Integration and Analysis of Alternative SMT Solvers for Software Verification

Master's Thesis Project Description

Frederik Rothenberger

October 9, 2015

1 Introduction

A common approach in software verification is to encode a given program property as boolean formulas in order to verify that this property holds in the context of said program. These boolean formulas are then verified by a satisfiability modulo theory solver (SMT solver). Although some logical theories are decidable, most program properties require undecidable theories to be expressed. SMT solvers try to approach these undecidable problems using clever heuristics and exhaustive search (while risking infinite runtime).

The heuristics are usually hidden away under opaque abstraction layers. It is therefore difficult to predict the runtime based on the input problem, especially since small changes in the formulation can make huge differences in runtime. This leads to the unsatisfactory situation that tools relying on SMT solvers might become unpredictable. Even worse, it is sometimes unclear how to understand such problematic situations and avoid them, due to the lack of powerful debugging tools for SMT solvers.

There are multiple SMT solver implementations available. In this project an analysis of these solvers will be made in order to gain a deeper understanding of the power of these various implementations as well as the trade-offs chosen with respect to the solver heuristics. This understanding should then translate to improvements to the Viper software verification tool chain [5].

2 Challenges and Core Goals

In this section the main goals and challenges are outlined.

Addition of an alternative SMT solver to the Viper tool chain. Another SMT solver, namely CVC4 [2], will be added to the existing Viper tool chain. This includes adding CVC4 support to both Silicon and Boogie [1], which is the back end to Carbon. Challenges might arise from the fact that the current implementation relies on specialities of Z3 [4], the current SMT solver implementation of the Viper tool chain.

SMT solver analysis. CVC4 will be compared to Z3. The main goal of this analysis is to get a good understanding of the performance of these solvers given various kinds of problems. This might allow to formulate the SMT solver input in a way that avoids formulas which are hard to solve.

Another area of special interest are differences with regard to the solver heuristics used. Such differences, if any, could be exploited by having both solvers available to the Viper tool chain and handing each problem to the solver better suited for this particular problem.

Further direction. The further direction of this thesis will be decided based on the results of the SMT solver analysis. One option is the 'Analytical Tools and Profiling' direction which deals with improving analysis tools and inspecting the inner workings of SMT solvers. Alternatively, there is also the 'Exploiting Differences' direction with the goal to translate the findings of the core analysis into improvements to the Viper tool chain. These two different directions are outlined in section 3 and 4 respectively.

3 Analytical Tools and Profiling

The following goals are within the scope of this direction:

- **Core:** Extending the Z3 Axiom Profiler bundled with VCC [3]. This debugging tool allows to analyse the inner state of Z3, but only in a hierarchical text based form. Visualizing this information would help convey the information provided and facilitate debugging.
- **Core:** The analysis of Z3 and CVC4 should be extended to include additional analysis tools such as performance profilers. Rich statistics should allow for in depth comparisons of various performance metrics.
- **Extension:** Adapt tools so that the Z3 Axiom Profiler can be used to analyse CVC4.
- **Extension:** During this thesis a need for more powerful debugging tools and logging facilities might arise. These tools should then be designed and implemented.
- **Extension:** The Axiom Profiler should be extended to detect and visualize matching loops. Matching loops are a common issue with adverse problem formulations. More specifically, it means that some facts containing quantifiers get instantiated over and over again unnecessarily.

4 Exploiting Differences

This direction deals with the following goals:

- **Core:** Design additional tests to really pinpoint the differences between the solvers and classify adverse problem formulations. The Viper tool chain should then be improved to avoid such formulations.

- **Core:** Implementation of solver selection for the Viper tool chain. The goal is to automatically select the best SMT solver with appropriate configuration depending on the input problem.
- **Extension:** Investigate fine grained problem splitting. In addition to the coarse solver selection based on the whole input problem, finer grained problem splitting could increase performance of the Viper tool chain. A given problem would have to be encoded in several independent subproblems. These subproblems could then be solved separately by the most suitable solver based on the new, smaller formulation.
- **Extension:** Explore alternative problem formulations. Some problems can be formulated in different ways. Formulations that fall in to the high difficulty category should be avoided and replaced by alternative encodings.

References

[1] Mike Barnett, Bor-Yuh Evan Chang, Robert DeLine, Bart Jacobs, and K Rustan M Leino. Boogie: A modular reusable verifier for object-oriented programs. In *Formal methods for Components and Objects*, pages 364–387. Springer, 2006.

[2] Clark Barrett, Christopher L Conway, Morgan Deters, Liana Hadarean, Dejan Jovanović, Tim King, Andrew Reynolds, and Cesare Tinelli. CVC4. In *Computer aided verification*, pages 171–177. Springer, 2011.

[3] Markus Dahlweid, Michal Moskal, Thomas Santen, Stephan Tobies, and Wolfram Schulte. VCC: Contract-based modular verification of concurrent C. In *Software Engineering-Companion Volume, 2009. ICSE-Companion 2009. 31st International Conference on*, pages 429–430. IEEE, 2009.

[4] Leonardo De Moura and Nikolaj Bjørner. Z3: An efficient SMT solver. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340. Springer, 2008.

[5] U. Juhasz, I. T. Kassios, P. Müller, M. Novacek, M. Schwerhoff, and A. J. Summers. Viper: A verification infrastructure for permission-based reasoning. Technical report, ETH Zurich, 2014.