Proposal

Name: Zixuan Zhao

ID:932 282 628

For my project, I am going to implement Feedback-directed Random Test Generation. This idea is from paper "*Feedback-directed Random Test Generation*" by Carlos Pacheco, Shuvendu K. Lahiri, Michael D. Emst, and Thomas Ball.

The mechanism of Feedback-directed Random Test Generation is to incrementally test functions in a sequence. Generally speaking, it is similar with the sequential variable selection for model selection in Statistics. We will randomly test each functions in a sequence. We consider each testing unit as the subsequence. Sequence will consist of these subsequences randomly and will be stored in a pool with these subsequences. To start a sequence, we will randomly pick up a subsequence, and check whether we have the same sequence in the pool after we append the subsequence into a sequence. If it is existed, we will randomly pick another one. Otherwise, we will test whether the updated sequence responses correctly. If sequence is tested correctly, we will classify it as normal and put back to the pool. If this sequence does not pass the test, will classify it as abnormal and output the error of this sequence. We will repeat these actions until enough times.

This method is somehow avoiding the conflictions in the order of each functions. On a aspect, it improve the efficiency of generating different testers. In the paper "*Feedback-directed Random Test Generation*", authors give us an example of Feedback-directed Random Test Generation on Java class, it is helpful for me to construct and apply it to TSTL. However, to implement this test generation into TSTL, there are mainly these problems.

The first thing is how to create a pool. This feedback test generator sequentially generates and tests. But, it is still a simple tree structure. Each sequence is like the path in this tree. What I need to consider is how to store each subsequence and sequence into the pool, since it is related to the efficiency of comparison in sequences and picking up appropriate subsequences.

The second is how to organize the actions, properties into this generator. The action is similar with subsequence. However, we can not record each new sequence as an action, which means I may need to think about a new variable or data structure to deal with it. And property can reduce the illegal input or actions so that avoids logic problem. In this case, it also reduces the redundant test generations. I would like to

The last one is to compare whether we already have the same tested sequence or generating an unappropriated sequence. As I mentioned in the first problem, it may relate with how I store data in a pool. What's more, the properties and actions in TSTL would also affect.

Here are the three main part I would like to research and implement in this project. What I expect from this generator is that it can reduce the redundant generation from random test a lot and avoid logic problem which is not suitable to the program. In this case, we can indicate more true error in the program. At the same time, I wish this test generator would be more efficient.

Reference

Carlos Pacheco , Shuvendu K. Lahiri , Michael D. Ernst , Thomas Ball, Feedback-Directed Random Test Generation, Proceedings of the 29th international conference on Software Engineering, p.75-84, May 20-26, 2007