Automated Support for Mathematical Datatypes in the Viper Tool Infrastructure

Peter Müller
Chair of Programming Methodology
ETH Zürich
peter.mueller@inf.ethz.ch

January 30, 2015

Viper ("Verification Infrastructure for PErmission- based Reasoning") is a collection of tools developed at ETH Zurich [5], oriented around a new intermediate verification language called Silver. Silver provides support for permission-based reasoning (la Separation Logic [9]) using the logic of implicit dynamic frames [11, 8, 10]. This infrastructure is particularly well suited for supporting verification techniques based on separation logic and other permission logics. Front-end tools for source programming languages can be encoded to verification problems in Silver, for which there are two verifiers available as back-ends: Silicon (based on symbolic execution) and Carbon (based on verification condition generation via Boogie [6]).

The intermediate language Silver supports a few primitive types and other types can be encoded by front-end tools. It also provides support for polymorphic sequence and set datatypes. The sequence and set types are supported by axiomatizations based on those used in the Dafny verifier [7]. The axiomatisations have been developed using theories available in the SMT solver (such as uninterpreted functions, integer arithmetic, etc.) and consist of universally-quantified formulas controlled by triggers to guide the E-matching procedure in Z3 [2]. The initial task of the internship will be to understand the existing encodings in detail. In the context of the Viper tools, these axiomatisations perform reasonably well, but in some cases there are performance issues when handling complicated examples (particularly those with heavy use of set and sequence constructs). Understanding the performance of Z3 with respect to different axiomatisations requires a lot of experimentation. The VCC project [1] provides one tool which can be used for debugging and profiling Z3′s behaviour in these cases. Finding a solution to this will give good and predicatable support for set and sequences data types encoding which would also be relevant for the Dafny program verifier and others based on Boogie/Z3′s technology; potential improvements could greatly benefit a variety of existing tools and verifiers.

In the internship we propose to investigate the support for these mathematical types in the following ways:

1. Analysing the existing encodings for mathematical datatypes of set and sequences; trying examples and understanding the worst case behaviour for some existing encodings.
2. Investigating alternative approaches such as using different theorem provers or custom decision procedures.

3. Based on the above investigations, find a suitable solution, and document the insights which lead to its development. We may consider formalising any completeness/termination guarantees (for triggering-based axiomatisations, some existing papers suggest techniques for defining these properties formally) [4, 3].

4. Further work may include specifying involved examples and exploiting the resulting solutions, and/or transferring the ideas learned to support for other features of the verification infrastructure.

References

[1] Cohen, E., Dahlweid, M., Hillebrand, M., Leinenbach, D., Moskal, M., Santen, T., Schulte, W., and Tobies, S. VCC: A practical system for verifying concurrent C.

[2] de Moura, L., and Bjrner, N. Z3: An efficient SMT solver. In *Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29-April 6, 2008. Proceedings* (2008), vol. 4963 of *Lecture Notes in Computer Science*, Springer, pp. 337–340.

[3] Dross, C., Conchon, S., Kanig, J., and Paskevich, A. Reasoning with Triggers. In *SMT 2012* (2013), P. Fontaine and A. Goel, Eds., vol. 20 of *EPiC Series*, EasyChair, pp. 22–31.

[4] Ge, Y., and Moura, L. Complete Instantiation for Quantified Formulas in Satisfiabiliby Modulo Theories. In *Proceedings of the 21st International Conference on Computer Aided Verification* (Berlin, Heidelberg, 2009), CAV '09, Springer-Verlag, pp. 306–320.

[5] Juhasz, U., Kassios, I. T., Müller, P., Novacek, M., Schwerhoff, M., and Summers, A. J. Viper: A Verification Infrastructure for Permission-Based Reasoning. Tech. rep., ETH Zurich, 2014.

[6] Leino, K. R. M. This is Boogie 2. Tech. rep., June 2008.

[7] Leino, K. R. M. Dafny: An Automatic Program Verifier for Functional Correctness. In *Logic for Programming, Artificial Intelligence, and Reasoning - 16th International Conference, LPAR-16, Dakar, Senegal, April 25-May 1, 2010, Revised Selected Papers* (2010), pp. 348–370.

[8] Parkinson, M. J., and Summers, A. J. The Relationship Between Separation Logic and Implicit Dynamic Frames. *Logical Methods in Computer Science 8*, 3 (2012).

[9] Reynolds, J. C. Separation logic: A Logic for Shared Mutable Data Structures. In *Proceedings of the 17th Annual IEEE Symposium on Logic in Computer Science* (Washington, DC, USA, 2002), LICS '02, IEEE Computer Society, pp. 55–74.

[10] Smans, J. *Specification and Automatic Verification of Frame Properties for Java-like Programs (Specificatie en automatische verificatie van frame eigenschappen voor Java-achtige programma's).* PhD thesis, Informatics Section, Department of Computer Science,

Faculty of Engineering Science, May 2009. Piessens, Frank and Clarke, Dave (supervisors).

[11] Smans, J., Jacobs, B., and Piessens, F. Implicit Dynamic Frames: Combining dynamic frames and separation logic. In *ECOOP 2009 - Object-oriented Programming, 23rd European Conference, Genova, Italy, July 6-10, 2009, Proceedings, European Conference on Object-oriented Programming (ECOOP), Genova, 6-10 July 2009* (July 2009), S. Drossopoulou, Ed., Springer-Verlag, pp. 148–172.