Translation validation for synchronous data-flow equations in a Lustre compiler

Marc Pouzet  Francesco Zappa Nardelli
Marc.Pouzet@ens.fr  Francesco.Zappa_Nardelli@inria.fr

Location:  Département d’Informatique, École Normale Supérieure, 45 rue d’Ulm, 75230 Paris cedex 05.

Prerequisite:  interest in the design and implementation of programming languages, some experience with functional programming, familiarity with a proof assistant (Coq or Isabelle) is a plus.

Research Context  The synchronous data-flow language SCADE is the de-facto standard to implement reactive systems in critical domains, including nuclear energy, avionics, railways, and automotive (http://www.esterel-technologies.com). The SCADE compiler is written in OCaml and is “qualified” with the highest safety requirements (norm DO-178C, level A). This certification is instrumental for the SCADE success but imposes a major overhead to implement even simple modifications of the compiler.

Objective  In this internship we will explore some directions towards a formal certification of the SCADE compiler, aided by the appropriate tool support. In particular, we propose to design a translation validation strategy between some intermediate phases of the SCADE compiler.

The idea behind translation validation [6] is to check the equivalence between the source code and the generated code at compilation time. This approach has been applied for checking the correctness of the code generation of a discrete subset of Simulink [8] and extended to deal with optimising C compilers [5, 7, 9, 10].

Our aim is to realise an independent tool that checks the equivalence between some of the intermediate steps realized by the SCADE compiler. Compared to the existing work, this task involves novel challenges: the semantics of SCADE programs is radically different from the semantics of C programs, and the SCADE compiler can be seen as sequence of source-to-source transformation applied to an internal representation of clocked data-flow equations with a final and rather simple translation to sequential imperative code [1]. Our tool will work directly on the intermediate clocked data-flow representation, and the first step will be defining a normal form for clocked equations and related manipulation functions. We will then determine which information the compiler must provide to make the semantic check simple to implement and to formalise, while being reasonably efficient. Depending on the interest of the candidate, we might rely on external SMT solvers or implement, and prove correct, a dedicated checking function in a theorem prover. Even a tool not entirely proved correct will increase the safety of the compiler and easy its maintenance with
respect to the certification authorities. To make the project manageable, we might start on a simple Lustre compiler (Lustre can be considered the core language of SCADE), considering only later the fancy features of SCADE (including the type based clock calculus [3], the initialisation analysis [4], hierarchical automata [2]), etc).

**Long term goal** The ambitious, long-term, goal of this project would be to provide tools to get a formal certification (in the mathematical sense) of the industrial compiler of SCADE 6. The successful student can pursue this work with a PhD thesis in the PARKAS group, in collaboration with the compiler group of Esterel-Technologies, for instance extending the techniques to a production compiler such as the one of SCADE 6. Possibility of CIFRE funding.

**References**


