Machine-verified Code Generation for a Lustre Compiler

Timothy Bourke and Marc Pouzet

March–August 2016

This internship will contribute to the development of techniques, software, and proofs for a prototype synchronous language compiler in Coq.

As synchronous dataflow languages and their compilers are most often used in the development of safety-critical applications, like fly-by-wire controllers and power plant monitoring software, their correctness is paramount. Modelling and verifying compilers in a proof assistant like Coq[1] is a promising way to ensure that the semantics of source programs are precisely understood and completely preserved right down to the code that actually executes. Such certification in formal tools is complementary to industrial certifications, like the DO-178B guidelines for airborne systems.

Compilers of synchronous dataflow languages, like that used by the SCADE Suite tool[2] are constructed as a series of source-to-source transformations [4, 5, 3, 2]. The initial transformations successively reduce complex structures, like hierarchical automata, into simpler subsets of the language. Later transformations produce and optimize imperative C code from which executables are finally generated.

Earlier work [1] showed how to formally model and verify several of the important dataflow-to-dataflow transformations. Very recently, we have solved one of the key remaining difficulties: justifying the difficult step from dataflow to imperative code in Coq. Further work is required to expand these results, and, importantly, to connect them with the CompCert C compiler [6].[3] This connection not simply a matter of sending the stdout of one compiler into the stdin of the other: the underlying types and operations of the target language (C) must be imported into the semantic model of the source language (a subset of Lustre/SCADE). The imperative effects of the underlying machine, like division by zero and the side-effects of certain operations, must also be accounted for. This internship would be based on an existing prototype that is closely aligned with real compiler architectures.

Prerequisites A strong motivation to apply techniques from programming languages theory and formal logic to the design and improvement of real-world systems. The ability to discuss and work constructively with others.

- Solid background in the design of programming languages. (*required*)

[1] https://coq.inria.fr

[2] Developed and marketed by ANSYS/Esterel Technologies: http://www.esterel-technologies.com/products/scade-suite/

[3] http://compcert.inria.fr

- Programming experience in OCaml. (*preferred*)
- Experience in formal modelling and verification in a proof assistant (like Coq, HOL, PVS, or Isabelle). (*preferred*)
- Prior experience with synchronous programming. (*optional*)

Administrative information The internship will take place in the PARKAS team of INRIA and DI, ENS Paris: http://www.di.ens.fr/ParkasTeam.html

Advisors. Timothy Bourke and Marc Pouzet.

The intern will receive a stipend or salary according to his or her administrative status. The internship is partially funded by the *ASSUME* European project.

PhD thesis. The internship is an entry point for a motivated and talented student to a 3 year PhD thesis at the École normale supérieure in Paris. The PARKAS team has strong working links with development teams at Esterel Technologies and ongoing research and development at SAFRAN and Airbus.

References

[1] C. Auger. *Compilation certifiée de SCADE/LUSTRE*. PhD thesis, Univ. Paris Sud 11, Orsay, France, Apr. 2013.

[2] D. Biernacki, J.-L. Colaço, G. Hamon, and M. Pouzet. Clock-directed modular code generation for synchronous data-flow languages. In *Proc. 2008 ACM SIGPLAN Conf. on Languages, Compilers, and Tools for Embedded Systems (LCTES 2008)*, pages 121–130, Tucson, AZ, USA, June 2008. ACM, ACM Press.

[3] J.-L. Colaço, B. Pagano, and M. Pouzet. A conservative extension of synchronous data-flow with state machines. In W. Wolf, editor, *Proc. 5th ACM Int. Conf. on Embedded Software (EMSOFT 2005)*, pages 173–182, Jersey City, USA, Sept. 2005. ACM Press.

[4] J.-L. Colaço and M. Pouzet. Clocks as first class abstract types. In R. Alur and I. Lee, editors, *Proc. 3rd Int. Conf. on Embedded Software (EMSOFT 2003)*, volume 2855 of *Lecture Notes in Comp. Sci.*, pages 134–155, Philadelphia, Pennsylvania, USA, Oct. 2003. Springer.

[5] J.-L. Colaço and M. Pouzet. Type-based initialization analysis of a synchronous dataflow language. *Int. J. Software Tools for Technology Transfer*, 6(3):245–255, Aug. 2004.

[6] X. Leroy. Formal verification of a realistic compiler. *Comms. ACM*, 52(7):107–115, 2009.