Project Report 1

Zheng Zhou
Student ID: 932463162

May 5, 2016

1 Introduction

This report is for reporting the initial version of my test generation algorithm. In my last proposal, I was intending to implement Adaptive Random Tester using TSTL. Adaptive Random Testing algorithm is quite effective and efficient for finding failures based on random testing. Nevertheless, I felt that it is hard to implement and improve Adaptive Random Testing using TSTL API. Thus I plan to consider another simple but also effective algorithm for this project. My brand new algorithm in first phase will base on breath first search which has been proven as a slow but comprehensive algorithm. Due to the slowness of BFS, some of the bugs in SUTs have to take hours even days to explore. What I am trying to improve is instead of going through all nodes in one level, I set a time limitation for each level (depth) to explore deeper with less time required in the state tree. This is going to be first phase of my algorithm. In first phase I will collect the coverage data and statement information for using in next phase. After collected information in the first phase, I will restart the testing procedure. But this time I will focus on more interesting parts of the test (Not implement yet). More interesting parts of the test can be defined as the statements that did not cover too much. But instead of checking the statements that have coverage below mean, I will concentrate on the statements that have the least 1/3 coverage.

2 Implementation

For this first version, I only implement phase one for initially testing deeper in the structure. This will help me collect enough coverage data and statement information. Also I implement all of the seven parameters: timeout, seed, depth, width, faults, coverage, running. In my implementation user should not only type the parameter value but also type the option short name. For example for timeout, user should use -t option to specify the timeout value. These parameters are stored in Config variable. The initial testing state is stored in state_queue. Each element (state) in the queue will be pop out when go through the level. The main loop can be iterated as most *depth* times. That corresponds to the parameter depth (-d). The time limitation for each level is set by max_depth_time. For now it just a hardcoded variable which equal to 30, I will probably treat it as and program option in the future. The other constrain for each level is set by the *width* program option, meaning each level can only go through as most *width* state nodes. I also recorded a visited list for every nodes that have already gotten. For each level, the tester goes through the queue and pick up enabled actions.

3 Future Plan

The first version does not do too much jobs for now. The intuition of my algorithm is for comprehensively testing the SUT and also does pay more attention on critical part of testing. Hence the next phase I will focus on the statistic analysis of the coverage information that I got from phase one. I will continuously work on the collection part of phase one and mathematical model construction in phase two. Finally, I will try to enhance my algorithm if the result I get from phase two is not good enough.