Bibliography

[AA93] Zena Ariola and Arvind. Graph rewriting systems: Capturing sharing of computation in language implementations. Computation Structure Group Memo 347, MIT Laboratory for Computer Science, April 1993.

[Ada91] Stephen Adams. *Modular Grammars for Programming Language Prototyping.* PhD thesis, Department of Electronics and Computer Science, University of Southampton, March 1991.

[Ame89] American National Standards Institute. *American National Standard for Information Systems Programming Language Fortran: S8(X3.9-198x)*, 1989.

[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.

[Ash86] E. A. Ashcroft. Dataflow and eduction: Data-driven and demand-driven distributed computing. In J. W. deBakker, W.-P. de Roever, and G. Rozenberg, editors, *Current Trends in Concurrency: Overviews and Tutorials*, pages 1–50. Springer-Verlag, 1986. Lecture Notes in Computer Science, Number 224.

[ASS85] Harold Abelson, Gerald Jay Sussman, and Julie Sussman. *Structure and Interpretation of Computer Programs.* MIT Press, 1985.

[ASU86] Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman. *Compilers: principles, techniques, and tools.* Addison-Wesley, 1986.

[Axe86] Tim S. Axelrod. Effects of synchronization barriers on multiprocessor performance. *Parallel Computing*, 3(2):129–140, May 1986.

[B$^+$87] H.P. Barendregt et al. Toward an intermediate language based on graph rewriting. In *PARLE: Parallel Architectures and Languages Europe, Volume 2*, pages 159 – 175. Springer-Verlag, 1987. Lecture Notes in Computer Science, Number 259.

[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.

[Bar84] H.P Barendregt. *The Lambda-calculus: Its Syntax and Semantics.* North-Holland, 1984.

[Bar92] Paul S. Barth. Atomic data structures for parallel computing. Technical Report MIT/LCS/TR-532, MIT Laboratory for Computer Science, March 1992.

[Baw86] Alan Bawden. Connection graphs. In *Symposium on Lisp and Functional Programming*, pages 258–265. ACM, August 1986.

[Baw92] Alan Bawden. *Linear Graph Reduction: Confronting the Cost of Naming.* PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, June 1992.

[Bel86] Francoise Bellegarde. Rewriting systems on FP expressions that reduce the number of sequences yielded. *Science of Computer Programming*, 6(1):11–34, January 1986.

[Bir84] R. S. Bird. Using circular programs to eliminate multiple traversals of data. *Acta Informatica*, pages 239–250, 1984.

[Bir86] Richard S. Bird. An introduction to the theory of lists. In Manfred Broy, editor, *Logic of Programming and Calculi of Discrete Design (NATO ASI Series, Vol. F36)*, pages 5–42. Springer-Verlag, 1986.

[Bir88] Richard S. Bird. Lectures on constructive functional programming. In Manfred Broy, editor, *Constructive Methods in Computing Science (NATO ASI Series, Vol. F55)*, pages 5–42. Springer-Verlag, 1988.

[Bir89a] R. S. Bird. Algebraic identities for program calculation. *The Computer Journal*, pages 122–126, 1989.

[Bir89b] Andrew Birrel. An introduction to programming with threads. SRC Report 35, Digital Equipment Corporation, January 1989.

[BL93] Robert D. Blumofe and Charles E. Leiserson. Space-efficient scheduling of multithreaded computations. In *25th Annual ACM Symposium on Theory of Computing.* ACM, May 1993.

[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.

[Bud88] Timothy Budd. *An APL Compiler.* Springer-Verlag, 1988.

[CBF91] Siddhartha Chatterjee, Guy E. Blelloch, and Allan Fisher. Size and access inference for data-parallel programs. In *Programming Language Design and Implementation '91*, pages 130–144. ACM, 1991.

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

[Chi92] Wei-Ngan Chin. Safe fusion of functional expressions. In *Symposium on Lisp and Functional Programming*, pages 11–20. ACM, 1992.

[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.

[Coh83] Norman H. Cohen. Eliminating redundant recursive calls. *ACM Transactions on Programming Languages and Systems*, 5(3):265–299, July 1983.

[CR$^+$91] William Clinger, Jonathan Rees, et al. Revised^4 report on the algorithmic language Scheme. *Lisp Pointers*, 4(3), 1991.

[Dar82] John Darlington. Program transformation. In J. Darlington, P. Henderson, and D. A. Turner, editors, *Functional Programming and its Applications*, pages 177–192. 1982.

[DC90] G. D. P. Dueck and G. V. Cormack. Modular attribute grammars. *The Computer Journal*, 33(2):164–172, April 1990.

[Den75] Jack B. Dennis. First version of a data flow procedure language. Computation Structure Group Memo MIT/LCS/TM-61, MIT Laboratory for Computer Science, May 1975.

[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.

[DJL88] Pierre Deransart, Martin Jordan, and Bernard Lorho. *Atribute Grammars*. Springer-Verlag, 1988. Lecture Notes in Computer Science, Number 323.

[DK82] Alan L. Davis and Robert M. Keller. Data flow program graphs. *IEEE Computer*, pages 26–41, February 1982.

[DR76] John Darlington and R.M.Burstall. A system which automatically improves programs. *Acta Informatica*, pages 41–60, 1976.

[FMY92] R. Farrow, T. J. Marlowe, and D. M. Yellin. Composable attribute grammars: Support for modularity in translator design and implmentation. In *Nineteenth Annual ACM Symposium on Principles of Programming Languages*, pages 223–234, 1992.

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

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

[GJSO92] David Gifford, Pierre Jouvelot, Mark 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.

[GLJ93] Andrew Gill, John Launchbury, and Simon L. Peyton Jones. A short cut to deforestation. In *Functional Programming and Computer Architecture*, 1993.

[GM84] Richard P. Gabriel and John McCarthy. Queue-based multi-processing Lisp. In *Symposium on Lisp and Functional Programming*, pages 25–44. ACM, August 1984.

[GW78] Leo J. Guibas and Douglas K. Wyatt. Compilation and delayed evaluation in APL. In *Conference Record of the Fifth ACM Conference on the Principles of Programming Languages*, pages 1–8, 1978.

[HA87] Paul Hudak and Steve Anderson. Pomset interpretation of parallel functional programs. In *Functional Programming Languages and Computer Architecture*, pages 234–256, September 1987. Lecture Notes in Computer Science, Number 274.

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

[Hen80] Peter Henderson. *Functional Programming: Application and Implementation.* Prentice-Hall, 1980.

[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.

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

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

[HS86a] W. Daniel Hillis and Guy L. Steele Jr. Data parallel algorithms. *Communications of the ACM*, 29(12), 1986.

[HS86b] Paul Hudak and Lauren Smith. Para-functional programming: A paradigm for programming multiprocessor systems. In *Thirteenth Annual ACM Symposium on Principles of Programming Languages*, pages 243–254, January 1986.

[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.

[Hug83] R. J. M. Hughes. *The Design and Implementation of Programming Languages*. PhD thesis, Oxford Universiy Computing Laboratory, Programming Research Group, July 1983.

[Hug84] R. J. M. Hughes. Parallel functional languages use less space. Technical report, Oxford University Programming Research Group, 1984.

[Hug85] R. J. M. Hughes. Lazy memo-functions. In *Functional Programming Languages and Computer Architecture*, pages 129–146, 1985. Lecture Notes in Computer Science, Number 201.

[Hug90] R. J. M. Hughes. Why functional programming matters. In David Turner, editor, *Research Topics in Functional Programming*, pages 17–42. Addison Wesley, 1990.

[Ive87] Kenneth E. Iverson. A dictionary of APL. *APL QUOTE QUAD*, 18(1):5–40, September 1987.

[Jay] C. Barry Jay. Personal correspondence.

[JG89] Pierre Jouvelot and David K. Gifford. Communication effects for message-based concurrency. Technical Report MIT/LCS/TM-386, MIT Laboratory for Computer Science, February 1989.

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

[Joh87] Thomas Johnsson. Attribute grammars as a functional programming paradigm. In *Functional Programming Languages and Computer Architecture*, pages 154–173, 1987. Lecture Notes in Computer Science, Number 274.

[K$^+$86] David Kranz et al. Orbit: An optimizing compiler for Scheme. In *Proceedings of SIGPLAN '86 Symposium on Compiler Construction*, pages 219–233. ACM, June 1986.

[Kat84] Takuya Katayama. Translation of attribute grammars into procedures. *Transactions on Programming Languages and Systems*, 6(3):345–369, 1984.

[Knu68] Donald E. Knuth. Semantics of context-free languages. *Mathematical Systems Theory*, 2(2):127–145, 1968.

[Knu73] Donald E. Knuth. *The Art of Computer Programming. 2nd ed. Vol. 1: Fundamental algorithms.* Addison-Wesley, 1973.

[Kos84] Kai Koskimes. A specification language for one-pass semantic analysis. In *ACM SIGPLAN '84 Symposium on Compiler Construction*, pages 179–189, Tama-City, Tokyo, June 1984.

[KP84] Brian W. Kernighan and Rob Pike. *The UNIX Programming Environment.* Prentice-Hall, Englewood Cliffs, NJ, 1984.

[KW92] U. Kastens and W. M. Waite. Modularity and reusability in attribute grammars. Technical Report CU-CS-613-92, University of Colorado at Boulder, September 1992.

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

[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.

[Mey] Albert R. Meyer. A substitution model for Scheme: Formal definitions. In preparation.

[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.

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

[PA85] Keshav Pingali and Arvind. Efficient demand-driven evaluation (I). *ACM Transactions on Programming Languages and Systems*, 7(2):311–333, April 1985.

[PA86] Keshav Pingali and Arvind. Efficient demand-driven evaluation (II). *ACM Transactions on Programming Languages and Systems*, 8(1):109–139, January 1986.

[Pet77] James L. Peterson. Petri nets. *ACM Computing Surveys*, 9(3):223–250, 1977.

[Pet84] Alberto Pettorossi. A powerful strategy for deriving efficient programs by transformation. In *ACM Conference on Lisp and Functional Programming*, pages 273–281, 1984.

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

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

[Pra86] V. R. Pratt. Modeling concurrency with partial orders. *International Journal of Parallel Programming*, 15(1):33–71, February 1986.

[RAM83] Jonathan Rees, Norman I. Adams, and James R. Meehan. The T manual (third edition). Technical report, Yale University Department of Computer Science, March 1983.

[Ric81] Charles Rich. Inspection methods in programming. Technical Report AI-TR-604, MIT Artificial Intelligence Laboratory, June 1981.

[Rob65] J. A. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12(1):23–41, 1965.

[RS87] J. R. Rose and Guy L. Steele Jr. C*: An extended C language for data parallel programming. In *Proceedings of the Second International Conference on Supercomputing, Vol 2*, pages 2–16, 1987.

[RW90] Charles Rich and Richard C. Waters. *The Programmer's Apprentice*. Addison-Wesley, 1990.

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

[Sme93] Jacobus Edith Willem (Sjaak) Smetsers. *Graph Rewriting and Functional Languages*. PhD thesis, Katholieke Universiteit Nijmegan, February 1993.

[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.

[Ste90] Guy L. Steele Jr. *Common Lisp: The Language*. Digital Press, 1990.

[Str71] H. R. Strong. Translating recursion equations into flow charts. *Journal of Computer and System Sciences*, 5:254–285, 1971.

[Tra88] Kenneth R. Traub. Sequential implementation of lenient programming languages. Technical Report MIT/LCS/TR-417, MIT Laboratory for Computer Science, October 1988.

[Tur79] D. A. Turner. A new implementation for applicative languages. *Software – Practice and Experience*, 9:31–49, 1979.

[Tur85] D. A. Turner. Miranda: A non-strict functional language with polymorphic types. In *Functional Programming Languages and Computer Architecture*, pages 1–16, 1985. Lecture Notes in Computer Science, Number 201.

[Tur94] Franklyn Turbak. Slivers: Computational modularity via synchronized lazy aggregates. Technical Report AI-TR-1466, MIT Artificial Intelligence Laboratory, 1994. Revised version of doctoral dissertation. In prepartion.

[WA85] William W. Wadge and Edward A. Ashcroft. *Lucid, the Dataflow Programming Language*. Academic Press, 1985.

[Wad84] Philip Wadler. Listlessness is better than laziness: Lazy evaluation and garbage collection at compile-time. In *ACM Symposium On Lisp and Functional Programming*, pages 45–52, 1984.

[Wad85] Philip Wadler. Listlessness is better than laziness II: Composing listless functions. In *Programs As Data Objects*, pages 282–305. Springer-Verlag, 1985. Lecture Notes in Computer Science 217.

[Wad88] Philip Wadler. Deforestation: Transforming programs to eliminate trees. In *2nd European Symposium on Programming*, pages 344–358, 1988. Lecture Notes in Computer Science, Number 300.

[Wat] Richard C. Waters. To NReverse when consing a list or by pointer manipulation to avoid it; that is the question. *Lisp Pointers (to appear).*

[Wat78] Richard C. Waters. Automatic analysis of the logical structure of programs. Technical Report AI-TR-492, MIT Artificial Intelligence Laboratory, December 1978.

[Wat79] Richard C. Waters. A method for analyzing loop programs. *IEEE Transactions on Software Engineering*, SE-5(3):237–247, May 1979.

[Wat84] Richard C. Waters. Expressional loops. In *ACM Symposium on the Principles of Programming Languages*, pages 1–10, 1984.

[Wat87] Richard C. Waters. Efficient interpretation of synchronizable series expressions. In *ACM SIGPLAN '87 Symposium on Interpreters and Interpretive Techniques*, volume SE-5, pages 74–85, 1987.

[Wat90] Richard C. Waters. Series. In Guy L. Steele Jr., editor, *Common Lisp: The Language*, pages 923–955. Digital Press, 1990.

[Wat91] Richard C. Waters. Automatic transformation of series expressions into loops. *ACM Transactions on Programming Languages and Systems*, 13(1):52–98, January 1991.

[Wat92] D. A. Watt. Modular description of programming languages. *The Computer Journal*, 35:A009–A0028, 1992.

[WCRS91] Daniel Weise, Roland Conybeare, Erik Ruf, and Scott Seligman. Automatic online partial evaluation. In *Proceedings of the Conference on Functional Programming Languages and Computer Architecture*. Springer-Verlag, August 1991.

[WS73] S. A. Walker and H. R. Strong. Characterizations of flowchartable recursions. *Journal of Computer and System Sciences*, 7:404–447, 1973.