Summary of
"Improving Structural Testing of Object-Oriented Programs via Integrating Evolutionary Testing and Symbolic Execution"

Nikhil Satish Pai Vishal Mishra Nikhil Anand

30th September 2015

Important Keywords:

ii1. Evolutionary testing*: refers to the use of metaheuristic search techniques for test data generation. Whenever a test objective may be expressed numerically, the deployment of evolutionary testing for the automation of test case design is possible. The appropriate formalisation of the test objective is the key to success. Each generated test datum is executed and its performance is evaluated with respect to the test objective.

ii2. Symbolic execution (also symbolic evaluation) is a means of analyzing a program to determine what inputs cause each part of a program to execute.

ii3. Argument Transformation: The argument-transformation allows jCUTE's symbolic execution technique to do concrete and symbolic execution on the primitive arguments. This paper's argument transformation process involves parsing method sequences generated by evolutionary testing to identify method invocations and for each method invocation requiring a method argument, they replace instances of concrete method arguments with equivalent symbolic arguments used to drive symbolic execution.

ii4. Chromosome Construction: chromosome-construction component constructs chromosomes out of method sequences generated using symbolic execution. these method sequences are made available to evolutionary testing through chromosome encoding.
Steps:
1st extract the method sequences from symbolic tests with a dynamic analysis mechanism
2nd entire method sequence is transformed to a chromosome.

*Wegener, J.: Evolutionary Testing - Overview. Workshop - Testprozeß, Testfallgenerierung, Testfallspezifikation, Munich, Germany, January 2002.

Brief notes:

iii1.Motivational Statement:

Achieving high structural coverage for unit testing of object oriented programs is difficult because of 2 reasons :

1. Some branches have complex logic which in order to solve, require deep understanding of the program.
2. Many methods have non primitive arguments which need to be solved in order to achieve the desirable state.

For the above problems symbolic execution and evolutionary algorithms have been proposed as solutions respectively. The paper suggests an integrated approach to solving this problem using a new standard framework called Evacon.

iii2.Baseline Results : The paper implemented Evacon framework in a tool for testing java programs. The effectiveness of this framework was tested against four tools which are eTOC, jCute, JUnit Fact and Randoop. The above were evaluated for a set of 13 experimental subjects. The average branch coverage for Evacon was 93.05% which was the highest among all the tools tested.

iii3.Related work: In order to generate test data cheaply, randomized processes are used by DART *,Randoop ** and Jartege***. Search based test generation uses genetic algorithms to find both method arguments and method sequences for the program under test but suffers from the path problem. This paper addresses the path problem by means of leveraging symbolic execution and constraint solving.

* P. Godefroid, N. Klarlund, and K. Sen, "DART: Directed automated random testing," in Proc. ACM SIGPLAN 2005 Conference on Programming Language Design and Implementation, 2005, pp. 213–223

** C. Pacheco and M. D. Ernst, "Randoop: Feedback-directed random testing for Java," in Companion to ACM SIGPLAN Conference on Object Oriented Programming Systems and Applications Companion, 2007, pp. 815–816.

*** C. Oriat, "Jartege: A tool for random generation of unit tests for java classes," .

Area of Improvement:

iv1. The unit test verifies coverage based on standard data structures and only two real world object classes, more real world examples need to be tested.

iv2. More third party tools need to be tested along with the existing ones.

Relation to Original paper :

The current paper introduces an integrated solution for testing coverage for a program.

The original paper proposes an improved algorithm of achieving higher coverage, in the current approach the combination of the two techniques is serial and cannot overcome nested predicates containing problematic constraints for both search and constraint-solving.

Reference to the Paper:

K. Inkumsah and T. Xie, "Improving Structural Testing of Object- Oriented Programs via Integrating Evolutionary Testing and Symbolic Execution," Proc. IEEE/ACM Int'l Conf.Automated Software Eng., pp. 297-306, 2008