M2 internship: Bridging the gap between a verified library and a proof assistant

Guillaume Melquiond

Inria, Université Paris-Saclay

1 Context

One of the long-term goals of the ERC project Fresco[1] is to turn the Coq proof assistant into a competitive tool for doing verified computer algebra. In particular, this requires the ability to implement and verify well-known libraries such as GMP or BLAS/LAPACK. A significant milestone was the design of Capla, a safe low-level imperative language suitable for implementing such algorithms, as well as the development of a formally verified compiler for this language [4].

It is now possible to write a library using Capla, to compile it to machine code, to verify its correctness using Coq, and to invoke its functions from C code. But to turn the Coq proof assistant into a usable computer algebra system, one should also be able to execute such Capla functions from a Coq script. Moreover, one should be able to use the resulting values inside a Coq proof, as if they had been produced by pure Coq functions.

This raises several questions, both theoretical and practical. First, Capla functions are partial, while Coq's consistency fundamentally rely on termination. So, how to reconcile both worlds? Second, the runtime of Coq and the runtime of Capla encode data structures differently, *e.g.*, OCaml-like lists for one and C-like arrays for the other, not counting that Coq allows open terms. Again, how to reconcile both worlds? Third, how to give Capla functions a suitable semantics so that Coq proofs can make use of the computed values?

2 Objectives

This 6-month internship at the Master 2 level aims at answering these questions. Here is a tentative work program:

- survey the literature for projects that aim at bridging the gap between Coq and compiled code [2, 1];
- understand how the evaluation strategies of Coq behave, and how data structures are represented in Capla and Coq;
- devise a representation of Capla values (*i.e.*, machine integers, multi-dimensional arrays) as Coq terms, and conversely;
- devise a way to express Capla computations as Coq terms (*e.g.*, functions, relations, monads, etc), as well as a way to associate some useful semantics to these terms [3];
- modify Coq so that it can effectively perform these computations, *i.e*, convert the input Coq values to Capla, execute the corresponding compiled code, and convert the result back.

This internship should result in a small prototype that makes it possible to execute a simple Capla function inside a Coq proof (*e.g.*, the addition of long integers) and to make use of its result as if it had been computed by a pure Coq function inside the formal system.

[1] `https://fresco.gitlabpages.inria.fr/`

3 Location

The internship will take place at the Formal Methods Laboratory[2] of Université Paris-Saclay, in the Inria team Toccata, which is dedicated to writing tools for deductive program verification.[3]

4 Prerequisites

Knowledge of the Coq proof assistant or of a closely-related formal system (*e.g.*, Lean), is highly recommended. Knowledge about the semantics of programming languages, as well as the issue of foreign function interfaces (especially between C and OCaml), is also recommended.

References

[1] Abhishek Anand, Andrew W. Appel, Greg Morrisett, Zoe Paraskevopoulou, Randy Pollack, Olivier Savary Bélanger, Matthieu Sozeau, and Matthew Z. Weaver. CertiCoq: A verified compiler for Coq. In *3rd International Workshop on Coq for Programming Languages*, 2017. URL: `https://www.cs.princeton.edu/~appel/papers/certicoq-coqpl.pdf`.

[2] Benjamin Grégoire and Xavier Leroy. A compiled implementation of strong reduction. In *7th ACM SIGPLAN International Conference on Functional Programming*, pages 235–246, Pittsburgh, United States, October 2002. `doi:10.1145/581478.581501`.

[3] Assia Mahboubi and Guillaume Melquiond. Manifest termination. In Eduardo Hermo Reyes and Alicia Villanueva, editors, *29th International Conference on Types for Proofs and Programs*, pages 172–174, Valencia, Spain, June 2023. URL: `https://hal.science/hal-04172297`.

[4] Josué Moreau and Guillaume Melquiond. A safe low-level language for computer algebra and its formally verified compiler. In Brigitte Pientka, editor, *29th ACM SIGPLAN International Conference on Functional Programming*, volume 8 of *Proceedings of the ACM on Programming Languages*, pages 121–146, Milan, Italy, September 2024. `doi:10.1145/3674629`.

[2]`https://lmf.cnrs.fr/`

[3]`https://toccata.gitlabpages.inria.fr/toccata/`