Bibliography

[1] M. Abadi and L. Lamport. The existence of refinement mappings. *Theor. Comput. Sci.*, 82(2):253–284, 1991.

[2] J.H. Anderson, S. Ramamurthy, and K. Jeffay. Real-time computing with lock-free shared objects. *ACM Trans. Comput. Syst.*, 15(2):134–165, 1997.

[3] H. Attiya, A. Bar-Noy, D. Dolev, D. Peleg, and R. Reischuk. Renaming in an asynchronous environment. *J. ACM*, 37(3):524–548, 1990.

[4] H. Azatchi, Y. Levanoni, H. Paz, and E. Petrank. An on-the-fly mark and sweep garbage collector based on sliding views. In *Proceedings of the 18th ACM SIGPLAN conference on Object-oriented programing, systems, languages, and applications*, pages 269–281. ACM Press, 2003.

[5] R.J.R. Back and J. von Wright. Stepwise refinement of distributed systems: Models, formalism, correctness: Refinement calculus. In J.W. de Bakker, W.-P. de Roever, and G. Rozenberg, editors, *Stepwise Refinement of Distributed Systems*, volume 430 of *Lecture Notes in Computer Science*, pages 42–93. Springer-Verlag, 1990.

[6] G. Barnes. A method for implementing lock-free data structures. In *Proceedings of the 5th ACM Symposium on Parallel Algorithms and Architectures*, pages 261–270, June 1993.

[7] A. Bas-Noy and D. Dolev. Shared-memory vs. message-passing in an asynchronous distributed environment. In *Proceedings of the eighth annual ACM Symposium on Principles of distributed computing*, pages 307–318, 1989.

[8] M. Ben-Ari. Algorithms for on-the-fly garbage collection. *ACM Transactions on programming Languages and Systems*, 6(3):333–344, 1984.

[20] H. Gao, J.F. Groote, and W.H. Hesselink. Almost wait-free resizable hashtables (extended abstract). In *Proceedings of 18th International Parallel & Distributed Processing Symposium (IPDPS)*. IEEE Computer Society, April 2004.

[21] H. Gao, J.F. Groote, and W.H. Hesselink. Lock-free dynamic hash tables with open addressing. *Distributed Computing*, 2004. ISSN: 0178-2770 (Paper) 1432-0452 (Online) DOI: 10.1007/s00446-004-0115-2.

[22] H. Gao, J.F. Groote, and W.H. Hesselink. Lock-free parallel garbage collection by mark&sweep. Technical Report CS-Report CSR-04-31, Eindhoven University of Technology, The Netherlands, 2004.

[23] H. Gao and W.H. Hesselink. A formal reduction for lock-free parallel algorithms. In *Proceedings of the 16th Conference on Computer Aided Verification (CAV)*, July 2004.

[24] J.F. Groote, W.H. Hesselink, S. Mauw, and R. Vermeulen. An algorithm for the asynchronous write-all problem based on process collision. *Distributed Computing*, 14:75–81, 2001.

[25] S.P. Harbison. *Modula-3*. Prentice-Hall, Inc., 1992.

[26] K. Havelund. Mechanical verification of a garbage collector. In José Rolim et al., editors, *Parallel and Distributed Processing (Combined Proceedings of 11 Workshops)*, volume 1586 of *Lecture Notes in Computer Science*, pages 1258–1283. Springer-Verlag, April 1999. Presented at the Workshop on Formal Methods for Parallel Programming: Theory and Applications (FMPPTA).

[27] M. Herlihy. Wait-free synchronization. *ACM Transactions on Programming Languages and Systems*, 13(1):124–149, January 1991.

[28] M. Herlihy. A methodology for implementing highly concurrent data objects. *ACM Transactions on Programming Languages and Systems*, 15(5):745–770, November 1993.

[29] M.P. Herlihy, V. Luchangco, and M. Moir. The repeat offender problem: A mechanism for supporting dynamic-sized, lock-free data structure. In *Proceedings of 16th International Symposium on Distributed Computing*, pages 339–353. Springer-Verlag, October 2002.

[9] B.N. Bershad. Practical considerations for non-blocking concurrent objects. In *Proceedings of the Thirteenth International Conference on Distributed Computing Systems*, pages 264–274, 1993.

[10] H. Boehm, A. J. Demers, and S. Shenker. Mostly parallel garbage collection. In *Proceedings of the ACM SIGPLAN 1991 conference on Programming language design and implementation*, pages 157–164. ACM Press, 1991.

[11] F. Cassez, C. Jard, B. Rozoy, and M.D. Ryan. *Modeling and verification of parallel processes*. Springer-Verlag New York, Inc., 2001.

[12] K.M. Chandy and J. Misra. *Parallel program design: a foundation*. Addison-Wesley Longman Publishing Co., Inc., 1988.

[13] E. Clarke, O. Grumberg, and D. Long. Model checking and abstraction. *ACM Transactions on Programming Languages and Systems*, 16(5):1512–1542, 1994.

[14] C. Cornes, J. Courant, and et al. The coq proof assistant - reference manual v 6.1, 1997.

[15] D.L. Detlefs, P.A. Martin, M. Moir, and G.L. Steele Jr. Lock-free reference counting. *Distributed Computing*, 15(4):255–71, December 2002.

[16] E.W. Dijkstra, L. Lamport, A.J. Martin, C.S. Scholten, and E.F.M. Steffens. On-the-fly garbage collection: An exercise in cooperation. *Communications of the ACM*, 21(11):966–975, November 1978.

[17] D. Doligez and X. Leroy. A concurrent generational garbage collector for a multi-threaded implementation of ml. In *Proceedings of the 1993 ACM Symposium on Principles of Programming Languages*, pages 113–123, January 1993.

[18] T. Endo, K. Taura, and A. Yonezawa. A scalable mark-sweep garbage collector on large-scale shared-memory machines. In *Proceedings of the 1997 ACM/IEEE conference on Supercomputing (CDROM)*, pages 1–14. ACM Press, 1997.

[19] C. Flood, D. Detlefs, N. Shavit, and C. Zhang. Parallel garbage collection for shared memory multiprocessors. In *Usenix Java Virtual Machine Research and Technology Symposium (JVM '01)*, Monterey, CA, April 2001.

[30] M.P. Herlihy and J.E.B. Moss. Lock-free garbage collection for multiprocessors. *IEEE Transactions on Parallel and Distributed Systems*, 3(3):304–311, 1992.

[31] M.P. Herlihy and J.M. Wing. Linearizability: a correctness condition for concurrent objects. *ACM Trans. Program. Lang. Syst.*, 12(3):463–492, 1990.

[32] W.H. Hesselink. `http://www.cs.rug.nl/~wim/mechver/garbage_collection`

[33] W.H. Hesselink. `http://www.cs.rug.nl/~wim/mechver/hashtable`

[34] W.H. Hesselink. `http://www.cs.rug.nl/~wim/mechver/lockfree_reduction`

[35] W.H. Hesselink. Wait-free linearization with a mechanical proof. *Distributed Computing*, 9:21–36, 1995.

[36] W.H. Hesselink. Bounded delay for a free address. *Acta Informatica*, 33:233–254, 1996.

[37] W.H. Hesselink. Using eternity variables to specify and prove a serializable database interface. *Science of Computer Programming*, 51(1-2):47–85, 2004.

[38] W.H. Hesselink and J.F. Groote. Wait-free concurrent memory management by Create, and Read until Deletion. *Distributed Computing*, 14(1):31–39, January 2001.

[39] R. L. Hudson and J. E. B. Moss. Sapphire: copying gc without stopping the world. In *ISCOPE Conference on ACM 2001 Java Grande*, pages 48–57. ACM Press, 2001.

[40] L. Huelsbergen and J. R. Larus. A concurrent copying garbage collector for languages that distinguish (im)mutable data. In *Proceedings of the fourth ACM SIGPLAN symposium on Principles and practice of parallel programming*, pages 73–82. ACM Press, 1993.

[41] IBM. *IBM System/370 Extended Architecture, Principles of Operation*, 1983.

[42] E.H. Jensen, G.W. Hagensen, and J.M. Broughton. A new approach to exclusive data access in shared memory multiprocessors. Technical Report UCRL-97663, Lawrence Livemore National Laboratory, January 1987.

[43] R. Jones. *Garbage Collection: Algorithms for Automatic Dynamic Memory Management*. John Wiley and Sons, July 1996. With a chapter on Distributed Garbage Collection by Rafael Lins. Reprinted 1997 (twice), 1999, 2000.

[44] R. Jones and R. Lins. *Garbage collection: algorithms for automatic dynamic memory management*. John Wiley & Sons, Inc., 1996.

[45] J.E. Jonker. On-the-fly garbage collection for several mutators. *Distributed Computing*, 5:187–199, 1992.

[46] P.C. Kanellakis and A. A. Shvartsman. *Fault-Tolerant Parallel Computation*. Kluwer Academic Publishers, 1997.

[47] D.E. Knuth. *The art of computer programming, volume 3: (2nd ed.) sorting and searching*. Addison Wesley Longman Publishing Co., Inc., 1998.

[48] A. LaMarca. A performance evaluation of lock-free synchronization protocols. In *Proceedings of the thirteenth annual ACM symposium on Principles of distributed computing*, pages 130–140. ACM Press, 1994.

[49] L. Lamport. The temporal logic of actions. *ACM Transactions on Programming Languages and Systems*, 16(3):872–923, 1994.

[50] Y. Levanoni and E. Petrank. An on-the-fly reference counting garbage collector for java. In *Proceedings of the 16th ACM SIGPLAN conference on Object oriented programming, systems, languages, and applications*, pages 367–380. ACM Press, 2001.

[51] V. Luchangco, M. Moir, and N. Shavit. Nonblocking k-compare-single-swap. In *Proceedings of the fifteenth annual ACM symposium on Parallel algorithms and architectures*, pages 314–323. ACM Press, 2003.

[52] N.A. Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers, 1996.

[53] Z. Manna and A. Pnueli. *The temporal logic of reactive and concurrent systems: Specification*. Springer-Verlag New York, Inc., 1992.

[54] H. Massalin and C. Pu. A lock-free multiprocessor os kernel. Technical Report CUCS-005-91, Columbia University, 1991.

[55] M.M. Michael. High performance dynamic lock-free hash tables and list-based sets. In *Proceedings of the fourteenth annual ACM symposium on Parallel algorithms and architectures*, pages 73–82. ACM Press, 2002.

[56] M.M. Michael. Safe memory reclamation for dynamic lock-free objects using atomic reads and writes. In *Proceedings of the twenty-first annual symposium on Principles of distributed computing*, pages 21–30. ACM Press, 2002.

[57] M. Moir. Practical implementations of non-blocking synchronization primitives. In *Proceedings of the sixteenth annual ACM symposium on Principles of distributed computing*, pages 219–228. ACM Press, 1997.

[58] A.J. Mooij. Non-blocking implementations of LL, VL and SC. Private Communication, 2004.

[59] L. Moreau and J. Duprat. A construction of distributed reference counting. *Acta Inf.*, 37(8):563–595, 2001.

[60] Y. Ossia, O. Ben-Yitzhak, I. Goft, E.K. Kolodner, V. Leikehman, and A. Owshanko. A parallel, incremental and concurrent gc for servers. *SIGPLAN Not.*, 37(5):129–140, 2002.

[61] J. O'Toole and S. Nettles. Concurrent replicating garbage collection. In *Proceedings of the 1994 ACM conference on LISP and functional programming*, pages 34–42. ACM Press, 1994.

[62] S. Owicki and L. Lamport. Proving liveness properties of concurrent programs. *ACM Trans. Program. Lang. Syst.*, 4(3):455–495, 1982.

[63] S. Owre, N. Shankar, J.M. Rushby, and D.W.J. Stringer-Calvert. *PVS Version 2.4: System Guide, Prover Guide, PVS Language Reference*, 2001.

[64] R. Rajwar and J.R. Goodman. Transactional lock-free execution of lock-based programs. In *Proceedings of the 10th international conference on Architectural support for programming languages and operating systems*, pages 5–17. ACM Press, 2002.

[65] H. Rodrigues and R. Jones. Cyclic distributed garbage collection with group merger. In *Proceedings of 12th European Conference on Object-Oriented Programming, ECOOP98*, pages 249–273, Brussels, July 1998. Springer.

[66] O. Shalev and N. Shavit. Split-ordered lists: lock-free extensible hash tables. In *Proceedings of the twenty-second annual symposium on Principles of distributed computing*, pages 102–111. ACM Press, 2003.

[67] H. Sundell and P. Tsigas. Scalable and lock-free concurrent dictionaries. In *Proceedings of the 2004 ACM Symposium on Applied computing*, pages 1438–1445, 2004.

[68] J.D. Valois. Implementing lock-free queues. In *Proceedings of the Seventh International Conference on Parallel and Distributed Computing Systems*, pages 64–69, Las Vegas, NV, 1994.

[69] J.D. Valois. Lock-free linked lists using compare-and-swap. In *Proceedings of the fourteenth annual ACM symposium on Principles of distributed computing*, pages 214–222. ACM Press, 1995.

[70] N. Wirth. *Algorithms + Data Structures = Programs.* Prentice Hall PTR, 1978.