Verifying weak memory programs in the Viper ecosystem

Anouk Paradis

Supervisors :
Gaurav Parthasarathy
Alexander J. Summers

April 15, 2019

1 Introduction

When reasoning about concurrent programs, the most intuitive setting is sequential consistency [2], where we consider the program to behave as if each of the threads' executions were simply interleaved. However, due to the many optimizations performed by modern compilers and hardware, such a model is too simplistic to accurately represent the actual executions of the program. We can hence only assume a weak memory model, which allows many more possible executions of the accesses to shared memory locations. However, these new possibilities of execution make reasoning about programs more difficult, and at times counterintuitive. Writing correct programs is hence extremely challenging in such a setting.

FSL++ [1] is a program logic developed to reason about concurrent programs in C11, which are governed by the weak memory model defined in the C11 standard. This program logic allows for compositional proofs, through preconditions and postconditions for functions for instance. However, such proofs originally had to be encoded using the Coq interactive theorem prover, thus requiring a lot of ad hoc work for each new program considered.

To try to alleviate this proof burden, part of this program logic has been encoded into Viper [3] by Summers and Müller [5]. However it did not provide any encoding for the notion of custom *ghost state*, an important feature used to model information that cannot be deduced solely from the program state. This *ghost state* is defined quite generally in FSL++, and needs to be specialized for each example, which makes their encoding and their use quite challenging. Parthasarathy et al. [4] provided such a specialization and showed that it was sufficient to prove the correctness of a variety of example programs. Building on this work, Wiesmann et al. [6] was then able to define a specification syntax, and its encoding in Viper, that allowed easier use of this specialization, without having to deal with lower level details. Wiesmann et al. [6] also implemented a C++ frontend for Viper using this specialization to provide more automation for proofs.

2 Core Tasks

The main goal of this project is to evaluate on some real world examples the infrastructure developed around Viper for weak memory programs. This infrastructure builds on different levels of abstraction. The specification syntax defined by Wiesmann et al. [6] only encodes a simplification of the ghost state defined by Parthasarathy et al. [4] and hence does not allow to fully use the possibilities of this ghost state. This ghost state is already a specialization of the more generic ghost state definition by FSL++. Furthermore, the C++ frontend implemented by Wiesmann et al. [6] does not support all features. However, the hope is that most of these restrictions are not too problematic in practice, and that they are expressive enough to allow for the proof of a variety of examples.

This goal can be further divided into the following subgoals:

- Identifying interesting examples from real-world libraries, and understanding the underlying synchronization mechanisms;
- Proving those examples using each level of abstraction, from the ghost state specialization defined by Parthasarathy et al. [4] to the C++ front-end and the encoding into Viper it provides defined by Wiesmann et al. [6];
- Identifying the limitations of the different levels of abstraction in terms of expressiveness, ease of use and efficiency.

3 Further Work

Some extensions of this project could be:

- Extending the current infrastructure to overcome some of the identified limitations;
- Proving more challenging examples, involving more interleaved functions and synchronization mechanisms;
- Formalizing and proving the soundness of the different levels of abstraction.

References

[1] M. Doko and V. Vafeiadis. Tackling real-life relaxed concurrency with FSL++. In *European Symposium on Programming (ESOP)*, pages 448–475. Springer, 2017.

[2] L. Lamport. How to make a correct multiprocess program execute correctly on a multiprocessor. *IEEE Trans. Comput.*, 46(7):779–782, July 1997.

[3] 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.

[4] G. Parthasarathy. Applying and extending the weak-memory logic FSL++. Research in Computer Science Project, ETH Zürich, 2017.

[5] A. J. Summers and P. Müller. Automating deductive verification for weak-memory programs. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, LNCS, pages 190–209. Springer-Verlag, 2018.

[6] P. Wiesmann. Deductive verification of real-world C++ weak-memory programs. Master's thesis, ETH Zürich, 2019.