Sahar Alizadeh
Prof. Alex Groce
CS 569
Part 3- Competitive Milestone 2
Feedback Directed Random Test Generation in TSTL
Email:alizades@oregonstate.edu

Introduction:
In this report I would explain about improvement of implementation of test generation programming using TSTL API and find improvement in testing part. The program was presented with name tester.py, which used sut.py.
There are 5 important command lines with these ascriptions:
1-Timeout: is represented time to cover a test with second.
2-Seed: It is for Python random objects and it used random number generation in the code
3-Depth: It is presented maximum length of a test in the algorithm.
4-Width: It is used for maximum memory/ BFS queue with searching width.
5-Faults: It is used for checking faults in SUT either is 0 or 1. If it is true, it should be saved for each discovered failures in the current directory as failure1.test failure2.test and etc.
6-Coverage: It should be reported coverage of testing as a final coverage report by using TSTL's internalReport () function.
7-Running: It should produced branch of coverage after running either is 0 or 1 by using TSTl randomtester.py.

The algorithm Feedback Directed Random Test Generator has two important properties:

Incremental Creation of Sequence:

One method or multiple methods with arguments are randomly picked to generate a sequence. The randomly generated sequence is append to the previous sequence to create new sequence. This new sequence is checked if it is already present or not.

Contracts and Filters:

Contracts are used to find errors in the system and filters are used to avoid bad tests.

Explanation of Implementation:

For milestone1, I have added code for random testing of creation of new branches. This code would test sut.py for the time passed as an argument. In this code if the argument running is set to 1, then I am checking for new branches in sut. If it is not equal to null set, then I iterate over the branches and print the time taken along with total branch count and the new branch in every iteration. If it is null then I do nothing.

New Improvement:
For milestone2, I added these functions as improvement for each of the following arguments as bellows:
1- Faults: First, I checked faults; if it is 1 it incremented bugs and write sut.failure to a file.

```
if FAULT:
bugs += 1
fault = sut.failure();
fname = 'fault'+str(bugs)+'.test'
wfile = open(fname,'w')
wfile.write(str(fault))
wfile.close()
sut.restart()
```

The number of bugs was 0.

2- Second, if coverage is 1 then it called internal report and it printed all these:

```
TSTL INTERNAL COVERAGE REPORT:
/Users/sahar/cs569_new_alex/cs569sp16/projects/alizades/avl.py
ARCS: 14 [(-1, 35), (-1, 45), (-1, 56), (-1, 258), (35, 36),
(36, 37), (37, 38), (38, 40), (40, -34), (45, 46), (46, -44),
(56, -55), (258, 259), (259, -257)]
/Users/sahar/cs569_new_alex/cs569sp16/projects/alizades/avl.py
LINES: 10 [35, 36, 37, 38, 40, 45, 46, 56, 258, 259]
TSTL BRANCH COUNT: 14
TSTL STATEMENT COUNT: 10
```

3- Print: Then I printed the following information:

```
print "Total Number of bugs",bugs
print "Total number of actions",actCount
print "Total Runtime",time.time()-start
print "FINAL POP BRANCHCOV",len(sut.allBranches())
print "FINAL POP STATEMENTCOV",len(sut.allStatements())
```

4- Mutation: I implemented new branches due to mutation with this result:

Added this mutate method

```
def mutate(test):
    tcopy = list(tet)
    i = rgen.randint(0,len(tcopy))
#    sut.replay(tcopy[:i])
    e = sut.randomEnabled(rgen)
    sut.safely(e)
    trest = [e]
    for s in tcopy[i+1:]:
        if s[1]():
            trest.append(s)
            sut.safely(s)
```

```
    tcopy = test[:i]+trest
    if len(sut.newCurrBranches()) != 0:
        print "NEW BRANCHES DUE TO
MUTATION:",sut.newCurrBranches()
    return tcopy
```

This mutate method will print the new branches due to mutation.

Example output -

```
NEW BRANCHES DUE TO MUTATION:
set([(u'/Users/sahar/cs569_new_alex/cs569sp16/projects/alizade
s/avl.py', (-1, 35)),
(u'/Users/sahar/cs569_new_alex/cs569sp16/projects/alizades/avl
.py', (259, -257)),
(u'/Users/sahar/cs569_new_alex/cs569sp16/projects/alizades/avl
.py', (46, -44)),
(u'/Users/sahar/cs569_new_alex/cs569sp16/projects/alizades/avl
.py', (38, 40)),
(u'/Users/sahar/cs569_new_alex/cs569sp16/projects/alizades/avl
.py', (40, -34)),
(u'/Users/sahar/cs569_new_alex/cs569sp16/projects/alizades/avl
.py', (35, 36)),
(u'/Users/sahar/cs569_new_alex/cs569sp16/projects/alizades/avl
.py', (-1, 56)),
(u'/Users/sahar/cs569_new_alex/cs569sp16/projects/alizades/avl
.py', (36, 37)),
(u'/Users/sahar/cs569_new_alex/cs569sp16/projects/alizades/avl
.py', (-1, 45)),
(u'/Users/sahar/cs569_new_alex/cs569sp16/projects/alizades/avl
.py', (45, 46)),
(u'/Users/sahar/cs569_new_alex/cs569sp16/projects/alizades/avl
.py', (-1, 258)),
(u'/Users/sahar/cs569_new_alex/cs569sp16/projects/alizades/avl
.py', (258, 259)),
(u'/Users/sahar/cs569_new_alex/cs569sp16/projects/alizades/avl
.py', (56, -55)),
(u'/Users/sahar/cs569_new_alex/cs569sp16/projects/alizades/avl
.py', (37, 38))])


FINAL POP BRANCHCOV 14
FINAL POP STATEMENTCOV 10
```