Master Thesis

From Viper to Grasshopper

Andrea Helfenstein

2016-04-07

1 Background

The Viper project[1], developed at ETH Zürich, provides a number of tools for verifying programs written in the intermediate language Silver. Verification problems of higher level concurrent programming languages can be encoded in Silver. Currently there are two backend verifiers available to check these encoded programs. Both verifiers use the SMT solver Z3 to prove the relevant properties.

The GRASShopper tool[2][3][4], developed at New York University (NYU), can verify programs at a similar level of abstraction, but only for a subset of the logic that is supported by the Viper tools. This restricted logic is decidable and its use requires fewer annotations in the source program.

The goal of this project is to investigate the use of GRASShopper as an alternative backend verifier for Viper. The main goal will be to automatically translate a subset of Silver to the GRASShopper input language. While Viper supports arguing about heap state and permissions in a general way, GRASShopper supports only a subset of these problems using a very specialized logic. Because of this restriction, GRASShopper needs less direction from the user in the form of annotations than Viper. Additionally, the GRASS logic is decidable, and the GRASShopper tool can generate counterexamples for failing verifications. It will be interesting to see, if the use of the GRASShopper tool as a backend for Viper could reduce the number of annotations needed in Silver, and if using GRASShopper instead of the current Viper backends would be more efficient for the supported subset of problems.

2 Core Tasks

- Identify which subset of problems supported by Viper correspond to the set of problems supported by GRASShopper. In order to get an

understanding of this subset and the relationship between Viper and GRASShopper, it will be useful to create some examples of analogous programs by hand.

- Define a mapping from problems in Viper to directly analogous problems in GRASShopper. By "directly analogous" we mean that the problems don't need complex rewriting, e.g. the representation of a linked list in Viper will be translated to the corresponding encoding of linked lists in GRASShopper.
- Implement the mapping from Viper to GRASShopper, and map the results back to Viper. This includes mapping back error messages, indicating the failing expression and corresponding line number from the original Silver program.
- Evaluate limitations and performance of the new backend. The limitations of the backend can be measured in the number of Viper testcases it is able to solve. When evaluating performance, it is not only interesting to see the difference in timings between the backends, but also the difference between the solution approaches without the translation overhead.

3 Extensions

The following list describes possible extensions to the core tasks. In the course of the thesis we will choose which of them to explore, it is beyond the scope of this thesis to explore all of them.

- Translate problems in GRASShopper to problems in Viper. The composition of the two translations then allows for an analysis of the differences of the original program and the program after the two translations.
- GRASShopper is able to produce counterexamples for programs that fail to verify. Mapping those counterexamples back into the Viper tool could simplify the debugging of such programs.
- For problems in Viper that are not supported by GRASShopper, split the problem into parts that are supported by GRASShopper and parts that are not supported. Use GRASShopper on the supported parts, and another backend tool on the remaining parts, then combine the results.
- Extend the mapping from Viper to GRASShopper, by allowing more complicated translations and rewriting, e.g. mapping recursive definitions to point-wise definitions.

References

[1] P. Müller, M. Schwerhoff, and A. J. Summers. Viper: A verification infrastructure for permission-based reasoning. In B. Jobstmann and K. R. M. Leino, editors, *Verification, Model Checking, and Abstract Interpretation (VMCAI)*, volume 9583 of *LNCS*, pages 41–62. Springer-Verlag, 2016.

[2] Ruzica Piskac, Thomas Wies, and Damien Zufferey. *Automating separation logic with trees and data*, volume 8559 LNCS of *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, pages 711–728. Springer Verlag, 2014.

[3] Ruzica Piskac, Thomas Wies, and Damien Zufferey. *Automating separation logic using SMT*, volume 8044 LNCS of *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, pages 773–789. 2013.

[4] Ruzica Piskac, Thomas Wies, and Damien Zufferey. *GRASShopper: Complete heap verification with mixed specifications*, volume 8413 LNCS of *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, pages 124–139. Springer Verlag, 2014.