

# Analyse statique par interprétation abstraite des réseaux de programmes synchrones communicants à horloge imparfaite

Julien Bertrane  
bertrane@di.ens.fr

14 janvier 2011

## 1 Introduction

## 2 Modélisation

- Contraintes sur la modélisation
- Sémantique “temps continu”
- Syntaxe graphique

## 3 Analyse abstraite

- Vers une analyse abstraite
- Domaine des contraintes abstraites
- Domaine du comptage des Changements de valeurs
- Domaine des encadrement des intégrales

## 4 Analyse de code

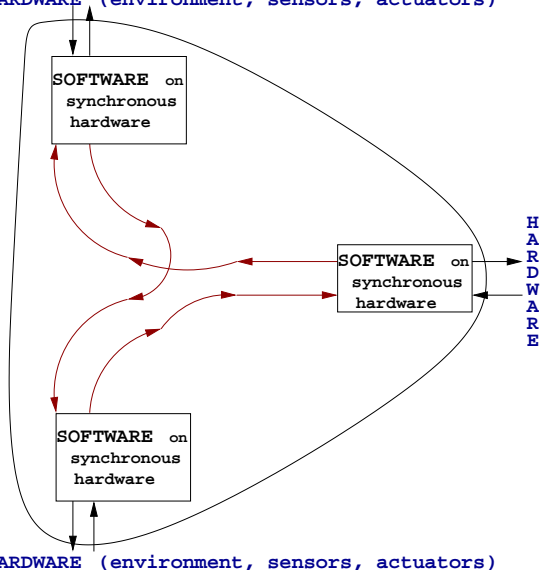
- Type de programmes considérés
- Analyse : phase de pré-analyse
- Analyse

## 5 conclusion

# Introduction

# Système type à analyser : systèmes embarqués

HARDWARE (environment, sensors, actuators)



HARDWARE (environment, sensors, actuators)

# Programme synchrone

- ▶ Initialize(S)
  - ▶ while true do
    - ★ (O, S) := Compute (S, I)
    - ★ wait for clock
  - ▶ od

où I : entrées, S : variables d'état, O : sortie

# Objectifs

- Analyser des Systèmes composés de **plusieurs programmes synchrones communicants** (chacun avec sa propre horloge). Pourquoi ?
  - ▶ certains systèmes embarqués sont **trop grands** pour une unique horloge : la transmission de l'information serait **trop lente**
  - ▶ les systèmes critiques sont **redondants** pour résister à la panne d'une unité ( $\Rightarrow$  plusieurs horloges)
  - ▶ Concevoir des sous-systèmes synchrones et les vérifier formellement prends du temps et est coûteux. Ils sont donc implantés plusieurs fois dans un même systèmes.

# Objectifs

- Analyser des Systèmes composés de **plusieurs programmes synchrones communicants** (chacun avec sa propre horloge). Pourquoi ?
  - ▶ certains systèmes embarqués sont **trop grands** pour une unique horloge : la transmission de l'information serait **trop lente**
  - ▶ les systèmes critiques sont **redondants** pour résister à la panne d'une unité ( $\Rightarrow$  plusieurs horloges)
  - ▶ Concevoir des sous-systèmes synchrones et les vérifier formellement prends du temps et est coûteux. Ils sont donc implantés plusieurs fois dans un même systèmes.
- la preuve doit être automatique, rapide, robuste aux changements mineurs dans le code. Basé sur la théorie de l'interprétation abstraite

# Spécifications à vérifier

- Spécifications de **sécurité**
  - ▶ **Pour tout comportement  $s$ , à tout instant  $t$ ,  $s(t) \neq true$**

# Spécifications à vérifier

- Spécifications de **sécurité**
  - ▶ **Pour tout comportement  $s$ , à tout instant  $t$ ,  $s(t) \neq true$**
- Spécifications **temporelles**
  - ▶ **Pour tout comportement  $s$ , à aucun instant  $t$  on a :**

pour tout  $t' \in [t, t + \alpha], s(t') = true$

# Spécifications à vérifier

- Spécifications de **sécurité**
  - ▶ **Pour tout comportement  $s$ , à tout instant  $t$ ,  $s(t) \neq true$**
- Spécifications **temporelles**
  - ▶ **Pour tout comportement  $s$ , à aucun instant  $t$  on a :**

$$\text{pour tout } t' \in [t, t + \alpha], s(t') = true$$

- Spécifications **quantitative**
  - ▶ les **sorties** de 2 systèmes redondants sont **égales au moins 50% du temps** durant chaque intervalle de temps de largeur minimale  $\delta$ .

# Modélisation

# Contraintes sur le système étudié

- Imperfections matérielles inévitables :
  - ▶ Les horloges des unités de commande se **désynchronisent**

# Contraintes sur le système étudié

- Imperfections matérielles inévitables :
  - ▶ Les horloges des unités de commande se **désynchronisent**
  - ▶ Les communications ne sont **pas instantanées**

# Contraintes sur le système étudié

- Imperfections matérielles inévitables :
  - ▶ Les horloges des unités de commande se **désynchronisent**
  - ▶ Les communications ne sont **pas instantanées**
  - ▶ Le temps de communication n'est **pas constant**

# Hypothèses pour ce modèle

- **synchronie imparfaite** :

- ▶ désynchronisation : la durée de chaque cycle (période entre deux **ticks**) est dans  $[\alpha, \beta]$ ,  $\alpha > 0$ . Physiquement (horloge quartz) : c'est toujours le cas.
- ▶ différent du quasi-synchronisme introduit par P. Caspi

# Hypothèses pour ce modèle

- **synchronie imparfaite** :
  - ▶ désynchronisation : la durée de chaque cycle (période entre deux **ticks**) est dans  $[\alpha, \beta]$ ,  $\alpha > 0$ . Physiquement (horloge quartz) : c'est toujours le cas.
  - ▶ différent du quasi-synchronisme introduit par P. Caspi
  
- Transmission en **série** entre 2 systèmes synchrones

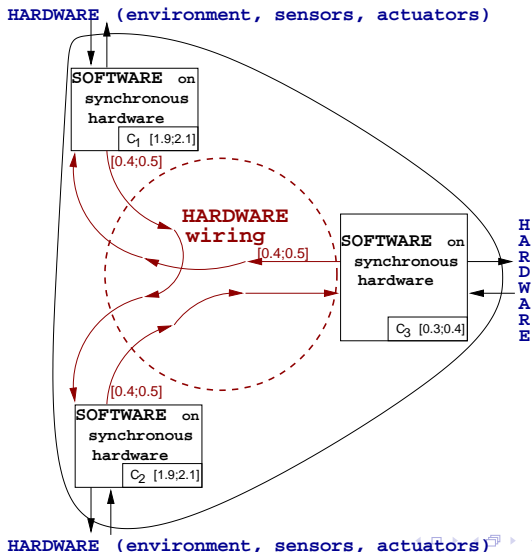
# Hypothèses pour ce modèle

- **synchronie imparfaite** :
  - ▶ désynchronisation : la durée de chaque cycle (période entre deux **ticks**) est dans  $[\alpha, \beta]$ ,  $\alpha > 0$ . Physiquement (horloge quartz) : c'est toujours le cas.
  - ▶ différent du quasi-synchronisme introduit par P. Caspi
- Transmission en **série** entre 2 systèmes synchrones
- **Blackboard** à l'entrée de chaque système

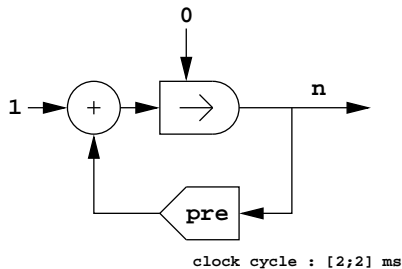
# Hypothèses pour ce modèle

- **synchronie imparfaite** :
  - ▶ désynchronisation : la durée de chaque cycle (période entre deux **ticks**) est dans  $[\alpha, \beta]$ ,  $\alpha > 0$ . Physiquement (horloge quartz) : c'est toujours le cas.
  - ▶ différent du quasi-synchronisme introduit par P. Caspi
- Transmission en **série** entre 2 systèmes synchrones
- **Blackboard** à l'entrée de chaque système
- **À l'initialisation** toutes les variables contiennent 0 ou *false*

# Système type à analyser : détails quantifiés des imperfections matérielles



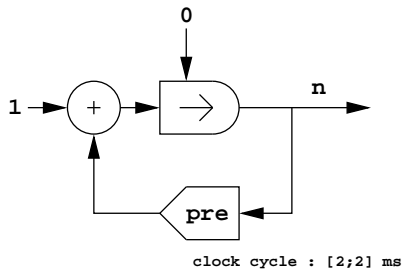
# Quelles sémantiques ?



- $n = 0 \rightarrow (1 + pre\ n)$
- $a_i = pre(b_i) \Leftrightarrow \begin{matrix} a_0 \text{ undefined} \\ a_{i+1} = b_i \end{matrix}$
- $a_i = b_i \rightarrow c_i \Leftrightarrow \begin{matrix} a_0 = b_0 \\ a_{i+1} = c_{i+1} \end{matrix}$

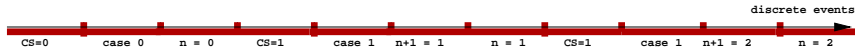
# Quelles sémantiques ?

La sémantique classique d'un programme est + ou - celle du C

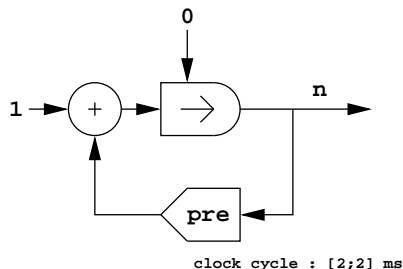


```
switch (global_state->current_state){
case 0 :
global_state->n= 0;
global_state->current_state = 1; break;
break;

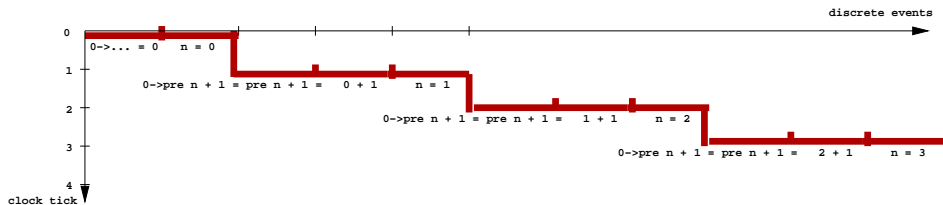
case 1 :
global_state->n= (global_state->n)+ 1;
global_state->current_state = 1; break;
break;}
```



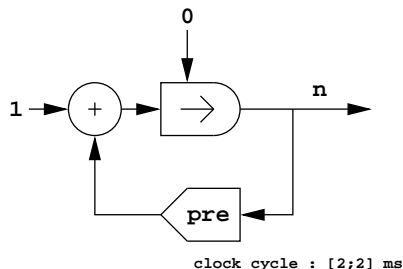
# Quelles sémantiques ?



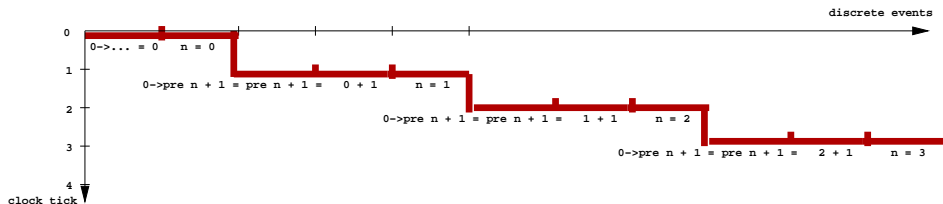
- Pour soft embarqué, il important de respecter des timings



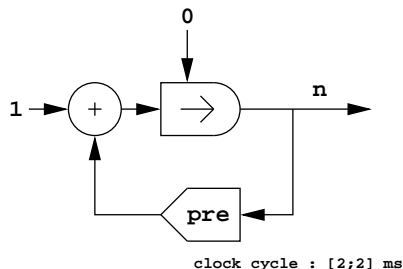
# Quelles sémantiques ?



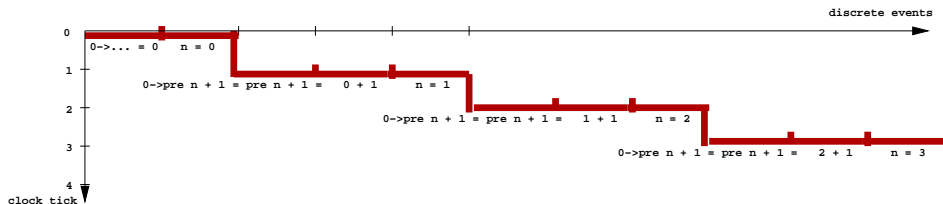
- Pour soft embarqué, il est important de respecter des timings
- en C, on n'a aucune garantie sur ces timings. Ils dépendent
  - ▶ de l'ordinateur exécutant le code
  - ▶ du compilateur,
  - ▶ des optimisations éventuelles ...



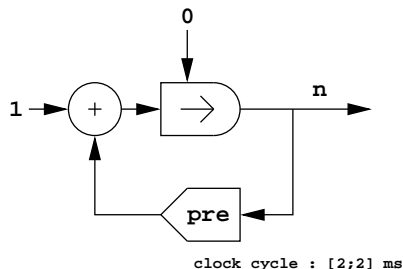
# Quelles sémantiques ?



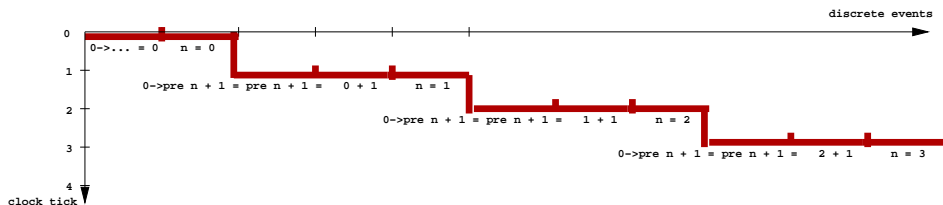
- On peut définir des cycles d'horloge rythmant l'exécution



# Quelles sémantiques ?



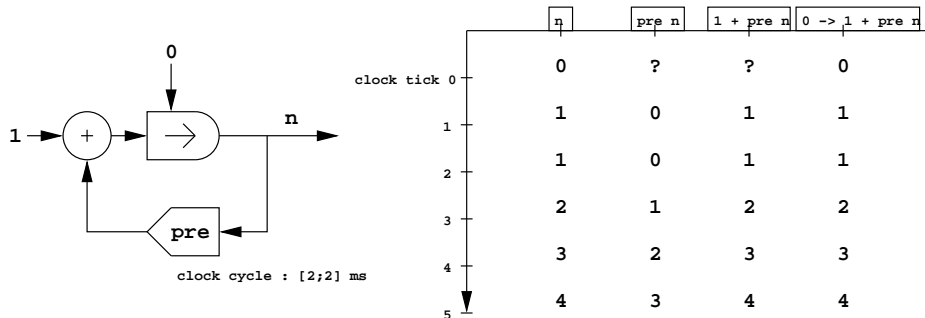
- On peut définir des cycles d'horloge rythmant l'exécution
- Question : l'ensemble des instructions réparties sur un niveau a-t-il assez de temps pour s'exécuter entre deux tick consécutifs d'horloge (WCET) ?



# Avec quelles sémantiques doit-on travailler

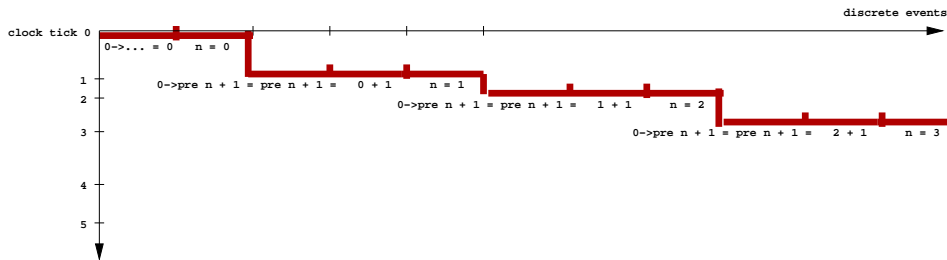
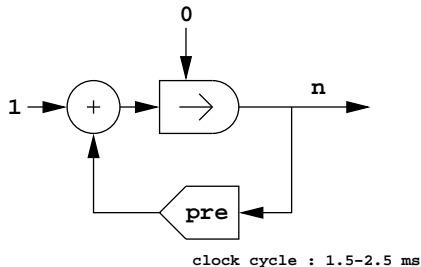
On est donc conduit à définir une autre sémantique : celle du Lustre

$$\llbracket S \rrbracket : P \rightarrow (\mathbb{N} \rightarrow (\mathbb{N} \cup \{\text{undefined}\}))$$



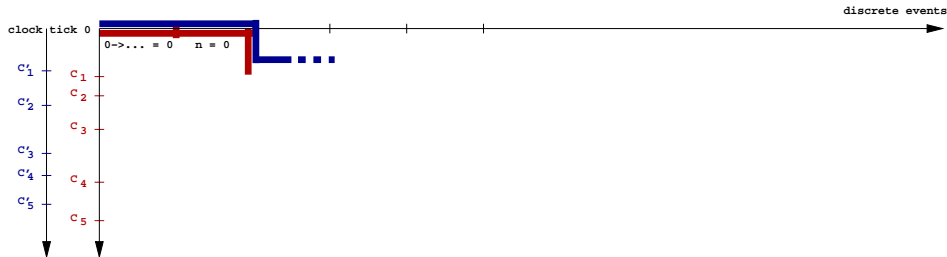
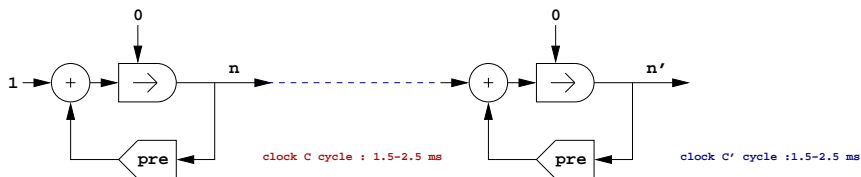
# Avec quelles sémantiques doit-on travailler

- Et si l'horloge est imprécise ?



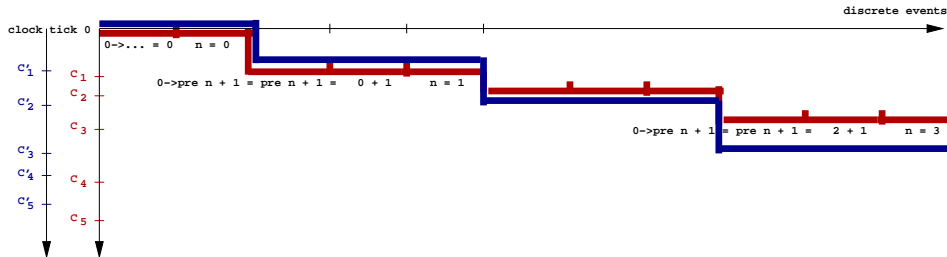
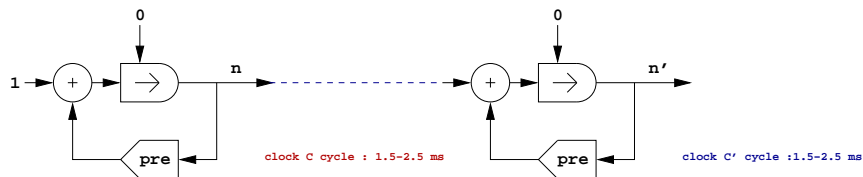
# Avec quelles sémantiques doit-on travailler

Cas de plusieurs systèmes, chacun ayant son horloge imprécise :



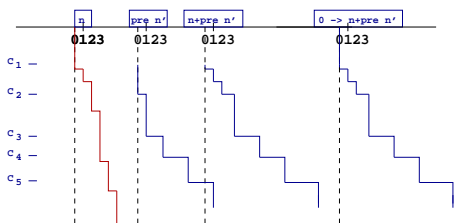
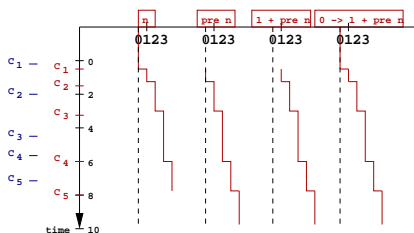
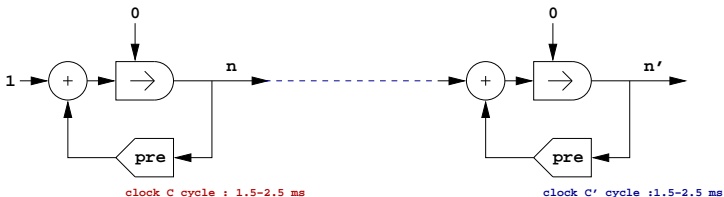
# Avec quelles sémantiques doit-on travailler

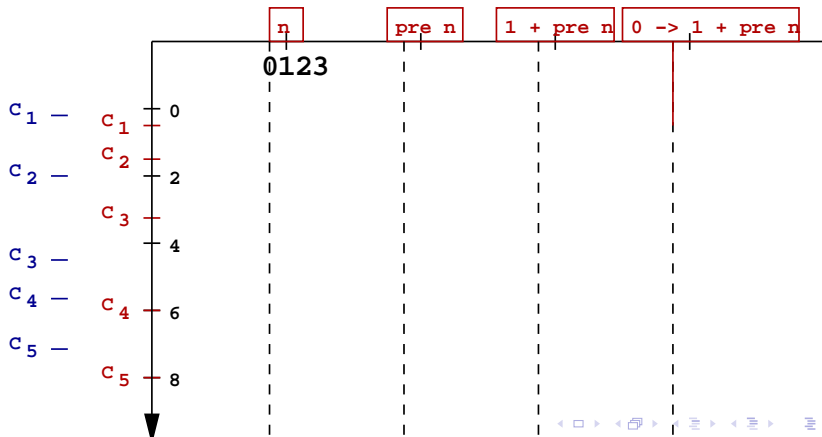
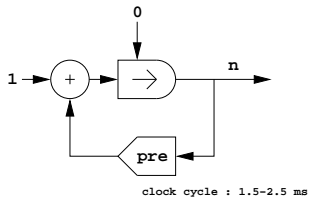
Cas de plusieurs systèmes, chacun ayant son horloge imprécise :

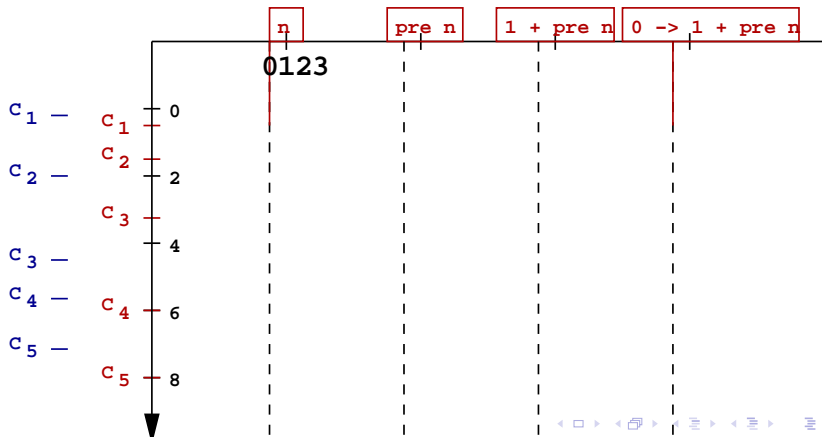
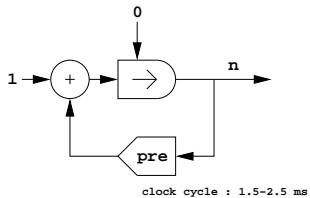


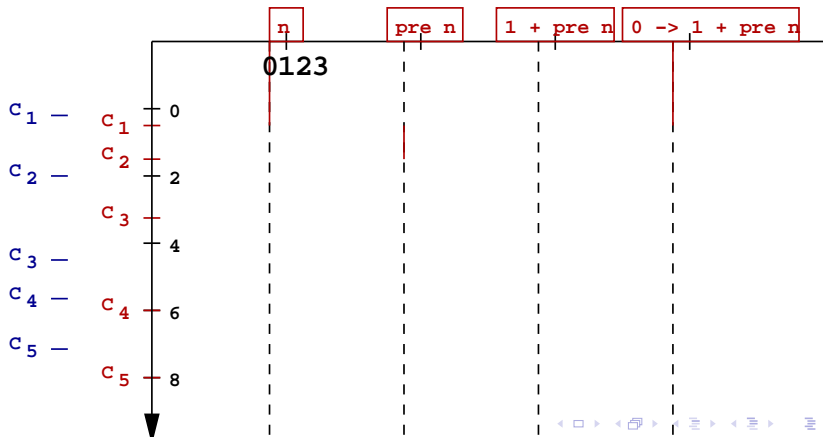
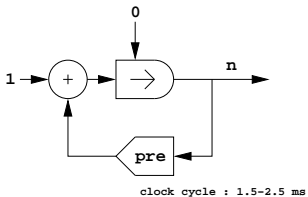
# Avec quelles sémantiques doit-on travailler

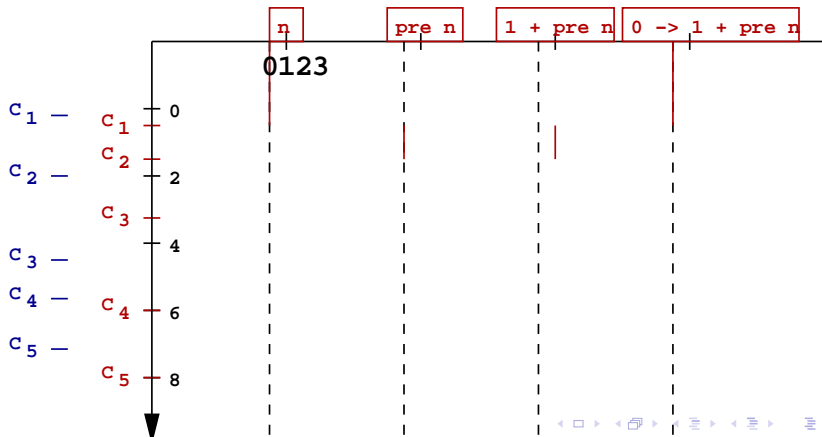
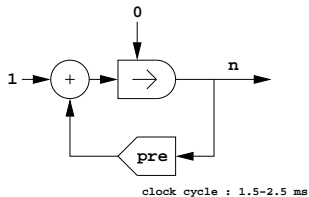
Une sémantique totalement continue permet d'étudier la communication de plusieurs systèmes

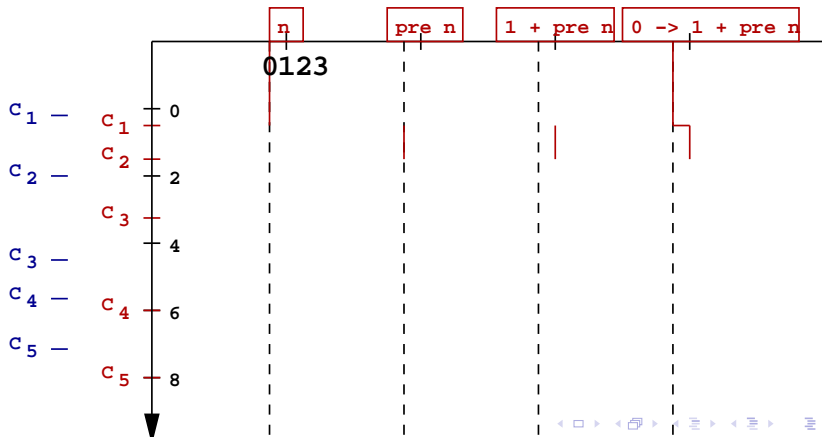
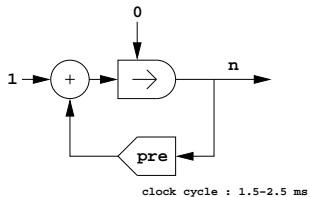


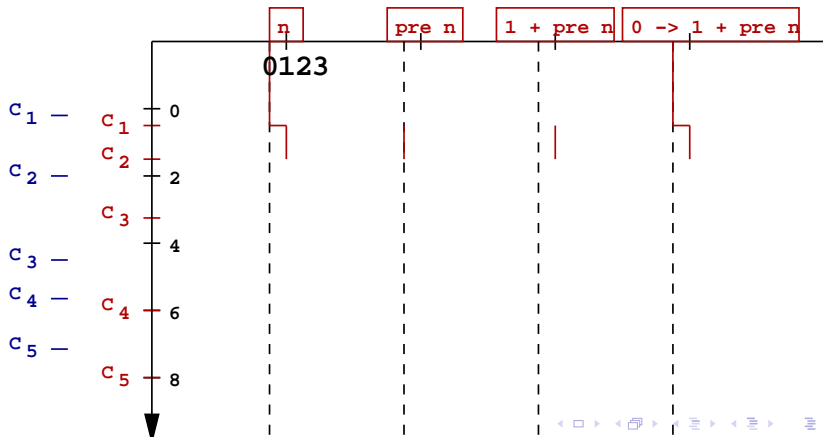
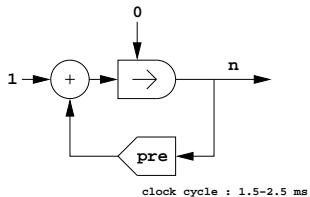


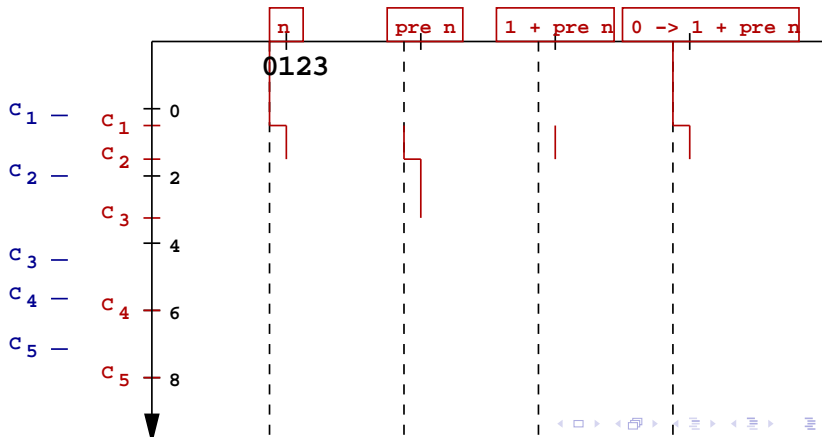
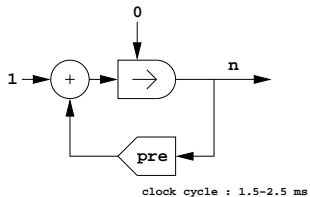


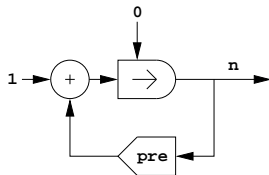




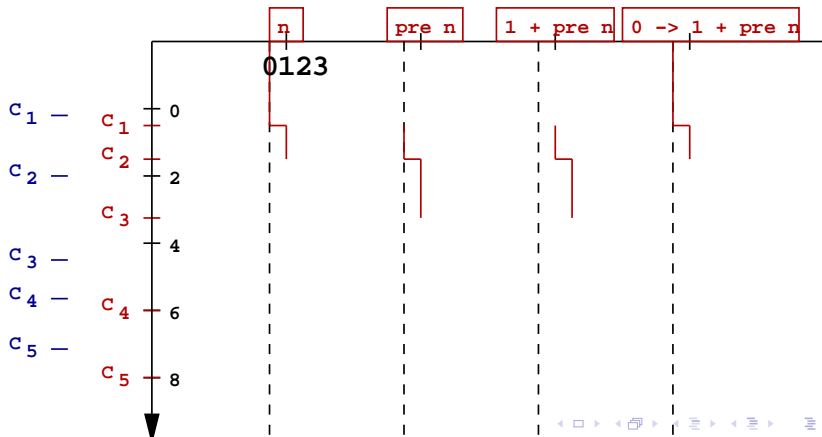


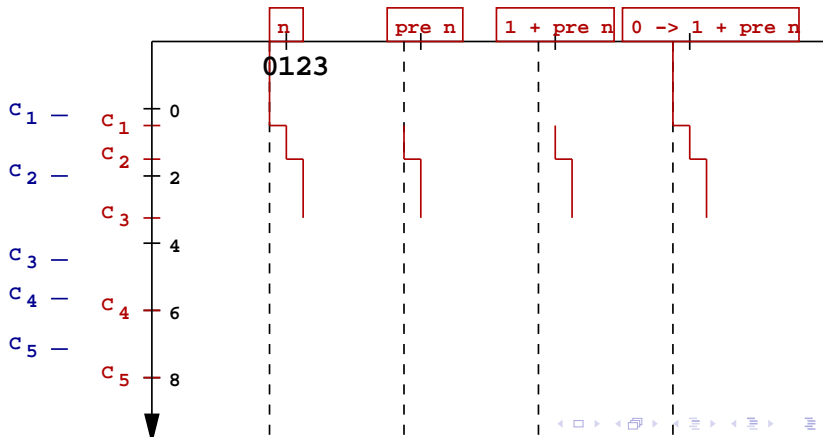
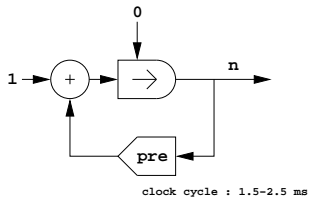


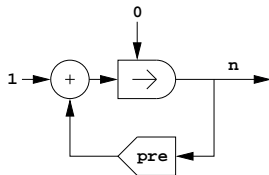




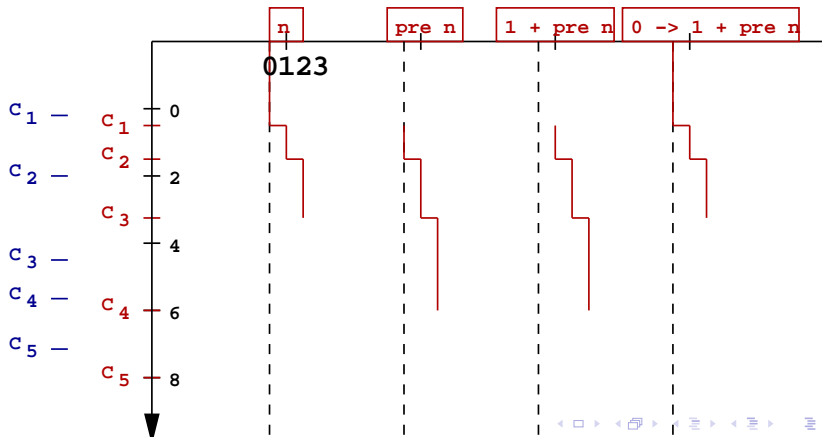
clock cycle : 1.5-2.5 ns

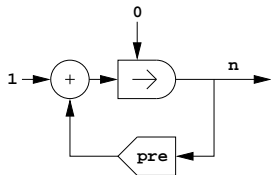




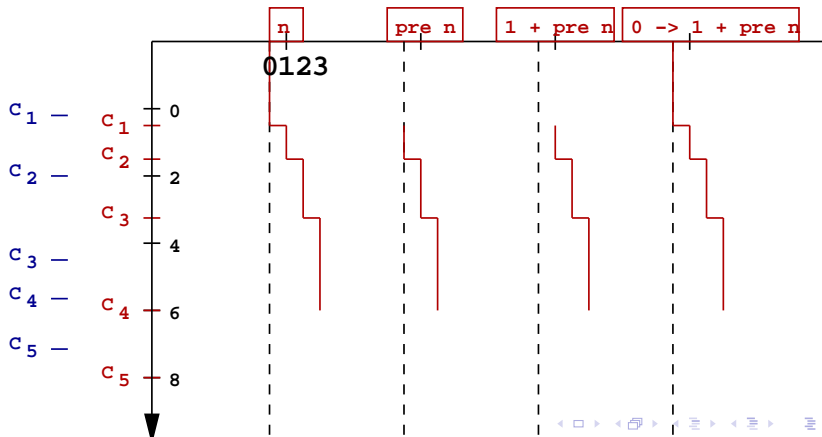


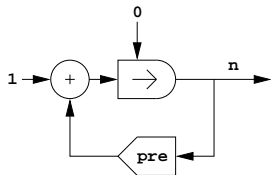
clock cycle : 1.5-2.5 ms



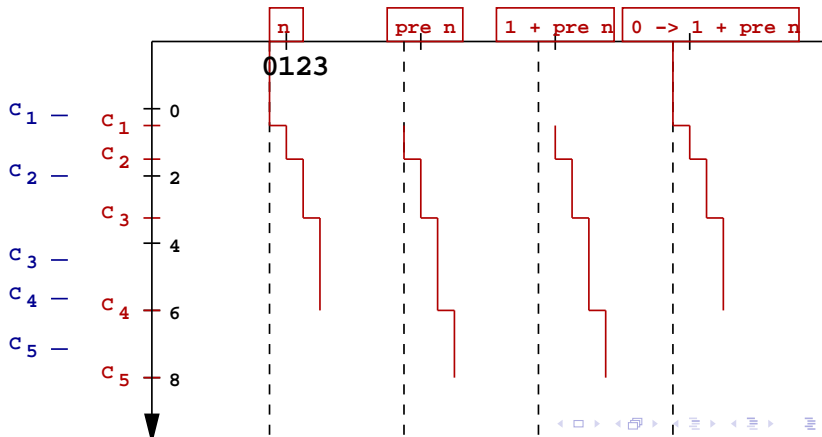


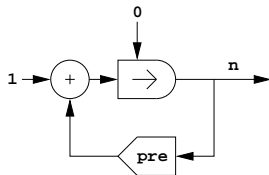
clock cycle : 1.5-2.5 ms



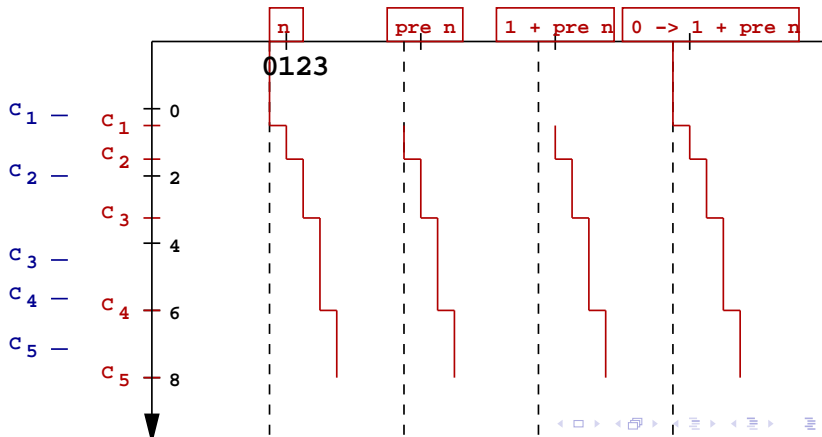


clock cycle : 1.5-2.5 ms

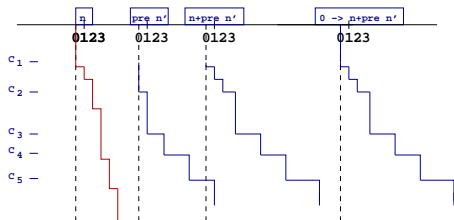
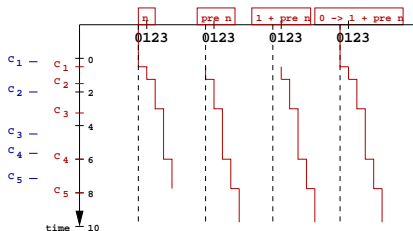
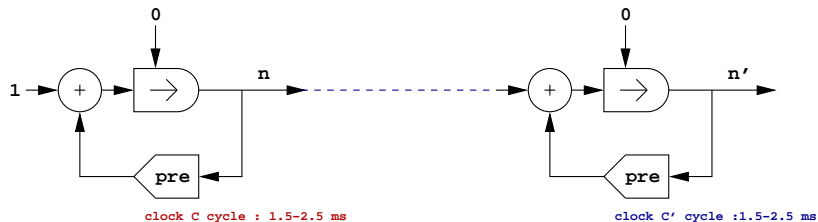




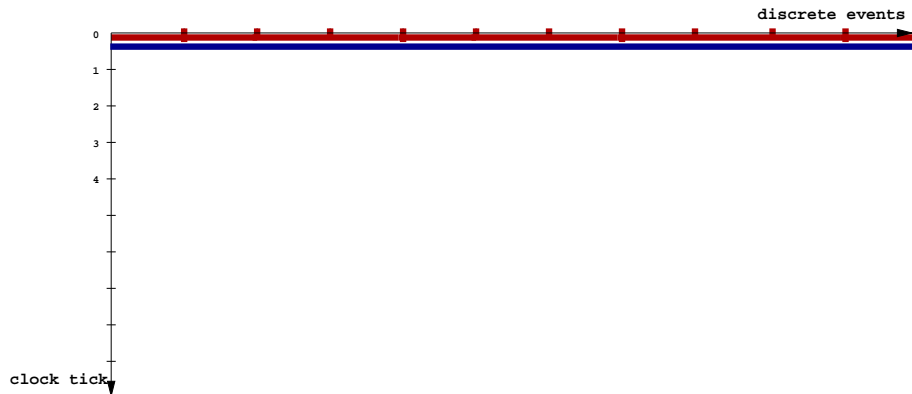
clock cycle : 1.5-2.5 ms



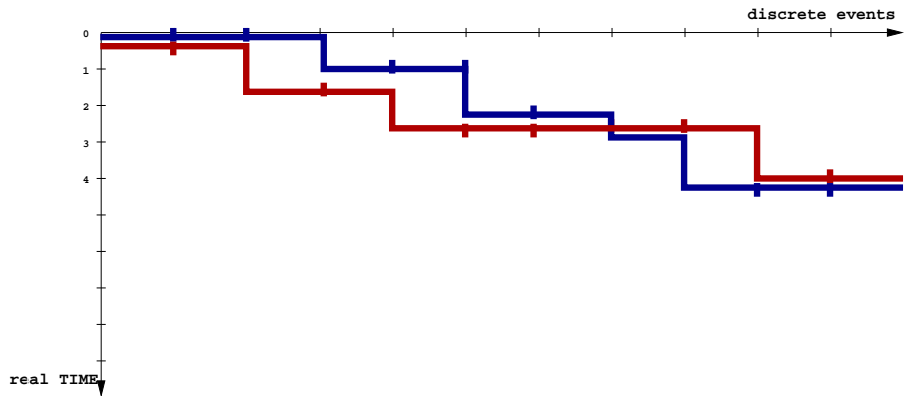
# Avec quelles sémantiques doit-on travailler



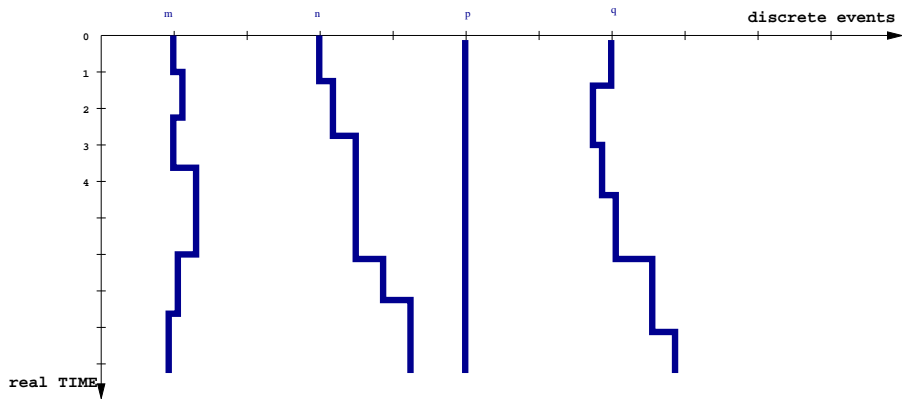
# Résumé des modifications de la sémantique



# Résumé des modifications de la sémantique



# Résumé des modifications de la sémantique



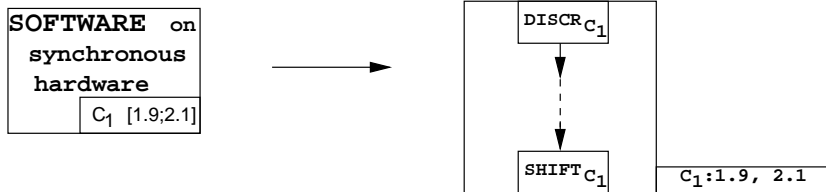
# Ancienne sémantique abstraction de nouvelle sémantique

$\alpha :$

$$\left( \begin{array}{l} \mathcal{P}(P \rightarrow (\mathbb{R}^+ \rightarrow \mathbb{N})) \rightarrow \mathcal{P}(P \rightarrow (\mathbb{N} \rightarrow \mathbb{N})) \\ \llbracket S \rrbracket : p \mapsto \llbracket S \rrbracket_p \mapsto p \mapsto \left\{ \begin{array}{l} \exists c_p \in \llbracket S \rrbracket_p \\ c_p(n \delta_p)_{n \in \mathbb{N}}, \delta_p \text{ période du sous-} \\ \text{système incluant } p \end{array} \right\} \end{array} \right)$$

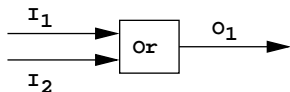
# Comportement d'un système synchrone

- une horloge est une fonction  $:\mathbb{N} \rightarrow \mathbb{R}^+$
- paramétrée par  $:\ [\alpha, \beta]$ , avec  $\alpha, \beta \in \mathbb{R}^+$  et  $0 < \alpha \leq \beta$
- une horloge  $c$  satisfait  $[\alpha, \beta]$  ssi  $c_{n+1} - c_n \in [\alpha, \beta]$



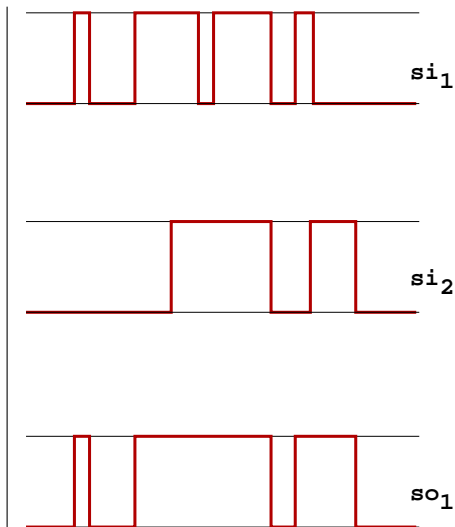
- $\text{DISCR}_{C_1}$  modélise la lecture périodique du blackboard à l'entrée du système
- $\text{SHIFT}_{C_1}$  modélise l'attente du prochain tick d'horloge, et l'émission du résultat lors de ce tick

## Sémantique des opérateurs atemporels

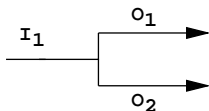


$$so_1(t) = \begin{cases} \bullet \text{ true} \\ \text{if } si_1(t) = \text{true} \\ \text{or } si_2(t) = \text{true} \\ \bullet \text{ false} \text{ else} \end{cases}$$

$$so_1 \triangleq \Psi_{OR}(si_1, si_2)$$



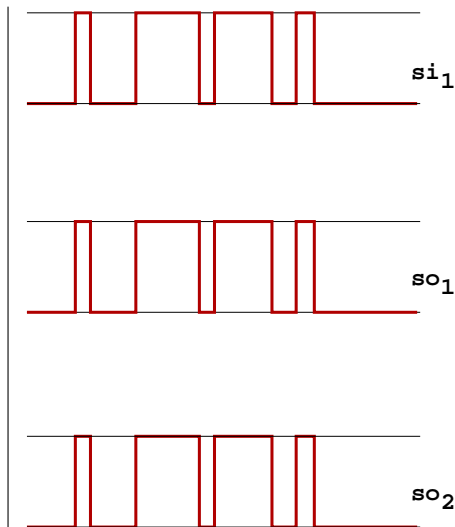
## Sémantique des opérateurs atemporels



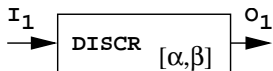
- $so_1(t) = si_1(t)$

- $so_2(t) = si_1(t)$

$$(so_1, so_2) \triangleq \Psi_{\text{DUPL}}(si_1)$$



## Sémantique des opérateurs temporels

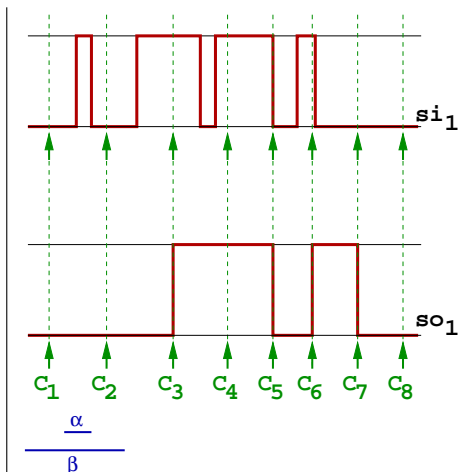


Il existe une horloge  $C$  de paramètre  $[\alpha, \beta]$

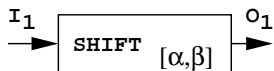
i.e. :  $c_{n+1} - c_n \in [\alpha, \beta]$  telle que

$$so_1(t) = \begin{cases} \bullet \text{ false} & \text{if } t < c(0) \\ \bullet \text{ } si_1(c_n) & \text{if } t \in [c_n, c_{n+1}) \end{cases}$$

$$so_1 \triangleq \Psi_{DISCR_{[\alpha, \beta]}}(si_1)$$



## Sémantique des opérateurs temporels

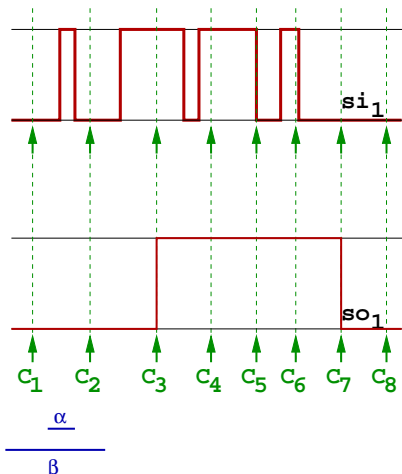


Il existe une horloge  $C$  de  
paramètre  $[\alpha, \beta]$

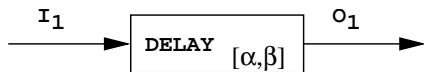
i.e. :  $c_{n+1} - c_n \in [\alpha, \beta]$  telle que

$$so_1(t) = \begin{cases} \bullet \text{ false} & \text{if } t < c(0) \\ \bullet \lim_{\substack{t \rightarrow c_n \\ t < c_n}} si_1(t) & \text{if } t \in [c_n, c_{n+1}) \end{cases}$$

$$so_1 \triangleq \Psi_{\text{SHIFT}_{[\alpha, \beta]}}(si_1)$$



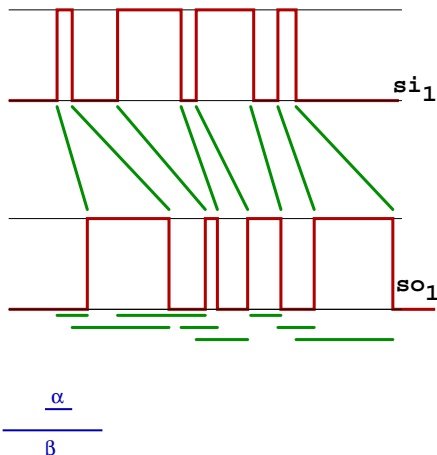
## Sémantique des opérateurs temporels



Il existe une fonction  $\delta$   
 bijection croissante de  $\mathbb{R} \rightarrow \mathbb{R}$   
 de paramètre  $[\alpha, \beta]$  i.e. :  
 $\forall t \in \mathbb{R}, \delta(t) - t \in [\alpha, \beta]$  telle que

$$so_1(\delta(t)) = si_1(t)$$

$$so_1 \triangleq \Psi_{\text{DELAY}_{[\alpha, \beta]}}(si_1)$$



## semantique concrète

- un point de contrôle  $\triangle$  équivalent graphique d'une variable
- L'ensemble des points de contrôle est noté  $P$ .

## semantique concrète

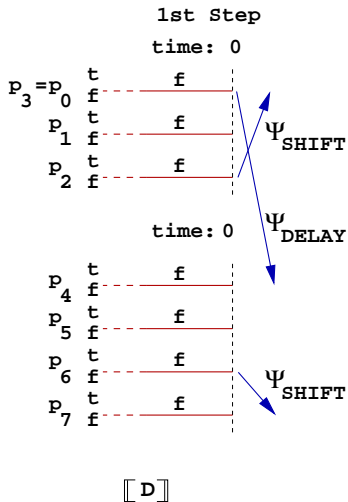
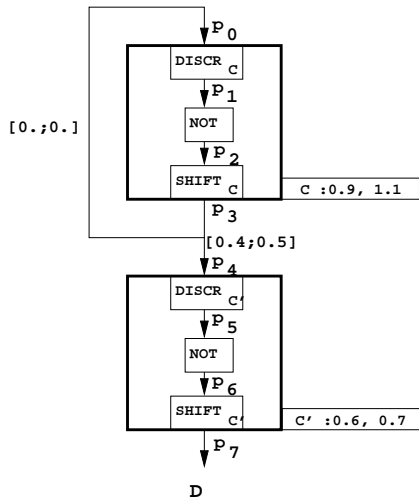
- un point de contrôle  $\triangleq$  equivalent graphique d'une variable
- L'ensemble des points de contrôle est noté  $P$ .
- $D$  : ensemble de systèmes synchrones,  $[[D]] \subseteq P \rightarrow (\mathbb{R}^+ \rightarrow \mathbb{B})$

# semantique concrète

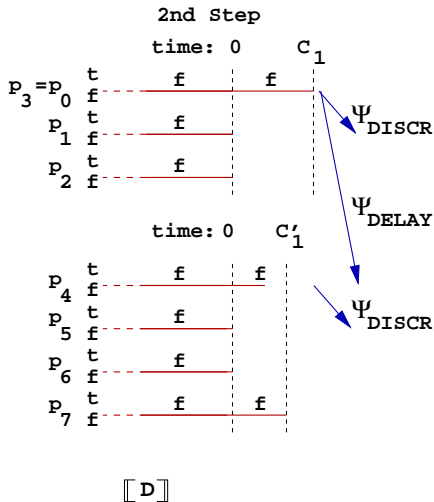
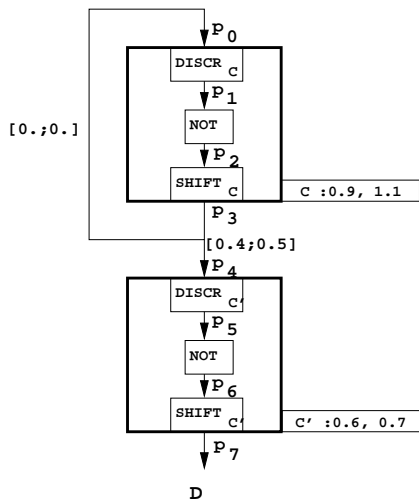
- un point de contrôle  $\triangleq$  équivalent graphique d'une variable
- L'ensemble des points de contrôle est noté  $P$ .
- $D$  : ensemble de systèmes synchrones,  $\llbracket D \rrbracket \subseteq P \rightarrow (\mathbb{R}^+ \rightarrow \mathbb{B})$

- $\left( \begin{array}{ccc} p_1 & \mapsto & s_1 \\ & \vdots & \\ p_{\#P-1} & \mapsto & s_{\#P-1} \end{array} \right) \in \llbracket D \rrbracket$  ssi il satisfait les équations de toutes les portes de  $D$ .

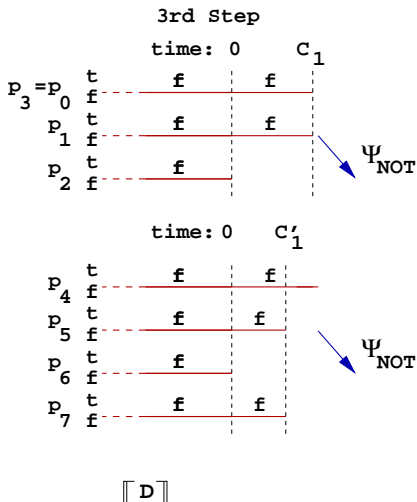
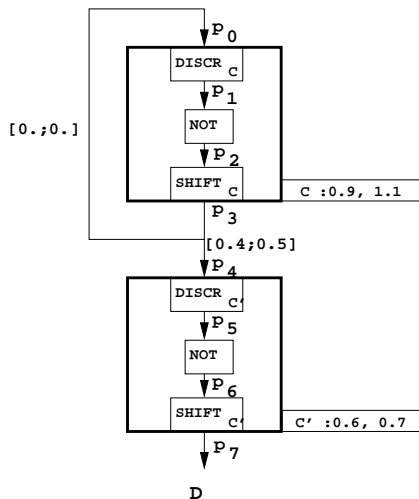
# Syntaxe et sémantique



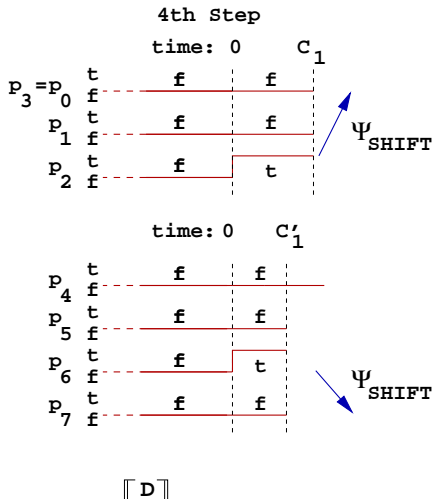
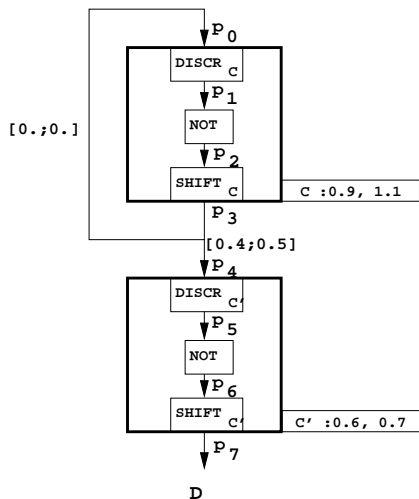
# Syntaxe et sémantique



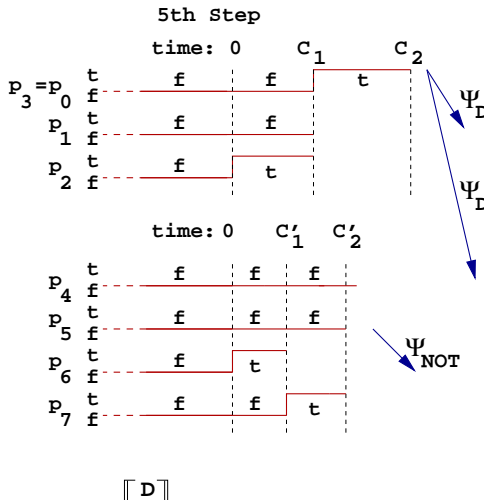
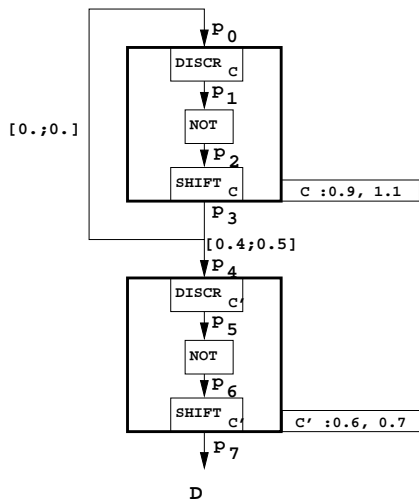
# Syntaxe et sémantique



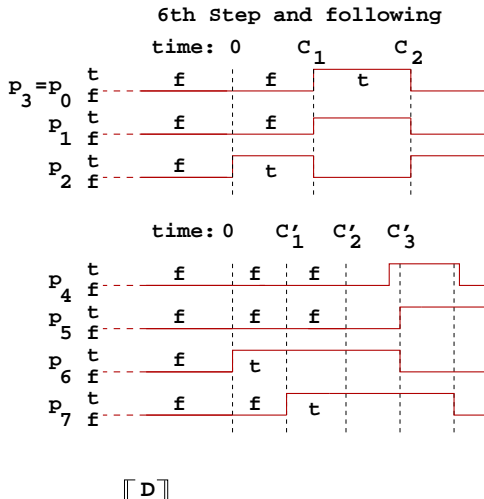
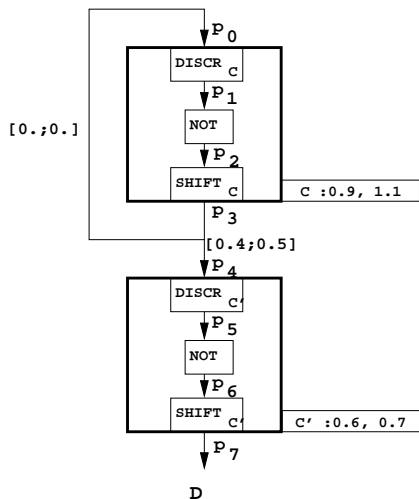
# Syntaxe et sémantique



# Syntaxe et sémantique



# Syntaxe et sémantique



## Vers une sémantique collectrice

- La sémantique définie est trop sensible aux approximations temporelles

## Vers une sémantique collectrice

- La sémantique définie est trop sensible aux approximations temporelles
- difficile à représenter et manipuler (par un ordinateur)

## Vers une sémantique collectrice

- La sémantique définie est trop sensible aux approximations temporelles
- difficile à représenter et manipuler (par un ordinateur)
- On effectue une abstraction canonique

$$\alpha : \left( \begin{array}{l} \mathcal{P}(P \rightarrow (\mathbb{R}^+ \rightarrow \mathbb{B})) \rightarrow P \rightarrow \mathcal{P}(\mathbb{R}^+ \rightarrow \mathbb{B}) \\ \llbracket S \rrbracket \mapsto p \mapsto \{f, \exists t \in \llbracket S \rrbracket, t(p) = f\} \end{array} \right)$$

## Vers une sémantique collectrice

- La sémantique définie est trop sensible aux approximations temporelles
- difficile à représenter et manipuler (par un ordinateur)
- On effectue une abstraction canonique

$$\alpha : \left( \begin{array}{l} \mathcal{P}(P \rightarrow (\mathbb{R}^+ \rightarrow \mathbb{B})) \rightarrow P \rightarrow \mathcal{P}(\mathbb{R}^+ \rightarrow \mathbb{B}) \\ \llbracket S \rrbracket \mapsto p \mapsto \{f, \exists t \in \llbracket S \rrbracket, t(p) = f\} \end{array} \right)$$

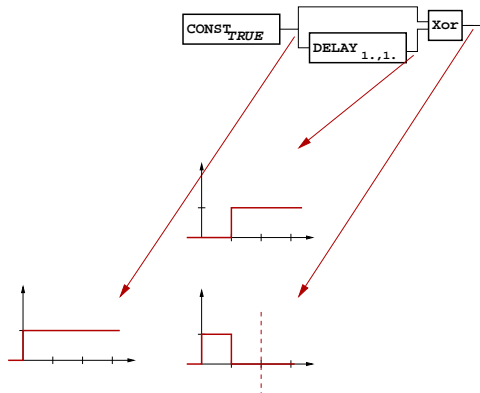
- malgré cette transformation, les comportements restent trop nombreux.

## Pourquoi une syntaxe graphique ?

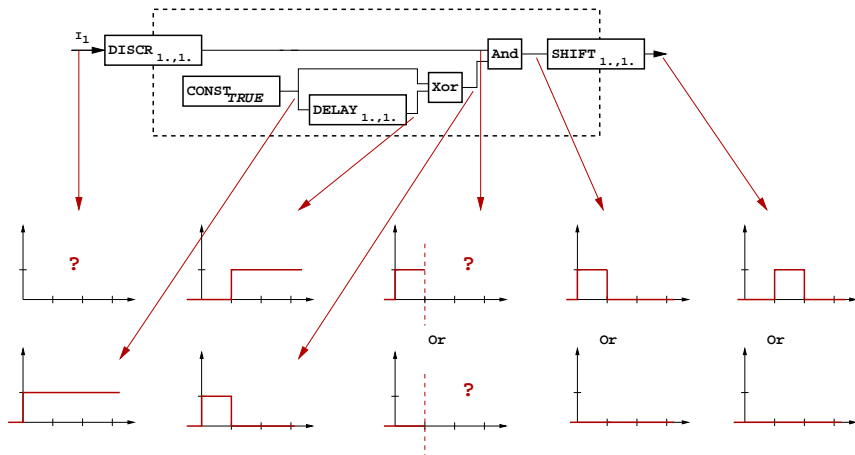
- Rappelle les circuit **électriques** dont les systèmes synchrones sont proches par la sémantique (temps-continu, équations)
- Rappelle les schémas de l'**automatique**. Or les programmeur de logiciels embarqués sont souvent des anciens automaticiens (la transition automatique → électronique n'a pas été brusque)
- SCADE est un langage graphique !
- permet d'identifier visuellement les parties “hardware” (et de quantifier leurs défauts) et les parties software.

# Analyse abstraite

# Analyse par regroupement

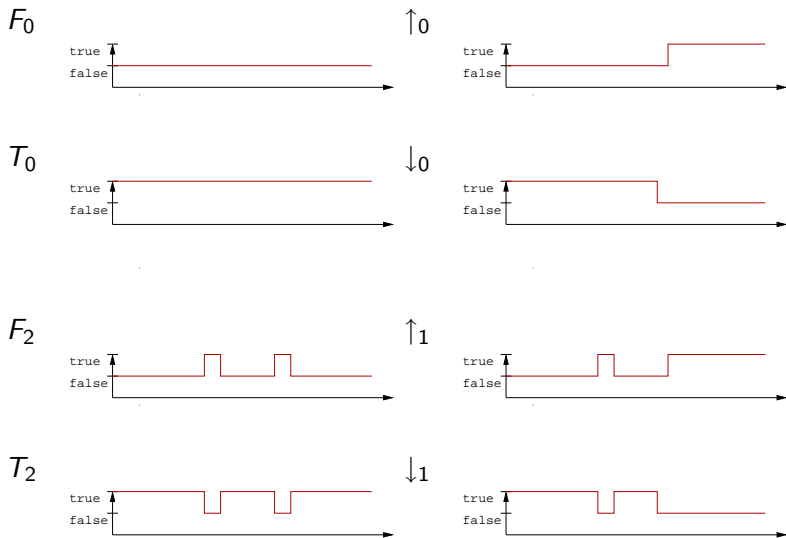


# Analyse par regroupement



- Manipulation possible de données inconnues ou “infinies”

# Abstraction Achrone



# Circuits Asynchrones

- proposé par Sarah Thompson et Alan Mycroft
- modélise de façon plus réaliste les circuits, en tenant compte des **glitches** :
  - courte pulsation qui peut induire une instabilité (par exemple sur un flip-flop)



&amp;



&amp;



# Circuits Asynchrones

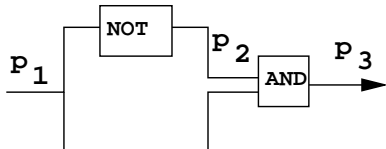
- **domaine concret**  $\mathbb{R} \dashrightarrow \mathbb{B}$  (i.e. finitairement constant par morceaux)
- **domaine abstrait**  
 $\{F_n, n \in \mathbb{N}\} \cup \{T_n, n \in \mathbb{N}\} \cup \{\uparrow_n, n \in \mathbb{N}\} \cup \{\downarrow_n, n \in \mathbb{N}\}$
- **fonction d'abstraction**

$$\alpha_0 \triangleq \left( \begin{array}{ll} \mathbb{R} \dashrightarrow \mathbb{B} & \rightarrow F_* \cup T_* \cup \uparrow_* \cup \downarrow_* \\ f \text{ such that } \forall i \in [1, n] & \mapsto F_p \text{ if } n = 2p \ \& \ f(-\infty) = \textit{false} \\ f(k_i) = \neg f(k_{i+1}) & T_p \text{ if } n = 2p \ \& \ f(-\infty) = \textit{true} \\ f([k_i, k_{i+1}[) = f(k_i) & \uparrow_p \text{ if } n = 2p + 1 \ \& \ f(-\infty) = \textit{false} \\ f(]-\infty, k_1]) = \neg f(k_1) & \downarrow_p \text{ if } n = 2p + 1 \ \& \ f(-\infty) = \textit{true} \end{array} \right)$$

Exemple : l'opérateur abstrait de conjonction  $\wedge$ 

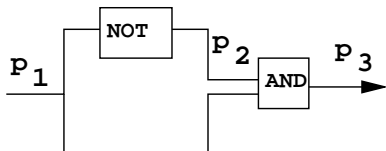
$\wedge^\#$	$F_0$	$F_n$	$T_0$	$T_n$	$\uparrow_0$	$\uparrow_n$	$\downarrow_0$	$\downarrow_n$
$F_0$	$F_0$	$F_0$	$F_0$	$F_0$	$F_0$	$F_0$	$F_0$	$F_0$
$F_m$	$F_0$	$F_{0..m+n-1}$	$F_m$	$F_{0..m+n}$	$F_{0..m}$	$F_{0..m+n}$	$F_{0..m}$	$F_{0..m+n}$
$T_0$	$F_0$	$F_n$	$T_0$	$T_n$	$\uparrow_0$	$\uparrow_n$	$\downarrow_0$	$\downarrow_n$
$T_m$	$F_0$	$F_{0..m+n}$	$T_m$	$T_{1..m+n}$	$\uparrow_{0..m}$	$\uparrow_{0..m+n}$	$\downarrow_m$	$\downarrow_{0..m+n}$
$\uparrow_0$	$F_0$	$F_{0..n}$	$\uparrow_0$	$\uparrow_{0..n}$	$\uparrow_0$	$\uparrow_{0..n}$	$F_{0..1}$	$F_{0..n+1}$
$\uparrow_m$	$F_0$	$F_{0..m+n}$	$\uparrow_m$	$\uparrow_{0..m+n}$	$\uparrow_{0..m}$	$\uparrow_{0..m+n}$	$F_{0..m+1}$	$F_{0..m+n+1}$
$\downarrow_0$	$F_0$	$F_{0..n}$	$\downarrow_0$	$\downarrow_n$	$F_{0..1}$	$F_{0..n+1}$	$\downarrow_0$	$\downarrow_{0..n}$
$\downarrow_m$	$F_0$	$F_{0..m+n}$	$\downarrow_m$	$\downarrow_{0..m+n}$	$F_{0..m+1}$	$F_{0..m+n+1}$	$\downarrow_{0..m}$	$\downarrow_{0..m+n}$

Exemple : prouver  $a \wedge \neg a = \text{false}$



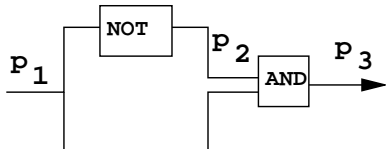
$p_1$	$p_2 = \neg p_1$	$p_3 = p_1 \wedge \neg p_1$
$F_0$		
$T_0$		
$\uparrow_0$		
$\downarrow_0$		
$F_n$		
$T_n$		
$\uparrow_n$		
$\downarrow_n$		

Exemple : prouver  $a \wedge \neg a = \text{false}$



$p_1$	$p_2 = \neg p_1$	$p_3 = p_1 \wedge \neg p_1$
$F_0$	$T_0$	
$T_0$	$F_0$	
$\uparrow_0$	$\downarrow_0$	
$\downarrow_0$	$\uparrow_0$	
$F_n$	$T_n$	
$T_n$	$F_n$	
$\uparrow_n$	$\downarrow_n$	
$\downarrow_n$	$\uparrow_n$	

Exemple : prouver  $a \wedge \neg a = \text{false}$

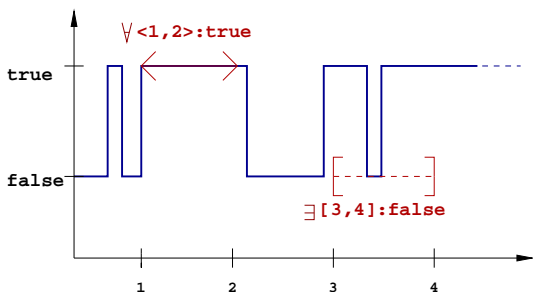


$p_1$	$p_2 = \neg p_1$	$p_3 = p_1 \wedge \neg p_1$
$F_0$	$T_0$	$F_0$
$T_0$	$F_0$	$F_0$
$\uparrow_0$	$\downarrow_0$	$F_{0\dots 1}$
$\downarrow_0$	$\uparrow_0$	$F_{0\dots 1}$
$F_n$	$T_n$	$F_{0\dots 2n}$
$T_n$	$F_n$	$F_{0\dots 2n}$
$\uparrow_n$	$\downarrow_n$	$F_{0\dots 2n+1}$
$\downarrow_n$	$\uparrow_n$	$F_{0\dots 2n+1}$

# Difficultés de l'analyse abstraite dans le cas des systèmes imparfaitement synchrones

- abstraction totale du temps
- abstraction des relation temporelles : événement  $a$  avant  $b$

# 1er domaine abstrait : les contraintes



- La constraint  $\exists[a; b] : x$  garantit que les signaux prennent la valeur  $x$  **au moins une fois** pendant  $[a; b]$ .
- La constraint  $\forall\langle a; b \rangle : x$  garantit que les signaux prennent la valeur  $x$  **durant tout l'intervalle**  $[a; b]$ .

# 1er domaine abstrait : les contraintes

- Permet d'exprimer de nombreuses **propriétés temporelles**
- Le signal  $s$  prend la valeur *true* pendant plus de 3 secondes sans que l'alarme  $a$  soit déclenchée moins d'une seconde plus tard :

$$s : \forall \langle \delta; \delta + 3 \rangle : True \wedge a : \forall \langle \delta + 3; \delta + 4 \rangle : False$$

# Domaine Abstrait and Concrétisation

- **L'ensemble des contraintes** est noté  $\mathcal{Z}$ .
- **Conjonctions de contraintes** :  $\mathcal{F}_\wedge = \left\{ \bigwedge_{c_j \in J} c_j, J \subseteq \mathcal{Z} \right\}$
- **Domaine abstrait** :  $P \rightarrow \mathcal{F}_\wedge$ .

# Domaine Abstrait and Concrétisation

- **L'ensemble des contraintes** est noté  $\mathcal{Z}$ .

$$\gamma_{\mathcal{Z}} = \left( \begin{array}{ll} \mathcal{Z} & \rightarrow \mathcal{P}(S) \\ \exists[a; b] : y & \mapsto \{s : \mathbb{R} \rightarrow \mathbb{B}, \exists t \in [a, b], s(t) = y\} \\ \forall[a; b] : y & \mapsto \{s : \mathbb{R} \rightarrow \mathbb{B}, \forall t \in [a, b], s(t) = y\} \end{array} \right)$$

- **Conjonctions de contraintes** :  $\mathcal{F}_{\wedge} = \left\{ \bigwedge_{C_j \in J} C_j, J \subseteq \mathcal{Z} \right\}$

$$\gamma_{\mathcal{F}_{\wedge}}(\bigwedge_i C_j) = \bigcap_j \gamma_{\mathcal{Z}}(C_j)$$

- **Domaine abstrait** :  $P \rightarrow \mathcal{F}_{\wedge}$ .

$$\gamma_{P \rightarrow \mathcal{F}_{\wedge}} :$$

$$\left( \begin{array}{ll} (P \rightarrow \mathcal{F}_{\wedge}) & \rightarrow \mathcal{P}(P \rightarrow S) \\ \lambda p. f_{\wedge} p & \mapsto \{ \lambda p. f_{j,p}, \text{ tel que } \forall j \forall p, f_{j,p} \in \gamma_{\mathcal{F}_{\wedge}}(f_{\wedge} p) \} \end{array} \right)$$

## Exemples

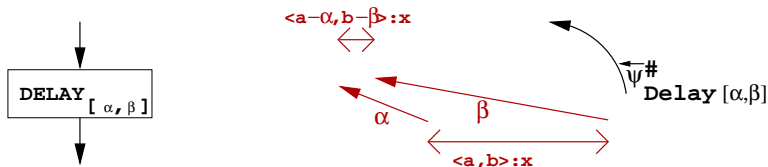
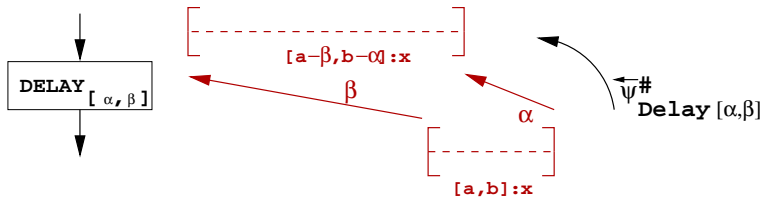
$$\bullet \gamma \left( \begin{array}{l} 0 \mapsto \exists[0;1] : \text{True} \quad \wedge \quad \exists[0;1] : \text{False} \\ 1 \mapsto \forall\langle 2;3 \rangle : \text{False} \end{array} \right)$$

$$\triangleq \left\{ \begin{array}{l} 0 \mapsto f : \mathbb{R}^+ \mapsto \mathbb{B} \text{ changeant sa valeur entre 0 et 1} \\ 1 \mapsto g : \mathbb{R}^+ \mapsto \mathbb{B} \text{ stable entre 2 et 3} \end{array} \right\}$$

$$\bullet \gamma \left( \begin{array}{l} 0 \mapsto \forall\langle 0;0 \rangle : \text{True} \quad \wedge \quad \forall\langle 0;0 \rangle : \text{False} \\ 1 \mapsto \forall\langle 2;3 \rangle : \text{False} \end{array} \right)$$

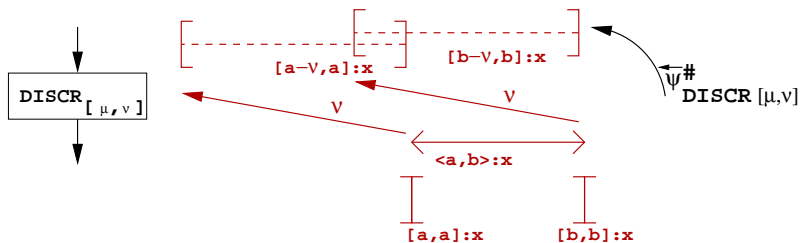
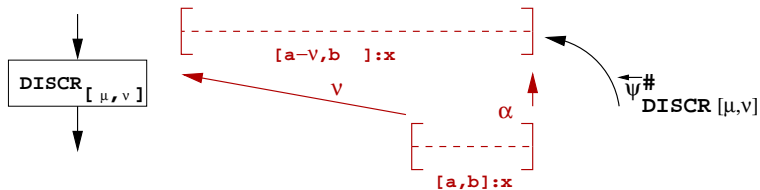
$$\subseteq \left\{ \begin{array}{l} 0 \mapsto f : \mathbb{R}^+ \mapsto \mathbb{B}, f(0) = \text{true} \wedge f(0) = \text{false} \\ 1 \mapsto g : \mathbb{R}^+ \mapsto \mathbb{B} \text{ stable entre 2 et 3} \end{array} \right\} = \emptyset$$

## Opérateurs Abstraits et Contraintes : exemple



- $\overleftarrow{\Psi}^{\#}_{\text{DELAY}[\alpha, \beta]}(\exists \langle a; b \rangle : x) \triangleq \exists \langle a - \beta; b - \alpha \rangle : x$
- $\overleftarrow{\Psi}^{\#}_{\text{DELAY}[\alpha, \beta]}(\forall \langle a; b \rangle : x) \triangleq \forall \langle a - \alpha; b - \beta \rangle : x$

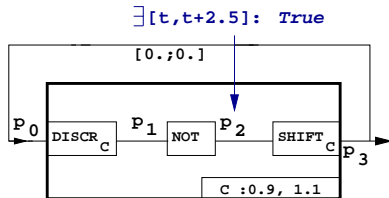
## Opérateurs Abstraits et Contraintes : exemple

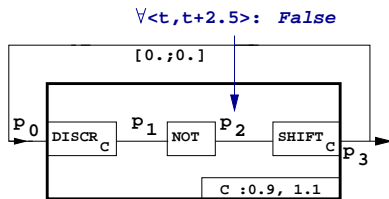


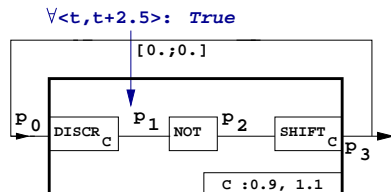
- $\overleftarrow{\Psi}^{\#}_{\text{DISCR}_{[\mu, \nu]}}(\exists[a; b] : x) \triangleq \exists[a - \nu; b] : x$
- $\overleftarrow{\Psi}^{\#}_{\text{DISCR}_{[\mu, \nu]}}(\forall \langle a; b \rangle : x) \triangleq \bigwedge_{u \in [a, b]} \exists[u - \nu; u] : x$

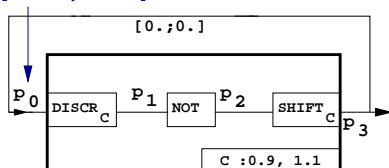
# Analyse dans le domaine abstrait des **Contraintes**

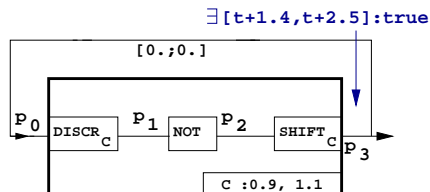
- Prouver la propriété abstraite suivante.



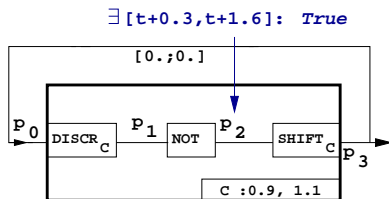
Analyse dans le domaine abstrait des **Contraintes**

Analyse dans le domaine abstrait des **Contraintes**

Analyse dans le domaine abstrait des **Contraintes**
 $\exists [t+1.4, t+2.5]: \text{True}$ 


Analyse dans le domaine abstrait des **Contraintes**

# Analyse dans le domaine abstrait des **Contraintes**



Au point  $p_2$  : 2 contraintes doivent être satisfaites :

- $\forall \langle t; t + 2.5 \rangle : False$
- $\exists [t; t + 2.5] : True$

ce qui est impossible et invalide donc l'hypothèse initiale  
 $\forall \langle t; t + 2.5 \rangle : False$ .

- Donc  $\exists [t; t + 2.5] : True$  est certifié.

## Analyse par interprétation abstraite

- $\llbracket D \rrbracket$  : sémantique de l'ensemble de systèmes  $D$ .
- Pour toute propriété  $P$ ,  $[P]$  est l'ensemble de comportements satisfaisant  $P$ .
- **Ancien objectif** : Prouver que  $\llbracket D \rrbracket \subseteq [P]$ .

# Analyse par interprétation abstraite

- $\llbracket D \rrbracket$  : sémantique de l'ensemble de systèmes  $D$ .
- Pour toute propriété  $P$ ,  $[P]$  est l'ensemble de comportements satisfaisant  $P$ .
- **Ancien objectif** : Prouver que  $\llbracket D \rrbracket \subseteq [P]$ .
- **Alternative** : Prouver que  $\llbracket D \rrbracket \cap [\neg P] = \emptyset$ .

## Analyse par interprétation abstraite

- $\llbracket D \rrbracket$  : sémantique de l'ensemble de systèmes  $D$ .
- Pour toute propriété  $P$ ,  $[P]$  est l'ensemble de comportements satisfaisant  $P$ .
- **Ancien objectif** : Prouver que  $\llbracket D \rrbracket \subseteq [P]$ .
- **Alternative** : Prouver que  $\llbracket D \rrbracket \cap [\neg P] = \emptyset$ .
- **Or** :

$$\llbracket D \rrbracket \cap [\neg P] \subseteq \text{gfp}_{[\neg P]}(\Psi \cap Id)$$

## Analyse par interprétation abstraite

- $\llbracket D \rrbracket$  : sémantique de l'ensemble de systèmes  $D$ .
- Pour toute propriété  $P$ ,  $[P]$  est l'ensemble de comportements satisfaisant  $P$ .

- **Ancien objectif** : Prouver que  $\llbracket D \rrbracket \subseteq [P]$ .

- **Alternative** : Prouver que  $\llbracket D \rrbracket \cap [\neg P] = \emptyset$ .

- **Or** :

$$\llbracket D \rrbracket \cap [\neg P] \subseteq \text{gfp}_{[\neg P]}(\Psi \cap Id)$$

- **Nouvel objectif** :

$$\text{gfp}_{[\neg P]}(\Psi \cap Id) \subseteq^? \emptyset$$

# Analyse par interprétation abstraite

Théorème : Si :

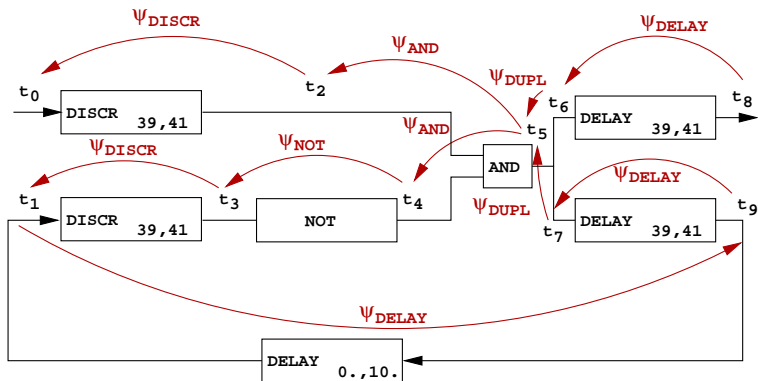
- $F$  est continue.
- $F^\#$  est continue.
- $F \circ \gamma \subseteq \gamma \circ F^\#$

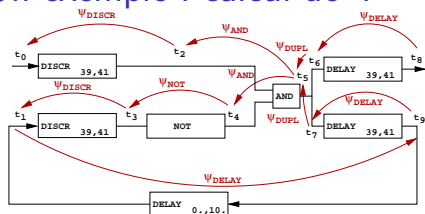
alors :  $\text{gfp}F \subseteq \gamma(\text{gfp}F^\#)$

- **Nouvel objectif**, avec  $F \triangleq \Psi \cap Id$  :

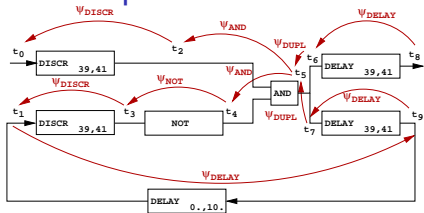
$$\text{gfp}_{[-P]}(\Psi^\# \cap^\# Id^\#) \subseteq^\# \emptyset^\# = \perp$$

# Propagation de l'information abstraite



Un exemple : calcul de  $\Psi$ 

$$\vec{T} = \Psi \begin{pmatrix} t_0 \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \\ t_6 \\ t_7 \\ t_8 \\ t_9 \end{pmatrix} \quad (*)$$

Un exemple : calcul de  $\Psi$ 

$$\vec{T} = \Psi \begin{pmatrix} t_0 \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \\ t_6 \\ t_7 \\ t_8 \\ t_9 \end{pmatrix} \quad (*)$$

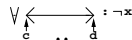
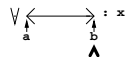
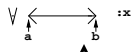
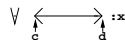
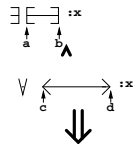
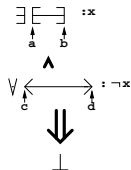
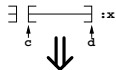
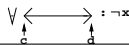
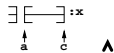
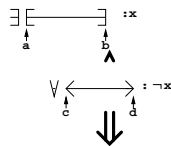
$$(*) \Leftrightarrow \vec{T} =$$

$$\begin{pmatrix} \Psi_{\text{DISCR}_{[39,41]}}(t_2) \\ \Psi_{\text{DISCR}_{[39,41]}}(t_3) \\ p_1(\Psi_{\text{AND}}(t_5)) \\ \Psi_{\text{NON}}(t_4) \\ p_2(\Psi_{\text{AND}}(t_5)) \\ \Psi_{\text{DUPL}}(t_6, t_7) \\ \Psi_{\text{DELAY}_{[39-41]}}(t_8) \\ \Psi_{\text{DELAY}_{[39-41]}}(t_9) \\ t_8 \\ \Psi_{\text{DELAY}_{[0-10]}}(t_1) \end{pmatrix}$$

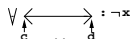
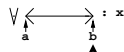
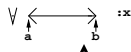
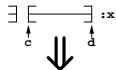
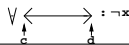
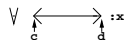
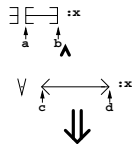
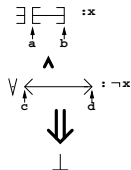
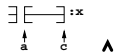
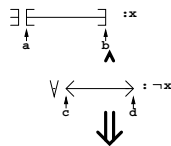
$$(\Psi_G \cap Id) \circ \gamma \subseteq \gamma \circ (\Psi_G \cap Id)^\#$$

$$\begin{array}{c}
 \left( \begin{array}{c} i_0 \mapsto C_0^\# \\ \vdots \\ i_n \mapsto C_n^\# \end{array} \right) \\
 \swarrow \gamma \\
 \left\{ \left( \begin{array}{c} i_0 \mapsto s_0^\# \\ \vdots \\ i_n \mapsto s_n^\# \end{array} \right), \begin{array}{c} s_0 \in \dot{\gamma}(C_0^\#) \\ \vdots \\ s_n \in \dot{\gamma}(C_n^\#) \end{array} \right\} \\
 \searrow \Psi^\# \cap Id^\# \\
 \left( \begin{array}{c} i_0 \mapsto C_0^\# \\ \vdots \\ i_j \mapsto C_j^\# \dot{\cap} \dot{\Psi}_G^\#(C_k^\#) \\ \vdots \\ i_k \mapsto C_k^\# \\ \vdots \\ i_n \mapsto C_n^\# \end{array} \right) \\
 \xrightarrow{\gamma} \\
 \left\{ \left( \begin{array}{c} i_0 \mapsto s_0^\# \\ \vdots \\ i_j \mapsto s_j^\# \\ \vdots \\ i_k \mapsto s_k^\# \\ \vdots \\ i_n \mapsto s_n^\# \end{array} \right), \begin{array}{c} s_0 \in \dot{\gamma}(C_0^\#) \\ \vdots \\ s_j \in \dot{\gamma}(C_j^\# \dot{\cap} \dot{\Psi}_G^\#(C_k^\#)) \\ \vdots \\ s_k \in \dot{\gamma}(C_k^\#) \\ \vdots \\ s_n \in \dot{\gamma}(C_n^\#) \end{array} \right\} \\
 \cap \\
 \left\{ \left( \begin{array}{c} i_0 \mapsto s_0^\# \\ \vdots \\ i_j \mapsto s_j^\# \\ \vdots \\ i_k \mapsto s_k^\# \\ \vdots \\ i_n \mapsto s_n^\# \end{array} \right), \begin{array}{c} s_0 \in \dot{\gamma}(C_0^\#) \\ \vdots \\ s_j \in \dot{\gamma}(C_j^\#) \\ s_j \in \dot{\Psi}_G^\#(s_k) \\ \vdots \\ s_k \in \dot{\gamma}(C_k^\#) \\ \vdots \\ s_n \in \dot{\gamma}(C_n^\#) \end{array} \right\}
 \end{array}$$

## Opérations sur les contraintes : intersection abstraite

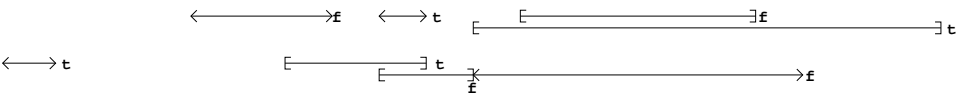


## Opérations sur les contraintes : intersection abstraite



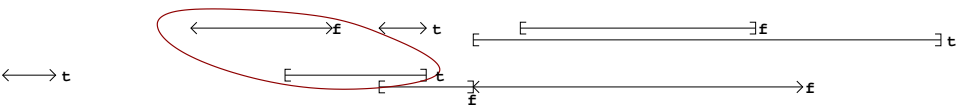
- 2 objectifs : terminer ! (tester  $\subseteq \# \emptyset \#$ ) et analyse plus rapide

# Conjonction optimisée sur les Contraintes

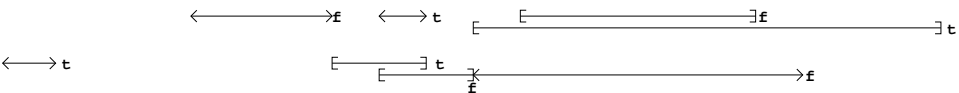




# Conjonction optimisée sur les Contraintes

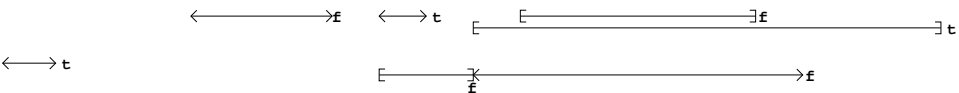


# Conjonction optimisée sur les Contraintes

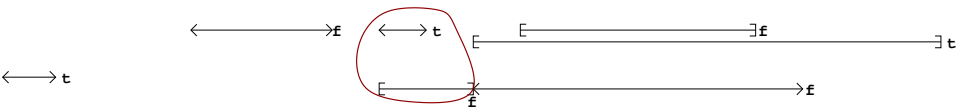




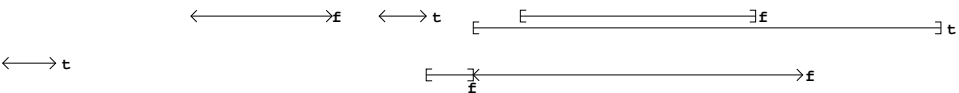
# Conjonction optimisée sur les Contraintes



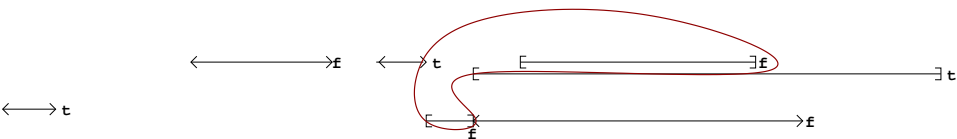
# Conjonction optimisée sur les Contraintes



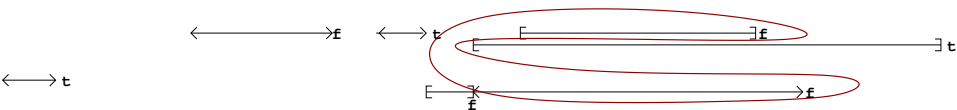
# Conjonction optimisée sur les Contraintes



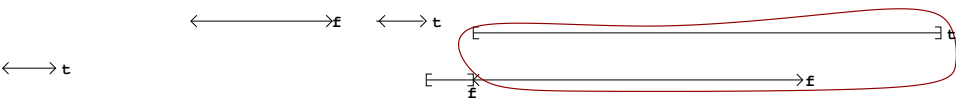
# Conjonction optimisée sur les Contraintes



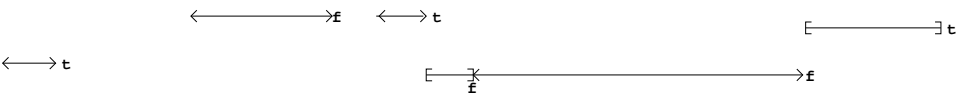
# Conjonction optimisée sur les Contraintes



# Conjonction optimisée sur les Contraintes

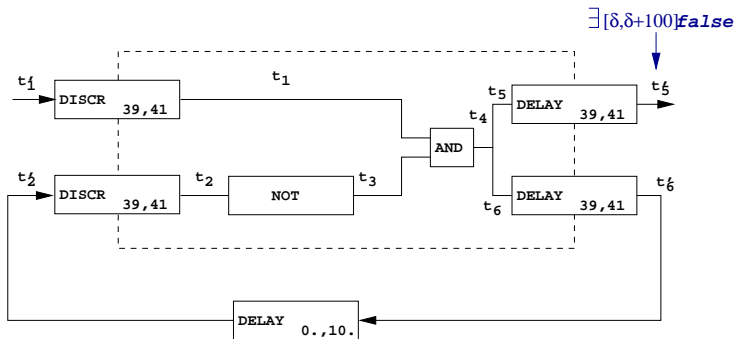


# Conjonction optimisée sur les Contraintes

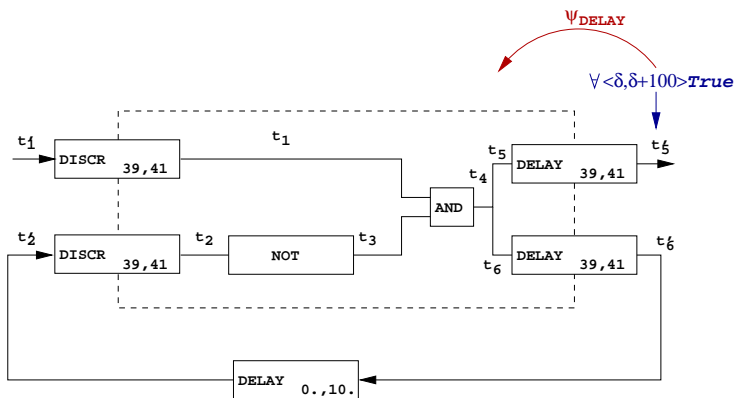


# Exemple d'analyse : Itération jusqu'au point fixe vide

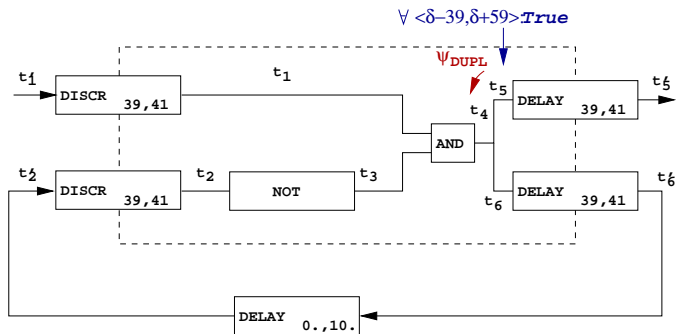
- prouver la propriété abstraite suivante



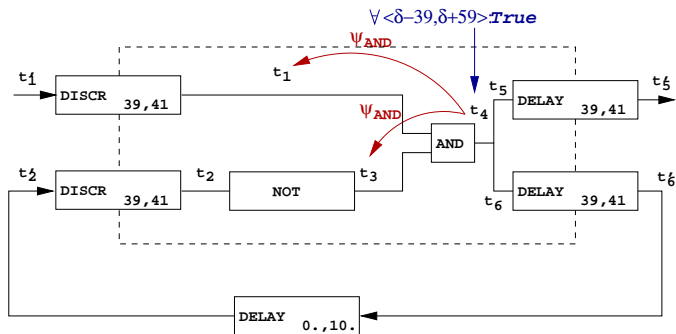
## Exemple : Itération jusqu'au point fixe (vide ?)



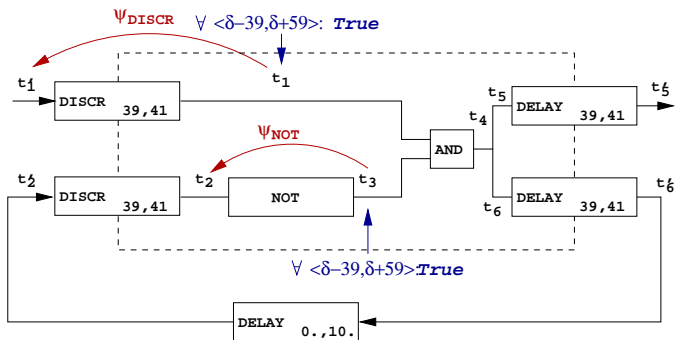
## Exemple : Itération jusqu'au point fixe (vide ?)



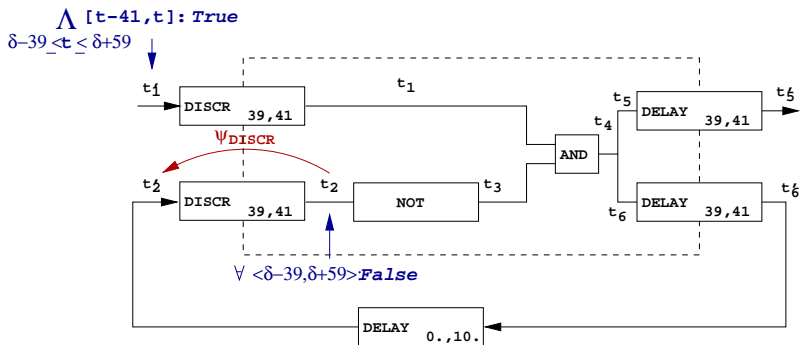
## Exemple : Itération jusqu'au point fixe (vide ?)



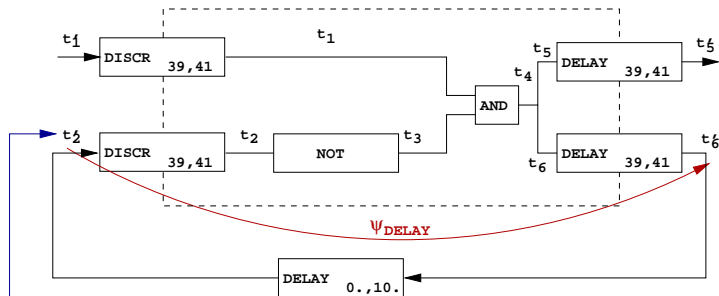
## Exemple : Itération jusqu'au point fixe (vide ?)



## Exemple : Itération jusqu'au point fixe (vide ?)

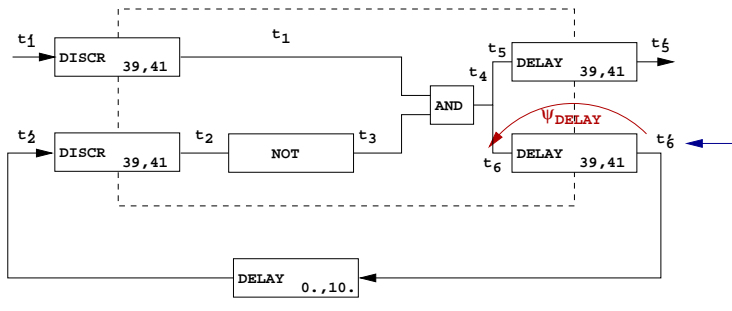


## Exemple : Itération jusqu'au point fixe (vide ?)



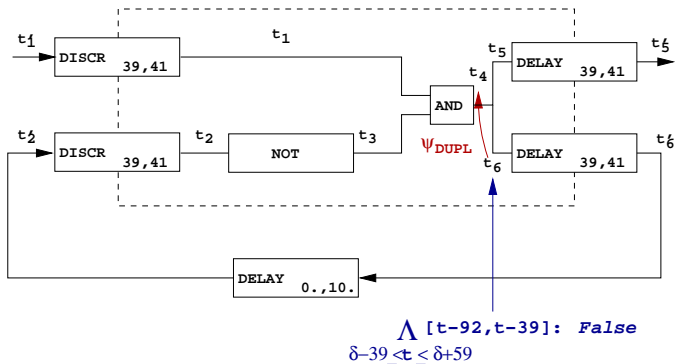
$\bigwedge [t-41, t]: \text{False}$   
 $\delta-39 \leq t \leq \delta+59$

## Exemple : Itération jusqu'au point fixe (vide ?)

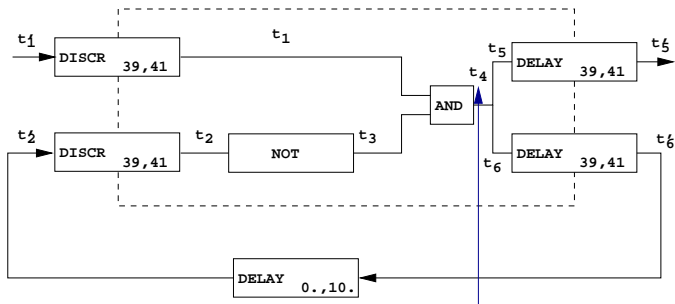


$\bigwedge [t-51, t]: \text{False}$   
 $\delta-39 \leq t \leq \delta+59$

## Exemple : Itération jusqu'au point fixe (vide ?)



## Exemple : Itération jusqu'au point fixe (vide ?)



$$\bigwedge [t-92, t-39]: \text{False}$$

$$\delta-39 \leq t \leq \delta+59$$

## Exemple : Itération jusqu'à un point fixe vide

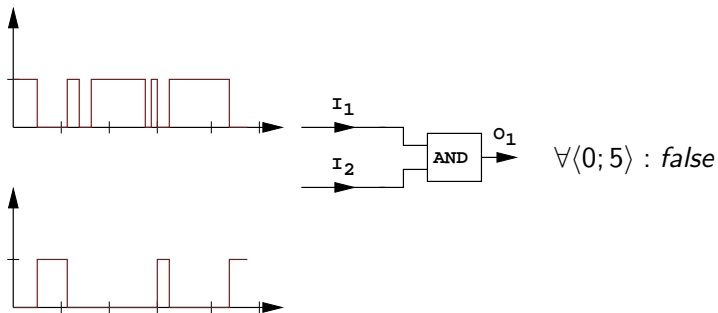
Les signaux doivent donc satisfaire au point de contrôle  $t_4$  :

$$\forall \langle \delta - 39; \delta + 59 \rangle : \textit{True} \textbf{ and } \bigwedge_{\delta - 39 \leq t \leq \delta + 59} ([t - 92, t - 39] : \textit{False})$$

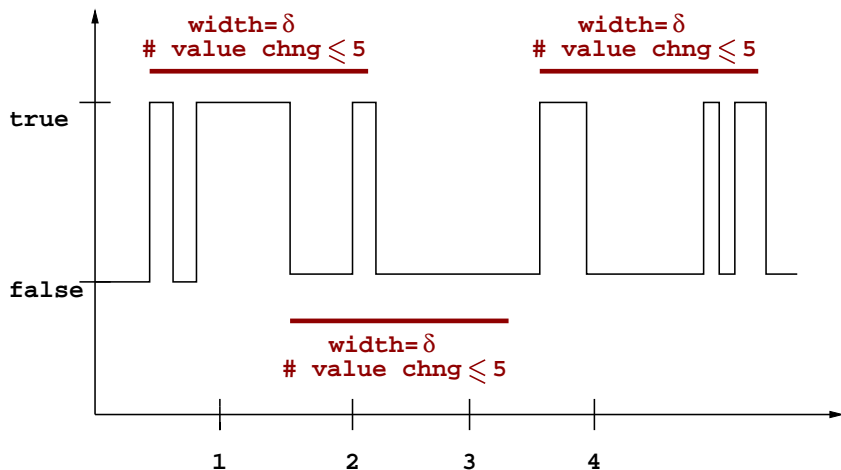
ce qui implique  $[\delta - 33 = \delta + 59 - 92, \delta + 20 = \delta + 59 - 39] : \textit{False}$

# Faiblesses du domaine des contraintes

- Domaine précis dans le cas de : DELAY, DISCR, SHIFT, NOT,
- Grande perte de précision dans le cas de : AND, OR, XOR



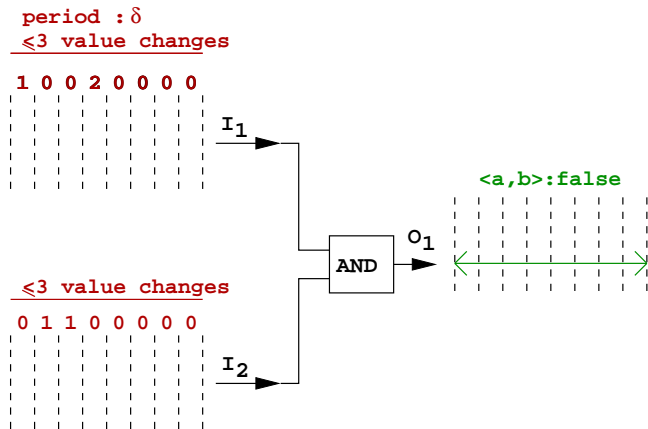
## 2ème Domaine Abstrait : Domaine du comptage des Changements de valeurs



- Permet d'exprimer des spécifications de "stabilité" des variables.

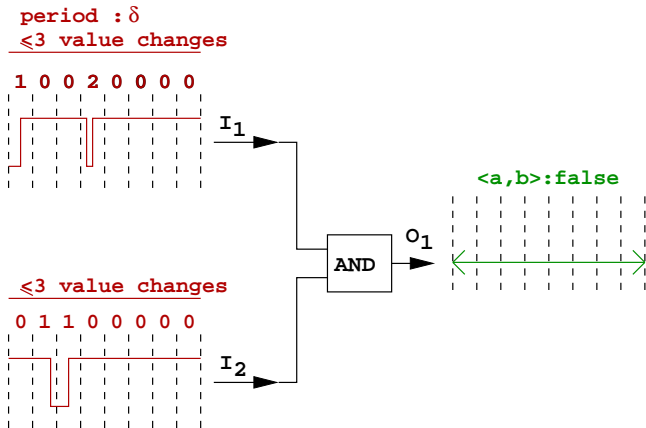
## 2ème Domaine Abstrait : Domaine du comptage des Changements de valeurs

### Utilisation



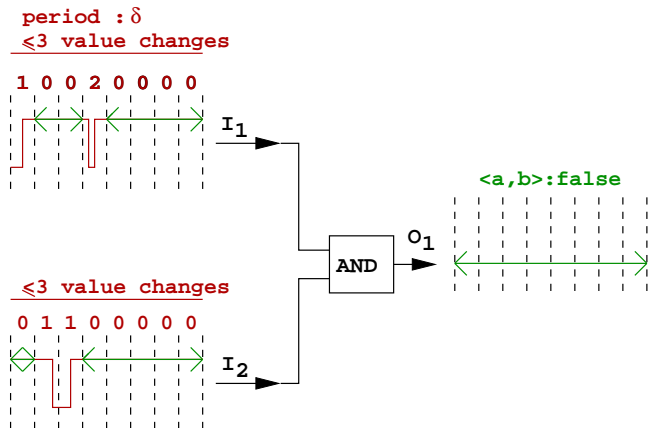
## 2ème Domaine Abstrait : Domaine du comptage des Changements de valeurs

### Utilisation

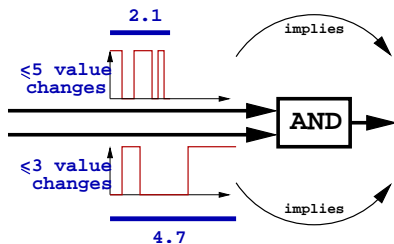


## 2ème Domaine Abstrait : Domaine du comptage des Changements de valeurs

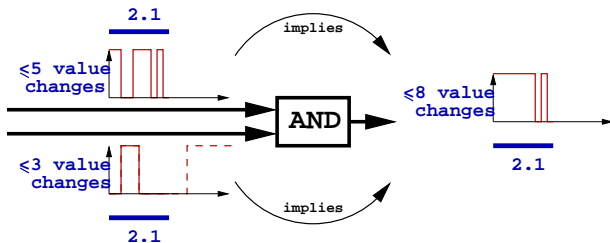
### Utilisation



# un opérateur abstrait **atemporel**

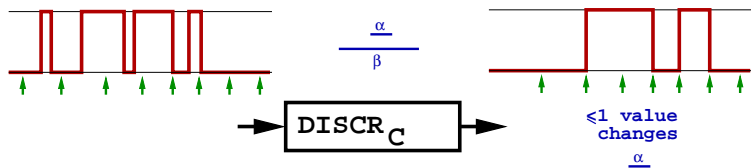


# un opérateur abstrait **atemporel**



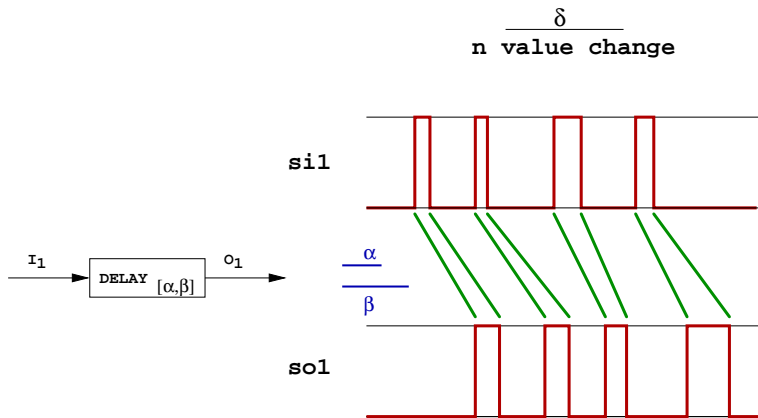
$$\vec{\Psi}_{\text{AND}}^{\#}((n_1, \delta_1)_{\mathcal{N}}, (n_2, \delta_2)_{\mathcal{N}}) \triangleq (\tilde{n}_1 + \tilde{n}_2, \tilde{\delta}_1)_{\mathcal{N}}$$

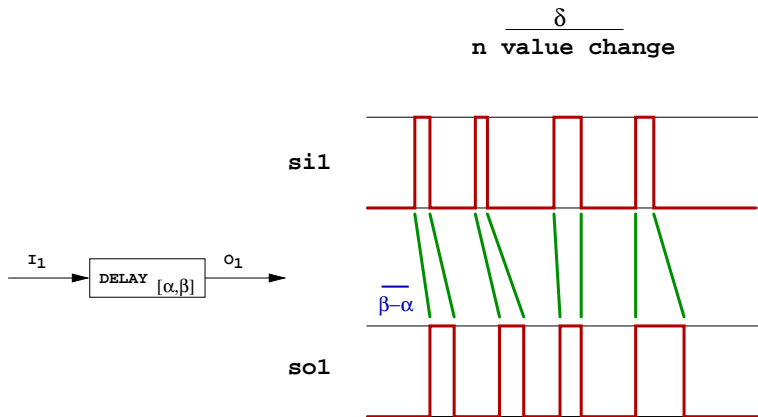
# un opérateur abstrait temporel



- $[\alpha, \beta]$  paramètre de l'horloge  $C$
- $\vec{\Psi}_{\text{DISCR}_{[\alpha, \beta]}}^{\#}(-) \triangleq (1, \alpha)$

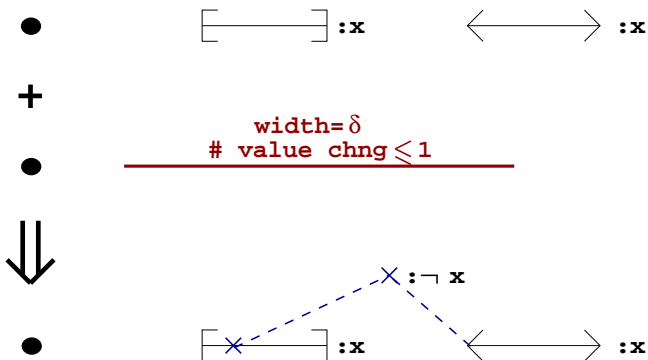
# un opérateur abstrait temporel



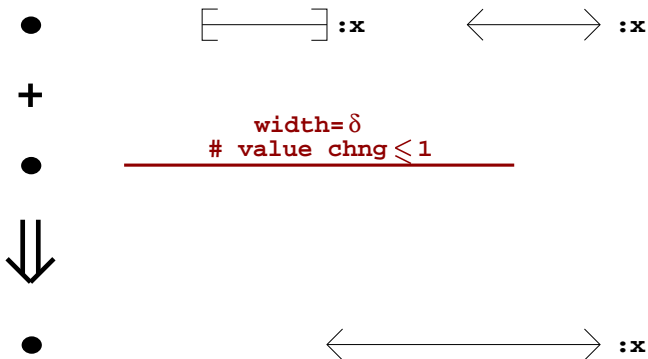
un opérateur abstrait **temporel**

$$\vec{\Psi}^\#(n, \delta)_{\mathcal{N}} \triangleq (n, \delta - \beta + \alpha)$$

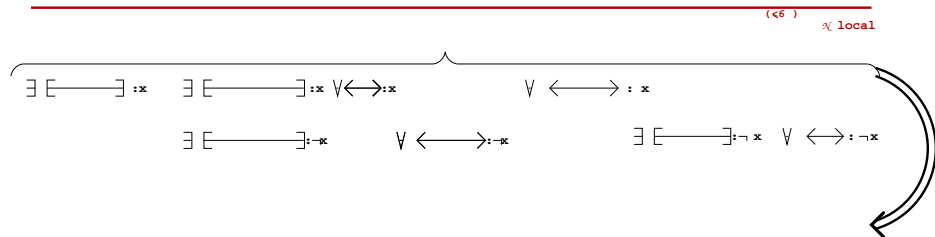
# Produit Réduit Contraintes-nombre de changements de valeur



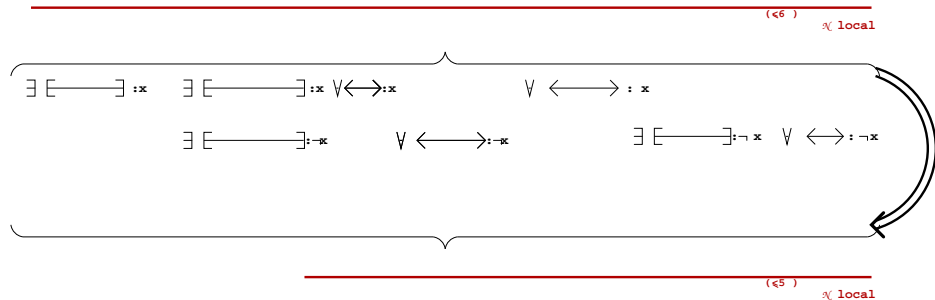
# Produit Réduit Contraintes-nombre de changements de valeur



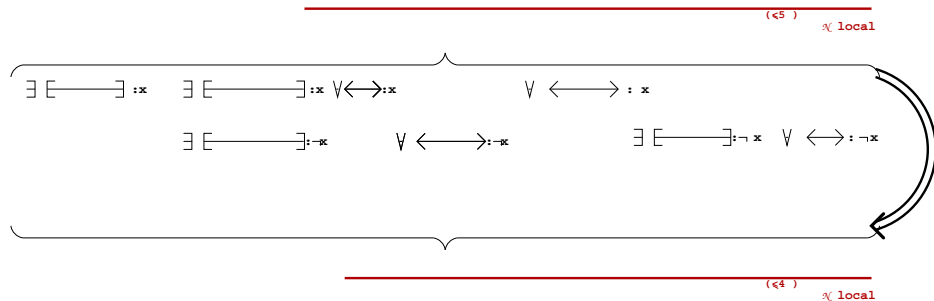
# Réduction en temps linéaire : un exemple



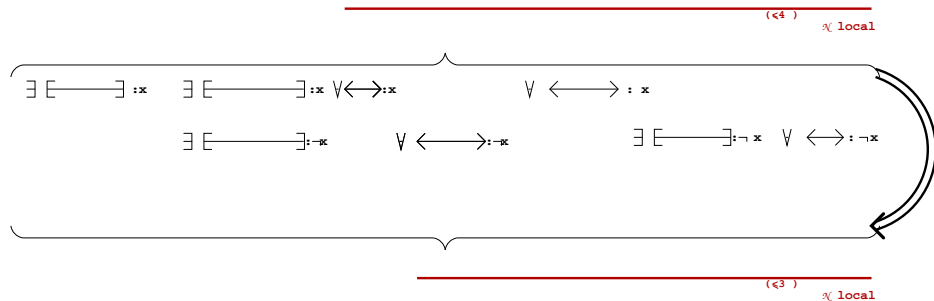
## Réduction en temps linéaire : un exemple



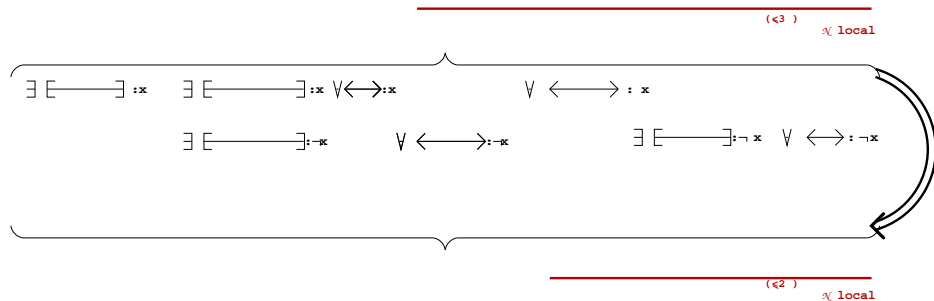
## Réduction en temps linéaire : un exemple



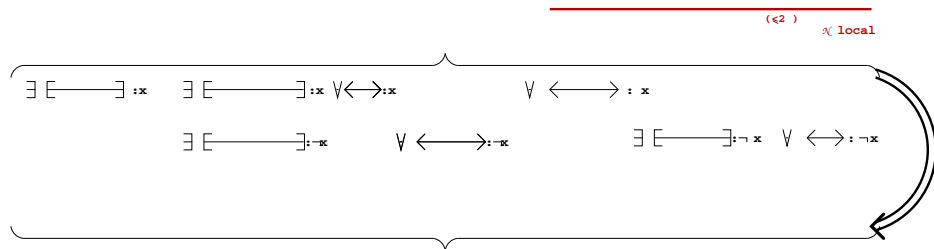
# Réduction en temps linéaire : un exemple



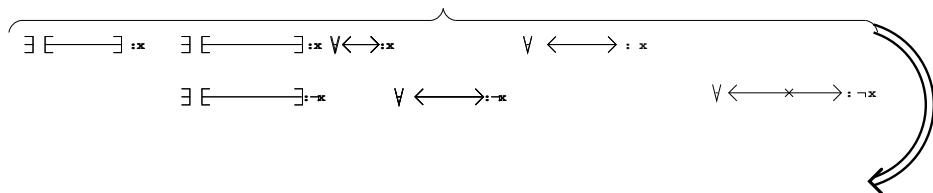
# Réduction en temps linéaire : un exemple



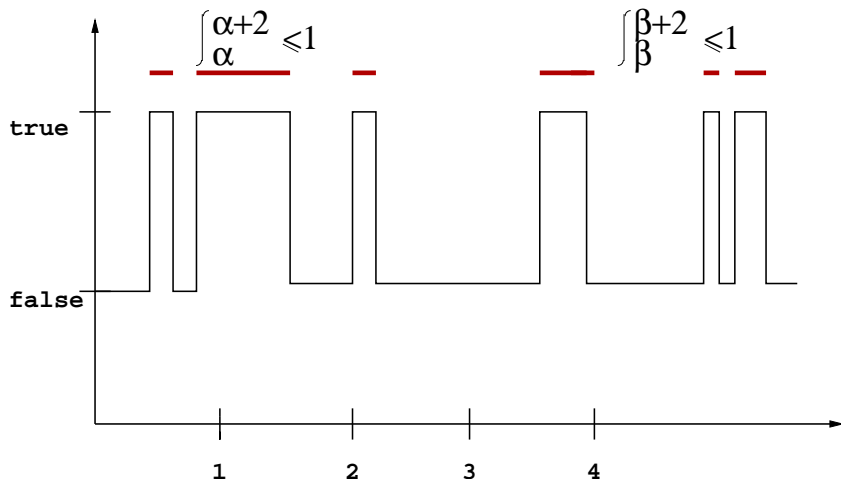
# Réduction en temps linéaire : un exemple



# Réduction en temps linéaire : un exemple



## 3ème domaine abstrait : encadrement des intégrales



# Coopération entre domaines

## encadrements des intégrales et domaine des contraintes

- Supposons à un point de contrôle les informations abstraites suivantes :
  - ▶  $0 \leq \int_x^{x+4} v(t) dt \leq 3.$
  - ▶  $\forall \langle x; x + 1 \rangle : \text{True}$  et  $\forall \langle x + 3; x + 4 \rangle : \text{True}$

# Coopération entre domaines

## encadrements des intégrales et domaine des contraintes

- Supposons à un point de contrôle les informations abstraites suivantes :
  - ▶  $0 \leq \int_x^{x+4} v(t) dt \leq 3.$
  - ▶  $\forall \langle x; x + 1 \rangle : \text{True}$  et  $\forall \langle x + 3; x + 4 \rangle : \text{True}$
- $\int_x^{x+4} v(t) dt$

$$= \int_x^{x+1} v(t) dt + \int_{x+1}^{x+3} v(t) dt + \int_{x+3}^{x+4} v(t) dt$$

$$= 2 + \int_{x+1}^{x+3} v(t) dt,$$

# Coopération entre domaines

## encadrements des intégrales et domaine des contraintes

- Supposons à un point de contrôle les informations abstraites suivantes :

- ▶  $0 \leq \int_x^{x+4} v(t) dt \leq 3.$

- ▶  $\forall \langle x; x + 1 \rangle : \text{True}$  et  $\forall \langle x + 3; x + 4 \rangle : \text{True}$

- $\int_x^{x+4} v(t) dt$

$$= \int_x^{x+1} v(t) dt + \int_{x+1}^{x+3} v(t) dt + \int_{x+3}^{x+4} v(t) dt$$

$$= 2 + \int_{x+1}^{x+3} v(t) dt,$$

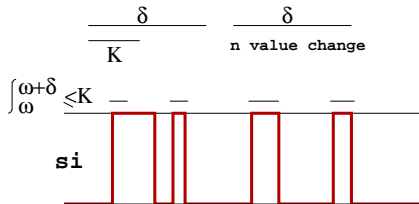
- $0 \leq \int_{x+1}^{x+3} v(t) dt \leq 1.$

# Coopération entre domaines : optimisation des $\Psi^\#$

encadrements des intégrales et comptage des changements de valeur

- $\forall \omega, \int_{\omega}^{\omega+\delta} s_i(t) dt \leq K$

- $(\leq n, \triangleright \delta \triangleleft)$

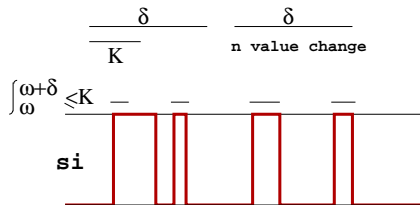
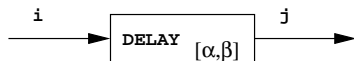


# Coopération entre domaines : optimisation des $\Psi^\#$

encadrements des intégrales et comptage des changements de valeur

- $\forall \omega, \int_{\omega}^{\omega+\delta} s_i(t) dt \leq K$

- $(\leq n, \triangleright \delta \triangleleft)$

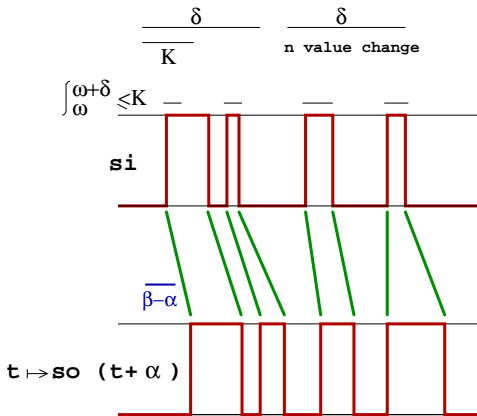
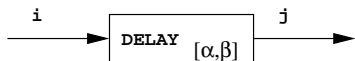


# Coopération entre domaines : optimisation des $\Psi^\#$

encadrements des intégrales et comptage des changements de valeur

- $\forall \omega, \int_{\omega}^{\omega+\delta} s_i(t) dt \leq K$

- $(\leq n, \triangleright \delta \triangleleft)$

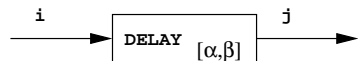


# Coopération entre domaines : optimisation des $\Psi^\#$

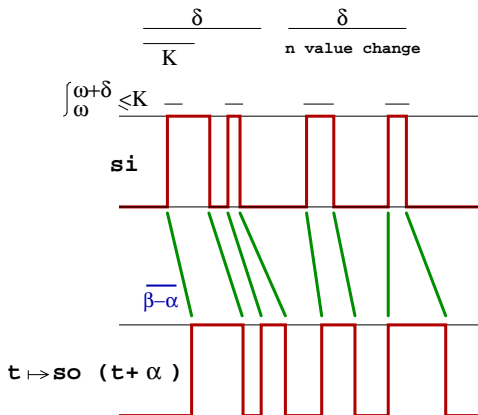
encadrements des intégrales et comptage des changements de valeur

- $\forall \omega, \int_{\omega}^{\omega+\delta} s_i(t) dt \leq K$

- $(\leq n, \triangleright \delta \triangleleft)$



- $\forall \omega, \int_{\omega}^{\omega+\delta-\beta+\alpha} |s_i(t) - s_j(t+\alpha)| dt$

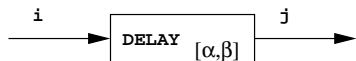


# Coopération entre domaines : optimisation des $\Psi^\#$

encadrements des intégrales et comptage des changements de valeur

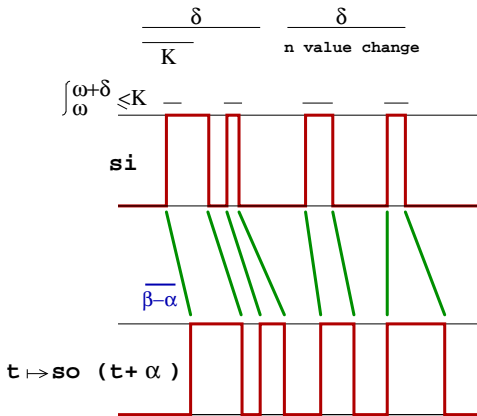
- $\forall \omega, \int_{\omega}^{\omega+\delta} s_i(t) dt \leq K$

- $(\leq n, \triangleright \delta \triangleleft)$



- $\forall \omega, \int_{\omega}^{\omega+\delta-\beta+\alpha} |s_i(t) - s_j(t+\alpha)| dt$

$$\leq n \times (\beta - \alpha)$$

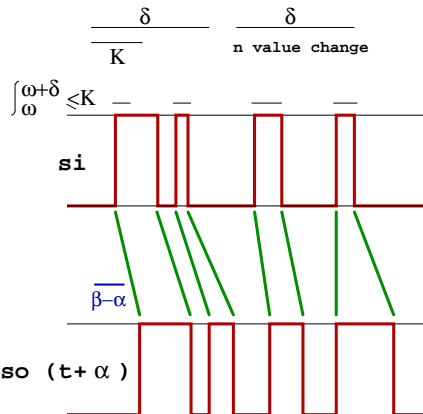
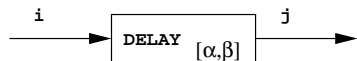


# Coopération entre domaines : optimisation des $\Psi^\#$

encadrements des intégrales et comptage des changements de valeur

- $\forall \omega, \int_{\omega}^{\omega+\delta} s_i(t) dt \leq K$

- $(\leq n, \triangleright \delta \triangleleft)$



- $\forall \omega, \int_{\omega}^{\omega+\delta-\beta+\alpha} |s_i(t) - s_j(t+\alpha)| dt$

$$\leq n \times (\beta - \alpha)$$

thus, if  $\delta - \beta + \alpha \geq 0$  :

$$\forall \omega, \int_{\omega}^{\omega+\delta-\beta+\alpha} s_j(t) dt \leq K + n(\beta - \alpha)$$

- $\vec{\Psi}^\# (\int_{\Delta=\delta} \leq K) \triangleq \int_{\Delta=\delta-\beta+\alpha} \leq K + n(\beta - \alpha)$

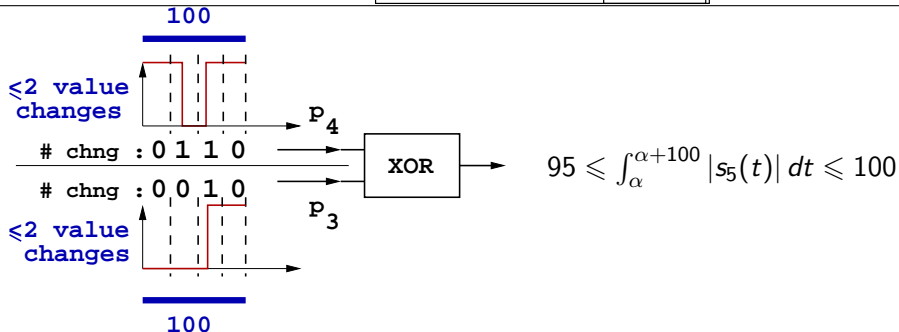
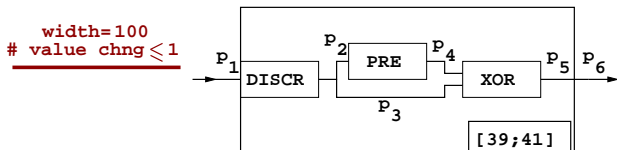
## Coopération entre domaines : optimisation des $\Psi^\#$

Il y a donc plusieurs étapes dans l'écriture d'un analyseur statique :

- définir des domaines abstraits **simples** (mathématiquement) afin qu'ils soient fiables
- définir un analyseur à partir de leur **produit cartésien** afin de couvrir plus de propriété et de code
- définir un analyseur à partir de leur **produit réduit**  $\times^\#$ , c'est à dire que connaître des information abstraite dans 2 domaines sur des éléments permette de prouver plus facilement des propriétés sur les éléments concrets qu'il représente que le produit cartésien. Les analyse sont plus rapide et précises.
- définir des opérateur abstraits plus précis que

$$\lambda a \times^\# b . \Psi^\#(a) \times^\# \Psi^\#(b)$$

# Coopération : exemple des 3 domaines abstraits



# Analyse de code

# Analyse de la redondance et du vote

- Plusieurs unités de calcul redondantes
- Périodiquement, un résultat est choisi, parmi les sorties de ces unités (ou une moyenne éventuellement pondérée si ce sont des flottants)
- Comment choisir ? Quand choisir ?

## voter2/2

P. Caspi and R. Salem, *Threshold and Bounded-Delay Voting in Critical Control Systems*, September 2000.

```
voter2/2(x1,x2,nmax) = x
```

```
where (x, n) =  
  if x1=x2  
  then (x1, 0)  
  else if 0 -> pre n < nmax-1  
        then (x0->pre x, 0 -> (pre n) +1)  
        else alarm
```

# Exécution du voter avec $nmax=2$ et $x0=0$

```
##### STEP 1 #####
x1 (integer) ? 1
x2 (integer) ? 1
x = 1, alarm = false
##### STEP 2 #####
x1 (integer) ? 1
x2 (integer) ? 2
x = 1, alarm = false
##### STEP 3 #####
x1 (integer) ? 2
x2 (integer) ? 1
x = 0, alarm = true
##### STEP 4 #####
x1 (integer) ? 2
x2 (integer) ? 1
x = 0, alarm = false
##### STEP 5 #####
x1 (integer) ? 2
x2 (integer) ? 2
x = 2, alarm = false
##### STEP 6 #####
x1 (integer) ? 1
x2 (integer) ? 1
x = 1, alarm = false
##### STEP 7 #####
x1 (integer) ? 1
x2 (integer) ? 2
x = 1, alarm = false
##### STEP 8 #####
x1 (integer) ? 2
x2 (integer) ? 2
x = 2, alarm = false
```

# Problèmes du voter

- Si les signaux d'entrée sont instables et décalés, `alarm` est toujours *true*!
- Il faut donc des signaux stabilisés en entrée, mais qui continuent à refléter les valeurs des senseurs

## Imposer la stabilisation d'un signal

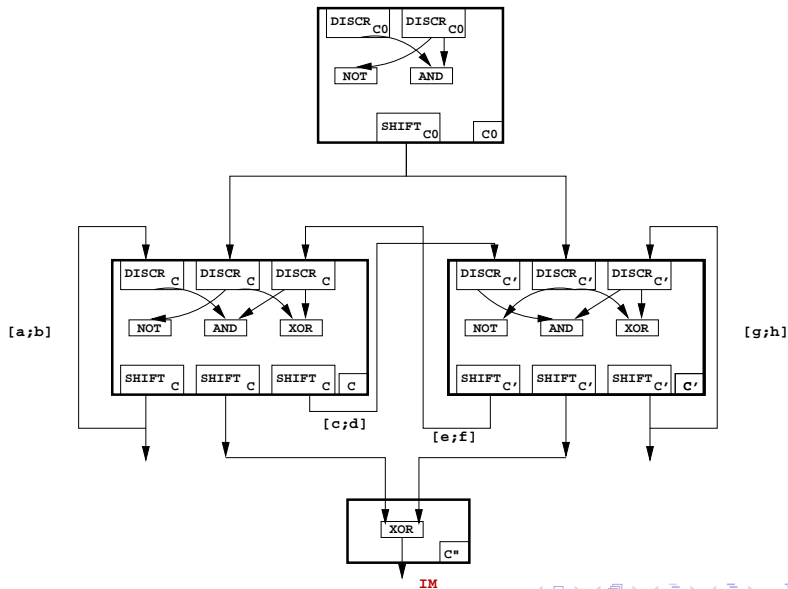
```
node confirm (xp:int) returns (xs: int);
var n, nmax, xo : int;
let
  xo = 314;
  nmax=4;
  xs,n = if (xp=( xo-> pre xp))
    then
      (if ((0 -> pre n) < nmax - 1)
        then
          xo -> pre xs, 0 -> pre (n+1)
        else
          xp, 0 -> pre n
        )
    else
      (xo -> pre xs), 0;
tel;
```

# Imposer la stabilisation d'un signal

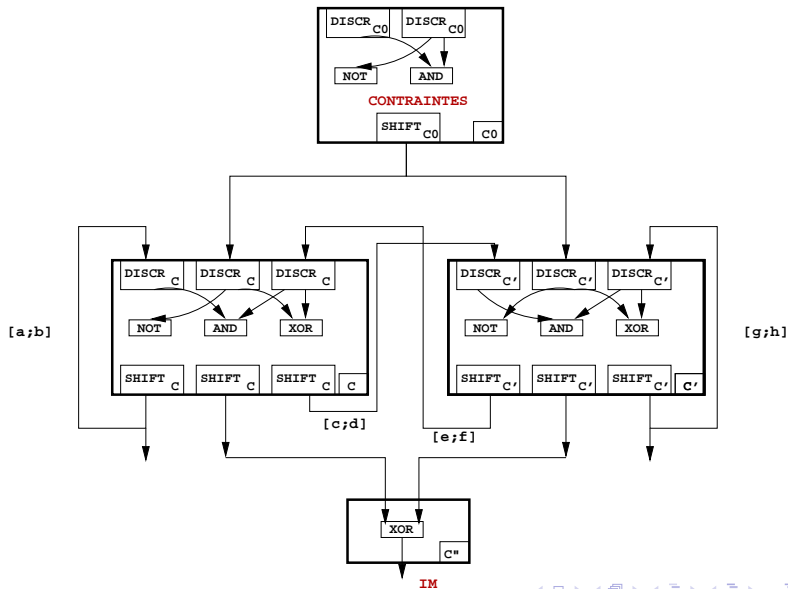
```
##### STEP 1 #####
xp (integer) ? 1
xs = 314
##### STEP 2 #####
xp (integer) ? 2
xs = 314
##### STEP 3 #####
xp (integer) ? 3
xs = 314
##### STEP 4 #####
xp (integer) ? 1
xs = 314
##### STEP 5 #####
xp (integer) ? 1
xs = 314
##### STEP 6 #####
xp (integer) ? 1
xs = 1
##### STEP 7 #####
xp (integer) ? 2
xs = 1
##### STEP 8 #####
xp (integer) ? 2
xs = 1
```

```
##### STEP 9 #####
xp (integer) ? 3
xs = 1
##### STEP 10 #####
xp (integer) ? 3
xs = 1
##### STEP 11 #####
xp (integer) ? 4
xs = 1
##### STEP 12 #####
xp (integer) ? 4
xs = 1
##### STEP 13 #####
xp (integer) ? 4
xs = 4
```

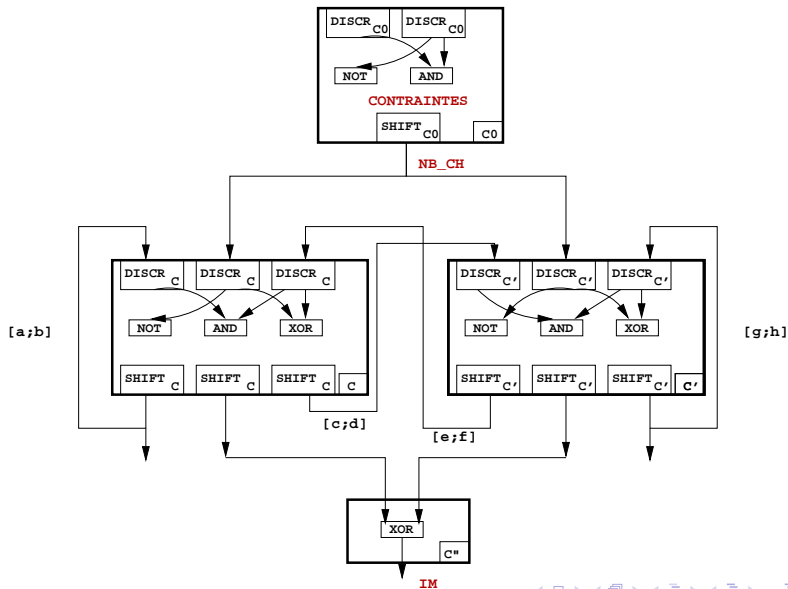
# Propagation efficace de l'information abstraite



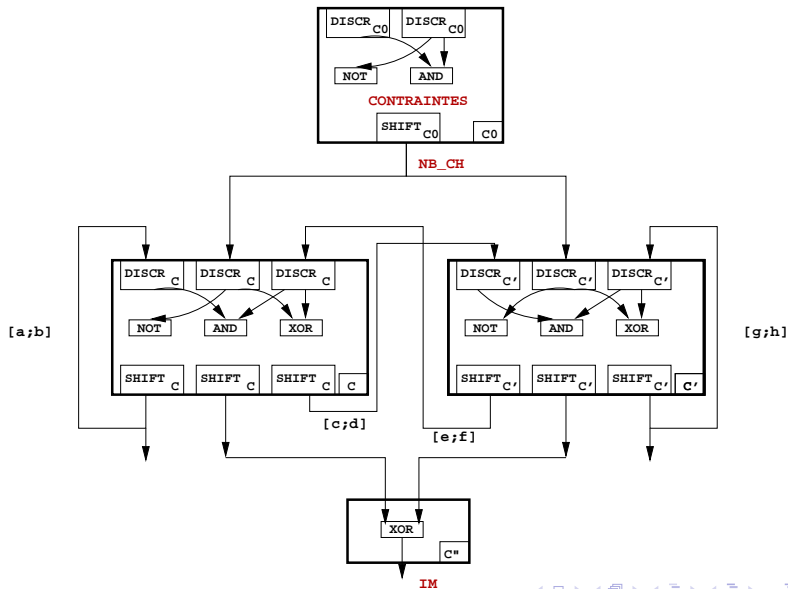
# Propagation efficace de l'information abstraite



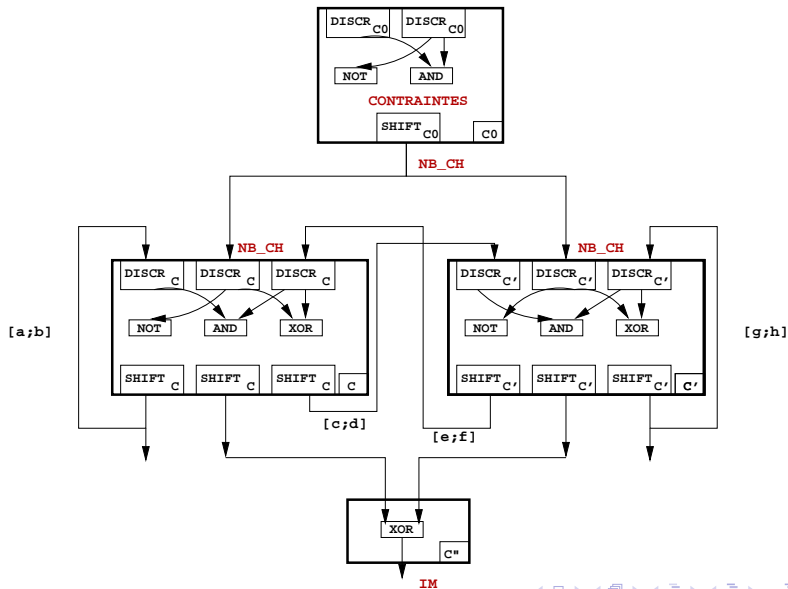
# Propagation efficace de l'information abstraite



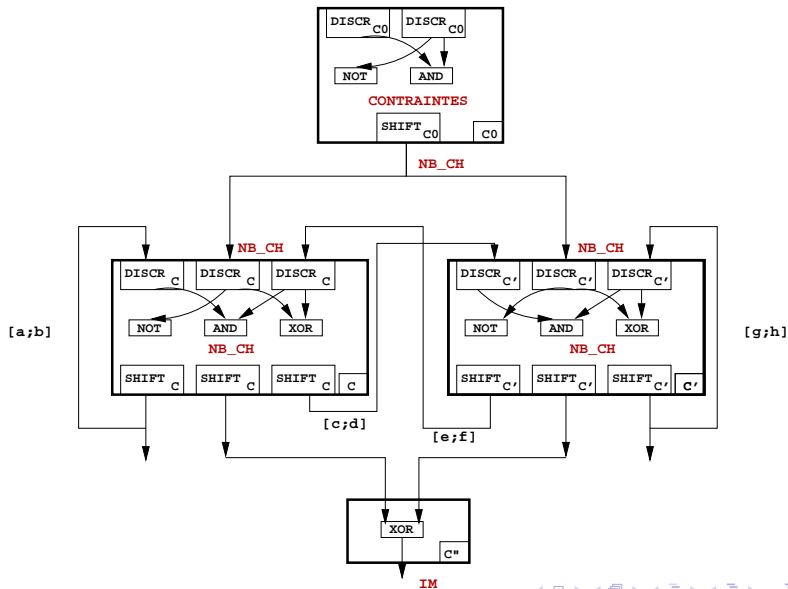
# Propagation efficace de l'information abstraite



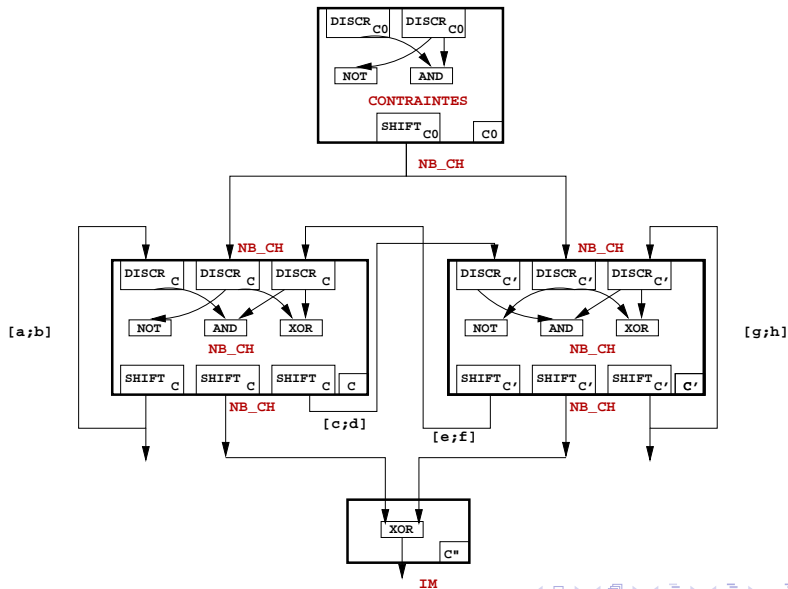
# Propagation efficace de l'information abstraite



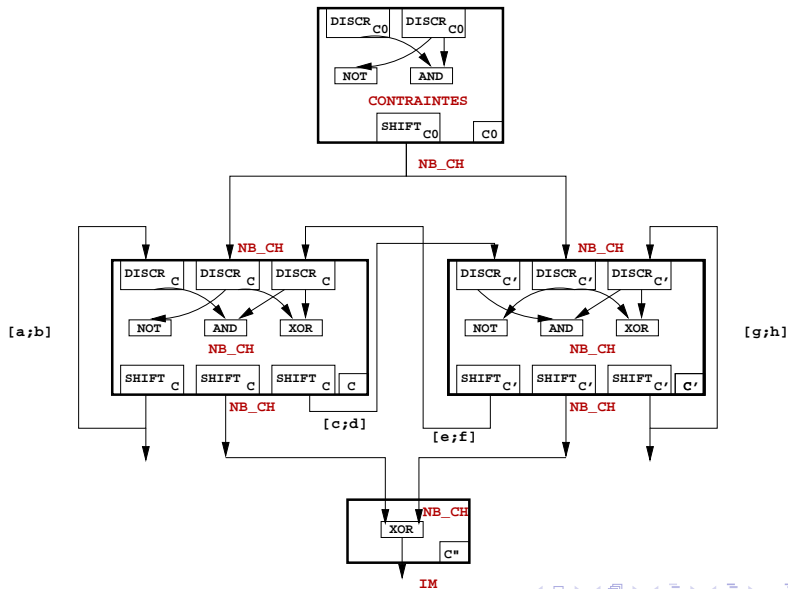
# Propagation efficace de l'information abstraite



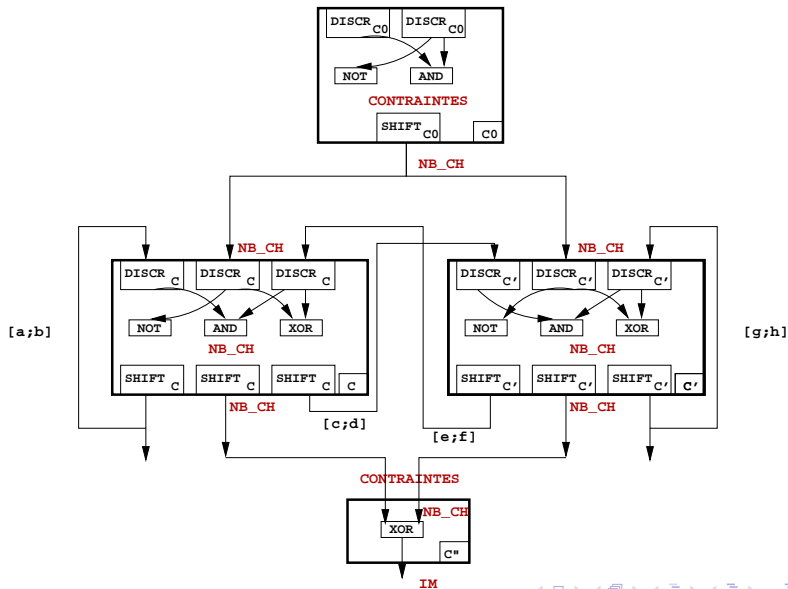
# Propagation efficace de l'information abstraite



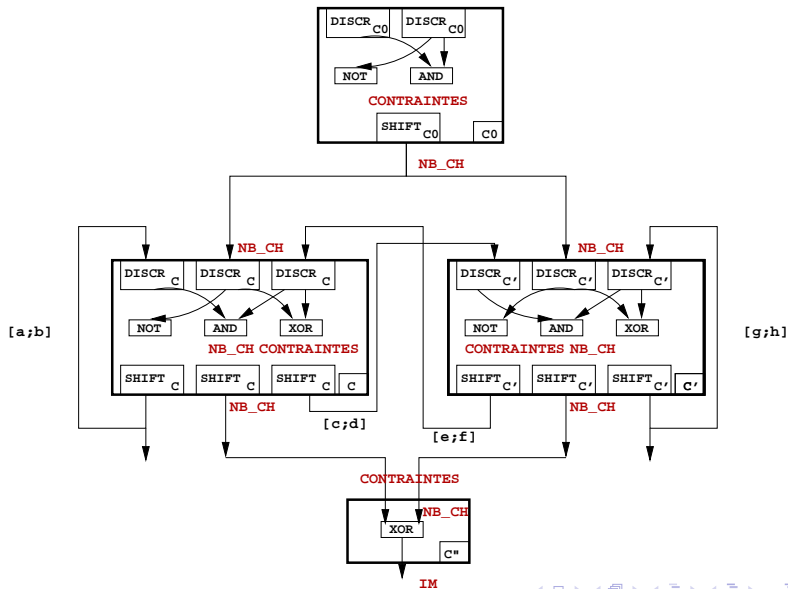
# Propagation efficace de l'information abstraite



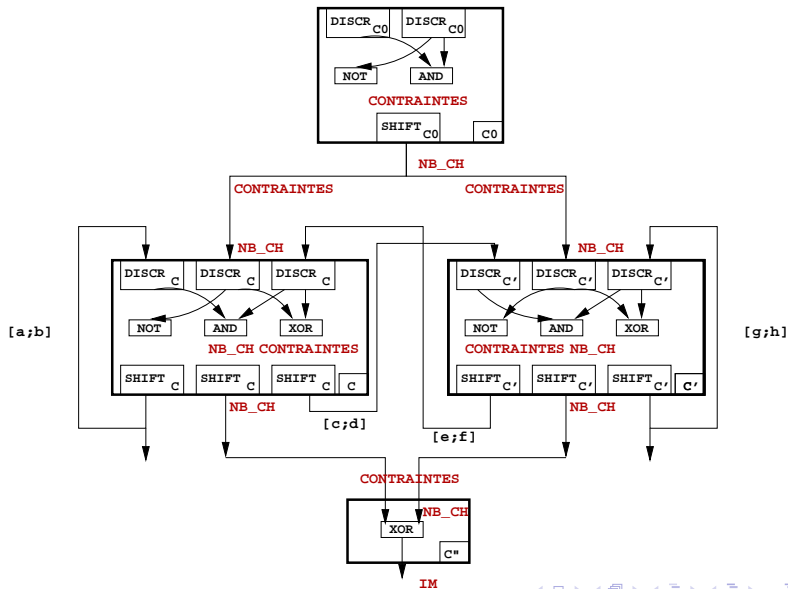
# Propagation efficace de l'information abstraite



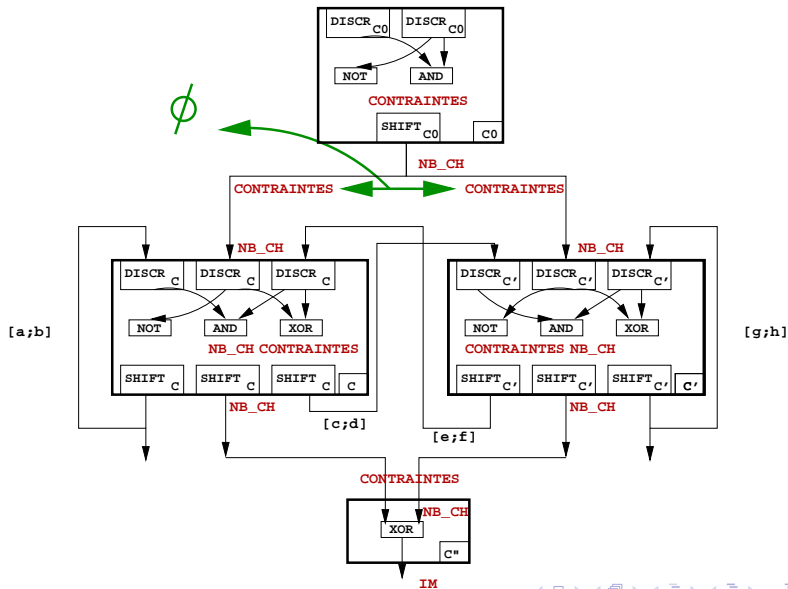
# Propagation efficace de l'information abstraite



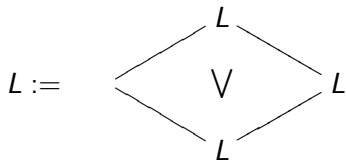
# Propagation efficace de l'information abstraite



# Propagation efficace de l'information abstraite



# Produit réduit disjonctif Contraintes - Nombre de changements de valeur



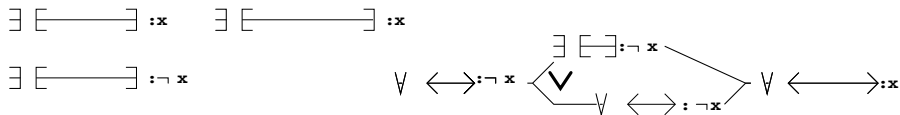
|  $C :: L$

|  $\mathcal{N}_{\text{local}} :: L$

|  $\square$  (empty element)

# Produit réduit disjonctif Contraintes - Nombre de changements de valeur

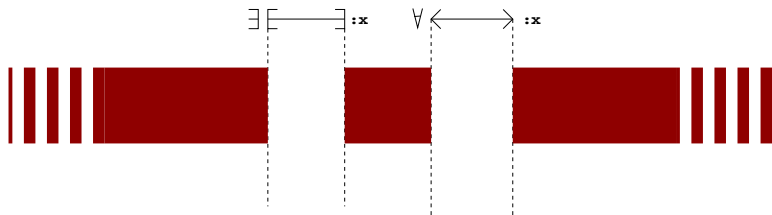
Un exemple



# Caractéristiques communes des domaines temporels

## Support temporel

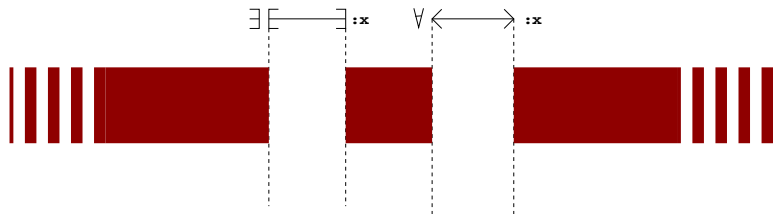
- En dehors du **support temporel**, un élément abstrait n'a pas d'influence
- Par exemple, pour les contraintes :



# Caractéristiques communes des domaines temporels

## Support temporel

- En dehors du **support temporel**, un élément abstrait n'a pas d'influence
- Par exemple, pour les contraintes :



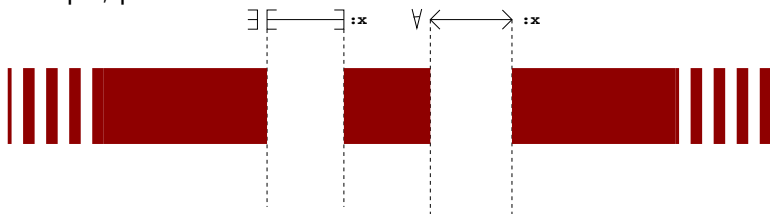
$$\text{right\_border} : \begin{cases} \forall \langle a; b \rangle : x \mapsto b \\ \exists [a; b] : x \mapsto b \end{cases}$$

$$\text{left\_border} : \begin{cases} \forall \langle a; b \rangle : x \mapsto a \\ \exists [a; b] : x \mapsto a \end{cases}$$

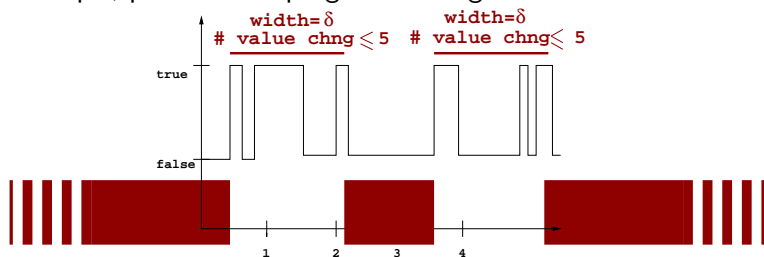
# Caractéristiques communes des domaines temporels

## Support temporel

- Par exemple, pour les contraintes :



- Par exemple, pour les comptages de changements de valeur :



# Caractéristiques communes des domaines temporels

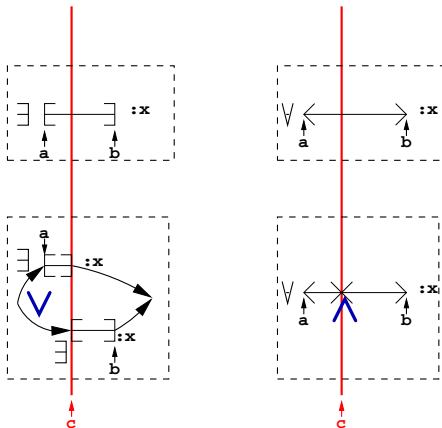
## Slicing temporel

- La *fonction de slicing* fournit un élément affaibli mais contenue dans la slice temporelle.

# Caractéristiques communes des domaines temporels

## Slicing temporel

- La *fonction de slicing* fournit un élément affaibli mais contenue dans la slice temporelle.



# Caractéristiques communes des domaines temporels

## Slicing temporel

- La *fonction de slicing* fournit un élément affaibli mais contenue dans la slice temporelle.

# Caractéristiques communes des domaines temporels

## Slicing temporel

- La *fonction de slicing* fournit un élément affaibli mais contenue dans la slice temporelle.
- Le *slicing* du comptages de changements de valeur  $(\leq 3, a \blacktriangleright, \blacktriangleleft b)_{\mathcal{N}}$  est :

$$((\leq 3, a \blacktriangleright, \blacktriangleleft c)_{\mathcal{N}} \wedge (\leq 0, c \blacktriangleright, \blacktriangleleft b)_{\mathcal{N}})$$

$$\vee ((\leq 2, a \blacktriangleright, \blacktriangleleft c)_{\mathcal{N}} \wedge (\leq 1, c \blacktriangleright, \blacktriangleleft b)_{\mathcal{N}})$$

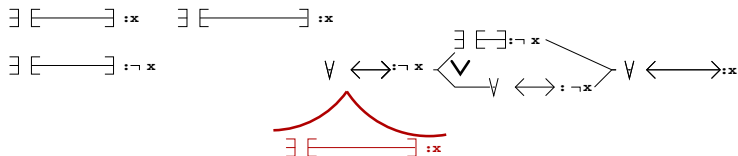
$$\vee ((\leq 1, a \blacktriangleright, \blacktriangleleft c)_{\mathcal{N}} \wedge (\leq 2, c \blacktriangleright, \blacktriangleleft b)_{\mathcal{N}})$$

$$\vee ((\leq 0, a \blacktriangleright, \blacktriangleleft c)_{\mathcal{N}} \wedge (\leq 3, c \blacktriangleright, \blacktriangleleft b)_{\mathcal{N}})$$

# Caractéristiques communes des domaines temporels

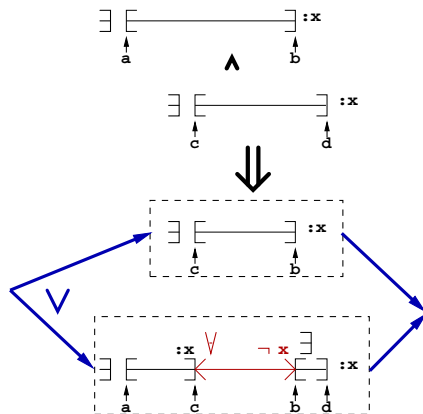
## Interactions temporelles

- Les *interactions temporelles* de 2 éléments dont les supports temporels se chevauchent est une sur-approximation du résultat de leur conjonction.

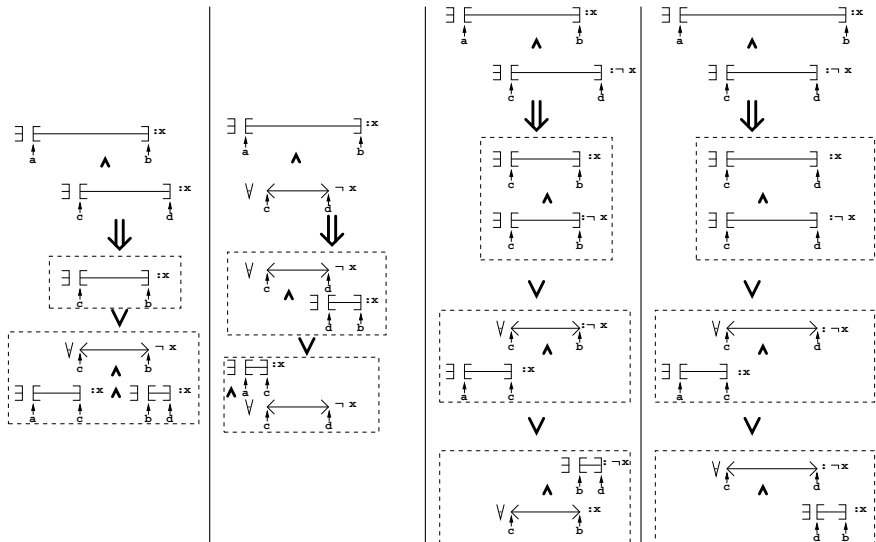


# Caractéristiques communes des domaines temporels

## Interactions temporelles

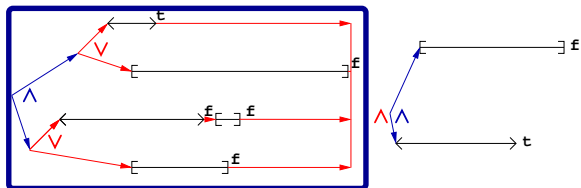
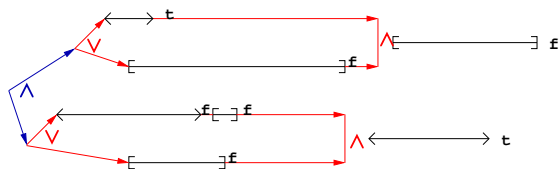


# Produit réduit disjonctif : règles additionnelles



# Temporel domains : Conjunction

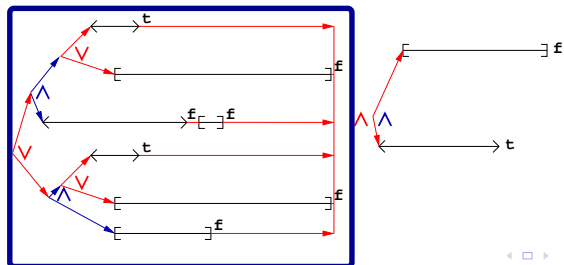
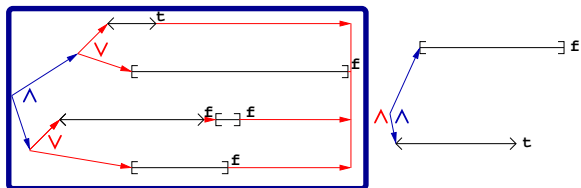
- Comment calculer la conjonction de 2 éléments :



Slicing temporel

# Temporel domains : Conjunction

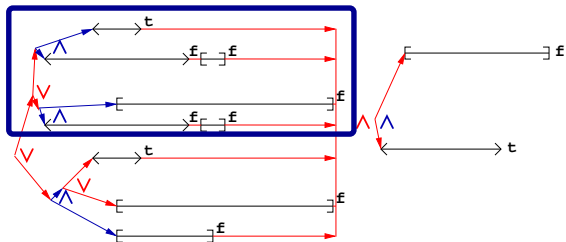
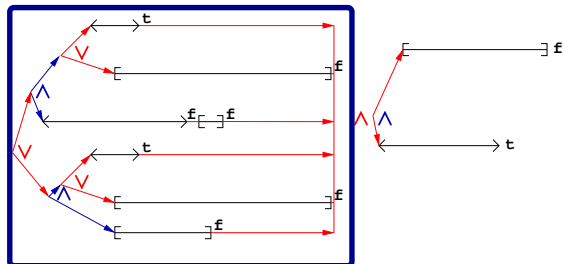
- Comment calculer la conjonction de 2 éléments :



Propagation

# Temporel domains : Conjunction

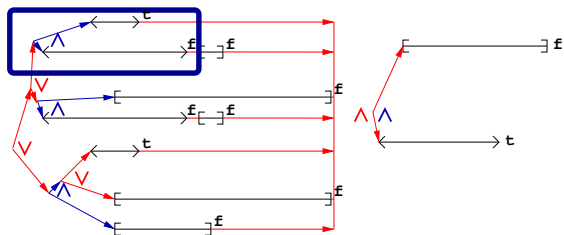
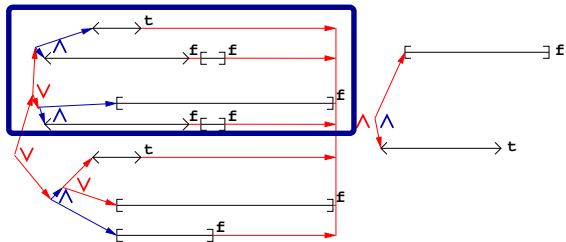
- Comment calculer la conjonction de 2 éléments :



Propagation

# Temporel domains : Conjunction

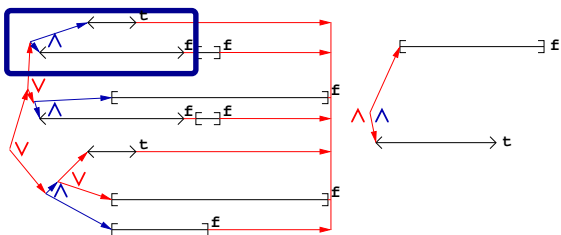
- Comment calculer la conjonction de 2 éléments :



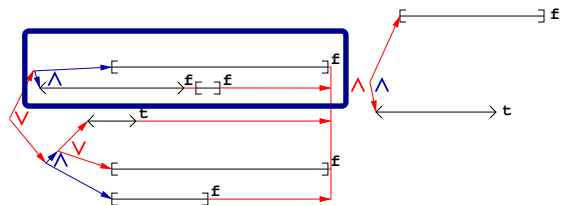
Slicing temporel

# Temporel domains : Conjunction

- Comment calculer la conjonction de 2 éléments :

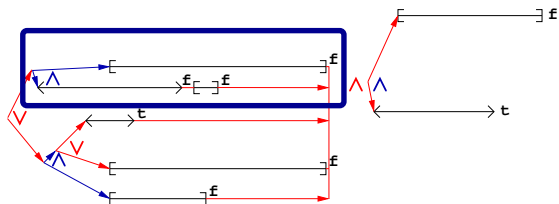


Interaction ( $\perp$ )

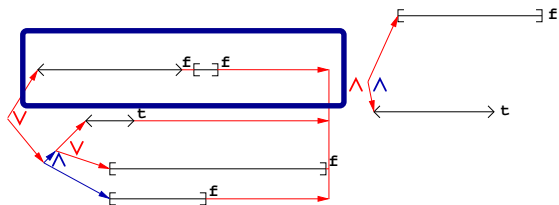


# Temporel domains : Conjunction

- Comment calculer la conjonction de 2 éléments :

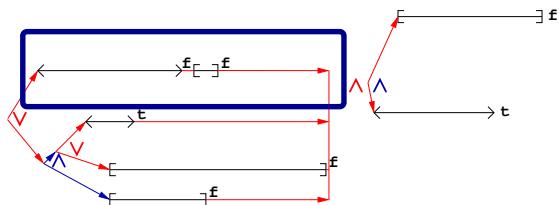


Interaction

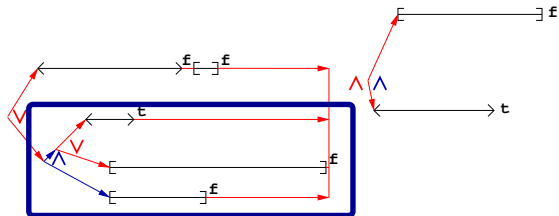


# Temporel domains : Conjunction

- Comment calculer la conjonction de 2 éléments :

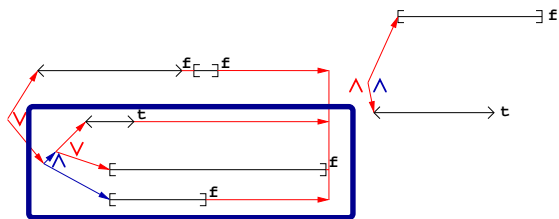


Slicing temporel

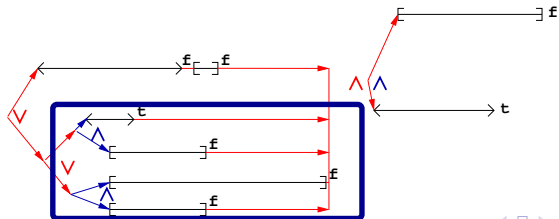


# Temporel domains : Conjunction

- Comment calculer la conjonction de 2 éléments :

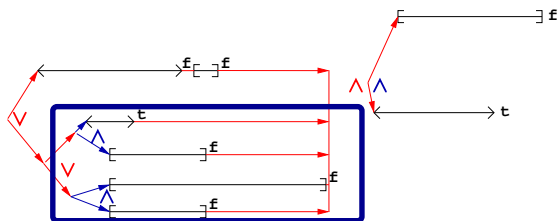


Propagation

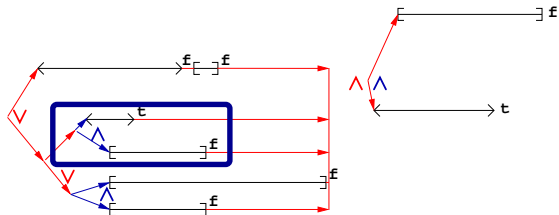


# Temporel domains : Conjunction

- Comment calculer la conjonction de 2 éléments :

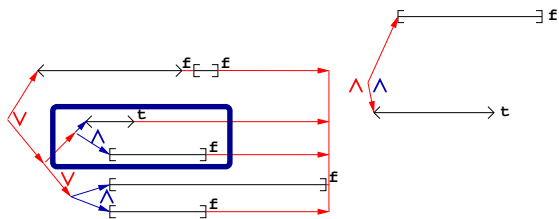


Slicing temporel

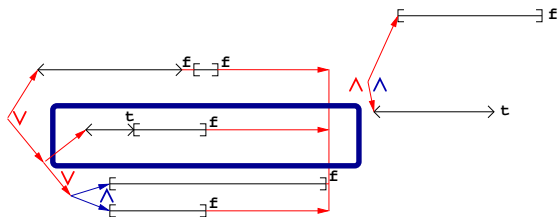


# Temporel domains : Conjunction

- Comment calculer la conjonction de 2 éléments :

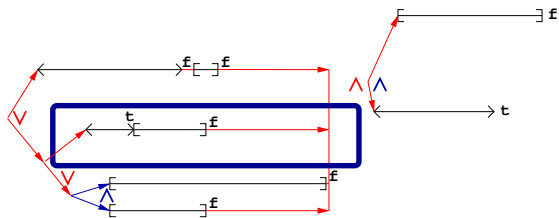


Interaction

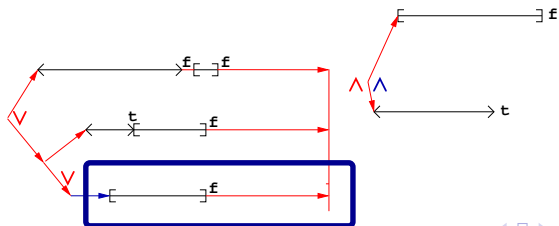


# Temporel domains : Conjunction

- Comment calculer la conjonction de 2 éléments :

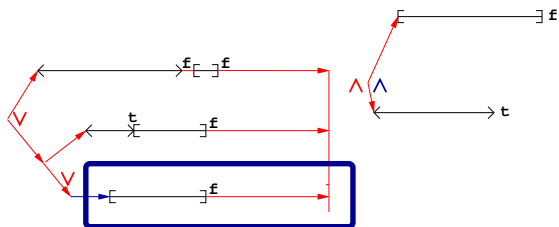


Interaction

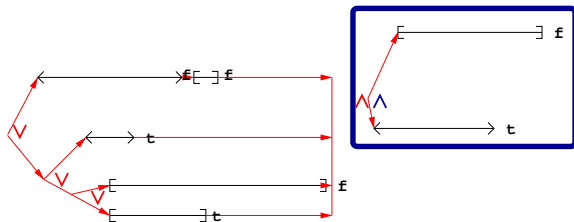


# Temporel domains : Conjunction

- Comment calculer la conjonction de 2 éléments :

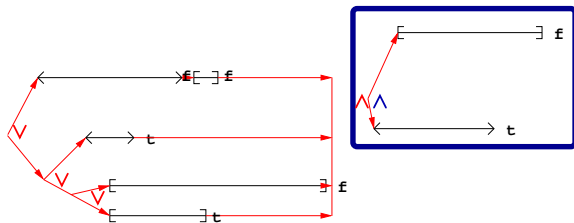


Slicing temporel

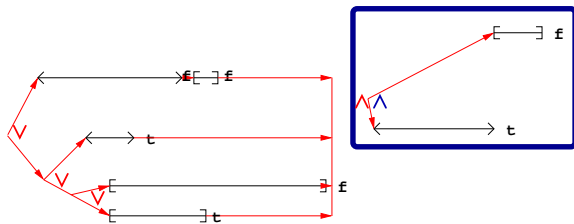


# Temporel domains : Conjunction

- Comment calculer la conjonction de 2 éléments :

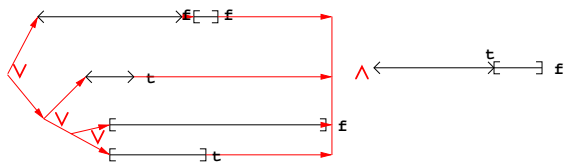
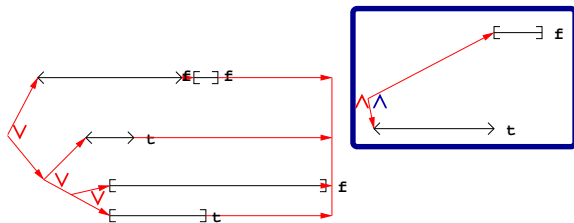


Interaction



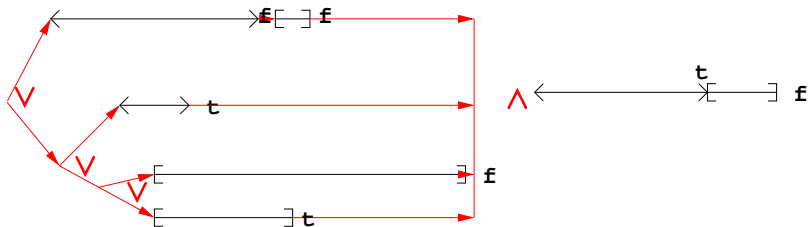
# Temporel domains : Conjunction

- Comment calculer la conjonction de 2 éléments :

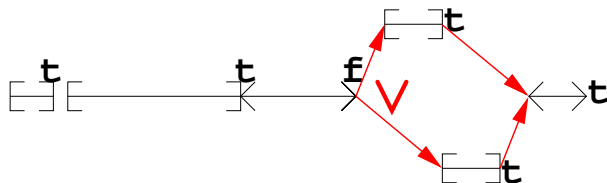


# Temporel domains : Conjunction

- Comment calculer la conjonction de 2 éléments :



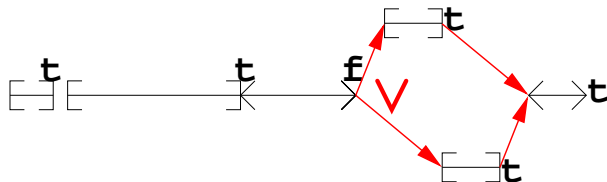
# Produit réduit disjonctif : un exemple




---

$(\leq 2, 5.5)$

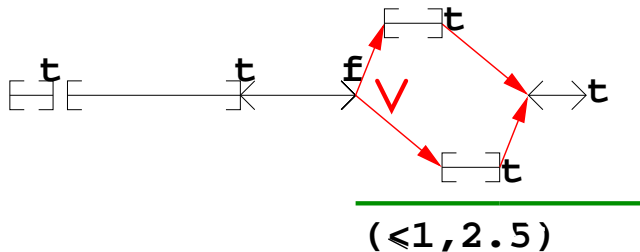
# Produit réduit disjonctif : un exemple



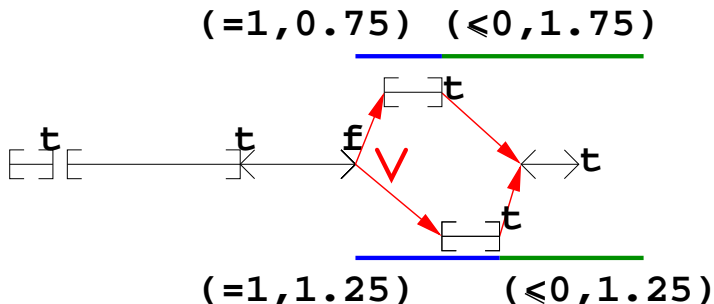

---

$(\leq 2, 5.125)$

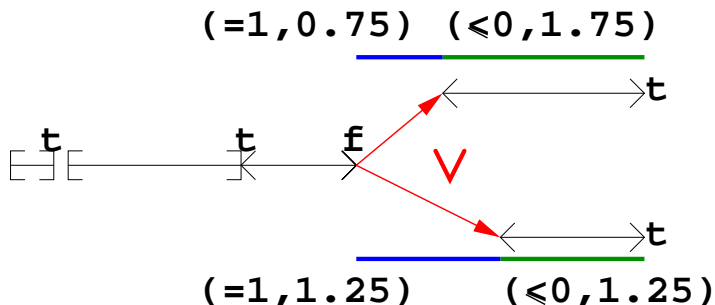
# Produit réduit disjonctif : un exemple



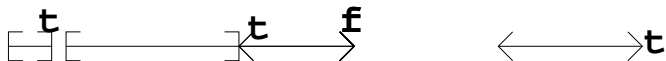
# Produit réduit disjonctif : un exemple



# Produit réduit disjonctif : un exemple



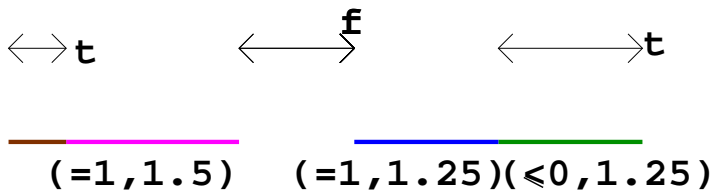
# Produit réduit disjonctif : un exemple



$(=0, .5)$

$(=1, 1.5)$   $(=1, 1.25)$   $(\leq 0, 1.25)$

# Produit réduit disjonctif : un exemple



# conclusion

- Grande différence entre le **modèle** pour lequel sont **écrits les systèmes synchrones communicants** et l'exécution réelle du programme (horloge imparfaite, communication non-instantanée)

- Grande différence entre le **modèle** pour lequel sont **écrits les systèmes synchrones communicants** et l'exécution réelle du programme (horloge imparfaite, communication non-instantanée)
- On peut **quantifier** ces imperfection grâce à une syntaxe graphique

- Grande différence entre le **modèle** pour lequel sont **écrits les systèmes synchrones communicants** et l'exécution réelle du programme (horloge imparfaite, communication non-instantanée)
- On peut **quantifier** ces imperfection grâce à une syntaxe graphique
- On modélise (certaines de?) ces différences grâce à une **sémantique temps-continu**

- Grande différence entre le **modèle** pour lequel sont **écrits les systèmes synchrones communicants** et l'exécution réelle du programme (horloge imparfaite, communication non-instantanée)
- On peut **quantifier** ces imperfection grâce à une syntaxe graphique
- On modélise (certaines de?) ces différences grâce à une **sémantique temps-continu**
- Mais
  - ▶ les **preuves de correction** sont beaucoup plus difficiles à obtenir (à la main : impossible)
  - ▶ les analyses automatiques nécessitent des **domaines abstraits traitant du temps** ad hoc

- Grande différence entre le **modèle** pour lequel sont **écrits les systèmes synchrones communicants** et l'exécution réelle du programme (horloge imparfaite, communication non-instantanée)
- On peut **quantifier** ces imperfection grâce à une syntaxe graphique
- On modélise (certaines de?) ces différences grâce à une **sémantique temps-continu**
- Mais
  - ▶ les **preuves de correction** sont beaucoup plus difficiles à obtenir (à la main : impossible)
  - ▶ les analyses automatiques nécessitent des **domaines abstraits traitant du temps** ad hoc
- L'**interprétation abstraite** s'accomode très bien de l'**analyse temporelle** et permet la **construction incrémentale** d'un analyseur sûr : modularité vis à vis des domaines abstraits, optimisations (Produit réduit, opérateurs abstraits optimisés)

- Grande différence entre le **modèle** pour lequel sont **écrits les systèmes synchrones communicants** et l'exécution réelle du programme (horloge imparfaite, communication non-instantanée)
- On peut **quantifier** ces imperfection grâce à une syntaxe graphique
- On modélise (certaines de?) ces différences grâce à une **sémantique temps-continu**
- Mais
  - ▶ les **preuves de correction** sont beaucoup plus difficiles à obtenir (à la main : impossible)
  - ▶ les analyses automatiques nécessitent des **domaines abstraits traitant du temps** ad hoc
- L'**interprétation abstraite** s'accomode très bien de l'**analyse temporelle** et permet la **construction incrémentale** d'un analyseur sûr : modularité vis à vis des domaines abstraits, optimisations (Produit réduit, opérateurs abstraits optimisés)
- Les domaines abstraits temporels sont naturellement construits à partir de mathématiques continues plus simples et moins utilisées que les mathématiques discrètes.

# Questions ?

Transparents : [www.di.ens.fr/~bertrane](http://www.di.ens.fr/~bertrane)

Contact : [bertrane@di.ens.fr](mailto:bertrane@di.ens.fr)

Stages : <http://www.di.ens.fr/~cousot/cours/MPRI/M2/0809/>  
<http://www.di.ens.fr/~cousot/enseignement/stages.shtml>