-the-fly garbage collection: an exercise in cooperation. *Comm ACM,* **21**, (11), pp. 966–975.

Eekelen M.C.J.D. van (1988). *Parallel graph rewriting, some contributions to its theory, its implementation and its application*. Ph.D. Thesis, University of Nijmegen.

Eekelen M.C.J.D. van, Plasmeijer M.J. and Smetsers J.E.W. (1991). Parallel graph rewriting on loosely coupled machine architectures. In *Proc. Conditional and Typed Rewriting Systems (CTRS '90),* Montreal (Kaplan S. and Okada M., eds.), *LNCS*, **516**, pp. 354–

369. Berlin: Springer-Verlag.

Eekelen M.C.J.D. van, Huitema H., Nöcker E.G.J.M.H., Plasmeijer M.J. and Smetsers J.E.W. (1992). *Concurrent Clean, Language Manual – Version 0.8*. Technical Report 92-18, University of Nijmegen.

Ehrig H., Pfender M. and Schneider H.J. (1973). Graph-grammars: an algebraic approach. In *Proc. IEEE Conf. on Automata and Switching Theory*, pp. 167–180.

Field A.J. and Harrison P.G. (1988). *Functional Programming*. Reading MA: Addison-Wesley.

Fraenkel A.A. (1922). Zu den Grundlagen der Cantor-Zermeloschen Mengenlehre. *Mathematische Annalen,* **86**, pp. 230–237.

Girard J.-Y. (1987). Linear logic. *Theoretical Computer Science,* **50**, pp. 1–102.

Glaser H.W. and Thompson P. (1985). Lazy garbage collection. *Software Practice & Experience,* **17**, (1), pp. 1–4.

Glauert J.R.W., Kennaway J.R. and Sleep M.R. (1987). DACTL: a computational model and compiler target language based on graph reduction. *ICL Technical J,* **5**, pp. 509–537.

Goguen J., Kirchner C. and Meseguer J. (1986). Concurrent term rewriting as a model of computation. In *Proc. Workshop on Graph Reduction,* Santa Fe NM (Fasel J.H. and Keller R.M., eds.), LNCS, **279**, pp. 53–94. Berlin: Springer-Verlag.

Goldsmith R., McBurney D.L. and Sleep M.R. (1993). Parallel execution of Concurrent Clean on ZAPP. In *Term Graph Rewriting* (Sleep M.R., Plasmeijer M.J. and Eekelen M.C.J.D. van, eds.). New York: John Wiley.

Groningen J.H.G. van (1990). *Implementing the ABC-machine on M680x0 based architectures*. Master Thesis, University of Nijmegen.

Groningen J.H.G. van, Nöcker E.G.J.M.H. and Smetsers J.E.W. (1992). Some implementation aspects of Concurrent Clean on distributed memory architectures. In *Proc. 4th International Workshop on the Implementation of Functional Languages on Parallel Architectures,* Aachen (Kuchen H. and Loogen R., eds.). Aachen: RWTH.

Groot D. de and Lindstrom G (1986). *Logic Programming: Functions, Relations, Equations*. Englewood Cliff NY: Prentice-Hall.

Harper R., MacQueen D. and Milner R. (1986). *Standard ML.* Internal Report ECS-LFCS-86-2, Edinburgh University.

Hartel P.H. and Langendoen K.G. (1993). Benchmarking implementations of lazy functional languages. In *Proc. Functional Programming and Computer Architectures (FPCA '93),* Copenhagen. ACM Press, in press.

Hindley J.R. (1969). The principal type scheme of an object in combinatory logic. *Trans American Mathematical Society,* **146**, pp. 29–60.

Hoare C.A.R. (1978). Communicating sequential processes. *Comm ACM,* **21**, (8), pp. 666–677.

Hudak P. and Smith L. (1986). Para-functional programming: a paradigm for programming multiprocessor systems. In *Proc. 12th ACM Symp. on Principles of Programming Languages*, pp. 243–254.

Hudak P., Peyton Jones S., Wadler Ph., Boutel B., Fairbairn J., Fasel J., Hammond K., Hughes J., Johnsson Th., Kieburtz D., Nikhil R., Partain W. and Peterson J. (1992). Report on the programming language Haskell. *ACM SigPlan Notices,* **27**, (5), pp. 1–164.

Huet G. and Lévy J.-J. (1979). *Call-by-need computations in non-ambiguous linear term rewriting systems*. Internal Report 359, INRIA.

Huet G. and Oppen D.C. (1980). Equations and rewrite rules: a survey. In *Formal Languages: Perspectives and Open Problems* (Book R. V., ed.). London: Academic Press.

Hughes R.J.M. (1982). Super-combinators: a new implementation method for applicative languages. In *Proc. ACM Symposium on Lisp and Functional Programming,* Pittsburgh, pp. 1–10.

Johnsson Th. (1984). Efficient compilation of lazy evaluation. In *Proc. ACM SIGPLAN '84, Symposium on Compiler Construction,* Montreal, pp. 58–69.

Johnsson Th. (1985). Lambda lifting: transforming programs to recursive equations. In *Proc. Conference on Functional Programming Languages and Computer Architecture,* Nancy (Jouannaud J.P., ed.), *LNCS*, **201**, pp. 190–203. Berlin: Springer-Verlag.

Kennaway J.R. (1988). The correctness of an implementation of functional Dactl by parallel rewriting. In *Proc. UK IT 88 ("Alvey") Conference (I.E.E.)*, pp. 254–257. British Computer Society.

Kennaway J.R. (1990). Graph rewriting in a category of partial morphisms. In *Proc. 4th International Workshop on Graph Grammars and their Applications,* Bremen (Ehrig H., Kreowski H.J. and Rozenberg G., eds.), *LNCS*, **532**, pp. 490–504. Berlin: Springer-Verlag.

Kennaway J.R., Klop J.W., Sleep M.R. and de Vries F.J. (1993a). An infinitary Church–Rosser property for non-collapsing orthogonal term rewriting systems. In *Term Graph Rewriting* (Sleep M.R., Plasmeijer M.J. and Eekelen M.C.J.D. van, eds.). New York: John Wiley.

Kennaway J.R., Klop J.W., Sleep M.R. and de Vries F.J. (1993b). The adequacy of term graph rewriting for simulating term rewriting. In *Term Graph Rewriting* (Sleep M.R., Plasmeijer M.J. and Eekelen M.C.J.D. van, eds.). New York: John Wiley.

Kernighan B., Ritchie W. and Dennis M. (1978). *The C Programming Language*. Englewood Cliff NY: Prentice-Hall.

Kesseler M.H.G. (1991). Implementing the ABC machine on transputers. In *Proc. 3rd International Workshop on Implementation of Functional Languages on Parallel Architectures,* Southampton

(Glaser H. and Hartel P., eds.). Technical Report 91-07, pp. 147–192, University of Southampton.

Kesseler M.H.G. (1992). Communication issues regarding parallel functional graph rewriting. In *Proc. 4th International Workshop on the Implementation of Functional Languages on Parallel Architectures,* Aachen (Kuchen H. and Loogen R., eds.). Aachen: RWTH.

Kingdon H., Lester D. and Burn G.L. (1991). The HDG-machine: a highly distributed graph-reducer for a transputer network. *The Computer J,* **34**, (4).

Klop J.W. (1992). Term rewriting systems. In *Handbook of Logic in Computer Science* (Abramsky S., Gabbay D. and Maibaum T., eds.), **2**, pp. 2–108. Oxford: Oxford University Press.

Kluge W.E. (1983). Cooperating reduction machines. *IEEE. Transactions on Computers,* **C-32**, (11), pp. 1002–1012.

Koopman P.W.M. (1987). Interactive programs in a functional language: a functional implementation of an editor. *Software Practice & Experience,* **17**, (9), pp. 609–622.

Koopman P.W.M. (1990). *Functional programs as executable specifications*. Ph.D. Thesis, University of Nijmegen.

Koopman P.W.M. and Nöcker E.G.J.M.H. (1988). *Compiling functional languages to term graph rewriting systems*. Technical Report 88-1, University of Nijmegen.

Koopman P.W.M., Eekelen M.C.J.D. van, Nöcker E.G.J.M.H., Plasmeijer M.J. and Smetsers J.E.W. (1990). *The ABC-machine: a sequential stack-based abstract machine for graph rewriting*. Technical Report 90-22, University of Nijmegen.

Koopman P.W.M., Eekelen M.C.J.D van and Plasmeijer M.J. (1993). Operational machine specification in a functional programming language. *Software Practice & Experience,* in press.

Lamping J. (1990). An algorithm for optimal lambda calculus reduction. In *Proc. 17th ACM Symp. of Principles of Programming Languages,* San Francisco, pp. 16–30.

Landin P.J. (1964). The mechanical evaluation of expressions. *Computer J,* **6**, pp. 308–20.

Landin P.J. (1966). The next 700 programming languages. *Comm ACM,* **9**, (3), pp. 157–164.

Lévy J.J. (1980). Optimal reductions in the lambda-calculus. In *To H.B. Curry: Essays on Combinatory Logic, Lambda-Calculus and Formalism* (Seldin J.P. and Hindley J.R., eds.), pp. 159–191. New York: Academic Press.

Liebermann H. and Hewitt C. (1983). A real-time garbage collector based on the lifetime of objects. *Comm ACM,* **26**, pp. 419–429.

Markovsky G. (1976). Chain-complete p.o.-sets and directed sets with applications. *Algebra Universalis,* **6**, pp. 53–68.

McCarthy J. (1960). Recursive functions of symbolic expressions and their computation by machine. *Comm ACM,* **4**, pp. 184–195.

Meinke K. and Tucker J.V. (1992). Universal algebra. In *Handbook of Logic in Computer Science* (Abramsky S., Gabbay D. and Maibaum T., eds.), **1**, pp. 189–398. Oxford: Oxford University Press.

Milner R.A. (1978). Theory of type polymorphism in programming. *J of Computer and System Sciences,* **17**, (3), pp. 348–375.

Minsky M.L. (1963). *A Lisp garbage collection algorithm using serial secondary storage*. Project MAC Memo 58 (rev.), Massachusetts Institute of Technology.

Morris F.L. (1978). A time- and space-efficient garbage compaction algorithm. *Comm ACM,* **21**, (8), pp. 662–665.

Motorola (1984–1985). *MC68020 32-Bit Microprocessor User's Manual*. Englewood Cliff NY: Prentice-Hall.

Mycroft A. (1981). *Abstract interpretation and optimising transformations for applicative programs*. Ph.D. Thesis, University of Edinburgh.

Mycroft A. (1984). Polymorphic type schemes and recursive definitions. In *Proc. International Conference on Programming,* Toulouse (Paul M. and Robinet B., eds.), LNCS, **167**, pp. 217–239. Berlin: Springer-Verlag.

Nöcker E.G.J.M.H. (1988). Strictness analysis based on abstract reduction of term graph rewrite systems. In *Proc. Workshop on Implementation of Lazy Functional Languages,* University of Göteborg (Johnsson Th., Peyton Jones S. and Karlsson K., eds.). Report 53, pp. 451–462, Chalmers University of Technology, Programming Methodology Group.

Nöcker E.G.J.M.H. (1989). *The PABC Simulator, v0.5. Implementation Manual*. Technical Report 89-19, University of Nijmegen.

Nöcker E.G.J.M.H. (1990). *Strictness analysis using abstract reduction*. Technical report 90-14, University of Nijmegen.

Nöcker E.G.J.M.H. (1993). Strictness analysis using abstract reduction. In *Proc. Conference on Functional Programming Languages and Computer Architectures (FPCA '93),* Copenhagen. ACM Press, in press.

Nöcker E.G.J.M.H. and Smetsers J.E.W. (1993). Partially strict non-recursive data types. *J of Functional Programming*, in press.

Nöcker E.G.J.M.H., Plasmeijer M.J. and Smetsers J.E.W. (1991a). The parallel ABC machine. In *Proc. 3rd International Workshop on Implementation of Functional Languages on Parallel Architectures,* Southampton (Glaser H. and Hartel P., eds.), Technical Report 91-07, pp. 383–407, University of Southampton.

Nöcker E.G.J.M.H., Smetsers J.E.W., Eekelen M.C.J.D. van and Plasmeijer M.J. (1991b). Concurrent Clean. In *Proc. Parallel Architectures and Languages Europe (PARLE '91),* Eindhoven (Aarts E.H.L., Leeuwen J. and Rem M., eds.), *LNCS*, **506**, pp. 202–219. Berlin: Springer-Verlag.

O'Donnell M.J. (1985). *Equational Logic as a Programming Language*. Foundations of Computing Series. Massachusetts: MIT Press.

Ong C.-H. L. (1988). *Lazy lambda calculus: an investigation into the foundations of functional programming*. Ph.D. Thesis, Imperial College, London.

Peyton Jones S.L. (1987). *The Implementation of Functional Programming Languages*. Englewood Cliffs NY: Prentice-Hall.

Peyton Jones S.L. and J. Salkild (1989). The spineless tagless G-machine. In *Proc. Conference on Functional Programming Languages and Computer Architectures (FPCA '89)*, pp. 184–201. Reading MA: Addison-Wesley.

Peyton Jones S.L. and Wadler P. (1993). Imperative functional programming. In *Proc. POPL '93*, in press.

Plasmeijer M.J., Eekelen M.C.J.D. van, Nöcker E.G.J.M.H. and Smetsers J.E.W. (1991). The Concurrent Clean System, functional programming on the Macintosh. In *Proc. 7th Internernational Conference of the Apple European University Consortium,* Paris, pp. 19–24. Apple Consortium.

Raoult J.C. (1984). On graph rewritings. *Theoretical Computer Science,* **32**, pp. 1–24.

Renardel de Lavalette G.R. (1988). *Strictness analysis for POLYREC, a language with polymorphic and recursive types*. Technical Report, University of Utrecht.

Robinson J.A. (1965). A machine-oriented logic based on the resolution principle. *J of the ACM,* **12**, (1), pp. 23–41.

Schönfinkel M. (1924). Über die Bausteine der mathematischen Logik. *Mathematische Annalen,* **92**, pp. 305–316.

Smetsers J.E.W. (1989). *Compiling Clean to abstract ABC-machine code*. Technical Report 89-20, University of Nijmegen.

Smetsers J.E.W. (1993). *Graph rewriting and functional languages*. Ph. D. Thesis, University of Nijmegen.

Smetsers J.E.W., Barendsen E., Eekelen M.C.J.D. van and Plasmeijer M.J. (1993). *Guaranteeing safe destructive updates through a type system with uniqueness information for graphs*. Technical report 93-4, University of Nijmegen.

Smetsers J.E.W., Nöcker E.G.J.M.H., Groningen J.H.G. van and Plasmeijer M.J. (1991). Generating efficient code for lazy functional languages. In *Proc. Conference on Functional Programming Languages and Computer Architecture (FPCA '91),* Cambridge MA (Hughes J., ed.), *LNCS*, **523**, pp. 592-617. Berlin: Springer-Verlag.

Staples J. (1980). Computation on graph-like expressions. *Theoretical Computer Science,* **10**, pp. 171-185.

Stoy J.E. (1977). *Denotational Semantics: The Scott–Strachey Approach to Programming Language Theory*. Massachusetts: MIT Press.

Toyama Y., Smetsers J.E.W., Eekelen M.C.J.D. van and Plasmeijer

M.J. (1993). The functional strategy and transitive term rewriting systems. In *Term Graph Rewriting* (Sleep M.R., Plasmeijer M.J. and Eekelen M.C.J.D. van, eds.). New York: John Wiley.

Turing A.M. (1937). Computability and λ-definability. *J of Symbolic Logic,* **2**, pp. 153-163.

Turner D.A. (1979a). A new implementation technique for applicative languages. *Software Practice & Experience,* **9**, (1), pp. 31-49.

Turner D.A. (1979b). *SASL Language Manual.* St. Andrews: University of St Andrews.

Turner D.A. (1985). Miranda: a non-strict functional language with polymorphic types. In *Proc. Conference on Functional Programming Languages and Computer Architecture,* Nancy, France (Jouannaud J.P., ed.), *LNCS*, **201**, pp. 1-16. Berlin: Springer-Verlag.

Vree W.G. and Hartel P.H. (1988). *Parallel graph reduction for divide-and-conquer applications; Part I – program transformations*. PRM Project Internal Report D-15, University of Amsterdam.

Wadler P. (1987). Strictness analysis on non-flat domains (by abstract interpretation over finite domains). In *Abstract Interpretation of Declarative Languages* (Abramsky S. and Hankin C., eds.). Chichester: Ellis Horwood.

Wadler P. (1990). Linear types can change the world! In *Programming Concepts and Methods* (Broy M. and Jones C.B., eds.). Amsterdam: North-Holland.

Wadsworth C.P. (1971). *Semantics and pragmatics of the lambda-calculus*. Ph.D. Thesis, University of Oxford.

Watson P. and Watson I. (1987). An efficient garbage collection scheme for parallel computer architectures. In *Proc. Parallel Architectures and Languages Europe (PARLE '87),* Eindhoven (Bakker J.W. de, Nijman A.J. and Treleaven P.C., eds.), *LNCS*, **259**, pp. 432-443. Berlin: Springer-Verlag.

Wirth N. (1982). *Programming in Modula-2*. Berlin: Springer-Verlag.

Zermelo E. (1908). Untersuchungen über die Grundlagen der Mengelehre. *International Bibliography, Information and Documentation,* **65**, pp. 261-281.