Research Internship 2021-2022 Topic
Formally Verified Compilation of C and Rust

Supervisors: Sylvain Boulmé and David Monniaux at Verimag[*]
in collaboration with their Phd students

mailto:Sylvain.Boulme@univ-grenoble-alpes.fr,David.Monniaux@univ-grenoble-alpes.fr

CompCert[1] is a compiler for the C programming language for the assembly languages of several processor architectures. In contrast to compilers such as Visual C++, GCC, or LLVM, its compilation phases are proved mathematically correct, and thus the compiled program always matches the source program: the formal correctness of CompCert states that if the compiler succeeds to produce an executable, then the *observable behaviors* of the executable are also *observable* on the source program [1, 2]. Other compilers may contain bugs that in some cases result in incorrect code being generated. The possibility of compilation bug cannot be tolerated in certain applications with high safety requirements, and then costly solutions such as disabling all optimizations are used to get assembly code that is close to the source. In contrast, CompCert, despite not optimizing as well as gcc -O3 or clang -O3, allows using optimizations safely [3, 4]. CompCert is itself written as a combination of the Coq interactive theorem prover[2] and the OCaml programming language[3].

CompCert-KVX[4] is a variant of CompCert developed at Verimag which provides the first formally verified efficient optimizations for superscalar and VLIW processors [5, 6, 7, 8, 9]. We propose several internship topics around formally verified compilation (to be discussed according to the interests of the students), through modifications to CompCert-KVX or in independent tools. Here are examples of hot topics for us:

- Extend the superblock scheduling of [9] to enable exchange of branching and basic instructions modulo code duplication. We already have an experimental prototype of this extension. We would like to redesign it from scratch, in order to have both a simpler and more efficient implementation of this formally verified transformation. Experiment mostly in OCaml, with experimental evaluation of performance, and with maybe a bit of Coq.

- Currently, CompCert uses an unsound interface for embedding OCaml foreign code within the Coq programming language. A solution has been proposed in [10] and marginally experimented in CompCert-KVX. We would like to evaluate the feasibility of fixing the whole CompCert in this way. Experiment mostly in Coq with a bit of OCaml.

[*] http://www-verimag.imag.fr/
[1] https://compcert.org/
[2] https://coq.inria.fr/
[3] https://ocaml.org/
[4] https://gricad-gitlab.univ-grenoble-alpes.fr/certicompil/compcert-kvx

- We are currently investigating a Rust[5] frontend for CompCert. In a first step, we would like to implement an unverified prototype compiler from a small subset of MIR[6] to C, or more exactly Clight a subset of the C language internal to CompCert. Experiment mostly in Rust with a bit of OCaml, and maybe Coq.

- Certain arithmetic operations may be implemented natively or by composition of elementary operations: we would like to implement and prove such expansions in a pass before the superblock scheduling.

- CompCert for secure applications: see details in https://www-verimag.imag.fr/IMG/pdf/sujet-compcert.pdf

References

[1] X. Leroy, "Formal verification of a realistic compiler," *Communications of the ACM*, vol. 52, no. 7, 2009. HAL: inria-00415861.

[2] ——, "A formally verified compiler back-end," *Journal of Automated Reasoning*, vol. 43, no. 4, pp. 363–446, 2009. [Online]. Available: http://xavierleroy.org/publi/compcert-backend.pdf.

[3] R. Bedin França, S. Blazy, D. Favre-Felix, X. Leroy, M. Pantel, and J. Souyris, "Formally verified optimizing compilation in ACG-based flight control software," in *Embedded Real Time Software and Systems (ERTS 2012)*, 2012. [Online]. Available: http://hal.inria.fr/hal-00653367/.

[4] D. Kästner, U. Wünsche, J. Barrho, M. Schlickling, B. Schommer, M. Schmidt, C. Ferdinand, X. Leroy, and S. Blazy, "CompCert: Practical experience on integrating and qualifying a formally verified optimizing compiler," in *ERTS 2018: Embedded Real Time Software and Systems*, SEE, Jan. 2018. [Online]. Available: http://xavierleroy.org/publi/erts2018_compcert.pdf.

[5] C. Six, S. Boulmé, and D. Monniaux, "Certified and efficient instruction scheduling: Application to interlocked VLIW processors," *Proc. ACM Program. Lang.*, vol. 4, no. OOPSLA, 129:1–129:29, 2020. DOI: 10.1145/3428197. HAL: hal-02185883.

[6] D. Monniaux and C. Six, "Simple, light, yet formally verified, global common subexpression elimination and loop-invariant code motion," in *LCTES '21: 22nd ACM SIGPLAN/SIGBED International Conference on Languages, Compilers, and Tools for Embedded Systems, Virtual Event, Canada, 22 June, 2021*, J. Henkel and X. Liu, Eds., ACM, 2021, pp. 85–96. DOI: 10.1145/3461648.3463850. [Online]. Available: https://doi.org/10.1145/3461648.3463850.

[7] C. Six, "Optimized and formally-verified compilation for a VLIW processor," PhD thesis, Université Grenoble Alpes, France, Jul. 2021. [Online]. Available: https://hal.archives-ouvertes.fr/tel-03326923.

[5] https://www.rust-lang.org/

[6] https://rustc-dev-guide.rust-lang.org/mir/index.html

[8] L. Gourdin, “Formally verified postpass scheduling with peephole optimization for aarch64,” in *20èmes journées Approches Formelles dans l’Assistance au Développement de Logiciels, AFADL 2021*, Jun. 2021. [Online]. Available: https://www.lirmm.fr/afadl2021/papers/afadl2021_paper_9.pdf.

[9] C. Six, L. Gourdin, S. Boulmé, and D. Monniaux, “Verified Superblock Scheduling with Related Optimizations,” working paper or preprint, Apr. 2021, [Online]. Available: https://hal.archives-ouvertes.fr/hal-03200774.

[10] S. Boulmé, “Formally verified defensive programming (efficient Coq-verified computations from untrusted ML oracles),” See also http://www-verimag.imag.fr/ boulme/hdr.html, Habilitation à diriger des recherches, Université Grenoble-Alpes, Sep. 2021. [Online]. Available: https://hal.archives-ouvertes.fr/tel-03356701.