Bibliography

[AF00] Andrew W. Appel and Amy Felty. A semantic model of types and machine instructions for proof-carrying code. In *27th Symposium on the Principles of Programming Languages (POPL)*, pages 243–253, Boston, January 2000.

[AJ88] Andrew W. Appel and Trevor Jim. Continuation-passing, closure-passing style. Technical Report CS-TR-183-88, Princeton University Department of Computer Science, July (revised September) 1988.

[AM87] Andrew W. Appel and David B. MacQueen. *Proceedings of the Converence on Functional Programming and Computer Architecture*, volume 274 of *Lecture Notes in Computer Science*. Springer-Verlag, Portland, September 1987.

[AN89] Arvind and Rishiyur S. Nikhil. A dataflow approach to general-purpose parallel computing. Computation Structure Group Memo 302, MIT Laboratory for Computer Science, July 1989.

[ANP89] Arvind, Rishiyur S. Nikhil, and Keshav K. Pingali. I-structures: Data structures for parallel computing. *ACM Transactions on Programming Languages and Systems*, pages 598–632, October 1989.

[AP02] Andrew Appel and Jens Palsberg. *Modern Compiler Implementation In Java*. Cambridge University Press, second edition, 2002.

[Ape89] Andrew W. Apel. Runtime tags aren't necessary. *Lisp and Symbolic Computation*, 2:153–162, 1989.

[App90] Andrew W. Appel. A runtime system. *Lisp and Symbolic Computation*, 3:343–380, 1990.

[App92] Andrew W. Appel. *Compiling with Continuations*. Cambridge University Press, 1992.

[App98a] Andrew Appel. *Modern Compiler Implementation In C*. Cambridge University Press, 1998.

[App98b] Andrew Appel. *Modern Compiler Implementation In ML*. Cambridge University Press, 1998.

[ASS96] Harold Abelson, Gerald Jay Sussman, and Julie Sussman. *Structure and Interpretation of Computer Programs*. MIT Press and McGraw-Hill, second edition, 1996.

[ASU86] Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman. *Compilers: Principles, Techniques, and Tools*. Addison-Wesley, 1986.

[Bac78] John Backus. Can programming be liberated from the von Neuman style? A functional style and its algebra of programs. *Communications of the ACM*, 21(8):245–264, August 1978.

[Bar92a] H[enrik] P[ieter] Barendregt. Lambda calculi with types. In S[amson] Abramsky, Dov M. Gabbay, and T[homas] S. E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 2, chapter 2, pages 117–309. Oxford University Press, 1992.

[Bar92b] Paul S. Barth. Atomic data structures for parallel computing. Technical Report MIT/LCS/TR-532, MIT Laboratory for Computer Science, March 1992.

[BCT94] P. Briggs, K. D. Cooper, and L. Torczon. Improvements to graph coloring register allocation. *ACM Transactions on Programming Languages and Systems*, 16(3):428–455, May 1994.

[BDD80] H. Boehm, A. Demers, and J. Donahue. An informal description of russell. Technical Report TR80-430, Cornell University, Department of Computer Science, 1980.

[Bir89] Andrew Birrel. An introduction to programming with threads. SRC Report 35, Digital Equipment Corporation, January 1989.

[BKR99] Nick Benton, Andrew Kennedy, and George Russell. Compiling Standard ML to Java bytecodes. In *Proceedings of the ACM SIGPLAN International Conference on Functional Programming (ICFP '98)*, volume 34(1), pages 129–140, 1999.

[BL84] R. Burstall and B. W. Lampson. *A Kernel Language for Abstract Data Types and Modules*, volume 173 of *Lecture Notes in Computer Science*. Springer-Verlag, 1984.

[Ble90] Guy E. Blelloch. *Vector Models for Data-Parallel Computing*. MIT Press, 1990.

[Ble92] Guy E. Blelloch. NESL: A nested data-parallel language. Technical Report CMU-CS-92-103, Carnegie-Mellon University Computer Science Department, January 1992.

[BN98] Franz Baader and Tobias Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.

[Bri77] P. Brinch Hansen. *The Architecture of Concurrent Programs*. Prentice-Hall, Englewood Cliffs, NJ, 1977.

[BW90] Michael Barr and Charles Wells. *Category Theory for Computer Scientists*. Prentice-Hall, 1990.

[BWD95] Robert G. Burger, Oscar Waddell, and R. Kent Dybvig. Register allocation using lazy saves, eager restores, and greedy shuffling. In *Conference on Programming Language Design and Implementation*. ACM SIGPLAN, June 1995.

[BWW+89] J. Backus, J. H. Williams, E. L. Wimmers, P. Lucas, and A. Aiken. FL language manual, parts 1 and 2. Technical Report RJ 7100 (67163), IBM Research, 1989.

[BWW90] J. Backus, J. H. Williams, and E. L. Wimmers. An introduction to the programming language FL. In D. A. Turner, editor, *Research Topics in Functional Programming*. Addison-Wesley, Reading, MA, 1990.

[CAC+81] G. Chaitin, M. A. Auslander, A. K. Chandra, J. Cocke, M. E. Hopkins, and P. W. Markstein. Register allocation via coloring. *Computer Languages*, 6(1):47–57, January 1981.

[Car89] Luca Cardelli. Typeful programming. In *IFIP Advanced Seminar on Formal Description of Programming Concepts*, 1989.

[CG89] Nicholas Carriero and David Gelernter. Linda in context. *Communications of the ACM*, 32(4):444–458, 1989.

[Cha82] G. J. Chaitin. Register allocation and spilling via coloring. *SIGPLAN Notices*, 17(6):98–105, June 1982. Proceedings of the ACM SIGPLAN Symposium on Compiler Construction.

[CHD01] K. Crary, R. Harper, and D. Dreyer. A type system for higher-order modules, 2001.

[CHP99] Karl Crary, Robert Robert Harper, and Sidd Puri. What is a recursive module? In *Programming Language Design and Implementation (PLDI)*, June 1999.

[CJW00] Henry Cejtin, Suresh Jagannathan, and Stephen Weeks. Flow-directed closure conversion for typed languages. In *9th European Symposium on Programming*, pages 56–71, Berlin, Germany, March 2000.

[Cli82] William Clinger. Nondeterministic call by need is neither lazy nor by name. In *Proceedings of the ACM Symposium on Lisp and Functional Programming*, pages 226–234, Pittsburgh, PA, 1982.

[CM90] Eric Cooper and J. Gregory Morrisett. Adding threads to Standard ML. Technical Report CMU-CS-90-186, Carnegie Mellon University Computer Science Department, December 1990.

[Con63] Melvin E. Conway. Design of a separable transition-diagram compiler. *Communications of the ACM*, 6(7):396–408, 1963.

[Cou90] Bruno Courcelle. Graph rewriting: An algebraic and logic approach. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics*, pages 193–242. MIT Press/Elsevier, 1990.

[CT03] Keith D. Cooper and Linda Torczon. *Engineering a Compiler*. Morgan Kaufmann, 2003.

[CW85] Luca Cardelli and Peter Wegner. On understanding types, data abstraction, and polymorphism. *ACM Computing Surveys*, 17(4):471–522, 1985.

[DF92] Olivier Danvy and Andrzej Filinski. Representing control: A study of the CPS transformation. *Mathematical Structures in Computer Science*, 2:361–391, 1992.

[DF96] Paolo Di Blasio and Kathleen Fisher. A calculus for concurrent objects. In *Seventh International Conference on Concurrency Theory (CONCUR96)*, volume 637 of *Lecture Notes in Computer Science*, pages 655–670, Pisa, August 1996.

[Dij68] E. W. Dijkstra. Co-operating sequential processes. In F. Genuys, editor, *Programming Languages (NATO Advanced Study Institute)*, pages 43–112. London: Academic Press, 1968.

[DJ90] Nachum Dershowitz and Jean-Pierre Jouannaud. Rewrite systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics*, pages 243–320. MIT Press/Elsevier, 1990.

[DJG92] V. Dornic, P. Jouvelot, and D. Gifford. Polymorphic time systems for estimating program complexity. *ACM Letters on Programming Languages and Systems*, 1:33–45, 1992.

[DWM+01] Allyn Dimock, Ian Westmacott, Robert Muller, Franklyn Turbak, and J. B. Wells. Functioning without closure: Type-safe customized function representations for Standard ML. In *6th International Conference on Functional Programming*, pages 14–25, Firenze, Italy, September 2001. ACM.

[FF86] Matthias Felleisen and Daniel Friedman. Control operators, the SECD-machine, and the λ-calculus. In M. Wirsing, editor, *Formal Description of Programming Concepts — III*, pages 193–219. North-Holland, 1986.

[FH92] Matthias Felleisen and Robert Hieb. The revised report on the syntactic theories of sequential control and state. *Theoretical Computer Science*, 102:235–271, 1992.

[FKR+00] Robert Fitzgerald, Todd B. Knoblock, Erik Ruf, Bjarne Steensgaard, and David Tarditi. Marmot: an optimizing compiler for Java. *Software – Practice and Experience*, 30(3):199–232, 2000.

[For91] Alessandro Forin. Futures. In Peter Lee, editor, *Topics in Advanced Language Implementation*, pages 219–241. MIT Press, 1991.

[FSDF93] Cormac Flanagan, Amr Sabry, Bruce F. Duba, and Matthias Felleisen. The essence of compiling with continuations. In *Programming Language Design and Implementation*, pages 238–247. ACM, 1993. (SIGPLAN Notices, Volume 28, Number 6, June 1993).

[FWH01] Daniel P. Friedman, Mitchell Wand, and Christopher T. Haynes. *Essentials of Programming Languages*. MIT Press, second edition, 2001.

[GJ90] David Gelernter and Suresh Jagannathan. *Programming Linguistics*. MIT Press, 1990.

[GJSO92] David Gifford, Pierre Jouvelot, Mark A. Sheldon, and James O'Toole. Report on the FX-91 programming language. Technical Report MIT/LCS/TR-531, MIT Laboratory for Computer Science, February 1992.

[GM94] Carl A. Gunter and John C. Mitchell, editors. *Theoretical Aspects of Object-Oriented Programming: Types, Semantics, and Language Design*. MIT Press, 1994.

[Gor79] Michael J. C. Gordon. *The Denotational Description of Programming Languages*. Springer-Verlag, 1979.

[GS90] C. A. Gunter and D. S. Scott. Semantic domains. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics*, pages 633–674. MIT Press/Elsevier, 1990.

[Gun92] Carl A. Gunter. *Semantics of Programming Languages: Structures and Techniques*. MIT Press, 1992.

[Hal85] Robert Halstead. Multilisp: A language for concurrent symbolic computation. *ACM Transactions on Programming Languages and Systems*, pages 501–528, October 1985.

[Har86] Robert Harper. Modules and persistence in standard ml. Technical Report ECS-LFCS-86-11, University of Edinburgh, Laboratory for Foundations of Computer Science, September 1986.

[Hew77] Carl Hewitt. Viewing control structures as patterns of passing messages. *Artificial Intelligence*, pages 323–364, 1977.

[HH86] James G. Hook and Douglas J. Howe. Impredicative strong existential equivalent to type:type. Technical Report TR 86-760, Department of Computer Science, Cornell University, Ithaca, New York, June 1986.

[HJW+92] Paul Hudak, Simon Petyon Jones, Philip Wadler, et al. Report on the programming language Haskell, version 1.2. *ACM SIGPLAN Notices*, 27(5), May 1992.

[HMM90] Robert Harper, John C. Mitchell, and Eugenio Moggi. Higher-order modules and the phase distinction. In *Converence Record of the Seventeenth Annual ACM Symposium on Principles of Programming Languages*, pages 341–354, San Francisco, CA, January 1990.

[Hoa74] C.A.R. Hoare. Monitors: An operating system structuring concept. *Communications of the ACM*, 17(10):549–557, October 1974.

[Hoa85] C.A.R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.

[Hof80] Douglas R. Hofstadter. *Gödel, Escher, Bach: An Eternal Golden Braid*. Vintage Books, 1980.

[Hor95] Ellis Horowitz. *Programming Languages: A Grand Tour*. W H Freeman & Co., third edition, 1995.

[HS86] W. Daniel Hillis and Guy L. Steele Jr. Data parallel algorithms. *Communications of the ACM*, 29(12), 1986.

[HU79] John E. Hopcroft and Jeffrey Ullman. *Introduction to Automata Theory, Languages, and Computation*. Adison-Wesley, 1979.

[Hue90] Gerard Huet, editor. *Logical Foundations of Functional Programming*. Addison-Wesley, 1990.

[Hug82] R. J. M. Hughes. Super-combinators: A new implementation technique for applicative languages. In *Symposium on Lisp and Functional Programming*, pages 1–10, August 1982.

[JG89] P. Jouvelot and D. Gifford. Reasoning about continuations with control effects. In *Conference on Programming Language Design and Implementation (PLDI)*, pages 218–226, 1989.

[JG91] Pierre Jouvelot and David K. Gifford. Algebraic reconstruction of types and effects. In *Proceedings of the ACM Symposium on the Principles of Programming Languages*. ACM, 1991.

[JM88] Lalita A. Jategaonkar and John C. Mitchell. ML with extended pattern matching and subtypes. In *Procedings of the ACM Conference on Lisp and Functional Programming*, pages 198–211, Snowbird, Utah, July 1988. ACM.

[JM97] Simon Peyton Jones and Erik Meijer. Henk: A typed intermediate language. In *1997 ACM SIGPLAN Workshop on Types in Compilation*, Amsterdam, The Netherlands, June 1997. Boston College Computer Science Department Technical Report BCCS-97-03.

[Joh75] S. C. Johnson. Yacc — yet another compiler compiler. Computing Science Technical Report 32, Bell Laboratories, Murray Hill, N.J., 1975.

[Joh85] Thomas Johnsson. Lambda lifting: Transforming programs to recursive equations. In *Functional Programming Languages and Computer Architecture*, volume 201 of *Lecture Notes in Computer Science*, pages 190–203, September 1985.

[Jon96] Simon Peyton Jones. Compiling Haskell by program transformation: A report from the trenches. In *Proceedings of the European Symposium on Programming*. Springer, 1996.

[JW93] Simon Peyton Jones and Philip Wadler. Imperative functional programming. In *Proceedings of the 20th Symposium on Principles of Programming Languages*. ACM, 1993.

[Kah87] Gilles Kahn. Natural semantics. In *Proceedings of STACS '87, 4th Annual Symposium on Theoretical Aspects of Computer Science*, volume 247 of *Lecture Notes in Computer Science*, pages 22–39. Springer-Verlag, 1987.

[Kam90] Samuel Kamin. *Programming Languages: An Interpreter-Based Approach*. Addison-Wesley, 1990.

[Kel89] Richard Kelsey. *Compilation by Program Transformation*. PhD thesis, Yale University, 1989.

[KH89] Richard Kelsey and Paul Hudak. Realistic compilation by program transformation. In *Principles of Programming Languages*, pages 281–292. ACM, 1989.

[KKR+86] David Kranz, Richard Kelsey, Jonathan A. Rees, Paul Hudak, James Philbin, and Norman I. Adams. Orbit: an optimizing compiler for Scheme. In *Proceedings of the SIGPLAN '86 Symposium on Compiler Construction*, pages 219–233. ACM, June 1986.

[L+79] Barbara Liskov et al. CLU reference manual. Technical Report MIT/LCS/TR-225, MIT Laboratory for Computer Science, October 1979.

[Lan64] P. J. Landin. The mechanical evaluation of expressions. *The Computer Journal*, pages 308–320, January 1964.

[Lea99] Douglas Lea. *Concurrent Programming in Java: Design Principles and Patterns, Second Edition*. Addison-Wesley, Boston, 1999.

[Ler95] Xavier Leroy. Applicative functors and fully transparent higher-order modules. In *22nd Symposium on Principles of Programming Languages*. ACM, 1995.

[Les75] M. E. Lesk. Lex — a lexical analyzer generator. Computing Science Technical Report 39, Bell Laboratories, Murray Hill, N.J., 1975.

[LG88] John M. Lucassen and David K. Gifford. Polymorphic effect systems. In *Proceedings of the ACM Symposium on the Principles of Programming Languages*, pages 47–57, 1988.

[Mac84] David MacQueen. Modules for standard ML. In *Proceedings ACM Symposium on Lisp and Functional Programming*, 1984.

[Mac86] D. B. MacQueen. Using dependent types to express modular structure. In *Symposium on Principles of Programming Languages*. ACM, 1986.

[Mac88] David MacQueen. An implementation of standard ml modules. Part of the SMLNJ Distribution, March 1988.

[Mac99] Bruce J. MacLennan. *Principles of Programming Languages: Design, Evaluation, and Implementation*. Oxford University Press, third edition, 1999.

[McC60] John McCarthy. Recursive functions of symbolic expressions and their computation by machine, part i. *Communications of the ACM*, 3(4):184–195, 1960.

[McC62] John McCarthy. Towards a mathematical science of computation. In *Information Processing*, pages 21–28, Amsterdam, 1962. North Holland.

[McC67] John McCarthy. A basis for a mathematical theory of computation. In P. Braffort and D. Hirschberg, editors, *Computer Programming and Formal Systems*. North-Holland, Amsterdam, 1967.

[Mil78] Robin Milner. A theory of type polymorphism in programming. *Journal of Computer and System Sciences*, pages 348–375, 1978.

[Mil87] James S. Miller. MultiScheme: A parallel processing system based on MIT Scheme. Technical Report MIT/LCS/TR-402, MIT Laboratory for Computer Science, September 1987.

[Mil89] Robin Milner. *Communication and Concurrency*. Prentice-Hall, 1989.

[Min67] Marvin Minsky. *Finite and Infinite Machines*. Prentice-Hall, 1967.

[Mit96] John C. Mitchell. *Foundations for Programming Languages*. MIT Press, Cambridge, Massachusetts, 1996.

[Mit03] John C. Mitchell. *Concepts in Programming Languages*. Cambridge University Press, 2003.

[ML86] Michael Marcotty and Henry Ledgard. *The World of Programming Languages*. Springer-Verlag, 1986.

[MMS78] James G. Mitchell, William Maybury, and Richard Sweet. Mesa language manual (version 4.0). System development division, Xerox Palo Alto Research Center, May 1978.

[Mor95] Greg Morrisett. *Compiling with Types*. PhD thesis, Carnegie Mellon University, 1995.

[Mos90] Peter Mosses. Denotational semantics. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics*, pages 575–631. MIT Press/Elsevier, 1990.

[MP84] John C. Mitchell and Gordon D. Plotkin. Abstract types have existential type. In *Principles of Programming Languages*, pages 37–51. ACM, 1984.

[MT91] Robin Milner and Mads Tofte. *Commentary on Standard ML*. MIT Press, Cambridge, MA, 1991.

[MTH90] Robin Milner, Mads Tofte, and Robert Harper. *The Definition of Standard ML*. MIT Press, Cambridge, MA, 1990.

[MTHM97] Robin Milner, Mads Tofte, Robert Harper, and David MacQueen. *The Definition of Standard ML (Revised)*. MIT Press, Cambridge, MA, 1997.

[Muc97] Steven S. Muchnick. *Advanced Compiler Design and Implementation*. Morgan Kaufmann, 1997.

[MWCG99] G. Morrisett, D. Walker, K. Crary, and N. Glew. From System F to typed assembly language. *ACM Transactions on Programming Languages and Systems*, 21(3):528–569, May 1999.

[NL98] George C. Necula and Peter Lee. The design and implementation of a certifying compiler. In *Programming Language Design and Implementation (PLDI'98)*, pages 333–344, Montreal, June 1998. ACM.

[NNH98] Flemming Nielson, Hanne Riis Nielson, and Chris Hankin. *Principles of Program Analysis*. Springer, 1998.

[NO93] Scott M. Nettles and James W. O'Toole. Real-time replication garbage collection. In *SIGPLAN Symposium on Programming Language Design and Implementation*. ACM, June 1993.

[NOG93] Scott Nettles, James O'Toole, and David Gifford. Concurrent garbage collection of persistent heaps. Technical Report MIT/LCS/TR-569, MIT Laboratory for Computer Science, June 1993.

[NOPH92] Scott M. Nettles, James W. O'Toole, David Pierce, and Nicholas Haines. Replication-based incremental copying collection. In *Proceedings of the SIGPLAN International Workshop on Memory Management*, pages 357–364. ACM, Springer-Verlag, September 1992.

[occ95] *occam 2.1 reference manual*. SGS-Thomson Microeclectronics Limited, May 1995.

[OG89] James William O'Toole, Jr. and David K. Gifford. Type reconstruction with first-class polymorphic values. In *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 207–217, Portland, Oregon, June 1989. ACM.

[Pey87] Simon L. Peyton Jones. *The Implementation of Functional Programming Languages*. Prentice-Hall, 1987.

[Pie91] Benjamin C. Pierce. *Basic Category Theory for Computer Scientists*. MIT Press, 1991.

[Pie02] Benjamin C. Pierce. *Types and Programming Languages*. MIT Press, Cambridge, Massachusetts, 2002.

[Plo75] Gordon D. Plotkin. Call-by-name, call-by-value and the lambda calculus. *Theoretical Computer Science*, 1:125–159, 1975.

[Plo81] Gordon D. Plotkin. A structural approach to operational semantics. Technical Report DAIMI FN-19, Aarhus University Computer Science Department, September 1981.

[Plu02] Mike Plusch. *Water: Simplified Web Services and XML Programming*. John Wiley & Sons, Hoboken, NJ, December 2002. ISBN: 0764525360.

[Rey74] J. C. Reynolds. Towards a theory of type structure. In *Proceedings, Colloque sur la Programmation*, volume 19 of *Lecture Notes in Computer Science*, pages 408–425. Springer-Verlag, 1974.

[Rey93] John C. Reynolds. The discoveries of continuations. *Lisp and Symbolic Computation: An International Journal*, 6:233–247, 1993. A history of continuations.

[Rey98] John C. Reynolds. *Theories of Programming Languages*. Cambridge University Press, 1998.

[RG94] Brian Reistad and David K. Gifford. Static dependent costs for estimating execution time. In *1994 ACM Conference on Lisp and Functional Programming*, pages 65–78. ACM, June 1994.

[Roz84] Guillermo J. Rozas. Liar, an Algol-like compiler for Scheme. Master's thesis, EECS Department, MIT, January 1984.

[Sab88] Gary W. Sabot. *The Paralation Model*. MIT Press, 1988.

[Sch86a] David Schmidt. *Denotational Semantics: A Methodology for Language Development*. Allyn and Bacon, 1986.

[Sch86b] David A. Schmidt. *Denotational Semantics: A Methodology for Language Development*. Allyn and Bacon, Newton, MA, 1986.

[Sch94] David Schmidt. *The Structure of Typed Programming Languages*. MIT Press, 1994.

[Sco77] Dana S. Scott. Logic and programming languages. *Communications of the ACM*, 20(9):634–641, September 1977. Turing Award Lecture.

[SF92] Amr Sabry and Matthias Felleisen. Reasoning about programs in continuation-passing style. In *Proceedings of the 1992 ACM Conference on Lisp and Functional Prgramming*, pages 288–298. ACM, 1992.

[SF93] Amr Sabry and Matthias Felleisen. Reasoning about programs in continuation-passing style. *Lisp and Symbolic Computation*, 6(3–4):289–360, 1993.

[SG90] Mark A. Sheldon and David K. Gifford. Static dependent types for first class modules. In *Symposium on Lisp and Functional Programming*. ACM, 1990.

[Sha97] Zhong Shao. An overview of the FLINT/ML compiler. In *Proc. 1997 ACM SIGPLAN Workshop on Types in Compilation (TIC'97)*, Amsterdam, The Netherlands, 1997.

[SS76] Guy L. Steele Jr. and Gerald Jay Sussman. LAMBDA: The Ultimate Imperative. Technical Report AIM-353, MIT Artificial Intelligence Laboratory, March 1976.

[Ste77] Guy L. Steele Jr. Debunking the "expensive procedure call" myth, or procedure call implementations considered harmful, or LAMBDA, the Ultimate Goto. Technical Report AIM-443, MIT Artificial Intelligence Laboratory, October 1977.

[Ste78] Guy Lewis Steele Jr. Rabbit: a compiler for Scheme. MIT AI Memo 474, Massachusetts Institute of Technology, Cambridge, Mass., May 1978.

[Sto77] Joseph E. Stoy. *Denotational Semantics: The Scott-Strachey Approach to Programming Language Theory*. MIT Press, 1977.

[Sto85] Joseph E. Stoy. *Denotational Semantics: The Scott-Strachey Approach to Programming Language Theory*. MIT Press, 1985.

[SW74] C. Strachey and C. Wadsworth. Continuations: A mathematical semantics which can deal with full jumps. Monograph PRG-11, Oxford University Computing Laboratory, Programming Research Group, Oxford, UK, 1974.

[SW97] Paul Steckler and Mitchell Wand. Lightweight closure conversion. *ACM Transactions on Programming Languages and Systems*, 19(1):48–86, January 1997.

[SW00] Christopher Strachey and Christopher P. Wadsworth. Continuations: A mathematical semantics which can deal with full jumps. *Higher-Order and Symbolic Computation*, 13(1–2):135–152, 2000.

[Ten76] R. D. Tennent. The denotational semantics of programming languages. *Communications of the ACM*, 19(8), August 1976.

[TMC+96] D. Tarditi, G. Morrisett, P. Cheng, C. Stone, R. Harper, and P. Lee. TIL: A type-directed optimizing compiler for ML. In *Programming Language Design and Implementation*. ACM, 1996.

[TO98] Andrew P. Tolmach and Dino Oliva. From ML to Ada: Strongly-typed language interoperability via source translation. *Journal of Functional Programming*, 8(4):367–412, 1998.

[Wad87] Philip Wadler. Views: A way for pattern matching to cohabit with data abstraction. In Muchnik Steve, editor, *Proceedings of the 14th Symposium on Principles of Programming Languages*, Munich, Germany, January 1987. ACM. Revised March 1987.

[Wad95] Philip Wadler. Monads for functional programming. In J. Jeuring and E Meijer, editors, *Advanced Functional Programming*, volume 925 of *Lecture Notes in Computer Science*. Springer-Verlag, 1995.

[Wel99] J. B. Wells. Typability and type checking in System F are equivalent and undecidable. *Annals of Pure and Applied Logic*, 98(1–3):111–156, 1999. Supersedes [**?**].