CS569
Jingyuan Xu
xujing@oregonstate.edu
932428597

Project Millstone 2

Feedback-directed Random Test Generation

Introduction

In software testing method, random testing can be used to supplement functional testing. It mainly according to the tester's experience in checking functionality and performance for a software. Random testing can retest models not been covered by test cases in a software. It also can be used in testing new features in a software. The key points are checking special circumstances points, the special usage of the environment , and concurrency. Usually each of the test versions of software need to perform random testing, especially for the final version, it should pay more attention.

Tester2.py

Feedback-directed test method is a technique that improves random test generation. It can outperform systematic and undirected random test generation, in terms of coverage and error detection [1]. The code, which follow the method in Professor Groce's newCover.py file [3], implements the algorithm "Feedback-directed Random Test Generation". It contains three lists: newseq, error, noerror, they all use sut.currStatements() to store values. newseq stores all new values, error stores bugs, and no error stores non bug values. If program find a bug, then appened to error sequence, otherwise

For the first option, the code uses "**while** time.time()-start < int(sys.argv[1]):" to check if the time is out or not. If the condition successful, then use the method from "newCover.py", call randomAction(), to check the sut.safty in order to get bugs. Failure function is defined like:

```
if not ok:
  print(sut.failure())
error.append(sut.currStatements())
```

, if perivious steps failure, the bugs will record into error list. If no errors, then the code can store the non bug value with sut.currStatements(). If the coverage number be entered, then the code can call "sut.internalReport()" to show the coverage message.

The new changes for the algorithm implementation are:

1. give newSeq all statements value, then justify error or no error sequence

2. output a .out file when error sequence is not empty
3. According to the paper, there are two conditions for noerror sequence append sequences, and change “and” keyword to “or” keyword
4. minor errors changed in running option

Future Work

The algorithm I use is from paper “Feedback-Directed Random Test Generation”[1]. The algorithm in there requires sequences that can be executed and get the flag value equals to redundant or illegal, and then creating a new sequence. Also, the algorithm checks two sequences are equivalent if they translate to the same code, modulo variable names, if it is, then the algorithm will generate a new sequence again, and repect these steps. If it returns satisfied or violated, then we can get the test result. What I need to finish next phase is setup flags for sequences comparing.

Reference

[1] C. Pacheco, S. K. Lahiri, M. D. Ernst, and T. Ball, “Feedback-Directed Random Test Generation,” *29th International Conference on* Software *Engineering (ICSE'07)*, 2007.
[2] A. Groce and J. Pinto, “A Little Language for Testing,” *Lecture Notes in Computer Science NASA Formal Methods*, pp. 204–218, 2015.
[3] “agroce/cs569sp16,” GitHub. [Online]. Available at: https://github.com/agroce/cs569sp16/blob/master/suts/newcover.py. [Accessed: 06-May-2016].