A FORMAL SEMANTICS FOR VIPER

MASTER'S THESIS PROJECT DESCRIPTION

JUNE 2016, ETH ZURICH, SWITZERLAND

Student: Cyrill Gössi

cgoessi@student.ethz.ch

Supervisor: Dr. Alexander J. Summers

alexander.summers@inf.ethz.ch

1. BACKGROUND

The Verification Infrastructure for Permission-based Reasoning (Viper) [1] is a suite of tools developed by the Chair of Programming Methodology research group at ETH Zurich. Viper includes an intermediate language (IL) which is based on a flexible permission model allowing for simple encodings of permission-based reasoning techniques. The infrastructure is targeted by multiple front-end tools such as Chalice [3]. Verification problems are encoded into Viper IL and the infrastructure ultimately verifies the problem by means of symbolic execution or usage of verification condition generation. The result of the verification is then propagated back to the front-end.

As of today, the Viper IL lacks a formal semantics. Hence, encodings of front-end languages into Viper IL cannot formally be reasoned about and ultimately the result of the verification cannot be rigorously argued to be sound.

The aim of this master's thesis is the development of a formal semantics for the Viper IL and thereby laying a foundation for a more rigorous treatment of the verification process.

2. MAIN TASKS

Various features of the Viper IL have already been extensively studied and even have been given a formal semantics (with the treatment of recursive predicates by Summers and Drossopoulou [2] as an example) but no semantics for the IL as a whole exists. Developing such a semantics for the whole IL is the main task of this thesis. In order for the project to be considered successful, the following points were defined to be the core tasks:

- Development of a formal semantics for the following set of Viper features: permissions including fractional-permission-related constructs and quantified permissions, magic wands, pure expressions (including functions), predicates and control flow (including methods and loops).
- As the Viper semantics should be such that encodings of front-end-languages into Viper can, with relative ease, be proven sound, the following proof-of-concept has to be developed: first, a set of core features of Chalice has to be defined and a semantics has to be developed for it. This set then has to be encoded into Viper and the encoding has to be proven sound.

3. EXTENSIONS

The core tasks defined above allow for the following set of natural extensions.

- Extension of the formal semantics for Viper to also include forperm quantification, domains, paired assertions and labelled old-expressions as well as finding a model of triggers for quantifiers.
- Extension of the formal semantics for Chalice to include all of Chalice.
- Mechanization of the formal semantics. Possible tools of choice are the proof-assistants Coq [4] and Isabelle/HOL [5].
- Mechanization of the encoding of Chalice into Viper.

4. 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, VMCAI, volume 9583 of LNCS, pages 4162. Springer-Verlag, 2016.

[2] A. Summers and S. Drossopoulou. A formal semantics for isorecursive and equirecursive state abstractions. In ECOOP, volume 7920 of LNCS, pages 129-153. Springer, 2013.

[3] K. R. M. Leino, P. Müller, and J. Smans. Verification of concurrent programs with Chalice. In A. Aldini, G. Barthe, and R. Gorrieri, editors, Foundations of Security Analysis and Design V, volume 5705 of LNCS, pages 195222. Springer, 2009.

[4] G. Huet, G. Kahn, C. Paulin-Mohring, The Coq Proof Assistant: a tutorial, Technical Report 204, INRIA-Rocquencourt, 1997.

[5] T. Nipkow, L. C. Paulson, and M. Wenzel. Isabelle/HOL A Proof Assistant for Higher-Order Logic, volume 2283 of LNCS. Springer, 2002.