Summary of
"*Combining Search-based and Constraint-based testing*"

Nikhil Satish Pai Vishal Mishra Nikhil Anand

28th August 2015

Important keywords:

ii1. Automated test data generators: These are an essential part of software engineering used in the generation of data sets to test the code coverage of a software application. They generate input data-set to traverse the code paths that need to be tested.

ii2. Constraint based testing: Provided an objective, constraints are extracted for a program. This objective is satisfied when the inputs satisfy the constraints. Inputs can then be derived using constraint solver to make the program follow this path. Classical Symbolic execution and Dynamic Symbolic execution are two approaches to the fulfillment of this method.

ii3. Search based testing: Search-based testing casts the testing problem as a search problem and applies efficient algorithms to find inputs that can serve as suitable tests. The search space is the input domain of the test object. A set of inputs to that test is a potential solution to this object. The search is guided by a fitness function that estimates how close a candidate solution is to satisfying a coverage goal.

ii4. Hybrid Search based and Constraint based testing: As the name suggests, this method is a combination of the aforementioned testing techniques, i.e. genetic algorithm and constraint solver are used to efficiently generate and explore various test scenarios.

Brief notes:

iii1. Motivation: Meta-heuristic search and constraint-based techniques are the two most successful approaches to automatically generate test data. However, both have their drawbacks; the search based techniques work adequately only if the heuristic provides sufficient guidance and the constraint based techniques do not fare well when judged on the number of types of constraints and on the scalability aspect.

iii2. Baseline Results : The prototype was evaluated on a set of standard benchmark examples comparing Random Search, Genetic Algorithm, Dynamic Symbolic Execution and the proposed Hybrid approach and the result was analyzed using the Branch Coverage obtained in each of the cases for all the benchmarks.

iii3. Related Work : An existing evolutionary tool was combined with a Dynamic Symbolic Execution by Inkumsah and Xie*. Lakhotia et al**. worked on combining floating points constraints with search based techniques; but this approach has been generalized for other constraints in this paper. Majumdar and Sen*** interleaved DSE with random search; however in this combination only one technique is used at a time.

iii4. Commentary: The prototype used in the evaluation of the algorithm was based on the Java PathFinder which is a verification and testing environment for Java integrating model checking, program analysis and testing.For the genetic algorithm, a search-based approach (Evosuite) to optimize whole test suites was used. A constraint solver (Choco) provided for an alternating search mechanism with constraint filtering algorithms.

Area of Improvement:

iv1. The conclusion states that the use of search techniques allows us to easily change the search objectives - all that is required is changing the fitness function; but there is no data or information available in the paper that asserts this statement.

iv2. The limit of 50000 test execution results for RA, GA and GA-DSE and the limit of 50000 constraint solving for DSE does not necessarily baseline them to the same level. If it does, there is no proof to support this assumption.

iv3. The empirical determination of 500ms as empirical timeout value provides no solid reason. Some methods may prove to be better at other timeout values than GA-DSE. Therefore, the prototype should be tested on these benchmarks for other timeout values also.

Reference to the Paper:

[1] Jan Malburg and Gordon Fraser. 2011. Combining search-based and constraint-based testing. In *Proceedings of the 2011 26th IEEE/ACM International Conference on Automated Software Engineering* (ASE '11). IEEE Computer Society, Washington, DC, USA, 436-439. DOI=10.1109/ASE.2011.6100092

* K. Inkumsah and T. Xie, “Improving structural testing of object-oriented programs via integrating evolutionary testing and symbolic execution,” in *Proceedings of the 2008 23rd IEEE/ACM International Conference on Automated Software Engineering (ASE’08)*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 297–306.

** K. Lakhotia, N. Tillmann, M. Harman, and J. de Halleux, “FloPSy - search-based floating point constraint solving for symbolic execution,” in *22nd IFIP International Conference on Testing Software and Systems*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2010, pp. 142–157.

*** R. Majumdar and K. Sen, “Hybrid concolic testing,” in *Proceedings of the 29th International Conference on Software Engineering (ICSE’07)*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 416–426.