CS569: Test generator Document

Oregon State University
Tien-Lung, Chang
Email: changti@onid.oregonstate.edu

This is a document for tester1.py. In this project I will implement a test generator base on the information cover in class. The program mainly do three things. First, generate a random test and let the action be a random action. Second, check whether the action is safety or not. If the action is not safety, we will determine it as a failure case and add it to our fail case list. If it is a safe case we will check whether the statement is valid or not.

To execute the program, we should input the 7 variable into the command line. The command will look like this " python2.7 tester1.py 30 30 100 1 1 1 1 ". The first argument is "Time". This argument will manage the time for test generator in second. We use the function "time.time()" to get the current time and check if it is below our input time. The second argument is "SEED". This argument is a seed for the function random.random() to generate random number. The third argument is "DEPTH". This argument define the maximum depth of how deep the test generator are. The fourth argument is "WIDTH". This argument is similar with the depth argument, but it is use in define the maximum width of how wide the test generator are. The fifth argument is the "FAULT" argument. This argument is a boolean variable, which indicate that the tester would like to check the faults or not by determine 1 or 0. If the user give the argument a integer 1, we will save test case for each discovered failure in the same directory. The sixth argument is "COVERAGE". This argument is also a boolean variable, which indicate that the final coverage report would be generated or not. Although we would collect all the coverage to guide the test, the generation for final report is depend. If the user give a integer 1, we will produce a coverage report by using the function define in TSTL, which is called "sut.internalReport()". The last argument is "RUNNING" argument. This argument is a boolean variable that indicate the running info branch coverage will be produces or not.

There are three function in the program. The first one will collect the coverage when the test is generating. The purpose for this function is to collect all the coverage to guide the test. The second function is failure(). This function will capture the bugs and collect the information of that case. After manipulate the failure case, we have to do sut.reduce() and sut.restart() to arrange the process. The last function is the newStatement() function. When the action is a safe case, we will check the length of the new statement and see if it is a valid statement. If it is a valid statement, we will add to our list pool. Finally, we will deal with all the coverage that we collect and get the mean coverage number.

Reference:

[1] Alex Groce coverTester.py. Retrived May 5, 2016, from https://github.com/agroce/cs569sp16/blob/master/SUTs/coverTester.py

[2] Alex Groce NewCover.py. Retrived May 5, 2016, from https://github.com/agroce/cs569sp16/blob/master/SUTs/newCover.py

[3] Chen, T., Leung, H., & Mak, I. (n.d.). Adaptive Random Testing. Retrieved April 19, 2016, from http://www.utdallas.edu/~ewong/SYSM-6310/03-Lecture/02-ART-paper-01.pdf

[4] E Godefroid, P. (n.d.). Compositional Dynamic Test Generation. Compositional Dynamic Test Generation. Retrieved April 18, 2016, from http://dl.acm.org/citation.cfm?id=1190226

[5] Zhang, S., Staff, D., Bu, Y., & Ernst, M. D. (n.d.). Combined Static and Dynamic Automated Test Generation. Combined Static and Dynamic Automated Test Generation. Retrieved April 18, 2016, from http://dl.acm.org/citation.cfm?id=2001463