Consistency Techniques for Interprocedural Test Data Generation

Nguyen Tran Sy Yves Deville

Université catholique de Louvain
Place Saint-Barbe 2
B-1348 Louvain-la-Neuve, Belgium
{tsn,yde}@info.ucl.ac.be

Abstract

This paper[1] presents a novel approach for automated test data generation of imperative programs containing *integer*, *boolean* and/or *float* variables. It extends our previous work [1] to programs with procedure calls and arrays. A test program (with procedure calls) is represented by an Interprocedural Control Flow Graph (ICFG). The classical testing criteria (statement, branch, and path coverage), widely used in unit testing, are extended to the ICFG. For path coverage, the specified path is transformed into a *path constraint*. Our previous consistency techniques, the core idea behind the solving of path constraints, have been extended to handle procedural calls and operations with arrays. For statement (and branch) coverage, paths reaching the specified node or branch are dynamically constructed. The search for suitable paths is guided by the interprocedural control dependences of the program. The search is also pruned by a new specialized consistency filter. Finally, test data are generated by the application of the proposed path coverage algorithm. A prototype has been implemented. Experiments show the feasibility of the approach.

Keywords software testing, test data generation, procedures, arrays, constraint satisfaction, consistency

Introduction Structural testing techniques are usually concerned with the use of the control-flow a program to guide the generation of test data. The control-flow, in turn, is represented by a Control Flow Graph (CFG). To adequately test the program at the structural level, we must consider structural elements (nodes, branches, or paths) of the CFG for coverage. For example, *statement coverage* requires developing test cases to execute certain nodes of the CFG. Similarly, *branch coverage* requires test cases to traverse certain branches, and *path coverage* requires test cases to execute certain paths. Structural testing thus includes (1) choice of a criterion (statement, branch or path), (2) identification of a set of nodes, branches or paths, and (3) generation of test data for each element of this set. The automation of the last phase is a vital challenge in software testing.

[1]This paper has been published in the Proceedings of the Joint 9th European Software Engineering Conference and 11th ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE'03), Helsinki , Finland, 2003

Existing approaches Classical testing approaches can be classified into the following categories. *Random* test data generation consists in trying test data generated randomly until an element is executed. *Symbolic evaluation* consists in replacing input variables by symbolic values, and then symbolically evaluates the statements along a path. It is however limited in handling arrays and procedure calls. *Program execution based* (or *dynamic*) approaches start by executing the program with an arbitrary test input. This input is then iteratively refined, by execution of the program, to obtain a final input, executing a path, a branch, or a statement. Although dynamic approaches are powerful in handling arrays and dynamic data structures, it may require a great number of executions of the program. Other approaches, based on *genetic algorithms* or *Constraint Logic Programming*, have also been proposed.

In the following problem statement, an Interprocedural Control Flow Graph (ICFG) is a classical representation of programs.

Problem statement *Given a node n, a branch b or a path p of the ICFG associated with a test procedure P (possibly with procedure calls), generate a test input i such that P when executed on i will cause n, b or p to be traversed.*

We propose a novel consistency-based approach for interprocedural test data generation. Statement, branch and path coverage criteria are all handled. Path coverage is the core of our approach. It includes the following steps. (1) A path constraint is derived from a specified path of the ICFG. Such a constraint can involve operations with arrays. (2) The path constraint is solved by a new specialized interval-arithmetic-based constraint solver extended to handle constraints invol ving arrays. (3) A test case is extracted from the interval solutions.

For statement (and branch) coverage, paths reaching the specified node or branch are dynamically constructed. Our algorithm for path coverage is then applied on these paths to generate test data.

Contribution The main contribution of the paper is a novel approach (based on consistency techniques), which is an extension of our previous paper [1] to generate test data for numeric programs (programs with integer, boolean and float variables) with procedure calls and arrays. This approach handles *branch*, *statement* and *path* coverage criteria. Specific technical contributions of the paper include the following. (1) A new method to obtain a path constraint directly from a path's traversal. (2) Two mechanisms for passing parameters (pass-by-value and pass-by-reference) in procedure calls are handled. (3) An improvement on our previous consistency techniques to tackle specific constraints involving arrays. (4) The proposed interval constraints solver integrates integers, reals, and booleans, as well as the logical operators AND, OR, NOT. (5) For statement and branch coverage, interprocedural control dependences are used during the search process, when the test procedure contains procedure calls.

References

[1] Nguyen Tran Sy and Yves Deville. Automatic test data generation for programs with integer and float variables. In *16th IEEE International Conference on Automated Software Engineering(ASE01)*, November 2001.