CS 569

Instructor: Professor Alex Groce

Student Name: Yipeng Wang

ID: 931903609

Date: May 18, 2016

Third Part

Competitive Milestone 2

Feedback-directed Random Test Generation in TSTL

Introduction

In my project proposal, I choose the paper "Feedback-directed Random Test Generation", which was written by Pacheco, Carlos; Shuvendu K. Lahiri; Michael D. Ernst and Thomas Ball. In that paper, author pointed out a kind of method which could test program efficiently, which was called feedback-directed random test. This method is based on the random test generation, which will analyze the input data, if it has certain feature, it will do certain implementation. By using feedback-directed random test method, it could reduce the unnecessary data tested in many times efficiently. Therefore, it could improve the efficiency on program testing.

Algorithm from Feedback-directed Random Test Generation

For the feedback-directed random test generation, it will generate the incremental data sequences. Then it will take four types of input: classes, contracts, filters, and time limit. In time condition part, it will randomly choose the sequences to append with the data sequences which we test previous. After appending, classify the result, for each sequence, it has a Boolean flag to determine the set of contracts and filters: if it is redundant or illegal, we will put it into error sequence and ignore it; if it has bugs, we will record the bugs in logs; if it is useful, we can use it again in future test appending.

What I did in tester1.py

In second part of the project, I took the newCover.py from the Professor Alex's website

as the reference. In this part, there are seven important part need to pay more attention. The first is the timeout, we use it to limit the time consumption in this testing generation implementation, which means if the program runs for enough long period to reach the limitation, the testing generation will automatically halt. The second part is seed, for this parameter, we can use it to set the random numbers in random.random object in python. The third part is depth, for this parameter, we can use it to set the maximum length of the test generation implementation. The fourth part is width, for this parameter, we can use it to limit the maximum width of the testing generation implementation. The fifth part is faults, this part is very important. For this parameter, it will only have 0 or 1 which depends on the checking faulting mechanics we defined in sut file, if it returns true, it needs to store the faults result of this testing generation; if it returns false, it will not record the result in sut. The six part is coverage, for this parameter, it will also only contain 0 or 1, which depends on the final coverage report producing, it will use TSTL's internalReport() function, if it returns true, it will print the report, else, it will not. The last parameter is running, it is familiar with the previous two parameter, it only has 0 or 1, if it returns true, it will print; else, it will not. For the algorithm, I divided the test into two phases, each part will contain the half of the whole running time. For the benchmark part, I sort the cover times of the action, then get the median number of the list. If the cover time of the action is below than median, it will store the result; else, it will skip that. In the second part, I run the states randomly to get the new list which could improve the coverage ratio.

Improvement in Tester2.py

In milestone 2, I added a new function for my tester file. Since, I use the newcover.py as the reference, which has quite a limit space to improve the performance, because in newcover.py, it runs the program twice then compare the difference, so I choose to add a new function in milestone 2. In this part, I add a function which could output the coverage report for users. We can not only check the internal report to see whether the bugs exist, but also check the coverage report to check the coverage rate like I did in last term.

Reference:

[1] C, Pacheco, S, K. Lahiri, M, D. Ernst, and T, Ball. Feedback-directed random test generation. In Proceedings of the 29th International Conference on Software Engineering, 2007.