Native floats in Coq

Master 2 internship proposal

Team ACADIE, Lab. IRIT, Université Toulouse 3

November 2017

Practical information The internship will take place in the *Institut de Recherche en Informatique de Toulouse* (IRIT) on the campus of *Université Paul Sabatier* (UPS). It will be supervised by Érik MARTIN-DOREL (erik.martin-dorel@irit.fr) and Pierre ROUX (pierre.roux@onera.fr).

The internship can be funded. At the end of the internship, some PhD position can be opened.

Context Formal proof assistants are complex pieces of software that allow one to encode mathematical theories and algorithms in a formal language, formulate properties and theorems on these algorithms, develop proofs for these properties and mechanically check these formalized proofs with a very high level of guarantee.

Thanks to its very expressive underlying logic, the Coq proof assistant has builtin support for computation. It offers several reduction tactics such as `compute`, `vm_compute` [3] and `native_compute` [1], which can take advantage of compilation to bytecode or to native OCaml code, in order to increase the performances of the considered computation, for the price of a slightly bigger trusted computing base.

Furthermore, works have been accomplished to extend Coq with arithmetic libraries (`BigN`, `BigZ`, `BigQ`) [4] that take advantage of machine integers' efficiency.

Coupled with Coq's feature of proof by reflection, these reduction tactics and these efficient data-types have been the keys of many works in formally certified symbolic or numerical computation.

An example of such works is given by the ValidSDP[1] library: a reflexive tactic to automatically and formally prove inequalities on multivariate polynomials involving real-valued variables and rational constants [7].

The formalization of ValidSDP relies on a emulated implementation of double-precision floating-point arithmetic, using the multiple-precision floating-point algorithms from the CoqInterval library [8, 6] and `BigZ` integers.

This layer of the ValidSDP formalization constitutes the main bottleneck of the overall performances of the tactic (roughly speaking, there is a slowdown factor of three orders of magnitude for the arithmetic operations).

We thus would like to investigate the use of native floating-point numbers in Coq. Beyond the impact on the ValidSDP library, this work would improve on the current state-of-the-art of certified numerical computation using Coq, and could be the incentive to make approaches in rigorous numerical computation [9] amenable to formal proof.

[1] https://sourcesup.renater.fr/validsdp/

Expected contribution This internship aims at the following objectives:

1. develop a Coq theory of floating-point arithmetic—with rounding to nearest—that is compliant with the IEEE-754 standard [5] and computationally efficient;
2. extend this theory to handle "directed roundings" as well;
3. evaluate the performances of the approach on a collection of benchmarks, including those bundled with the ValidSDP library.

The formal developments could rely on the Flocq library [2] for floating-point arithmetic in Coq. And if time permits, the internship could lead to a collaboration with Guillaume Melquiond to investigate the integration of the developed prototype with respect to the CoqInterval library.

Prerequisites The candidate should have a solid background in functional programming and mathematical logic, have strong programming skills in OCaml, and ideally some experience using Coq or another formal proof assistant.

Knowledge of the Coq libraries cited in this subject (Flocq, CoqInterval, ValidSDP) is not required.

A taste for computer arithmetic or numerical computation is preferable.

References

[1] Mathieu Boespflug, Maxime Dénès, and Benjamin Grégoire. Full reduction at full throttle. In Jean-Pierre Jouannaud and Zhong Shao, editors, *Certified Programs and Proofs - First International Conference, CPP 2011, Kenting, Taiwan, December 7-9, 2011. Proceedings*, volume 7086 of *Lecture Notes in Computer Science*, pages 362–377. Springer, 2011. doi:10.1007/978-3-642-25379-9_26.

[2] Sylvie Boldo and Guillaume Melquiond. Flocq: A Unified Library for Proving Floating-Point Algorithms in Coq. In Elisardo Antelo, David Hough, and Paolo Ienne, editors, *IEEE Symposium on Computer Arithmetic*, pages 243–252. IEEE Computer Society, 2011. doi:10.1109/ARITH.2011.40.

[3] Benjamin Grégoire and Xavier Leroy. A compiled implementation of strong reduction. In Mitchell Wand and Simon L. Peyton Jones, editors, *Proceedings of the Seventh ACM SIGPLAN International Conference on Functional Programming (ICFP '02), Pittsburgh, Pennsylvania, USA, October 4-6, 2002.*, pages 235–246. ACM, 2002. doi:10.1145/581478.581501.

[4] Benjamin Grégoire and Laurent Théry. A Purely Functional Library for Modular Arithmetic and Its Application to Certifying Large Prime Numbers. In Ulrich Furbach and Natarajan Shankar, editors, *IJCAR*, volume 4130 of *Lecture Notes in Computer Science*, pages 423–437. Springer, 2006. doi:10.1007/11814771_36.

[5] IEEE Computer Society. IEEE Standard for Floating-Point Arithmetic. *IEEE Std 754–2008*, August 2008. doi:10.1109/IEEESTD.2008.4610935.

[6] Érik Martin-Dorel and Guillaume Melquiond. Proving Tight Bounds on Univariate Expressions with Elementary Functions in Coq. *Journal of Automated Reasoning*, pages 1–31, 2015. URL: https://hal.inria.fr/hal-01086460v2, doi:10.1007/s10817-015-9350-4.

[7] Érik Martin-Dorel and Pierre Roux. A reflexive tactic for polynomial positivity using numerical solvers and floating-point computations. In Yves Bertot and Viktor Vafeiadis, editors, *Proceedings of the 6th ACM SIGPLAN Conference on Certified Programs and Proofs, CPP 2017, Paris, France, January 16-17, 2017*, pages 90–99. ACM, 2017. doi:10.1145/3018610.3018622.

[8] Guillaume Melquiond. Floating-point arithmetic in the Coq system. *Inf. Comput.*, 216:14–23, 2012. doi:10.1016/j.ic.2011.09.005.

[9] Siegfried Rump. Verification methods: Rigorous results using floating-point arithmetic. *Acta Numerica*, 19:287–449, May 2010. URL: http://journals.cambridge.org/article_S096249291000005X, doi:10.1017/S096249291000005X.