Bachelor Thesis Description

Recording Symbolic Executions

Andreas Buob

Supervisor: Malte Schwerhoff

Introduction

Silicon is a verifier for the intermediate language Silver, an intermediate verification language for permission-based, automated program verification[1]. Verification is done using symbolic execution, meaning that the program isn't executed with actual input, but with symbolic values. A statement such as `x := s; x := x * 2`, where `s` is a local variable with symbolic value `s'`, does not result in a state where `x` has a concrete value, but rather a state where `x` is constrained by `x == s'*2`. Furthermore, the symbolic execution can branch into different execution paths. Symbolically executing a statement such as `s1; if (b) then s2 else s3; s4;` will branch execution after the statement `s1`: Down one path, `s2; s4` is executed under the assumption that `b` holds, down the other path, `s3; s4` is executed under the assumption `!b`. Paths can also be joined again, which leads to merged states.

This project consists of two parts: First, finding a way to record the individual steps of a symbolic execution and second, visualise the recorded executions. Since different possibilities on visualising symbolic execution have already been inspected by previous work[2], the focus of this project lies on recording symbolic executions.

Core Tasks

- Design and implement a datastructure that allows one to record the individual steps of the execution, including their current states
- Implement an execution recorder that provides an understandable API, whose calls can be integrated into the symbolic execution engine in an unintrusive and elegant way
- The execution recorder should be extensible so that recording the execution of features that will be added to Silicon in the future is facilitated
- Design and implement a prototypical visualisation that demonstrates that the recorded data can be used to inspect symbolic execution

Possible Extensions

- Improve visualisation and allow specialized views as described by previous work[2]
- The recorder supports additional Silver features such as magic wands or quantified permissions

References

[1] Uri Juhasz, Ioannis T. Kassios, Peter Müller, Milos Novacek, Malte Schwerhoff, Alexander J. Summers. *Viper: A Verification Infrastructure for Permission-Based Reasoning.* Technical Report, ETH Zurich, 2014

[2] Ivo Colombo. *Debugging Symbolic Execution.* Master's Thesis, ETH Zurich, 2012