Improve random testing in TSTL

Project proposal for CS569: Static Analysis/Model Checking, spring 2016

Zhen Tang

1 Project Description

Software testing is one of the most important methods to assure the quality of software. There exist some testing approaches which generate test cases in order to detect system failures. In these methods, random testing is a basic and wide-adopted one which just randomly pick objects from a set of all possible inputs, generate test cases and executing the program with those test cases. However, random testing were called “least effective” method by some people for using little information about the system under test in test case generation. That is the reason of people put forward the idea of Adaptive Random Testing, which is a method based on observations about failure patterns to improve the effectiveness of random testing.

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.

2 progress

During this weeks, basically I did two works. First is change the constant value of my action list split function based on my experiments. Data shows that split the list into too many parts didn’t really helps. So other than randomly generate the number of parts, I decide to fix it in to a constant. On the other hand, I revised the structure of the inner and outer loop to improve its performance.

3 future work

There are still some TSTL functions that I am not totally get understand so I think my tester may have better performance after replace some APIs.