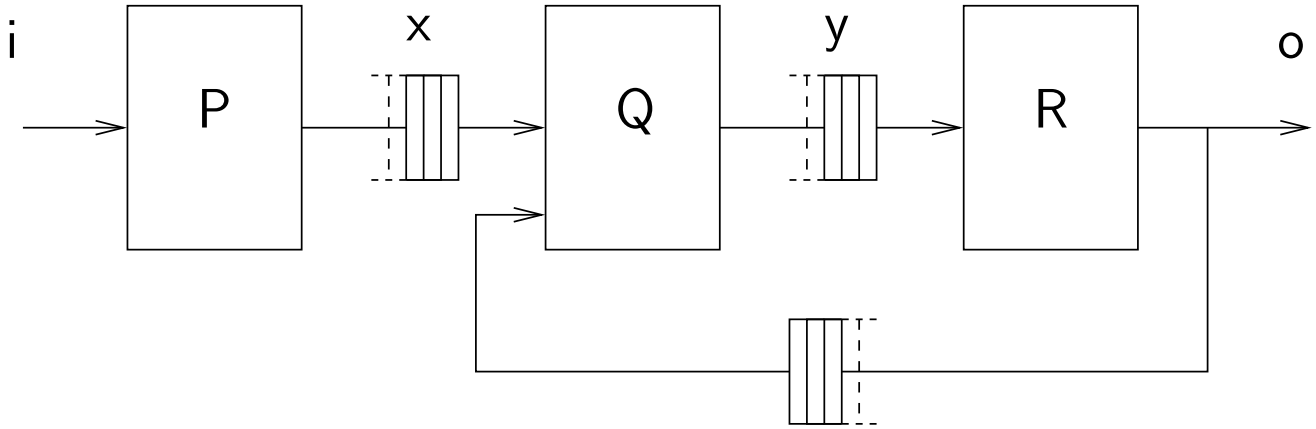
# Lucy-n : une extension n-synchrone de Lustre

Louis Mandel Florence Plateau Marc Pouzet
{mandel,plateau,pouzet}@lri.fr

Laboratoire de Recherche en Informatique, équipe Proval
Université Paris-Sud 11

JFLA – 02/02/2010

# Réseaux de Kahn [Gilles Kahn, 1974]




Si l'on dispose de buffers de taille suffisante, et si les programmes sont déterministes, alors le réseau est déterministe.

# Programmer des réseaux de Kahn

Problème : calculer les tailles suffisantes pour les buffers

- Risque de perte de données, de blocages
- Parfois, nécessité de buffers infinis

Objectif :

- Rejet des réseaux à buffers infinis
- Dimensionnement automatique des buffers

Travaux connexes :

- Synchronous Data Flow et variantes [Lee *et al.*] [Buck]
- Ordonnancement [Carlier, Chretienne] [Baccelli, Cohen, Quadrat]
- Network Calculus [Cruz], Realtime Calculus [Thiele *et al.*]

# Le modèle synchrone flot de données

Programmation de réseaux de Kahn sans buffers :

- Consommation instantanée des données produites
- Langages de programmation Lustre, Signal, Lucid Synchrone
- Garanties fortes : mémoire bornée, absence de blocage

Mais : communication sans buffers parfois trop restrictive
(e.g. applications multimédia)

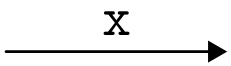
# Le modèle n-synchrone : programmation de réseaux de Kahn à mémoire bornée

Méthodes automatiques à la compilation pour

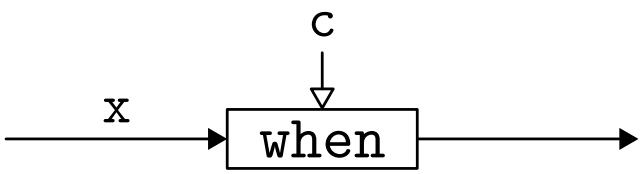
- Accepter de stocker les données dans des buffers
- Rejeter les réseaux qui nécessitent une mémoire infinie
- Calculer les rythmes d'activation des différents nœuds de calcul
- Calculer les tailles des buffers nécessaires

Plus de flexibilité, autant de garanties

# Un noyau synchrone flot de données




| flot | valeurs | horloge |
|---|---|---|
| x | 5 7 3 6 2 8 1 ... | 1111111... |




| flot | valeurs | horloge |
|---|---|---|
| x | 5 7 3 6 2 8 1 ... | 1111111... |
| c | 1 0 1 0 1 0 1 ... | |
| x when c | 5 3 2 1 ... | 1010101... |

$$\mathit{horloge}(\texttt{x when c}) = \mathit{horloge}(\texttt{x})\ \mathit{on}\ \texttt{c}$$

Opérateur *on* :

$$\begin{aligned} 0.w_1\ \mathit{on}\ w_2 &\overset{\mathit{def}}{=} 0.(w_1\ \mathit{on}\ w_2) \\ 1.w_1\ \mathit{on}\ 1.w_2 &\overset{\mathit{def}}{=} 1.(w_1\ \mathit{on}\ w_2) \\ 1.w_1\ \mathit{on}\ 0.w_2 &\overset{\mathit{def}}{=} 0.(w_1\ \mathit{on}\ w_2) \end{aligned}$$


| flot | valeurs | horloge |
|---|---|---|
| x | 5 7 3 6 2 8 1 ... | 1111111... |
| c | 1 0 1 0 1 0 1 ... | |
| x when c | 5 3 2 1 ... | 1010101... |
| c' | 1 0 1 1 ... | |
| (x when c) when c' | 5 2 1 ... | 1000101... |

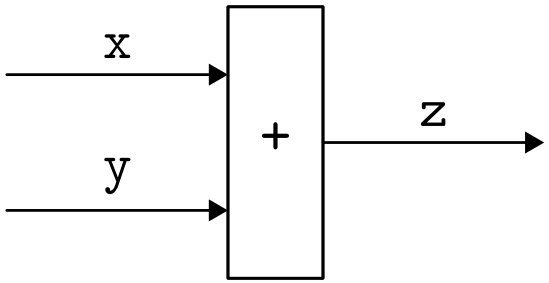
$$\mathit{horloge}(\texttt{(x when c) when c'}) = \mathit{horloge}(\texttt{x when c})\ \mathit{on}\ \texttt{c'}$$

Opérateur *on* :

$$\begin{aligned} 0.w_1\ \mathit{on}\ w_2 &\stackrel{\mathit{def}}{=} 0.(w_1\ \mathit{on}\ w_2) \\ 1.w_1\ \mathit{on}\ 1.w_2 &\stackrel{\mathit{def}}{=} 1.(w_1\ \mathit{on}\ w_2) \\ 1.w_1\ \mathit{on}\ 0.w_2 &\stackrel{\mathit{def}}{=} 0.(w_1\ \mathit{on}\ w_2) \end{aligned}$$

# Un noyau synchrone flot de données

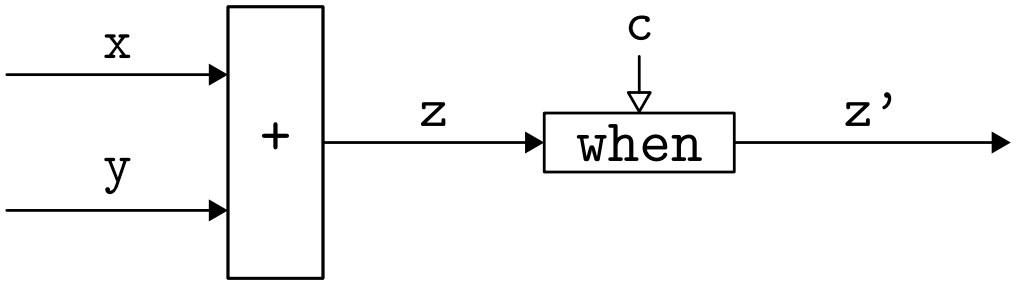





| flot | valeurs | | | | | | | | horloge |
|---|---|---|---|---|---|---|---|---|---|
| x | 5 | 7 | 3 | 6 | 2 | 8 | 1 | ... | 111111... |
| y | 3 | 2 | 1 | 5 | 4 | 1 | 7 | ... | 111111... |
| z = x + y | 8 | 9 | 4 | 11 | 6 | 9 | 8 | ... | 111111... |

# Un noyau synchrone flot de données

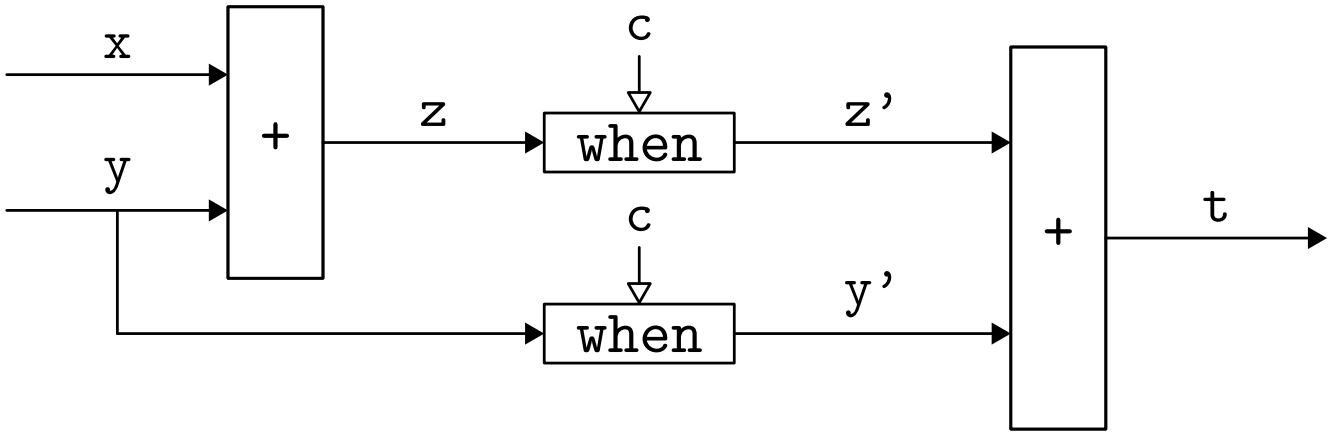





| flot | valeurs | | | | | | | | horloge |
|---|---|---|---|---|---|---|---|---|---|
| x | 5 | 7 | 3 | 6 | 2 | 8 | 1 | ... | 111111... |
| y | 3 | 2 | 1 | 5 | 4 | 1 | 7 | ... | 111111... |
| z = x + y | 8 | 9 | 4 | 11 | 6 | 9 | 8 | ... | 111111... |
| c | 1 | 0 | 1 | 0 | 1 | 0 | 1 | ... | |
| z' = z when c | 8 | | 4 | | 6 | | 8 | ... | 101010... |

# Un noyau synchrone flot de données

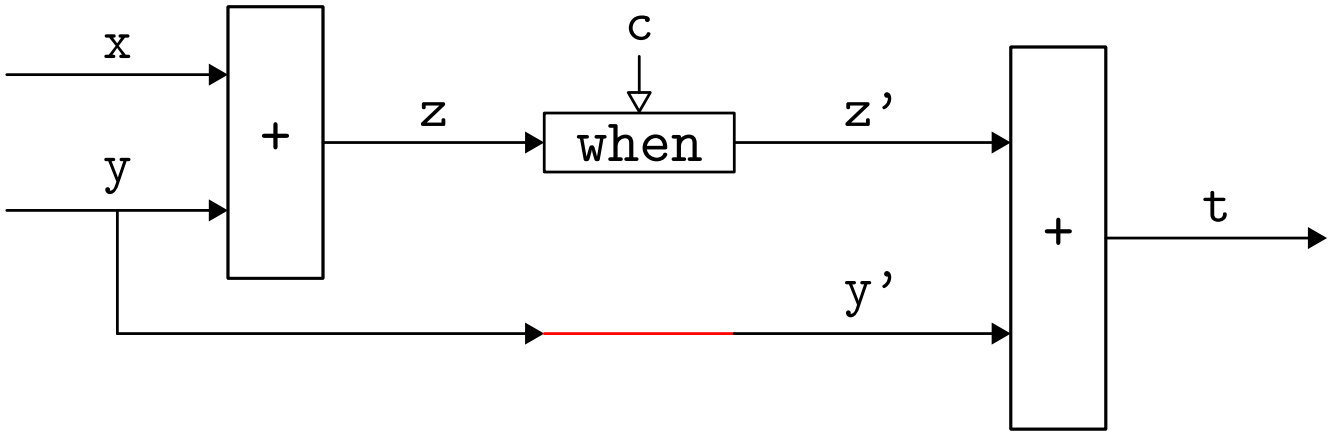





| flot | valeurs | | | | | | | | horloge |
|---|---|---|---|---|---|---|---|---|---|
| x | 5 | 7 | 3 | 6 | 2 | 8 | 1 | ... | 111111... |
| y | 3 | 2 | 1 | 5 | 4 | 1 | 7 | ... | 111111... |
| z = x + y | 8 | 9 | 4 | 11 | 6 | 9 | 8 | ... | 111111... |
| c | 1 | 0 | 1 | 0 | 1 | 0 | 1 | ... | |
| z' = z when c | 8 | | 4 | | 6 | | 8 | ... | 101010... |
| y' = y when c | 3 | | 1 | | 4 | | 7 | ... | 101010... |
| t = z' + y' | 11 | | 5 | | 10 | | 15 | ... | 101010... |

# Un noyau synchrone flot de données

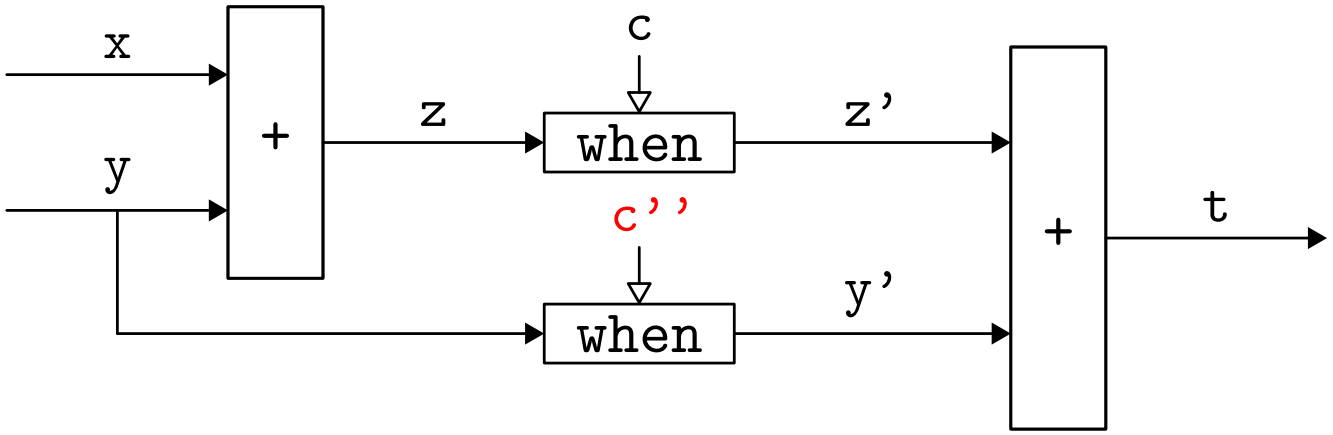





| flot | valeurs | | | | | | | | horloge |
|---|---|---|---|---|---|---|---|---|---|
| x | 5 | 7 | 3 | 6 | 2 | 8 | 1 | ... | 111111... |
| y | 3 | 2 | 1 | 5 | 4 | 1 | 7 | ... | 111111... |
| z = x + y | 8 | 9 | 4 | 11 | 6 | 9 | 8 | ... | 111111... |
| c | 1 | 0 | 1 | 0 | 1 | 0 | 1 | ... | |
| z' = z when c | 8 | | 4 | | 6 | | 8 | ... | 101010... |
| y' = y | 3 | 2 | 1 | 5 | 4 | 1 | 7 | ... | 111111... |
| t = z' + y' | rejeté | | | | | | | | |

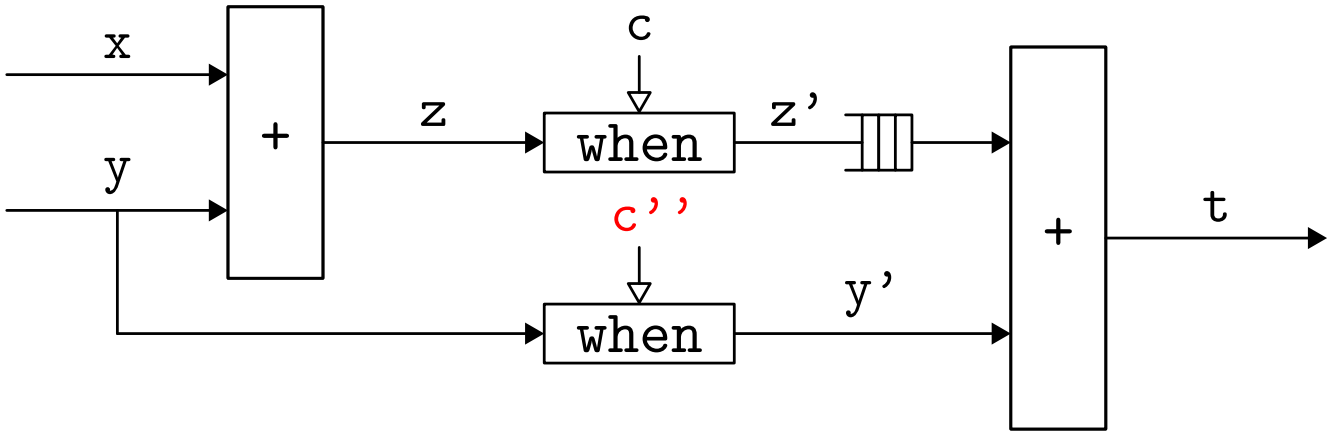
# Un noyau synchrone flot de données






| flot | valeurs | | | | | | | | horloge |
|---|---|---|---|---|---|---|---|---|---|
| x | 5 | 7 | 3 | 6 | 2 | 8 | 1 | ... | 111111... |
| y | 3 | 2 | 1 | 5 | 4 | 1 | 7 | ... | 111111... |
| z = x + y | 8 | 9 | 4 | 11 | 6 | 9 | 8 | ... | 111111... |
| c | 1 | 0 | 1 | 0 | 1 | 0 | 1 | ... | |
| z' = z when c | 8 | | 4 | | 6 | | 8 | ... | 101010... |
| y' = y when c'' | | 2 | | 5 | | 1 | | ... | 010101... |
| t = z' + y' | rejeté | | | | | | | | |

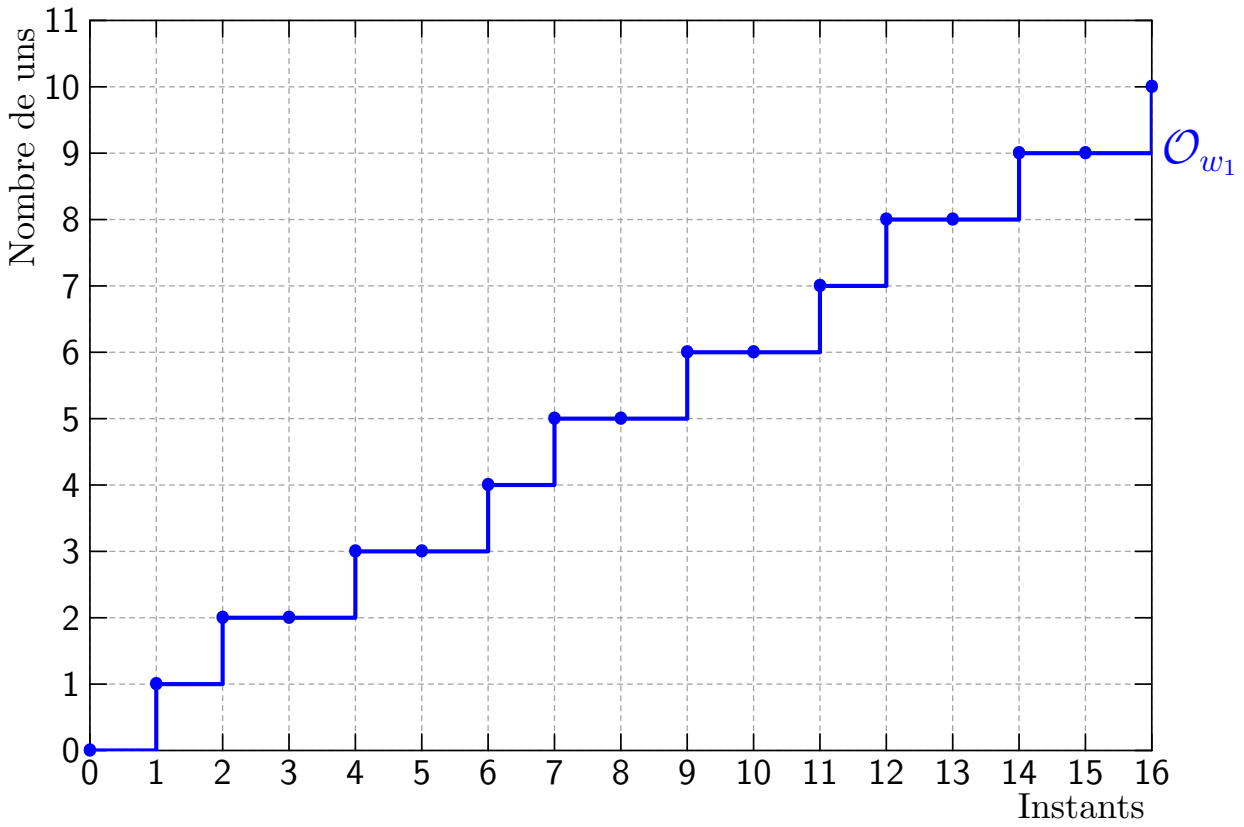
# Extension n-synchrone : opérateur de bufferisation




| flot | valeurs | | | | horloge |
|---|---|---|---|---|---|
| z' = z when c | 8 | 4 | 6 | 8 ... | 101010... |
| z'' = buffer(z') | 8 | 4 | 6 | ... | 010101... |
| y' = y when c'' | 2 | 5 | 1 | ... | 010101... |
| t = z'' + y' | 10 | 9 | 7 | ... | 010101... |

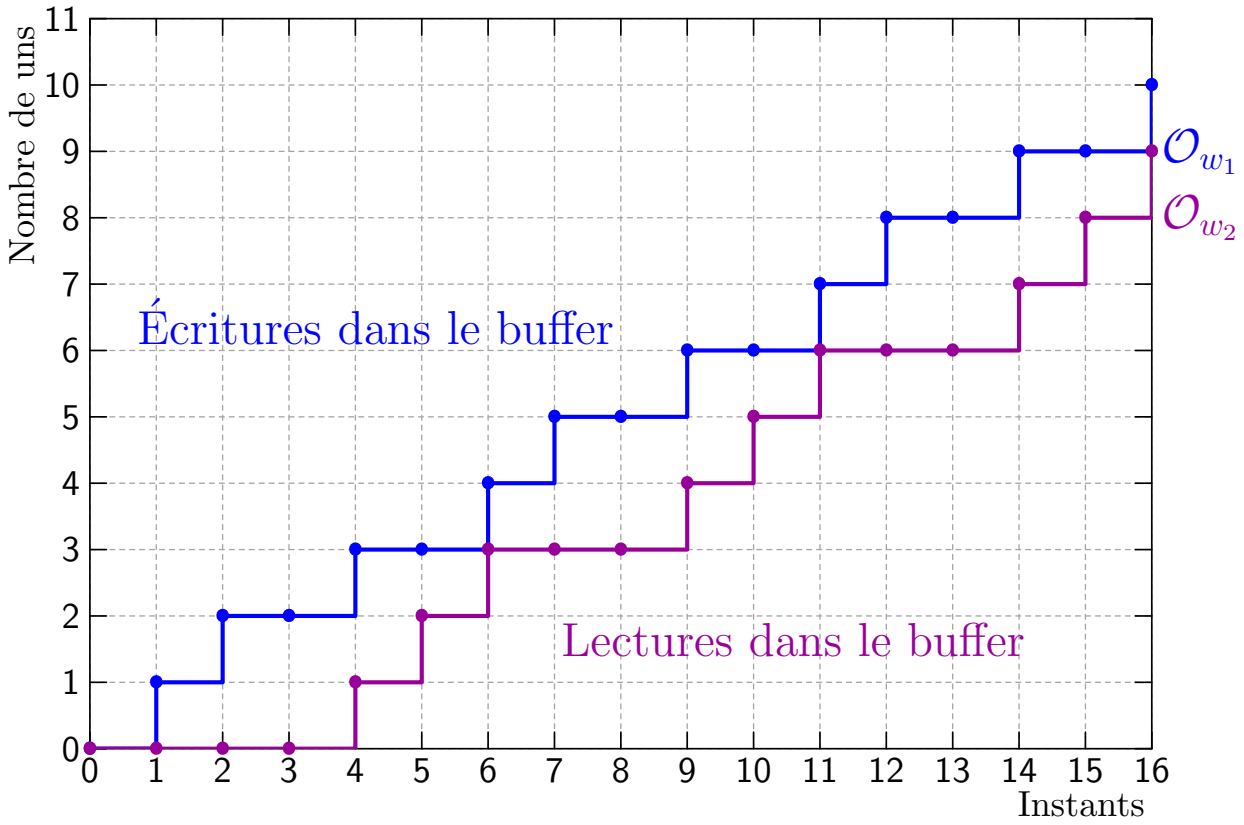
- Relation d'adaptabilité $\Rightarrow$ communication par buffer borné
- Exemple : 101010... $<:$ 010101...

# Relation d'adaptabilité




$\mathcal{O}_{w_1}$ : fonction de cumul du mot $w_1$

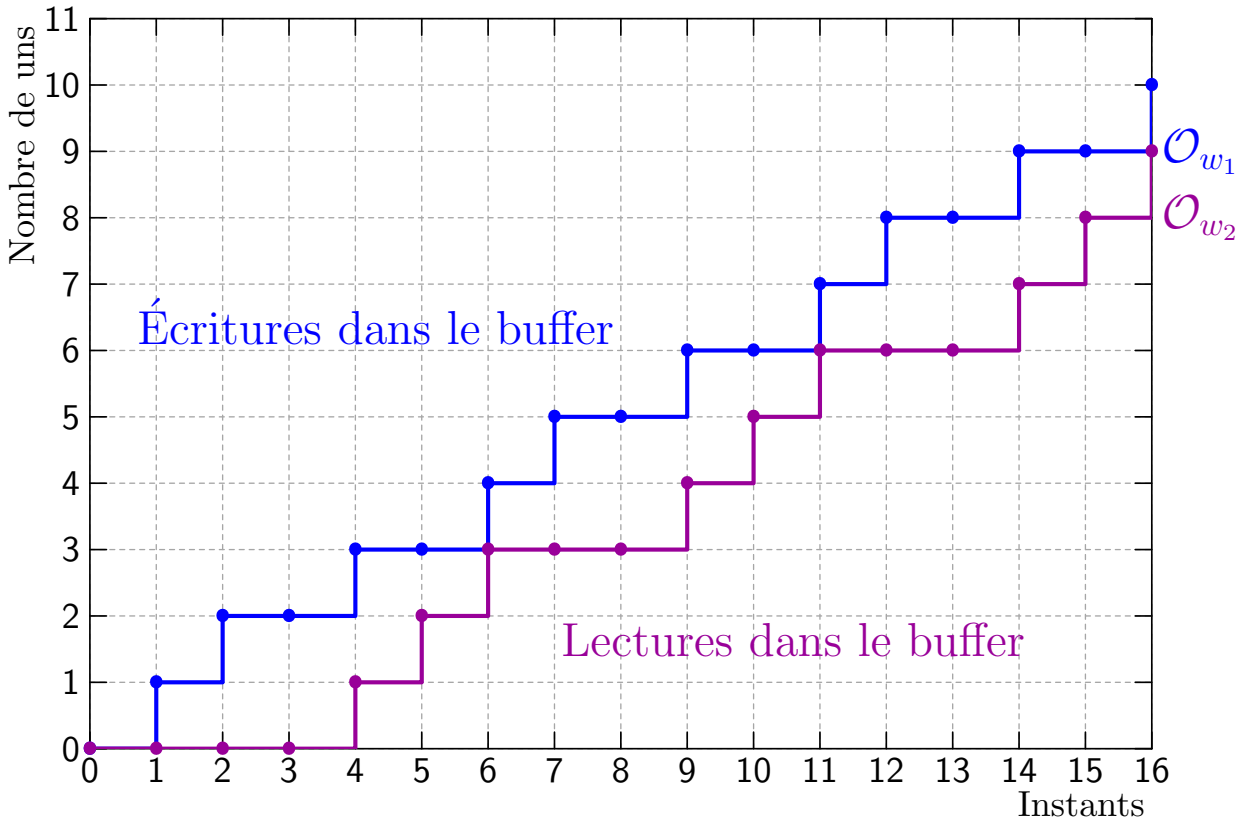
# Relation d'adaptabilité




taille du buffer $size(w_1, w_2) = \max_{i \in \mathbb{N}}(\mathcal{O}_{w_1}(i) - \mathcal{O}_{w_2}(i))$

adaptabilité $w_1 <: w_2 \overset{def}{\Leftrightarrow} \exists n \in \mathbb{N}, \forall i,\ 0 \leq \mathcal{O}_{w_1}(i) - \mathcal{O}_{w_2}(i) \leq n$

# Relation d'adaptabilité



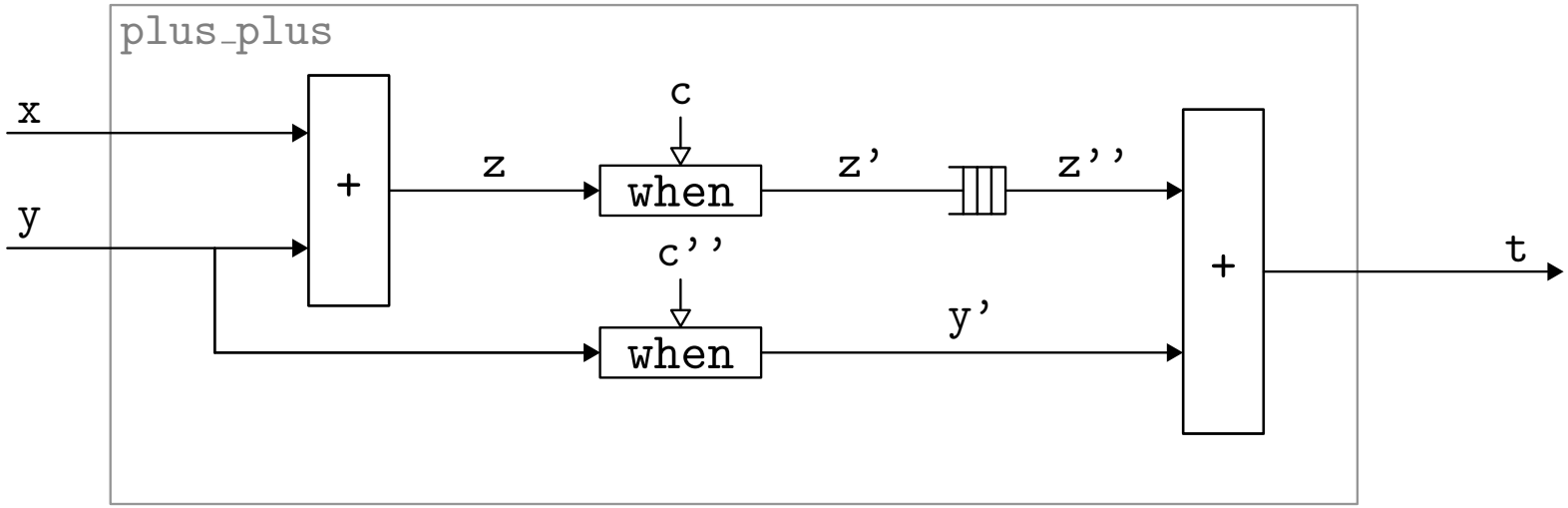

taille du buffer $size(w_1, w_2) = \max_{i \in \mathbb{N}}(\mathcal{O}_{w_1}(i) - \mathcal{O}_{w_2}(i))$

adaptabilité $w_1 <: w_2 \overset{def}{\Leftrightarrow} \exists n \in \mathbb{N}, \forall i,\ 0 \leq \mathcal{O}_{w_1}(i) - \mathcal{O}_{w_2}(i) \leq n$

synchronisabilité $w_1 \bowtie w_2 \overset{def}{\Leftrightarrow} \exists b_1, b_2 \in \mathbb{Z}, \forall i,\ b_1 \leq \mathcal{O}_{w_1}(i) - \mathcal{O}_{w_2}(i) \leq b_2$

précédence $w_1 \preceq w_2 \overset{def}{\Leftrightarrow} \forall i,\ \mathcal{O}_{w_1}(i) \geq \mathcal{O}_{w_2}(i)$

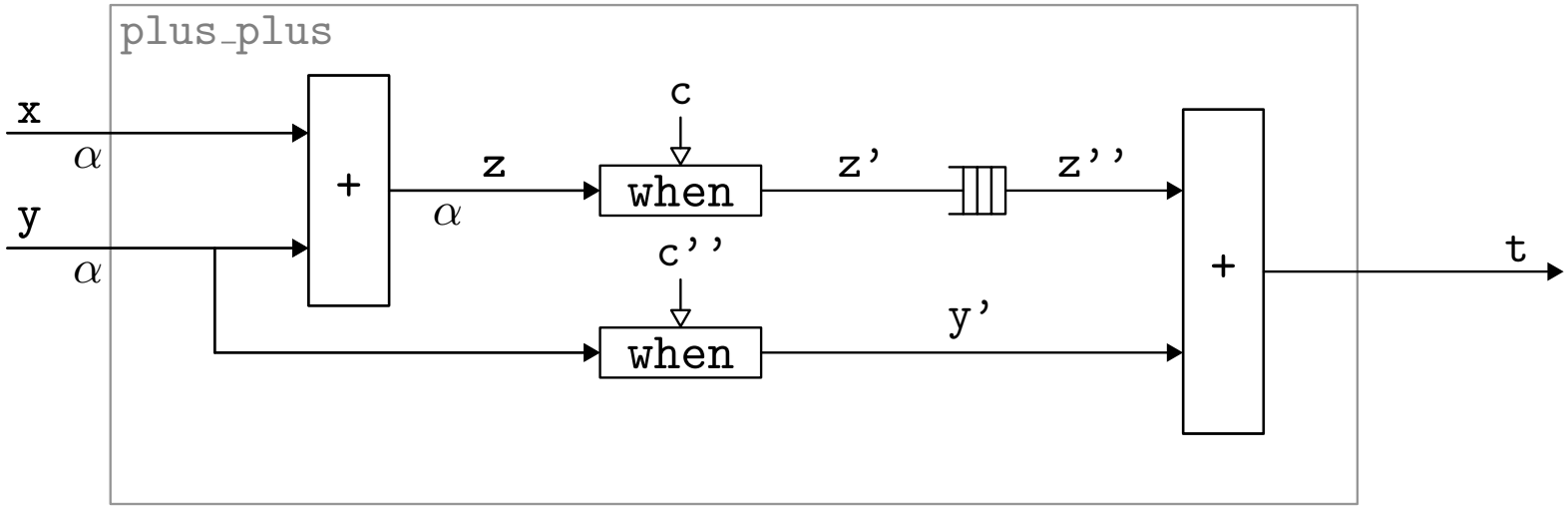
# Typage




```
4  let node plus_plus (x,y) = t where
5    rec z = x + y
6    and z' = z when c
7    and z''= buffer(z')
8    and y' = y when c''
9    and t = z'' + y'
```

```
val plus_plus : (int * int) -> int
val plus_plus :: forall 'a. ('a * 'a) -> 'a on c''
Buffer line 7, characters 11-21: size = 1
```

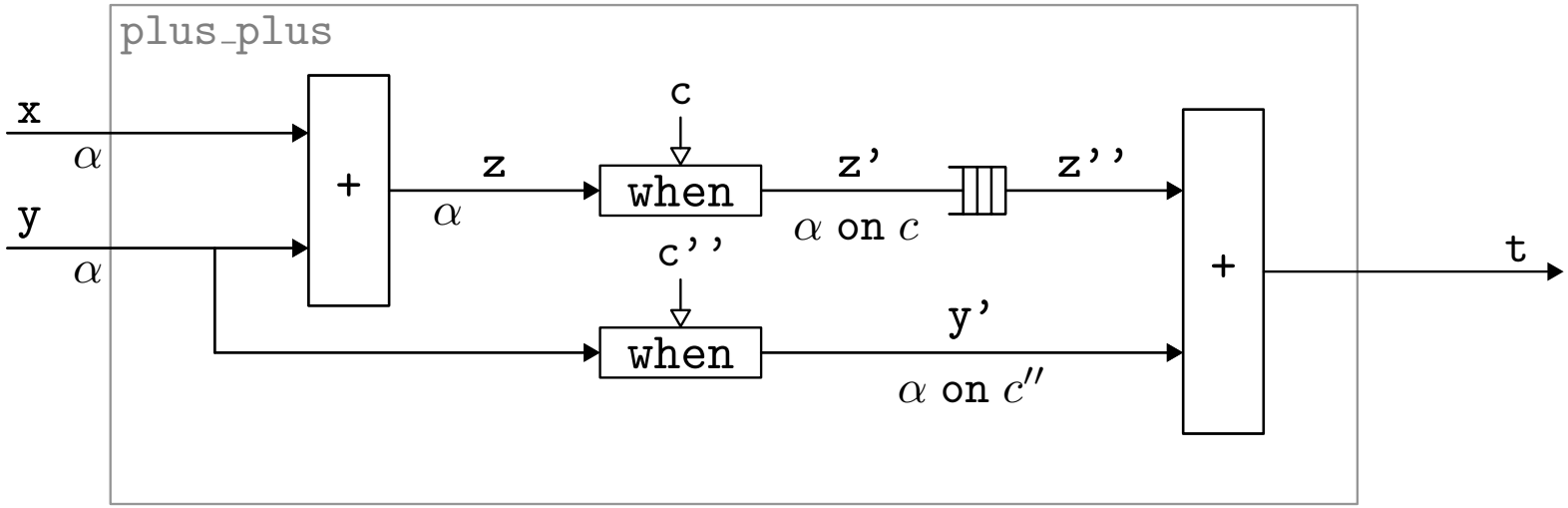
# Typage




```
4  let node plus_plus (x,y) = t where
5    rec z = x + y
6    and z' = z when c
7    and z''= buffer(z')
8    and y' = y when c''
9    and t = z'' + y'
```

```
val plus_plus : (int * int) -> int
val plus_plus :: forall 'a. ('a * 'a) -> 'a on c''
Buffer line 7, characters 11-21: size = 1
```

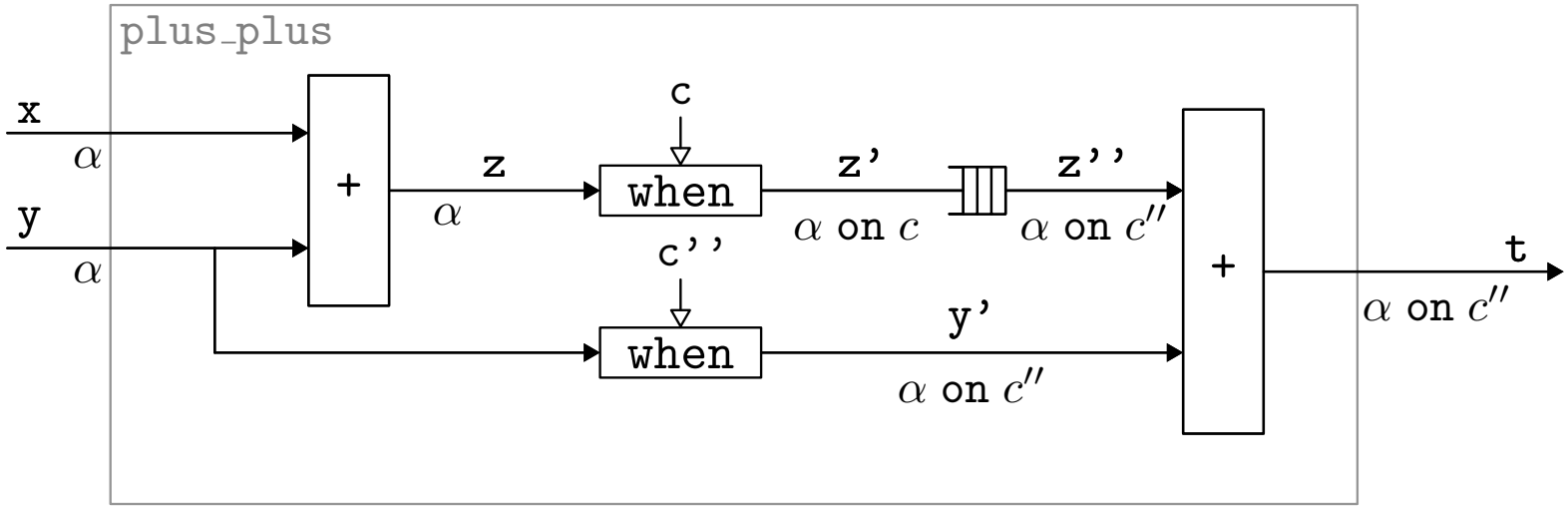
# Typage




```
4  let node plus_plus (x,y) = t where
5    rec z = x + y
6    and z' = z when c
7    and z''= buffer(z')
8    and y' = y when c''
9    and t = z'' + y'
```

```
val plus_plus : (int * int) -> int
val plus_plus :: forall 'a. ('a * 'a) -> 'a on c''
Buffer line 7, characters 11-21: size = 1
```

# Typage




```
4  let node plus_plus (x,y) = t where
5    rec z = x + y
6    and z' = z when c
7    and z''= buffer(z')
8    and y' = y when c''
9    and t = z'' + y'
```

```
val plus_plus : (int * int) -> int
val plus_plus :: forall 'a. ('a * 'a) -> 'a on c''
Buffer line 7, characters 11-21: size = 1
```

- Types d'horloges :

$$\begin{array}{lcl} \sigma & ::= & \forall \alpha_1, \ldots, \alpha_n. (ck \times ... \times ck) \rightarrow (ck \times ... \times ck) \\ ck & ::= & \alpha \quad | \quad ck \texttt{ on } ce \quad | \quad ck \texttt{ on not } ce \end{array}$$

- Exemple :

$$\frac{H \vdash e : ck}{H \vdash e \texttt{ when } ce : ck \texttt{ on } ce}$$

- Contraintes :

$$C \quad ::= \quad \{ck_1 = ck_2\} \cup C \quad | \quad \{ck_1 <: ck_2\} \cup C \quad | \quad \varnothing$$

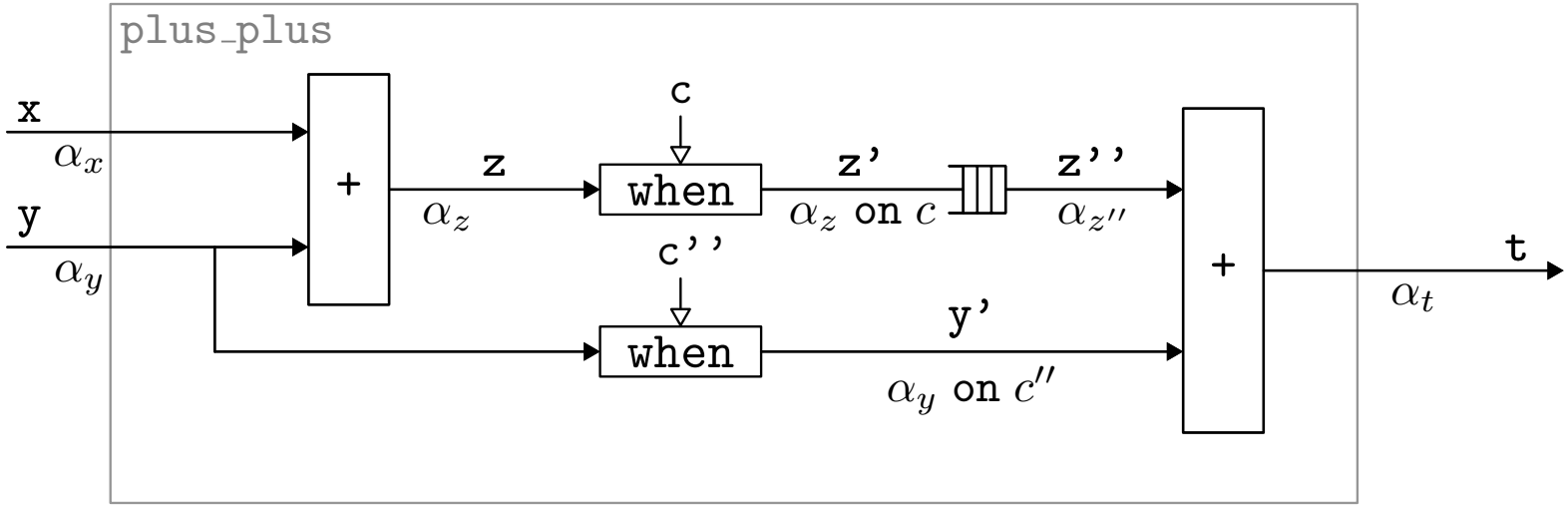
Collection des contraintes

- d'égalité :

$$\frac{H \vdash e_1 : ck_1 \mid C_1 \qquad H \vdash e_2 : ck_2 \mid C_2}{H \vdash e_1 \ \texttt{+}\ e_2 : \alpha \mid \{ck_1 = ck_2 = \alpha\} \cup C_1 \cup C_2}$$

- de sous-typage :

$$\frac{H \vdash e : ck \mid C}{H \vdash \texttt{buffer}(e) : \alpha \mid \{ck <: \alpha\} \cup C}$$




$$(\alpha_x \times \alpha_y) \to \alpha_t \qquad \text{tel que} \quad C = \left\{ \begin{array}{l} \alpha_x = \alpha_y = \alpha_z; \\ \alpha_y \text{ on } c'' = \alpha_{z''} = \alpha_t; \\ \alpha_z \text{ on } c <: \alpha_{z''} \end{array} \right\}$$

$$\rightsquigarrow \quad (\alpha \times \alpha) \to \alpha \text{ on } c'' \quad \text{tel que} \quad C = \left\{ \alpha \text{ on } c <: \alpha \text{ on } c'' \right\}$$

Résolution des contraintes

- Cas simple :

$$\alpha \texttt{ on } w_1 \texttt{ <: } \alpha \texttt{ on } w_2 \quad \Leftrightarrow \quad w_1 <: w_2$$

  $\Rightarrow$ vérification de la relation d'adaptabilité $w_1 <: w_2$

- Plus difficile :

$$\alpha_1 \texttt{ on } w_1 \texttt{ <: } \alpha_2 \texttt{ on } w_2 \quad \Leftarrow \quad \begin{cases} \alpha_1 \leftarrow \alpha \texttt{ on } c_1 \\ \alpha_2 \leftarrow \alpha \texttt{ on } c_2 \\ \alpha \texttt{ on } c_1 \texttt{ on } w_1 \texttt{ <: } \alpha \texttt{ on } c_2 \texttt{ on } w_2 \end{cases}$$

  $\Rightarrow$ inférence des rythmes $c_1$ et $c_2$ tels que $c_1$ *on* $w_1 <: c_2$ *on* $w_2$

Résolution des contraintes

- Cas simple :

$$\alpha \texttt{ on } w_1 \texttt{ <: } \alpha \texttt{ on } w_2 \quad \Leftrightarrow \quad w_1 <: w_2$$

  $\Rightarrow$ vérification de la relation d'adaptabilité $w_1 <: w_2$

- Plus difficile :

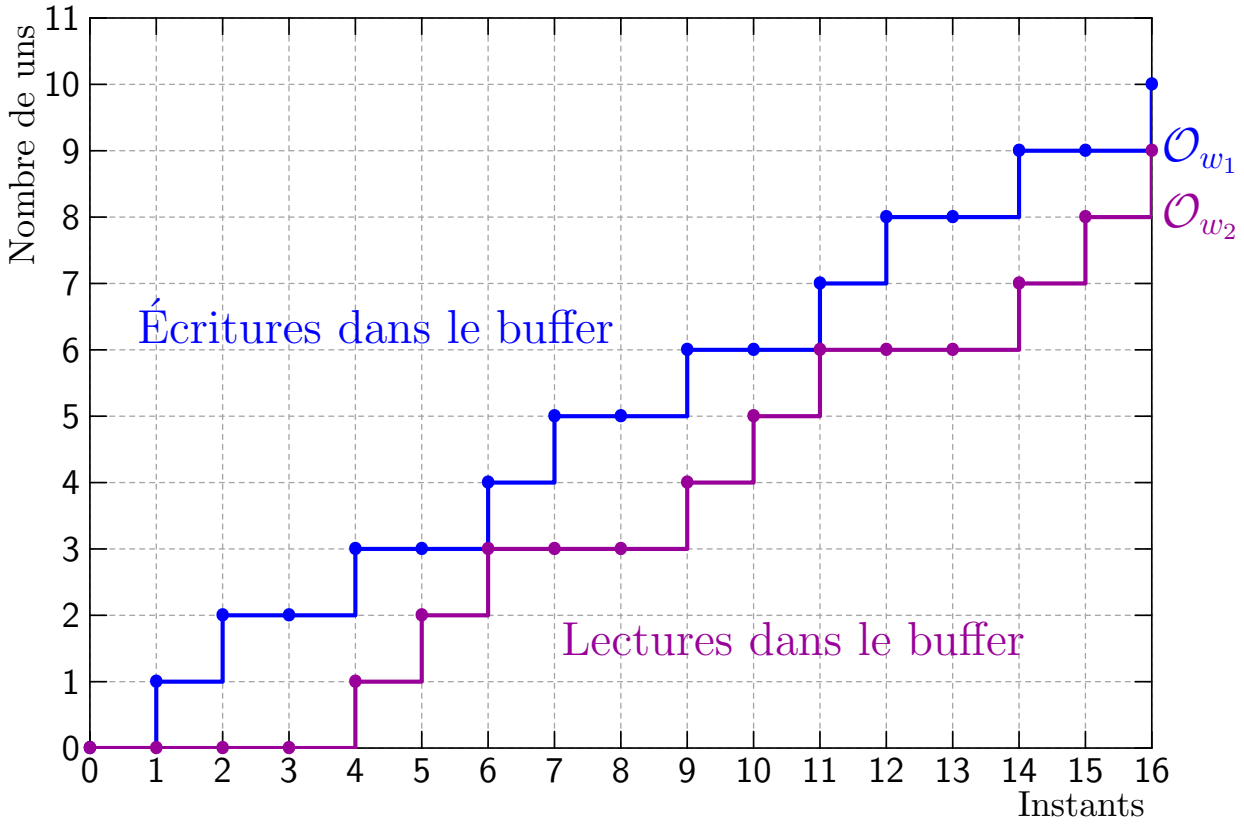
$$\alpha_1 \texttt{ on } w_1 \texttt{ <: } \alpha_2 \texttt{ on } w_2 \quad \Leftarrow \quad \begin{cases} \alpha_1 \leftarrow \alpha \texttt{ on } c_1 \\ \alpha_2 \leftarrow \alpha \texttt{ on } c_2 \\ \alpha \texttt{ on } c_1 \texttt{ on } w_1 \texttt{ <: } \alpha \texttt{ on } c_2 \texttt{ on } w_2 \end{cases}$$

  $\Rightarrow$ inférence des rythmes $c_1$ et $c_2$ tels que $c_1$ *on* $w_1 <: c_2$ *on* $w_2$

Calcul de la taille du buffer

$$\texttt{size}(\alpha \texttt{ on } w_1, \alpha \texttt{ on } w_2) = size(w_1, w_2)$$

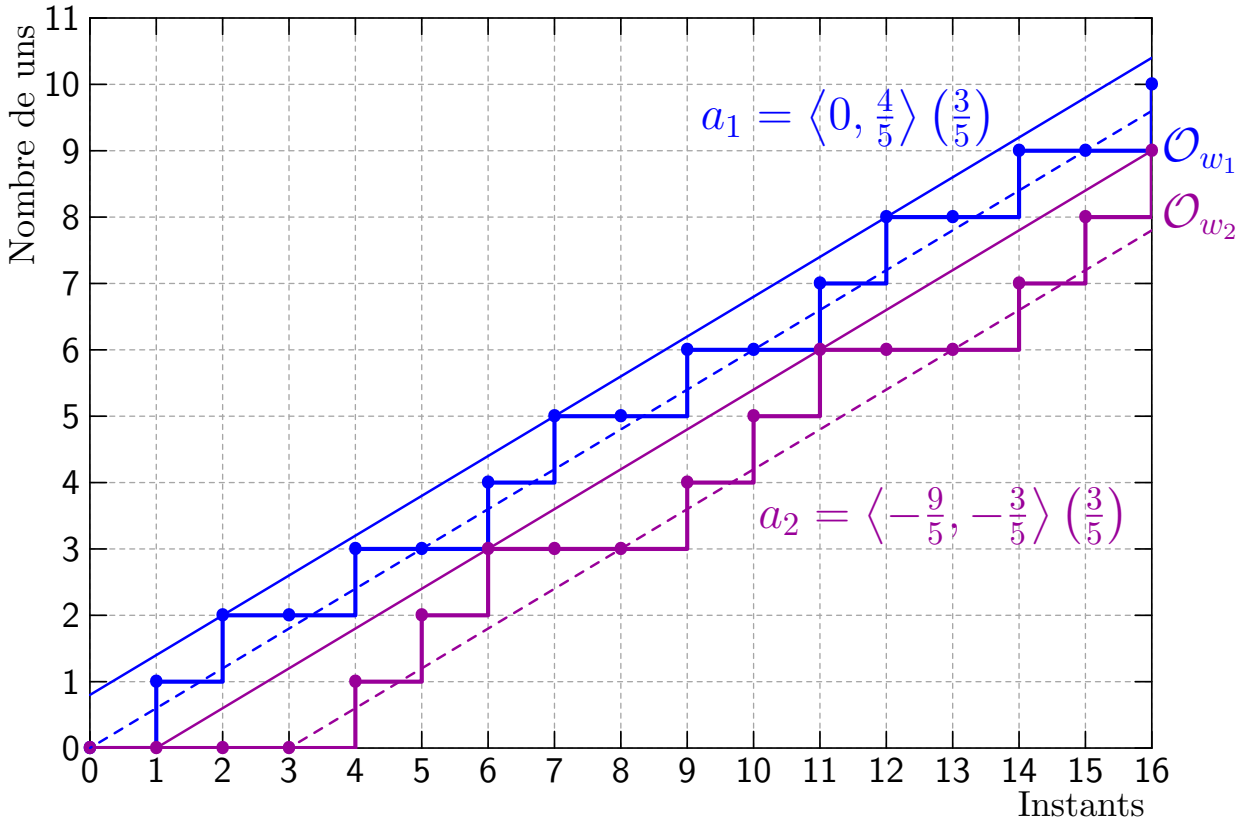
# Langages d'horloges étudiés




Horloges ultimement périodiques :

- test statique : $(11010) <: 0(00111)$
- inférence des rythmes : résolution d'un ensemble de contraintes linéaires sur les indices des 1 des rythmes à inférer.
- taille du buffer : $size((11010), 0(00111)) = 2$

# Langages d'horloges étudiés




Horloges abstraites :

- test statique : $\langle 0, \frac{4}{5} \rangle \left(\frac{3}{5}\right) <:^{\sim} \langle -\frac{9}{5}, -\frac{3}{5} \rangle \left(\frac{3}{5}\right)$
- inférence des rythmes : résolution d'un ensemble de contraintes linéaires sur les abstractions des rythmes à inférer.
- $size^{\sim}(\langle 0, \frac{4}{5} \rangle \left(\frac{3}{5}\right), \langle -\frac{9}{5}, -\frac{3}{5} \rangle \left(\frac{3}{5}\right)) = 2$

# Conclusion

- Modèle n-synchrone :
  composition plus souple des nœuds, sans perte de garanties
- Deux langages d'horloges étudiés
- Algorithme avec abstraction :
  efficace et utilisable sur des horloges pas strictement périodiques
- Algorithmes implémentés dans Lucy-n

# Perspectives

- Horloges entières
- Inférence des rythmes sur les horloges périodiques
- Buffers limités, buffers stricts
- Génération de code