A Prototype Verifier for Weak Memory Reasoning

Background

The Viper verification infrastructure, developed at ETH Zurich, consists of several front-ends and back-ends centered around the intermediate verification language Viper. It can be used to tackle a wide variety of concurrent and heap-based verification problems and is especially suited to encode permission-based reasoning techniques, as Silver includes assertions to denote permissions to access heap locations.

Relaxed Separation logic[1] (RSL) is a program logic used for reasoning about concurrent programs under the C11 memory model. It allows for modeling ownership of memory, including ownership transfer between threads upon certain atomic memory accesses. This is helpful to reason about very fine-grained concurrent access patterns, as used in lockfree programming.
There is currently no tool that automates proofs in relaxed separation logic and therefore analyzing examples of programs that can be proven requires a lot of manual effort.

The goal of this project is to build a front-end for Viper that translates annotated code from a source language into Viper's intermediate language. The translation is done in such a way that verifying the resulting program corresponds to constructing a proof about the original program in relaxed separation logic. The verification can then be done by one of the existing back-end verifiers for Viper.
The encoding of RSL concepts into Viper is already being developed by the Programming Methodology Group at ETH Zurich, but there is no automated front-end available yet.

Goals

- Develop the language of the code that the front-end takes. This might be a subset of C++ or a toy language that only contains the important operations for atomic and non-atomic memory accesses.
- Determine what annotations are needed to supply the essential parts of the proof that are necessary to get a verifiable resulting Viper program.
- Implement the actual front-end.
- Include support of additional concepts for memory fences introduced in Fenced Separation Logic[2].
- Evaluate the front-end on various examples, taken both from papers about the logic and open source libraries like, for example, folly[3], boost[4] and glib[5].

Possible Extensions

- Analyze some examples that cannot be verified using relaxed separation logic as is and work out some changes to the logic, map them to Viper again and test their behaviour on various examples.
- Include a proof about the equivalence of our encoding and relaxed separation logic proofs. There is a formal semantics of RSL available from the paper that introduces it and there is an ongoing master's thesis developing a formal semantics of Viper.
- Reduce the amount of annotations necessary by inferring some of them, using static analysis methods.
- Compare the new front-end and Viper encoding with a formalization in Coq, that was done by the authors of the original paper, both in terms of the length of our encoding, and how much needs to be written to verify examples.

---

1 https://mpi-sws.org/~viktor/papers/oopsla2013-rsl.pdf
2 https://www.mpi-sws.org/~viktor/papers/vmcai2016-fsl.pdf
3 https://github.com/facebook/folly
4 http://www.boost.org/
5 https://developer.gnome.org/glib/