[WC98b] Enoch Y. Wang and Betty H. C. Cheng. A rigorous object-oriented design process. In *Proc. of International Conference on Software Process*, Naperville, Illinos, June 1998.

[WRC97] Enoch Y. Wang, Heather A. Richter, and Betty H. C. Cheng. Formalizing and integrating the dynamic model within OMT. In *Proc. of IEEE International Conference on Software Engineering (ICSE97)*, Boston, MA, May 1997.

Automatic Verification of Hardware and Software Systems

Edmund M. Clarke
Carnegie Mellon University
Pittsburgh, Pa 15213
(emc@cs.cmu.edu)

Logical errors in finite-state concurrent systems such as sequential circuit designs and communication protocols are an important problem for computer scientists. They can delay getting a new product on the market or cause the failure of some critical device that is already in use. My research group has developed a verification method called *temporal logic model checking* for this class of systems. In this approach specifications are expressed in a propositional temporal logic, while circuits and protocols are modeled as state–transition systems. An efficient search procedure is used to determine automatically if a specification is satisfied by some transition system. The technique has been used in the past to find subtle errors in a number of non-trivial examples.

During the past decade, the size of the state-transition systems that can be verified by model checking techniques has increased dramatically. By representing transition relations implicitly using Binary Decision Diagrams (BDDs), we have been able to check some examples that would have required 10^{20} states with the original algorithm. Various refinements of the BDD-based techniques have pushed the state count up to 10^{100}. By combining model checking with various abstraction techniques, we have been able to handle even larger systems. For example, we have used this technique to verify the cache coherence protocol in the IEEE Futurebus+ Standard. We found several errors that had been previously undetected. Apparently, this was the first time that formal methods have been used to find nontrivial errors in an IEEE standard.

Although we believe that model checking is already useful for verifying many circuit and protocol designs that arise in industry, additional research is needed to exploit the full power of this method. We describe below some of the state-space reduction techniques that we are currently investigating.

Exploiting symmetry to reduce model size Our work in this area has focused on developing techniques for exploiting symmetry to alleviate the state explosion problem. Finite state concurrent systems frequently exhibit considerable symmetry. It is possible to find symmetry in memories, caches, register files, bus protocols, network protocols – anything that has a lot of replicated structure. For example, a ring of processes exhibits rotational symmetry. This fact can be used to obtain an equivalent reduced model of the system. Given a Kripke Structure, a symmetry group is a group acting on the state set that preserves the transition relation. A symmetry group acting on the state set partitions the state set into equivalence classes called orbits. A quotient model is constructed that contains one representative from each orbit. The state space of the quotient model will, in general, be much smaller than the original state space. This makes it possible to verify much larger structures. Based on these ideas we have built a prototype tool called SYMM. The user describes a system, gives a specification in CTL, and provides the symmetries of the system. SYMM checks that the symmetries provided are valid and then uses it during the model checking process. We have verified several examples using this tool. Two specific examples are the cache-coherence protocol and the arbiter circuit given in the IEEE Futurebus+ standard. For both examples, substantial reductions were obtained because of the use of symmetry. Similar results have been obtained by Emerson and Sistlay.

Verifying parameterized designs Most of the research done in the area of model checking focuses on verifying single finite-state systems. Typically, circuit and protocol designs are parameterized, i.e., define an infinite family of systems. For example, a circuit design to multiply two integers has the width of the integers as a parameter. We have investigated methods to verify such parameterized designs. This problem is difficult because the state space is unbounded. Formally, the problem of verifying parameterized designs can be stated as: Given an infinite family F offinite-state systems and a temporal specification f, determine whether Pi models f for all Pi in F.

In general the problem is undecidable. However, for specific families the problem can be solved. Most techniques are based on finding network invariants. Given an infinite family F and a reflexive, transitive relation ¡=, an invariant I is a process such that Pi ¡= I for all Pi in F. The relation ¡= should preserve the property f we are interested in, i.e., if I satisfies f, then Pi should also satisfy f. Once the invariant I is found, traditional model checking techniques can be used to check that I satisfies f. We have defined a formalism based on network grammars to describe parameterized designs and created a logic based on regular expressions which can be interpreted over these parameterized designs. Our methodology constructs invariants based on a technique called unfolding. We have developed a prototype tool INDUCT to generate invariants. We have been able to generate invariants automatically for a complicated token-ring example and a systolic parity tree circuit.

Partial order reduction for timed systems: We have also investigated how to approach the state explosion problem for timed systems, such as asynchronous circuits and protocols, or mixed hardware-software designs. Recently, we have demonstrated how this problem can be alleviated by using partial order reduction, which restricts exploration to a subset of representative trajectories in the state space, while preserving the verified property. The analyzed models are compositions of timed automata, a formalism that uses continuous-time clocks to represent the passage of time. Specifications are given in an extension of next-time free linear temporal logic, in which timing relationships of events are modeled by specifying bounds on the difference between two clocks of an automaton. We use a relaxed local time semantics for timed automata that can be extended to preserve properties in the selected logic. The resulting algorithm achieves reduction in both the control state space and the time space of the model.

Symbolic model checking without BDDs: Symbolic Model Checking has proven to be a powerful technique for the verification of reactive systems. BDDs have traditionally been used as a symbolic representation of the system. Recently, we have demonstrated how boolean decision procedures, like Stälmarck's Method or the Davis & Putnam Procedure, can replace BDDs. This new technique avoids the space blow up of BDDs, generates counterexamples much faster, and sometimes speeds up the verification. In addition, it produces counterexamples of minimal length. We have implemented a *bounded model checking* procedure for LTL which reduces model checking to propositional satisfiability. Bounded LTL model checking does not require an expensive tableau construction. Our initial experiments with the new model checking system are quite promising.

Specification Formalisms for Component-Based Concurrent Systems[7]

Rance Cleaveland
SUNY at Stony Brook
Stony Brook, NY 11794-4400
(rance@cs.sunysb.edu)
http://www.cs.sunysb.edu/~rance

Introduction and Project Goals.

This project builds on my ongoing research into design formalisms for, and the automatic verification of, concurrent systems. The difficulties such systems pose for system engineers are well-known and result in large part from the the complexities of process interaction and the possibilities for nondeterminism. My work is motivated by a belief that mathematically rigorous specification and verification techniques will ultimately lead to better and easier-to-build concurrent systems.

My specific research interests lie in the development of fully automatic analysis methods and process-algebraic design formalisms for modeling system behavior. I have worked on algorithms for checking properties of, and refinement relations between, system descriptions [CH93, CS93]; the implementation and release of a verification tool, the CWB- NC [CS96] (see http://www.cs.sunysb.edu/~rance to obtain the distribution); case studies [BCL99, ECB97]; and the formalization of system features, such as real time, probability, and priority, in process algebra [BCL99, CDSYar].

[7]NSF CCR-9996086.

The aims of this project include the development of expressive and usable formalisms for specifying and reasoning about properties of *open*, *component-based* concurrent systems. More specifically, my colleagues and I have been investigating new approaches for describing component requirements and automated techniques for determining when finite-state components meet their requirements. The key topics under study include the following.

A temporal logic for open systems. We are working on a notation for conveniently expressing properties constraining the behavior of open systems.

Implicit specifications. *Implicit specifications* use of system *contexts*, or "test harnesses," to define requirements for open systems. We are studying expressiveness issues and model-checking algorithms for such specifications.

Automatic model-checker generation. We have been developing a *model-checker generator* that, given a temporal logic and "proof rules" for the logic, automatically produces an efficient model checker.

Status.

Work so far has focused on the temporal logic for open systems and the model-checker generator. The former extends an existing temporal logic with "action formulas" that allow the characterization of an open system's response to environmental stimuli. We have also identified a useful intermediate notation to "compile" temporal formulas to and have designed and implemented a model checker for the notation in the Concurrency Workbench. Both pieces of work are the subject of a paper in preparation [BCG].

With researchers in Germany we have also studied a class of implicit specifications and showed that they may automatically be converted in explicit specifications [MOSC99]. The former are often easier to write, while the second provided clearer guidance to implementors.

We have also been studying temporal logics for probabilistic behavior of open systems [NCI99] and compositional design frameworks based on Statecharts [LvdBC99].

Future Plans

With the model-checking kernel in hand, we plan next to implement a package taking a logic and proof rules as input and generating routines for converting formulas into our intermediate format. Like the Process Algebra Compiler [CSar], this tool will be used to provide specialized front ends for the CWB-NC. In contrast with the PAC, however, the new tool will generate new analyzers for system properties rather than new design-language routines. We also plan to continue our investigations into implicit specifications and into the incorporation of probabilistic and performance information into models of open systems.

References

[BCG] G. Bhat, R. Cleaveland, and A. Groce. Efficient on-the-fly model checking via alternating Buechi tableau automata. In preparation.