Programs from the Book
Bit-level certified programming in Coq

Pierre-Évariste Dagand Julia Lawall
Whisper team

Abstract

This internship offers to explore the realm of certified bit-level programming in Coq, following the trail left by Warren in "Hacker's Delights". We propose an exercise in developing a certified library, serving as a foundation for later developments in the field of System programming. This internship will be help at LIP6 (Université Paris 6) in the Whisper team (Inria).

The Coq proof assistant has gained formidable traction as a framework for certified programming. Witness of its maturity is the Compcert [Leroy, 2009] certified C compiler. One could perhaps argue that certifying a compiler is not taking Coq far away from its comfort zone: it could be seen as, *in fine*, yet another exercise in symbolic manipulation, exercise at which it naturally excels.

Recent works dispel such concern: for instance, Kennedy et al. [2013] have shown that Coq can also be turned into an effective macro-assembler for the x86 architecture. Besides a verified macro-assembler, this project offers a program logic that can be used to prove the correctness of low-level machine code.

However, to work as such a low-level, one relies on a formalization of bit-level operations, including their formal specification and correctness proofs. For these operations to be efficiently executable, special care must be given to their extraction: where the Coq library handles a byte as a 8-tuple of booleans, the extraction to, say, OCaml should ideally translate those types to characters.

This internship proposal is inspired by the work of Gonthier [2008]. In tackling the formalization of the Four Colors theorem, the Mathematical Components project has made a lasting impact on the Coq ecosystem: it contributed a new tactic language [Mathematical Components, 2014b], a mathematical library [Mathematical Components, 2014a], and a methodology for proving in the large [Garillot et al., 2009]. To the seasoned hacker, Warren [2012]'s *opus magnus* "Hacker's delight" exerts a similar fascination[1] as the Four Colors theorem.

We propose to implement, specify and prove the correctness of some of the algorithms described by Warren [2012]. To paraphrase Paul Erdös, these "Programs from the Book" will provide us with concrete and real-life examples of bit-level computations. In the process, we shall aggregate a library of certified bit-level programs, extending the library `x86proved` developed by Kennedy and Benton [2015]. Bearing evaluation (in Coq) and compilation (through OCaml) in mind, we also wish to support efficient extraction to OCaml.

[1] One will for example find some of its algorithms mentioned in no less than OCaml's `asmcomp/cmmgen.ml` source file.

This internship will consist in:

- Implementing algorithms from the Book, thus contributing to a certified library;
- Specifying their correctness in Coq, thus advancing the art of programming in Coq;
- Proving the algorithms' correctness, thus exercising – and improving – the program logic.

Aside from the proof engineering effort, this internship's contribution will be twofold. On one hand, it aims at providing a general-purpose library of bit-level operations, supporting extraction to OCaml. One the other hand, it will explore methodological and practical issues related to the formalisation of low-level programs. As a whole, it shall bring Coq closer to being a full-fledged System programming language.

The team: This internship will be held at LIP6 (UPMC, Jussieu campus) in the Whisper team (Inria) headed by Gilles Muller. It will be supervised by Julia Lawall and co-supervised by Pierre-Évariste Dagand. The `x86proved` project is developed at Microsoft Research Cambridge (UK) by Andrew Kennedy and Nick Benton, with whom we have strong ties. We would actively support a strong student later applying for a Microsoft internship.

Student's profile: We are looking for a student at ease with low-level programming (either in C, or in assembly) and interested in rationalizing bit-level reasoning. Acquaintance with an interactive theorem prover (Coq, or Isabelle) is recommended. Nonetheless, a motivated student with a strong background in functional programming (OCaml, or Haskell) could certainly learn to use Coq along the way [Pierce et al., 2014, Chlipala, 2011].

References

A. Chlipala. *Certified Programming with Dependent Types*. MIT Press, 2011. URL `http://adam.chlipala.net/cpdt/`.

F. Garillot, G. Gonthier, A. Mahboubi, and L. Rideau. Packaging Mathematical Structures. In *Theorem Proving in Higher Order Logics*, volume 5674 of *Lecture Notes in Computer Science*, 2009.

G. Gonthier. Formal proof the four-color theorem. *Notices of the AMS*, 55(11):1382–1393, Dec. 2008.

A. Kennedy and N. Benton. x86proved, 2015. URL `http://x86proved.codeplex.com/`.

A. Kennedy, N. Benton, J. B. Jensen, and P.-E. Dagand. Coq: The world's best macro assembler? In *Proceedings of the 15th Symposium on Principles and Practice of Declarative Programming*, PPDP '13, pages 13–24. ACM, 2013.

X. Leroy. Formal verification of a realistic compiler. *Commun. ACM*, 52(7):107–115, 2009. doi:10.1145/1538788.1538814.

Mathematical Components. Mathcomp library, 2014a. URL `http://ssr.msr-inria.inria.fr/doc/mathcomp-1.5/`.

Mathematical Components. Ssreflect library, 2014b. URL `http://ssr.msr-inria.inria.fr/doc/ssreflect-1.5/`.

B. C. Pierce, C. Casinghino, M. Gaboardi, M. Greenberg, C. Hriţcu, V. Sjoberg, and B. Yorgey. *Software Foundations*. Electronic textbook, 2014. URL `http://www.cis.upenn.edu/~bcpierce/sf`.

H. S. Warren. *Hacker's Delight*. Addison-Wesley Professional, 2012. ISBN 0321842685.