Semantics and Compilation of Synchronous Dataflow Languages

Pierre-Évariste DAGAND
Whisper team

Synchronous dataflow languages are commonly used to implement reactive systems. Unlike usual software, a reactive system interacts directly with the physical world: it continuously receives inputs from its environment (e.g. through sensors), to which it must react by performing actions (e.g. through actuators). A dataflow language offers abstractions suited to manipulate streams of data. For instance, the Lustre \[1\] program below (left-hand side) counts the number of clock tops between two consecutive ticks. In Scade \[6\], this program is implemented by the diagram shown on the right-hand side:

```plaintext
node counting (tick: bool, top: bool)
  returns (counter: int)
  var v : int;
  let
    counter = if tick then v
              else (0 fby counter) + v;
    v = if top then 1 else 0
  tel
```

The intuition is as follows: at each logical instant, the node counting will be called with a value of tick and top. If tick is true, then the counter is reset. Otherwise, the counter is incremented by one if a top front has been detected.

This project aims at developing a certified Lustre compiler in the Coq proof assistant \[3\], taking inspiration from and building upon the Compcert \[4\] certified C compiler. This consists in:

1. Specifying a synchronous dataflow and an imperative semantics in Coq;
2. Implementing a compiler from the dataflow to the imperative paradigm, as a Coq program;
3. Proving that the semantics of dataflow programs are preserved through compilation

Being broad in scope, this project offers many opportunities for in-depth experiments as well as in-the-large developments, depending on the student’s interest.

Student’s profile: Acquaintance with an interactive theorem prover (Coq, or Isabelle) is recommended. Nonetheless, a motivated student with a strong background in functional programming (OCaml, or Haskell) could certainly learn to use Coq along the way \[5\, 2\]. No prior knowledge of a synchronous programming paradigm is expected: the development of the formal semantics in a proof assistant shall provide many opportunities to deepen one’s understanding of the formalism.

Supervision: This internship will take place at LIP6, in the Whisper team (INRIA – UPMC), where it will supervised by Pierre-Évariste Dagand. This project is part of a collaboration with the Parkas team at ENS Ulm (Timothy Bourke, Marc Pouzet) and Lionel Rieg at Collège de France.
References


