Formalizing higher-order separation logic in Lean

Master's Thesis Proposal

Thesis Supervisor: Max Vistrup

Abstract

One of the focuses of our group is the theoretical side of formal verification. We are interested in the *mathematics* and *logic* of reasoning about program executions. In order to know that our proofs are correct, we have a computer check them for us. Specifically, we currently use the Coq proof assistant: we write our definitions and proofs in Coq and have Coq ensure that our proofs contain no errors or oversights.

However, certain things that are easy to do when doing proofs "on paper" are frustrating to do in Coq. In the hope of addressing some of these issues, recently a competitor to Coq has emerged. Enter Lean. Lean has been popular among mathematicians who have used it to formalize certain research-level mathematics results, amounting to very impressive and successful strides in computer-checked mathematics. As of now, on the other hand, the applications of Lean in computer science have been explored less. Advocates of Lean point out some technical advantages over Coq, such as a better type-class system that can be used, for example, to define hierarchies of algebraic structures.

We would like to understand the advantages and disadvantages of Lean for our field of research (programming language theory). A lot of our work revolves around Iris, a Coq library implementing a state-of-the-art higher-order separation logic. Iris can be used to reason about programs involving concurrency, higher-order state (functions stored on the heap), and a number of other advanced features found in modern programming languages.

This project will port (a small fragment of) Iris to Lean in order to have a closer look at how Lean compares to Coq.

Goals

This project involves porting (a small fragment of) the model of Iris to Lean.

A part of Iris's front-end, the Iris proof mode, has already been ported to Lean. The project will use this front-end, instantiating the appropriate type-classes.

Over the course of the project, the student will have to become familiar with Lean and its tactics language. They will learn how to write formal proofs in Lean.

Furthermore, the student shall learn about (at least parts of) the model of Iris, including its recursive domain equation. One of the main goals of the project is to state this equation and construct a solution in Lean, as we already do in Coq. This constitutes the core of the model of Iris.

Finally, the student has to, in collaboration with the advisor, reflect on what turned out to be difficult in establishing the model of Iris in Lean, especially in comparison to its Coq formalization. A large portion of what makes this project interesting is that it can serve to evaluate Lean as a tool for future use in programming language theory.

Optional Goals

Iris is a huge library with many constructions and abstractions. We only expect to port a tiny fragment of Iris to Lean, but in principle one could be extremely ambitious and port even more. Then, one could port certain verification proofs such as those for concurrent data structures like Treiber stacks.

Required and Optional Skills

We acknowledge that this is a research-level project on the more difficult side. It is thus best suited for a strong and highly motivated student. In return, it is also a rewarding and potentially impactful project in the sense that there is a lot of interest and curiosity for the use of Lean among researchers in formal verification and programming language theory.

- Experience with proof assistants is required. This experience doesn't have to be with Lean necessarily; it could also be with other proof assistants such as Coq. For example, it would be helpful if the student had taken a course such as Formal Foundations of Programming Languages (FFPL).
- Some basic knowledge of type theory, category theory, or separation logic is advantageous but not required.

Further Reading

Lean

Resources on learning and understanding Lean can be found on the Lean website.

Iris

A very theoretical overview of the Iris model is found in Iris from the ground up.

A more example-oriented introduction to Iris is contained in the lectures notes from Aarhus Universitet.

Iris can be understood on different levels of abstraction, so don't worry if this looks like a lot—only a small part of it is actually required for this project.