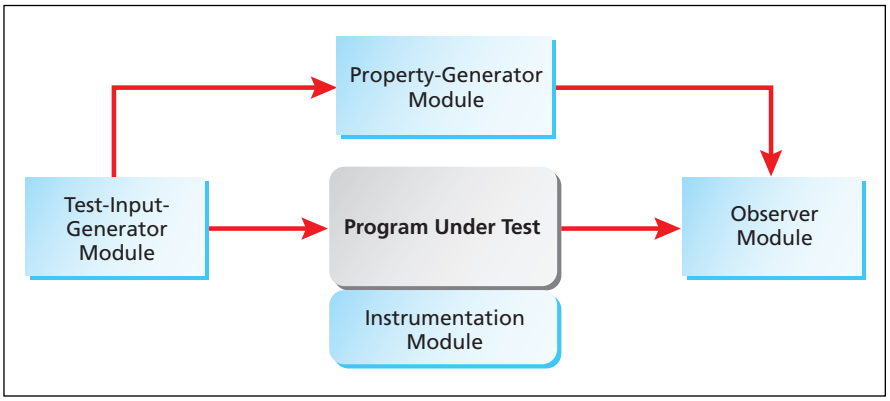
A Method of Partly Automated Testing of Software

Principles of symbolic execution and temporal monitoring are exploited.

Ames Research Center, Moffett Field, California

A method of automated testing of software has been developed that provides an alternative to the conventional mostly manual approach for software testing. The method combines (1) automated generation of test cases on the basis of systematic exploration of the input domain of the software to be tested with (2) run-time analysis in which execution traces are monitored, verified against temporal-logic specifications, and analyzed by concurrency-error-detection algorithms. In this new method, the user only needs to provide the temporal logic specifications against which the software will be tested and the abstract description of the input domain.

For testing a given computer program, the method involves the software analog of a hardware test harness, consisting of four software modules: a test-input-generator module, a property-generator module, a program-instrumentation module, and an observer module (see figure). The test-input-generator module automatically generates inputs to the program under test, one at a time, on the basis of a previously developed symbolic-execution approach. In this approach, symbolic values (instead of data) are used to represent program values and the state of a symbolically executed program is represented by a combination of (1) the symbolic values, (2) a program counter, and (3) a path condition in the form of a Boolean formula over the symbolic inputs that accumulate constraints that the inputs must satisfy in order to make execution follow a particular associated path.



Four Software Modules are used together to determine (1) what properties the program under test should have and (2) whether it does, indeed, have those properties.

An input generated by the test-input-generator module is fed to the property-generator module, which automatically generates a set of properties that the program under test is required to exhibit when executed in response to the given input. The input is then fed to the program under test. The program is then executed and generates an execution trace.

The instrumentation and observer modules perform the aforementioned run-time analysis. The instrumentation module must be constructed to report events that are relevant for determining whether the program exhibits the required properties during a particular execution. The observer accepts, as input, the execution trace and the set of properties generated by the property-generator module to determine whether the program exhibits the required properties.

The test-input-generator and property-generator modules must be constructed specifically for the program under test. It may be possible to automate the construction of the instrumentation module, depending on the nature of the program under test. The observer module is generic and can be re-used for testing other programs.

This work was done by Mike Lowry, Willem Visser, and Rich Washington of Ames Research Center; Cyrille Artho of Computer Systems Institute; Allen Goldberg, Klaus Havelund, and Corina Pasareanu of Kestrel Technology; Sarfraz Khurshid of Massachusetts Institute of Technology; and Grigore Roflu of the University of Illinois at Urbana-Champaign.

Inquiries concerning rights for the commercial use of this invention should be addressed to the Ames Technology Partnerships Division at (650) 604-2954. Refer to ARC-15244-1.