Project proposal

Zehuan Chen

OSU ID: 932541517

Instructor: Prof. Alex Groce

Introduction

The random testing is a simple and commonly used method in program testing. The testing generator creates the input for unit testing in order to replace the tedious work of manual error correction.

Although the random testing is very powerful during the error detection and coverage, the efficiency of generating test case still have chance to improve. “Feedback-directed Random Test Generation”, the paper that I have read present a technique that improves random test generation by incorporating feedback obtained from executing test inputs as they are created. Our technique builds inputs incrementally by randomly selecting a method call to apply and finding arguments from among previously-constructed inputs. As soon as an input is built, it is executed and checked against a set of contracts and filters.

An object-oriented unit test consists of a sequence of method calls that set up state (such as creating and mutating objects), and an assertion about the result of the final call.

The paper demonstrate that the new approach of random testing: Feedback-directed random test generation can outperform systematic and undirected random test generation, in terms of coverage and error detection. On four small but nontrivial data structures (used previously in the literature), our technique achieves higher or equal block and predicate coverage than model checking (with and without abstraction) and undirected random generation. On 14 large, widely-used libraries.

For example, For BinTree, BHeap and TreeMap, feedback-directed random generation achieved the same block and predicate coverage as shape abstraction.

My plan

In this semester, I plan to understand the core idea of random tester and error detection technique in software engineering area. Also, trying to implement some of the idea in the paper, the “feedback-directed random test generator” is based on random tester but add the function that we can make our future decision according to the previous actions.

So basically I could use some of the TSTL interface to save the labor and increase the accuracy, but first of all I need understand how TSTL works and how random tester generate the test case, then I need compare the action that TSTL does with the implementation that the new paper proposed, in order to find out the differences and make some feasible assumptions, would better improve it by using paper’s idea and my programing.

After that, I will following the pseudo code that paper provided and try to understand the method sequence and the container of test inputs, then I will using new method to generate the test case and using SUT.py and TSTL interface to calculate the coverage.

References

[1] Pacheco C, Lahiri SK, Ernst MD, Ball T. Feedback-directed random test generation. InSoftware Engineering, 2007. ICSE 2007. 29th International Conference on 2007 May 20 (pp. 75-84). IEEE.

[2] Pretschner, Alexander, Wolfgang Prenninger, Stefan Wagner, Christian Kühnel, Martin Baumgartner, Bernd Sostawa, Rüdiger Zölch, and Thomas Stauner. "One evaluation of model-based testing and its automation." In*Proceedings of the 27th international conference on Software engineering*, pp. 392-401. ACM, 2005.

[3] Chan FT, Chen TY, Mak IK, Yu YT. Proportional sampling strategy: guidelines for software testing practitioners. Information and Software Technology. 1996 Dec 31;38(12):775-82.