Typing of periodic clocks in Lucy-n

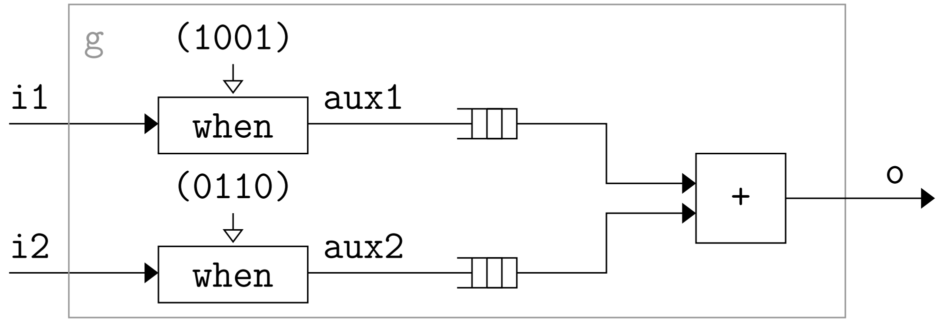
Louis Mandel   Florence Plateau

Équipe Parkas

École Normale Supérieure
Université Paris-Sud 11
INRIA

Synchron 2010

Lucy-n = Lustre + buffers

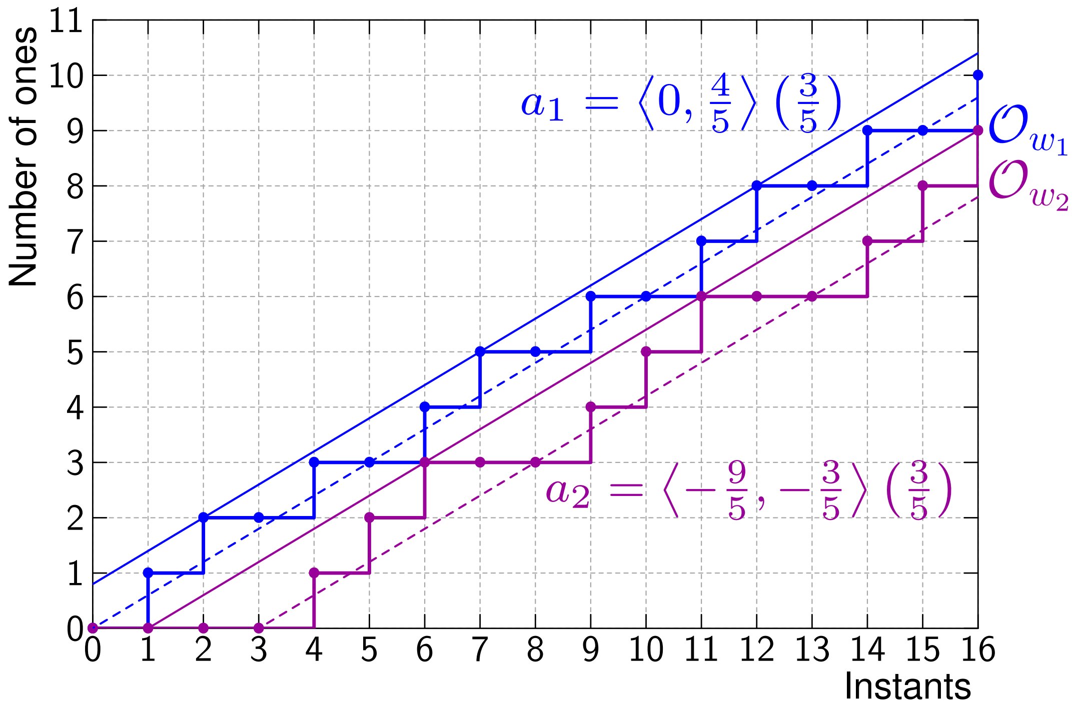



```
let node g (i1, i2) = o where
  rec aux1 = i1 when (1001)
  and aux2 = i2 when (0110)
  and o = buffer aux1 + buffer aux2
val g :: forall 'a. ('a * 'a) -> 'a on 0(10)
Buffer line 4, characters 10-21: size = 1
Buffer line 4, characters 24-35: size = 1
```

Clock calculus to automatically compute

- activation rhythms of nodes (schedules)
- buffers sizes needed for these schedules

Abstract Clocks



Advantages:

- efficient algorithm
- deals with not exactly periodic clocks

Disadvantage:

- over approximation of buffer sizes
- reject correct programs

$\Rightarrow$ an algorithm without abstraction is useful in certain cases

Overview

1. Algebraic properties of ultimately periodic binary words
2. Typing of n-synchronous programs
3. Discussion

Clocks

$$\xrightarrow[w]{\texttt{x}}$$

x	2	5		3		7	9	4			6	...
$w = clock(\texttt{x})$	1	1	0	1	0	1	1	1	0	0	1	...

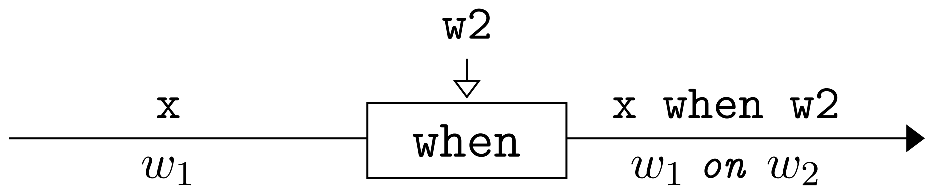
Ultimately periodic binary words

- definition: $p = u(v) \quad \overset{def}{\Leftrightarrow} \quad p = uw$ avec $w = vw$
- notation: prefix $= p.u$ and periodic part $= p.v$
- example: 1101(11100110) $=$ 1101 11100110 11100110 11100110 ...

Index of the j^{th} 1 of w

- notation: $\mathcal{I}_w(j)$
- example: $\mathcal{I}_w(4) = 6$

Sampling



x	2	5	3	7	9	...
w2	1	0	1	1	0	...
x when w2	2		3	7		...

w_1	1	1	0	1	0	1	1	...
w_2	1	0		1		1	0	...
w_1 *on* w_2	1	0	0	1	0	1	0	...

Definition:

$$0w_1 \textit{ on } w_2 \overset{def}{=} 0(w_1 \textit{ on } w_2)$$
$$1w_1 \textit{ on } 1w_2 \overset{def}{=} 1(w_1 \textit{ on } w_2)$$
$$1w_1 \textit{ on } 0w_2 \overset{def}{=} 0(w_1 \textit{ on } w_2)$$

on operator

Example:

p_1	1	1	0	1	(	1	1	1	0	0	1	1	0	)
p_2	1	0		1	(	1	0	0			1	0		)
$p_1 \ \mathit{on}\ p_2$	1	0	0	1	(	1	0	0	0	0	1	0	0	)

Properties:

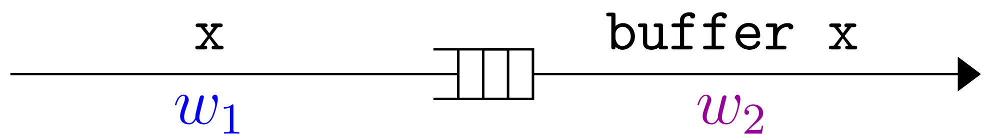
- size and number of 1:
  Let p_1 and p_2 such that $|p_1.u|_1 = |p_2.u|$ and $|p_1.v|_1 = |p_2.v|$. Then:

$$|(p_1 \ \mathit{on}\ p_2).u| = |p_1.u| \qquad |(p_1 \ \mathit{on}\ p_2).u|_1 = |p_2.u|_1$$
$$|(p_1 \ \mathit{on}\ p_2).v| = |p_1.v| \qquad |(p_1 \ \mathit{on}\ p_2).v|_1 = |p_2.v|_1$$

- index of the j^{th} 1 of $w_1 \ \mathit{on}\ w_2$:

$$\forall j \geq 1,\ \mathcal{I}_{w_1 \ \mathit{on}\ w_2}(j) = \mathcal{I}_{w_1}(\mathcal{I}_{w_2}(j))$$

Buffering



Communication through a bounded buffer:

- synchronizability test:

$$p_1 \bowtie p_2 \quad \Leftrightarrow \quad \frac{|p_1.v|_1}{|p_1.v|} = \frac{|p_2.v|_1}{|p_2.v|}$$

- precedence test: let $h = \max(|p_1.u|_1, |p_2.u|_1) + \text{ ppcm } (|p_1.v|_1, |p_2.v|_1)$,

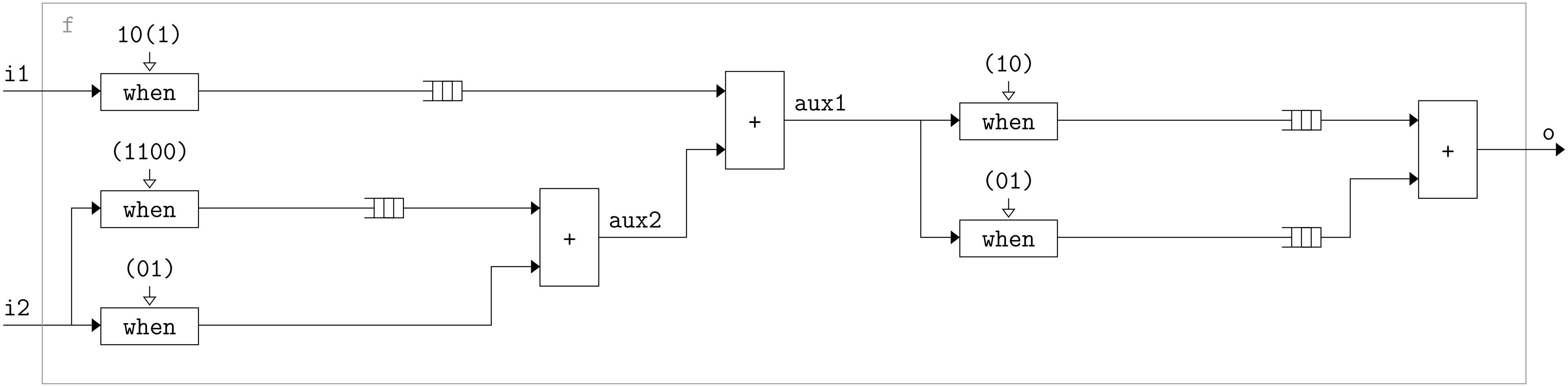
$$p_1 \preceq p_2 \quad \Leftrightarrow \quad \forall j,\ 1 \leq j \leq h,\ \mathcal{I}_{p_1}(j) \leq \mathcal{I}_{p_2}(j)$$

- adaptability test: $p_1 <: p_2 \quad \Leftrightarrow \quad p_1 \bowtie p_2 \quad \wedge \quad p_1 \preceq p_2$

Examples:

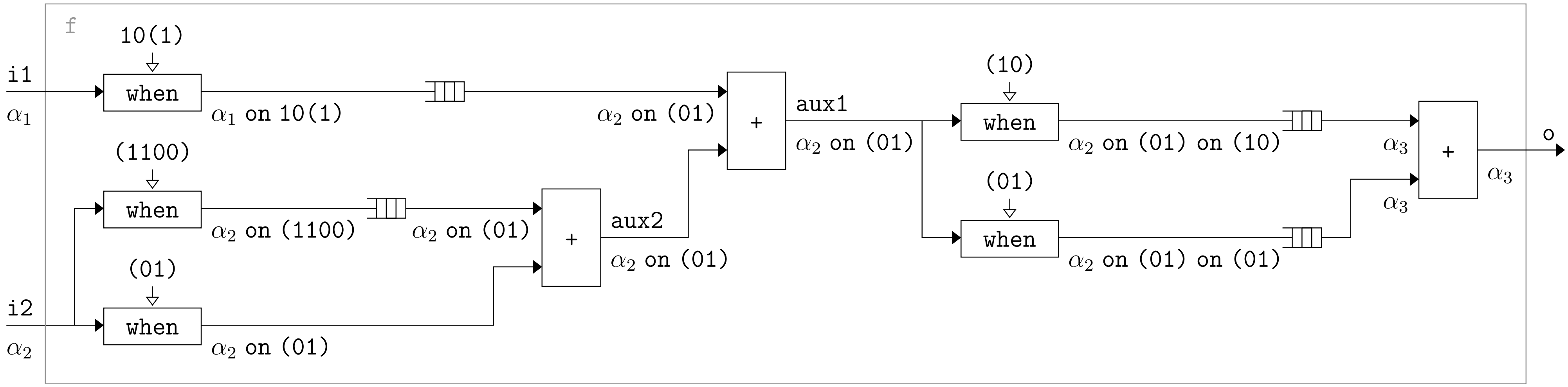
- synchronizability test: $(11010) \bowtie 0(00111)$
- precedence test: $(11010) \preceq 0(00111)$

Example of Lucy-n program




```
let node f (i1, i2) = o where
  rec aux1 = buffer (i1 when 10(1)) + aux2
  and aux2 = buffer (i2 when (1100)) + i2 when (01)
  and o = buffer (aux1 when (10)) + buffer (aux1 when (01))
```

Typing

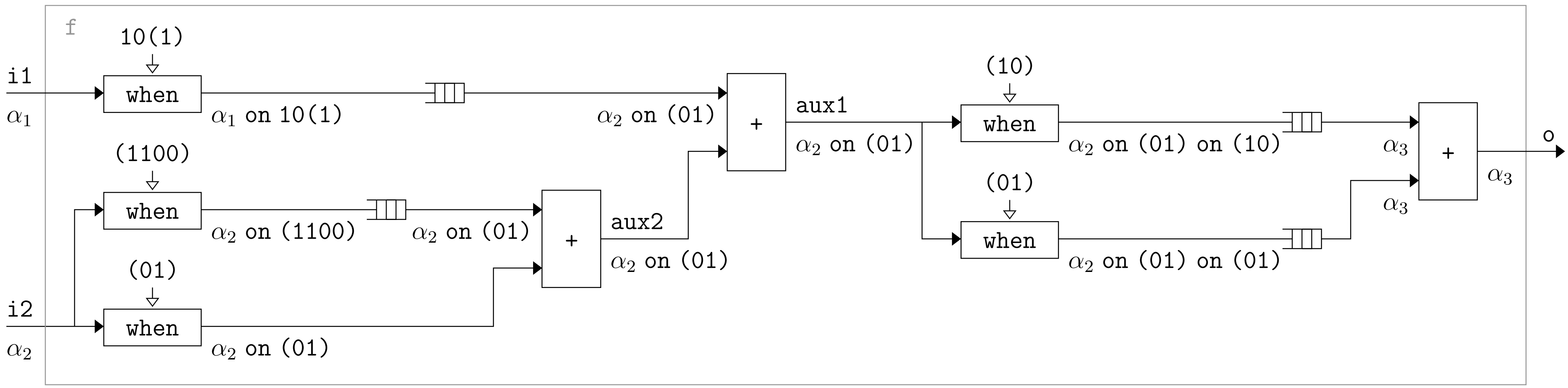


f :: $\alpha_1 \times \alpha_2 \rightarrow \alpha_3$ with the following constraints:

$$\left\{ \begin{array}{rcl} \alpha_1 \text{ on } 10(1) & <: & \alpha_2 \text{ on } (01) \\ \alpha_2 \text{ on } (1100) & <: & \alpha_2 \text{ on } (01) \\ \alpha_2 \text{ on } (01) \text{ on } (10) & <: & \alpha_3 \text{ on } (1) \\ \alpha_2 \text{ on } (01) \text{ on } (01) & <: & \alpha_3 \text{ on } (1) \end{array} \right\}$$

Question: find types α_1, α_2 and α_3
such that the constraints are always satisfied.

Typing



f :: $\alpha_1 \times \alpha_2 \rightarrow \alpha_3$ with the following constraints:

$$\left\{ \begin{array}{rcl} \alpha_1 \texttt{ on } 10(1) & <: & \alpha_2 \texttt{ on } (01) \\ \alpha_2 \texttt{ on } (1100) & <: & \alpha_2 \texttt{ on } (01) \\ \alpha_2 \texttt{ on } (01) \texttt{ on } (10) & <: & \alpha_3 \texttt{ on } (1) \\ \alpha_2 \texttt{ on } (01) \texttt{ on } (01) & <: & \alpha_3 \texttt{ on } (1) \end{array} \right\}$$

We can simplify constraints that depends on the same type variable

- Property: $\alpha \texttt{ on } ce_1 <: \alpha \texttt{ on } ce_2 \quad \Leftrightarrow \quad ce_1 <: ce_2$

$$\left\{\begin{array}{rcl} \alpha_1 \texttt{ on } \texttt{10(1)} & <: & \alpha_2 \texttt{ on } \texttt{(01)} \\ \texttt{(1100)} & <: & \texttt{(01)} \\ \alpha_2 \texttt{ on } \texttt{(01)} \texttt{ on } \texttt{(10)} & <: & \alpha_3 \texttt{ on } \texttt{(1)} \\ \alpha_2 \texttt{ on } \texttt{(01)} \texttt{ on } \texttt{(01)} & <: & \alpha_3 \texttt{ on } \texttt{(1)} \end{array}\right\}$$

We can check that the adaptability constraint is satisfied.

$$\left\{\begin{array}{rcl} \alpha_1 \texttt{ on } \texttt{10(1)} & <: & \alpha_2 \texttt{ on } \texttt{(01)} \\ & & \\ \alpha_2 \texttt{ on } \texttt{(01)} \texttt{ on } \texttt{(10)} & <: & \alpha_3 \texttt{ on } \texttt{(1)} \\ \alpha_2 \texttt{ on } \texttt{(01)} \texttt{ on } \texttt{(01)} & <: & \alpha_3 \texttt{ on } \texttt{(1)} \end{array}\right\}$$

We can express this system in function of a unique type variable by instantiation of α_1, α_2, α_3

$$\theta = \{\alpha_1 \leftarrow \alpha \texttt{ on } c_1;\ \alpha_2 \leftarrow \alpha \texttt{ on } c_2;\ \alpha_3 \leftarrow \alpha \texttt{ on } c_3;\}$$

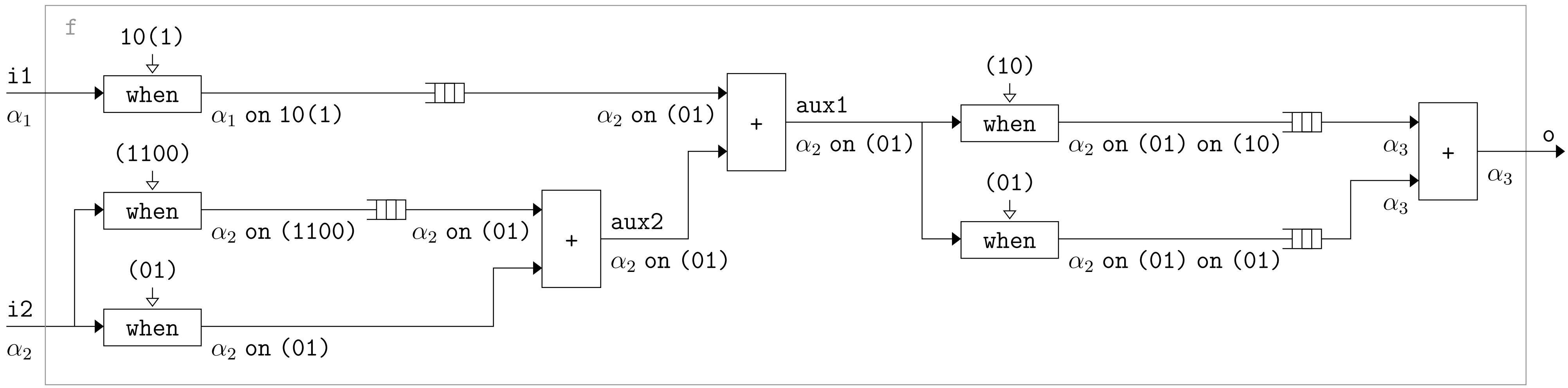
$$\left\{ \begin{array}{rcl} \alpha \texttt{ on } c_1 \texttt{ on } \texttt{10(1)} & <: & \alpha \texttt{ on } c_2 \texttt{ on } \texttt{(01)} \\ \\ \alpha \texttt{ on } c_2 \texttt{ on } \texttt{(01)} \texttt{ on } \texttt{(10)} & <: & \alpha \texttt{ on } c_3 \texttt{ on } \texttt{(1)} \\ \alpha \texttt{ on } c_2 \texttt{ on } \texttt{(01)} \texttt{ on } \texttt{(01)} & <: & \alpha \texttt{ on } c_3 \texttt{ on } \texttt{(1)} \end{array} \right\}$$

We can simplify the constraints

$$\left\{ \begin{array}{rcl} c_1 \ \textit{on} \ \texttt{10(1)} & <: & c_2 \ \textit{on} \ \texttt{(01)} \\ \\ c_2 \ \textit{on} \ \texttt{(01)} \ \textit{on} \ \texttt{(10)} & <: & c_3 \ \textit{on} \ \texttt{(1)} \\ c_2 \ \textit{on} \ \texttt{(01)} \ \textit{on} \ \texttt{(01)} & <: & c_3 \ \textit{on} \ \texttt{(1)} \end{array} \right\}$$

Question: find ultimately periodic binary words c_1, c_2 and c_3 such that the constraints are always satisfied.

Typing



f :: α on $c_1 \times \alpha$ on $c_2 \rightarrow \alpha$ on c_3 with the following constraints:

$$\left\{ \begin{array}{rcl} c_1 \; on \; 10(1) & <: & c_2 \; on \; (01) \\ \\ c_2 \; on \; (01) \; on \; (10) & <: & c_3 \; on \; (1) \\ c_2 \; on \; (01) \; on \; (01) & <: & c_3 \; on \; (1) \end{array} \right\}$$

We can compute *on*.

$$\left\{\begin{array}{rcl} c_1 \; \textit{on} \; \texttt{10(1)} & <: & c_2 \; \textit{on} \; \texttt{(01)} \\ \\ c_2 \; \textit{on} \; \texttt{(0100)} & <: & c_3 \; \textit{on} \; \texttt{(1)} \\ c_2 \; \textit{on} \; \texttt{(0001)} & <: & c_3 \; \textit{on} \; \texttt{(1)} \end{array}\right\}$$

We can adjust the system such that all the samplers of a same variable have the same size.

$$\left\{\begin{array}{rcl} c_1 \; \textit{on} \; \texttt{10(1)} & <: & c_2 \; \textit{on} \; \texttt{(0101)} \\ \\ c_2 \; \textit{on} \; \texttt{(0100)} & <: & c_3 \; \textit{on} \; \texttt{(1)} \\ c_2 \; \textit{on} \; \texttt{(0001)} & <: & c_3 \; \textit{on} \; \texttt{(1)} \end{array}\right\}$$

We **choose** the number of 1 of each c_n such that it is equal to the size of its samplers.

$$|c_1.u|_1 = 2 \quad |c_1.v|_1 = 1 \qquad |c_2.u|_1 = 0 \quad |c_2.v|_1 = 4 \qquad |c_3.u|_1 = 0 \quad |c_3.v|_1 = 1$$

We can split adaptability constraints into synchronizability and precedence constraints.

$$\left\{\begin{array}{c} c_1 \ \mathtt{on}\ \mathtt{10(1)} \bowtie c_2 \ \mathtt{on}\ \mathtt{(0101)} \\ \\ c_2 \ \mathtt{on}\ \mathtt{(0100)} \bowtie c_3 \ \mathtt{on}\ \mathtt{(1)} \\ c_2 \ \mathtt{on}\ \mathtt{(0001)} \bowtie c_3 \ \mathtt{on}\ \mathtt{(1)} \end{array}\right\} \wedge \left\{\begin{array}{c} c_1 \ \mathtt{on}\ \mathtt{10(1)} \preceq c_2 \ \mathtt{on}\ \mathtt{(0101)} \\ \\ c_2 \ \mathtt{on}\ \mathtt{(0100)} \preceq c_3 \ \mathtt{on}\ \mathtt{(1)} \\ c_2 \ \mathtt{on}\ \mathtt{(0001)} \preceq c_3 \ \mathtt{on}\ \mathtt{(1)} \end{array}\right\}$$

$$(Sync) \qquad\qquad (Prec)$$

We can apply the synchronizability test.

$$\left\{\begin{array}{c} \frac{|(c_1 \ \mathtt{on}\ \mathtt{10(1)}).v|_1}{|(c_1 \ \mathtt{on}\ \mathtt{10(1)}).v|} = \frac{|(c_2 \ \mathtt{on}\ \mathtt{(0101)}).v|_1}{|(c_2 \ \mathtt{on}\ \mathtt{(0101)}).v|} \\ \\ \frac{|(c_2 \ \mathtt{on}\ \mathtt{(0100)}).v|_1}{|(c_2 \ \mathtt{on}\ \mathtt{(0100)}).v|} = \frac{|(c_3 \ \mathtt{on}\ \mathtt{(1)}).v|_1}{|(c_3 \ \mathtt{on}\ \mathtt{(1)}).v|} \\ \frac{|(c_2 \ \mathtt{on}\ \mathtt{(0001)}).v|_1}{|(c_2 \ \mathtt{on}\ \mathtt{(0001)}).v|} = \frac{|(c_3 \ \mathtt{on}\ \mathtt{(1)}).v|_1}{|(c_3 \ \mathtt{on}\ \mathtt{(1)}).v|} \end{array}\right\} \wedge (Prec)$$

Thanks to the choice of the number of 1 of the c_n,
we can simplify the formulas.

$$\left\{\begin{array}{c} \dfrac{|\texttt{10(1)}.v|_1}{|c_1.v|} = \dfrac{|\texttt{(0101)}.v|_1}{|c_2.v|} \\ \\ \dfrac{|\texttt{(0100)}.v|_1}{|c_2.v|} = \dfrac{|\texttt{(1)}.v|_1}{|c_3.v|} \\ \dfrac{|\texttt{(0001)}.v|_1}{|c_2.v|} = \dfrac{|\texttt{(1)}.v|_1}{|c_3.v|} \end{array}\right\} \wedge (Prec)$$

We can rewrite the system.

$$\left\{\begin{array}{c} |\texttt{(0101)}.v|_1 \times |c_1.v| = |(\texttt{10(1)}).v|_1 \times |c_2.v| \\ \\ |\texttt{(1)}.v|_1 \times |c_2.v| = |\texttt{(0100)}.v|_1 \times |c_3.v| \\ |\texttt{(1)}.v|_1 \times |c_2.v| = |\texttt{(0001)}.v|_1 \times |c_3.v| \end{array}\right\} \wedge (Prec)$$

We can compute the number of 1 of the samplers.

$$\left\{\begin{array}{c} 2 \times |c_1.v| = |c_2.v| \\ \\ |c_2.v| = |c_3.v| \\ |c_2.v| = |c_3.v| \end{array}\right\} \wedge \left\{\begin{array}{c} c_1 \; \mathit{on} \; \mathtt{10(1)} \preceq c_2 \; \mathit{on} \; \mathtt{(0101)} \\ \\ c_2 \; \mathit{on} \; \mathtt{(0100)} \preceq c_3 \; \mathit{on} \; \mathtt{(1)} \\ c_2 \; \mathit{on} \; \mathtt{(0001)} \preceq c_3 \; \mathit{on} \; \mathtt{(1)} \end{array}\right\}$$

Thanks to the choice of the number of 1 of the c_n,
we can apply the precedence test.

$$(Sync) \wedge \left\{\begin{array}{l} \forall j,\ 1 \leq j \leq 3, \quad \mathcal{I}_{c_1 \; \mathit{on} \; \mathtt{10(1)}}(j) \leq \mathcal{I}_{c_2 \; \mathit{on} \; \mathtt{(0101)}}(j) \\ \\ \forall j,\ 1 \leq j \leq 1, \quad \mathcal{I}_{c_2 \; \mathit{on} \; \mathtt{(0100)}}(j) \leq \mathcal{I}_{c_3 \; \mathit{on} \; \mathtt{(1)}}(j) \\ \forall j,\ 1 \leq j \leq 1, \quad \mathcal{I}_{c_2 \; \mathit{on} \; \mathtt{(0001)}}(j) \leq \mathcal{I}_{c_3 \; \mathit{on} \; \mathtt{(1)}}(j) \end{array}\right\}$$

We can apply the *on* formula.

$$(Sync) \wedge \left\{ \begin{array}{l} \forall j,\ 1 \leq j \leq 3, \quad \mathcal{I}_{c_1}(\mathcal{I}_{\texttt{10(1)}}(j)) \leq \mathcal{I}_{c_2}(\mathcal{I}_{\texttt{(0101)}}(j)) \\ \\ \forall j,\ 1 \leq j \leq 1, \ \ \mathcal{I}_{c_2}(\mathcal{I}_{\texttt{(0100)}}(j)) \leq \mathcal{I}_{c_3}(\mathcal{I}_{\texttt{(1)}}(j)) \\ \forall j,\ 1 \leq j \leq 1, \ \ \mathcal{I}_{c_2}(\mathcal{I}_{\texttt{(0001)}}(j)) \leq \mathcal{I}_{c_3}(\mathcal{I}_{\texttt{(1)}}(j)) \end{array} \right\}$$

We can compute the index of the 1s in the periodic words.

$$\left\{ \begin{array}{r} 2 \times |c_1.v| = |c_2.v| \\ \\ |c_2.v| = |c_3.v| \\ |c_2.v| = |c_3.v| \end{array} \right\} \wedge \left\{ \begin{array}{l} \mathcal{I}_{c_1}(1) \leq \mathcal{I}_{c_2}(2) \\ \mathcal{I}_{c_1}(3) \leq \mathcal{I}_{c_2}(4) \\ \mathcal{I}_{c_1}(4) \leq \mathcal{I}_{c_2}(6) \\ \mathcal{I}_{c_2}(2) \leq \mathcal{I}_{c_3}(1) \\ \mathcal{I}_{c_2}(4) \leq \mathcal{I}_{c_3}(1) \end{array} \right\}$$

Question: find the sizes and the index of 1
such that the constraints are always satisfied
and they define well formed ultimately periodic words.

Well formed ultimately periodic binary words

ultimately periodic word	1	1	0	(1	0	1	1	0)				
infinite word	1	1	0	1	0	1	1	0	1	0	1	...
index	1	2		4		6	7		9		11	...

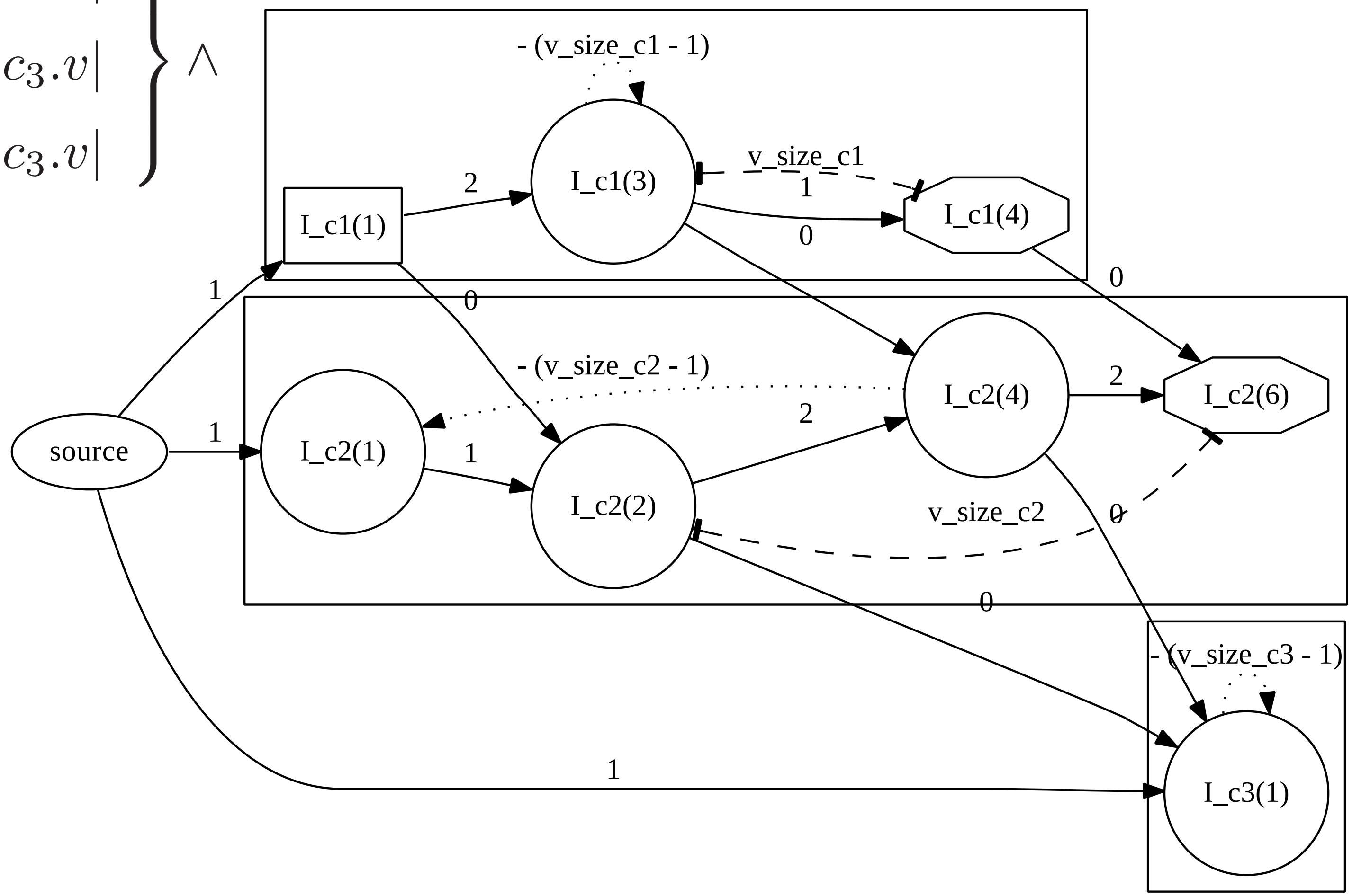
Well formation constraints:

- increasing indexes: $\forall j \geq 1,\ \mathcal{I}_w(j) < \mathcal{I}_w(j+1)$
- sufficient indexes: $\forall j \geq 1,\ \mathcal{I}_w(j) \geq j$
- periodicity: $\forall j > |p.u|_1,\ \mathcal{I}_p(j + |p.v|_1) = \mathcal{I}_p(j) + |p.v|$
- sufficient size: $|p.v| \geq 1 + \mathcal{I}_p(|p.u|_1 + |p.v|_1) - \mathcal{I}_p(|p.u|_1 + 1)$

Typing

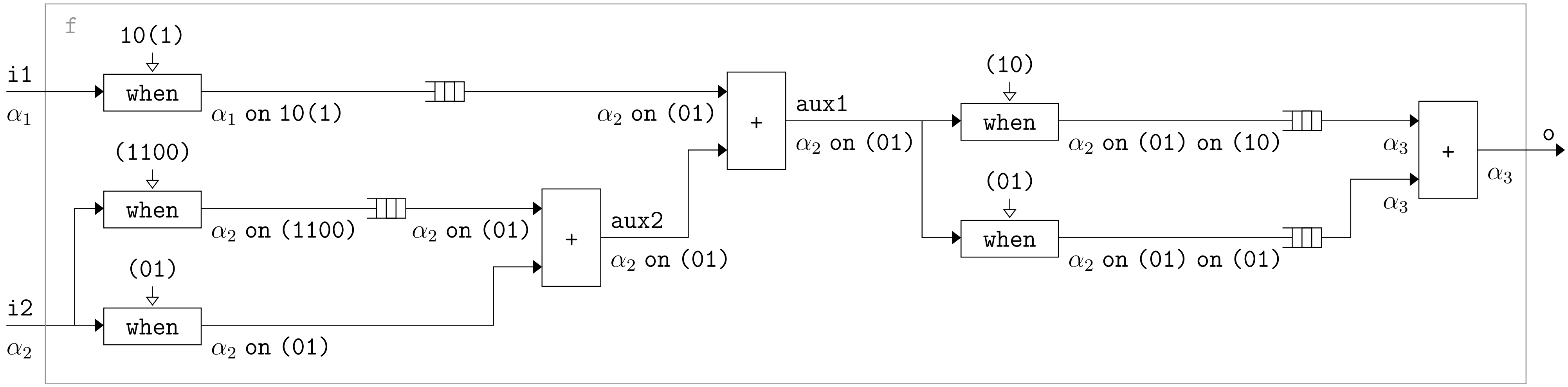
$\mathtt{f}\ ::\ \alpha\ \mathtt{on}\ c_1 \times \alpha\ \mathtt{on}\ c_2 \rightarrow \alpha\ \mathtt{on}\ c_3$ with the following constraints:

$$\left\{ \begin{array}{r} 2 \times |c_1.v| = |c_2.v| \\ |c_2.v| = |c_3.v| \\ |c_2.v| = |c_3.v| \end{array} \right\} \wedge$$



We can solve the constraints using GLPK.

Typing



f :: α on $c_1 \times \alpha$ on $c_2 \rightarrow \alpha$ on c_3 with the following constraints:

$$|c_1.v| = 2 \qquad |c_2.v| = 4 \qquad |c_3.v| = 4$$

$$\mathcal{I}_{c_1}(1) = 1 \quad \mathcal{I}_{c_1}(3) = 3 \quad \mathcal{I}_{c_1}(4) = 5$$

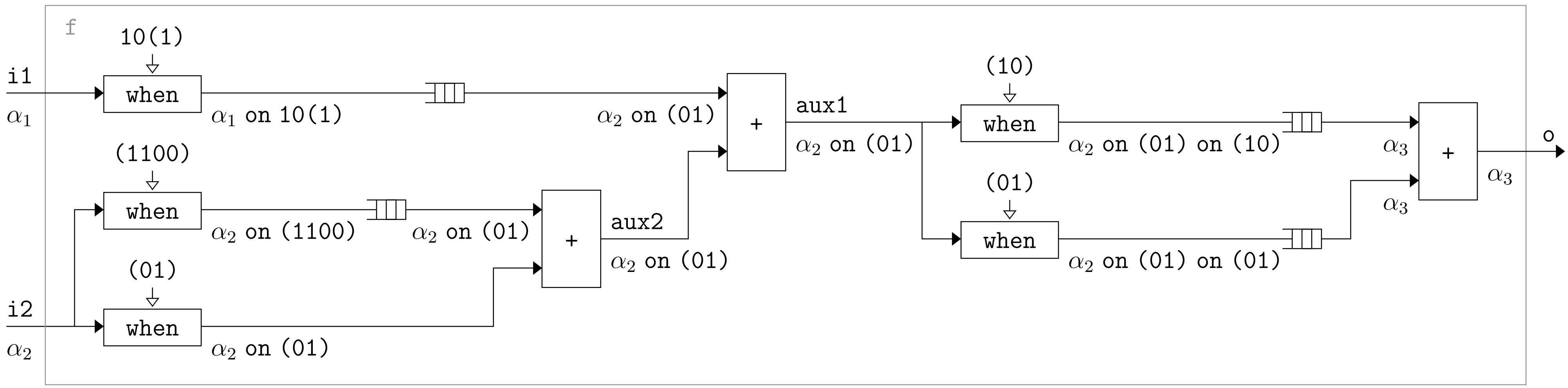
$$\mathcal{I}_{c_2}(1) = 1 \quad \mathcal{I}_{c_2}(2) = 2 \quad \mathcal{I}_{c_2}(4) = 4 \quad \mathcal{I}_{c_2}(6) = 6$$

$$\mathcal{I}_{c_3}(1) = 4$$

We can build the following solution:

$$c_1 = \mathtt{11(10)} \qquad c_2 = \mathtt{(1111)} = \mathtt{(1)} \qquad c_3 = \mathtt{000(1000)} = \mathtt{(0^3 1)}$$

Typing




```
let node f (i1, i2) = o where
  rec aux1 = buffer (i1 when 10(1)) + aux2
  and aux2 = buffer (i2 when (1100)) + i2 when (01)
  and o = buffer (aux1 when (10)) + buffer (aux1 when (01))
val f :: forall 'a. ('a on 11(10) * 'a) -> 'a on 0^3(10^3)
Buffer line 2, characters 13-35: size = 1
Buffer line 3, characters 13-36: size = 1
Buffer line 4, characters 10-33: size = 1
Buffer line 4, characters 36-59: size = 0
```

Comparison with the resolution using abstraction

Notation:

- abstract resolution = resolution algorithm that abstract clocks
- concrete resolution = resolution algorithm that does not abstract clocks

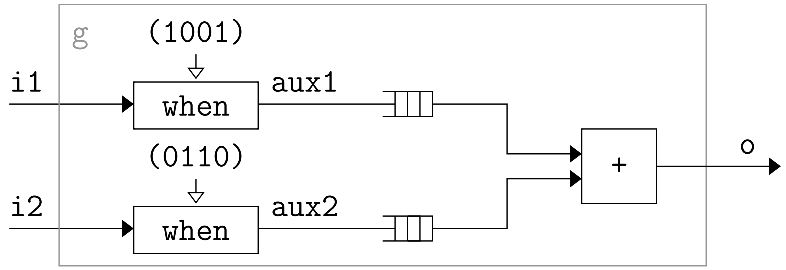
Advantages of the abstract resolution:

- much more efficient
- able to deal with not exactly periodic clocks

Advantages of the concrete resolution:

- more programs are accepted
- more precise buffer sizes
- better schedules

Throughput v.s. buffering




```
let node g (i1, i2) = o where
  rec aux1 = i1 when (1001)
  and aux2 = i2 when (0110)
  and o = buffer aux1 + buffer aux2
```

With bufferization:

```
  val g :: forall 'a. ('a * 'a) -> 'a on 0(10)
  Buffer line 4, characters 10-21: size = 1
  Buffer line 4, characters 24-35: size = 1
```

Without bufferization:

```
  val g :: forall 'a. ('a on 0(1^4 00) * 'a on (110011)) -> 'a on 0(100)
  Buffer line 4, characters 10-21: size = 0
  Buffer line 4, characters 24-35: size = 0
```

Objective function

It is possible to choose the objective function when the linear constraints are solves

- ASAP: minimize the index of 1s
- rate: minimize the sizes
- buffer: minimize the precedence constraints

Conclusion

New algorithm to type n-synchronous programs with periodic clocks.

Completeness depend only on the choice of the number of 1s in the solution.

Handles to choose the solution: buffering vs throughput.

Accepted to JFLA 2011:

`http://www.lri.fr/~plateau/jfla11`

Reminder

- size and number of 1:
  Let p_1 and p_2 such that $|p_1.u|_1 = |p_2.u|$ and $|p_1.v|_1 = |p_2.v|$. Then:

$$\begin{array}{rclcrcl} |(p_1 \ \textit{on} \ p_2).u| & = & |p_1.u| & \quad & |(p_1 \ \textit{on} \ p_2).u|_1 & = & |p_2.u|_1 \\ |(p_1 \ \textit{on} \ p_2).v| & = & |p_1.v| & \quad & |(p_1 \ \textit{on} \ p_2).v|_1 & = & |p_2.v|_1 \end{array}$$

- index of the j^{th} 1 of $w_1 \ \textit{on} \ w_2$:

$$\forall j \geq 1, \ \mathcal{I}_{w_1 \ \textit{on} \ w_2}(j) = \mathcal{I}_{w_1}(\mathcal{I}_{w_2}(j))$$

- synchronizability test:

$$p_1 \bowtie p_2 \qquad \Leftrightarrow \qquad \frac{|p_1.v|_1}{|p_1.v|} = \frac{|p_2.v|_1}{|p_2.v|}$$

- precedence test: let $h = \max(|p_1.u|_1, |p_2.u|_1) + \text{ ppcm } (|p_1.v|_1, |p_2.v|_1)$,

$$p_1 \preceq p_2 \qquad \Leftrightarrow \qquad \forall j, \ 1 \leq j \leq h, \ \mathcal{I}_{p_1}(j) \leq \mathcal{I}_{p_2}(j)$$

- adaptability test: $\qquad p_1 \ <: \ p_2 \qquad \Leftrightarrow \qquad p_1 \bowtie p_2 \quad \wedge \quad p_1 \preceq p_2$