Part 2: COMPETITIVE MILESTONE 1

Zhen Tang

CS569: Static Analysis/Model Checking, spring 2016

Progress description

In my last submission, I have mentioned that I would like to implement Mirror adaptive random testing method in TSTL for this project. However, based on the professor's suggestion and my investigation, MART seems not an ideal and possible method for us to use in this project. Thus, I decide to found something little bit simpler but more easy to be fulfilled.

During this phase, the main work I did was get a further understand of TSTL. Since our main goal is to improve the test generate algorithm, we need to get a fully understand of the whole system. Based on the professor's paper, we can find that the main role of TSTL is to automatically create test harness, which is a set of valid tests for the system under test. And our tester is to run these tests in a proper way and record all its results. Thus, our test generator would not target to any specific structures or systems. It should aim to all of those common issues. After searching and reading related papers, i feel that the best way to do the test should base on random testing. So I decided to begin my algorithm design work based on the structure of basic random testing algorithm. As we known, the basic idea of random tester is to randomly pick actions from the list of all possible actions and run the test until finished running all the tests or hit the time limit. However, from some early works we can found that pure random algorithm is kind of 'least effective' one. Indeed, many researchers reported that the code that caused failure are always close to each other. And the non-failure region are always contiguous. Thus, I think one way to improve the random testing algorithm is, instead of only use simple random function, split the action list into some sub lists and each time pick test actions from different sub lists. As we discussed above, codes around non-failure code are more likely non-failure, so when finish running a test successfully, choose some tests which are far away from the current test would seems a batter way.

I am planning to improve this idea in two ways: the first one is to choose new test vectors that are as far away from existing test inputs as possible, which is from Tom Chen and his colleagues' work 'Fast Anti-Random (FAR) Test Generation to Improve the Quality of Behavioral Model Verification'. Another idea is to just randomly pick the sub list from the test action lists.

Currently, I already implemented a very simple version of the improved random algorithm based on the basic random tester and the idea of split action set into small subset. Although it is a really simple. In next several days, I would revise the code and trying to make all the arguments works as required. Moreover, I would also implement my another idea in later works.