Constructive Polychronous Systems

Jens Brandt Mike Gemünde Klaus Schneider (Uni Kaiserslautern)
Sandeep Shukla (Virginia Tech) Jean-Pierre Talpin (INRIA)

To appear in LFCS'13

Partially supported by INRIA project Polycore, by USAFRL grant FA8750-11-1-0042 and DFG.

Synchronous Languages

- Esterel, Lustre, Signal, ...
  A. Benveniste, P. Caspi, S. Edwards, N. Halbwachs, P. Le Guernic and R. de Simone. *The Synchronous Languages Twelve Years Later*. Proceedings of the IEEE, 2003.
- domain-specific languages for embedded system design
- discrete control programs for continuous, physical environments
- express concurrency in a user-friendly manner
- same mathematical foundation: "the synchronous hypothesis", ...

..., what ?

- same mathematical foundation: "the synchronous hypothesis", but:
  - different visuals: block diagrams, hierachical automata, ...
  - different styles: imperative, dataflow, ...
  - different models: reactive, synchronous, polychronous, ...
  - different properties: constructive, determinism, endochrony, ...

Goal

Use the Signal/Polychrony toolset in the Averest/Quartz environment

Why ? ...

... use the Polychrony toolset in the Averest environment

- Gain productivity from communication and computation scheduling from automatic synthesis
  - Write sound functional modules
  - Automate scheduler synthesis
- Implied programming methodology
  - Specify/program imperative reactive modules in Quartz
  - Synthesize their scheduler from a Signal network specified by constraints, interfaces, contracts

Needs

A constructive framework encompassing reactive Quartz modules and endochronous Signal networks

How ?

A common constructivity framework

- Stabilization of electric circuits [Huffman, 7'1, Malik, '94]
- Electrical semantics of Esterel [Shiple, '96, Berry, '99]
- Constructive Boolean circuits [Mendler, Shiple, Berry, '12]

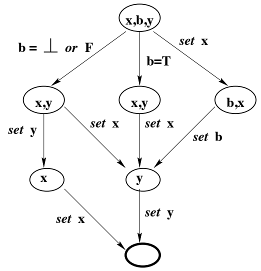
Goal

- A lattice and fixpoint theory for clocked streams
- A common semantic framework for Quartz and Signal
- An executable structured operational semantics of Signal

Starting point

- Game-theoretical micro-automata for Signal [Benveniste et al., 2000]

$[\![x \to^b y]\!] =$

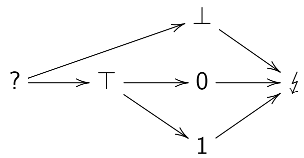


First sub-goal

⇒ Define a complete lattice / fixpoint theory for multi-clocked systems

A lattice for the status of variables/signals

? signal status is unknown
$\bot$ signal is absent or inhibited
$\top$ signal is present or activated
↯ signal is inconsistent



Basic principle

- A complete lattice of signal and variable status
- Monotonic functions to define status transitions
- A constructive fixed-point theory

Value of a boolean operation

Start from the truth table

$\wedge^{\mathcal{D}}$	?	$\bot$	$\top$	0	1	↯
?						
$\bot$						
$\top$						
0				0	0	
1				0	1	
↯						

Status of a boolean operation

Fill it with all possible errors

$\wedge^{\mathcal{D}}$	?	$\bot$	$\top$	0	1	$\lightning$
?						$\lightning$
$\bot$			$\lightning$	$\lightning$	$\lightning$	$\lightning$
$\top$		$\lightning$				$\lightning$
0		$\lightning$		0	0	$\lightning$
1		$\lightning$		0	1	$\lightning$
$\lightning$	$\lightning$	$\lightning$	$\lightning$	$\lightning$	$\lightning$	$\lightning$

Status of a boolean operation

Complete by being positive (monotonic)

$\wedge^{\mathcal{D}}$	?	$\bot$	$\top$	0	1	$\lightning$
?	?	$\bot$	$\top$	0	$\top$	$\lightning$
$\bot$	$\bot$	$\bot$	$\lightning$	$\lightning$	$\lightning$	$\lightning$
$\top$	$\top$	$\lightning$	$\top$	0	$\top$	$\lightning$
0	0	$\lightning$	0	0	0	$\lightning$
1	$\top$	$\lightning$	$\top$	0	1	$\lightning$
$\lightning$	$\lightning$	$\lightning$	$\lightning$	$\lightning$	$\lightning$	$\lightning$

Quartz as synchronous guarded actions

Model Quartz programs as guarded actions on (continuous) variables

$$
\begin{array}{rll}
p, q ::= & \gamma \Rightarrow x = \tau & \text{(immediate assignment)} \\
\mid & \gamma \Rightarrow \text{next}(x) = \tau & \text{(delayed assignment)} \\
\mid & \text{init}(x) = \tau & \text{(initialization)} \\
& & \\
\mid & p \,|\, q & \text{(composition)} \\
\mid & \text{var}\, x \colon p\, \text{default}\, v & \text{(action block)}
\end{array}
$$

Small-step semantics $s, p \rightharpoonup s', q$ of Quartz (actions)

For $\star \in \{\text{and}, \text{or}\}$, define $s, x \star y \rightharpoonup v$ iff $v = s(x) \star s(y) \in \mathbb{B}$

$$\frac{s, \gamma \rightharpoonup 0}{s, \gamma \Rightarrow x = \tau \rightharpoonup s, \text{done}\ (\gamma \Rightarrow x = \tau)}$$

$$\frac{s, \gamma \rightharpoonup 1 \quad s, \tau \rightharpoonup v}{s, \gamma \Rightarrow x = \tau \rightharpoonup s \uplus (x, v), \text{done}\ (\gamma \Rightarrow x = \tau)}$$

Small-step semantics of Quartz (delayed actions)

$$s,\ \mathrm{init}(x) = v \rightharpoonup s \uplus (x, v), ()$$

$$\frac{s, \gamma \rightharpoonup 0}{s, \gamma \Rightarrow \mathrm{next}(x) = \tau \rightharpoonup s, \mathrm{done}\ (\gamma \Rightarrow \mathrm{next}(x) = \tau)}$$

$$\frac{s, \gamma \rightharpoonup 1 \quad s, \tau \rightharpoonup v}{s, \gamma \Rightarrow \mathrm{next}(x) = \tau \rightharpoonup s, \mathrm{done}\ (\mathrm{init}(x) = v \,|\, \gamma \Rightarrow \mathrm{next}(x) = \tau)}$$

Small-step semantics of Quartz (scheduling)

$$\frac{s' = (s(x) \in \mathcal{D})?s, s \uplus (x, v) \quad w = (x \in \mathcal{M})?s'(x), v}{s, \mathrm{var}\ x\colon \mathrm{done}\ (p)\ \mathrm{default}\ v \rightharpoonup s', \mathrm{done}\ (\mathrm{var}\ x\colon p\ \mathrm{default}\ w)}$$

$$\frac{s, p \rightharpoonup s', p'}{s,\ p \,|\, q \rightharpoonup s',\ p' \,|\, q} \qquad \frac{s, p \rightharpoonup s', p'}{s,\ q \,|\, p \rightharpoonup s',\ q \,|\, p'}$$

$$s,\ \mathrm{done}\ p \,|\, \mathrm{done}\ q \rightharpoonup s, \mathrm{done}\ (p \,|\, q)$$

Semantics $s, p \rightharpoonup s', q$ for equations over signals (function)

Model a polychronous network as equations over clocked signals

$$\frac{s(x, y, z) \rightharpoonup_{\mathrm{and}} (a, b, c)}{s,\ x := y\ \mathrm{and}\ z \rightharpoonup s \uplus (x, a)(y, b)(z, c),\ x := y\ \mathrm{and}\ z}$$

Propagate knowledge of absence

$s(x)$	$s(y)$	$s(z)$	a	b	c
$\bot$	$?/\bot$	$?/\bot$	$\bot$	$\bot$	$\bot$
$?/\bot$	$\bot$	$?/\bot$	$\bot$	$\bot$	$\bot$
$?/\bot$	$?/\bot$	$\bot$	$\bot$	$\bot$	$\bot$

Equations over signals (function)

$$\frac{s(x, y, z) \rightharpoonup_{\mathrm{and}} (a, b, c)}{s,\ x := y\ \mathrm{and}\ z \rightharpoonup s \uplus (x, a)(y, b), (z, c),\ x := y\ \mathrm{and}\ z}$$

Propagate knowledge of presence

$s(x)$	$s(y)$	$s(z)$	a	b	c
$\top$	$?/\top$	$?/\top$	$\top$	$\top$	$\top$
$?/\top$	$\top$	$?/\top$	$\top$	$\top$	$\top$
$?/\top$	$?/\top$	$\top$	$\top$	$\top$	$\top$

Equations over signals (function)

$$\frac{s(x, y, z) \rightharpoonup_{\text{and}} (a, b, c)}{s,\ x := y \text{ and } z \rightharpoonup s \uplus (x, a)(y, b), (z, c),\ x := y \text{ and } z}$$

Progress by computations

$s(x)$	$s(y)$	$s(z)$	a	b	c
$?/\top$	a	$?/\top$	$\top$	a	$\top$
$?/\top$	$?/\top$	a	$\top$	$\top$	a
$?/\top$	b	a	$a \wedge b$	b	a

Notice that $\rightharpoonup_{\text{and}}$ is monotonic and increasing

* $a, b \in \mathbb{B}$

Equations over signals (sampling)

$$s(x, y, z) \rightharpoonup_{\text{when}} (a, b, c)$$

Progress by the logic of absence, presence and by computations

$s(x)$	$s(y)$	$s(z)$	a	b	c
$\bot$	$?/\bot$	$?/\bot$	$\bot$	$\bot$	$\bot$
$?/\bot$	$\bot$	$?/\bot$	$\bot$	$\bot$	$\bot$
$?/\bot$	$?/\bot$	$\bot$	$\bot$	$\bot$	$\bot$
$?/\bot$	x	0	$\bot$	x	0
$\top$	$?/\top$	b	$\top$	$\top$	$a \sqcup 1$
$?/\top$	a	1	a	a	1

* Rules for absence/presence encode the clock calculus

Equations over signals (merging)

$$s(x, y, z) \rightharpoonup_{\text{default}} (a, b, c)$$

Progress by the logic of absence, presence and by computations

$s(x)$	$s(y)$	$s(z)$	a	b	c
$\bot$	$?/\bot$	$?/\bot$	$\bot$	$\bot$	$\bot$
$?/\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$
$?$	$\top$	x^*	$\top$	$\top$	x
$?$	x	$\top$	$\top$	x	$\top$
$?/\top$	a	x	a	a	x
$?/\top$	$\bot$	a	a	$\bot$	a

* x is the "don't care", i.e., any status except error

Equations over signals (delay)

$$\frac{s(x, y), a \rightharpoonup_{\$\text{init}} (b, c, d)}{s,\ x := y \ \$\text{init}\ a \rightharpoonup s \uplus (x, b)(y, c),\ x := y \ \$\text{init}\ d}$$

Progress by the logic of absence, presence and by computations

$s(x)$	$s(y)$	a	b	c	d
$\bot$	$?/\bot$	a	$\bot$	$\bot$	a
$?/\bot$	$\bot$	a	$\bot$	$\bot$	a
$\top$	$?/\top$	a	a	$\top$	a
$?/\top$	$\top$	a	a	$\top$	a
$?/\bot$	b	a	a	b	b
a	b	a	a	b	b

... Wait!

Equations over signals (delay)

.... Where you listening ?

$$\frac{s(x,y), a \rightharpoonup_{\$\text{init}} (b,c,d)}{s,\ x := y \,\$\text{init}\, a \rightharpoonup s \uplus (x,b)(y,c),\ x := y \,\$\text{init}\, d}$$

It's not quite

$s(x)$	$s(y)$	a		b	c	d
$?/\bot$	b	a	$\not\sqsubseteq$	a	b	b
a	b	a	$\not\sqsubseteq$	a	b	b

but more exactly

$s(x)$	$s(y)$	a_{t-1}	a_t		b	c	a_{t-1}	d
$?/\bot$	b	a	$?/\top$	$\sqsubseteq$	a	b	a	b
a	b	a	$?/\top$	$\sqsubseteq$	a	b	a	b

Equations over signals (a run)

$$c := o \,\$\text{init}\, 1 \,\|\, o := n \,\text{default}\, \overbrace{c-1}^{x} \,\|\, n \triangleq \text{when}\ \overbrace{c=0}^{y}$$

$$\begin{array}{lll}
(c,?)(n,?)(o,\top_\bullet)(x,?)(y,?) & \rightharpoonup_{\$\text{init}} & (c,1_\bullet)\ (n,?)\ (o,\top)\ (x,?)\ (y,?) \\
 & \rightharpoonup_{sub} & (c,1)\ (n,?)\ (o,\top)(x,0_\bullet)\ (y,?) \\
 & \rightharpoonup_{eq} & (c,1)\ (n,?)\ (o,\top)\ (x,0)\ (y,ff_\bullet) \\
 & \rightharpoonup_{\triangleq} & (c,1)\ (n,\bot_\bullet)(o,\top)\ (x,0)\ (y,ff) \\
 & \rightharpoonup_{\text{default}} & (c,1)\ (n,\bot)\ (o,0_\bullet)\ (x,0)\ (y,ff) \\
 & \rightharpoonup_{\$\text{init}} & (c,1)\ (n,\bot)\ (o,0)\ (x,0)\ (y,ff)
\end{array}$$

$c := o \,\$\text{init}\, 0_\bullet \,\|\, \ldots$

From synchrony to asynchrony

Model FIFO buffers ...

$$\begin{array}{llll}
w & ::= & \epsilon & \text{(empty)} \\
 & | & a.w & \text{(read)} \\
 & | & w.a & \text{(write)}
\end{array}$$

... to define the interface of a process p with the network P

$$P, Q ::= \langle s, p\rangle \mid P \parallel Q \quad \text{(network)}$$

... and get something like that: progress with $\rightharpoonup$ and writeout with $\rightharpoondown$

$$E, \langle s_0, p\rangle \rightharpoonup^* F, \langle s, q\rangle \rightharpoondown^* G, \langle s_0, r\rangle \quad (\textit{big step})$$

$s_0 = \{(x,?) \mid x \in V(p)\}$

Interface semantics $E, P \rightharpoonup F, Q$ (trigger and read)

$$\frac{s, p \rightharpoonup t, q}{E, \langle s, p\rangle \rightharpoonup E, \langle t, q\rangle} \qquad (\textit{embedding rule})$$

$$\frac{x \in T(p)}{E, \langle s \uplus (x,?), p\rangle \rightharpoonup E, \langle s \uplus (x,\top), p\rangle} \qquad (\textit{trigger execution})$$

$$\frac{x \in I(p)}{E \uplus (x, a.w), \langle s \uplus (x,\top), p\rangle \rightharpoonup E \uplus (x, w), \langle s \uplus (x,a), p\rangle} \qquad (\textit{read inputs})$$

Interface semantics $E, P \rightharpoondown F, Q$ (write and sync)

$$\frac{x \in O(p) \wedge a \in \mathbb{B}}{E \uplus (x, w), \langle s \uplus (x, a), p\rangle \rightharpoondown E \uplus (x, w.a), \langle s, p\rangle} \quad (\textit{write output})$$

$$\frac{x \notin O(p) \vee a = \bot}{E, \langle s \uplus (x, a), p\rangle \rightharpoondown E, \langle s \uplus (x, ?), p\rangle} \quad (\textit{filter locals, absence})$$

Interface semantics $E, P \rightharpoondown F, Q$ (reactive modules)

Interface semantics of Quartz is much simpler:

$(\textit{read})$	$E \uplus (x, a.w), \langle s, p\rangle \rightharpoondown E \uplus (x, w), \langle s_x \uplus (x, a), p\rangle$	$x \in I(p)$
$(\textit{write})$	$E \uplus (x, w), \langle s \uplus (x, a), p\rangle \rightharpoondown E \uplus (x, w.a), \langle s_x \uplus (x, ?), p\rangle$	$x \in O(p)$
$(\textit{mask})$	$E, \langle s, p\rangle \rightharpoondown E, \langle s_x \uplus (x, ?), p\rangle$	$x \in L(p)$
$(\textit{restart})$	$E, \langle s, \text{done } p\rangle \rightharpoondown E, \langle s, p\rangle$	

Equations over clocked signals (interface with streams)

$$c := o \,\$\text{init}\, 0 \,|\, o := n \,\text{default}\, \overbrace{c - 1}^{x} \,|\, n \hat{=} \text{when}\, \overbrace{c = 0}^{y}$$

$$\begin{array}{lll}
 & \vdots & \\
 & [(n, 42)(o, \epsilon)] \vdash (c, ?)\ (n, ?)\ (o, \top)\ (x, ?)\ (y, ?) & \\
\rightharpoondown_{\$\text{init}} & [(n, 42)(o, \epsilon)] \vdash (c, 0)\ (n, ?)\ (o, \top)\ (x, ?)\ (y, ?) & \\
\rightharpoondown_{sub} & [(n, 42)(o, \epsilon)] \vdash (c, 0)\ (n, ?)\ (o, \top)(x, -1)(y, ?) & \\
\rightharpoondown_{eq} & [(n, 42)(o, \epsilon)] \vdash (c, 0)\ (n, ?)\ (o, \top)(x, -1)(y, tt) & \\
\rightharpoondown_{\hat{=}} & [(n, 42)(o, \epsilon)] \vdash (c, 0)\ (n, \top)\ (o, \top)(x, -1)(y, tt) & \\
\rightharpoondown & [(n, \epsilon_\bullet)(o, \epsilon)] \vdash (c, 0)(n, 42_\bullet)(o, \top)(x, -1)(y, tt) & (\textit{read}) \\
\rightharpoondown_{\text{default}} & [(n, \epsilon)(o, \epsilon)] \vdash (c, 0)\ (n, 42)\ (o, 42)(x, -1)(y, tt) & \\
\rightharpoondown_{\$\text{init}} & [(n, \epsilon)(o, \epsilon)] \vdash (c, 0)\ (n, 42)\ (o, 42)(x, -1)(y, tt) & \\
\rightharpoondown^{*} & [(n, \epsilon)(o, 42)_\bullet] \vdash ()_\bullet & (\textit{write})
\end{array}$$

$c := o \,\$\text{init}\, 42 \,|\, \ldots$

A constructive GALS semantics

$$\frac{E, P \rightharpoonup F, Q}{E, P \rightarrow E, Q} \quad (\textit{step})$$

$$\frac{E, \langle s, p\rangle \rightharpoondown^{*} F, \langle s', q\rangle}{E, \langle s, p\rangle \rightarrow F, \langle s', q\rangle} \quad (\textit{flush})$$

$$\frac{E, P \rightarrow E', P'}{E, P \parallel Q \rightarrow E', P' \parallel Q} \quad \frac{E, P \rightarrow E', P'}{E, P \parallel Q \rightarrow E', P' \parallel Q} \quad (\textit{schedule})$$

Formal properties

Definition (Synchronous constructivity)

A module p is synchronous constructive iff.

for all $s_0 = \{(x, a) \mid x \in I(p), a \in \mathbb{B}\} \cup \{(y, ?) \mid y \in V(p) \setminus I(p)\}$

we have $s_0, p \rightharpoonup^* s, q$ and s is defined on $\mathbb{B}$ (i.e. $s = lfp \rightharpoonup_p(s_0)$)

Proposition

*If p is synchronously constructive then p is reactive**

* a.k.a. synchronously deterministic

Formal properties

Definition (Asynchronous constructivity)

A process p is asynchronously constructive iff.

for any environment E of non-empty streams defined on $V(p)$

and $s_0 = \{(x, ?) \mid x \in V(p)\}$

we have $E, \langle s_0, p\rangle \rightharpoonup^* F, \langle s, q\rangle$

and s is defined on $\mathbb{B}^\perp$ (i.e. $(F, s) = lfp \rightharpoonup_p(E, s_0)$).

Proposition

If p is asynchronously constructive then p is asynchronously deterministic

No reaction to absence

A process should stutter when its triggers are inhibited

$$\frac{x \in T(p)}{E, \langle s \uplus (x, ?), p\rangle \rightharpoonup E, \langle s \uplus (x, \perp), p\rangle}$$

Definition (Stuttering)

A process p is robust to stuttering iff.

for $s = \{(x, \perp) \mid x \in T(p)\} \cup \{(x, ?) \mid x \in V(p) \setminus V(p)\}$

we have $\langle s, p\rangle \rightharpoonup^* \langle s', p\rangle$ and $s' = \{(x, \perp) \mid x \in V(p)\}$

Note: this captures (weakly) endochronous systems

Discussions and future works

- An executable operational semantics of Signal and Quartz
  - simulates scheduling of data-flow equations
  - takes into account their interface with streams
- A common constructive model for Quartz and Signal

⇒ Extend constructivity analysis to polychronous (concurrent) systems

⇒ A constructivity framework for abstract interpretation

Oversampling and clock domains

Define the sum $\Sigma_{i=1}^{n}$ from the counter (triggered by o),

$$sigma(n, s) \triangleq (counter(n, o) \,|\, (i := (i\$\mathrm{init}\,0) + o \,|\, s := i \,\mathrm{when}\,(o = 1)) \,/ i) \,/ o$$

Sigma is triggered by o, and, equivalently

$$sigma'(n, s) \triangleq (counter(n, o) \parallel (i := (i\$\mathrm{init}\,0) + o \,|\, s := i \,\mathrm{when}\,(o = 1)) \,/ i) \,/ o$$