# Chapter 4: Hypothesis

## Problem Statement:

Symbolic Execution, the latest state-of-the-art tool in dynamic bug detection, could theoretically achieve 100% code coverage thereby assuring that all possible paths are taken through ha program. However, the high code coverage comes at a price – path explosion – which leads to memory exhaustion and long runtimes. The path explosion is a direct result of the number of possible paths through a program that the executor has to traverse. Currently code coverage is sacrificed in order to contain the path explosion to ‘managable’ memory and runtime limits.

Symbolic Executors rely on constraint solvers to evaluate the complex sets of linear equations with multiple variables (symbols) to determine exactly which path to take. Are the constraint solvers more influenced by the number of constraints (equations/conditions), or the number of variables per constraint?

## Hypothesis:

My hypothesis is that it is actually the number and representation of the constraints that are the major contributor to the memory exhaustion and increasing runtime. I propose that there are portions of the constraint solver code which could be optimized thereby reducing the affects of path explosion and allowing the coverage limits to be increased.

## Proposed Approach:

I will demonstrate through a series of tests that the number of constraints has a greater impact on runtime and memory usage than does the number of symbols involved in the constraints. I will create a series of smtlib2 format constraint files which can be analyzed by the z3 constraint solver and record the runtime and memory usage for each test. Based on the results of these tests, I will trace the z3 solver to identify potential sections of code to be optimized.