A Logical Semantics for Modes and Signals [a]

Marc Pouzet

LRI

ACI Alidecs, June 14th, 2006

[a] Joint work with Jean-Louis Colaço and Grégoire Hamon

A Core Data-flow Language

$$
\begin{array}{lcl}
e & ::= & C^n(e,...,e) \mid x \mid e_1 \texttt{ ->}_k e_2 \mid \texttt{pre}_v(e) \\
 & & \quad \mid e \texttt{ where } D \mid op(e) \mid f(e)
\end{array}
$$

$$
\begin{array}{lcl}
D & ::= & D \texttt{ and } D \mid p = e \mid \texttt{let } D \texttt{ in } D \\
 & & \quad \mid \texttt{match } e \texttt{ with } p \texttt{ -> } D \ ... \ p \texttt{ -> } D \\
 & & \quad \mid \texttt{reset } D \texttt{ every } e
\end{array}
$$

$$
\begin{array}{lcl}
p & ::= & C^n(p,...,p) \mid x \mid \_ \\
op & ::= & \texttt{+} \mid ... \\
k & ::= & 0 \mid 1 \\
d & ::= & \texttt{let node } f(p) = e \mid d; d \\
td & ::= & \texttt{type } t \mid td; td \\
 & & \texttt{type } t = \{C^n : t \times ... \times t; ...; C^n : t \times ... \times t\}
\end{array}
$$

Expressions

- Constructed values $(C^n(e_1, ..., e_n))$
- E.g., `MouseClick`, `Key(A)`; (x, y) as a short-cut for $,(x, y)$
- Initialization operator e_1 `->`$_k$ e_2 (e_1`->`e_2 as a short-cut for e_1 `->`$_0$ e_2)

Equations

- A recursive set of equations
- The regular pattern-matching construction applies to streams

```
let node ifthenelse (c, x, y) = o where
    match c with
     | true -> o = x
     | false -> o = y
```

- A modular reset

Global definitions

- Types and nodes are defined globally

Synchronous Semantics

Values: $v ::= C^n(v, ..., v) \mid C^0$

Environments:

$R ::= [v_1/x_1, ..., v_n/x_n]$ *(locals)*

$I ::= R$ *(inputs)*

$O ::= R$ *(outputs)*

Reaction:

$$R \overset{k}{\vdash} e_1 \xrightarrow{v} e_2 \quad k \in \{0, 1\}$$

$$R \overset{k}{\vdash} D \xrightarrow{R'} D' \text{ with } R' \subseteq R \quad k \in \{0, 1\} \text{ and } Def(D) = Dom(R')$$

The execution of D_1 under a sequence $H_I = I_1.I_2...$ produces $H_O = O_1.O_2...$

$$I_i + O_i + R_i \overset{1}{\vdash} D_i \xrightarrow{O_i + R_i} D_{i+1}$$

A set of global node definitions produces an initial environment:

$$S = [\lambda p_1.e_1/f_1, ..., \lambda p_n.e_n/f_n]$$

SOS rules (I)

$$\text{(C)}\ \frac{R \overset{k}{\vdash} e_1 \xrightarrow{v_1} e_1' \quad \cdots \quad R \overset{k}{\vdash} e_n \xrightarrow{v_n} e_n'}{R \overset{k}{\vdash} C^n(e_1, ..., e_n) \xrightarrow{C(v_1,...,v_n)} C^n(e_1', ..., e_n')} \qquad \text{(app)}\ \frac{R \overset{k}{\vdash} e \xrightarrow{v} e' \quad v' = op(v)}{R \overset{k}{\vdash} op(e) \xrightarrow{v'} op(e')}$$

$$\text{(WHERE)}\ \frac{R + R' \overset{k}{\vdash} D \xrightarrow{R'} D' \quad R + R' \overset{k}{\vdash} e \xrightarrow{v} e'}{R \overset{k}{\vdash} e \texttt{ where } D \xrightarrow{v} e' \texttt{ where } D'} \qquad \text{(TAUT)}\ \frac{R(x) = v}{R \overset{k}{\vdash} x \xrightarrow{v} x}$$

$$\text{(APP)}\ \frac{S(f) = \lambda p.e \quad R \overset{k}{\vdash} e \texttt{ where } p = e_1 \xrightarrow{v} e'}{R \overset{k}{\vdash} f(e_1) \xrightarrow{v} e'}$$

SOS rules (II)

$$\text{(PRE)}\ \frac{R \overset{k}{\vdash} e \xrightarrow{v'} e'}{R \overset{k}{\vdash} \mathtt{pre}_v(e) \xrightarrow{v} \mathtt{pre}_{v'}(e')}$$

$$\rightarrow_k\ \frac{R \overset{0}{\vdash} e_1 \xrightarrow{v_1} e_1' \quad R \overset{0}{\vdash} e_2 \xrightarrow{v_2} e_2'}{R \overset{0}{\vdash} e_1\ \mathtt{->}_k\ e_2 \xrightarrow{v_1} e_1'\ \mathtt{->}_1\ e_2'}$$

$$\rightarrow_{10}\ \frac{R \overset{1}{\vdash} e_1 \xrightarrow{v_1} e_1' \quad R \overset{1}{\vdash} e_2 \xrightarrow{v_2} e_2'}{R \overset{1}{\vdash} e_1\ \mathtt{->}_0\ e_2 \xrightarrow{v_1} e_1'\ \mathtt{->}_1\ e_2'} \qquad \rightarrow_{11}\ \frac{R \overset{1}{\vdash} e_1 \xrightarrow{v_1} e_1' \quad R \overset{1}{\vdash} e_2 \xrightarrow{v_2} e_2'}{R \overset{1}{\vdash} e_1\ \mathtt{->}_1\ e_2 \xrightarrow{v_2} e_1'\ \mathtt{->}_1\ e_2'}$$

SOS rules (III)

$$\text{(DEF)}\ \frac{R \overset{k}{\vdash} e \xrightarrow{v} e'}{R \overset{k}{\vdash} p = e \xrightarrow{[v/p]} p = e'} \qquad \text{(AND)}\ \frac{R \overset{k}{\vdash} D_1 \xrightarrow{R_1} D_1' \quad R \overset{k}{\vdash} D_2 \xrightarrow{R_2} D_2'}{R \overset{k}{\vdash} D_1 \,\texttt{and}\, D_2 \xrightarrow{R_1+R_2} D_1' \,\texttt{and}\, D_2'}$$

$$\text{(LET)}\ \frac{R + R_1 \overset{k}{\vdash} D_1 \xrightarrow{R_1} D_1' \quad R + R_1 \overset{k}{\vdash} D_2 \xrightarrow{R_2} D_2'}{R \overset{k}{\vdash} \texttt{let}\, D_1 \,\texttt{in}\, D_2 \xrightarrow{R_2} \texttt{let}\, D_1 \,\texttt{in}\, D_2}$$

SOS rules (IV)

$$\text{(RESET-t)}\quad \frac{R \overset{k}{\vdash} e \xrightarrow{t} e' \quad R \overset{0}{\vdash} D \xrightarrow{R'} D'}{R \overset{k}{\vdash} \texttt{reset}\ D\ \texttt{every}\ e \xrightarrow{R'} \texttt{reset}\ D'\ \texttt{every}\ e'}$$

$$\text{(RESET-f)}\quad \frac{R \overset{k}{\vdash} e \xrightarrow{f} e' \quad R \overset{k}{\vdash} D \xrightarrow{R'} D'}{R \overset{k}{\vdash} \texttt{reset}\ D\ \texttt{every}\ e \xrightarrow{R'} \texttt{reset}\ D'\ \texttt{every}\ e'}$$

$$\text{(MATCH)}\quad \frac{R \overset{k}{\vdash} e \xrightarrow{v} e' \quad \forall j \in \{1, ..., i-1\},\ p_j \not\triangleright v \quad R + [v/p_i] \overset{k}{\vdash} D_i \xrightarrow{R'} D_i'}{R \overset{k}{\vdash} \begin{array}{l}\texttt{match}\ e\ \texttt{with} \\ \quad p_1\ \texttt{->}\ D_1\ \ldots\ p_n\ \texttt{->}\ D_n\end{array} \xrightarrow{R'} \begin{array}{l}\texttt{match}\ e\ \texttt{with} \\ \quad p_1\ \texttt{->}\ D_1\ \ldots\ p_i\ \texttt{->}\ D_i'\ \ldots\ p_n\ \texttt{->}\ D_n\end{array}}$$

Adding Modes and Signals

$$
\begin{array}{lcl}
D & ::= & ... \mid \texttt{emit}\ x = e \\
 & & \mid \texttt{present}\ si\ \texttt{->}\ D\ ...\ si\ \texttt{->}\ D \\
 & & \mid \texttt{automaton}_k^{vs} \\
 & & \quad sp\ \texttt{->}\ u\ \texttt{unless}\ s \\
 & & \quad ... \\
 & & \quad sp\ \texttt{->}\ u\ \texttt{unless}\ s \\
 & & \\
u & ::= & \texttt{let}\ D\ \texttt{in}\ u \mid \texttt{do}\ D\ \texttt{until}\ s \\
s & ::= & si\ \texttt{then}\ se\ s \mid si\ \texttt{continue}\ se\ s \mid \epsilon \\
 & & \\
si & ::= & si\ \&\ si \mid x\langle p \rangle \mid e \mid \_ \\
se & ::= & S \mid S(e, ..., e) \\
sp & ::= & S \mid S(p, ..., p)
\end{array}
$$

Synchronous Semantics

Values:	v'	$::=$	$v \mid abs$	
State values:	vs	$::=$	$S \mid S(v, ..., v)$	
Environments:	R	$::=$	$[v'_1/x_1, ..., v'_n/x_n]$	*(locals)*
	I	$::=$	R	*(inputs)*
	O	$::=$	R	*(outputs)*

Reaction: $R \overset{k}{\vdash} e_1 \xrightarrow{v} e_2 \qquad R \overset{k}{\vdash} D \xrightarrow{R'} D' \qquad k \in \{0, 1\}$

A synchronous reaction $R \overset{k}{\vdash} D \xrightarrow{R'} D'$ must verify the three invariants:

1. $x \in Def(D) \Rightarrow [v/x] \in R \wedge [v/x] \in R'$
2. $x \in Emit(D) \wedge [v/x] \in R' \Rightarrow [v/x] \in R$
3. $x \in Emit(D) \wedge [v/x] \notin R' \Rightarrow [abs/x] \in R$

Signal Matching

$$\text{(TEST-abs)}\quad R + [abs/x] \overset{k}{\vdash} x\langle p\rangle \xrightarrow[f]{\emptyset} x\langle p\rangle \qquad \text{(WILD)}\quad R \overset{k}{\vdash} \_ \xrightarrow[t]{\emptyset} \_$$

$$\text{(TEST-f)}\ \frac{p \not\rhd v}{R + [v/x] \overset{k}{\vdash} x\langle p\rangle \xrightarrow[f]{\emptyset} x\langle p\rangle} \qquad \text{(TEST-t)}\ \frac{p \rhd v}{R + [v/x] \overset{k}{\vdash} x\langle p\rangle \xrightarrow[t]{[v/p]} x\langle p\rangle}$$

$$\text{(\&)}\ \frac{R \overset{k}{\vdash} si_1 \xrightarrow[v_1]{R_1} si_1' \quad R \overset{k}{\vdash} si_2 \xrightarrow[v_2]{R_2} si_2'}{R \overset{k}{\vdash} si_1 \,\&\, si_2 \xrightarrow[v_1 \wedge v_2]{R_1 + R_2} si_1' \,\&\, si_2'} \qquad \text{(EXP)}\ \frac{R \overset{k}{\vdash} e \xrightarrow{v} e'}{R \overset{k}{\vdash} e \xrightarrow[v]{\emptyset} e'}$$

The `present` construction is similar to the `match/with` construction

Automata (I)

- when reseted, an automaton starts in its initial state
- strong transitions are fired at the begining of the reaction
- weak transitions are fired at the end

$$\frac{R \overset{0}{\underset{sp_1}{\vdash}} s_1 \overset{vs}{\underset{k}{\rightarrow}} s_1' \qquad sp_i \rhd vs \qquad R + [vs/sp_i] \overset{0}{\underset{vs}{\vdash}} u_i \overset{R'}{\underset{k_0', vs'_0}{\rightarrow}} u_i'}{R \overset{0}{\vdash} \texttt{automaton}^{vs_0}_{k_0} \overset{R'}{\longrightarrow} \texttt{automaton}^{vs'_0}_{k_0'}}$$

$$\begin{array}{ll}
sp_1 \texttt{ -> } u_1 \texttt{ unless } s_1 & sp_1 \texttt{ -> } u_1 \texttt{ unless } s_1' \\
\ldots & \ldots \\
sp_i \texttt{ -> } u_i \texttt{ unless } s_i & sp_i \texttt{ -> } u_i' \texttt{ unless } s_i \\
\ldots & \ldots \\
sp_n \texttt{ -> } u_n \texttt{ unless } s_n & sp_n \texttt{ -> } u_n \texttt{ unless } s_n
\end{array}$$

Automata (II)

- the automaton is annotated with the current active state and the reset bit
- they are determined by the previous weak transition

$$sp_j \rhd vs_0 \quad R \overset{k_0}{\underset{vs_0}{\vdash}} s_j \overset{vs}{\underset{k}{\longrightarrow}} s'_j \quad sp_i \rhd vs \quad R + [vs/sp_i] \overset{k}{\underset{vs}{\vdash}} u_i \overset{R'}{\underset{k'_0, vs'_0}{\longrightarrow}} u'_i$$

$$R \overset{1}{\vdash} \texttt{automaton}^{vs_0}_{k_0} \xrightarrow{R'} \texttt{automaton}^{vs'_0}_{k'_0}$$

$$\begin{array}{ll}
sp_1 \rightarrow u_1 \texttt{ unless } s_1 & \ldots \\
\ldots & sp_j \rightarrow u_j \texttt{ unless } s'_j \\
sp_i \rightarrow u_i \texttt{ unless } s_i & \ldots \\
\ldots & sp_i \rightarrow u'_i \texttt{ unless } s_i \\
sp_n \rightarrow u_n \texttt{ unless } s_n & \ldots \\
 & sp_n \rightarrow u_n \texttt{ unless } s_n
\end{array}$$

Escape Transitions

$$\text{(TH-t)}\ \frac{R \overset{k}{\vdash} si \xrightarrow[t]{R'} si' \quad R + R' \overset{k}{\vdash} se \xrightarrow{vs'} se' \quad R \overset{k}{\underset{vs}{\vdash}} s \xrightarrow[k']{vs''} s'}{R \overset{k}{\underset{vs}{\vdash}} si\ \texttt{then}\ se\ s \xrightarrow[0]{vs'} si'\ \texttt{then}\ se'\ s'}$$

$$\text{(TH-f)}\ \frac{R \overset{k}{\vdash} si \xrightarrow[f]{R'} si' \quad R \overset{k}{\underset{vs}{\vdash}} s \xrightarrow[k']{vs'} s'}{R \overset{k}{\underset{vs}{\vdash}} si\ \texttt{then}\ se\ s \xrightarrow[k']{vs'} si'\ \texttt{then}\ se'\ s'}$$

$$\text{(Epsilon)}\quad R \overset{k}{\underset{vs}{\vdash}} \epsilon \xrightarrow[1]{vs} \epsilon$$

- same rules for the transitions "by history" (reset bit $k = 1$ instead)

Translation Semantics

- The above semantics complements the translation semantics and is more direct
- The translation semantics express how programs are translated into a basic kernel
- The proof that they coincide has to be done

Example

```
let node sum (a, b, r) = o where
    automaton
    | Await -> do unless a⟨x⟩&b⟨y⟩ then Emit (x + y)
    | Emit (v) -> do emit o = v unless r then Await
```

- Signals can be represented as constructed values (this is how they would be implemented in LUSTRE)
- Signal testing becomes pattern-matching

```
let node sum (a, b, r) = o where
    match pnextstate with
    | Await -> match (a, b) with
                | (P(x), P(y) -> state = Emit(x + y)
                | _ -> state = Await
    | Emit(v) -> match r with
                  | true -> state = Await
                  | false -> state = Emit(v)
    and
    match state with
    | Await -> o = Abs and nextstate = Await
    | Emit(v) -> o =P(v) and nextstate = Emit(v)
    and
    pnextstate = Await -> pre nextstate
```

Discussion (I)

The "finger" semantics

- Only one set of equations is executed during a reaction (clear)
- Should we cross exactly one transition during a reaction?

In the above semantics:

- only one kind of transition during a reaction but
- a weak followed by a strong or a strong followed by a weak or

Other possibilities:

- forbid it (Scade V6), using extra bit testing
- should we forbid automata mixing the two kinds of transitions?
- should we forbid states mixing the two kinds of transitions?

Discussion (II)

- Transition "by history" *vs* States "by history" (StateFlow)?
- What about equations on transitions? (done in SCADE V6, not in Lucid Synchrone)
- Components:
  - a mode is a bunch of equations
  - their activation (controler) is defined by an automaton