Wyatt Kormick
CSCI 5802
Reading 3
4/9/18

Symbolic software execution is a way of thinking about software in order to generate test cases that might not normally be found by other methods. The main idea is that the software under test is abstracted to a tree of execution paths, a software state consisting of symbolic assignments to variables, and a constraint to the value of the variables that allows a specific path to be executed. The state changes every time a variable is (re)defined at a point. At every branch the constraint changes to the required variable assignments that would allow that branch to be taken. When multiple branches occur, the constraints are 'and'ed together. Finally, for every different path generated, a constraint satisfaction program is used to find concrete values that satisfy the path constraint. If it finds a concrete value, then it has found a new test case, if not, the path is infeasible and can never be taken. If the program isn't too complicated, then this method will create a test case for every feasible path in the program.

In practice, this can have a few problems. One problem that may occur is nonlinear variable definitions. When a variable's set of possible values becomes too complicated, it becomes much more difficult to reason about how the variable can be evaluated in relations. Another major problem is "path explosion". When a program has a large branching factor, or loops that can run many times, the number of paths to check can grow exponentially. In such cases the number of paths needs to be cut off somewhere leading to an incomplete "solution". A third problem stems from the overhead involved when running the constraint satisfaction program. When a path contains many branches, the path constraints start to become overly complicated. The more complicated the constraint gets, the slower the satisfaction program gets, and it is already relatively slow. There are versions of the symbolic execution strategy that fix one or more of these problems but they come with a trade off, usually of completeness.