Supporting Sequence Axiomatization on the SMT Solver Level for the Viper Project
Project Description

Lukas Schär
Supervisors: Arshavir Ter-Gabrielyan, Peter Müller
ETH Zürich, Switzerland

November 26, 2016

1 Introduction

The Viper framework[1] is an automatic deductive software verifier striving to achieve a good compromise between interactivity and full automation. Users benefit from an SMT solver for automatic verification of formulas, yet they only need to provide code annotations instead of full proof assistance. The annotated code is translated into the intermediate language Silver, from which it can be translated further by multiple back ends to the SMT solver level.

The power of this solver is based on the theories it incorporates, which include for example first order logic and linear arithmetic. For program verification, sequences are another interesting theory. They can represent the concept of connected data structures such as arrays or lists, and are therefore often used.

Viper uses the Z3 SMT solver[2] in a version which does not use such a sequence theory. Instead it encodes sequences as partially axiomatized mathematical functions. Unfortunately, this implementation has some weaknesses regarding completeness and performance. However, the Z3 solver has recently been upgraded with its own sequence and string theory[3]. The goal of this project is to investigate the problems of Viper's current approach based on the previous version of the Z3 solver and compare its performance with the new interface of Z3, and use the newly provided functionality of Z3 to possibly present a more stable solution, which improves performance as well as completeness.

[1]http://viper.ethz.ch
[2]https://github.com/Z3Prover/z3
[3]http://rise4fun.com/Z3/tutorial/sequences

2 Core Goals

The current implementation of sequence support in the Viper framework encodes sequences as a set of partially axiomatized mathematical functions which are passed on to Z3. This encoding is incomplete, meaning automated verification of correct Viper programs is not ensured. It also leads to performance problems, as the heavily used universal quantification of the sequence axioms can introduce intricate triggering problems, causing strange performance drops. The first part of the project will investigate the sequence axioms and the interfaces used in Viper to comprehend the current implementation and to develop a better understanding of some present problems.

The next step will be collecting several Viper programs that exhibit problems which can be traced back to the current sequence implementation. These programs can be used to further understand the problems of the axiomatization, as they provide tangible examples for the complications the current implementation produces.

Following this the new Z3 sequence support will be studied. We will try to improve the programs collected in the previous part of the project by manually encoding them with the new interface. This will improve the understanding of the interface and might highlight some of its potential restrictions. At this point we will also perform first evaluations by comparing Z3's execution time of the manually encoded programs to the execution time with the current implementation.

As the last major part of the project, we will alter Silicon, Viper's existing symbolic execution engine, to use the new Z3 support instead of the existing encoding. To conclude the project, we will compare this variant with the original Silicon back end with respect to execution time and memory footprint using the Viper test suite as input.

In short, this project will cover:

- Understanding the existing sequence axioms and interfaces.
- Collecting problematic Viper examples which include sequence axioms.
- Understanding the Z3 interface for sequences, manually encoding simple examples.
- Evaluating these Z3 encoded examples based on runtime.
- Modifying the current symbolic execution engine for native sequence support in Z3.
- Collecting statistics on performance changes to evaluate the new implementation.

3 Extensions

- Finding heuristics for deciding when to use sequence axioms or Z3's built-in sequences in concrete situations.
- Investigate the sequence encoding of Carbon (Viper's Verification Condition Generation back end).

- Improve debugging support for sequences.
- Enhance array support.

4 Schedule

Task	Estimated time
Finding/Encoding problematic examples with sequences in Viper	3/4 month
Manually encoding examples with Z3 sequences	1/2 month
Preparing the initial project presentation	1/4 month
Evaluating the encoded Z3 examples	1/4 month
Modifying Silicon and implement new interface	1 month
Collecting statistics for different sequence approaches	1/4 month
Extensional tasks	2 months
Writing report	3/4 month
Preparing the final project presentation	1/4 month
	6 months