Model Checking *

Edmund M. Clarke

Department of Computer Science
Carnegie Mellon, Pittsburgh

ABSTRACT: Model checking is an automatic technique for verifying finite-state reactive systems, such as sequential circuit designs and communication protocols. Specifications are expressed in temporal logic, and the reactive system is modeled as a state-transition graph. An efficient search procedure is used to determine whether or not the state-transition graph satisfies the specifications.

We describe the basic model checking algorithm and show how it can be used with binary decision diagrams to verify properties of large state-transition graphs. We illustrate the power of model checking to find subtle errors by verifying part of the *Contingency Guidance Requirements* for the Space Shuttle.

Keywords: automatic verification, temporal logic, model checking, binary decision diagrams

Model checking is an automatic technique for verifying finite-state reactive systems. Specifications are expressed in a propositional temporal logic, and the reactive system is modeled as a state-transition graph. An efficient search procedure is used to determine automatically if the specifications are satisfied by the state-transition graph. The technique was originally developed in 1981 by Clarke and Emerson [10, 11]. Quielle and Sifakis [18] independently discovered a similar verification technique shortly thereafter. An alternative approach based on showing inclusion between ω-automata was later devised by Robert Kurshan at ATT Bell Laboratories [14, 15].

Model checking has a number of advantages over verification techniques based on automated theorem proving. The most important is that the procedure is highly automatic. Typically, the user provides a high level representation of the model and the specification to be checked. The model checker will either terminate with the answer *true*, indicating that the model satisfies the specification, or give a counterexample execution that shows why the formula is not satisfied. The counterexamples are particularly important in finding subtle errors in complex reactive systems.

The first model checkers were able to verify small examples ([1], [2], [3], [4], [11], [13], [16]). However, they were unable to handle very large examples due to the *state explosion problem*. Because of this limitation, many researchers in formal verification predicted that model checking would never be useful in practice.

The possibility of verifying systems with realistic complexity changed dramatically in the late 1980's with the discovery of how to represent transition relations using *ordered binary decision diagrams (OBDDs)* [5]. This discovery was made independently by three

* This research is sponsored in part by the Wright Laboratory, Aeronautical Systems Center, Air Force Materiel Command, USAF, and the Advanced Research Projects Agency (ARPA) under grant F33615-93-1-1330. and in part by the National Science foundation under Grant No. CCR-9217549 and in part by the Semiconductor Research Corporation under Contract 92-DJ-294. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied of the U.S. government.

research teams [8, 12, 17] and is basically quite simple. Assume that the behavior of a reactive system is determined by n boolean state variables $v_1, v_2, \ldots, v_n$. Then the transition relation of the system can be expressed as a boolean formula

$$R(v_1, v_2, \ldots, v_n, v_1', v_2', \ldots, v_n')$$

where $v_1, v_2, \ldots, v_n$ represents the current state and $v_1', v_2', \ldots, v_n'$ represents the next state. By converting this formula to a BDD, a very concise representation of the transition relation may be obtained.

The original model checking algorithm, together with the new representation for transition relations, is called *symbolic model checking* [7, 8, 9]. By using this combination, it is possible to verify extremely large reactive systems. In fact, some examples with more than than 10^{120} states have been verified [6, 9]. This is possible because the number of nodes in the OBDDs that must be constructed no longer depends on the actual number of states or the size of the transition relation. Because of this breakthrough it is now possible to verify reactive systems with realistic complexity, and a number of major companies including Intel, Motorola, Fujitsu, and ATT have started using symbolic model checkers to verify actual circuits and protocols. In several cases, errors have been found that were missed by extensive simulation.

We illustrate the power of model checking to find subtle errors by considering a protocol used by the Space Shuttle. We discuss the verification of the *Three-Engines-Out Contingency Guidance Requirements* using the SMV model checker. The example describes what should be done in a situation where all of the three main engines of the Space Shuttle fail during the ascent. The main task of the *Space Shuttle Digital Autopilot* is to separate the shuttle from the external tank and dump extra fuel if necessary. The task involves a large number of cases and has many different input parameters. Thus, it is important to make sure that all possible cases and input values are taken into account and that the tank will eventually separate.

The Digital Autopilot chooses one of the six contingency regions depending on the current flight conditions. Each region uses different maneuvers for separating from the external tank. This involves computing a guidance *quaternion*. Usually, the region is chosen once at the beginning of the contingency and is maintained until separation occurs. However, under certain conditions a change of region is allowed. In this case, it is necessary to recompute the quaternion and certain other output values. Using SMV we were able to find a counterexample in the program for this task. We discovered that when a transition between regions occurs, the autopilot system may fail to recompute the quaternion and cause the wrong maneuver to be made. The guidance program consists of about 1200 lines of SMV code. The number of reachable states is $2 \cdot 10^{14}$, and it takes 60 seconds to verify 40 CTL formulas.

References

1. M. C. Browne and E. M. Clarke. Sml: A high level language for the design and verification of finite state machines. In *IFIP WG 10.2 International Working Conference from HDL Descriptions to Guaranteed Correct Circuit Designs, Grenoble, France.* IFIP, September 1986.
2. M. C. Browne, E. M. Clarke, and D. Dill. Checking the correctness of sequential circuits. In *Proceedings of the 1985 International Conference on Computer Design*, Port Chester, New York, October 1985. IEEE.

3. M. C. Browne, E. M. Clarke, and D. Dill. Automatic circuit verification using temporal logic: Two new examples. In *Formal Aspects of VLSI Design*. Elsevier Science Publishers (North Holland), 1986.
4. M. C. Browne, E. M. Clarke, D. L. Dill, and B. Mishra. Automatic verification of sequential circuits using temporal logic. *IEEE Transactions on Computers*, C-35(12):1035–1044, 1986.
5. R. E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, C-35(8), 1986.
6. J. R. Burch, E. M. Clarke, and D. E. Long. Symbolic model checking with partitioned transition relations. In A. Halaas and P. B. Denyer, editors, *Proceedings of the 1991 International Conference on Very Large Scale Integration*, August 1991. Winner of the Sidney Michaelson Best Paper Award.
7. J. R. Burch, E. M. Clarke, K. L. McMillan, and D. L. Dill. Sequential circuit verification using symbolic model checking. In *Proceedings of the 27th ACM/IEEE Design Automation Conference*. IEEE Computer Society Press, June 1990.
8. J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and J. Hwang. Symbolic model checking: 10^{20} states and beyond. In *Proceedings of the Fifth Annual Symposium on Logic in Computer Science*. IEEE Computer Society Press, June 1990.
9. Jerry R. Burch, Edmund M. Clarke, David E. Long, Kenneth L. McMillan, and David L. Dill. Symbolic model checking for sequential circuit verification. *IEEE Transactions on Computer-Aided Design of Integrated Circuits*, 13(4):401–424, April 1994.
10. E. M. Clarke and E. A. Emerson. Synthesis of synchronization skeletons for branching time temporal logic. In *Logic of Programs: Workshop, Yorktown Heights, NY, May 1981*, volume 131 of *Lecture Notes in Computer Science*. Springer-Verlag, 1981.
11. E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, 1986.
12. O. Coudert, C. Berthet, and J. C. Madre. Verification of synchronous sequential machines based on symbolic execution. In J. Sifakis, editor, *Proceedings of the 1989 International Workshop on Automatic Verification Methods for Finite State Systems, Grenoble, France*, volume 407 of *Lecture Notes in Computer Science*. Springer-Verlag, June 1989.
13. D. L. Dill and E. M. Clarke. Automatic verification of asynchronous circuits using temporal logic. *IEE Proceedings*, Part E 133(5), 1986.
14. Z. Har'El and R. P. Kurshan. Software for analytical development of communications protocols. *AT&T Technical Journal*, 69(1):45–59, Jan.–Feb. 1990.
15. R. P. Kurshan. Analysis of discrete event coordination. In J. W. de Bakker, W.-P. de Roever, and G. Rozenberg, editors, *Proceedings of the REX Workshop on Stepwise Refinement of Distributed Systems, Models, Formalisms, Correctness*, volume 430 of *Lecture Notes in Computer Science*. Springer-Verlag, May 1989.
16. B. Mishra and E.M. Clarke. Hierarchical verification of asynchronous circuits using temporal logic. *Theoretical Computer Science*, 38:269–291, 1985.
17. C. Pixley. A computational theory and implementation of sequential hardware equivalence. In R. Kurshan and E. Clarke, editors, *Proc. CAV Workshop (also DIMACS Tech. Report 90-31)*, Rutgers University, NJ, June 1990.
18. J.P. Quielle and J. Sifakis. Specification and verification of concurrent systems in CESAR. In *Proceedings of the Fifth International Symposium in Programming*, 1981.