Automated Test-data Generation from Formal Models of Software

Sanjai Rayadurgam
Dept. of Computer Science and Eng., University of Minnesota, Minneapolis, MN 55455, USA
E-mail: rsanjai@cs.umn.edu

Verification and Validation (V&V) of software for critical embedded control systems often consumes upto 70% of the development resources. *Testing* is one of the most frequently used V&V technique for verifying such systems. Many regulatory agencies that certify control systems for use require that the software be tested to certain specified levels of coverage. Currently, developing test cases to meet these requirements takes a major portion of the resources. Automating this task would result in significant time and cost savings.

The **objective** of this dissertation is to automate the generation of such test cases. We propose an approach [2] where we rely on a formal model of the required software behavior for test-case generation, as well as, an oracle to determine if the implementation produced the correct output during testing.

A **key hypothesis** of this approach is that *model checking* can be effectively used to automatically generate test cases [3]. It is a technique for exploring the reachable state-space of a system model to verify properties of interest. When a property violation is detected, a model-checker will produce a counter-example as a witness, which is a sequence of states from the initial system state to the violating state, where each state is an assignment of values to system variables. Such a counter-example can be viewed as a test sequence. For example, we can assert to the model checker that a certain transition in a specification cannot be taken. If this transition in fact can be taken, the model checker will generate a sequence of inputs (with associated outputs) that forces the model to take this transition — we have found a test case that exercises this transition. One could systematically generate such "challenges" to the model-checker based on some user-defined levels of coverage of the software artifact and obtain a set of test sequences that achieves the desired coverage.

The **research challenges** that must be addressed in order to realize this test generation approach form the focus of this work:

Formalism and criteria. We need a suitable formalism for expressing the behavior of the software for critical control systems. It should have enough structure in terms of which meaningful test criteria can be expressed [3]. It should also be general enough to capture various software artifacts of interest, like requirement models [4] and program code.

Abstraction and test data instantiation. To make model checking feasible one may have to abstract away details. Software models pose serious challenges since they typically include integer and real valued variables leading to large or infinite state spaces. Abstraction will give rise to test-cases that have abstract data which should then be instantiated with concrete values. One may use constraint solving techniques to achieve this. Also, certain abstraction methods may reduce the size of the input domains without introducing abstract data [1].

Reducing test set size. A straightforward generation of test cases will inevitably lead to a large number of redundant test cases. For example, a test case devised to test one specific transition in a specification will also exercise many other transitions. One could leverage this fact and develop techniques to eliminate redundant tests.

Measuring effectiveness. The usefulness of the techniques and the test criteria must be empirically validated through case studies and experiments. Of particular interest are the scalability of these techniques to realistic models and the actual coverage of the program code achieved by the generated test suites.

References

[1] Y. Choi, S. Rayadurgam, and M. P. E. Heimdahl. Automatic Abstraction for Model Checking Software Systems with Interrelated Numeric Constraints. In *9th ACM Foundations of Sofware Eng. Symp.*, September 2001.

[2] M. P. E. Heimdahl, S. Rayadurgam, and W. Visser. Specification Centered Testing. In *2nd Automated Program Analysis, Testing and Verification Workshop*, May 2001.

[3] S. Rayadurgam and M. P. E. Heimdahl. Coverage Based Test-Case Generation Using Model Checkers. In *8th IEEE Eng. of Computer Based Systems Conf.*, pages 83–91, April 2001.

[4] S. Rayadurgam and M. P. E. Heimdahl. Test-Sequence Generation from Formal Requirement Models. In *6th IEEE High-Assurance Systems Eng. Symp.*, October 2001.

Proceedings of the 16th Annual International Conference on Automated Software Engineering (ASE 2001)
1527-1366/01 $17.00