Augmenting Automatically Generated Unit-Test Suites with Regression Oracle Checking

Tao Xie
Department of Computer Science
North Carolina State University
Raleigh, NC 27695
xie@csc.ncsu.edu

Abstract. A test case consists of two parts: a test input to exercise the program under test and a test oracle to check the correctness of the test execution. A test oracle is often in the form of executable assertions such as in the JUnit testing framework. Manually generated test cases are valuable in exposing program faults in the current program version or regression faults in future program versions. However, manually generated test cases are often insufficient for assuring high software quality. We can then use an existing test-generation tool to generate new test inputs to augment the existing test suite. However, without specifications these automatically generated test inputs often do not have test oracles for exposing faults. In this paper, we have developed an automatic approach and its supporting tool, called Orstra, for augmenting an automatically generated unit-test suite with regression oracle checking. The augmented test suite has an improved capability of guarding against regression faults. In our new approach, Orstra first executes the test suite and collects the class under test’s object states exercised by the test suite. On collected object states, Orstra creates assertions for asserting behavior of the object states. On executed observer methods (public methods with non-void returns), Orstra also creates assertions for asserting their return values. Then later when the class is changed, the augmented test suite is executed to check whether assertion violations are reported. We have evaluated Orstra on augmenting automatically generated tests for eleven subjects taken from a variety of sources. The experimental results show that an automatically generated test suite’s fault-detection capability can be effectively improved after being augmented by Orstra.

1 Introduction

To expose faults in a program, developers create a test suite, which includes a set of test cases to exercise the program. A test case consists of two parts: a test input to exercise the program under test and a test oracle to check the correctness of the test execution. A test oracle is often in the form of runtime assertions such as in the JUnit testing framework. In Extreme Programming practice, writing unit tests has become an important part of software development. Unit tests help expose not only faults in the current program version but also regression faults introduced during program changes: these written unit tests allow developers to change their code in a continuous and controlled way. However, some special test inputs are often overlooked by developers and
typical manually created unit test suites are often insufficient for assuring high software quality. Then developers can use one of the existing automatic test-generation tools \cite{31,8,32,11,12,43,44} to generate a large number of test inputs to complement the manually created tests. However, without specifications, these automatically generated test inputs do not have test oracles, which can be used to check whether test executions are correct. In this paper, we have developed a new automatic approach that adds assertions into an automatically generated test suite so that the augmented test suite has an improved capability of guarding against regression faults.

Our approach focuses on object-oriented unit tests, such as the ones written in the JUnit testing framework \cite{19}. An object-oriented unit test consists of sequences of method invocations. Our approach proposes a framework for asserting the behavior of a method invocation in an object-oriented unit-test suite. Behavior of an invocation depends on the state of the receiver object and method arguments at the beginning of the invocation. Behavior of an invocation can be asserted by checking at the end of the invocation the return value of the invocation (when the invocation’s return is not void), the state of the receiver object, and the states of argument objects (when the invocation can modify the states of the argument objects). Automatic test-generation tools often do not create assertions but rely on uncaught exceptions or program crashes to detect problems in a program \cite{11,12}.

To address insufficient test oracles of an automatically generated test suite, we have developed an automatic tool, called Orstra, to augment the test suite for guarding against regression faults. Orstra executes tests in the test suite and collects the class under test’s object states exercised by the test suite; an object’s state is characterized by the values of the object’s transitively reachable fields \cite{43}. On collected object states, Orstra invokes observers (public methods with non-void returns) of the class under test, collects their actual return values, and creates assertions for checking the returns of observers against their actual collected values. In addition, for each collected object state $S$, Orstra determines whether there is another collected object state $S'$ that is equivalent to $S$ (state equivalence is defined by graph isomorphism \cite{8,43}); if so, Orstra reconstructs $S'$ with method sequences and creates an assertion for checking the state equivalence of $S$ and $S'$.

This paper makes the following main contributions:

- We propose a framework for asserting the behavior of a method invocation in an object-oriented unit-test suite.
- We develop an automatic test-oracle-augmentation tool that systematically adds assertions into an automatically generated test suite in order to improve its capability of guarding against regression faults.
- We evaluate our approach on augmenting automatically generated tests for eleven Java classes taken from a variety of sources. The experimental results show that our test-oracle augmentation can effectively improve the fault-detection capability of a test suite.

The rest of this paper is organized as follows. Section 2 presents an illustrating example. Section 3 presents our framework for asserting behavior of a method invocation in a test suite. Section 4 presents our Orstra tool for automatically augmenting a test suite.