CS 569 (Spring, 2016) Project Proposal

A Genetic Programming Approach to Automated Test Generation in TSTL

Yu-Chun Tseng
Apr. 19, 2016

1. Introduction

Testing is the widely used and adopted skill to verify and validate software systems. It is used to check which level of a software system is matching to its original design standard and to illustrate its correct operations [1]. Testing basically is a search problem that is related to identification of a finite number of good tests out of a shear, almost infinite number of possible test scenarios. "Good tests" are those runtime scenarios that are likely to uncover fails, or to illustrate correctness of the system under test(SUT). Recognizing good test cases generally follows predefined testing criteria, such as code coverage criteria [2].

Normally, a mortal tester promotes the test script and writes the test code for a SUT manually. Ideally, a testing tool should produce the whole test code automatically, but it is extremely tough to do so. Hence, there are parts of the testing procedure can be automated. The procedure of test automation can be split into 3 major actions.

- Generation of test scripts is based on testing criteria.
- Generation of a test predicts out of the SUT's standard.
- Combination of both, test scripts and predictions, into executable test cases.

Typically, an efficient method to achieve the automated generation of the test scripts is to adopt a random generator. Random testing can be adopted to make a volume of test scripts, but it does not exactly follow any test coverage criteria. Test tools are based on random testing, produce test scripts and simply check and demonstrate the coverage of the SUT. However, they cannot produce test scripts that are guided by the coverage.

2. Plan

In this project, I would like to propose genetic programming(GP) in TSTL. GP is specialization of genetic algorithm that is especially aimed at advancing computer programs based on the rules of natural development [3]. The chromosomes in genetic programs compose of arithmetic operations, mathematical functions, Boolean, conditional operations, and terminal symbols, such as types, numbers and strings. The genotype- phenotype mapping of GP is much more natural for the domain of test program generation comparing with standard genetic algorithm. In fact, GP is based on hierarchically organized trees requiring specialized genetic operators for recombination and mutation [3].

Recombination takes subtrees from formerly selected parent trees and swaps them in order to rebuild them into new trees. The chromosomes are often chipped and rebuilt at nodes, and not within nodes of the tree expressing the computer program. The mutation operator presents random changes in the tree by selecting a node of the tree randomly, deleting everything beyond that node, adding a randomly produced subtrees, or changing leaves of the tree randomly. These operations are all standard GP operators on the basis of [3].

3. Reference

[1]. IEEE. Standard Glossary of Software Engineering Terminology. Volume IEEE Std. 610.12-1990. IEEE, 1999.

[2]. B. Beizer. Software Testing Techniques. Van Nostrand Reinhold, 1990.

[3]. D. Koza. Genetic Programming, On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge, MA, 1992.