Part 3 Bug/progress Report in CS562

Kun Chen

Email: chenk4@oregonstate.edu

OSU ID: 932-702-228

Date: Mar. 1, 2016

1 Introduction

I will give a report about my primary achievement in testing a library of linked list via TSTL so far. As the tested library is gotten from Github in a personal website, then this program is expected to exist several bugs and I think it is a good library to be tested.

I have greatly improved the test system since the start. I selected 7 functions from the tested library, and in order to test these functions, I wrote 4 user defined functions in TESL. This content will be introduced in Chapter 2. I have found bug in one tested function called *removeElements*() function and the bug will be brief reported in Chapter 3. Finally, In Chapter 4, discussion about system under test (SUT) and coverage in the test will be described.

2 Description of functions

There are 7 functions in single linked list library which have been tested by TSTL so far, they are *Insert*, *printList*(), *ReversePrint*(), reverseList(), *getHead*(), *getTail*(), *removeElements*(). And *removeElements*() have been observed to have bugs, and other 6 functions have not been reported bugs. Its bugs are reported in Chapter 3.

In order to test the functions mentioned above, in my test system, I defined several user defined functions (UDF) in TSTL, such as *countLength*(), *findFstValue*(), *findLstValue*(), *findFstDuplicates*(). With these help functions, I could make assertion when testing a function in library. E.g., when testing insert function, I should use help function countLength() to get the length of list, and then compare the length of list before and after a number is inserted. Fig.1 shows the action in TSTL.

```
#insert the randomized numbers and test (insert function)
#Descprtion of (insert function): insert a value to the linked list
<LINKED>.insert(<CHAR>) => \
  (countLength(~<LINKED,1>)==pre<(countLength(~<LINKED,1>))>+1)
```

Fig.1 Action for insert() function in TSTL

3 Bug reports

Presently, bugs have been found in removeElements (). removeElements () in linked list library is a function to remove nodes from the singe linked list which have a given

value. In python, I found it could not work well for some cases: One example is that if the input linked list is [1,2,3,4,5]. we can not remove the value [1] in the first place from the list, for this case we are able to remove [2],[3],[4],[5] from the list. Another failure case is that when input linked list is [1,1,3,4,1]. we can not remove the first two value [1], [1] in the first place from the list, for this case we are able to remove [3],[4],[1] from the list.

Then we use TSTL to test the bugs in removeElements(). The action in TSTL is shown in Fig.2. We use help function findLastValue() to get the value at the tail of the single liked list and then remove all these values.

```
# ###### There are bugs in (function removeElements)
# test (removeElements function): we need use (findLstValue function ) to get the value
# in the tail of list and delete this value
# Description of (removeElements function): remove given value from the list
<LINKK>.insert(<INT>)
countLength(~<LINKK>) == 4 -> print "--------------------"; \
~<LINKK,1>.printList(<LINKK,1>.getHead());\
~<LINKK,1>.removeElements(~<LINKK,1>.getHead(),findLstValue(~<LINKK,1>)); \
~<LINKK,1>.printList(<LINKK,1>.getHead()); print "-       -     -     - "; => \
 (countLength(~<LINKK,1>) < 4)
```

Fig.2 Action for removeElements () function in TSTL

```
REDUCED IN 0.2863240242 SECONDS
STEP 0 INT1 =2
STEP 1 INT0 =2
STEP 2 LINKK0=linkedlist.linkedList()
STEP 3 LINKK0.insert(INT1)
STEP 4 LINKK0.insert(INT1)
STEP 5 LINKK0.insert(INT0)
STEP 6 LINKK0.insert(INT1)
STEP 7 print "--------------------"; LINKK0.printList(LINKK0.getHead());LINKK0.removeElements(LINKK0.get
Head(),findLstValue(LINKK0)); LINKK0.printList(LINKK0.getHead()); print "-       -     -     - ";

FINAL VERSION OF TEST, WITH LOGGED REPLAY:
INT1 =2    # STEP 0
INT0 =2    # STEP 1
LINKK0=linkedlist.linkedList()    # STEP 2
LINKK0.insert(INT1)     # STEP 3
LINKK0.insert(INT1)     # STEP 4
LINKK0.insert(INT0)     # STEP 5
LINKK0.insert(INT1)     # STEP 6
print "--------------------"; LINKK0.printList(LINKK0.getHead());LINKK0.removeElements(LINKK0.getHead(),
findLstValue(LINKK0)); LINKK0.printList(LINKK0.getHead()); print "-       -     -     - ";    # STEP 7
--------------------
{{{{{{{{{{{{{{{{{{{{{{{{{{{
2
2
2
2
}}}}}}}}}}}}}}}}}}}}}}}}}}}
{{{{{{{{{{{{{{{{{{{{{{{{{{{
2
2
2
2
}}}}}}}}}}}}}}}}}}}}}}}}}}}
-       -     -     -
ERROR: (<type 'exceptions.AssertionError'>, AssertionError(), <traceback object at 0x102b6e290>)
TRACEBACK:
  File "/Users/chenkun/tstl/generators/sut.py", line 3165, in safely
    act[2]()
  File "/Users/chenkun/tstl/generators/sut.py", line 2036, in act84
    assert  (countLength(self.p_LINKK[0]) < 4)
STOPPING TESTING DUE TO FAILED TEST
14.0211640212 PERCENT COVERED
0.482296943665 TOTAL RUNTIME
6 EXECUTED
552 TOTAL TEST OPERATIONS
0.137440443039 TIME SPENT EXECUTING TEST OPERATIONS
0.0123064517975 TIME SPENT EVALUATING GUARDS AND CHOOSING ACTIONS
0.000699043273926 TIME SPENT CHECKING PROPERTIES
0.138139486313 TOTAL TIME SPENT RUNNING SUT
0.00443911552429 TIME SPENT RESTARTING
0.286463975906 TIME SPENT REDUCING TEST CASES
51 BRANCHES COVERED
38 STATEMENTS COVERED
```

Fig.3 Screen shot of bug for removeElements () in TSTL

The bug is reported in Fig.3. In the analysis step, [2] is inserted into the list four times as in our actions, we restrict the length of complete list to be 4. Then we know the value in the tail of list is [2], and we want to remove all [2] from the list. Unfortunately, we could not remove [2] then TSTL have successfully reported this bug.

4. Discussions

My present work had tested 7 functions in the single linked list in the testing library. The coverage of the code is about 19%, as shown in Fig.4. One of the object is to increase the coverage to more than 30% in the final report. So I think I will select 3~5 more functions in the tested library and find the bugs in them. Of course, if possible, I want to have more assertions in the tested functions to see whether they could work well under more restricted conditions.

```
19.3121693122 PERCENT COVERED
1.20058703423 TOTAL RUNTIME
50 EXECUTED
5000 TOTAL TEST OPERATIONS
1.01793646812 TIME SPENT EXECUTING TEST OPERATIONS
0.0936789512634 TIME SPENT EVALUATING GUARDS AND CHOOSING ACTIONS
0.00521755218506 TIME SPENT CHECKING PROPERTIES
1.02315402031 TOTAL TIME SPENT RUNNING SUT
0.0247673988342 TIME SPENT RESTARTING
0.0 TIME SPENT REDUCING TEST CASES
68 BRANCHES COVERED
52 STATEMENTS COVERED
10-248-116-4:generators chenkun$ cat coverage.out
Name            Stmts   Miss Branch BrPart  Cover   Missing
-----------------------------------------------------------
linkedlist.py     256    204    122      5    19%   7-8, 12-13, 15, 26, 28-34, 37, 43,
 46, 48-49, 63-70, 82, 86, 103-349, 36->37, 45->46, 47->48, 71->-70, 85->86
```

Fig.4 Coverage display in TSTL

The quality of the system under test (SUT) is not quite good, as 1/7 of tested functions in the linked list has bugs. But the bug in removeElements () is not very severe, because it works for most of the cases, and readers do not need to do huge work to revise the bug. So I think the quality of SUT is higher than the median level. I also find some small bugs of TSTL, e.g., if we define the randomized number in one pool very huge, such as <[1..20000]>, the TSTL window will works very slowly.