Name: Changshuang Kou
ID: 932298781 (kouc@oregonstate.edu)
Due: 05/05/15

Lightweight Automated Tesing for TSTL

------ Milestone 1 for CS569 Project

1 Introduction

For the Lightweight methods, it is involved some concepts, such as the random test, adaption-based programming, reinforce learning and so on. First, let me ask what is a lightweight automated testing method? 1. Easy enough to implement that it is essentially available for all languages and environments: if it does not exist, anyone can code it up in an afternoon. 2. Easy enough to use that any programmer interested in writing automated tests can quickly code up a harness for small, moderate complexity, modules. 3. Fast enough to produce results quickly, so automated testing can be pursued if useful and abandoned if not productive. Next, I need to mention that the archetypal lightweight automated testing method is random testing. Last, testing with adaptation-based programming is like random testing, but different. The idea of testing with adaptation-based programming is that replace calls to pseudorandom number generator with calls to library for reinforcement learning.

2 Reinforcement Learning Algorithm

First, s means the state, a means action, Q(s, a) show the overall report of a state s return to give an estimation of the action a, r is the immediately re to 0port.

1. For every s and a, initialize the Q(s, a) to 0.
2. Observe the current state.
3. Repeat doing as following:
   Choose one action a and execute it, this action is action a that make Q(s, a) is the maximum .
   Receive the immediately report.
   Observe the new state s'.
   For the Q(s', a'), upload the data as following:
   $$Q(s, a) = r(s, a) + gama^* \max Q(s', a').$$
   s=s'.

The goal of Reinforcement Learning is to construct the control strategy to get the maximum performance.

In my project, I use reinforcement learning algorithm called SARSA(lamda) that is a fairly easy algorithm to implement in almost language. The framework for ABP-based testing is almost identical to that used for random testing.

3 Operations

1- Timeout: It will show the time to cover a test with second.
2- Seed: Generate the random number from code for the Python random objects.
3- Depth: Show the maximum length of a test in the algorithm.
4- Width: Show the maximum memory/BFS queue with searching width.
5- Faults: Checking the faults in SUT either is 1 or 0.

6- Coverage: It should be reported coverage of testing as a final coverage report by using TSTL's internalReport() function.
7- Running: After running either 1 or 0 by using TSTL randomtester.py, generate the branch of coverage.

4 Explanation for the work

In this milestone 1, I just do the random testing work, because the ABP-based testing is like to that used for random testing, but they are different. In this project, I will implement it at last. But, so far, I need to finish the random test first, and then improved it to the ABP-based testing in the following work. Above, I did not show the random testing algorithm, because it is just a part of my work, not all. And I think most important and hard work is reinforce learning and its' algorithm.

Reference

[1] Groce, A. Fern, J. Pinto & T. Bauer etl. "Lightweight Automated Testing with Adaptation-Based Programming" in Iternational Conference on Software Engineering, 2007.

[2] A. Groce, J. Pinto, P. Azimi & P. Mittal. "TSTL: A Language and Tool for Testing".

[3] C.Pacheco,S.K.Lahiri,M.D.Ernst,andT.Ball,"Feedback- directed random test generation," in *International Conference on Software Engineering*, 2007, pp. 75–84.

[4] http://baike.baidu.com/view/1882942.htm
[5] http://baike.baidu.com/view/1882942.htm