Abstract clocks for the N-synchronous model

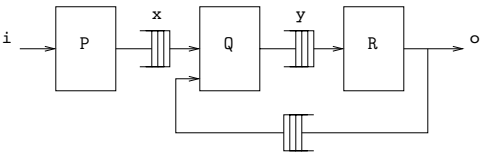
Florence Plateau - Marc Pouzet

LRI, Univ. Paris Sud - France

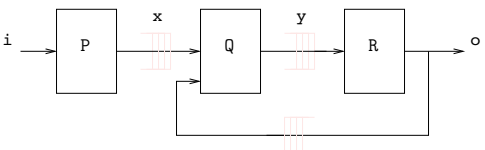
SYNCHRON - November 26th, 2007

Programming Kahn Networks

Kahn Networks :



Synchronous Dataflow Langages (Lustre, Signal, Lucid Synchrone) allow to program synchronous kahn networks.




```
let node my_skn i =
let x = P i
and y = Q x (0 fby o)
and o = R y in
o
```

Synchronous Dataflow Langages

flow	value	clock type
nat	0 1 2 3 4 5 ...	*base*
prime	1 3 5 7 11 13 ...	*base*
twoOvFour	t t f f t t ...	*base*

Synchronous Dataflow Langages

flow	value	clock type
nat	0 1 2 3 4 5 ...	*base*
prime	1 3 5 7 11 13 ...	*base*
twoOvFour	t t f f t t ...	*base*
$x = nat$ *when twoOvFour*	0 1 4 5 ...	*base* on *twoOvFour*
$y = prime$ *when twoOvFour*	1 3 11 13 ...	*base* on *twoOvFour*
$z = x + y$	1 4 15 18 ...	*base* on *twoOvFour*

Synchronous Dataflow Langages

flow	value	clock type
nat	0 1 2 3 4 5 ...	*base*
prime	1 3 5 7 11 13 ...	*base*
twoOvFour	t t f f t t ...	*base*
$x = nat$ *when twoOvFour*	0 1 4 5 ...	*base* on *twoOvFour*
$y = prime$ *when twoOvFour*	1 3 11 13 ...	*base* on *twoOvFour*
$z = x + y$	1 4 15 18 ...	*base* on *twoOvFour*
oneOvTwo	t f t f ...	*base* on *twoOvFour*
x when oneOvTwo	0 4 ...	*base* on *twoOvFour* on *oneOvTwo*
oneOvFour	t f f f t f ...	*base*
prime when oneOvFour	1 11 ...	*base* on *oneOvFour*

Clocks can be arbitrarily complex : `let clock` $c1 = (x \leq 0)\&b$

equality of clocks = syntaxic equality

Synchronous model

- clocks = boolean flows, arbitrarily complex
- composition : syntaxically equal clocks, structural unification
  ex.: $\alpha_1 \texttt{ on } c_1 \equiv \alpha_2 \texttt{ on } c_2 \texttt{ on } c_3$
  $\Leftrightarrow (c_1 \overset{synt}{=} c_3) \wedge (\alpha_1 \equiv \alpha_2 \texttt{ on } c_2)$

Periodic clocks

- ex. :

$$\begin{array}{lcl} 010(1101) & = & 010\ 1101\ 1101\ 1101\ 1101\ 1101\ \ldots \\ (1100) & = & 1100\ 1100\ 1100\ 1100\ 1100\ 1100\ \ldots \\ (10) & = & 10\ 10\ 10\ 10\ 10\ 10\ 10\ 10\ 10\ \ldots \end{array}$$

- we can compute the $\texttt{on} : (1100)\ \texttt{on}\ (10) = (1000)$
  it is associative $(p_1\ \texttt{on}\ p_2)\ \texttt{on}\ p_3 = p_1\ \texttt{on}\ (p_2\ \texttt{on}\ p_3)$.

flow	value	clock type
x	$x_0\ x_1\ x_2\ x_3\ x_4\ x_5\ x_6\ \ldots$	*base*
(1100)	$1\ 1\ 0\ 0\ 1\ 1\ 0\ \ldots$	*base*
x *when* (1100)	$x_0\ x_1 \qquad x_4\ x_5 \qquad \ldots$	*base* on (1100)
(10)	$1\ 0 \qquad 1\ 0 \qquad \ldots$	*base* on (1100)
x *when* (1100) *when* (10)	$x_0 \qquad x_4 \qquad \ldots$	*base* on (1100) on (10)

Synchronous Model with periodic clocks

- clocks = ultimately periodic boolean flows
- composition : equal clocks, computed unification.
  ex.: α_1 on (10) $\equiv \alpha_2$ on (1000)
  $\Leftrightarrow (\alpha_1 = \alpha'_1$ on (1100)) $\wedge (\alpha'_1 \equiv \alpha_2)$
- Particular case $0^d(10^n)$: affine clocks (I. Smarandache)

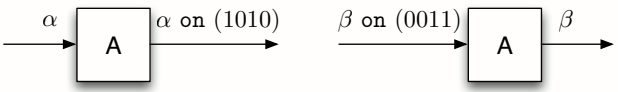
Relaxing the synchronous condition

Goal : more flexibility in the composition of flows

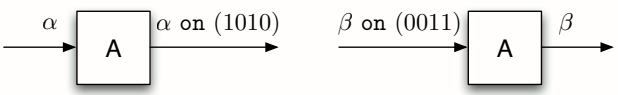
- 0-synchronous : $h_a = h_b$, no buffers.
- n-synchronous : $h_a <: h_b$, buffer of size n.
- inference of the buffers size
- automatic translation to a synchronous kahn network

Example

Synchronous Model:

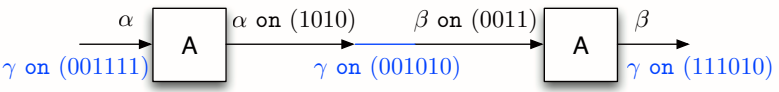


N-Synchronous model:

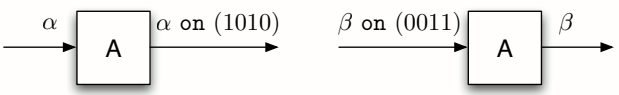


Example

Synchronous Model: unification

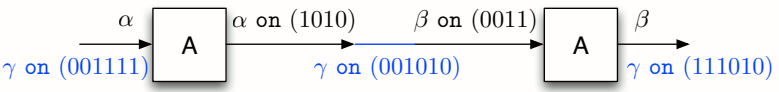


N-Synchronous model:

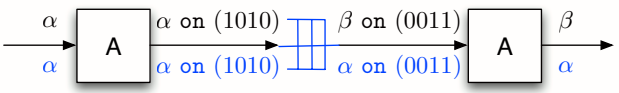


Example

Synchronous Model: unification

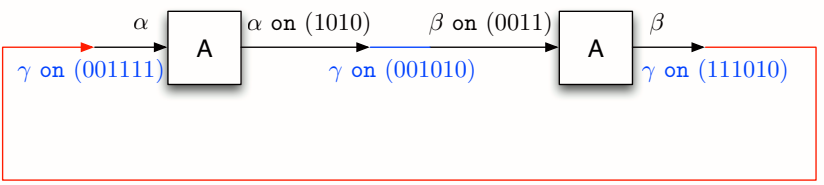


N-Synchronous model: sub-typing

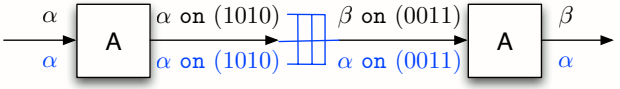


Example

Synchronous Model: the program is rejected

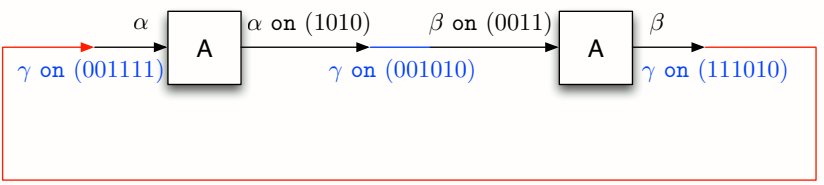


N-Synchronous model: sub-typing

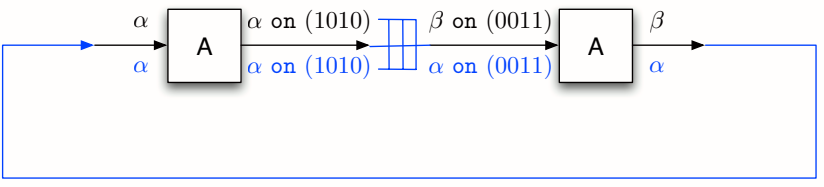


Example

Synchronous Model: the program is rejected



N-Synchronous model: the program is accepted



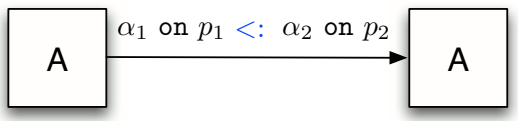
Sub-typing Relation

As we use periodic words,
we can compare the clocks:

- synchronizability: $p_1 \bowtie p_2$
  "p_1 as in average as many ones as p_2"
- precedence : $p_1 \preceq p_2$
  "the 1 of p_1 arrive before those of p_2"

sub-typing relation: $p_1 <: p_2 \Leftrightarrow p_1 \bowtie p_2 \wedge p_1 \preceq p_2$

Clock calculus



- We collect a set of constaints of the form $S : \{\alpha_i \text{ on } p_i <: \alpha_j \text{ on } p_j\}_{i,j}$
- Finding a valid instantiation of variables :
  Complete Resolution :
  $S \Leftrightarrow S' \Leftrightarrow ... \Leftrightarrow \{\ \}$
  Correct Resolution :
  $S \Leftarrow S' \Leftarrow ... \Leftarrow \{\ \}$

Difficulty 1 : no structural simplification of on

$$\alpha \text{ on } p <: \beta \text{ on } p \not\Rightarrow \alpha <: \beta$$

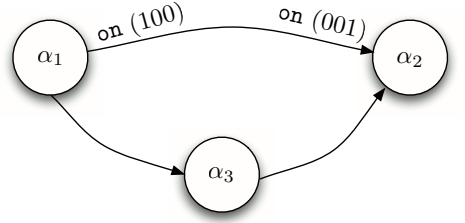
for ex. :

$$\begin{array}{rlcll} \alpha & \text{on } (01) & <: & \beta & \text{on } (01) \\ \gamma \text{ on } (0110) & \text{on } (01) & <: & \gamma \text{ on } (1001) & \text{on } (01) \\ \gamma \text{ on } (0110) & & \not<: & \gamma \text{ on } (1001) & \end{array}$$

$\Rightarrow$ The on in the constraints must be computed

$$\begin{array}{rlll} & \alpha \text{ on } w \quad \text{on } w_1 \ldots \text{ on } w_n & <: & \beta \text{ on } w' \quad \text{on } w_1 \ldots \text{ on } w_n \\ \not\Rightarrow & \alpha \text{ on } w & <: & \beta \text{ on } w' \end{array}$$

Difficulty 2 : we cannot always work locally



$$\left\{ \begin{array}{lcl} \alpha_1 \texttt{ on } (100) & <: & \alpha_2 \texttt{ on } (001) \\ \alpha_1 & <: & \alpha_3 \\ \alpha_3 & <: & \alpha_2 \end{array} \right\}$$

Here the constraint by transitivity is more difficult to satisfy than the direct constraint.

NB : We cannot always compare 2 constraints

Constraints resolution by the abstraction of periods

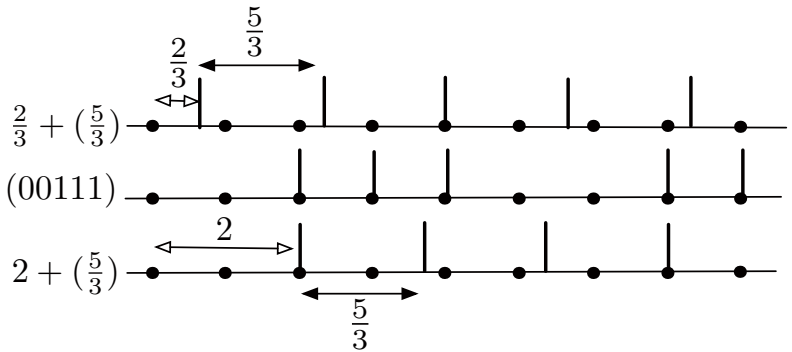
We approximate a period p by $[d, D] + (r)$.

r = rate of p
d,D = bounding of the ones indexes in p

Definition (abstraction)

$abs(p) = [d, D] + (r) \Leftrightarrow \forall i \geq 0, d + r \times i \leq [p]_i \leq D + r \times i$
with $d, D \in \mathbb{Q}$, d maximal and D minimal.

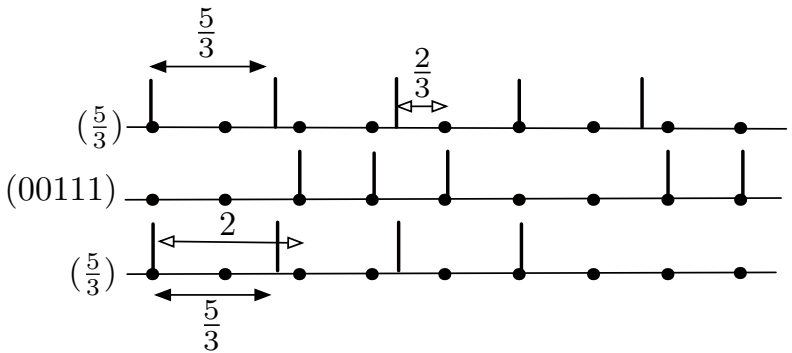
example: $abs\big((00111)\big) = [\frac{2}{3}, 2] + (\frac{5}{3})$



$\frac{2}{3} + (\frac{5}{3}) <: (00111) <: 2 + (\frac{5}{3})$

Abstraction of an infinite binary word

$$abs((00111)) = [\tfrac{2}{3}, 2] + (\tfrac{5}{3})$$



Concretization of an abstract period

Definition (concretization)

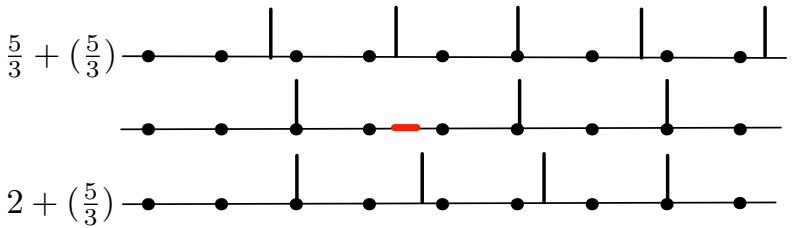
$concr([d, D] + (r)) = \{w, \forall i \geq 0,\ d + r \times i \leq [w]_i \leq D + r \times i\}$

"All words that come between d+(r) and D+(r)"

Infimum and supremum of the concretization set, if it's not empty:

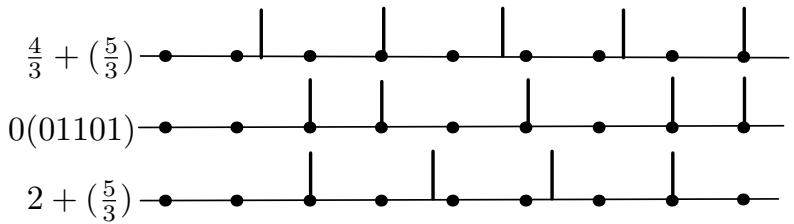
$$w_{inf} = inf(concr(pa)) \quad \Leftrightarrow \quad \forall i \geq 0, [w_{inf}]_i = \lceil d + r \times i \rceil$$
$$w_{sup} = sup(concr(pa)) \quad \Leftrightarrow \quad \forall i \geq 0, [w_{sup}]_i = \lfloor D + r \times i \rfloor$$

$D - d < 1 - \frac{1}{n} \Rightarrow concr\left([d, D] + (\frac{m}{n})\right) = \varnothing$



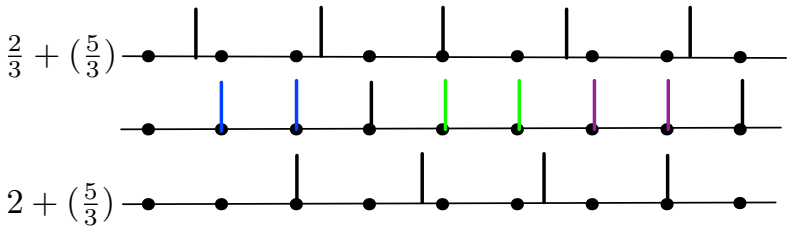
$concr\left([\frac{5}{3}, 2] + (5/3)\right) = \{\}$

$D - d = 1 - \frac{1}{n} \Rightarrow concr\big([d, D] + (\frac{m}{n})\big) = \{w\}$



$concr([\frac{4}{3}, 2] + (5/3)) = \{0(01101)\}$

$D - d > 1 - \frac{1}{n} \Rightarrow concr\left([d, D] + (\frac{m}{n})\right) = \{w_1, w_2, ..., w_k, ...\}$



$concr([\frac{2}{3}, 2] + (5/3)) = \{(00111),\ (01011),\ (010101011),\ ...\}$

Abstraction of the constraint system

We have : $C : \{\alpha_i \text{ on } p_i <: \alpha_j \text{ on } p_j\}_{i,j}$

$\alpha_i = base \text{ on } x_i$ and $\alpha_j = base \text{ on } x_j$.

We have to find x_i, x_j such that : $\{ x_i \text{ on } p_i <: x_j \text{ on } p_j \}_{i,j}$

We abstract :

$C^{\sim} : \{abs(x_i) \text{ on}^{\sim} abs(p_i) <:^{\sim} abs(x_j) \text{ on}^{\sim} abs(p_j)\}_{i,j}$

- $\text{on}^{\sim}$ operator ?
- $<:^{\sim}$ relation ?
- resolution of abstract constraints ?

$\text{on}^{\sim}$ operator

$$[d_1, D_1] + (r_1) \ \text{on}^{\sim} \ [d_2, D_2] + (r_2) = [d_{12}, D_{12}] + (r_1 \times r_2)$$

with :
$d_{12} = d_1 + d_2 \times r_1$
$D_{12} = D_1 + D_2 \times r_1$

property: $p_1 \ \text{on} \ p_2 \in concr(abs(p_1) \ \text{on}^{\sim} \ abs(p_2))$

Abstraction of a period:
$abs(w_1 \ \text{on} \ \ldots \ \text{on} \ w_k) = abs(w_1) \ \text{on}^{\sim} \ \ldots \ \text{on}^{\sim} \ abs(w_k)$

$<:^{\sim}$ relation

- Definition (synchronizability)

  $pa_1 \bowtie^{\sim} pa_2 \Leftrightarrow \forall p_1 \in concr(pa_1), p_2 \in concr(pa_2),\ p_1 \bowtie p_2$

  synchronizability test :
  $[d_1, D_1] + (r_1) \bowtie^{\sim} [d_2, D_2] + (r_2) \Leftrightarrow r_1 = r_2$

  property : $abs(p_1) \bowtie^{\sim} abs(p_2) \Leftrightarrow p_1 \bowtie p_2$

$<:^{\sim}$ relation

- Definition (precedence)

$pa_1 \preceq^{\sim} pa_2 \Leftrightarrow \forall p_1 \in concr(pa_1), p_2 \in concr(pa_2),\ p_1 \preceq p_2$

precedence test :

$$\begin{aligned} pa_1 \preceq^{\sim} pa_2 &\Leftrightarrow sup(concr(pa_1)) \preceq inf(concr(pa_2)) \\ &\Leftrightarrow \forall i \geq 0, \lfloor r_1 \times i + D_1 \rfloor \leq \lceil r_2 \times i + d_2 \rceil \end{aligned}$$

particular case :
if $r_1 = r_2 = \frac{m}{n}$ then $pa_1 \preceq^{\sim} pa_2 \Leftrightarrow D_1 - d_2 \leq 1 - \frac{1}{n}$

property : $abs(p_1) \preceq^{\sim} abs(p_2) \Rightarrow p_1 \preceq p_2$
$Sat(C^{\sim}) \Rightarrow Sat(C)$

Contraints resolution

$C^{\sim} : \{abs(x_i)\ \text{on}^{\sim}\ abs(p_i) <:^{\sim} abs(x_j)\ \text{on}^{\sim}\ abs(p_j)\}_{i,j}$
$C^{\sim} : \{xa_i\ \text{on}^{\sim}\ pa_i <:^{\sim} xa_j\ \text{on}^{\sim}\ pa_j\}_{i,j}$ with $\{concr(xa_k \neq \varnothing\}_k$

after the computation of the $\text{on}^{\sim}$:
$C^{\sim} : \{(r_i) + [d_i, D_i] <:^{\sim} (r_j) + [d_j, D_j]\}_{i,j}$

- a set of constraints on rates $C^{\sim}_{rates} : \{r_i = r_j\}_{i,j}$
- a set of constraints on delays
  $C^{\sim}_{delays} : \{D_i - d_j \leq c_{ij}\}_{i,j} + \{D_k - d_k \leq c_k\}$

Conclusion

- All valid systems without cycles are accepted
- Some valid systems with cycles are rejected.

\+ priority to the bufferization
(we always synchronize all the clocks on the slower one)

\+ dealing with less regular clocks
Type checking guarantee: " if the clocks correspond to their specification, then the program is n-synchronous"