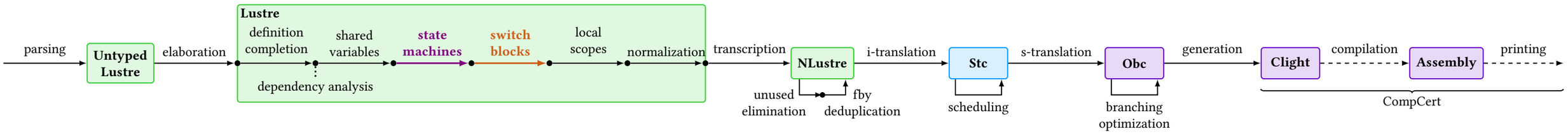
Verified Compilation of Synchronous Dataflow with State Machines

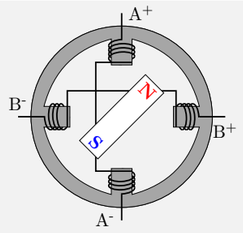
Timothy Bourke, Basile Pesin, Marc Pouzet — Inria Paris, École Normale Supérieure, PSL University

EMSOFT — September 2023



Compiling a controller for a stepper motor

```
node feed_pause(pause : bool) returns (ena, step : bool)
var time : int;
let
  reset
    time = count_up(50)
  every (false fby step);

  automaton initially Feeding
    state Feeding do
      ena = true;
      automaton initially Starting
        state Starting do
          step = true -> false
        unless false -> time >= 750 then Moving

        state Moving do
          step = true -> false
        unless time >= 500 then Moving

      end;
    unless pause then Holding

    state Holding do
      step = false;
      automaton initially Waiting
        state Waiting do
          ena = true
        unless time >= 500 then Modulating

        state Modulating do
          ena = pwm(true)

      end;
    unless
      | not pause and time >= 750 then Feeding
      | not pause continue Feeding

  end
tel
```



state machines

```
(pst, pres) = (Starting, false) fby (st, res);
switch pst
| Starting do
  reset
    (st, res) =
      if false -> time >= 750
      then (Moving, true)
      else (Starting, false)
  every pres
| Moving do ...
end;
switch st
| Starting do
  reset
    step = true -> false
  every res
| Moving do ...
end
```

switch blocks

```
resS = res when (st=Starting);
reset
  stepS = true when (st=Starting) -> false when (st=Starting)
every resS;
step = merge st (Starting => stepS) (Moving => stepM);
```

back-end to imperative code

```
switch(st) {
  case Starting:
    resS = res;
    if(resS) st.stepS = true;
    step = st.stepS;
    st.stepS = false;
    break;
  case Moving: ...
}
```

Relational Dataflow Semantics

$$\textsc{variable}\ \frac{H(x) \equiv s}{G, H \vdash x \Downarrow [s]} \qquad \textsc{equation}\ \frac{\forall i, H(xs_i) \equiv vss_i \qquad G, H \vdash es \Downarrow vss}{G, H \vdash xs = es}$$

$$\textsc{node}\ \frac{G(f) = \texttt{node}\ f(x_1, \ldots, x_n)\ \texttt{returns}\ (y_1, \ldots, y_m)\ blk \quad \forall i, H(x_i) \equiv xss_i \qquad \forall j, H(y_j) \equiv yss_j \qquad G, H \vdash blk}{G \vdash f(xss) \Downarrow yss}$$

$$\textsc{switch}\ \frac{G, H \vdash e \Downarrow [vs] \qquad \forall i, G, (\mathsf{when}^{C_i}\ vs\ H) \vdash blks_i}{G, H \vdash \texttt{switch}\ e\ [C_i\ \texttt{do}\ blks_i]^i\ \texttt{end}}$$

An operator per construct:

- `automaton` ↦ select
- `switch` ↦ when
- `reset` ↦ mask
- `last` ↦ fby

Compiler correctness in Coq

```
Theorem behavior_asm:
  ∀ D G Gp P main ins outs,
    elab_declarations D = OK (exist _ G Gp) →
    compile D main = OK P →
    sem_node G main (pStr ins) (pStr outs) →
    wt_ins G main ins →
    wc_ins G main ins →
    ∃ T, program_behaves (Asm.semantics P) (Reacts T)
         ∧ bisim_IO G main ins outs T.
```

Vélus

https://velus.inria.fr

Performances: WCET on armv7-a

	Vélus	*Hept+CC*		*Hept+gcc*		*Hept+gcci*	
stepper-motor	930	1185	(+27 %)	610	(−34 %)	535	(−42 %)
chrono	505	970	(+92 %)	570	(+12 %)	570	(+12 %)
cruisecontrol	1405	1745	(+24 %)	960	(−31 %)	895	(−36 %)
heater	2415	3125	(+29 %)	730	(−69 %)	515	(−78 %)
buttons	1015	1430	(+40 %)	625	(−38 %)	625	(−38 %)
stopwatch	1305	1970	(+50 %)	1290	(−1 %)	1290	(−1 %)

ENS | PSL

Projet financé par l'ANR — Project funded by the ANR

Embedded Systems Week