CS 569 - Part 2 – Prof. Alex Groce

Eman Almadhoun – 932909951

Introduction:

It seems that creating test generation for testing the Software Under Test is a bit hard with misleading a perfect technique. Not every test generation can cover all the statements and branches of the SUTs. Especially when you want to get to the narrow paths for finding any missed bugs on them. With using a good cheap algorithm or some knowledge of machine learning, the test generation might be better to find most failures and cover most the code. In machine learning, there is a technique that we can explore thing to look at an interesting data to collect and exploit which means we get to this data. In my project, I will follow this method for find the bugs in the Software Under Test and put a lot of affording to maximize code coverage.

Algorithm Explanation:

The idea of this algorithm came from Professor Alex who guides us a lot. I want to improve this idea and test it over the branch coverage. My test method for testing the SUT is dealing with it as black box testing. This mean, I will implement the test generator without knowing the source code which I will test it. More explanation about the algorithm as follow: Firstly, I will run pure random tests which will generate random tests with random actions. After that, I will look up for any failure while the tests are generated. Then, I will collect all the tests in a big pool. I will calculate the average for this pool and try to find all the test below the average which might have uncaught failures. Then I put the result in another pool and try to do some statistics on them for exploit. In this stage, the algorithm is not doing the analysis part correctly. It needs more improvements to find all the bugs. So, next part will be more efficient. Now the algorithm sometime can find the bugs and others cannot.

Run Test Generation:

The test generation is able to be run in the command line which allows us to write parameters (timeout, seed, depth, width, faults, coverage and running) with it. To run the test generation on the command line for example: python tester1.py 30 1 100 1 0 1 1. 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.