Feedback – directed Random Testing by using TSTL

Yifan Shen
04/19/2016

1 Background:

There are two controversial testing methods, the random testing and systematic testing. Random test is a black-box software testing technique where programs are tested by generating random, independent inputs. The results of the outputs are compared with the software specifications to verify that the test output is pass of fail.[1] Most of time, we use the random test to do some automated software testing, because the random test has many advantages, such as cheap to use, has no bias, quick to find bugs. However, the random test has its own disadvantages for the random characteristic. Different from the random test, the systematic test refers to a complete, conformance testing approach to software testing, in which the test unit is shown to conform exhaustively to a specification, up to the testing assumptions.[2]The systematic test usually has higher coverage and is able to generate some interesting tests. Attracted by the advantages of this two kinds of testing methods, many researchers try to find a method to combine these two methods together. One attempt is the feedback-directed random test.[3][4]

2 Introduction:

In the class, we have done many random tests by using BFS, DFS, swarm. After adding some manual faults into the AVL trees, sometimes the random test can find the bugs, sometimes it couldn't. In this case, the random test does some useless testing. So we want to find some ways to make the test become wiser. For example, as the work implemented by Carlos Pacheco, he randomly generates unit tests for programs. This test contains a sequence of method calls . When the test is ran, method or constructor will be invoked. All these operations are like the existing TSTL random generate test program. The new content is that all the testing sequences are legal. Each time the test program do a test, it will record the result and feed back the result to the program. The test program will generate new test sequence by the feedback from existing results. In this way, the random test program find more new branches and avoid repeated test sequence. For example, if there is a call a=1/0, if we use the feedback-directed random test, we do not need to generate the sequence of a=1/0, b=1/a; It is straightforward that by this way, many useless test sequences will be avoided.

3plan:

Because the feedback-directed random test is based on the random test, I will

spend two weeks to be familiar with the existing interface in the sut. After mastering these interfaces, I will write the first version of the improved random test following the guidance of the paper. After that, I will do some improvements in the first version to make it better.

References

1. Richard Hamlet (1994). "Random Testing". In John J. Marciniak. Encyclopedia of Software Engineering (PDF) (1 ed.). John Wiley and Sons. ISBN 0471540021. Retrieved 16 June 2013.

2. A J H Simons, A theory of regression testing for behaviourally compatible object types, Software Testing, Verification and Reliability, 16 (3), UKTest 2005 Special Issue, September, eds. M Woodward, P McMinn, M Holcombe and R Hierons (Chichester: John Wiley, 2006), 133-156.

3. Pacheco, Carlos, and Michael D. Ernst. "Randoop: feedback-directed random testing for Java." Companion to the 22nd ACM SIGPLAN conference on Object-oriented programming systems and applications companion. ACM, 2007.

4. Garg, Pranav, et al. "Feedback-directed unit test generation for C/C++ using concolic execution." Proceedings of the 2013 International Conference on Software Engineering. IEEE Press, 2013.