Sahar Alizadeh
Prof. Alex Groce
CS 569

Part 2- Competitive Milestone
Feedback Directed Random Test Generation in TSTL

Email:alizades@oregonstate.edu

Introduction:
In this report I would explain about implementation of test generation programming using TSTL API and find improvement in testing part. The program was presented with name tester.py, which used sut.py.
There are 5 important command lines with these ascriptions:

1- Timeout: is represented time to cover a test with second.
2- Seed: It is for Python random objects and it used random number generation in the code
3- Depth: It is presented maximum length of a test in the algorithm.
4- Width: It is used for maximum memory/ BFS queue with searching width.
5- Faults: It is used for checking faults in SUT either is 0 or 1. If it is true, it should be saved for each discovered failures in the current directory as failure1.test failure2.test and etc.
6- Coverage: It should be reported coverage of testing as a final coverage report by using TSTL's internalReport() function.
7- Running: It should produced branch of coverage after running either is 0 or 1 by using TSTl randomtester.py.

The algorithm Feedback Directed Random Test Generator has two important properties –

Incremental Creation of Sequence
One method or multiple methods with arguments are randomly picked to generate a sequence. The randomly generated sequence is append to the previous sequence to create new sequence. This new sequence is checked if it is already present or not.

Contracts and Filters

Contracts are used to find errors in the system and filters are used to avoid bad tests.

Explanation of Implementation:

For this milestone, I have added code for random testing of creation of new branches. This code would test sut.py for the time passed as an argument.
In this code if the argument running is set to 1, then I am checking for new branches in sut.
If it is not equal to null set, then I iterate over the branches and print the time taken along with total branch count and the new branch in every iteration.
If it is null then I do nothing.

For the next milestone, I will write functions for each of the following arguments as bellows:
-Faults: For faults I will pass failure count and sut. I will write sut.failure in a file.
EnableValue – For this method I will pass sut and set a flag to false. Then I will iterate over sut.state and return flag or the variable.
-Filters: The arguments that I would pass to this method would be new sequence, number of action classes and depth. Would return either is less than max or is ok or is less than depth.
-New: For this method I will pass new sequence, error sequence and non-error sequences.
I will create a new set and iterate over error sequences. If the new set is less that the error sequence then I will return true. Then for the non-error sequences also will be compared with new set and true or false would be returned.
-Print: Sequence is passed as parameter and would print the sequence.
-Contracts: For this method sut, faults, failure count , and error sequences would be passed. If the faults were true, then I would increment the failure count and sequence to failure method to write the failure into a file. Also, for the error sequences set the failed sequence would be appended.