Feedback-directed Random Test Generation
CS 569 Project Milestone # 1

Muhammad Bangash (932-676-646)

When I presented the project proposal, I decided to go with Kernel Density Adaptive Random testing but after seeing its complexity I decided to go with Dynamic Analysis using Model checking and Static Analysis mentioned in “Feedback-directed Random Test Generation”, which is raised by Pacheco, Carlos; Shuvendu K. Lahiri; Michael D. Ernst; and Thomsa Ball. The writers have found a new way to progress the efficiency of random testing and they have mentioned this in their paper. The techniques presented by the authors, they are improving random test generation by utilizing the response found from the input test cases which the generator has created before.

The algorithm I have used has three arrangements. 1. error sequence, 2. non-error sequence, and 3. new sequence. These error sequence and non-error sequence are being used to accumulate the cases that have been tested. The new sequence in algorithm is used to store the new case that generator creates. Starting from the beginning, the generator will create a test case and put it into new sequence. Secondly after checking the new test case, stored in the new sequence, has been tested before. When testing the new case has been tested, create a new case. Thirdly, now executing the new sequence and check the feedback. When the feedback has no error then store the new sequence into non-error sequence. If there is error in feedback, then store the new sequence into error sequence.

I have tried a to implement a basic algorithm for a novel test generation algorithm expending the TSTL API. My algorithm meets the basic necessity and support all of required command line by defining a function which is known as parse_args. I am using almost the same function as random tester.py. In my function I have added command line parameters arguments and set input sys.argv into this parameters and defined a function which is called make_config (pargs and parser). The functionality of this function is that it will return a dictionary, this will let me use command line by calling config function. I have also defined a function which is called check_action. This function will also get a random action using sut.randomEnabled(random seed) and this will check whether the action by using sut.safely(action). If the action is not performed as required, then it’s a bug and failure message is displayed. Tester1.py uses the algorithm of “Feedback-directed Random Test Generation”. There are three list: news, error, and nonerror. news will using to store sut.newStatements(). error will store the sut.currStatements() and it finds a bug. nonerror will store sut.currStatments() which will not contain a bug. The tester1.py allow user give a time how long will the tester run. alking about when the algorithm produces a new statement, the algorithm will check whether the statements have been tested. If statements have been tested, then it will generate a new statement.

In my algorithm I will try to put more parameters and SUTs. I will also try to make my algorithm more efficient.

REFERNCES:

[1]. P. S. Loo and W. K. Tsai. “Random testing revisited”. Information and Software Technology, Vol.30. No 7, pp. 402-417, 1988.

[2]. T. Y. Chen, H. Leung, and I. K. Mak, “Adaptive random testing” in Proceedings of the 9th Asian Computing Science Conference, LNCS 3321, pp. 320-329, 2004.

[3]. T.Y. Chen, F.-C. Kuo, R.G. Merkel, and T.H. Tse, “Adaptive random testing: the ART of test case diversity,” *Journal of Systems and Software*, vol. 83, no. 1, pp. 60–66, 2010.

[4]. Pacheco, Carlos; Shuvendu K. Lahiri; Michael D. Ernst; Thomas Ball (May 2007)."Feedback-directed random test generation". ICSE '07: Proceedings of the 29th International Conference on Software Engineering (IEEE Computer Society): 75–84.