CS 569 spring 2016 – Project

Competitive Milestone 1

Qi Wang

Onid: wangq5

Students ID: 932-439-151

May 5, 2016

Introduction

In the last term, I use the random test method to test the tstl program. The random testing method is a black-box software testing technique, so the program uses the randomly generated inputs[1]. It actually has many advantages. In the most complex systems, the random testing could produce some inputs that are would not think to try for users, For the most programs, this testing approach is useful, but there are still some programs are not always found bugs when using the random testing, because this method could not make sure every possible inputs would be tested. So I will start my implementation based on the random test generation. Then the main idea is that I will implement a new testing generation that could try to use fewer test cases to detect the software faults in TSTL.

Explanation of Algorithm

In this project, the main idea to implement a test generation as follows: use fewer test cases to detect the software faults in TSTL[2][3]. So this is a document for tester1.py file. In this project, I will implement a testing generator base on the information taught in the class. So the program will do the following things. Firstly, it will use generate the random test, and check the coverage of the statement. Then finding the coverage lower than a fix number, which means this part is important. Then storing these statements, when find a bug, printing it in the TSTL API. In the first part of this project, I define a function, which is the same function as the randomtester.py The function will add command line parameters, it also set input sys.argv to the parameters. Then define a function called make_config, which is also use the randomtester.py as reference, and it returns a dictionary.

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 0 1 1.The first number

denotes the timeout that the testing generation will stop when the program runs this time. The second denotes seed, which is for python random.random object used for random number generation in the code. The third parameter denotes maximum length of the test generated. The next represents the maximum memory that is basically a search width. The next three parameters can be set in 0 or 1. The last third parameter means the test generation will not check for faults in the SUT. The following two parameters mean that the final coverage report will be produced and also the branch coverage will be produced. In the future work, I will add more functions and make it more interesting. I will keep revising my code, I will try my best to define functions to handle failure.

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%3 A%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.