Project Milestone II

Bowen Yu

932-439-028

5/18/2016

Introduction

The random testing method is a black-box software testing technique, so the program uses the randomly generated inputs. 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 implement a testing generation that could try to use fewer test cases to detect the software faults in TSTL. I will use Adaptive Random Testing, which is an enhanced form of random testing. Adaptive random testing seeks to distribute test cases more evenly within the input space. It is based on the intuition that for non-point types of failure patterns, an even spread of test cases is more likely to detect failures using fewer test cases than ordinary random testing. Experiments are performed using published programs. Results show that adaptive random testing does outperform ordinary random testing significantly.

Explanation

They main idea of this project is to improve tester1.py. I will implement a testing generator to do that. For this project, I firstly generate the random test, and I make some random action and check if it is safe. When it is safe, this part is the bellowed mean, so I will test that part again. The function will add some new command line to test. It also set input to the parameters. Then testing generator will find the bugs and print them out.

Run the testing generation:

The user could run the test generation in the command line and write their only parameter, like timeout, seed, depth, width, faults, coverage and running. 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 indicates when the program runs this time, the testing generation will stop. The second is seed, it is used for generating a random number for python random object. The third parameter represents the 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.

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.