Temporal Logic Model Checking: Two Techniques for Avoiding the State Explosion Problem

Edmund M. Clarke, Jr.
School of Computer Science
Carnegie Mellon

ABSTRACT

Verifying the correctness of finite-state concurrent programs has been an important problem for a long time. However, lack of any formal and efficient method of verification has prevented the creation of practical design aids for this purpose. Since all known techniques of simulation and prototype testing are time-consuming and not very reliable, there is an acute need for such tools. We describe an automatic verification system for such systems in which specifications are expressed in a propositional temporal logic. We use a procedure called temporal logic model checking to determine automatically if a specification is true of a large state-transition graph. In contrast to most other mechanical verification systems, our system does not require user assistance and will produce a counterexample trace to show how an error occurs if this is possible.

The main difficulty with our approach is that the state space we have to search may grow exponentially with the number of circuit components. Recently, however, we have made significant progress on this problem. In this talk we outline how our verification method works and describe two of the techniques that we have developed for dealing with the state explosion problem.

The first technique is an automatic procedure for reducing the state-space of a modular system by abstracting away details of a module's behaviour that are not relevant to the specification being checked. This method relies on a description of the system as a hierarchy of concurrently executing modules. To provide a vehicle for such descriptions, we have extended the facilities of the language we use for describing synchronous hardware controllers, to allow the design of modular systems. We illustrate the use of this approach to verify the controller of a CPU with decoupled access and execution units.

We have also modified the model checking algorithm to represent a state graph using binary decision diagrams (BDD's). Because this representation captures some of the regularity in the state space of sequential circuits with data path logic, we are able to verify circuits with an extremely large number of states. We demonstrate this new technique on a synchronous pipelined design with approximately 5 times 10^20 reachable states. We give empirical results on the performance of the algorithm applied to other examples of both synchronous and asynchronous cirucits with data path logic.