Part 2: Competitive Milestone 2

Student: Xinran Peng

Onid: pengxin

Class: CS569s16

Instructor: Prof Groce Alex

2016/5/18

Section 1: Introduction

Random testing is a black-box software testing technique where programs are tested by generating random, independent inputs. Results of the output are compared against software specifications to verify that the test output is pass or fail. In this project, we propose a modified version of random testing called adaptive random testing. This method 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.

Section 2: Algorithms

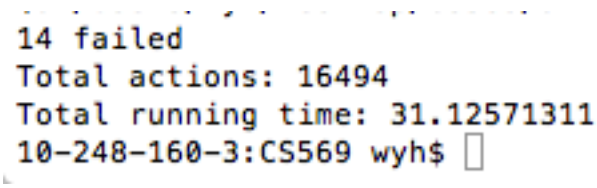
In this project, my idea of this algorithm came from Professor Alex. The main idea of this project is using fewer tests to detect the software faults in TSTL. The project also uses some special predefined values that have high tendency of finding faults in the SUT. In my tester2.py file, what I mainly do is use the random test to generate test cases, and then do the random action. Then the program will check if the action is safe action. If the action is safe, then it will check if the coverage of the random test is smaller than the mean, then it will test these parts. In the first part of the program, I use the same function as the randomtester.py. it will add command line parameters. Then define a function also used in randomtester.py called make_config, and returns a dictionary.

Section 3: Run Test Generation

This program is able to run in the command line to enter parameters (timeout, seed, depth, width, faults, coverage and running) with it. We can use the input 'python tester1.py 30 1 100 1 0 1 1' as an example. The first parameter is for the

timeout which the test generation will stop when the runtime reaches that number. The second parameter is for seeding which generates a random number of seeds. The third parameter is for the depth and The fourth parameter is for width. The fifth parameter is for fault which is if we put 1 that mean the test generation will search for bugs and if it is 0, means the test generation do not look to find bugs. The sixth parameter is for printing TSTL internal coverage report to know the statement and branches were covered during the test to guide it. This will work when we specify 1 and 0 for do not print it. The last parameter is for running which will print the elapsed time, total branch count and new branch. It works when we put 1, 0 will not do anything.

Here is the performance I got when using "python tester2.py 30 1 100 1 0 1 1".



Reference

[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] I. M. T.Y. Chen, H. Leung. Adaptive random testing, 2004.

[3] Richard Hamlet (1994). "Random Testing". In John J. Marciniak. Encyclopedia of Software Engineering (PDF) (1 ed.). John Wiley and Sons.