Bibliography

[1] Kavi Arya. A functional animation starter-kit. *Journal of Functional Programming*, 4(1):1–18, January 1994.

[2] Lex Augusteijn. Sorting morphisms. In S. D. Swierstra, P. R. Henriques, and J. N. Oliveira, editors, *Advanced Functional Programming*, volume 1608 of *Lecture Notes in Computer Science*, pages 1–27. Springer-Verlag, 1999.

[3] Lennart Augustsson. Cayenne: A language with dependent types. *SIGPLAN Notices*, 34(1):239–250, January 1999.

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

[5] Roland Backhouse, Patrik Jansson, Johan Jeuring, and Lambert Meertens. Generic programming: An introduction. In S. D. Swierstra, P. R. Henriques, and J. N. Oliveira, editors, *Advanced Functional Programming*, volume 1608 of *Lecture Notes in Computer Science*, pages 28–115. Springer-Verlag, Berlin, 1999.

[6] Denis Baggi. *Computer-Generated Music*. IEEE Computer Society Press, Las Alamitos, CA, 1992.

[7] Joel F. Bartlett. Don't fidget with widgets, draw! Technical Report 6, DEC Western Digital Laboratory, May 1991.

[8] Kenneth E. Batcher. Sorting networks and their applications. In *AFIPS Spring Joint Conference*, pages 307–314, 1968.

[9] Gérard Berry and Georges Gonthier. The Esterel synchronous programming language: Design, semantics, implementation. *Science of Computer Programming*, 19(2):87–152, 1992.

[10] A. S. Bhandal, V. Considine, and G. E. Dixon. An array processor for video picture motion estimation. In J. McCanny, J. McWhirter, and E. Swartzlander, editors, *Systolic Array Processors*, pages 369–378. Prentice-Hall International, 1989.

[11] Richard Bird. An introduction to the theory of lists. In M. Broy, editor, *Proceedings of the NATO Advanced Study Institute on Logic of Programming and Caculi of Discrete Design*. Springer Verlag, June 1987.

[12] Richard Bird and Oege de Moor. *Algebra of Programming*. Prentice Hall, 1997.

[13] Richard Bird and John Hughes. The alpha–beta algorithm: an exercise in program transformation. *Information Processing Letters*, 24(1):53–57, January 1987.

[14] Richard S. Bird. The promotion and accumulation strategies in transformational programming. *ACM Transactions on Programming Languages and Systems*, 6(4):487–504, October 1984. Addendum in 7(3), p. 490–492.

[15] Richard S. Bird. *Introduction to Functional Programming using Haskell*. Prentice Hall Europe, 2nd edition, 1998.

[16] Richard S. Bird and Jeremy Gibbons. Arithmetic coding with folds and unfolds. In Johan Jeuring and Simon Peyton Jones, editors, *Advanced Functional Programming*. Springer-Verlag, 2002. To appear.

[17] Per Bjesse, Koen Claessen, Mary Sheeran, and Satnam Singh. Lava: Hardware design in Haskell. In *International Conference on Functional Programming*. ACM, 1998.

[18] Phelim Boyle, Mark Broadie, and Paul Glasserman. Monte Carlo methods for security pricing. *Journal of Economic Dynamics and Control*, 21:1267–1321, 1997.

[19] Paul Caspi, Daniel Pilaud, Nicholas Halbwachs, and John A. Plaice. LUSTRE: A declarative language for programming synchronous systems. In *14th ACM Symposium on Principles of Programming Languages*, pages 178–188, Munich, 1987.

[20] James Cheney and Ralf Hinze. A lightweight implementation of generics and dynamics. In Manuel M.T. Chakravarty, editor, *Proceedings of the 2002 ACM SIGPLAN Haskell Workshop*, October 2002.

[21] Olaf Chitil. Pretty printing with lazy dequeues. In *ACM SIGPLAN Haskell Workshop*, pages 183–201, Firenze, Italy, 2001. Universiteit Utrecht UU-CS-2001-23.

[22] Seonghun Cho and Sartaj Sahni. Weight biased leftist trees and modified skip lists. In *International Computing and Combinatorics Conference*, pages 361–370, June 1996.

[23] Koen Claessen and John Hughes. QuickCheck: a lightweight tool for random testing of Haskell programs. In *International Conference on Functional Programming*, pages 268–279. ACM, 2000.

[24] Koen Claessen and John Hughes. Testing Monadic Code with QuickCheck. In *Haskell Workshop*. ACM SIGPLAN, 2002.

[25] Koen Claessen and David Sands. Observable sharing for functional circuit description. In *Asian Computer Science Conference*, pages 62–73, Phuket, Thailand, 1999. ACM SIGPLAN.

[26] William F. Clocksin and Christopher S. Mellish. *Programming in Prolog*. Springer Verlag, second edition, 1984.

[27] Antony Courtney and Conal Elliott. Genuinely functional user interfaces. In *Haskell Workshop*, pages 41–69, 2001.

[28] John C. Cox, Stephen A. Ross, and Mark Rubinstein. Option pricing: a simplified approach. *Journal of Financial Economics*, 7:229–263, 1979.

[29] Clark Allan Crane. *Linear lists and priority queues as balanced binary trees*. PhD thesis, Computer Science Department, Stanford University, February 1972. Available as STAN-CS-72-259.

[30] Olivier Danvy. Functional unparsing. *Journal of Functional Programming*, 8(6):621–625, November 1998.

[31] Olivier Danvy, Morten Rhiger, and Kristoffer H. Rose. Normalization by evaluation with typed abstract syntax. *Journal of Functional Programming*, 11(6):673–680, November 2001.

[32] Oege de Moor and Ganesh Sittampalam. Generic program transformation. In *Third International Summer School on Advanced Functional Programming*, volume 1608 of *Lecture Notes in Computer Science*, pages 116–149. Springer-Verlag, 1998.

[33] Oege de Moor and Ganesh Sittampalam. Higher-order matching for program transformation. *Theoretical Computer Science*, 269:135–162, 2001.

[34] Conal Elliott. An embedded modeling language approach to interactive 3D and multimedia animation. *IEEE Transactions on Software Engineering*, 25(3):291–308, May/June 1999. Special Section: Domain-Specific Languages (DSL).

[35] Conal Elliott, Sigbjørn Finne, and Oege de Moor. Compiling embedded languages. *Journal of Functional Programming*, 2001. To appear.

[36] Conal Elliott and Paul Hudak. Functional reactive animation. In *International Conference on Functional Programming*, pages 263–273, 1997.

[37] Levent Erkök and John Launchbury. Recursive monadic bindings. In *International Conference on Functional Programming*, pages 174–185, 2000.

[38] Sigbjorn Finne and Simon Peyton Jones. Pictures: A simple structured graphics model. In *Glasgow Functional Programming Workshop*, Ullapool, July 1995.

[39] Alexandre Frey, Gérard Berry, Patrice Bertin, François Bourdoncle, and Jean Vuillemin. Jazz. Available from `http://www.cma.ensmp.fr/jazz`, 1998.

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

[41] Jeremy Gibbons. *Algebras for Tree Algorithms.* D. Phil. thesis, Programming Research Group, Oxford University, 1991. Available as Technical Monograph PRG-94.

[42] Jeremy Gibbons. Deriving tidy drawings of trees. *Journal of Functional Programming*, 6(3):535–562, 1996.

[43] Jeremy Gibbons. A pointless derivation of radixsort. *Journal of Functional Programming*, 9(3):339–346, 1999.

[44] Jeremy Gibbons. Calculating functional programs. In Roland Backhouse, Roy Crole, and Jeremy Gibbons, editors, *Algebraic and Coalgebraic Methods in the Mathematics of Program Construction*, volume 2297 of *Lecture Notes in Computer Science*, pages 148–203. Springer-Verlag, 2002.

[45] Jeremy Gibbons and Geraint Jones. The under-appreciated unfold. In *International Conference on Functional Programming*, pages 273–279, September 1998.

[46] Andrew Gill, John Launchbury, and Simon Peyton Jones. A short cut to deforestation. In *Functional Programming Languages and Computer Architecture*, pages 223–232, 1993.

[47] Carlos Gonzalía. Análisis asintótico amortizado en lenguajes funcionales perezosos. In *Latin-American Conference on Functional Programming*, October 1997.

[48] Nicholas Halbwachs, Fabienne Lagnier, and Pascal Raymond. Synchronous observers and the verification of reactive systems. In *Algebraic Methodology and Software Technology*, pages 83–96. Springer Verlag, 1993.

[49] Nicolas Halbwachs, Paul Caspi, Pascal Raymond, and Daniel Pilaud. The synchronous dataflow programming language LUSTRE. *Proceedings of the IEEE*, 79(9):1305–1320, September 1991.

[50] Richard Hamlet. Random testing. In J. Marciniak, editor, *Encyclopedia of Software Engineering*, pages 970–978. Wiley, 1994.

[51] Michael Hanus, Herbert Kuchen, and Juan Jose Moreno-Navarro. Curry: A truly functional logic language. In *ILPS'95 Workshop on Visions for the Future of Logic Programming*, pages 95–107, 1995.

[52] Peter Henderson. Functional geometry. In *ACM Symposium on LISP and Functional Programming*, pages 179–187, 1982.

[53] Ralf Hinze. A new approach to generic functional programming. In Thomas W. Reps, editor, *Proceedings of the 27th Symposium on Principles of Programming Languages*, pages 119–132, January 2000.

[54] Ralf Hinze. Functional Pearl: Formatting: a class act. *J. Functional Programming*, 2002. to appear.

[55] Thomas Ho and Sang-Bin Lee. Term Structure Movements and Pricing Interest Rate Contingent Claims. *Journal of Finance*, 41:1011–1028, 1986.

[56] Douglas R. Hofstadter. *Gödel, Escher, Bach: an Eternal Golden Braid*. Basic Books, New York, 1979.

[57] Gerard J. Holzmann. *Beyond Photography — the Digital Darkroom*. Prentice-Hall, Englewood Cliffs, New Jersey, 1988.

[58] Paul Hudak. Building domain-specific embedded languages. *ACM Computing Surveys*, 28, December 1996.

[59] Paul Hudak. Haskore music tutorial. In *Second International School on Advanced Functional Programming*, pages 38–68. Springer Verlag, LNCS 1129, August 1996.

[60] Paul Hudak. Modular domain specific languages and tools. In P. Devanbu and J. Poulin, editors, *Fifth International Conference on Software Reuse*, pages 134–142. IEEE Computer Society Press, 1998.

[61] Paul Hudak. *The Haskell School of Expression: Learning Functional Programming through Multimedia*. Cambridge University Press, New York, 2000.

[62] Paul Hudak and Jonathan Berger. A model of performance, interaction, and improvisation. In *Proceedings of International Computer Music Conference*. International Computer Music Association, 1995.

[63] Paul Hudak and Mark P. Jones. Haskell vs. Ada vs. C++ vs Awk vs . . . : An experiment in software prototyping productivity. Technical report, Yale, 1994.

[64] Paul Hudak, Tom Makucevich, Syam Gadde, and Bo Whong. Haskore music notation: An algebra of music. *Journal of Functional Programming*, 6(3):465–483, May 1996.

[65] John Hughes. A novel representation of lists and its application to the function ‘reverse’. *Information Processing Letters*, 22:141–144, 1986.

[66] John Hughes. The design of a pretty-printer library. In Johan Jeuring and Erik Meijer, editors, *Advanced Functional Programming*, volume 925 of *LNCS*. Springer, 1995.

[67] John Hughes. Generalising monads to arrows. *Science of Computer Programming*, 37:67–111, May 2000.

[68] MIDI 1.0 detailed specification: Document version 4.1.1, February 1990.

[69] Michael A. Jackson. *Principles of Program Design.* Academic Press, 1975.

[70] Patrik Jansson and Johan Jeuring. PolyP—a polytypic programming language extension. In *Conference Record 24th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'97), Paris, France*, pages 470–482. ACM Press, January 1997.

[71] Steven Johnson. *Synthesis of Digital Designs from Recursion Equations.* The ACM Distinguished Dissertation Series, The MIT Press, 1984.

[72] Geraint Jones and Mary Sheeran. Collecting butterflies. Technical Monograph PRG-91, Oxford University Computing Laboratory, Programming Research Group, February 1991.

[73] Geraint Jones and Mary Sheeran. The study of butterflies. In Graham Birtwistle, editor, *Proc. 4th Banff Workshop on Higher Order*. Springer Workshops in Computing, 1991.

[74] Geraint Jones and Mary Sheeran. Circuit design in Ruby. In J. Staunstrup, editor, *Formal Methods for VLSI Design.* North Holland, 1992.

[75] Geraint Jones and Mary Sheeran. Designing arithmetic circuits by refinement in Ruby. In R. Bird, C. Morgan, and J. Woodcock, editors, *Mathematics of Program Construction*, volume 669 of *Lecture Notes in Computer Science*, pages 208–232. Springer, 1993.

[76] Geraint Jones and Mary Sheeran. Designing Arithmetic Circuits by Refinement in Ruby. *Science of Computer Programming*, 22(1-2), 1994.

[77] Simon Peyton Jones. Haskell pretty-printer library. `http://www.haskell.org/libraries/#prettyprinting`, 1997.

[78] André Joyal, Ross Street, and Dominic Verity. Traced monoidal categories. *Mathematical Proceedings of the Cambridge Philosophical Society*, 119(3):447–468, 1996.

[79] Jerzy Karczmarczuk. Functional approach to texture generation. In Shriram Krishnamurthi and C. R. Ramakrishnan, editors, *Practical Aspects of Declarative Languages*, volume 2257 of *Lecture Notes in Computer Science*, pages 225–242. Springer, 2002.

[80] Donald E. Knuth. *Searching and Sorting*, volume 3 of *The Art of Computer Programming.* Addison-Wesley, 1973.

[81] Richard E. Ladner and Michael J. Fischer. Parallel prefix computation. *Journal of the ACM*, 27:831–838, 1980.

[82] D. Lahti. Applications of a functional programming language to hardware synthesis. Master's thesis, UCLA, 1980.

[83] Ralf Lämmel and Simon Peyton Jones. Scrap your boilerplate: a practical approach to generic programming. Available from `http://research.microsoft.com/~simonpj/papers/hmap/`, 2002.

[84] John Launchbury, Jeff Lewis, and Byron Cook. On embedding a microarchitecture design language within Haskell. In *International Conference on Functional Programming*. ACM, 1999.

[85] John Launchbury and Tim Sheard. Warm fusion: Deriving build-catas from recursive definitions. In *Functional Programming Languages and Computer Architecture*, pages 314–323. ACM Press, 1995.

[86] Daan Leijen and Erik Meijer. Domain-specific embedded compilers. In *Proceedings of the 2nd Conference on Domain-Specific Languages*, pages 109–122, Berkeley, CA, October 1999. USENIX Association.

[87] Peter Lucas and Stephen N. Zilles. Graphics in an applicative context. Technical report, IBM Almaden Research Center, 650 Harry Road, San Jose, CA 95120-6099, July 8 1987.

[88] Wayne Luk. Systematic serialisation of array-based architectures. *Integration, the VLSI Journal*, 14(3), February 1993.

[89] Wayne Luk, Geraint Jones, and Mary Sheeran. Computer-based tools for regular array design. In J McCanny, J McWhirter, and E Swartzlander, editors, *Systolic Array Processors*, pages 589 – 598. Prentice-Hall International, 1989.

[90] John Maeda. *Design By Numbers*. MIT Press, May 1999.

[91] John Matthews and John Launchbury. Elementary microarchitecture algebra. In Nicolas Halbwachs and Doron Peled, editors, *Computer Aided Verification*, volume 1633 of *Lecture Notes in Computer Science*, pages 333–360. Springer, 1999.

[92] Lambert Meertens. Paramorphisms. *Formal Aspects of Computing*, 4(5):413–424, 1992.

[93] Erik Meijer, Maarten Fokkinga, and Ross Paterson. Functional programming with bananas, lenses, envelopes and barbed wire. In John Hughes, editor, *Functional Programming Languages and Computer Architecture*, volume 523 of *Lecture Notes in Computer Science*, pages 124–144. Springer-Verlag, 1991.

[94] Erik Meijer and Graham Hutton. Bananas in space: Extending fold and unfold to exponential types. In *Functional Programming Languages and Computer Architecture*, 1995.

[95] Jayadev Misra. Powerlist: A structure for parallel recursion. *ACM Transactions on Programming Languages and Systems*, 16(6):1737–1767, November 1994.

[96] Marek Musiela and Marek Rutkowski. *Martingale Methods in Financial Modelling*. Springer, 1997.

[97] John O'Donnell. Hydra: Hardware description in a functional language using recursion equations and higher order combining forms. In G. J.

Milne, editor, *The Fusion of Hardware Design and Verification*, pages 363–382. North-Holland, 1988.

[98] John O'Donnell. From transistors to computer architecture: Teaching functional circuit specification in Hydra. In *Functional Programming Languages in Education*, volume 1022 of *Lecture Notes in Computer Science*, pages 195–214. Springer-Verlag, 1996.

[99] Chris Okasaki. Amortization, lazy evaluation, and persistence: Lists with catenation via lazy linking. In *IEEE Symposium on Foundations of Computer Science*, pages 646–654, October 1995.

[100] Chris Okasaki. Simple and efficient purely functional queues and deques. *Journal of Functional Programming*, 5(4):583–592, 1995.

[101] Chris Okasaki. The role of lazy evaluation in amortized data structures. In *ACM SIGPLAN International Conference on Functional Programming*, pages 62–72, May 1996.

[102] Chris Okasaki. *Purely Functional Data Structures.* Cambridge University Press, 1998.

[103] Yoshiyuki Onoue, Zhenjiang Hu, Hideya Iwasaki, and Masato Takeichi. A calculational fusion system HYLO. In Richard S. Bird and Lambert Meertens, editors, *Algorithmic Languages and Calculi*, pages 76–106. Chapman and Hall, 1997.

[104] Derek Oppen. Pretty-printing. *ACM Transactions on Programming Languages and Systems*, 2(4):465–483, 1980.

[105] Dorab Patel, Martine D. F. Schlag, and Milos D. Ercegovac. νFP: An environment for the multi-level specification, analysis, and synthesis of hardware algorithms. In *Functional Programming Languages and Computer Architecture*, volume 201 of *Lecture Notes in Computer Science*, pages 238–255. Springer-Verlag, 1985.

[106] Ross Paterson. A new notation for arrows. In *International Conference on Functional Programming*, pages 229–240. ACM Press, September 2001.

[107] John Power and Edmund Robinson. Premonoidal categories and notions of computation. *Mathematical Structures in Computer Science*, 7(5):453–468, October 1997.

[108] Vaughan Pratt. *Shellsort and Sorting Networks.* PhD thesis, Stanford University, 1971. Also Garland, New York, 1979.

[109] Daniel Revuz and Marc Yor. *Continuous Martingales and Brownian Motion.* Springer, 1991.

[110] Curtis Roads, editor. *The Music Machine (Selected Readings from Computer Music Journal).* MIT Press, Cambridge, MA, 1989.

[111] Edward Rothstein. *Emblems of Mind: The Inner Life of Music and Mathematics.* Times Books, New York, 1995.

[112] Robert Sedgewick. Analysis of Shellsort and related algorithms. In *European Symposium on Programming*, 1996.

[113] Eleanor Selfridge-Field, editor. *Beyond MIDI (The Handbook of Musical Codes).* MIT Press, Cambridge, MA, 1997.

[114] Silvija Seres. *The Algebra of Logic Programming.* D.Phil., Programming Research Group, University of Oxford, 2001.

[115] Mary Sheeran. μFP, an algebraic VLSI design language. D.Phil., Programming Research Group, Oxford University, 1983.

[116] Mary Sheeran. Puzzling permutations. In P. Trinder, editor, *Glasgow Functional Programming Workshop*, 1996.

[117] Donald L. Shell. A high-speed sorting procedure. *Communications of the ACM*, 2(7):30–32, 1959.

[118] Karl Sims. Artificial evolution for computer graphics. *ACM Computer Graphics*, 25(4):319–328, July 1991.

[119] Ganesh Sittampalam and Oege de Moor. Higher-order pattern matching for automatically applying fusion transformations. In O. Danvy and A. Filinski, editors, *Second Symposium on Programs as Data Objects*, volume 2053 of *Lecture Notes in Computer Science*, pages 198–217. Springer-Verlag, 2001.

[120] Daniel D. K. Sleator and Robert E. Tarjan. Self-adjusting heaps. *SIAM Journal on Computing*, 15(1):52–69, February 1986.

[121] Alvy Ray Smith. Image compositing fundamentals. Technical Report Technical Memo #4, Microsoft, July 1995. `http://www.alvyray.com/Memos`.

[122] Zoltan Somogyi, Fergus Henderson, and Thomas Conway. Mercury: An efficient purely declarative logic programming language. In *Proceedings of the Australian Computer Science Conference*, pages 499–512, Glenelg, Australia, 1995.

[123] J. Michael Spivey. Unification: A case-study in data refinement,. *Formal Aspects of Computing*, 7(2):158–168, 1995.

[124] J. Michael Spivey. Combinators for breadth-first search. *Journal of Functional Programming*, 10(4):397–408, 2000.

[125] Leon Sterling and Ehud Y. Shapiro. *The Art of Prolog.* MIT Press, second edition, 1994.

[126] S. Doaitse Swierstra and Luc Duponcheel. Deterministic, error-correcting combinator parsers. In John Launchbury, Erik Meijer, and Tim Sheard, editors, *Advanced Functional Programming*, volume 1129 of *Lecture Notes in Computer Science*, pages 184–207. Springer, 1996.

[127] Robert E. Tarjan. Amortized computational complexity. *SIAM Journal on Algebraic and Discrete Methods*, 6(2):306–318, April 1985.

[128] Hervé Touati and Mark Shand. PamDC: a C++ library for the simulation and generation of Xilinx FPGA designs. Available from `http://research.compaq.com/SRC/pametta/PamDC.pdf`, 1999.

[129] David A. Turner. An overview of Miranda. *SIGPLAN Notices*, 21(12):158–166, 1986.

[130] Arie van Deursen and Paul Klint. Little languages: little maintenance? *Journal of Software Maintenance*, 10:75–92, 1998.

[131] Arie van Deursen, Paul Klint, and Joost Visser. Domain-specific languages: an annotated bibliography. Technical report, Centrum voor Wiskunde en Informatica, Amsterdam, 2000.

[132] Varmo Vene. *Categorical Programming with Inductive and Coinductive Types*. PhD thesis, University of Tartu, 2000.

[133] Varmo Vene and Tarmo Uustalu. Functional programming with apomorphisms (corecursion). *Proceedings of the Estonian Academy of Sciences: Physics, Mathematics*, 47(3):147–161, 1998. 9th Nordic Workshop on Programming Theory.

[134] Jean Vuillemin. On circuits and numbers. *IEEE Transactions on Computers*, 43:8:868–879, 1994.

[135] Jean Vuillemin, Patrice Bertin, Didier Roncin, Mark Shand, Hervé Touati, and Philippe Boucard. Programmable Active Memories: the Coming of Age. *IEEE Trans. on VLSI*, 4(1), March 1996.

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

[137] Philip Wadler. Deforestation: Transforming programs to eliminate trees. *Theoretical Computer Science*, 73:231–248, 1990.

[138] Philip L. Wadler. How to replace failure by a list of successes. In J.-P. Jouannaud, editor, *Functional Programming Languages and Computer Architecture*, volume 201 of *Lecture Notes in Computer Science*, pages 113–128. Springer-Verlag, 1985.

[139] Paul Willmot, Jeff N. Dewynne, and Sam D. Howison. *Option Pricing: Mathematical Models and Computation*. Oxford Financial Press, 1993.

[140] Stephen N. Zilles, Peter Lucas, T.M. Linden, Jeff B. Lotspiech, and A.R. Harbury. The Escher document imaging model. In *ACM Conference on Document Processing Systems*, pages 159–168, December 5–9 1988.