Summary of
"Parallel Symbolic Execution for Structural Test Generation"

Nikhil Satish Pai   Nikhil Anand

5th October 2015

Important Keywords:

***ii1.* Symbolic Execution:** It is an automated technique for generating test cases to achieve high coverage. The execution is performed by executing programs with symbolic inputs, when conditional branches are encountered both the condition constraint on the branch and its negation are evaluated to generate the test cases.

***ii2.* Java PathFinder:**Java PathFinder is a software model checker for Java bytecode. It is a virtual machine that executes the program in all possible ways, checking for violations like deadlocks or unhandled exceptions along all possible execution paths. If errors are found then JPF reports the every step of how the defect was reached.

***ii3.* Simple Static Partitioning:**Execution methodology where a queue of path constraints are generated and are distributed among the various processors to perform dynamic symbolic execution.

***ii4.* Symbolic Execution Tree:**Tree representing all possible paths through the program which may be followed during symbolic execution.

Brief notes:

iii1.Motivational Statement

Symbolic execution techniques achieve high structural coverage but suffer scalability issues as the number of symbolic paths which need to be evaluated are very large; to solve this, the paper proposes a technique called simple static Partitioning which sets a pre-conditions on a symbolic execution tree to partition it and decrease the time required for execution of the tree.

iii2.Future Work:

The current technique works on partitioning path conditions generated during symbolic execution. The authors suggest that other partitioning techniques based on contexts like fault detection. Further the authors suggest testing the performance with different tree partitioning techniques along dynamic partitioning.

iii4.Baseline Results :

The technique was evaluated with four standard Java data structures Binary Heap, Binary Tree, Fibonacci Heap, TreeMap along with 2 real world classes Altitude switch and wheel brake system. The performance of Simple Static Partitioning was measured against random Depth first search. The time to evalute the test cases and generate tests meeting the coverage criteria were verified.

iii3.Related work:
The method of using parallel techniques for explicit state space exploration was explored in [9, 12, 13, 24].
[7, 15], operate by dynamically partitioning the symbolic execution tree for load balancing.
Boyapati et al[5] use Korat, an explicit state enumeration technique for generating test inputs in a Java method
[15], King parallelized symbolic execution using a dynamic load balancing approach.

Area of Improvement:

iv1.Testing needs to be done on more input types to check the performance on different types of symbolic execution trees
iv2.Testing bounds need to be varied to check the effect on the results.

Relation to Original paper :

The paper evaluates real world examples of the Altitude Switch from the avionics domain, a Wheel Brake System from the automotive domain along with standard java data structures to evaluate the performance of the proposed algorithm. The original paper uses these real world data structures to compare the performance.

Reference to the Paper:

M. Staats and C. S. Pasareanu. Parallel symbolic execution for structural test generation. In Proceedings of the Nineteenth International Symposium on Software Testing and Analysis (ISSTA 2010), pages 183–194, 2010.