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

Sujet proposé par Marc POUZET
Directeur de thèse: Marc POUZET
Encadré par Marc POUZET
Unité de recherche UMR 8548 Laboratoire d'Informatique de l'École Normale Supérieure

Domaine: Sciences et technologies de l'information et de la communication

Projet

Location: Équipe PARKAS, 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.

Advisors: Marc Pouzet (Marc.Pouzet@ens.fr); Francesco Zappa Nardelli (Francesco.Zappa-Nardelli@inria.fr)

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 6 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 6 success but imposes a major overhead to implement even simple modifications of the compiler.

Objective: In this PhD. thesis proposal we will explore some directions towards a formal certification of the SCADE 6 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 ease its maintenance with respect to the certification authorities. The project will start on a simple Lustre compiler (Lustre can be considered the core language of SCADE), considering only later the novel features provided in SCADE 6 (including the type based clock calculus [3], the initiation analysis [4], hierarchical automata [2], etc).


Enjeux

The ambitious, long-term, goal of this project would be to provide tools to get a formal certification (in the mathematical sense) of the SCADE 6 compiler. Instead of the development of a new compiler (possibly written in Coq), our goal is identify traceability information that must be added to an industrial compiler, allowing to check with the highest confidence that the generated code is semantically correct.

Ouverture à l'international

This PhD. thesis will be done with close collaboration with the SCADE team of Esterel-Technologies. The formal certification of an industrial compiler for a block diagram language would constitute an important achievement with possible impact in other classical tools of embedded system design such as Simulink/Stateflow and Modelica.