CS 569 Part3 Report

Name: Wei Liu

Email: liuw3@oregonstate.edu

ID: 932- 305- 582

Date: 5/5/2016

Section 1: Introduction

In my opinion, test generation for testing software is not a exactly perfect technique, because it is hard to find all bugs or mistakes. We don't have any tester or test generation can cover all SUT files to find bugs or mistakes. Even though all effective process may have error or revealing test cases, and sometimes most of test cases cannot be reachable. When you are trying to find any bugs by narrow paths, its disadvantage will be obvious. By this fact, we need a cheap and easy algorithm or machine learning for test SUTs. In my project, I will try to find a machine learning way or an easy algorithm, which is ART algorithm to collect data and try to figure out all bugs and in this way, I prefer to use this algorithm to cover software and find bugs and try my best to cover more than now. I think I can make it better.

Section 2: Algorithm Describe

This algorithm comes from Alex, which is our professor. I used it to test software and use it as black box algorithm. I think this algorithm can not operate the analysis part effectively. So I will try to find more improvement to make it perfect. I still need to search more articles or resources for this algorithm and reread our research paper. I think I can make it perfect before final part.

Section 3: Testing process of generation

When running the testing generation, users can just run in the command line with parameters. First of all, put the sut.py file and tester1.py file in the same directory. To run the test generation on the command line, use the following code:

python tester1.py 30 1 100 1 1 1 1

The first number, which is 30 indicates the test generation timeout will stop running.

That means the tester will run 30 seconds. The second represents the seed, python random. Random objects used for random number generation code. The third parameter, which is 100 indicates the maximum length of the test generation. The next represents the maximum memory is basically a search width. The next three parameters can be set to 0 or 1. Finally, the third parameter is the test generation SUT does not check for errors. The following two parameters means that the final report will generate coverage and branch coverage will be generated.

Section 4: References

[1] Alex Groce, Gerard Holzmann, and Rajeev Joshi. Randomized differential testing as a prelude to formal verification. Software Engineering, 2007. ICSE 2007. 29th International Conference on, pages 621-631. IEEE, 2007.

[2] T.Y. Chen, F.C. Kuo, H. Liu, and W.E. Wong. Code coverage of adaptive random testing. IEEE Transactions on Reliability, 62(1):226–237, March 2013. http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6449335&url=http%3A%2F%2Fieeexplore.ieee.org%2Fiel5%2F24%2F6471782%2F06449335.pdf%3Farnumber%3D6449335

[3] T.Y. Chen, H. Leung, and I.K. Mak. Adaptive random testing. http://www.utdallas.edu/~ewong/SYSM-6310/03-Lecture/02-ART-paper-01.pdf

[4] T.Y. Chen, F. Kuo, R. Merkel, and T.H.Tse. Adaptive random testing: The art of test case diversity. Journal of Systems and Software, 83(1):60–66, 2010.