Program Flow Analysis for Reducing and Estimating the Cost of Test Coverage Criteria

Martina Marré

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Advisor: Antonia Bertolino

November 1997

Abstract

Testing a software system consists of executing it over a suitable sample of input data and then checking if the output produced matches what was expected. Testing is widely used to enhance software quality, and, specifically, to uncover bugs that are inevitably introduced during the software development process.

One of the most difficult problems in testing is knowing when to stop the testing process. On the one hand, it is not possible in general to give an answer to whether a test suite guarantees the absence of faults. On the other hand, we need a way to limit the cost of testing. Therefore, it is useful to have criteria to determine when a program has been tested "enough". Ideally, the testing process should be planned in advance. However, in practice the tests are incorporated little by little, until some adequacy criterion is satisfied. In particular, different coverages can be used to determine when the program has been tested enough. The idea is to guarantee that each statement, decision or other feature of the program has been executed at least once under some test.

A major problem is that testing takes a considerable amount of the time and resources spent on producing software [**?**]. Therefore, it would be useful to have ways

1. to *reduce* the cost of testing, and

2. to *estimate* this cost.

In particular, *the number of tests to be executed* impacts heavily on the cost of testing. In fact, the time and resources needed for testing increase as the number of test cases increases. Hence, to reduce the cost of testing, the number of test cases generated to satisfy a selected test criterion should be as small as possible. Moreover, a bound on the number of tests that have to be performed to satisfy a selected test strategy can be used by managers and testers to estimate the effort needed to carry out the tests.

A test criterion in practice sets a collection of *requirements* to be fulfilled. For structural coverage criteria, these requirements are mapped onto a set of *entities* in the program flowgraph that must be covered when the tests are executed.

In this work we present a method for reducing and estimating the number of tests needed to satisfy structural coverage criteria, based on the new concept of *spanning sets* of entities. This concept is based on the observation that one test generally covers more than one entity. However, this fact is not traditionally considered when coverage is measured and not covered entities are identified, or when more tests have to be selected in order to augment coverage. Methods for selecting tests generate a test datum for covering one entity selected arbitrarily and considered in isolation from the other entities. If the generated test datum also exercises other entities, these will then be considered as covered a-posteriori. But no effort is made to generate a-priori test data that satisfy multiple requirements.

Our method overcomes these drawbacks by identifying with static analysis a *minimum* subset of entities with the property that any set of tests covering this subset covers every entity in the program. We call this minimum subset a "spanning set of entities". In this work, we first define the sets of entities that are associated with a whole family of popular test coverage criteria. Then we present a generalized method of identifying a spanning set of entities for the criteria considered.

Our method can be automated. Once it has been included in the software testing process, this information can be used for

- evaluating test adequacy more effectively;
- reducing the cost of testing;
- estimating the cost of testing;
- generating test suites.

In this thesis we present our study of the use of spanning sets of entities in coverage testing.