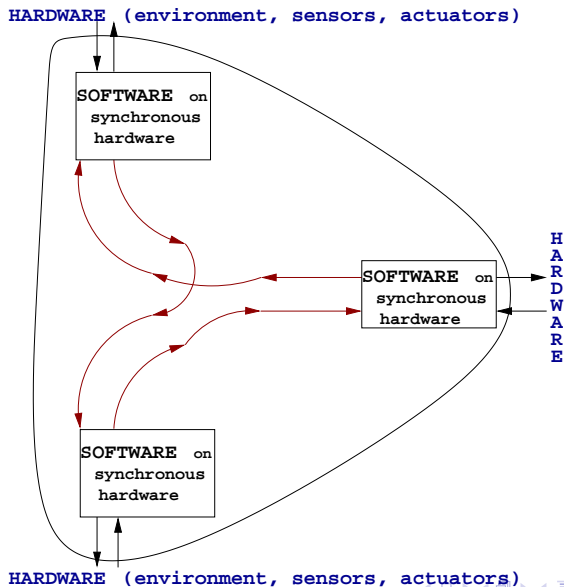


Static Analysis by Abstract Interpretation of communicating imperfectly-clocked Synchronous Programs

Julien Bertrane
bertrane@di.ens.fr

2 décembre 2006

System to analyse



Difficulties and subsequent hypotheses

Framework includes realistic executions issues :

- Clock **desynchronization** allowed
- **Non-constant delays** during communications
- **Graphical** syntax

Simplifications :

- **Quasi-synchrony** :
 - ▶ desynchronization : the cycle duration (period between two consecutive **ticks**) belongs to $[\alpha, \beta]$, $\alpha > 0$.
- Presently considered variables only **booleans**
- **blackboard** for synchronous units input
- **Serial transmission** between synchronous systems
- at initialization, all the “variables” are set to *false*

Goal : Automatic proofs of specifications

- **safety** specifications
 - ▶ **For any behaviour s , at any time t , $s(t) = true$**

Goal : Automatic proofs of specifications

- **safety** specifications
 - ▶ For any behaviour s , at any time t , $s(t) = true$
- **temporal** specifications
 - For any behaviour s , there is no t such that :

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

Goal : Automatic proofs of specifications

- **safety** specifications
 - ▶ For any behaviour s , at any time t , $s(t) = true$

- **temporal** specifications

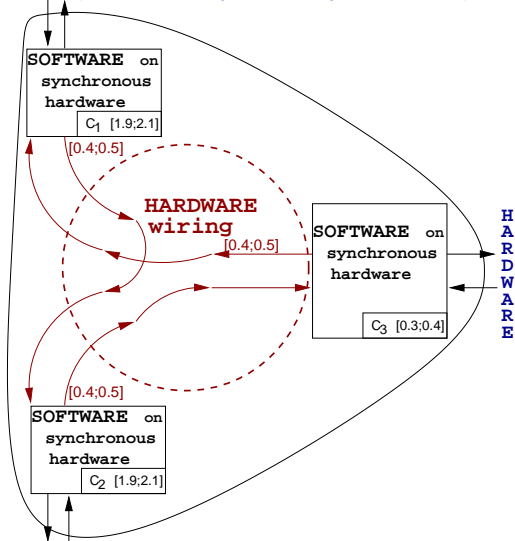
For any behaviour s , there is no t such that :

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

- **quantitative** specifications
 - ▶ the outputs of 2 redundant systems match at least half the time of any interval of width δ .

Typical system : details of hardware hypotheses

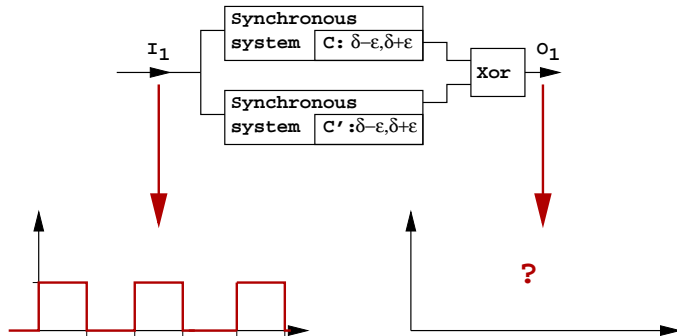
HARDWARE (environment, sensors, actuators)



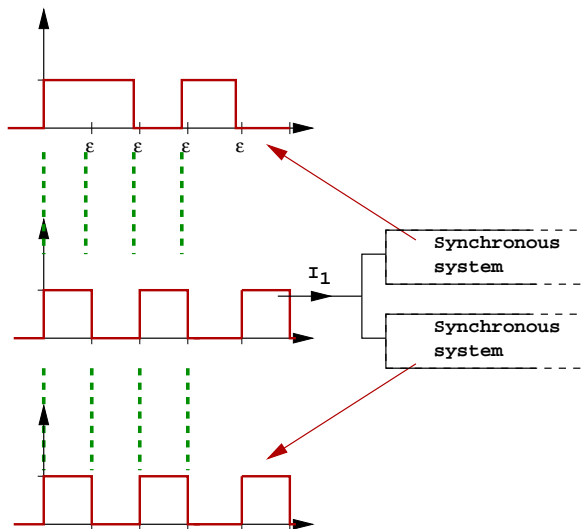
HARDWARE (environment, sensors, actuators)

Subsequent difficulties

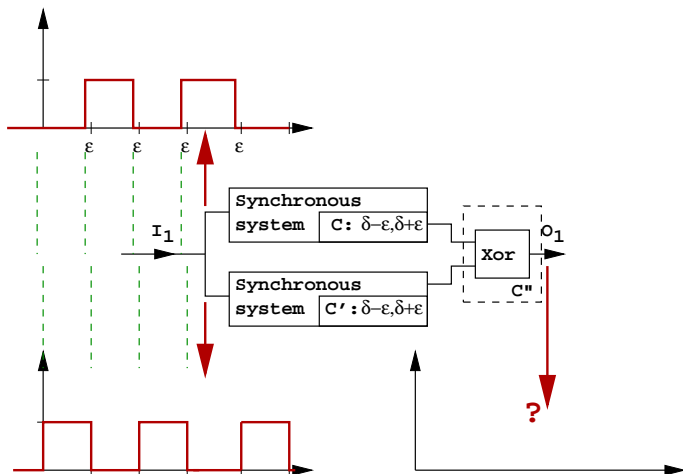
- clock skew + delays in communications \Rightarrow **non denumerable** set of behaviors



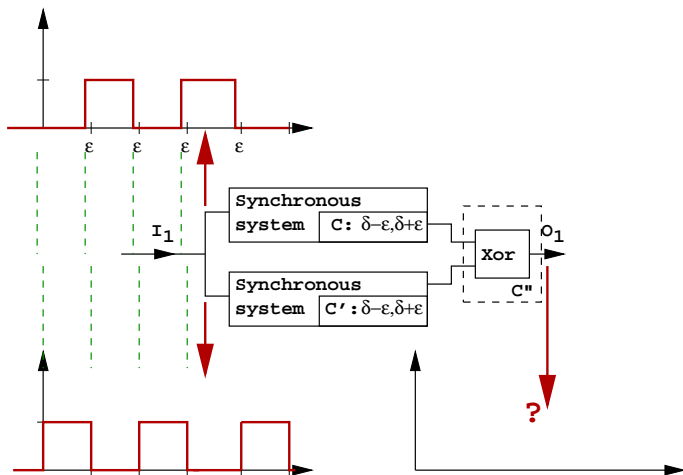
Subsequent difficulties



Subsequent difficulties

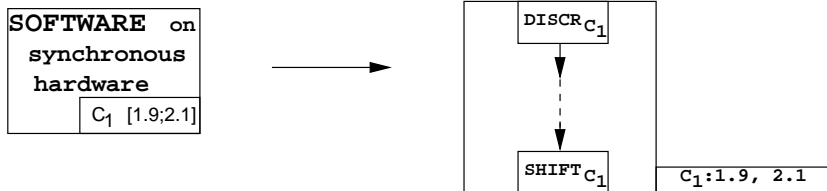


Subsequent difficulties



- Proving specifications is difficult
- This is not the right way to handle redundancy

Behavior of a synchronous system

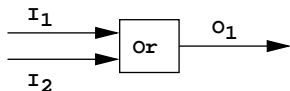


- a clock is a function $:\mathbb{N} \rightarrow \mathbb{R}^+$
- clock parameter : $[\alpha, \beta]$, with $\alpha, \beta \in \mathbb{R}^+$ and $0 < \alpha \leq \beta$
- a clock c satisfies $[\alpha, \beta]$ iff $c_{n+1} - c_n \in [\alpha, \beta]$
- $DISCR_{C_1}$ models the periodic reading of the input buffer
- $SHIFT_{C_1}$ models the waiting for the next clock tick, and the emission of its result at this next clock tick

Semantics : choices

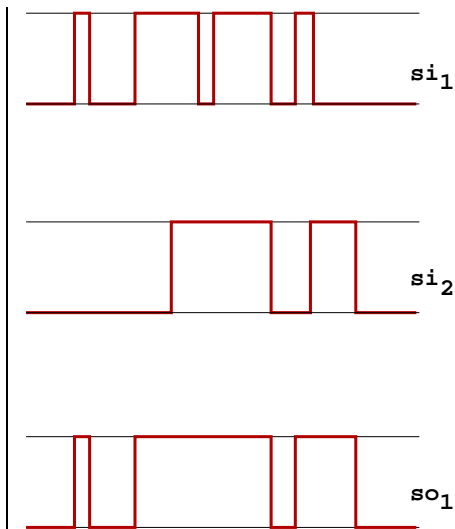
- Continuous-time semantics instead of classical discrete one (PC, Message passing,...)
- the semantics connects each point of control to a set of **signals** (i.e. element of $f : \mathbb{R}^+ \rightarrow \mathbb{B}$)
- a signal belongs to the semantics at point p if there is a vector connecting each any point but p to a signal **compatible** with p .
- if no-empty, the semantics often contains a non-countable infinity of signals

Semantics of time-independent operators

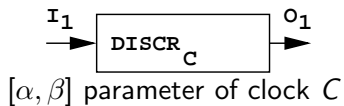


$$so_1(t) = \begin{cases} \bullet \text{ true} \\ \text{if } si_1(t) = \text{true} \\ \text{or } si_2(t) = \text{true} \\ \bullet \text{ false else} \end{cases}$$

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

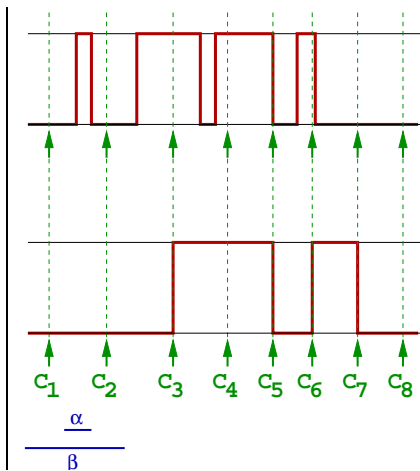


Semantics of time-dependent operators

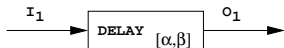
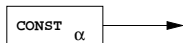
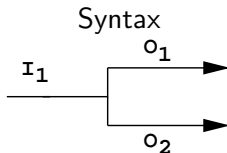


$$so_1(t) = \begin{cases} \bullet \text{ false} & \text{if } t < c(0) \\ \bullet si_1(c_n) & \text{if } t \in [c_n, c_{n+1}) \end{cases}$$

$$so_1 \triangleq \Psi_{DISCR_c}(si_1)$$



Syntax and semantics



Semantics

$$\forall t \in \mathbb{R}^+, O_1(t) = I_1(t)$$

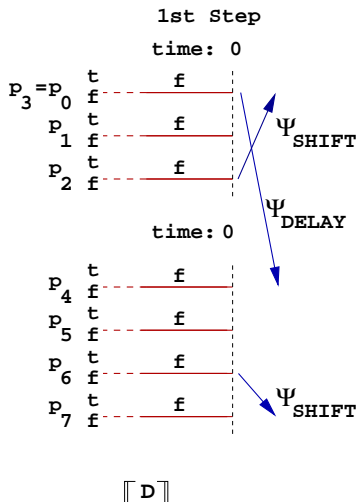
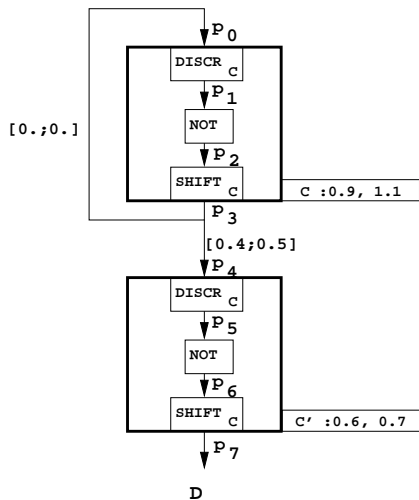
$$\forall t \in \mathbb{R}^+, O_2(t) = I_1(t)$$

$$\forall t \in \mathbb{R}^+, O_1(t) = \alpha$$

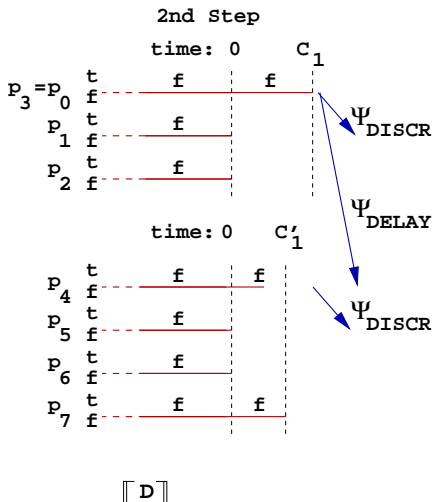
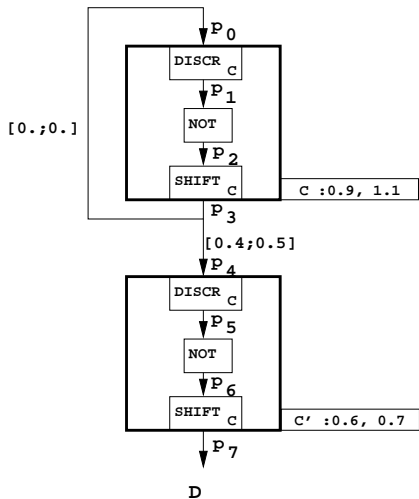
$$\forall t \in \mathbb{R}, O_1(t) = I_1(\delta(t))$$

$$\delta : \begin{cases} \exists \delta : \mathbb{R} \rightarrow \mathbb{R}, \text{monotonic,} \\ \forall t \in \mathbb{R}, \delta(t) - t \in [\alpha, \beta] \end{cases}$$

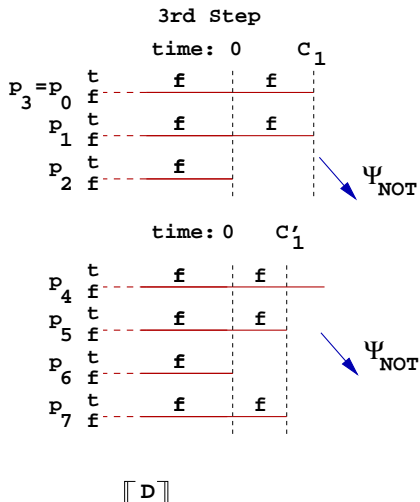
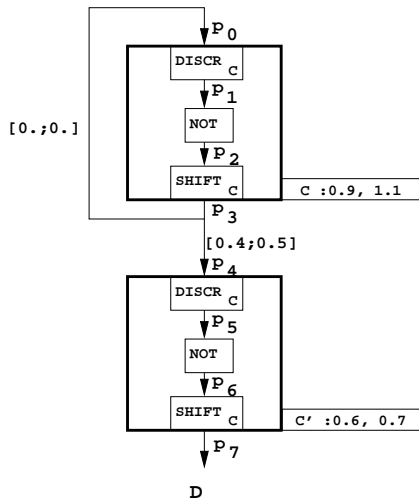
Syntaxe et sémantique



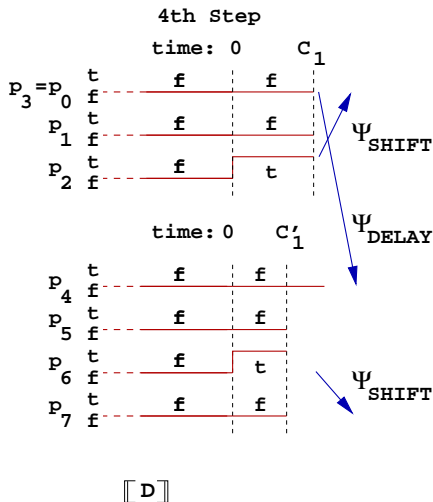
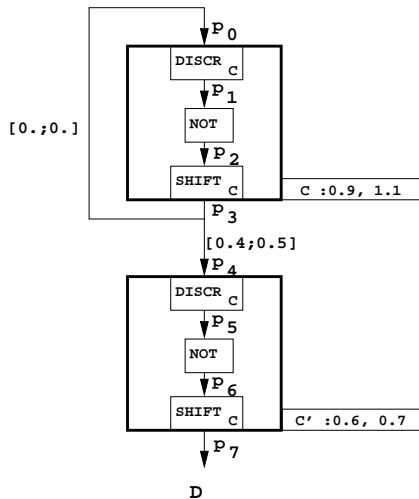
Syntaxe et sémantique



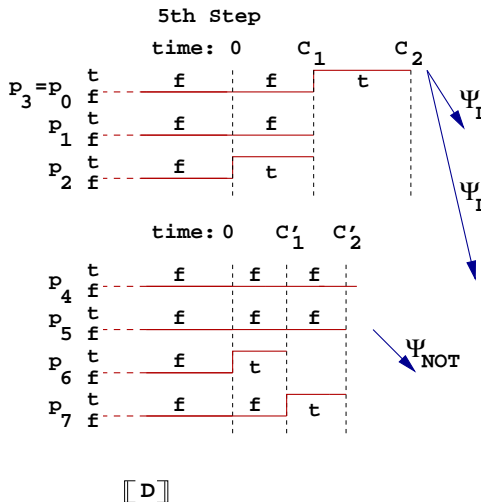
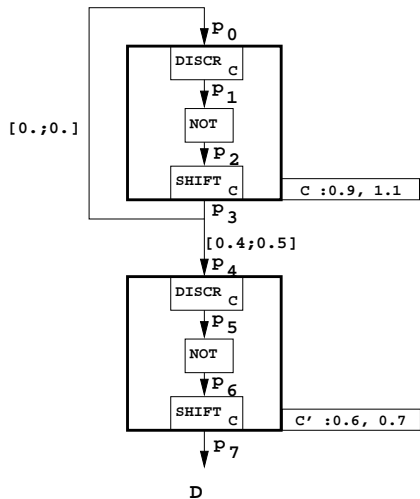
Syntaxe et sémantique



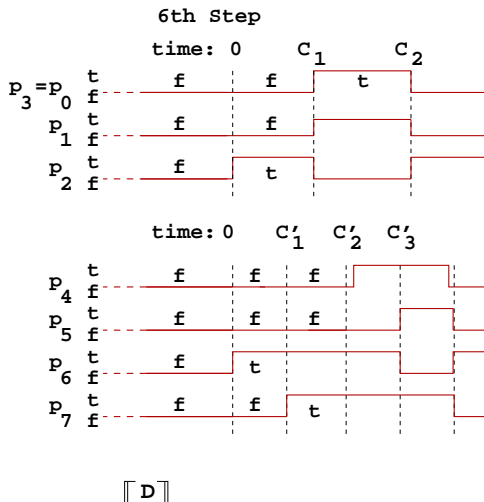
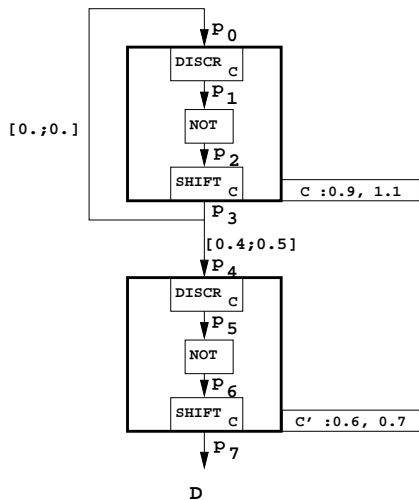
Syntaxe et sémantique



Syntaxe et sémantique



Syntaxe et sémantique



- **Not design : specification proof**
- **Difficulties : tricks**
 - ▶ for robustness to desynchronization
 - ▶ for error recovery
 - ▶ for error robustness
- Idea of **separation of design people and verification people**
- **Automatically generated code** : classical **patterns difficult** to recognize
- **pattern may be simplified** because classical academic tricks assume almost nothing
- Prototype and theory based on **Abstract Interpretation**
- Not complete : even safety undecidable

Abstract interpretation

- A set of elements
- $A^\#$ set of abstract elements
- $\alpha : A \rightarrow A^\#$
- $\gamma : A^\# \rightarrow A$

Abstract interpretation

- A set of elements
 - $A^\#$ set of abstract elements
 - $\alpha : A \rightarrow A^\#$
 - $\gamma : A^\# \rightarrow A$
-
- $A = \mathbb{Z}$
 - $A^\# = \mathbb{Z}/9\mathbb{Z}$
 - $\alpha : x \mapsto x \bmod 9$
 - $\gamma : y \mapsto \{x, x = y \bmod 9\}$

Abstract interpretation

- A set of elements
- $A^\#$ set of abstract elements
- $\alpha : A \rightarrow A^\#$
- $\gamma : A^\# \rightarrow A$

- $A = \mathbb{Z}$
- $A^\# = \mathbb{Z}/9\mathbb{Z}$
- $\alpha : x \mapsto x \bmod 9$
- $\gamma : y \mapsto \{x, x = y \bmod 9\}$

- $+^\#(4 \bmod 9, 6 \bmod 9) = 1 \bmod 9$
- **if** $\Psi \circ \gamma \subseteq \gamma \circ \Psi^\#$
- $\text{gfp } \Psi \subseteq \gamma(\text{gfp } \Psi^\#)$

Abstract interpretation based analysis

- $\llbracket D \rrbracket$ is the semantics of a set D of systems.
- $[P]$ is the set of behaviors satisfying a property P .
- **Former goal** : Prove that $\llbracket D \rrbracket \subseteq [P]$.
- **Now** : $(\Psi \cap Id)(\llbracket D \rrbracket \cap [\neg P]) \subseteq \llbracket D \rrbracket \cap [\neg P]$

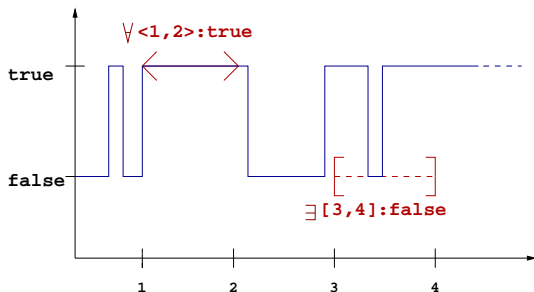
- **Thus** :

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

- **True if (not iff)** :

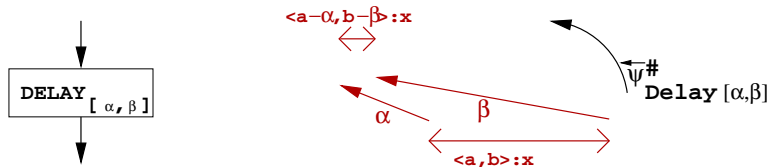
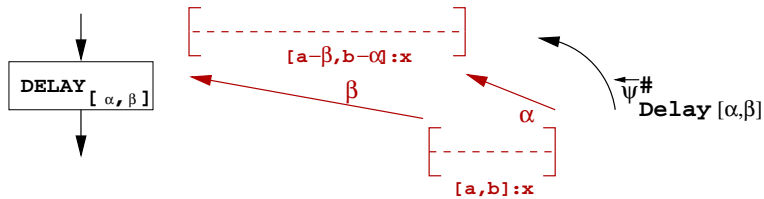
$$\text{gfp}_{[\neg P]}^{\#}(\Psi \cap Id) \subseteq^{\#} \emptyset^{\#} = \perp$$

1st abstract domain



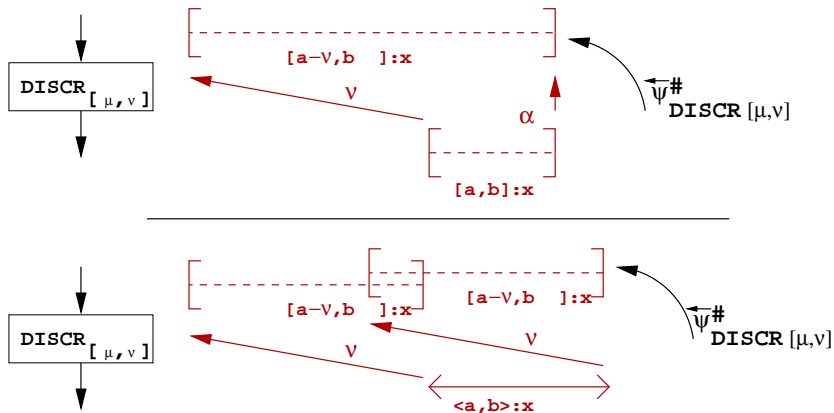
- A constraint $\exists [a; b] : x$ forces signals to be equal to x **at least once** during $[a; b]$.
- A constraint $\forall \langle a; b \rangle : x$ forces signals to be equal to x **during the whole** $[a; b]$.

Abstract Operators and Constraints : an example



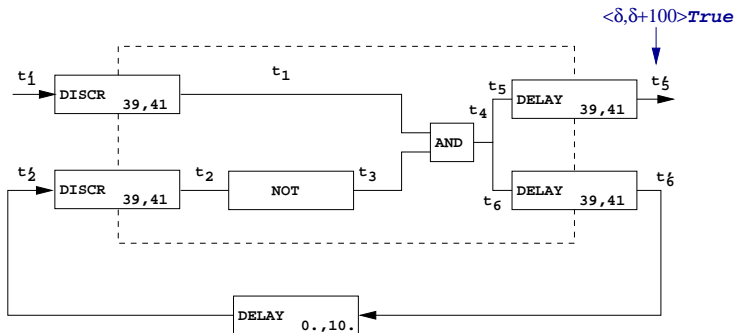
- $\overleftarrow{\Psi}^\#_{\text{DELAY}[\alpha, \beta]} (\exists [a; b] : x) \triangleq \exists [a - \beta; b - \alpha] : x$
- $\overleftarrow{\Psi}^\#_{\text{DELAY}[\alpha, \beta]} (\forall \langle a; b \rangle : x) \triangleq \forall \langle a - \alpha; b - \beta \rangle : x$

Abstract Operators and Constraints : an example

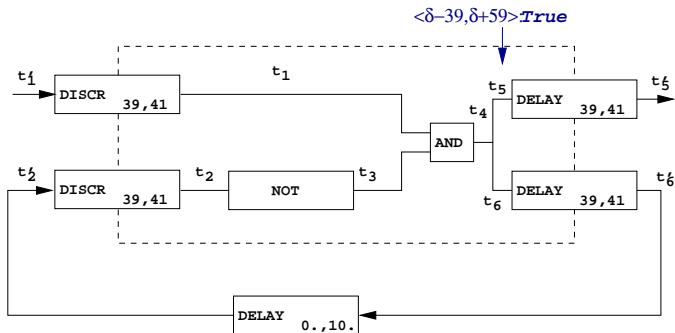


- $\overleftarrow{\Psi}^{\#}_{\text{DISCR}[\mu, \nu]}(\exists[a; b] : x) \triangleq \exists[a - \nu; b] : x$
- $\overleftarrow{\Psi}^{\#}_{\text{DISCR}[\mu, \nu]}(\forall \langle a; b \rangle : x) \triangleq \bigwedge_{t \in [a, b]} \exists[t - \nu; t] : x$

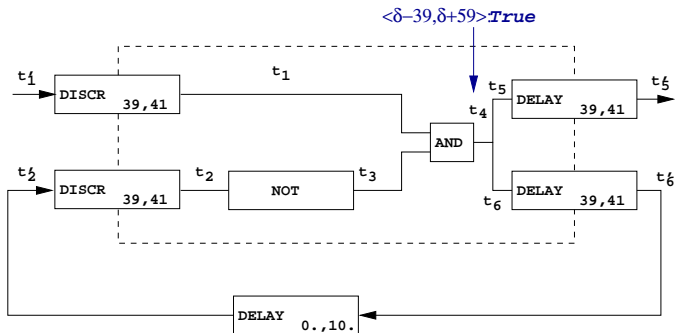
Iterating up to a fixpoint



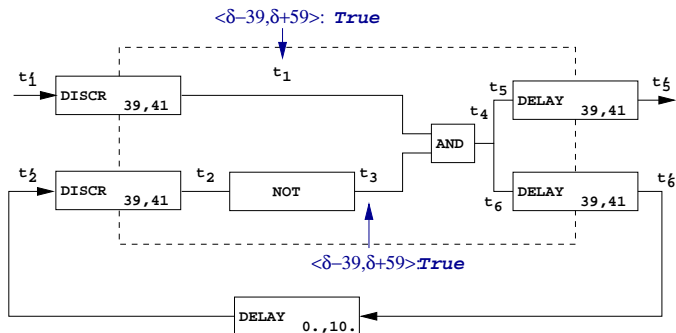
Iterating up to a fixpoint



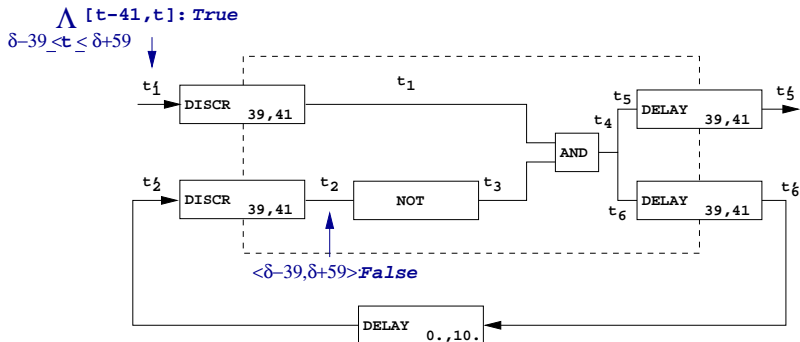
Iterating up to a fixpoint



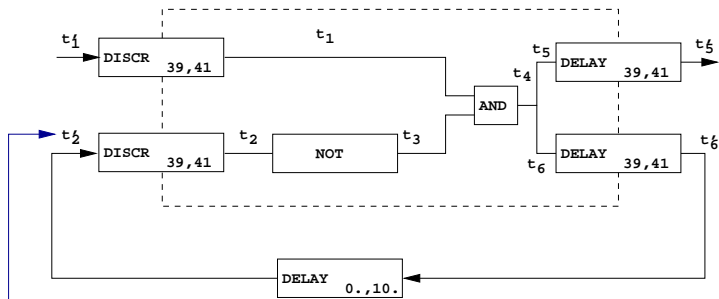
Iterating up to a fixpoint



Iterating up to a fixpoint

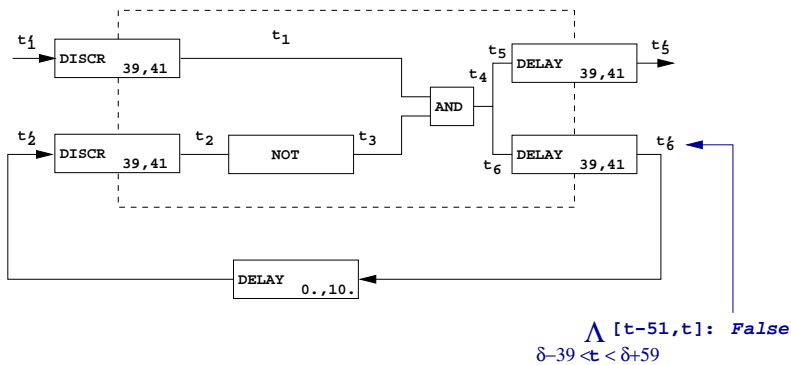


Iterating up to a fixpoint

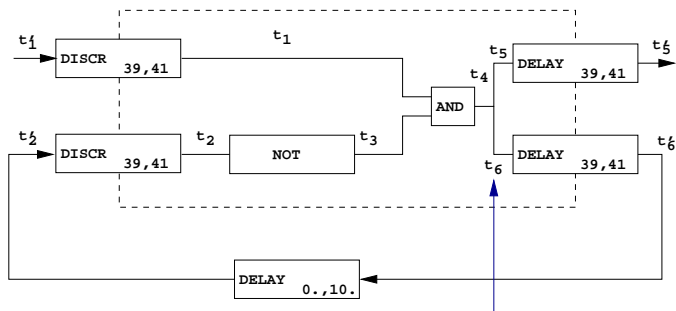


$\bigwedge [t-41, t]: \text{False}$
 $\delta-39 \leq t \leq \delta+59$

Iterating up to a fixpoint

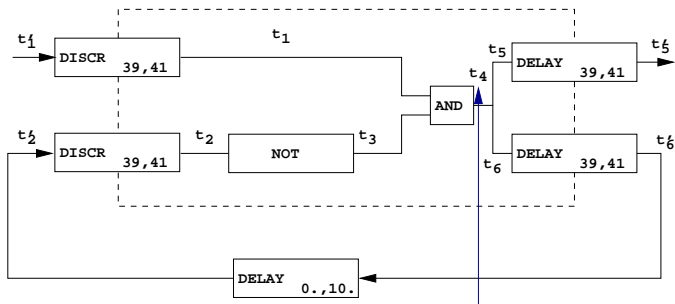


Iterating up to a fixpoint



$\bigwedge [t-92, t-39]: \text{False}$
 $\delta-39 \leq t \leq \delta+59$

Iterating up to a fixpoint



$\bigwedge [t-92, t-39]: \text{False}$
 $\delta-39 \leq t \leq \delta+59$

Example : Result of the analysis

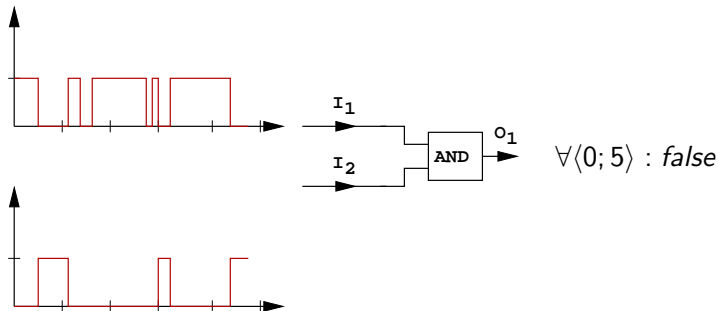
Hence no behaviour can satisfy at control point t_4 :

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

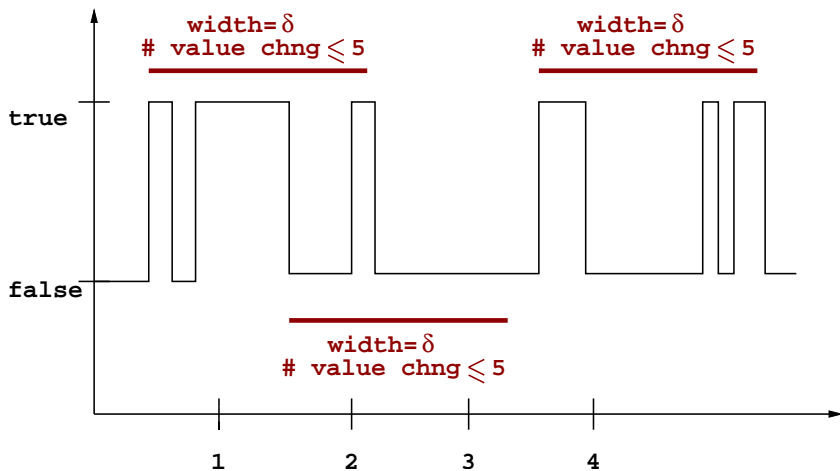
because it implies $[\delta - 33, \delta + 20] : \textit{False}$

Weaknesses of the Constraints domain

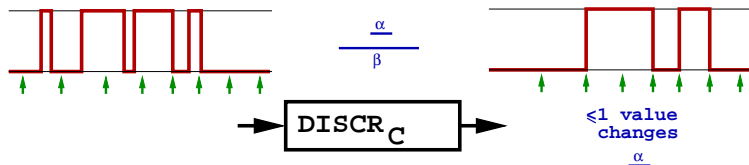
- Weak loss of precision in the case of : DELAY, DISCR, SHIFT, NOT,
- Unwished loss of precision in the case of : AND, OR, XOR



2nd Abstract Dom. : Changes Counting Dom.



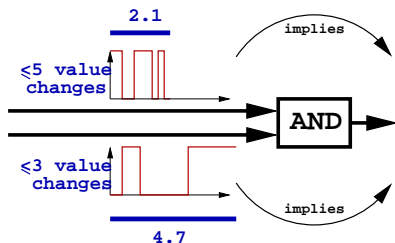
Time-dependent Abstract Operators inside the Changes Counting Domain



- $[\alpha, \beta]$ parameter of clock C

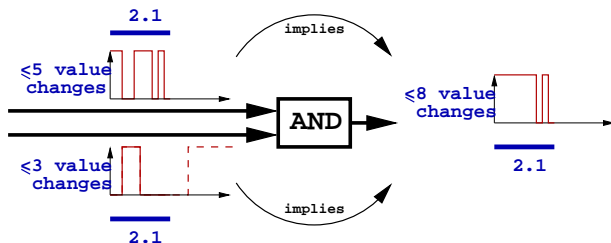
- $\vec{\Psi}_{\text{DISCR}_{[\alpha, \beta]}}^{\#}(-) \triangleq (1, \alpha)$

Time-independent Abstract Operators inside the Changes Counting Domain



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

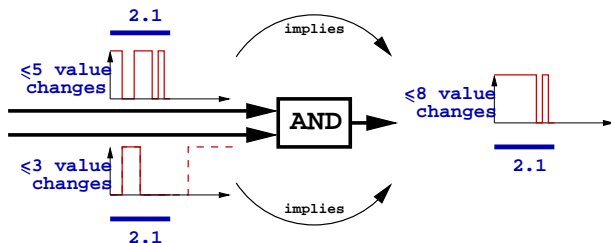
Time-independent Abstract Operators inside the Changes Counting Domain



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

- φ is a reframing function and

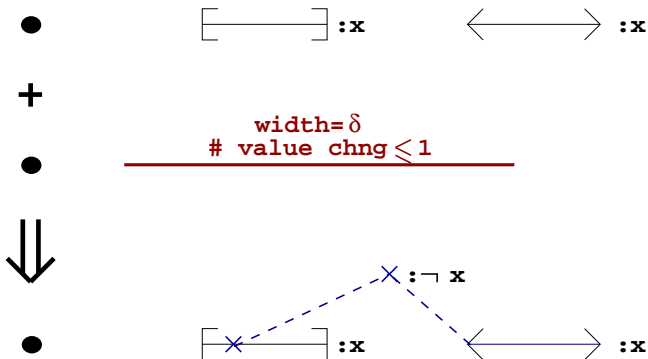
Time-independent Abstract Operators inside the Changes Counting Domain



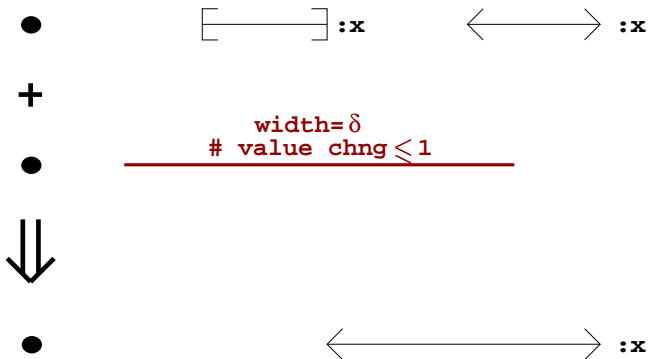
- $\vec{\Psi}_{\text{AND}}^{\#}((n_1, \delta_1), (n_2, \delta_2)) \triangleq (\tilde{n}_1 + \tilde{n}_2, \tilde{\delta}_1)$

- ▶ φ is a reframing function and
- ▶ $\varphi((n_1, \delta_1), (n_2, \delta_2)) = ((\tilde{n}_1, \tilde{\delta}_1), (\tilde{n}_2, \tilde{\delta}_1))$.

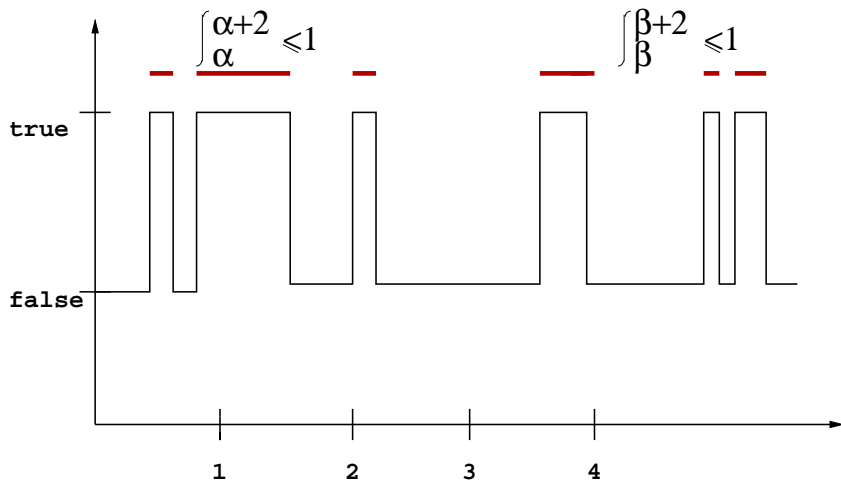
Reduced Product Constraints-Changes Counting Domain



Reduced Product Constraints-Changes Counting Domain



3rd abstract domain : Integral bounding Dom.



- Express quantitative properties

Conclusion

- Realistic (?) model of execution of imperfectly-clocked communicating synchronous systems (imperfect clocks, non-instantaneous delays, blackboard)
- Syntax includes annotations about hardware imperfections
- Semantics mixes discrete/continuous notions, but mainly continuous-time
- Analysis retrieves discrete behaviors (Value changes counting)
- Need for a better knowledge of robustness/redundancy techniques