

Reduced product of abstract domains for the static analysis of properties of imperfectly-clocked synchronous systems

Julien Bertrane
bertrane@di.ens.fr

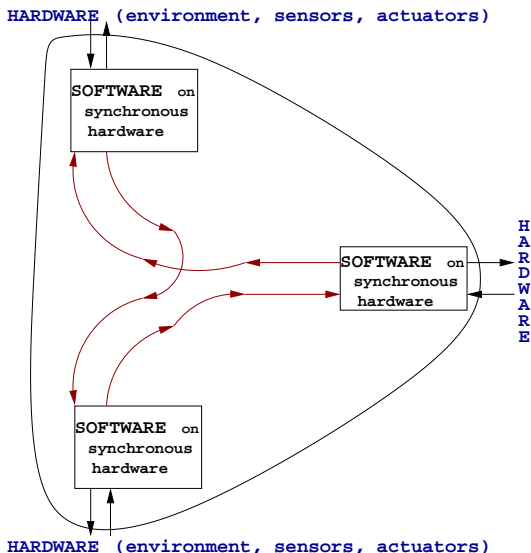
ENS Paris

august 5th, 2008

- 1 Introduction
- 2 Syntax and Semantics
 - Constraints on the chosen syntax and semantics
 - A “time-continuous” semantics
- 3 Abstract Analysis
 - Toward an abstract Analysis
 - Abstract constraints domain
 - Value changes counting domain
 - Integral bounding domain
- 4 Real code Analysis
 - Typical programs to certify
 - Analysis
- 5 Conclusion

Introduction

Typical system to verify : embedded systems



Synchronous programming

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

with I : Inputs, S : (internal) State variables, O : Outputs

Why studying such systems ?

- Analyze Systems built on top of **several communicating synchronous programs** (each one with its own clock). Why?
 - ▶ some embedded systems are **too big** for a single clock : transmitting data through the system would be **too slow**

Verifying specifications

- **Safety specifications**
 - ▶ for any behavior s , at any moment t , $s(t) \neq true$

Verifying specifications

- **Safety specifications**

- ▶ for any behavior s , at any moment t , $s(t) \neq true$

- **Temporal specifications**

- ▶ for any behavior s , there is no time t such that :

for any $t' \in [t, t + \alpha]$, $s(t') = true$

Verifying specifications

- **Safety specifications**

- ▶ for any behavior s , at any moment t , $s(t) \neq true$

- **Temporal specifications**

- ▶ for any behavior s , there is no time t such that :

for any $t' \in [t, t + \alpha]$, $s(t') = true$

- **Quantitative specifications**

- ▶ The **outputs** of 2 redundant systems remain **equal at least 50% of the time** during any time interval of width at least δ .

Syntax and Semantics

Constraints from the studied cases

- Unavoidable hardware imperfections :
 - ▶ The clocks of the control units may **de-synchronize**

Constraints from the studied cases

- Unavoidable hardware imperfections :
 - ▶ The clocks of the control units may **de-synchronize**
 - ▶ The communications **cannot always** be considered **instantaneous**

Constraints from the studied cases

- Unavoidable hardware imperfections :
 - ▶ The clocks of the control units may **de-synchronize**
 - ▶ The communications **cannot always** be considered **instantaneous**
 - ▶ The Communication delays **cannot always** be considered **constant**

Hypotheses for the semantics

- **Imperfect-synchronicity** :

- ▶ de-synchronization : the length of each clock *cycle* (period between two clock **ticks**) belongs to $[\alpha, \beta]$, $\alpha > 0$. Physically (quartz clock) : this always is the case.
- ▶ hypothesis stronger yet fair than “quasi-synchronicity” introduced by P. Caspi

Hypotheses for the semantics

- **Imperfect-synchronicity** :
 - ▶ de-synchronization : the length of each clock *cycle* (period between two clock **ticks**) belongs to $[\alpha, \beta]$, $\alpha > 0$. Physically (quartz clock) : this always is the case.
 - ▶ hypothesis stronger yet fair than “quasi-synchronicity” introduced by P. Caspi

- **Serial** transmissions between imperfectly synchronous systems

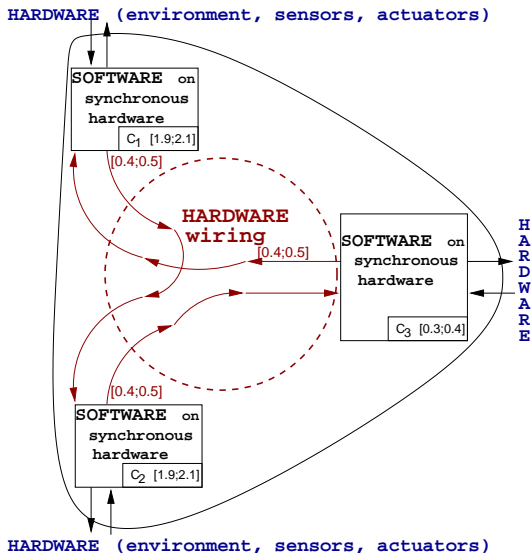
Hypotheses for the semantics

- **Imperfect-synchronicity** :
 - ▶ de-synchronization : the length of each clock *cycle* (period between two clock **ticks**) belongs to $[\alpha, \beta]$, $\alpha > 0$. Physically (quartz clock) : this always is the case.
 - ▶ hypothesis stronger yet fair than “quasi-synchronicity” introduced by P. Caspi
- **Serial** transmissions between imperfectly synchronous systems
- **Blackboard** at each unit inputs

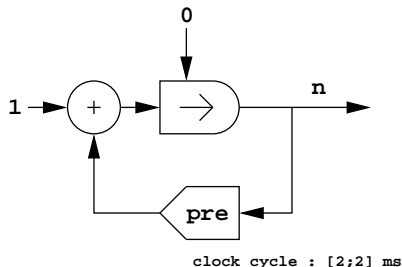
Hypotheses for the semantics

- **Imperfect-synchronicity** :
 - ▶ de-synchronization : the length of each clock *cycle* (period between two clock **ticks**) belongs to $[\alpha, \beta]$, $\alpha > 0$. Physically (quartz clock) : this always is the case.
 - ▶ hypothesis stronger yet fair than “quasi-synchronicity” introduced by P. Caspi
- **Serial** transmissions between imperfectly synchronous systems
- **Blackboard** at each unit inputs
- **Initialization** of all variables to 0 or *false*

Typical system : **quantifying** the hardware imperfections



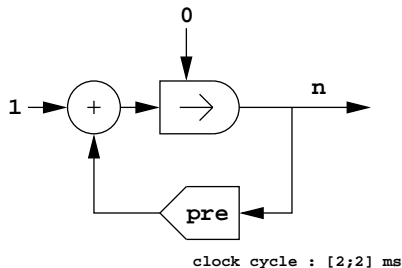
Which syntax : *à la* SCADE ? *à la* Lustre ?



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

Which semantics ?

Synchronous programs can be compiled to C



```

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

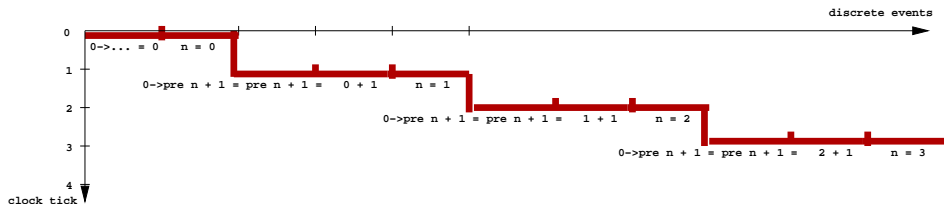
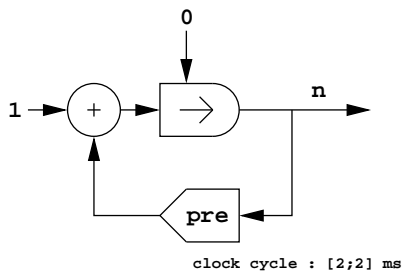
  case 1 :
    global_state->n= (global_state->n)+ 1 ;
    global_state->current_state = 1 ; break ;
  break ;}}

```

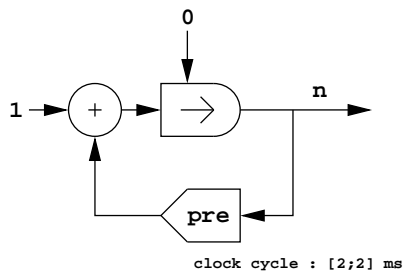


Which semantics ?

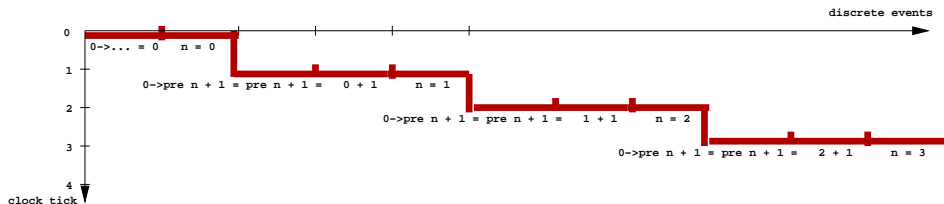
- Embedded software \Rightarrow temporal specification should be provable



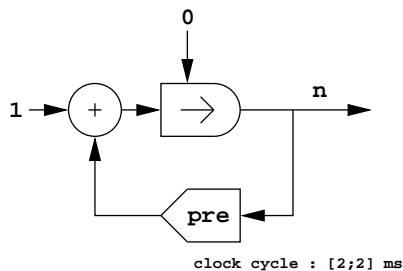
Which semantics ?



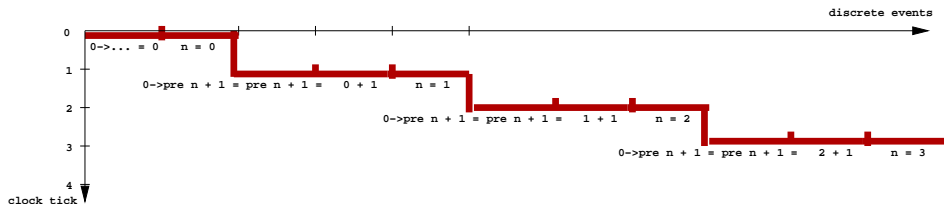
- Embedded software \Rightarrow temporal specification should be provable
- in C, there is no notion of verifiable warranty on execution-time (it depends on hardware, optimizations...)



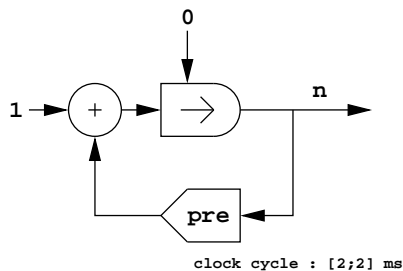
Which semantics ?



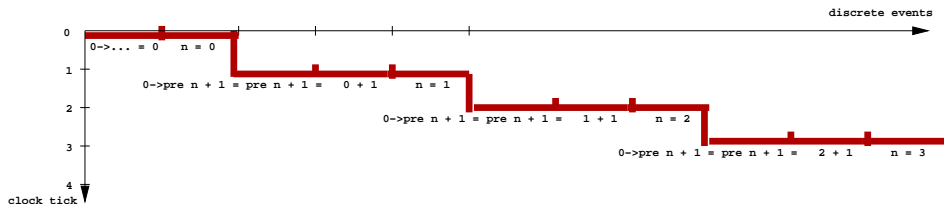
- Embedded software \Rightarrow temporal specification should be provable
- in C, there is no notion of verifiable warranty on execution-time (it depends on hardware, optimizations...)
- What about introducing clock cycles timing the execution



Which semantics ?



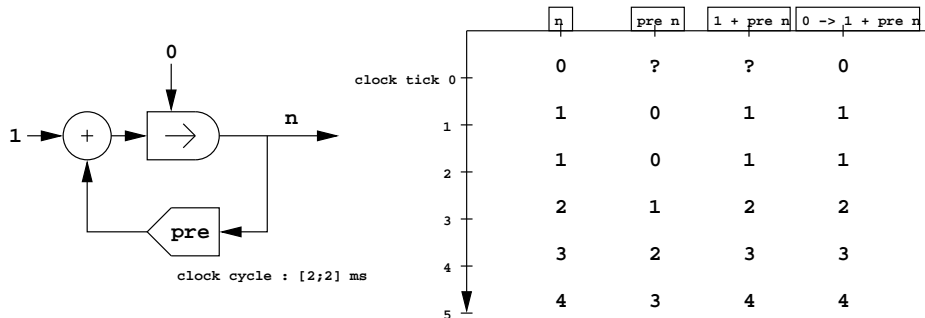
- Embedded software \Rightarrow temporal specification should be provable
- in C, there is no notion of verifiable warranty on execution-time (it depends on hardware, optimizations...)
- What about introducing clock cycles timing the execution
- Does the execution of the instructions of one cycle actually fits in the real clock period (WCET) ?



Toward a non-standard semantics ?

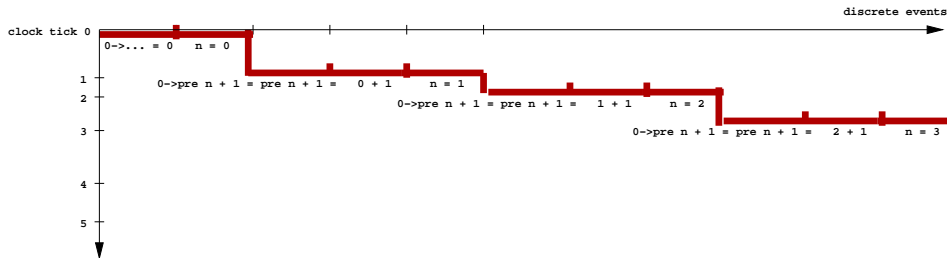
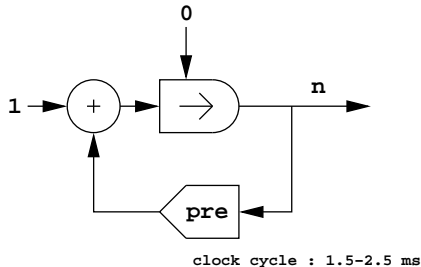
Using the Lustre semantics directly :

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



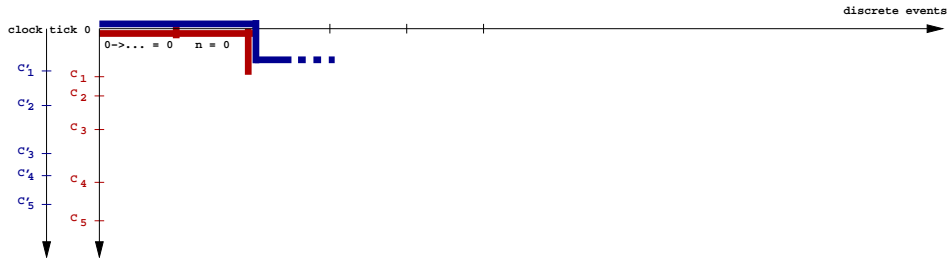
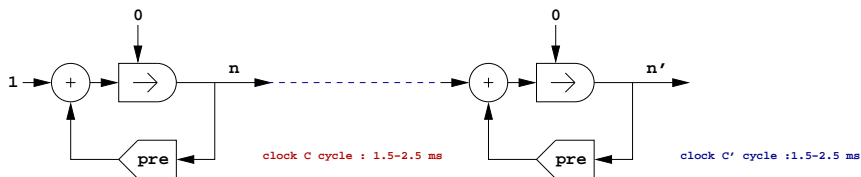
Toward a non-standard semantics ?

- What if the clock is imprecise ?



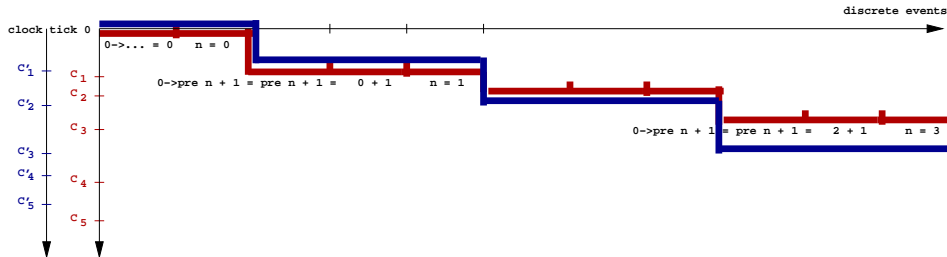
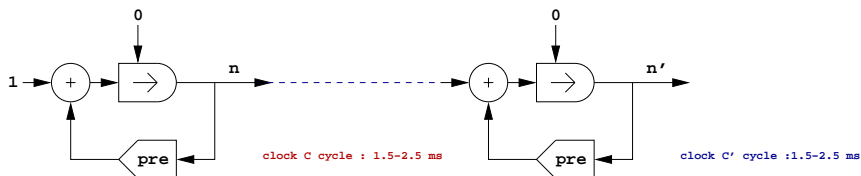
Toward a non-standard semantics ?

Case of several systems, each with its (imprecise) clock :



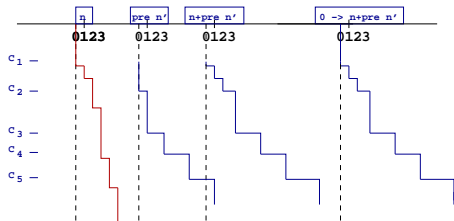
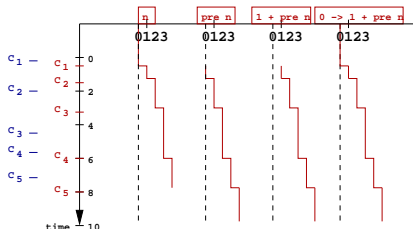
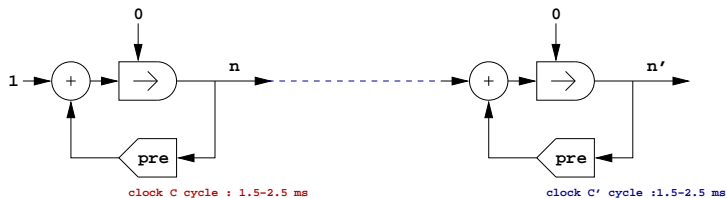
Toward a non-standard semantics ?

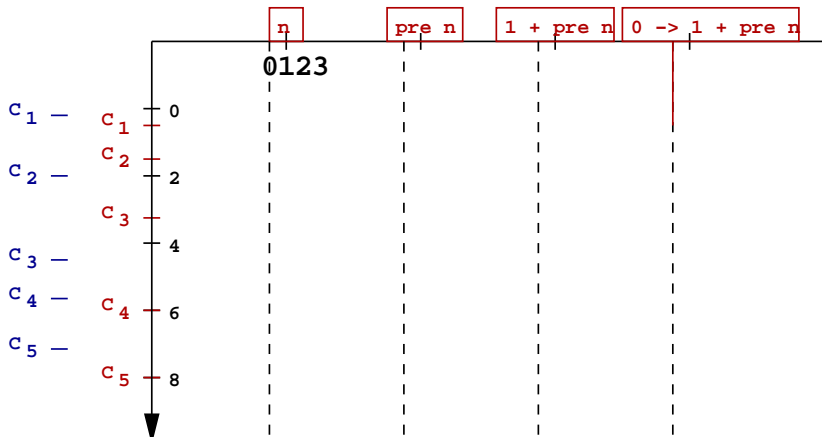
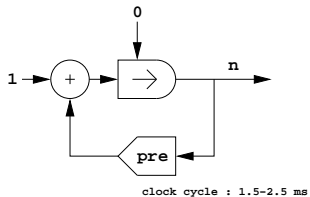
Case of several systems, each with its (imprecise) clock :

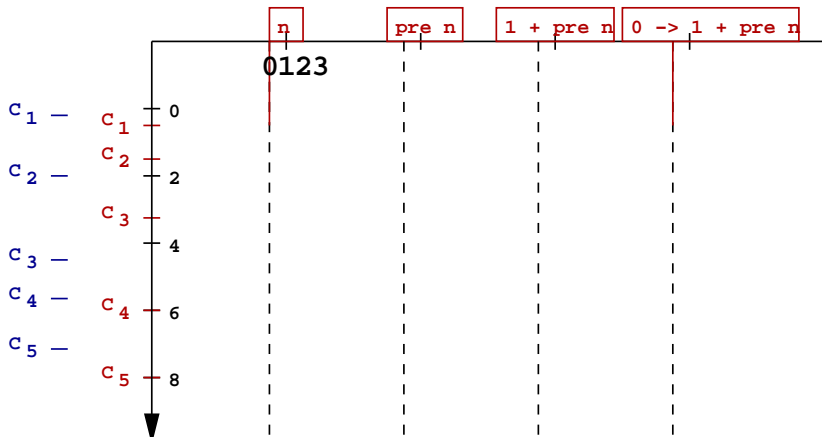
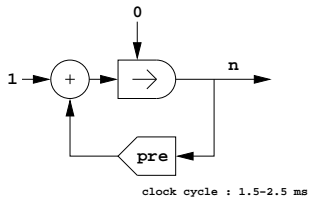


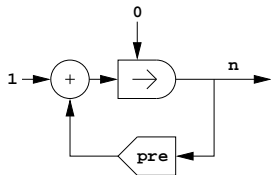
Toward a non-standard semantics ?

A continuous-time semantics allow the study of the communication of several systems

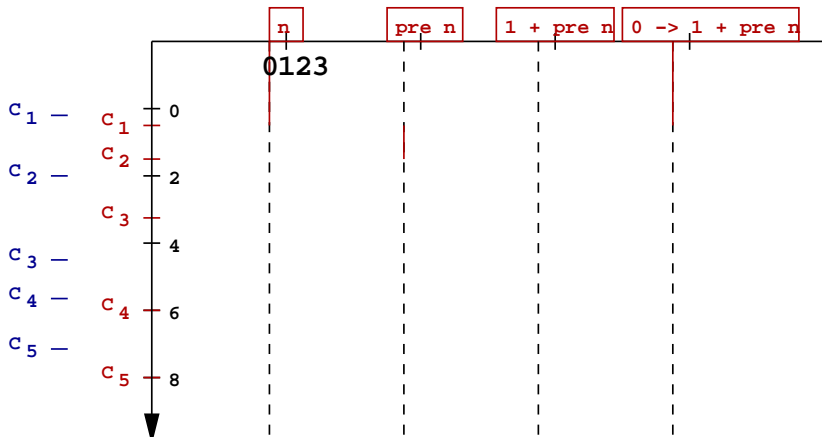


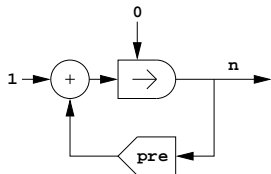




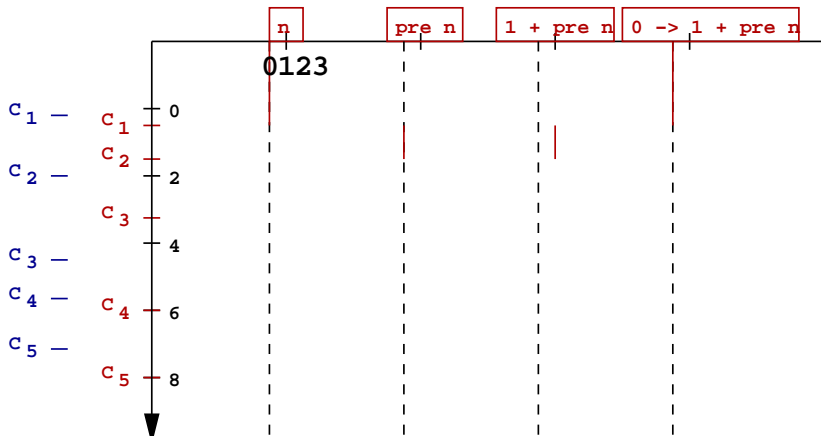


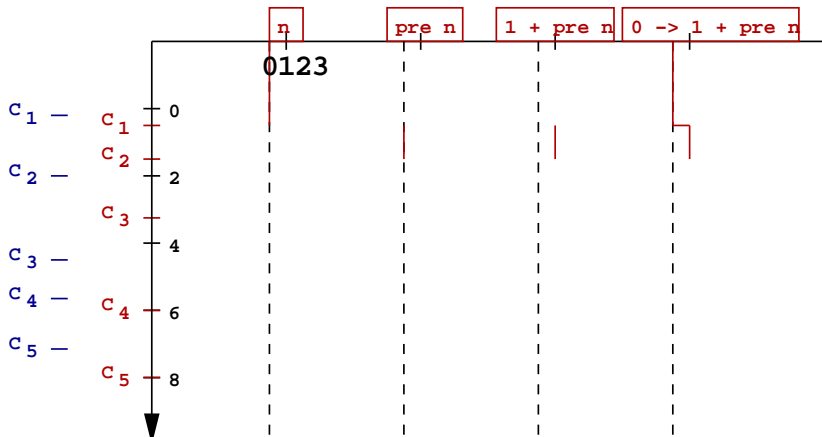
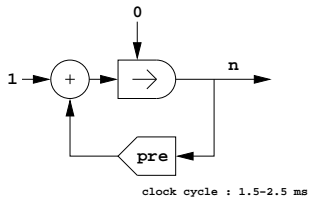
clock cycle : 1.5-2.5 ns

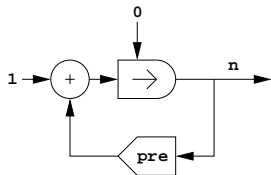




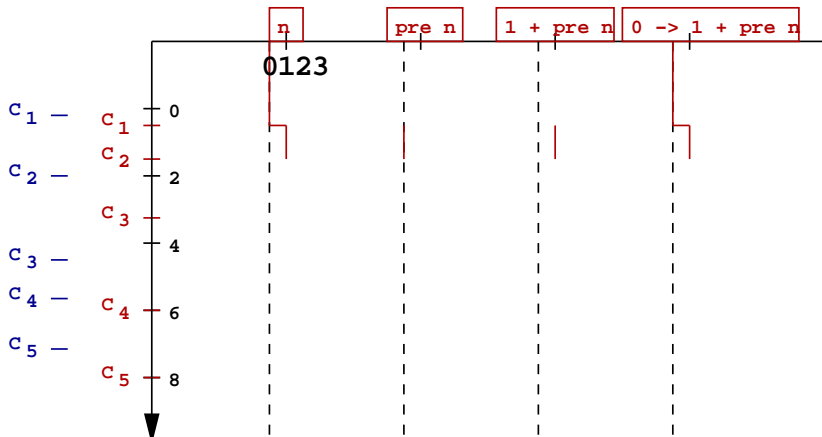
clock cycle : 1.5-2.5 ms

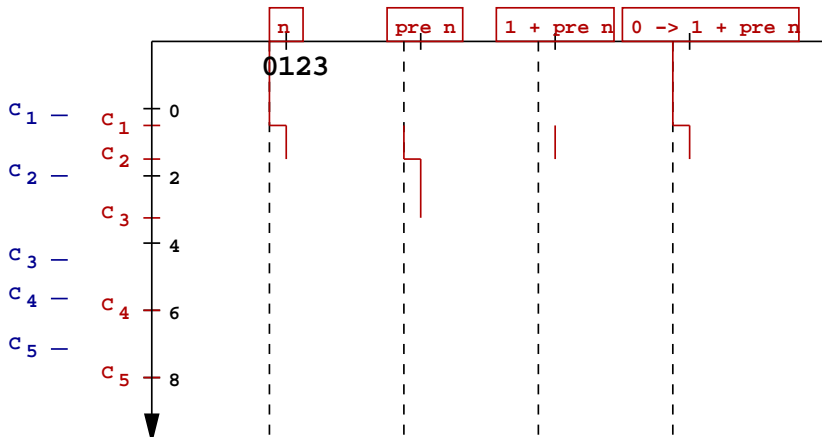
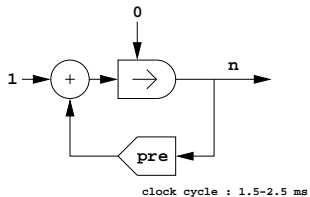


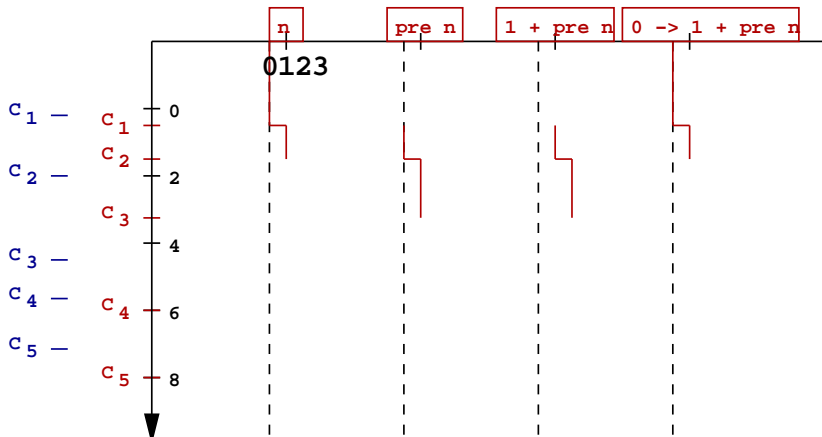
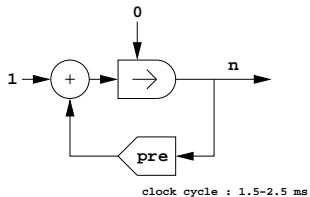


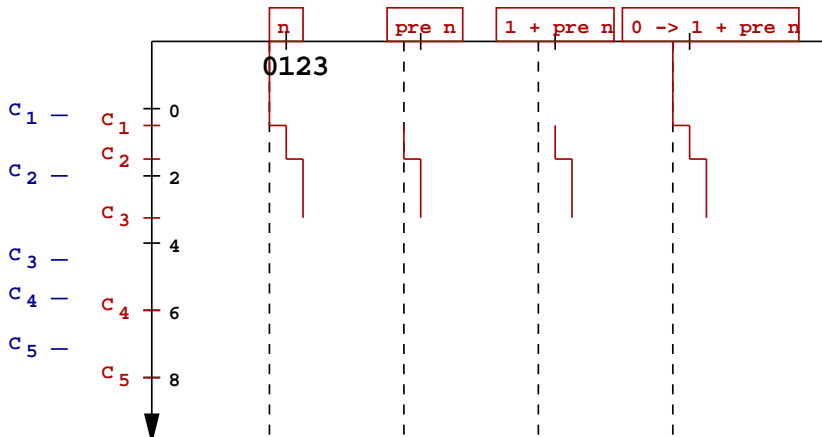
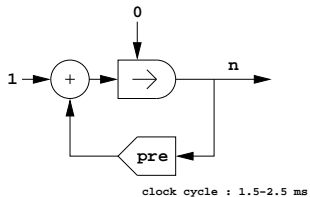


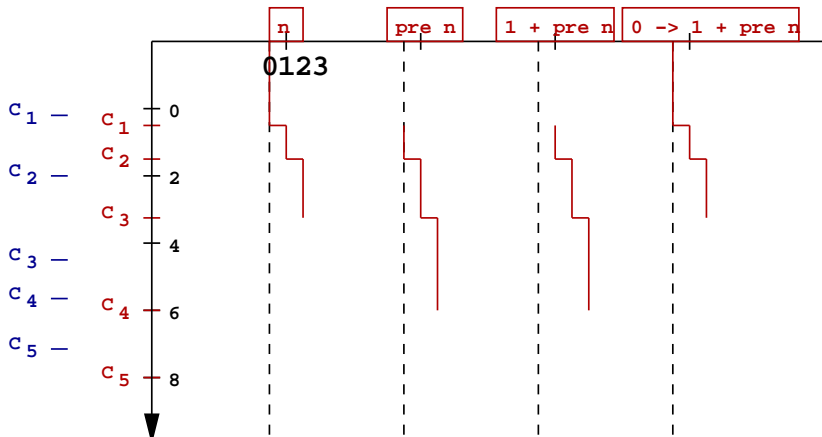
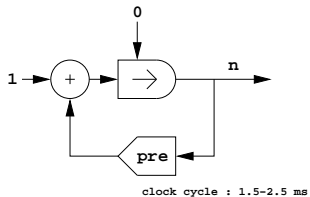
clock cycle : 1.5-2.5 ms

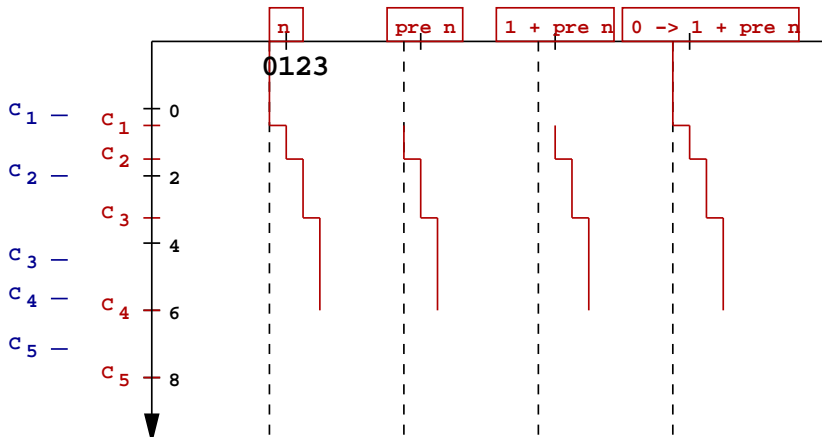
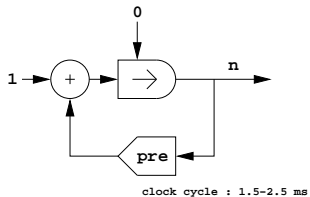


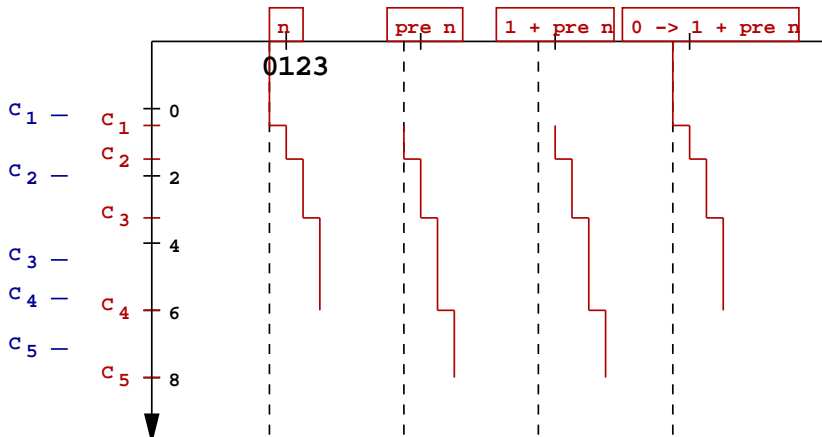
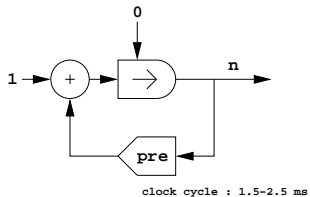


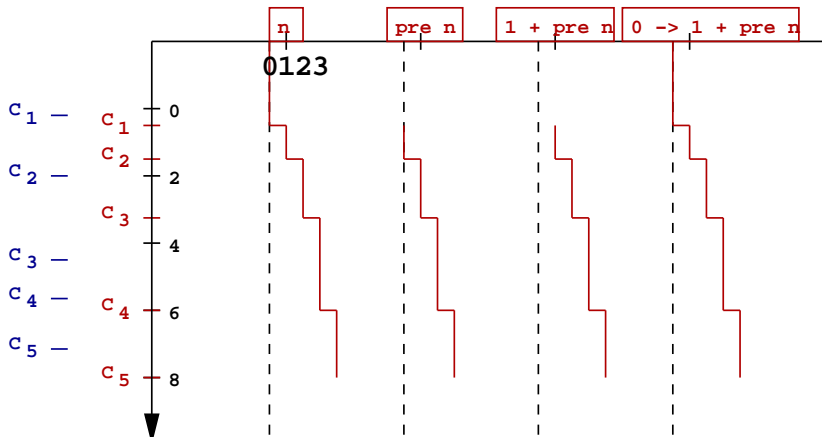
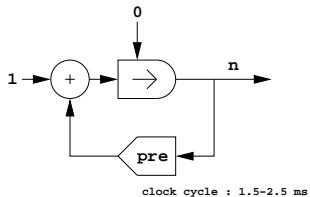




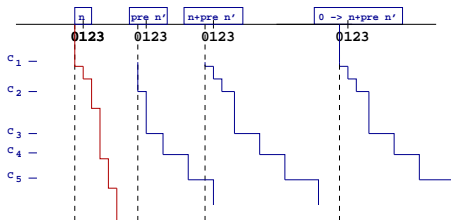
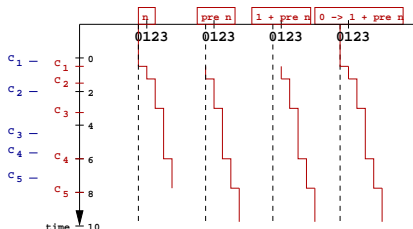
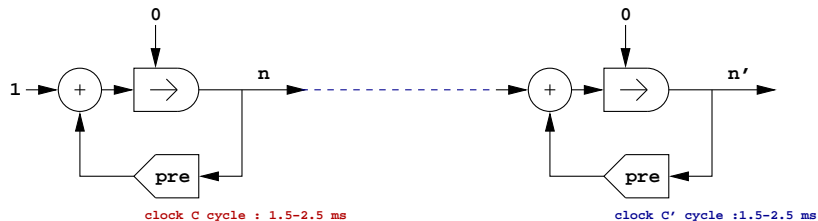






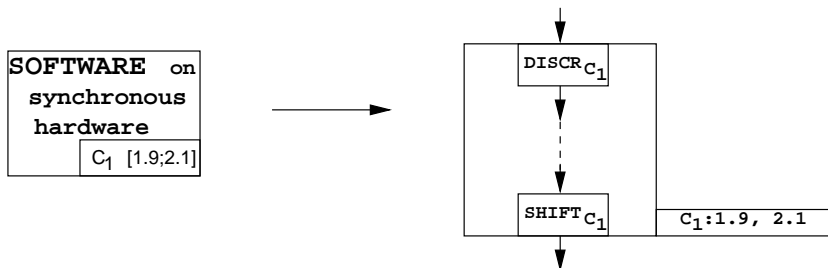


Toward a non-standard semantics ?



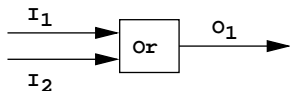
Behavior of a standard synchronous system

- A clock is a function $:\mathbb{N} \rightarrow \mathbb{R}^+$
- connected to the parameter $:\ [\alpha, \beta]$, with $\alpha, \beta \in \mathbb{R}^+$ and $0 < \alpha \leq \beta$
- a clock c satisfies $[\alpha, \beta]$ iff $c_{n+1} - c_n \in [\alpha, \beta]$



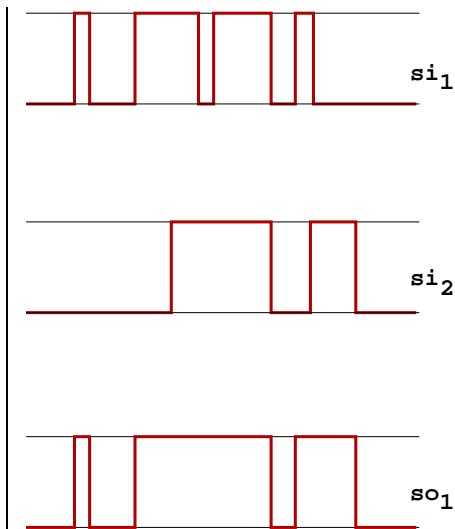
- $DISCR_{C_1}$ models the periodic reading of a blackboard (at the input of a synchronous unit)
- $SHIFT_{C_1}$ models the waiting of the next clock tick, then the emission of the result precisely at this tick

Semantics of non-temporal operators

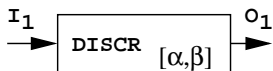


$$so_1(t) = \begin{cases} \bullet \text{ true} \\ \text{if } si_1(t) = \text{true} \\ \text{or } si_2(t) = \text{true} \\ \bullet \text{ false} \text{ else} \end{cases}$$

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



Semantics of temporal operators

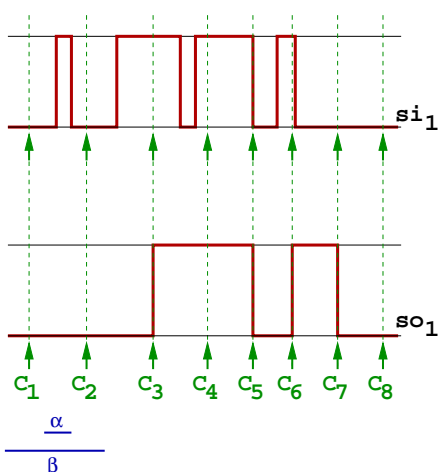


There exists a clock C of parameter $[\alpha, \beta]$

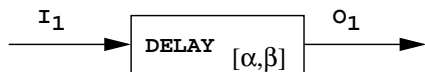
i.e. : $c_{n+1} - c_n \in [\alpha, \beta]$ such that

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

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



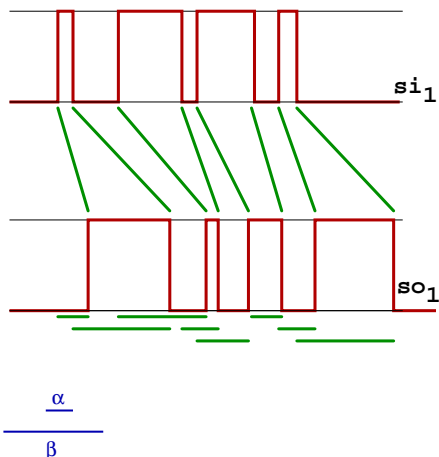
Semantics of temporal operators



There exists a function δ
 increasing bijection of $\mathbb{R} \rightarrow \mathbb{R}$
 of parameter $[\alpha, \beta]$ i.e. :
 $\forall t \in \mathbb{R}, \delta(t) - t \in [\alpha, \beta]$ such that

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

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



Concrete semantics

- A control point \triangleq graphical equivalent of a variable
- The set of control points is denoted by P .

Concrete semantics

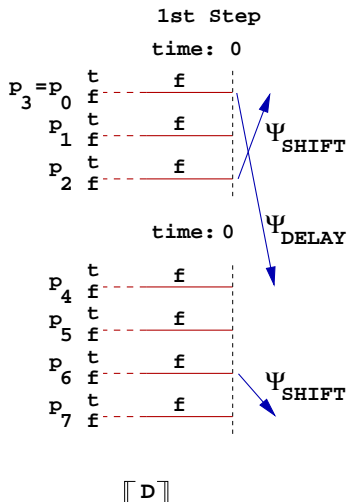
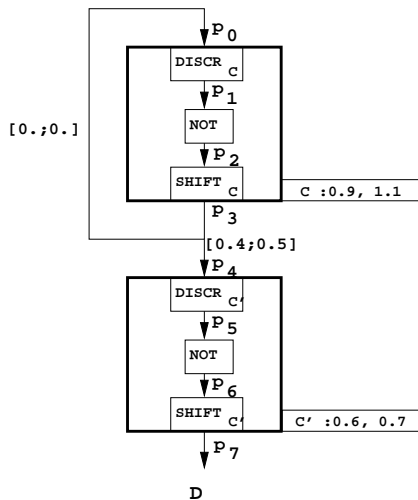
- A control point \triangleq graphical equivalent of a variable
- The set of control points is denoted by P .
- D : set of synchronous systems, $\llbracket D \rrbracket \in P \rightarrow \mathcal{P}(\mathbb{R}^+ \rightarrow \mathbb{B})$

Concrete semantics

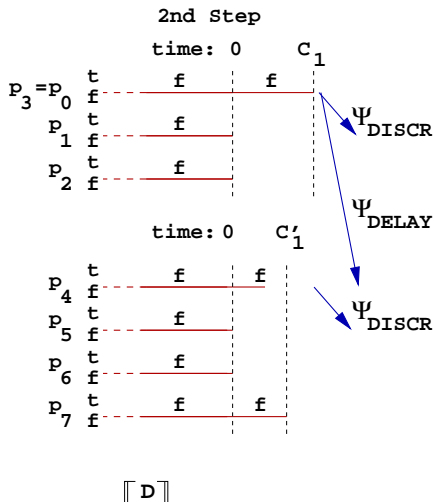
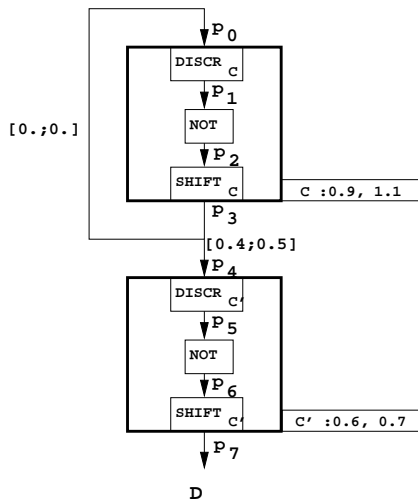
- A control point \triangleq graphical equivalent of a variable
- The set of control points is denoted by P .
- D : set of synchronous systems, $\llbracket D \rrbracket \in P \rightarrow \mathcal{P}(\mathbb{R}^+ \rightarrow \mathbb{B})$

- $s_j \in \llbracket D \rrbracket(p_i)$ **iff** exists $\left(\begin{array}{ccc} p_1 & \mapsto & s_1 \\ & \vdots & \\ p_i & \mapsto & s_i \\ & \vdots & \\ p_{\#P} & \mapsto & s_{\#P} \end{array} \right)$ satisfying all equations of the gates in D .

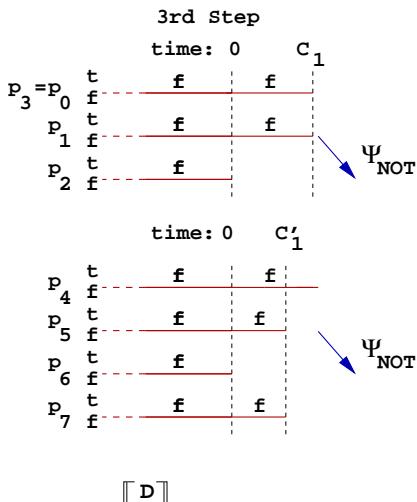
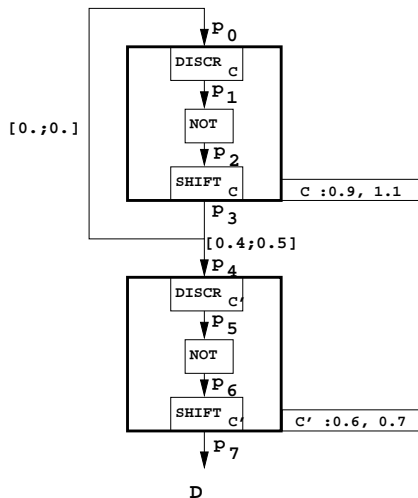
Syntax and semantics



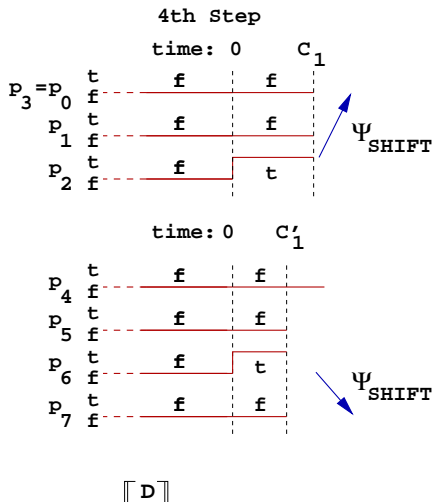
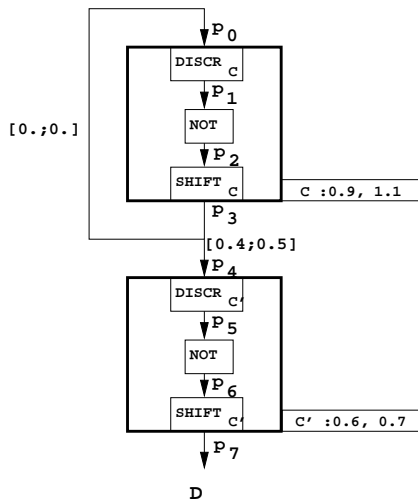
Syntax and semantics



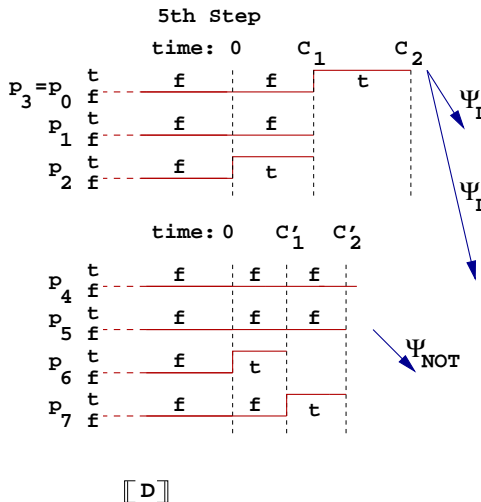
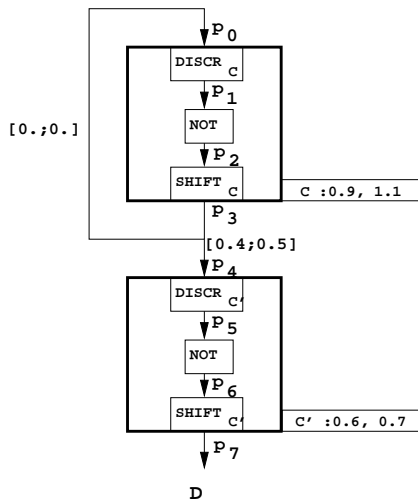
Syntax and semantics



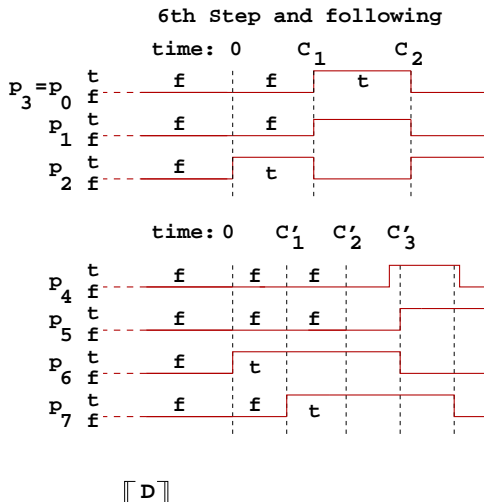
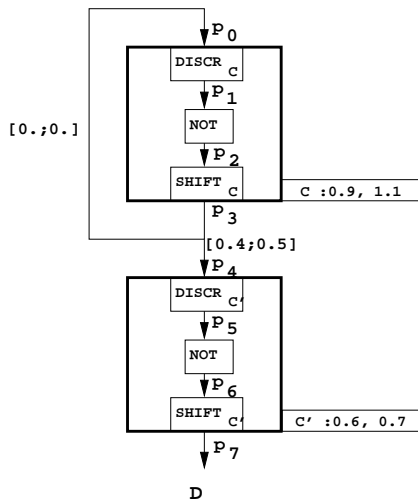
Syntax and semantics



Syntax and semantics

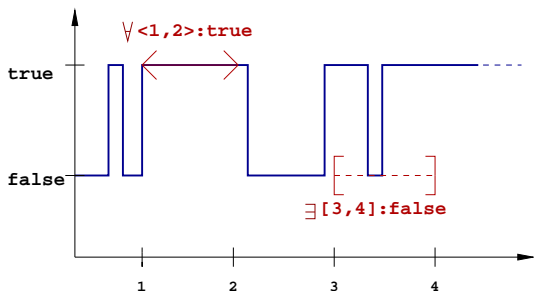


Syntax and semantics



Abstract Analysis

1st abstract domain : constraints



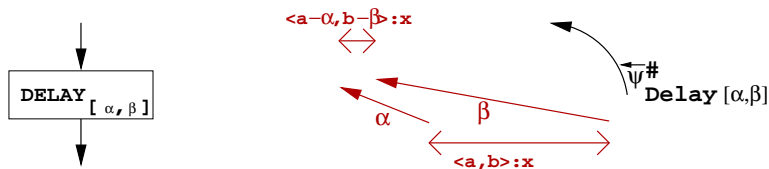
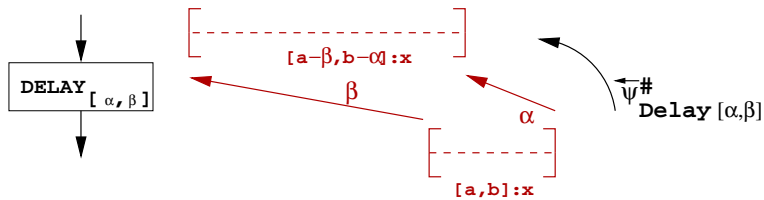
- The constraint $\exists[a; b] : x$ asserts that the signals take the value x **at least once** during $[a; b]$.
- The constraint $\forall \langle a; b \rangle : x$ asserts that the signals take the value x **during the whole interval** $[a; b]$.

1st abstract domain : constraints

- Allows the expression of numerous **temporal properties**
- The signal s takes the value *true* during more than 3 seconds without the alarm a being raised less than 1 second later :

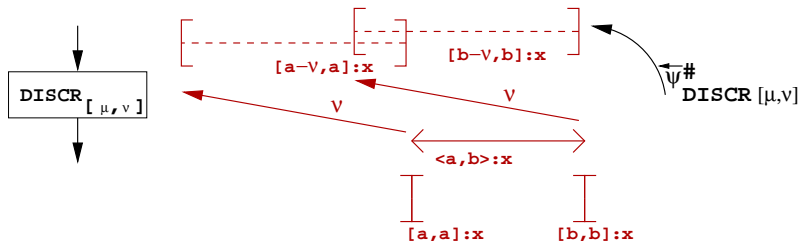
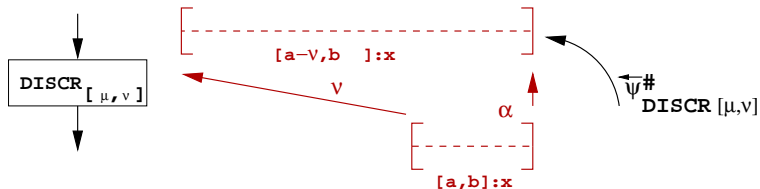
$$s : \forall \langle \delta; \delta + 3 \rangle : \text{True} \wedge a : \forall \langle \delta + 3; \delta + 4 \rangle : \text{False}$$

Abstract Operators and Constraints : example



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

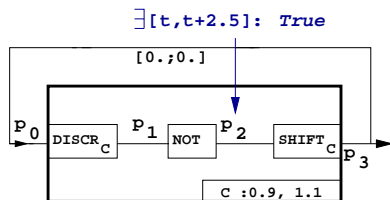
Abstract Operators and Constraints : example



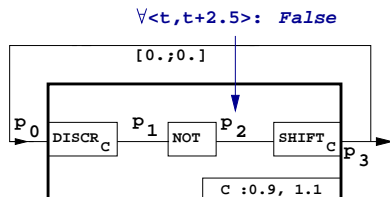
- $\overleftarrow{\Psi}_{\text{DISCR}[\mu, \nu]}^{\#}(\exists[a; b] : x) \triangleq \exists[a - \nu; b] : x$
- $\overleftarrow{\Psi}_{\text{DISCR}[\mu, \nu]}^{\#}(\forall\langle a; b \rangle : x) \triangleq \bigwedge_{u \in [a, b]} \exists[u - \nu; u] : x$

Analysis inside the abstract domain of **Constraints**

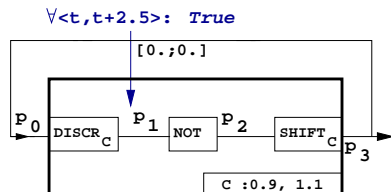
- Proving the following abstract property.



Analysis inside the abstract domain of **Constraints**

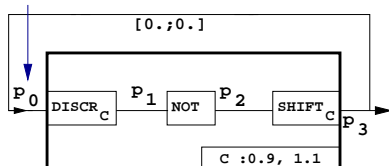


Analysis inside the abstract domain of **Constraints**

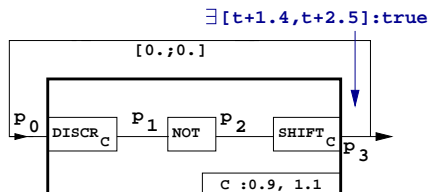


Analysis inside the abstract domain of **Constraints**

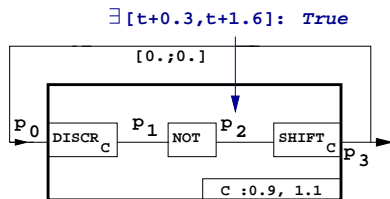
$\exists [t+1.4, t+2.5]: \text{True}$



Analysis inside the abstract domain of **Constraints**



Analysis inside the abstract domain of **Constraints**



At point p_2 : 2 constraints have to be satisfied :

- $\forall \langle t; t + 2.5 \rangle : \text{False}$
- $\exists [t + 0.3; t + 1.6] : \text{True}$

which is impossible and thus invalidates the initial hypothesis

$\forall \langle t; t + 2.5 \rangle : \text{False}$.

- Thus $\exists [t; t + 2.5] : \text{True}$ is certified.

Analysis by interpretation abstract

- $\llbracket D \rrbracket$: semantics of the set D of systems.
- For any property P , $[P]$ is the set of behaviors that satisfy P .
- **Former goal** : Prove that $\llbracket D \rrbracket \subseteq [P]$.

Analysis by interpretation abstract

- $\llbracket D \rrbracket$: semantics of the set D of systems.
- For any property P , $\llbracket P \rrbracket$ is the set of behaviors that satisfy P .
- **Former goal** : Prove that $\llbracket D \rrbracket \subseteq \llbracket P \rrbracket$.
- **Alternative** : Prove that $\llbracket D \rrbracket \cap \llbracket \neg P \rrbracket = \emptyset$.

Analysis by interpretation abstract

- $\llbracket D \rrbracket$: semantics of the set D of systems.
- For any property P , $\llbracket P \rrbracket$ is the set of behaviors that satisfy P .
- **Former goal** : Prove that $\llbracket D \rrbracket \subseteq \llbracket P \rrbracket$.
- **Alternative** : Prove that $\llbracket D \rrbracket \cap \llbracket \neg P \rrbracket = \emptyset$.
- **Now** :

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

Analysis by interpretation abstract

- $\llbracket D \rrbracket$: semantics of the set D of systems.
- For any property P , $[P]$ is the set of behaviors that satisfy P .
- **Former goal** : Prove that $\llbracket D \rrbracket \subseteq [P]$.
- **Alternative** : Prove that $\llbracket D \rrbracket \cap [\neg P] = \emptyset$.

- **Now** :

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

- **New goal** :

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

Analysis by abstract interpretation

Theorem : If :

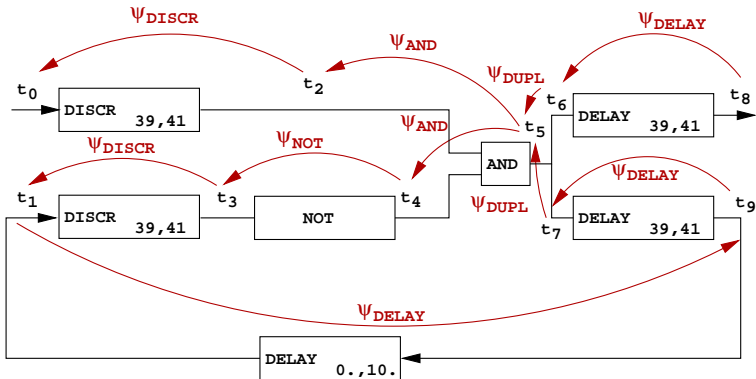
- F is continuous.
- $F^\#$ is continuous.
- $F \circ \gamma \subseteq \gamma \circ F^\#$

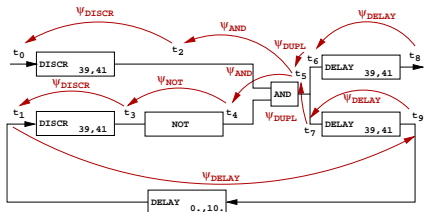
then : $\text{gfp}F \subseteq \gamma(\text{gfp}F^\#)$

- **New goal**, with $F \triangleq \Psi \cap Id$:

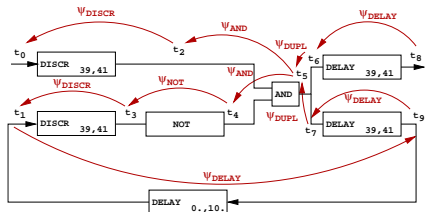
$$\text{gfp}_{[-P]}(\Psi^\# \cap^\# Id^\#) \subseteq^\# \emptyset^\# = \perp$$

Propagating the abstract information



An example : computing Ψ 

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

An example : computing Ψ 

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

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

$$\begin{pmatrix} \Psi_{\text{DISCR}_{[39,41]}}(t_2) \\ \Psi_{\text{DISCR}_{[39,41]}}(t_3) \\ \rho_1(\Psi_{\text{AND}}(t_5)) \\ \Psi_{\text{NON}}(t_4) \\ \rho_2(\Psi_{\text{AND}}(t_5)) \\ \Psi_{\text{DUPL}}(t_6, t_7) \\ \Psi_{\text{DELAY}_{[39-41]}}(t_8) \\ \Psi_{\text{DELAY}_{[39-41]}}(t_9) \\ t_8 \\ \Psi_{\text{DELAY}_{[0-10]}}(t_1) \end{pmatrix}$$

Operations on constraints : abstract conjunction

- $\lambda p. f \wedge p \cap^{\#} \lambda p. g \wedge p \triangleq \lambda p. (f \wedge p \wedge g \wedge p)$

Operations on constraints : abstract conjunction

$$\bullet \lambda p. f \wedge p \sqcap^{\#} \lambda p. g \wedge p \triangleq \lambda p. (f \wedge p \wedge g \wedge p)$$

 $\llbracket _ \rrbracket :x$
 $\llbracket _ \rrbracket :x$
 $\llbracket _ \rrbracket :x$
 \wedge
 \wedge
 \wedge
 $\longleftrightarrow : \neg x$
 $\longleftrightarrow :x$
 $\llbracket _ \rrbracket :x$
 \Downarrow
 \Downarrow
 \Downarrow
 \perp
 $\longleftrightarrow :x$
 $\llbracket _ \rrbracket :x$

Operations on constraints : abstract conjunction

$$\bullet \lambda p. f \wedge p \sqcap^{\#} \lambda p. g \wedge p \triangleq \lambda p. (f \wedge p \wedge g \wedge p)$$

$$\boxed{\quad} :x$$

$$\boxed{\quad} :x$$

$$\boxed{\quad} :x$$

$$\wedge$$

$$\wedge$$

$$\wedge$$

$$\longleftrightarrow : \neg x$$

$$\longleftrightarrow :x$$

$$\boxed{\quad} :x$$

$$\Downarrow$$

$$\Downarrow$$

$$\Downarrow$$

$$\perp$$

$$\longleftrightarrow :x$$

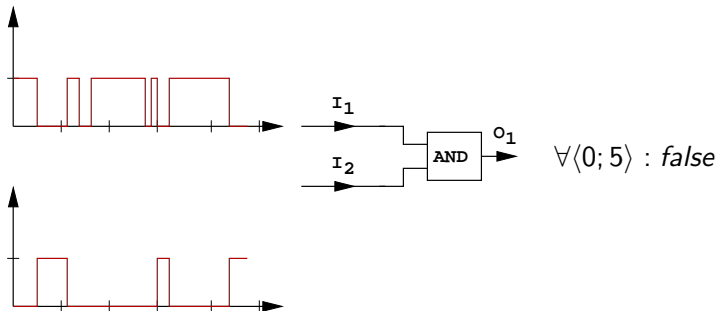
$$\boxed{\quad} :x$$

- 2 goals :

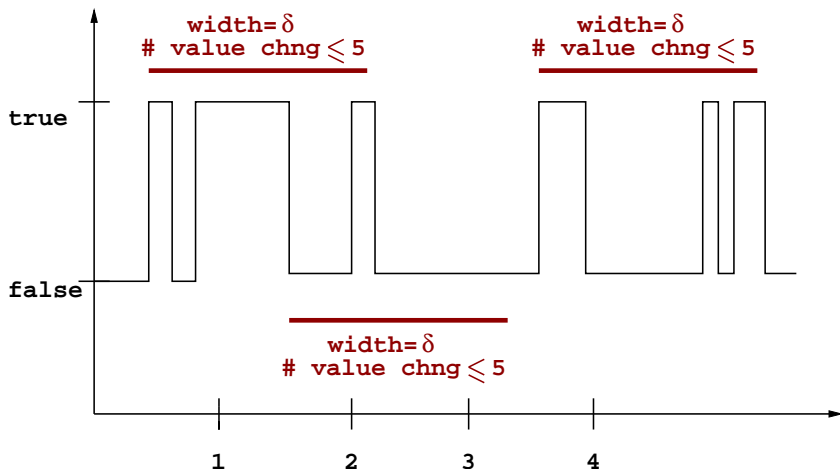
- ▶ An analysis that terminates! (without this $\subseteq^{\#} \emptyset^{\#}$ is not provable)
- ▶ A faster analysis

Weaknesses of the constraints domain

- this domain is precise in the case of : DELAY, DISCR, SHIFT, NOT,
- huge loss of precision in the case of : AND, OR, XOR

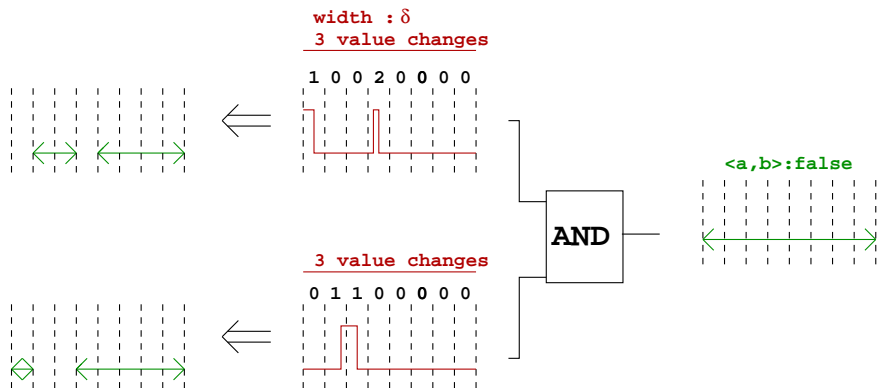


2nd Abstract Domain : value changes counting domain

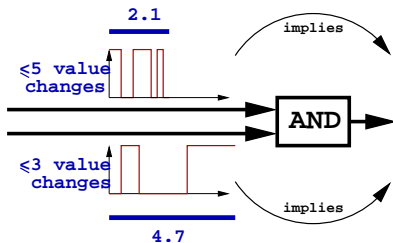


- Allows the expression of “stability” specifications .

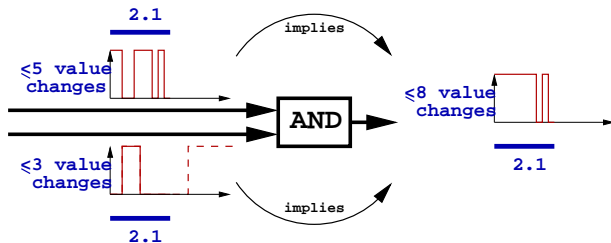
2nd Abstract Domain : value changes counting domain



A non-temporal abstract operator

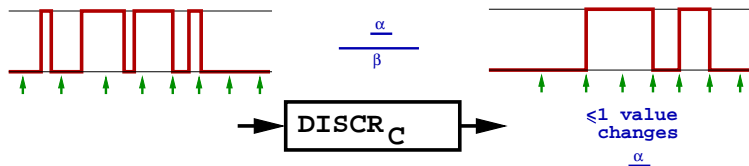


A non-temporal abstract operator



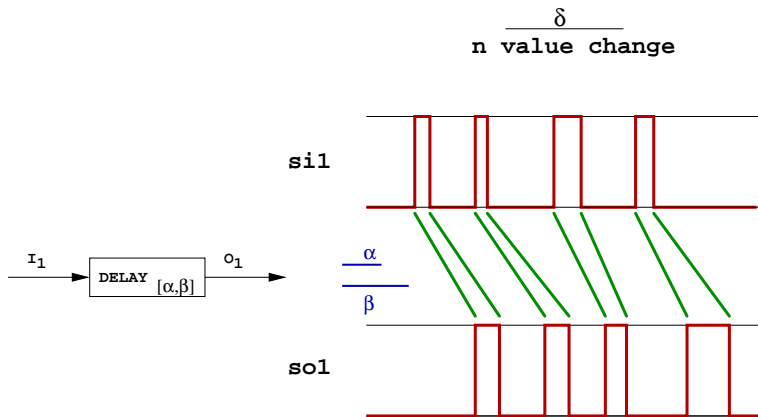
$$\vec{\Psi}_{\text{AND}}^{\#}((n_1, \delta_1)_{\mathcal{N}}, (n_2, \delta_2)_{\mathcal{N}}) \triangleq (\tilde{n}_1 + \tilde{n}_2, \tilde{\delta}_1)_{\mathcal{N}}$$

A temporal abstract operator

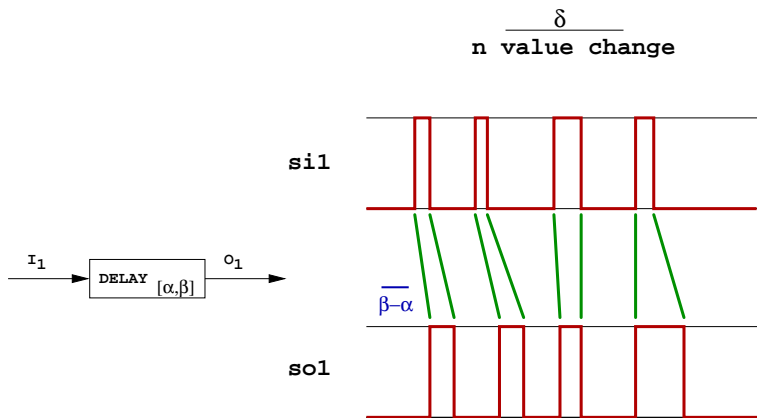


- $[\alpha, \beta]$ parameter of clock C
- $\vec{\Psi}_{\text{DISCR}_{[\alpha, \beta]}}^{\#}(-) \triangleq (\geq 1, \alpha) \mathcal{N}_{\text{global}}$

A temporal abstract operator

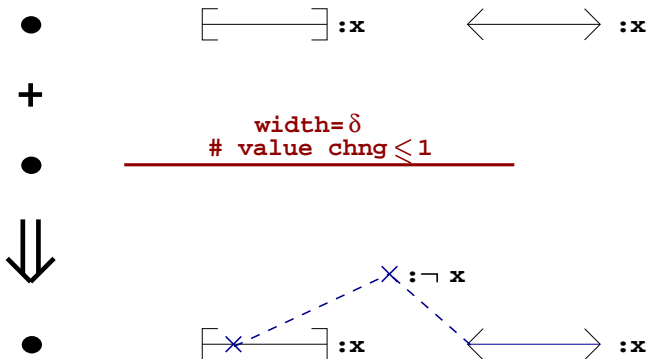


A temporal abstract operator

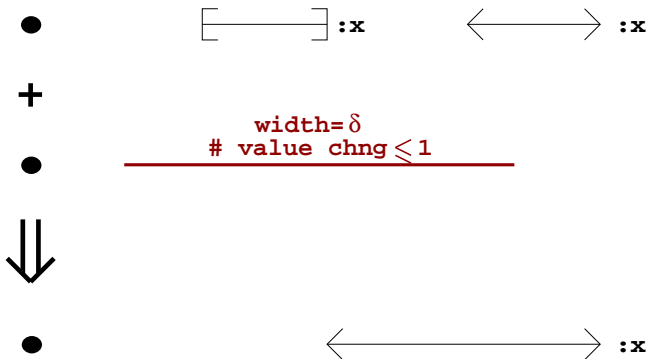


$$\vec{\Psi}^\# (n, \delta)_{\mathcal{N}} \triangleq (n, \delta - \beta + \alpha)$$

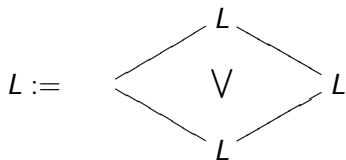
Reduced Product Constraints- value changes countings



Reduced Product Constraints-value changes countings



Disjunctive Reduced Product Constraints-value changes countings



| $\mathcal{C} :: L$

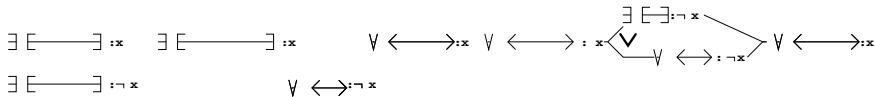
| $\mathcal{N}_{\text{local}} :: L$

| \square (empty element)

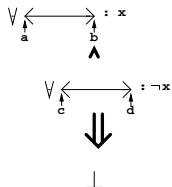
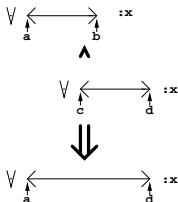
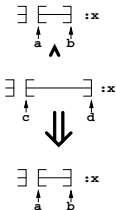
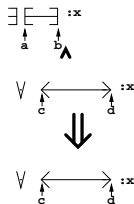
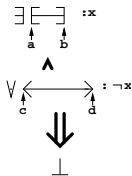
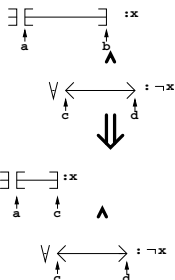
Disjunctive Reduced Product Constraints-value changes

countings

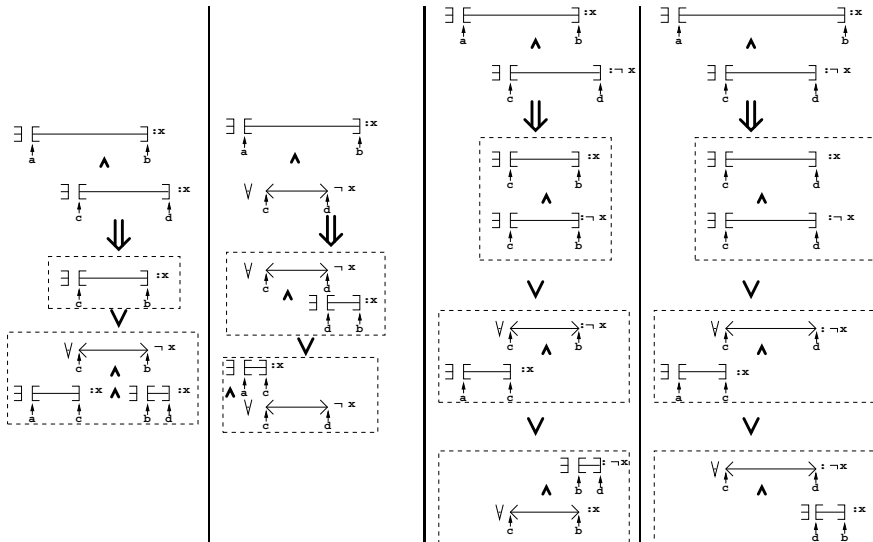
An example



Disjunctive Reduced Product : non-overlapping



Disjunctive Reduced Product : non-overlapping



Disjunctive Reduced Product : non-overlapping Hypotheses

- If $L_1 = \begin{array}{c} L_2 \\ \diagdown \quad \diagup \\ \vee \\ \diagup \quad \diagdown \\ L_3 \end{array} L_4$ and c in L_2 or in L_3 , and d in L_4

then $\text{right_limit}(c) \leq \text{left_limit}(d)$,

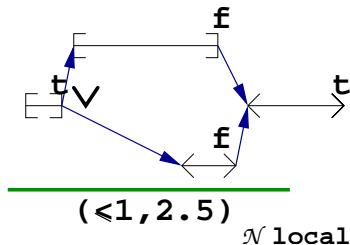
- If $L_1 = L_2 :: L_3$, c in L_2 , and d in L_3 , then $c <'_c d$ i.e.
 - ▶ either $\text{right_limit}(c) \leq \text{left_limit}(d)$
 - ▶ or $c = \exists[a; b] : \text{false}$ and $d = \exists[a; b] : \text{true}$

Disjunctive Reduced Product Constraints-value changes

countings

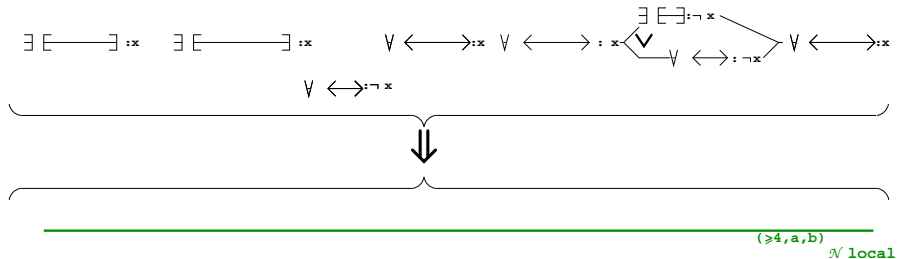
Reductions

- 1st type of reductions : discover incoherence of sub-elements and remove them.
- Example :



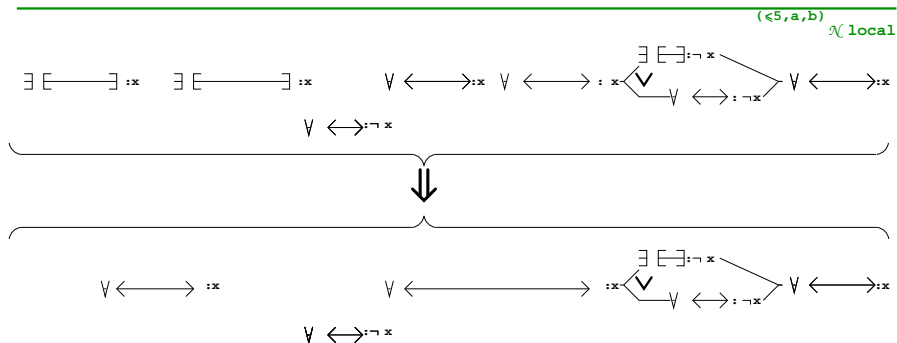
Disjunctive Reduced Product Constraints-value changes countings

- 2nd type of reduction generating a value changes counting min. bound



Disjunctive Reduced Product Constraints-value changes countings

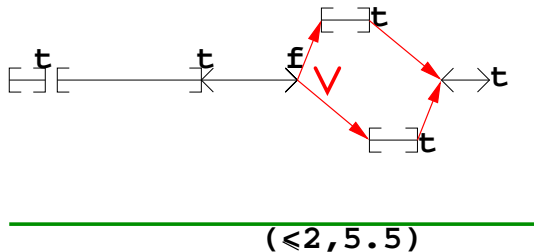
- 2nd type of reduction generating a value changes counting min. bound
- Limit case in presence of a precomputed maximal bound



Disjunctive Reduced Product Constraints-value changes

countings

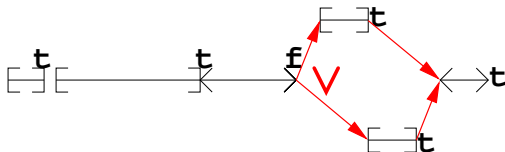
An example



Disjunctive Reduced Product Constraints-value changes

countings

An example

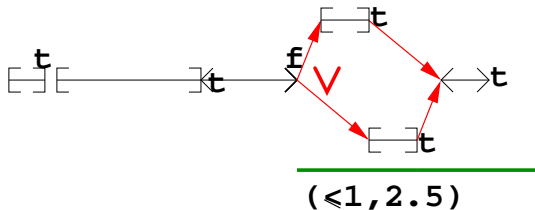


$(\leq 2, 5.125)$

Disjunctive Reduced Product Constraints-value changes

countings

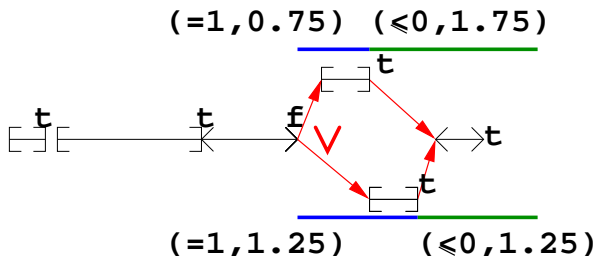
An example



Disjunctive Reduced Product Constraints-value changes

countings

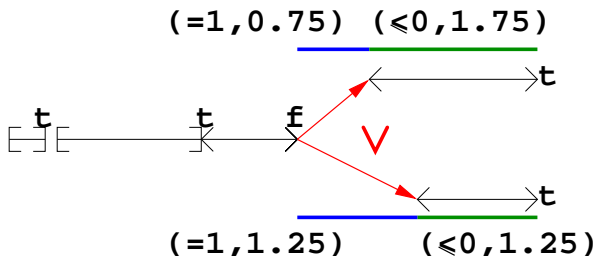
An example



Disjunctive Reduced Product Constraints-value changes

countings

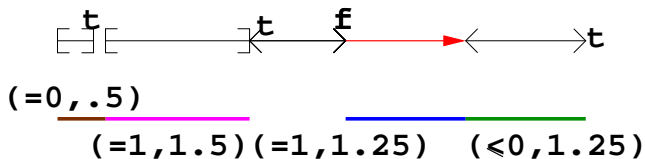
An example



Disjunctive Reduced Product Constraints-value changes

countings

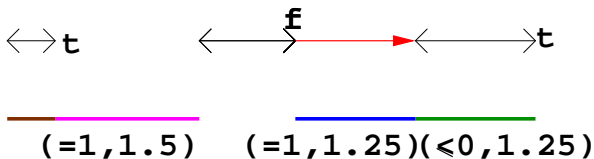
An example



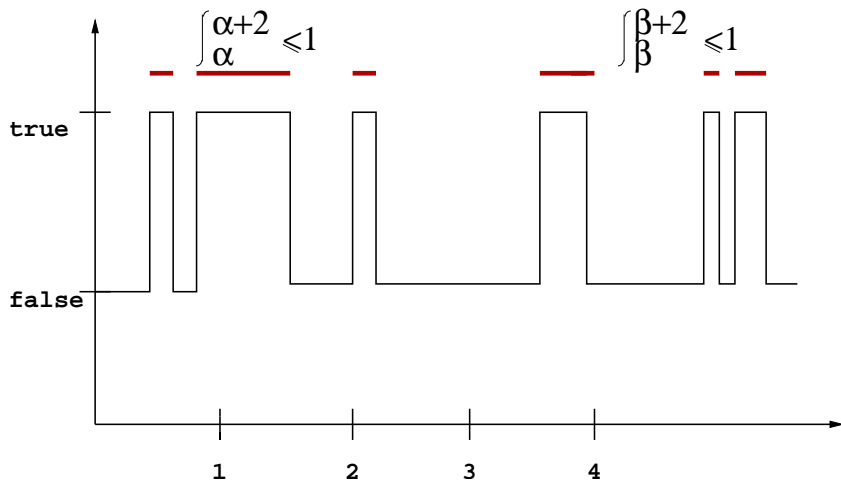
Disjunctive Reduced Product Constraints-value changes

countings

An example



3rd abstract domain : Integral bounding domain



Cooperation between domains

Integral bounding domain and Constraints Domain

- Let us assume that at a control point the following abstract informations are already known :
 - ▶ $0 \leq \int_x^{x+4} v(t)dt \leq 3.$
 - ▶ $\forall \langle x; x + 1 \rangle : \text{True}$ and $\forall \langle x + 3; x + 4 \rangle : \text{True}$

Cooperation between domains

Integral bounding domain and Constraints Domain

- Let us assume that at a control point the following abstract informations are already known :
 - $0 \leq \int_x^{x+4} v(t) dt \leq 3.$
 - $\forall \langle x; x + 1 \rangle : \text{True}$ and $\forall \langle x + 3; x + 4 \rangle : \text{True}$
- $\int_x^{x+4} v(t) dt$

$$\begin{aligned}
 &= \int_x^{x+1} v(t) dt + \int_{x+1}^{x+3} v(t) dt + \int_{x+3}^{x+4} v(t) dt \\
 &= 2 + \int_{x+1}^{x+3} v(t) dt,
 \end{aligned}$$

Cooperation between domains

Integral bounding domain and Constraints Domain

- Let us assume that at a control point the following abstract informations are already known :

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

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

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

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

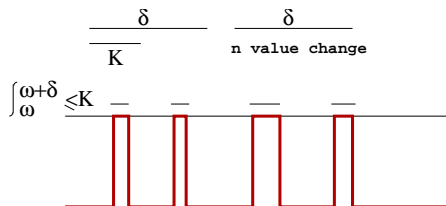
$$= 2 + \int_{x+1}^{x+3} v(t) dt,$$

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

Cooperation between domains : **optimizing** $\Psi^\#$

Integral bounding domain and **value changes counting domain**

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

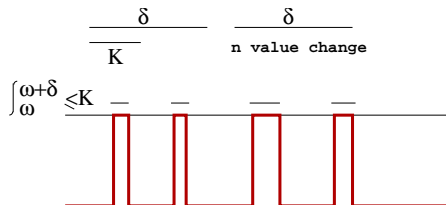
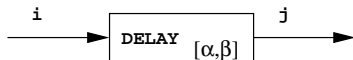


Cooperation between domains : optimizing $\Psi^\#$

Integral bounding domain and value changes counting domain

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

- $(n, \delta)_{\mathcal{N}}$

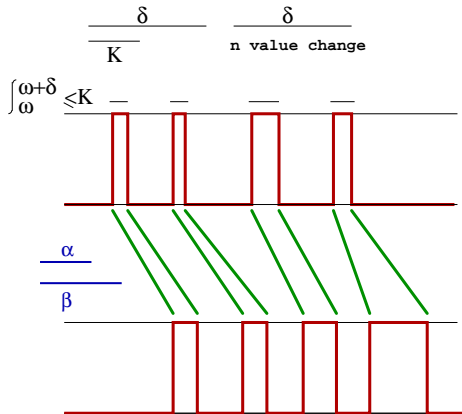
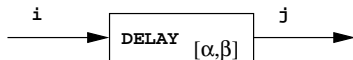


Cooperation between domains : optimizing $\Psi^\#$

Integral bounding domain and value changes counting domain

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

- $(n, \delta)_{\mathcal{N}}$

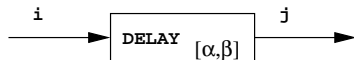


Cooperation between domains : optimizing $\Psi^\#$

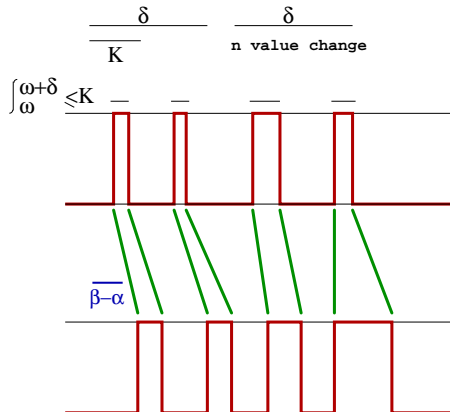
Integral bounding domain and value changes counting domain

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

- $(n, \delta)_{\mathcal{N}}$



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

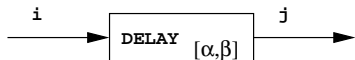


Cooperation between domains : optimizing $\Psi^\#$

Integral bounding domain and value changes counting domain

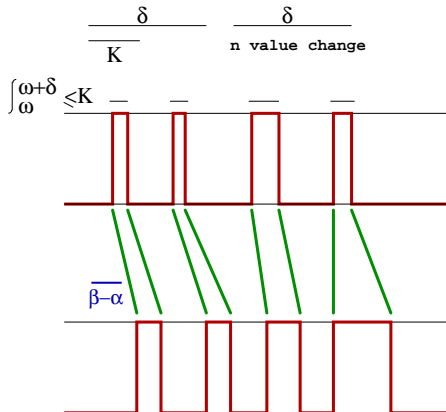
- $\forall \omega, \int_{\omega}^{\omega+\delta} s_i(t) dt \leq K$

- $(n, \delta)_{\mathcal{N}}$



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

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

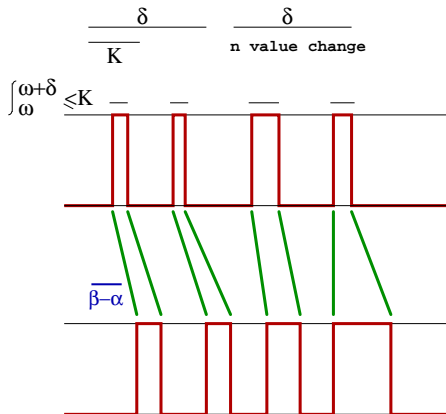
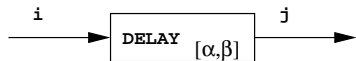


Cooperation between domains : optimizing $\Psi^\#$

Integral bounding domain and value changes counting domain

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

- $(n, \delta)_{\mathcal{N}}$



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

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

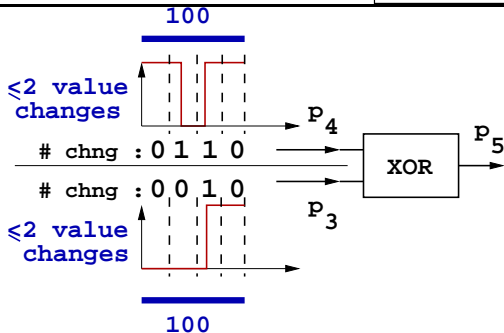
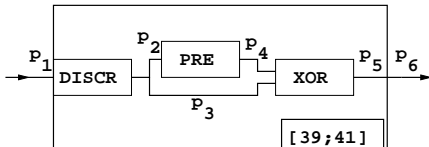
thus, if $\delta - \beta + \alpha \geq 0$:

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

- $\vec{\Psi}_{\text{DELAY}_{[\alpha;\beta]}}^\# \left(\int_{\Delta=\delta} \leq K \right) \triangleq \int_{\Delta=\delta-\beta+\alpha} \leq K + n(\beta - \alpha)$

Cooperation example of the 3 abstract domains

width=100
value chng ≤ 1



$$95 \leq \int_{\alpha}^{\alpha+100} |s_5(t)| dt \leq 100$$

- If the **reduced product** is denoted $\times^{\#}$,
- We may define abstract transfer functions **more precise** than

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

Real code Analysis

Analysis of the redundancy and of the vote

- Several redundant computation units
- Periodically one of the results is chosen among the outputs of these units (or an average value in the case of floats)
- Which one to choose? When to choose?

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
```

Execution of the voter with $nmax=2$ and $x0=0$

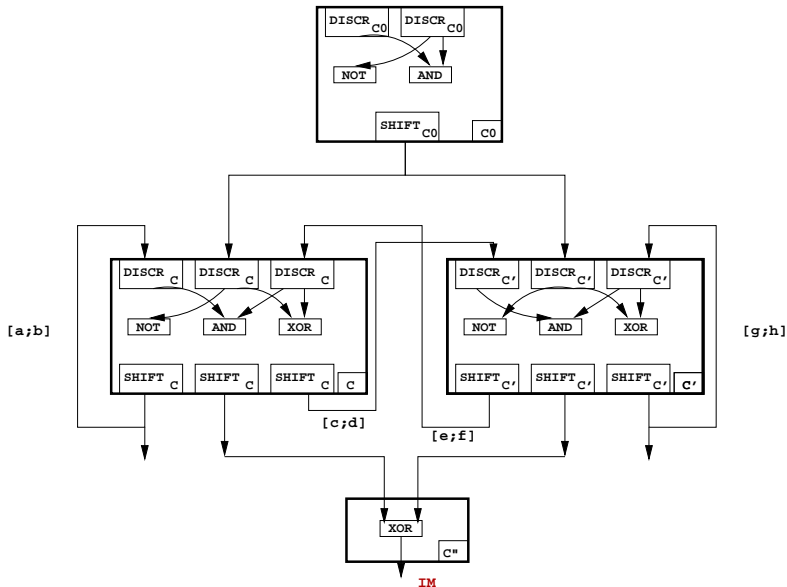
```
##### STEP 1 #####
x1 (integer)? 1
x2 (integer)? 1
x = 1, alarm = false
##### STEP 2 #####
x1 (integer)? 1
x2 (integer)? 2
x = 1, alarm = false
##### STEP 3 #####
x1 (integer)? 2
x2 (integer)? 1
x = 0, alarm = true
##### STEP 4 #####
x1 (integer)? 2
x2 (integer)? 1
x = 0, alarm = false
##### STEP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

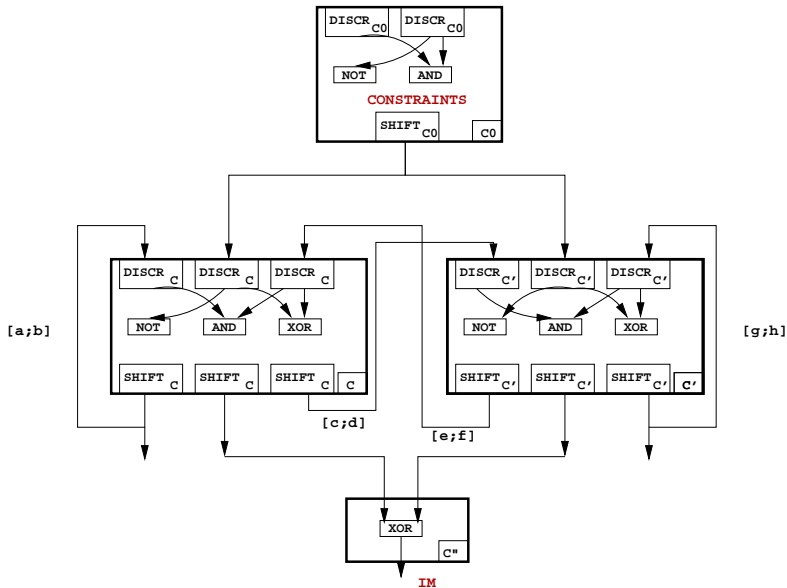
Difficulties with the voter

- In the case of unstable or desynchronized input signals , `alarm` is always set to *true* !
- We thus need stabilized input signals , that yet “look like” sensors values

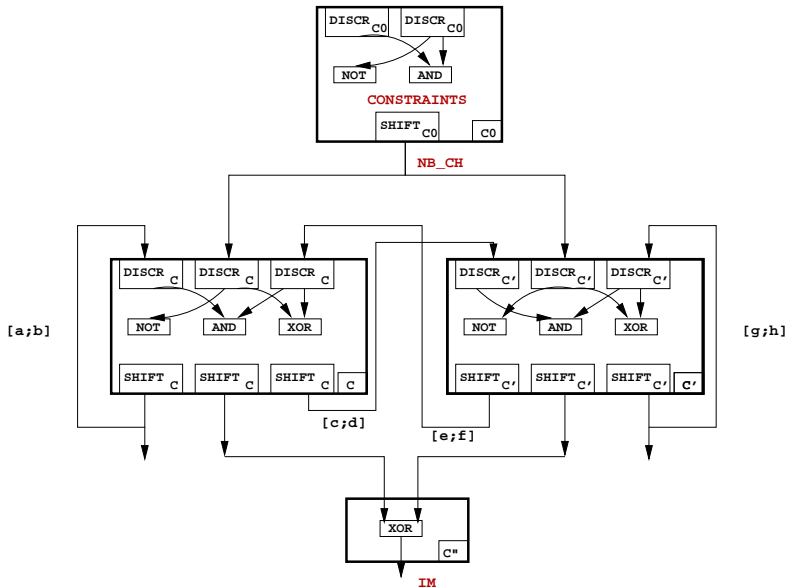
Effective propagation of the abstract information



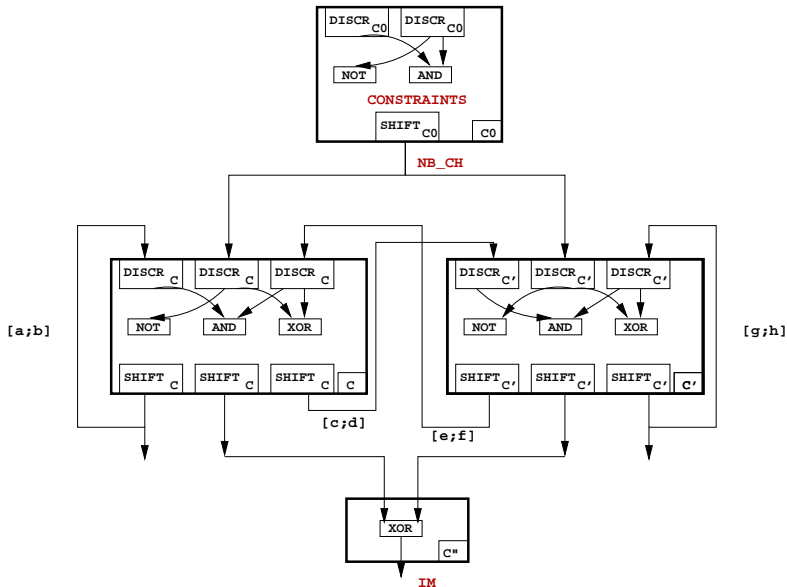
Effective propagation of the abstract information



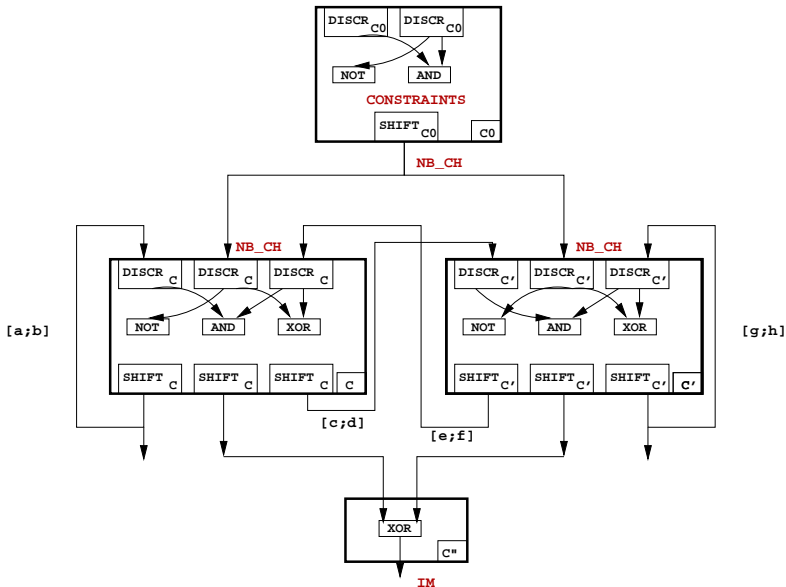
Effective propagation of the abstract information



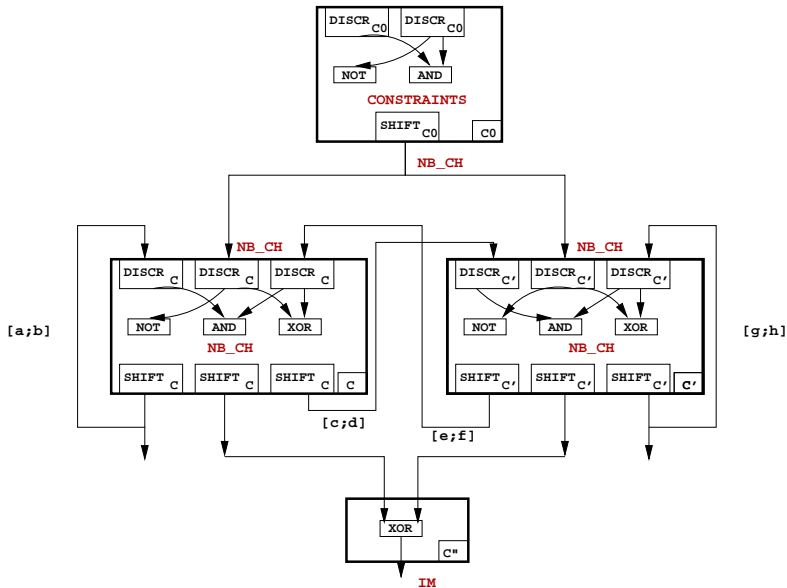
Effective propagation of the abstract information



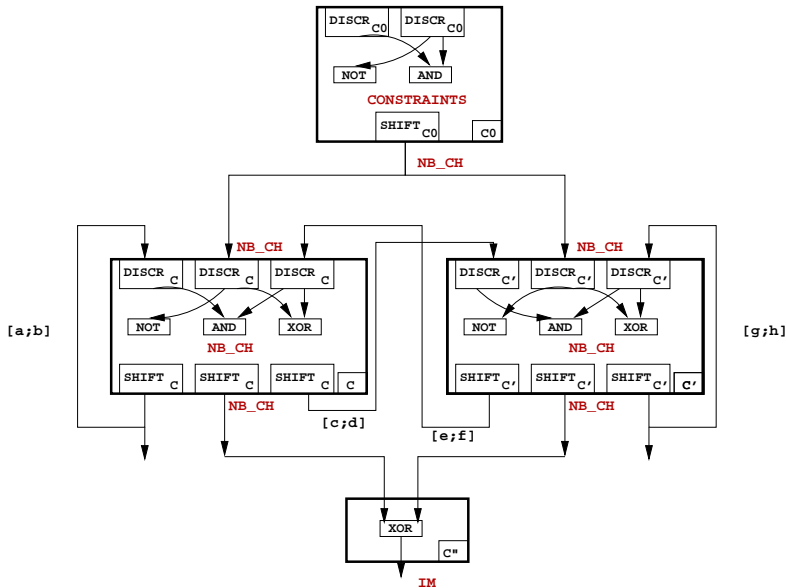
Effective propagation of the abstract information



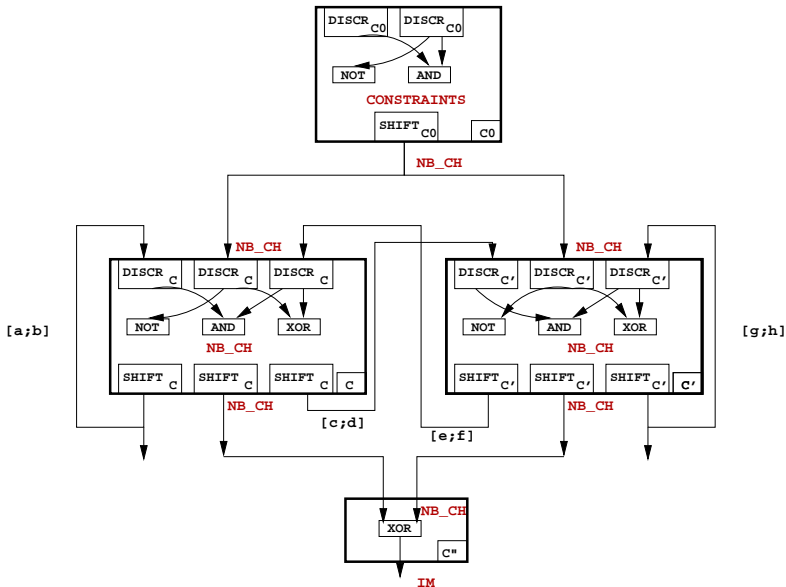
Effective propagation of the abstract information



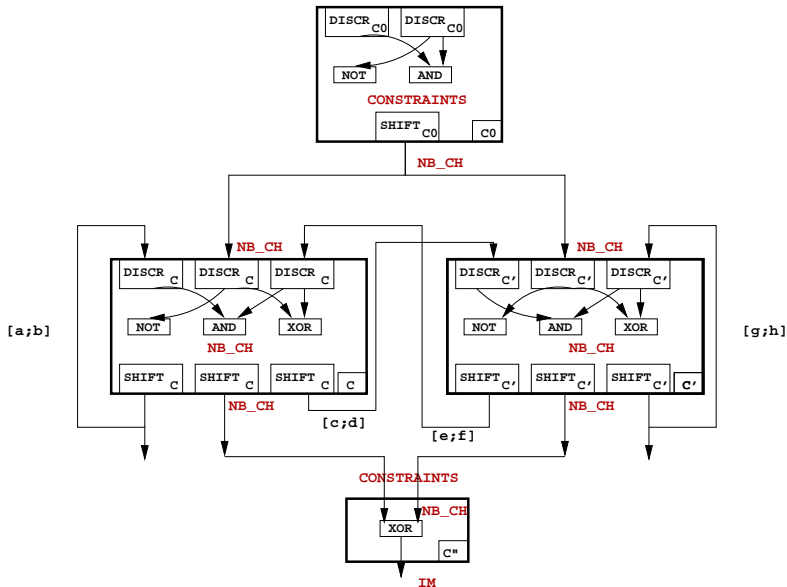
Effective propagation of the abstract information



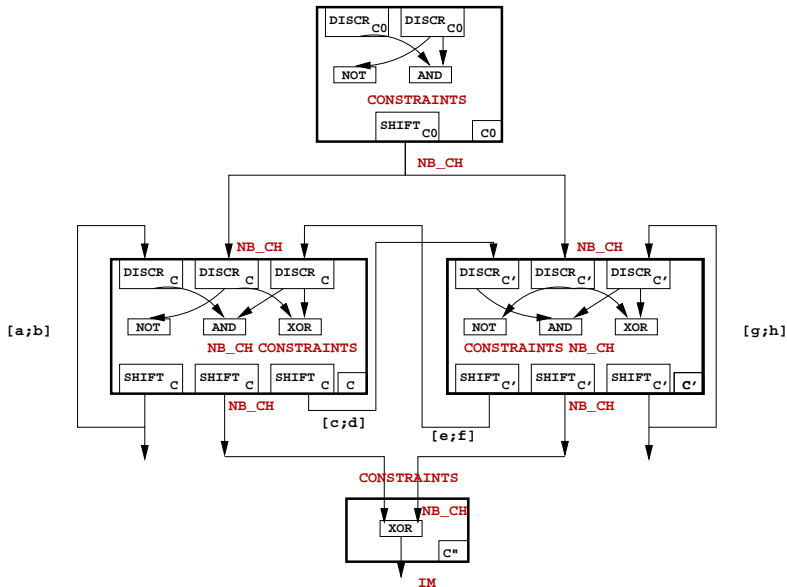
Effective propagation of the abstract information



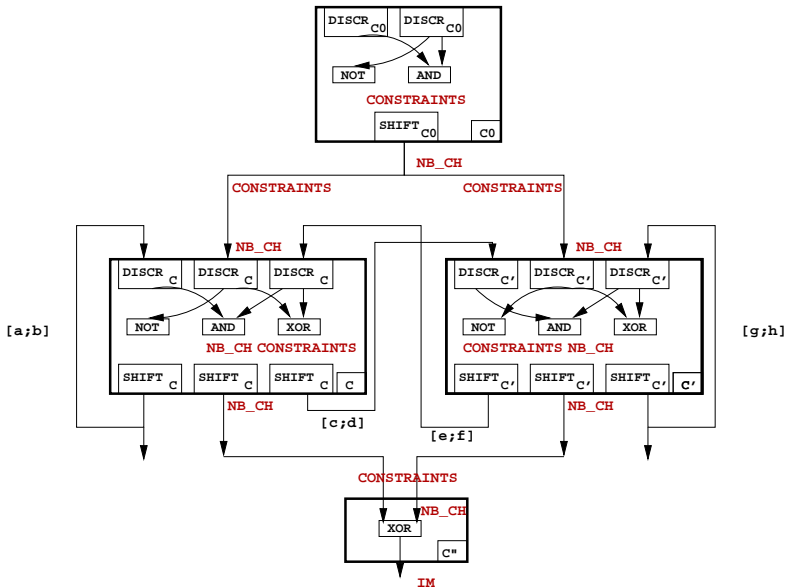
Effective propagation of the abstract information



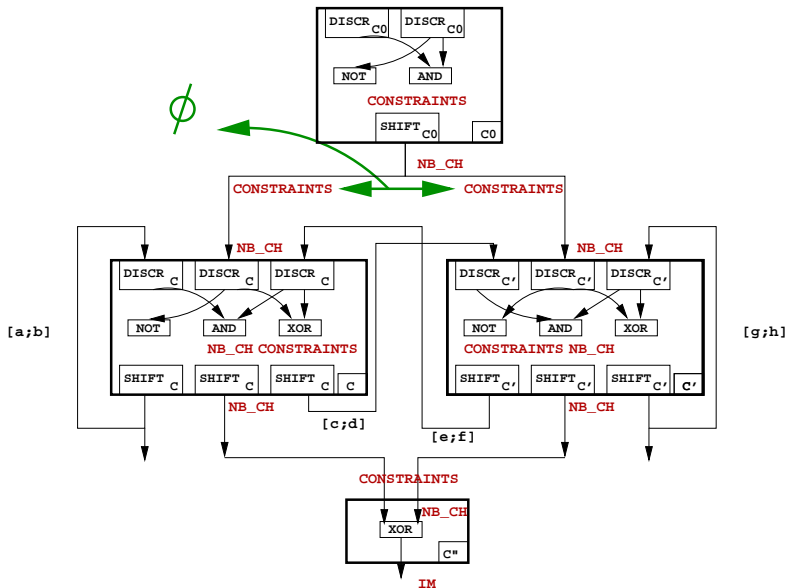
Effective propagation of the abstract information



Effective propagation of the abstract information



Effective propagation of the abstract information



Conclusion

- Previous semantics for communicating synchronous systems very different from the actual execution of the programs with imperfect clock and non-instantaneous communication.

- Previous **semantics** for **communicating synchronous systems** very different from the actual execution of the programs with imperfect clock and non-instantaneous communication.
- We model these differences thanks to a **continuous-time semantics**

- Previous **semantics** for **communicating synchronous systems** very different from the actual execution of the programs with imperfect clock and non-instantaneous communication.
- We model these differences thanks to a **continuous-time semantics**
- But
 - ▶ The **proofs of correction** are much more difficult to build (by hand : impossible)
 - ▶ The automatic analyses need *ad hoc* **abstract domains that takes the time into consideration**

- Previous **semantics** for **communicating synchronous systems** very different from the actual execution of the programs with imperfect clock and non-instantaneous communication.
- We model these differences thanks to a **continuous-time semantics**
- But
 - ▶ The **proofs of correction** are much more difficult to build (by hand : impossible)
 - ▶ The automatic analyses need *ad hoc* **abstract domains that takes the time into consideration**

- Previous **semantics** for **communicating synchronous systems** very different from the actual execution of the programs with imperfect clock and non-instantaneous communication.
- We model these differences thanks to a **continuous-time semantics**
- But
 - ▶ The **proofs of correction** are much more difficult to build (by hand : impossible)
 - ▶ The automatic analyses need *ad hoc* **abstract domains that takes the time into consideration**
- **abstract interpretation** allows a **safe incremental construction** of the usually difficult parts of an analyzer : modularity with respect to abstract domains, optimizations (Reduced product, optimized abstract operators)

Questions ?

Slides : www.di.ens.fr/~bertrane/

Contact : bertrane@di.ens.fr