Model checking

Abstract: This article contains bibiliography on Model checking, Software aspects of model checking, concrete numerative model checking and properties of model checking.

Bibiliography:

Abadi, M. and Lamport, L. 1993. Composing specifications. *ACM Transactions on Programming Languages and Systems 15,* 1, 73-132.

Abadi, M. and Lamport, L. 1995. Conjoining specifications. *ACM Transactions on Programming Languages and Systems 17,* 3, 507-534.

Agerwala, T. and Misra, J. 1978. Assertion graphs for verifying and synthesizing programs. Tech. Rep. 83, University of Texas, Austin.

Alpern, B. and Schneider, F. 1987. Recognizing safety and liveness. *Distributed Computing 3,* 3, 117-126.

Alur, R., Dang, T., and Ivanˇ´ F. 2006. Counterexample-guided predicate abstraction of cic, hybrid systems. *Theoretical Computer Science 354,* 2, 250-271.

Alur, R. and Henzinger, T. 1999. Reactive modules. *Formal Methods in System Design 15,* 1, 7-48.

Armando, A., Mantovani, J., and Platania, L. 2006. Bounded model checking of software using SMT solvers instead of SAT solvers. In *Model Checking Software: SPIN Workshop*. Lecture Notes in Computer Science 3925. Springer-Verlag, 146-162.

Babic, D. and Hu, A. 2008. Calyso: scalable and precise extended static checking. In *ICSE 08: International Conference on Software Engineering*. ACM, 211-220.

Bagnara, R., Hill, P. M., Mazzi, E., and Zaffanella, E. 2005. Widening operators for weakly-relational numeric abstractions. In *SAS 05: Static Analysis Symposium*. Lecture Notes in Computer Science 3672. Springer-Verlag, 3-18.

Bagnara, R., Hill, P. M., and Zaffanella, E. 2008. The Parma Polyhedra Library: Toward a complete set of numerical abstractions for the analysis and verification of hardware and software systems. *Science of Computer Programming 72,* 1-2, 3-21.

Ball, T., Bounimova, E., Cook, B., Levin, V., Lichtenberg, J., McGarvey, C., Ondrusek, B., Rajamani, S. K., and Ustuner, A. 2006. Thorough static analysis of device drivers. In *EuroSys*. 73-85.

Ball, T., Majumdar, R., Millstein, T., and Rajamani, S. K. 2001. Automatic predicate abstraction of C programs. In *PLDI 01: Programming Languages Design and Implementation*. ACM, 203-213.

Ball, T., Millstein, T. D., and Rajamani, S. K. 2005. Polymorphic predicate abstraction. *ACM Transactions on Programming Languages and Systems 27,* 2, 314-343.

Ball, T., Podelski, A., and Rajamani, S. K. 2001. Boolean and Cartesian abstractions for model checking C programs. In *TACAS 01: Tools and Algorithms for Construction and Analysis of Systems*. Lecture Notes in Computer Science 2031. Springer-Verlag, 268-283.

Ball, T., Podelski, A., and Rajamani, S. K. 2002. Relative completeness of abstraction refinement for software model checking. In *TACAS 02: Tools and Algorithms for Construction and Analysis of Systems*. Lecture Notes in Computer Science 2280. Springer-Verlag, 158-172.

Ball, T. and Rajamani, S. 2002a. Generating abstract explanations of spurious counterexamples in C programs. Tech. Rep. MSR-TR-2002-09, Microsoft Research.

Ball, T. and Rajamani, S. 2002b. The SLAM project: debugging system software via static

analysis. In *POPL 02: Principles of Programming Languages*. ACM, 1-3.

Ball, T. and Rajamani, S. K. 2000a. Bebop: A symbolic model checker for Boolean programs. In *SPIN 00: SPIN Workshop*. Lecture Notes in Computer Science 1885. Springer-Verlag, 113-130.

Ball, T. and Rajamani, S. K. 2000b. Boolean programs: a model and process for software analysis. Tech. Rep. MSR Technical Report 2000-14, Microsoft Research.

Beckman, N., Nori, A. V., Rajamani, S. K., and Simmons, R. J. 2008. Proofs from tests. In *ISSTA 08: International Symposium on Software Testing and Analysis*. ACM, 3-14.

Beyer, D., Chlipala, A. J., Henzinger, T. A., Jhala, R., and Majumdar, R. 2004. Generating tests from counterexamples. In *ICSE 04: International Conference on Software Engineering*. ACM, 326-335.

Beyer, D., Henzinger, T., Majumdar, R., and Rybalchenko, A. 2007a. Invariant synthesis in combination theories. In *VMCAI 07: Verification, Model Checking, and Abstract Interpretation*. Lecture Notes in Computer Science 4349. Springer-Verlag, 378-394

Beyer, D., Henzinger, T., Majumdar, R., and Rybalchenko, A. 2007b. Path invariants. In *PLDI 07: Programming Language Design and Implementation*. ACM, 300-309.

Beyer, D., Henzinger, T. A., Jhala, R., and Majumdar, R. 2007. The software model checker Blast. *Software Tools for Technology Transfer 9,* 5-6, 505-525.

Beyer, D., Henzinger, T. A., and Th´ duloz, G. 2007. Configurable software verification: eo Concretizing the convergence of model checking and program analysis. In *CAV 07: Computer-Aided Verification*. Lecture Notes in Computer Science 4590. Springer-Verlag, 504-518.

Biere, A., Cimatti, A., Clarke, E. M., Fujita, M., and Zhu, Y. 1999. Symbolic model checking using SAT procedures instead of BDDs. In *DAC 99: Design Automation Conference*. ACM, 317-320.

Bradley, A., Manna, Z., and Sipma, H. 2005. The polyranking principle. In *ICALP 05: International Colloquium on Automata, Languages, and Programming*. Lecture Notes in Computer Science 3580. Springer-Verlag, 1349-1361.

Brat, G., Drusinsky, D., Giannakopolou, D., Goldberg, A., Havelund, K., Lowry, M., Pasareanu, C., Venet, A., Washington, R., and Visser, W. 2004. Experimental evaluation of verification and validation tools on Martian rover software. *Formal Methods in Systems Design 25*.

Bruttomesso, R., Cimatti, A., Franzen, A., Griggio, A., and Sebastiani, R. 2008. The ´ MathSAT 4 SMT solver. In *CAV 08: Computer-Aided Verification*. Lecture Notes in Computer Science 5123. Springer-Verlag, 299-303.

Bryant, R. 1986. Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers C-35,* 8, 677-691.

Burch, J., Clarke, E., McMillan, K., Dill, D., and Hwang, L. 1992. Symbolic model checking: 10^{20} states and beyond. *Information and Computation 98,* 2, 142-170.

Bustan, D. and Grumberg, O. 2003. Simulation-based minimization. *ACM Transactions on Computational Logic 4*, 181-206.

Cadar, C., Ganesh, V., Pawlowski, P., Dill, D., and Engler, D. 2006. EXE: automatically generating inputs of death. In *CCS 02: Conference on Computer and Communications Security*. ACM.

Chaki, S., Clarke, E., Groce, A., Ouaknine, J., Strichman, O., and Yorav, K. 2004. Efcient verification of sequential and concurrent C programs. *Formal Methods in System Design 25,* 2-3, 129-166.

Chaki, S., Clarke, E., Kidd, N., Reps, T., and Touili, T. 2006. Verifying concurrent message-passing C programs with recursive calls. In *TACAS 06: Tools and Algorithms for the Construction and Analysis of Systems*. Lecture Notes in Computer Science 3920. Springer-Verlag, 334-349.

Chaki, S., Clarke, E. M., Groce, A., and Strichman, O. 2003. Predicate abstraction with minimum predicates. In *CHARME*. 19-34.

Chen, H. and Wagner, D. 2002. MOPS: an infrastructure for examining security properties of software. In *ACM Conference on Computer and Communications Security 2002*. 235-244.

Clarke, E. M. and Emerson, E. A. 1981. Synthesis of synchronization skeletons for branching time temporal logic. In *Logic of Programs*. Lecture Notes in Computer Science 131. Springer-Verlag, 52-71.

Clarke, E. M., Grumberg, O., Jha, S., Lu, Y., and Veith, H. 2000. Counterexample-guided abstraction refinement. In *CAV 00: Computer-Aided Verification*. Lecture Notes in Computer Science 1855. Springer-Verlag, 154-169.

Clarke, E. M., Grumberg, O., and Long, D. 1992. Model checking and abstraction. In *POPL 92: Principles of Programming Languages*. ACM, 343-354.

Clarke, L. 1976. A system to generate test data and symbolically execute programs. *IEEE Transactions in Software Engineering 2(2)*, 215-222.

Colon, M. and Sipma, H. 2001. Synthesis of linear ranking functions. In *TACAS 01: Tools and* ´

Algorithms for the Construction and Analysis of Systems. Lecture Notes in Computer Science 2031. Springer-Verlag, 67-81

Col´ M. and Sipma, H. 2002. Practical methods for proving program termination. In *CAV 02:* on, *Computer-Aided Verification*. Lecture Notes in Computer Science, vol. 2404. Springer-Verlag, 442-454.

Constable, R. 1986. *Implementing Mathematics with the Nuprl Proof Development System*. Prentice-Hall.

Cook, B., Podelski, A., and Rybalchenko, A. 2006. Termination proofs for systems code. In *PLDI 06: Programming Languages Design and Implementation*. ACM, 415-426.

Cook, S. A. 1978. Soundness and completeness of an axiom system for program verification. *SIAM Journal of Computing 7,* 1, 70-90.

Corbett, J., Dwyer, M., Hatcliff, J., Pasareanu, C., Robby, Laubach, S., and Zheng, H. 2000. Bandera: Extracting finite-state models from Java source code. In *ICSE 00: Software Engineering*. 439-448.

Courcoubetis, C., Vardi, M., Wolper, P., and Yannakakis, M. 1992. Memory-efcient algorithms for the verification of temporal properties. *Formal Methods in System Design 1*, 275-288.

Cousot, P. and Cousot, R. 1976. Static determination of dynamic properties of programs. In *ISOP*. 106-130.

Cousot, P. and Cousot, R. 1977. Abstract interpretation: a unified lattice model for the static analysis of programs. In *POPL 77: Principles of Programming Languages*. ACM, 238-252.

Cousot, P. and Cousot, R. 1979. Systematic design of program analysis frameworks. In *POPL 79: Principles of Programming Languages*. 269-282.

Cousot, P. and Cousot, R. 2000. Temporal abstract interpretation. In *POPL 00: Principles of Programming Languages*. ACM, 12-25.

Cousot, P. and Halbwachs, N. 1978. Automatic discovery of linear restraints among variables of a program. In *POPL 78: Principles of Programming Languages*. ACM.

Dijkstra, E. 1976. *A Discipline of Programming*. Prentice-Hall.

Dill, D. 1996. The Murphi verification system. In *CAV 96: Computer-Aided Verification*. Lecture Notes in Computer Science 1102. Springer-Verlag, 390-393.

Dimitrova, R. and Podelski, A. 2008. Is lazy abstraction a decision procedure for broadcast protocols? In *VMCAI 08: Verification, Model Checking, and Abstract Interpretation*. Lecture Notes in Computer Science 4905. Springer-Verlag, 98-111.

Flanagan, C., Freund, S., and Qadeer, S. 2002. Thread-modular verification for shared-memory programs. In *ESOP 02: European Symposium on Programming*. Lecture Notes in Computer Science 2305. Springer-Verlag, 262-277.

Flanagan, C., Freund, S., Qadeer, S., and Seshia, S. 2005. Modular verification of multithreaded programs. *Theoretical Computer Science 338*, 153-183.

Flanagan, C., Joshi, R., and Leino, K. R. M. 2001. Annotation inference for modular checkers. *Information Processing Letters 77,* 2-4, 97-108.

Flanagan, C., Leino, K., Lillibridge, M., Nelson, G., Saxe, J. B., and Stata, R. 2002. Extended static checking for Java. In *PLDI 02: Programming Language Design and Implementation*. ACM, 234-245.

Flanagan, C. and Qadeer, S. 2002. Predicate abstraction for software verification. In *POPL 02: Principles of Programming Languages*. ACM, 191-202.

Flanagan, C. and Saxe, J. 2000. Avoiding exponential explosion: generating compact verification conditions. In *POPL 00: Principles of Programming Languages*. ACM, 193-205.

Floyd, R. 1967. Assigning meanings to programs. In *Mathematical Aspects of Computer Science*. American Mathematical Society, 19-32.

Foster, J., Terauchi, T., and Aiken, A. 2002. Flow-sensitive type qualifiers. In *PLDI 02: Programming Language Design and Implementation*. ACM, 1-12.

Francez, N. 1986. *Fairness*. Springer-Verlag.

Gopan, D., Reps, T. W., and Sagiv, S. 2005. A framework for numeric analysis of array operations. In *POPL 05: Principles of Programming Languages*. ACM, 338-350.

Graf, S. and Sa¨ i, H. 1997. Construction of abstract state graphs with PVS. In *CAV*. Lecture ıd Notes in Computer Science 1254. Springer-Verlag, 72-83.

Gulavani, B. S., Chakraborty, S., Nori, A. V., and Rajamani, S. K. 2008. Automatically refining abstract interpretations. In *TACAS 08: Tools and Algorithms for the Construction and Analysis of Systems*. Lecture Notes in Computer Science 4963. Springer-Verlag, 443-458.

Gulavani, B. S., Henzinger, T. A., Kannan, Y., Nori, A. V., and Rajamani, S. K. 2006. Synergy: a new algorithm for property checking. In *FSE 06: Foundations of Software Engineering*. ACM, 117-127.

Gulwani, S., McCloskey, B., and Tiwari, A. 2008. Lifting abstract interpreters to quantified logical domains. In *POPL 08: Principles of Programming Languages*. 235-246.

Gulwani, S. and Tiwari, A. 2006. Combining abstract interpreters. In *PLDI 2006: Programming Language Design and Implementation*. ACM, 376-386.

Gupta, A., Henzinger, T., Majumdar, R., Rybalchenko, A., and Xu, R. 2008. Proving non-termination. In *POPL 08: Principles of Programming Languages*. ACM, 147-158.

Gupta, A., Majumdar, R., and Rybalchenko, A. 2009. From tests to proofs. In *TACAS 09: Tools and Algorithms for the Construction and Analysis of Systems*. Lecture Notes in Computer Science 5505. Springer-Verlag, 262-276.

Hackett, B. and Aiken, A. 2006. How is aliasing used in systems software? In *FSE 06: Foundations of Software Engineering*. ACM, 69-80.

Hardekopf, B. and Lin, C. 2007. The ant and the grasshopper: fast and accurate pointer analysis for millions of lines of code. In *PLDI 07: Programming Language Design and Implementation*. ACM, 290-299.

Hart, P., Nilsson, N., and Raphael, B. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics SSC4 2*, 100-107.

vancic, F., Yang, Z., Ganai, M. K., Gupta, A., Shlyakhter, I., and Ashar, P. 2005. F-soft: Software verification platform. In *CAV 05: Computer-Aided Verification*. 301-306.

Jain, H., Ivanˇ´ F., Gupta, A., Shlyakhter, I., and Wang, C. 2006. Using statically computed cic, invariants inside the predicate abstraction and refinement loop. In *CAV 06: Computer-Aided Verification*. 137-151.

Jha, S. K., Krogh, B. H., Weimer, J. E., and Clarke, E. M. 2007. Reachability for linear hybrid automata using iterative relaxation abstraction. In *HSCC*. 287-300.

Jhala, R. and Majumdar, R. 2005. Path slicing. In *PLDI 05: Programming Language Design and Implementation*. ACM, 38-47.

Jhala, R. and McMillan, K. 2006. A practical and complete approach to predicate refinement. In *TACAS 06: Tools and Algorithms for the Construction and Analysis of Systems*. Lecture Notes in Computer Science 2987. Springer-Verlag, 298-312.

Jhala, R. and McMillan, K. L. 2005. Interpolant-based transition relation approximation. In *CAV 05*. Lecture Notes in Computer Science. Springer-Verlag, 39-51.

Jones, C. 1983. Tentative steps toward a development method for interfering programs. *ACM Transactions on Programming Languages and Systems 5(4)*, 596-619.

Kahlon, V. and Gupta, A. 2007. On the analysis of interacting pushdown systems. In *POPL 07: Principles of Programming Languages*. 303-314.

Katz, S. and Peled, D. 1992. Verification of distributed programs using representative interleaving sequences. *Distributed Computing 6,* 2, 107-120.

Killian, C. E., Anderson, J. W., Jhala, R., and Vahdat, A. 2007. Life, death, and the critical transition: Finding liveness bugs in systems code (awarded best paper). In *NSDI*.

King, J. 1976. Symbolic execution and program testing. *Communications of the ACM 19(7)*, 385-394.

Korf, R. 1985. Depth-first iterative deepening: an optimal admissible tree search. *Artificial Intelligence 27*, 97-109.

Kroening, D., Clarke, E., and Yorav, K. 2003. Behavioral consistency of C and Verilog programs using bounded model checking. In *DAC 03: Design Automation Conference*. ACM, 368-371.