CS569
Instructor: Professor Alex Groce
Student: Kazuki Kaneoka
ID: 932-277-488
Email: kaneokak@oregonstate.edu

Part 1
Project Proposal
Feedback-directed Random Test Generation in TSTL

Background
Testing is significant in software developing. However, creating test case usually takes time. Also, it is difficult to create efficient test (in terms of the high percent of code coverage and the detecting bug). Because of this nature, the automated testing is highly demanded and very active area in academic and in industry. One issue that makes automated testing hard is about test data generation. What kinds of data should we test to get the high coverage ratio and to detect bugs? Since we can't test all possibility (it is easy to be exponential), we need to narrow and select somehow the number of input data that we are going to test. The simple way is to choose the input data randomly and test it under the system. But, since we do random generation, it is easy to create the same test or the equivalent test frequently. Therefore, we need to deal with this issue to avoid such redundant for random test generation. How to do that? There are many algorithms to improve random test generation to deal with this issue. The common idea among those algorithms is:
1. At the beginning of test, check the test data in previous test
2. And, create new test data such that the new test data differs from the previous one

How difference between new and previous test is depended on algorithms.

Describe Your Project Plan / What Is Your Idea For Test Generation
I would like to investigate in one of directed random test generations, *Feedback-directed Random Test Generation* [1]. The concepts of this algorithm are:
1. Create test data sequences incrementally.
   Pick up test data randomly and append it into the previous test data sequence.
2. Evaluate runtime information to identify the direction of the next test generation.
   After creating test data sequences by appending, execute it and classify the result: redundant, illegal, bug, useful. If it is redundant or illegal, we don't investigate this sequence anymore. If it has bug, we record it. If it is useful, we will utilize this sequence in the next test generation.

Feedback-directed Random Test Generation (FRT) is useful because we can avoid redundant and illegal test generation to reduce number of executions. However, the recent work showed that FRT had low coverage and unstable [2]. Although FRT showed low coverage and unstable, it provided lots of useful test generations. So, I would expect FRT is useful and might be more efficient than normal random test generation.

My plan for this project is:

1. Investigate relative papers. Mainly, 2 papers I mentioned in the previous sections [1] [2].
2. To come up with ideas such that how to apply FRT into TSTL. Specifically, pool sequences and classifying the result are key points in FRT. It is necessary to consider how to implement those futures in TSTL. There are Java libraries for FRT. So, my first approach might be to try how to use those Java libraries in TSTL.
3. Since FRT needs to check the previous sequence at each step, it might spend more execution time than normal random test generation. So, we need to have some trick to make it faster. My intuitive ideas are using “lookup table” or “hash table” to store the sequence of data.

Reference:

[1] C. Pacheco, S.K. Lahiri, M. D. Ernst, and T. Ball. *Feedback-directed random test generation*. In *Proceedings of the 29th International Conference on Software Engineering*, ICSE’07, pages 75-84, 2007.

[2] K. Yatoh, K. Sakamoto, F. Ishikawa, S, Honiden. *Feedback-Controlled Random Test Generation*. In *ISSTA 2015*.