Project Proposal for CS569: Static Analysis and Model Checking for Dynamic Analysis

Xiao Han

April 19, 2016

I. Introduction

Random testing has been known as serval strengths: cheap to use, does not have any bias, and quick to find bugs. However, there are some weaknesses of common random testing. The test cases are generated randomly by the generator. There could have some meaningless test cases in unguided random testing. In this way, some think that random testing is not effective as systematic testing. There are some enhanced random testing algorithm. Adaptive Random Testing is one way to decrease the meaningless test cases by trying to generate test cases more evenly. In the paper "Feedback-directed Random Test Generation" [1], the authors find another way to improve the efficacy of random testing. The technique that authors present is improving random test generation by taking care of the feedback obtained from the input test cases that the generator created before [1]. If some of the input test cases, which have been tested, make no sense, the generator should avoid create a new test case with the same values. The authors give the algorithm of Feedback-directed Random Test. There are three sequences: error sequence, non-error sequence, and new sequence. Error sequence and non-error sequence are used to store the cases that have been tested. New sequence is used to store the new case that generator creates. First of all, the generator will create a test case and put it into new sequence. Second, checking whether the new test case, which is stored in the new sequence, has been tested before. If the new test case has been tested then create a new case. Third, executing the new sequence and check the feedback. If the feedback has no error then store the new sequence into non-error sequence. If the feedback has error then store the new sequence into error sequence. This algorithm will avoid lots of non-sense test cases. The authors implement the algorithm in RANDOOP. RANDOOP is a unit test generator for Java. It will automatically create unit tests for your classes in Junit format [2]. I want to implement Feedback-directed Random Test in TSTL. In the same time, let my random tester could deal with more SUTs and have a higher efficacy than unguided random test algorithm.

II. Plan of implementing

1. I will get more details about the algorithm, which is given by the authors, from the paper "Feedback-directed Random Test Generation".
2. I need to look up the source code of how the authors implement the Feedback-directed random test in RANDOOP.
3. I will look up the source code of random tester in TSTL.
4. After I get the details above, I will get the first edition of Feedback-directed Random

Test which implemented by python. The first edition will meet the requirement of milestone 1.

5. Changing the first edition and making it could deal with more SUTs to meet the requirement of milestone 2.
6. Finally, I will implement the final edition of tester and compare with the unguided random test to find out whether Feedback-directed Random Test is more efficient than unguided random test.

References

[1] Pacheco, Carlos; Shuvendu K. Lahiri; Michael D. Ernst; Thomas Ball (May 2007). "Feedback-directed random test generation". ICSE '07: Proceedings of the 29th International Conference on Software Engineering (IEEE Computer Society): 75–84.

[2] https://github.com/randoop/randoop