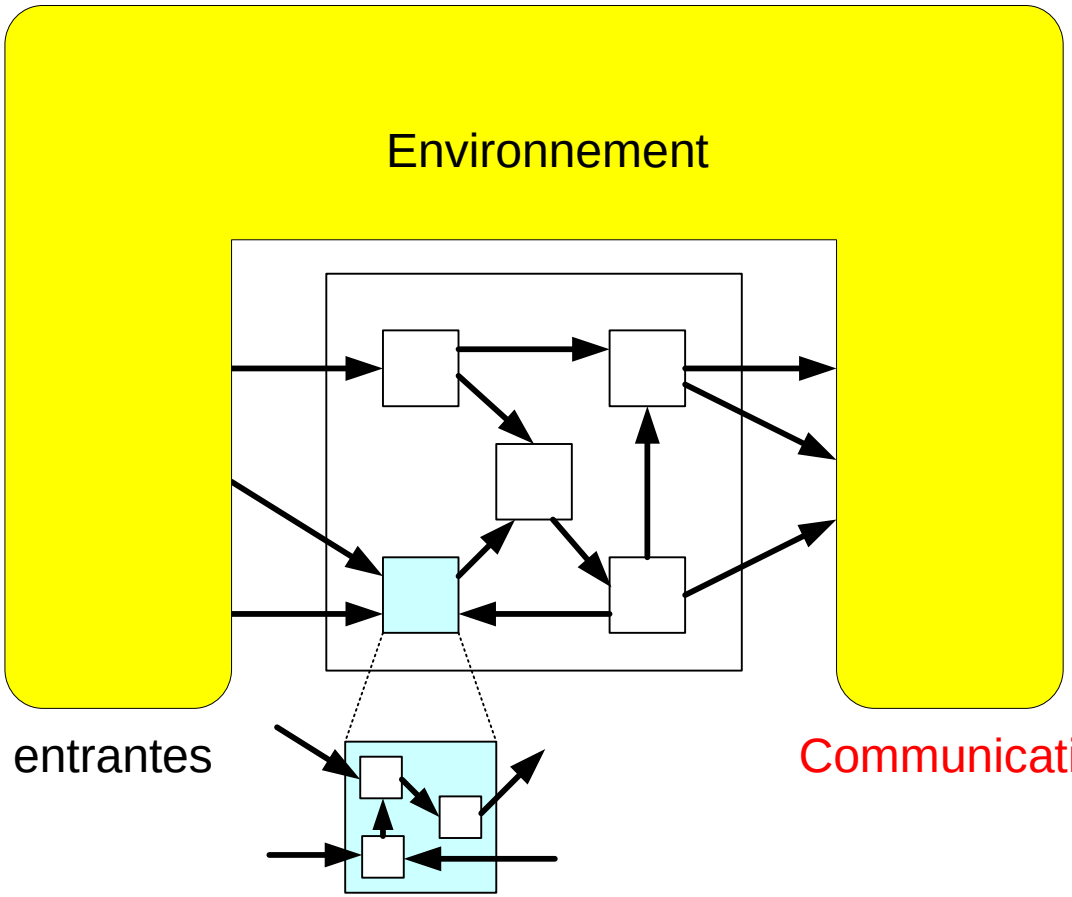
Langages Synchrones

Charles ANDRÉ
Projet AOSTE
Lab. I3S (UNSA/CNRS/INRIA)
FAC 2006 23-24 Mars 2006

Langages spécialisés

- Les langages synchrones sont des langages spécialisés (à opposer à langages à usage général).
- Spécialisés pour la programmation d'applications réactives.
- Ils sont généralement traduits en programmes dans un langage d'usage général (langage hôte).

Programmes réactifs



Communications entrantes

Communications sortantes

Réactions des acteurs

et communications internes

Besoins d'expression

- Communications
  - Avec l'extérieur / internes
- Évolutions simultanées
  - Indépendantes/Concurrentes/Parallèles/ Séquentielles
- Modularité / Interfaces
- Hiérarchie

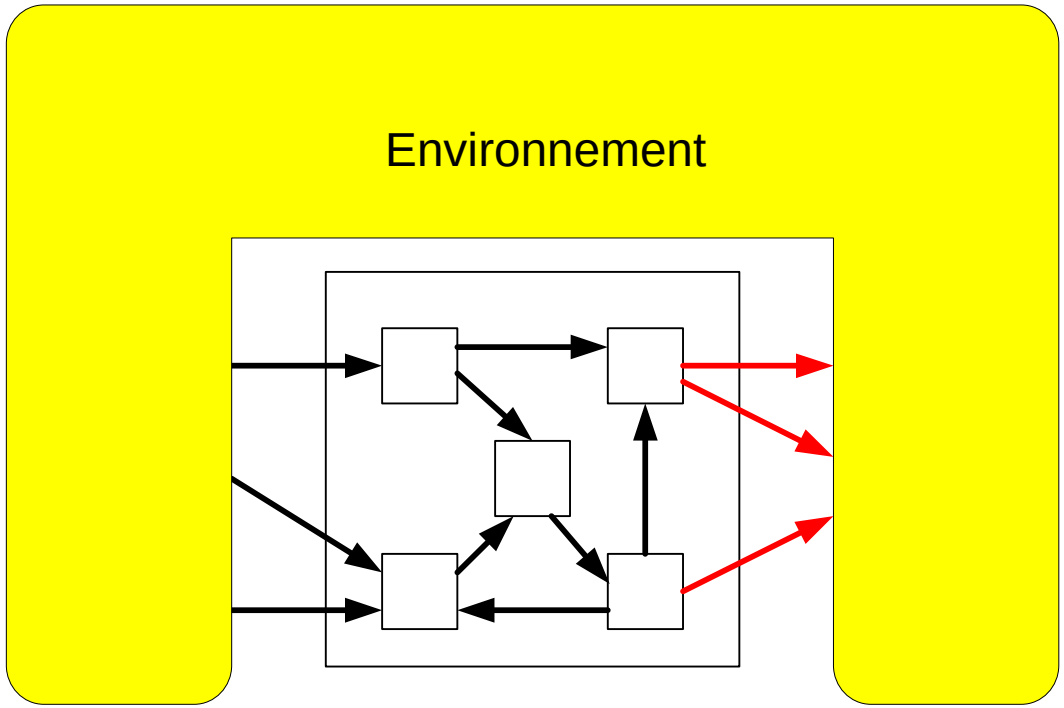
Approche synchrone

- Notion d'instant logique
  - Simultanéité
  - Priorité (réaction à l'absence)
- Signaux
  - Abstraction des communications
  - Diffusion instantanée
- Contrôle
  - Déterministe, dans l'instant
  - Préemption
- Actions instantanées possibles

Programmes réactifs synchrones (1)

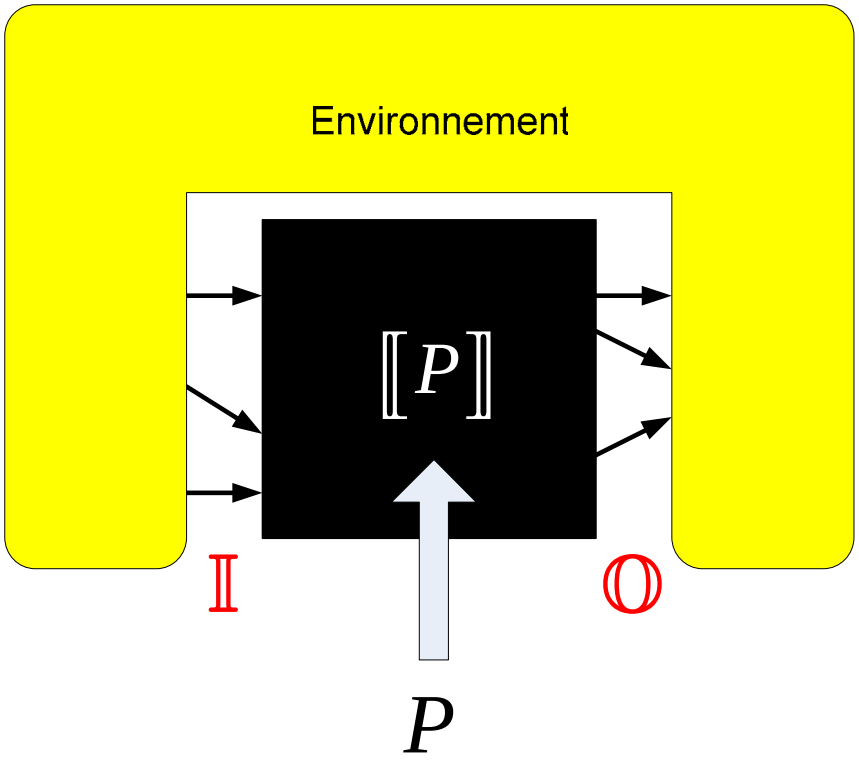
Exécution atomique:

- Lecture
- Calculs
- Ecriture



Ecriture

Programmes réactifs synchrones (2)

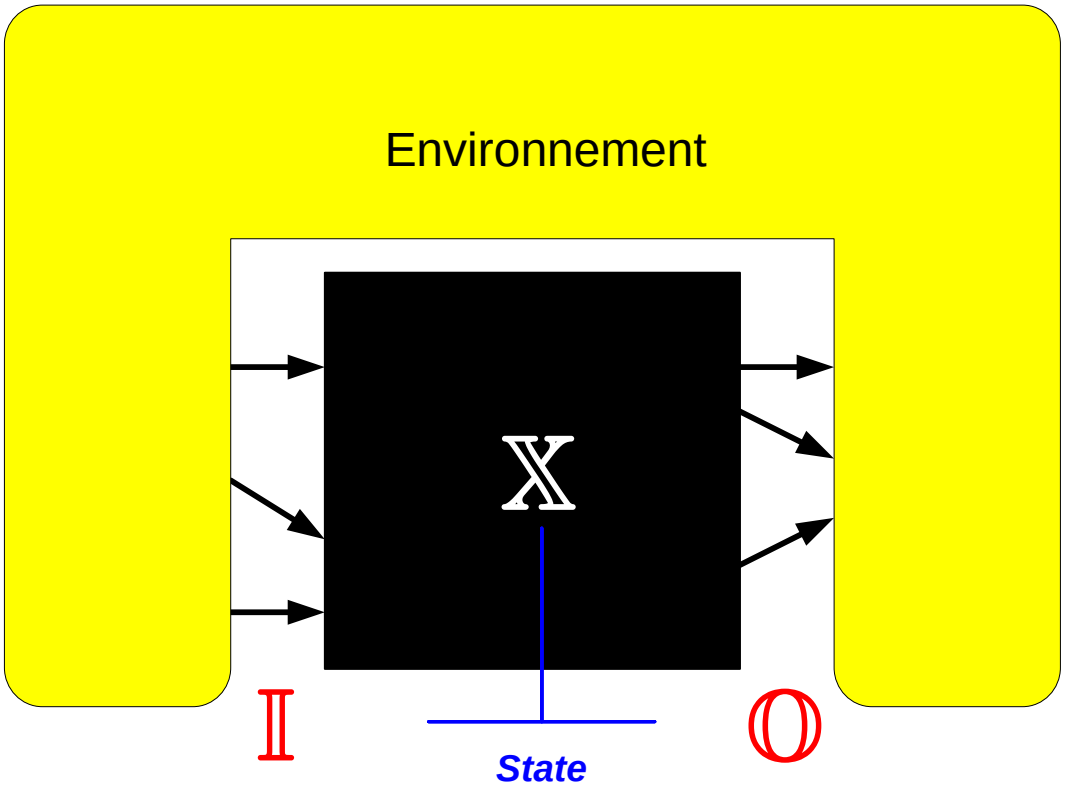


$$[\![P]\!] : \mathbb{I}^* \rightarrow \mathbb{O}^*$$

Transformation déterministe de séquences d'entrée en séquences de sortie.

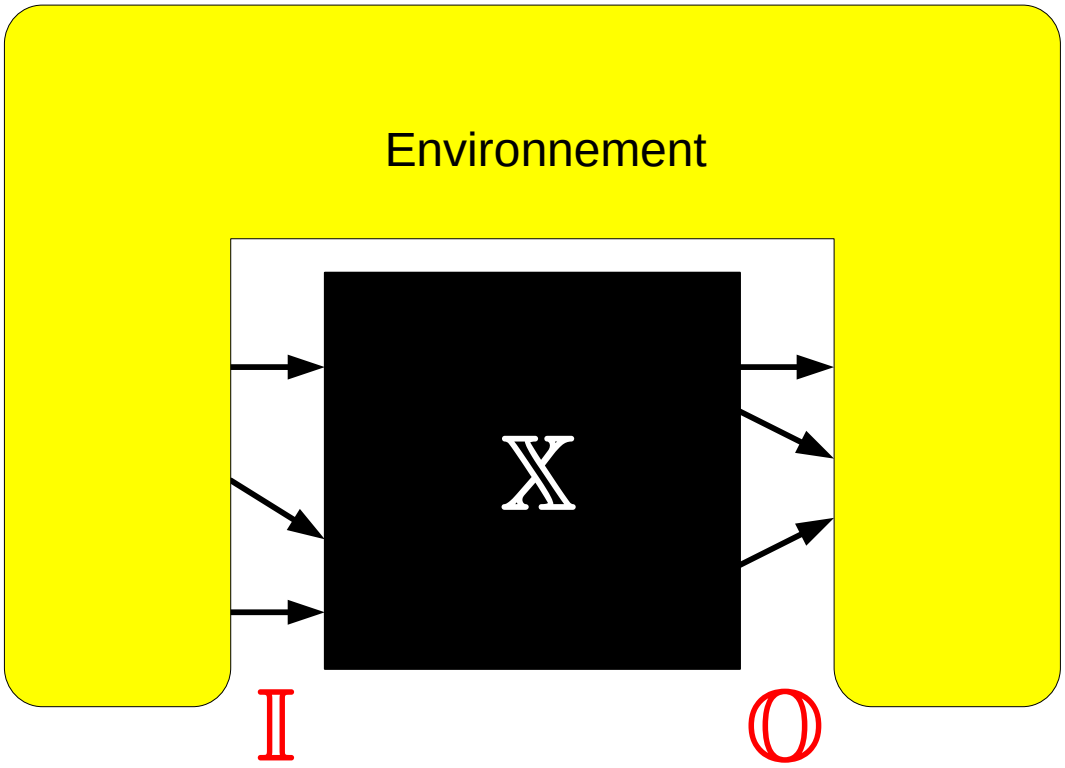
Séquences indexées par les instants logiques.

Programmes réactifs synchrones (3)



Réaction : $(\mathbb{X}, \mathbb{I}) \mapsto (\mathbb{X}', \mathbb{O})$

Programmes réactifs synchrones (3)

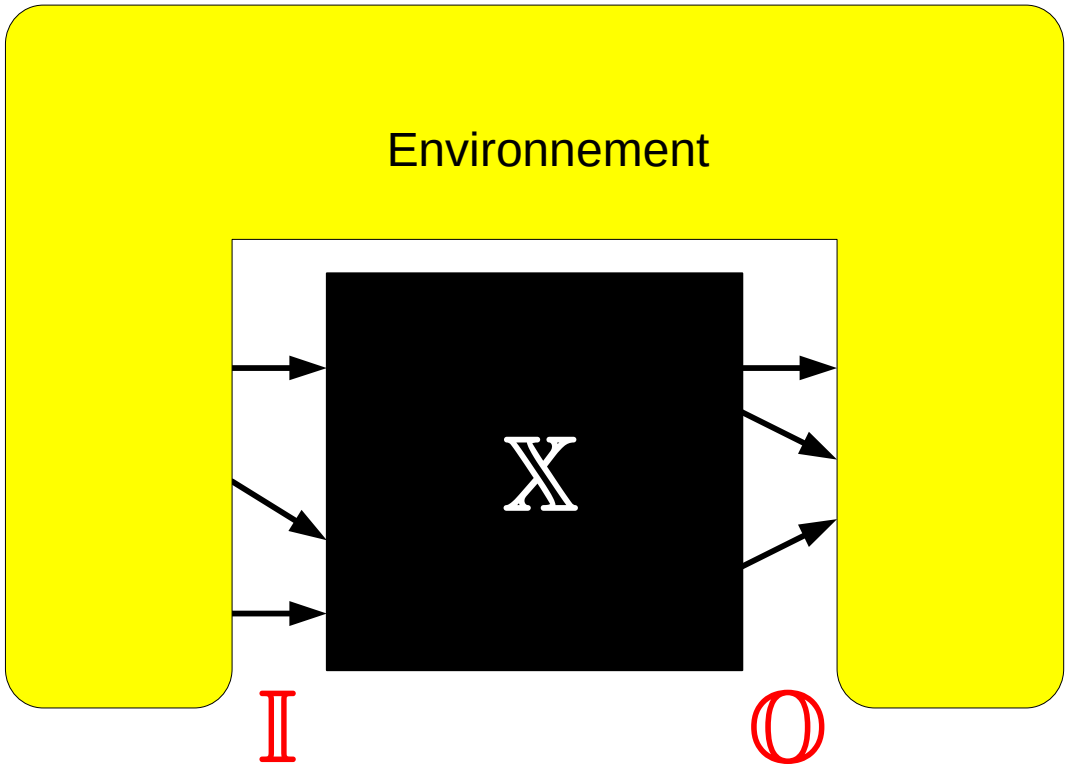


Réaction : $(\mathbb{X}, \mathbb{I}) \mapsto (\mathbb{X}', \mathbb{O})$

$\lambda : \mathcal{X} \times \mathcal{I} \to \mathcal{X}$ (fonction état suivant)

$\omega : \mathcal{X} \times \mathcal{I} \to \mathcal{O}$ (fonction de sortie)

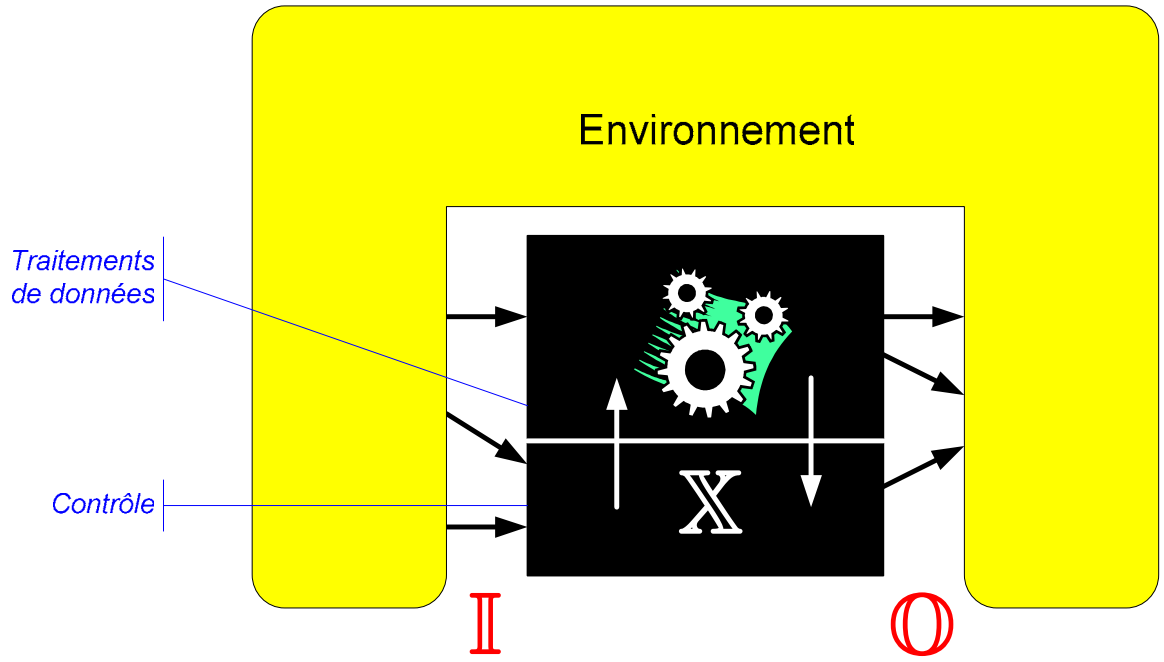
Programmes réactifs synchrones (3)



$$P \xrightarrow[\mathbb{I}]{\mathbb{O}} P' \text{ (ré-écriture)}$$

$$P = P_0 \xrightarrow[\mathbb{I}_1]{\mathbb{O}_1} P_1 \xrightarrow[\mathbb{I}_2]{\mathbb{O}_2} P_2 \cdots P_{n-1} \xrightarrow[\mathbb{I}_n]{\mathbb{O}_n} P_n \cdots$$

Programmes réactifs synchrones (4)



Réaction :

$(\mathbb{X}, \mathbb{I}) \mapsto (\mathbb{X}', \mathbb{O})$ Contrôle (états finis)

\+ Traitements de données contrôlés

$P \rightarrow$ un ordonnanceur compilé

Programmes réactifs synchrones (5)

- Cette vision est trop rigide (à chaque instant $\mathbb{I}$ considère toutes les entrées et $\mathbb{O}$ toutes les sorties).
- Les langages synchrones peuvent n'utiliser qu'une valuation partielle des ensembles $\mathcal{I}$ (en entrée) et $\mathcal{O}$ (en sortie).
- Des signaux peuvent être absents lors d'une réaction.
- A chaque instant un signal est caractérisé par une et une seule paire (présence,valeur)

D'où viennent les langages synchrones?

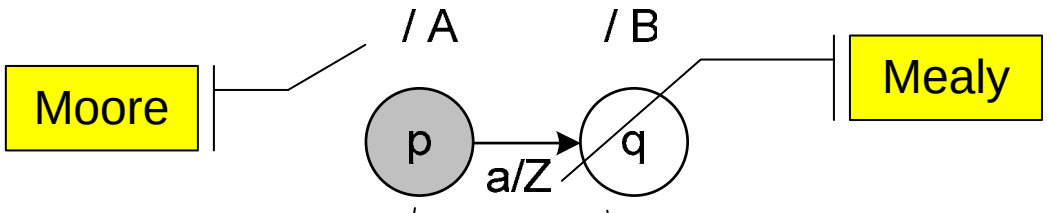
- Début des années 80: langages synchrones
  - Estérel (Sophia Antipolis)
  - Lustre (Grenoble)
  - Signal (Rennes)
- Modèles quasi synchrones
  - Grafcet
  - Statecharts
- Néo synchronisme
  - Argos / SyncCharts
  - Lucid synchrone
- Le modèle synchrone en circuits
- Proto synchronisme
  - Plus proactif que réactif: Génèse – Gn 1,1-1,31 et 2,1-2,3

Styles de programmation

	Déclaratif		Impératif
	Fonctionnel	Relationnel	
Textuel	LUSTRE Lucid-Synchrone	SIGNAL	ESTEREL
Graphique	SCADE	Format Schéma blocs	SYNC-CHARTS (SSM)

Une apparente simplicité …

Modèle Etat/Transition

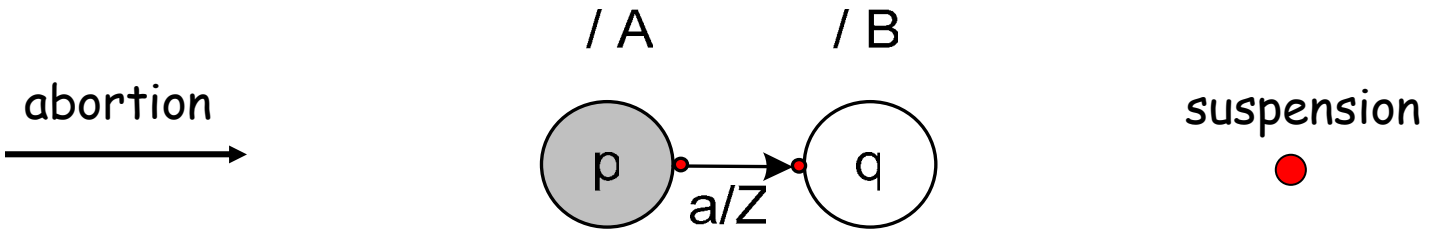



```
weak abort
      sustain A
 ||
      run p
 when a do
      emit Z
 end abort;
```

```
      sustain B
 ||
      run q
```

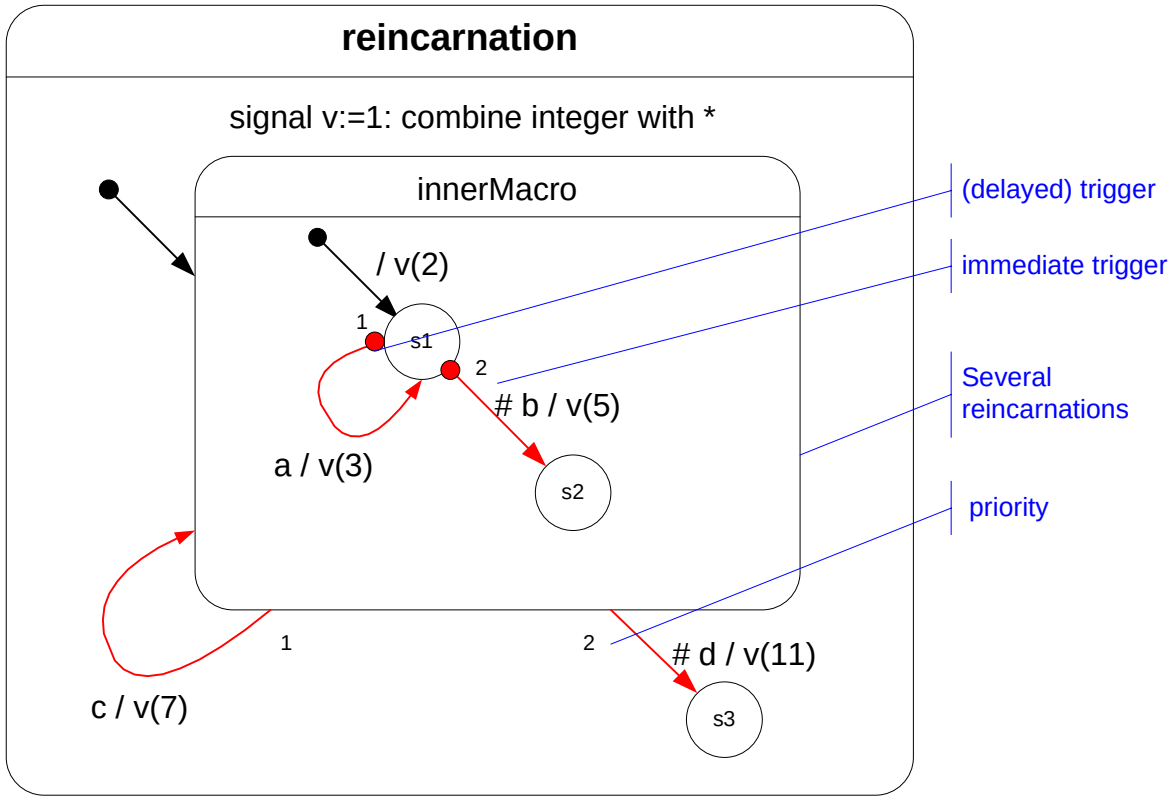
SyncCharts vers Esterel dans ce cas très très simple

Modèle Etat/Transition



instant		k-1	k	k+1
input			a	
Seq. Machine		p	p	q
		A	A,Z	B
Synch. Model		p	q	q
Weak abortion		A	A,Z,B	B
Strong abortion		A	Z,B	B
		A	A,Z	B
		A	Z	B

Puissant (trop?)



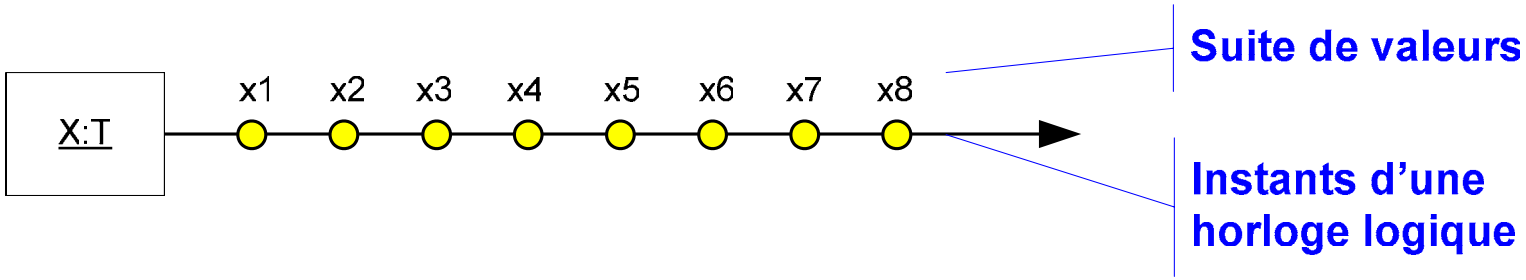
Si s1 est actif et que a,b,c,d sont présents : 6 transitions franchies simultanément, dont une 2 fois. Emission de v avec la valeur $2*3*5^2*7*11=11550$

http://www.elsevier.nl/locate/entcs/volume88.html

Ingrédients des langages synchrones

Entités de base

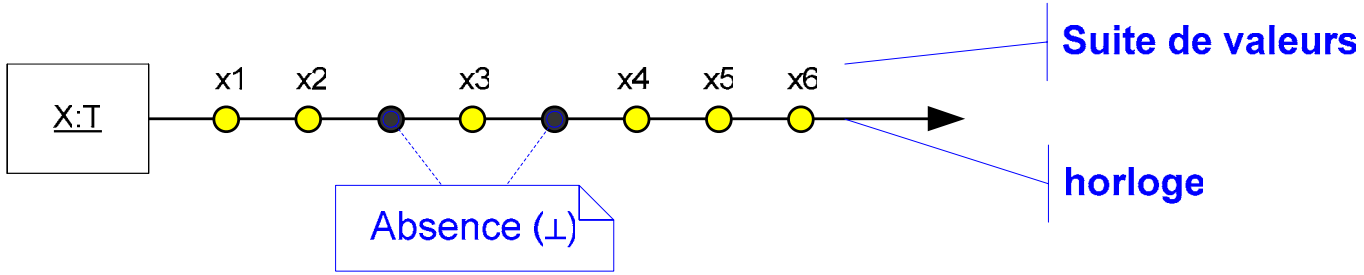
- **LUSTRE**
  - Flot = séquence infinie de valeurs d'un type donné
  - Equation = définition d'un flot par une expression portant sur des flots



X: T $\Leftrightarrow$ $(X_1, X_2, \cdots, X_n, \cdots)$ où $X_k : T$

Entités de base

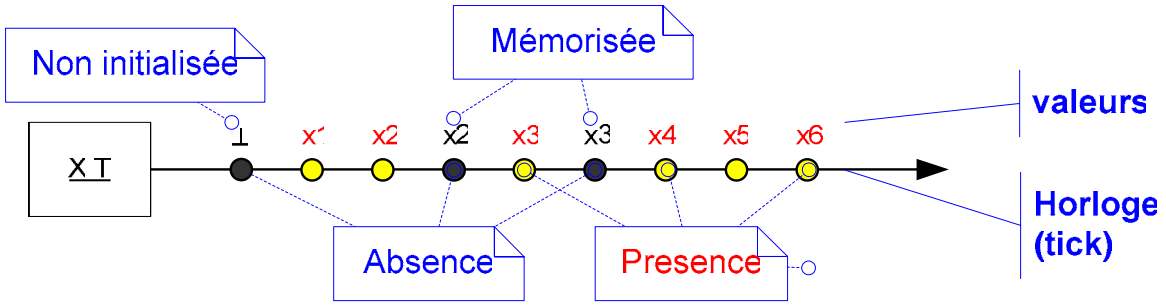
- **SIGNAL**
  - Signal = une horloge + 1 séquence de valeurs typées, avec possibilité d'absence à certains instants (⊥)
  - Relations imposées sur des signaux.



integer X ⇔ $X = (X_t)_{t \in C_X}$

Entités de base

- **ESTEREL**
  - Signal = abstraction des communications. Paire (présence, valeur) à un instant global
  - Threads d'exécution, hiérarchiques et concurrents, définis par des instructions impératives/blocs de contrôle.



X: integer ⇔ à l'instant k, (X.status, X.value:integer)

Unités de programmation

- **LUSTRE**
  - node = fonction sur les flots
  - interface = paramètres formels (in ou out)

```
node COMPTEUR_MOD (RAZ:bool;MOD:int)
        returns (C:int);
```

Unités de programmation

- **SIGNAL**
  - process = ensemble de signaux + relations
  - interface = paramètres formels (paramètres, in,out)

```
process COMPTEUR_MOD =
            { integer N } % parameters %
            (  ? event RAZ
               ! integer CPT )
            (| .... |)
end
```

Unités de programmation

- **ESTEREL**
  - module = header + corps impératif
  - interface = paramètres formels (in,out,inout)
  - data = déclarations de types, constantes, …

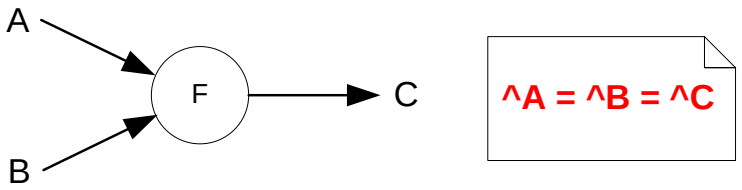
  Généricité

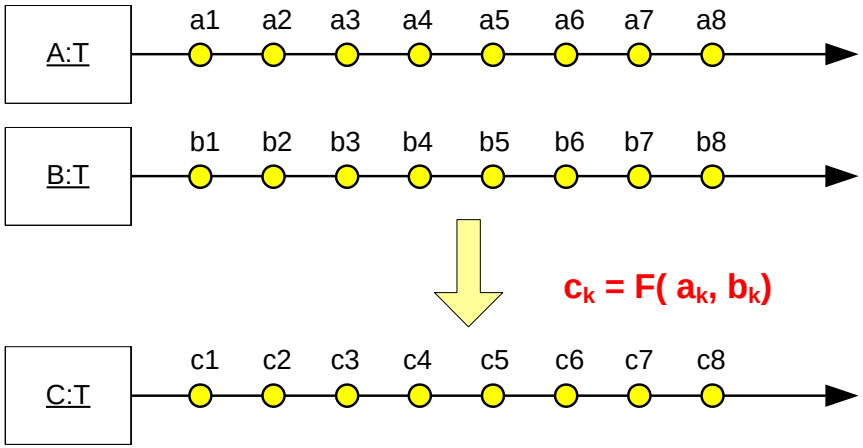
```
data ComptData:
        constant N: unsigned = 8;
end data
interface ComptIntf:
        extends data ComptData;
        output C: unsigned<N>;
end interface
module COMPTEUR :
        extends interface ComptIntf;
        // imperative part
end module
```

Calcul d'horloge

Calcul d'horloge

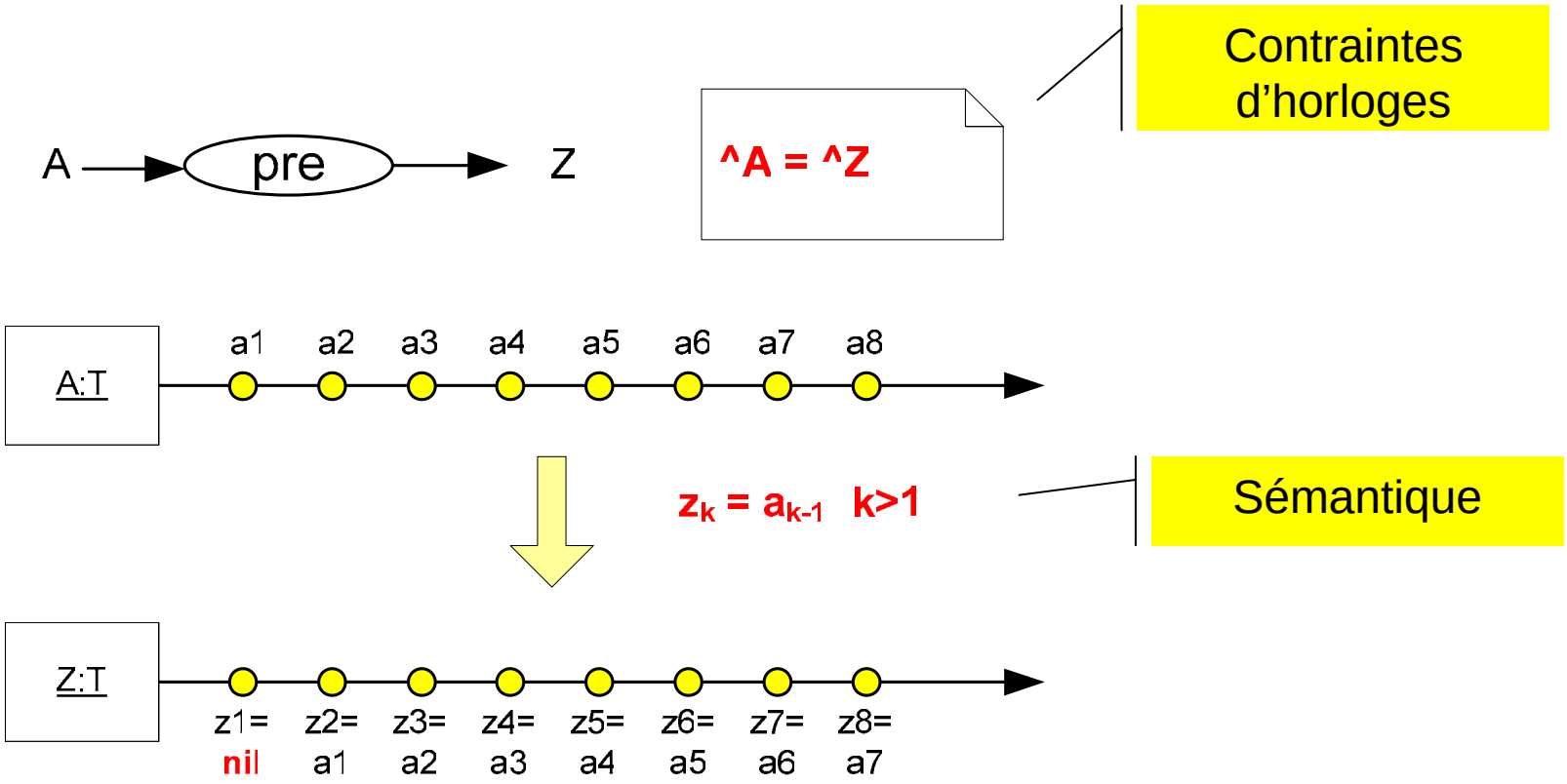
- Spécifique au synchrone
  - Les signaux d'entrée sont imposés
  - Pour les signaux locaux et de sortie il faut déterminer
    - leur valeur
    - et les instants de présence ⇒ calcul d'horloge
  - Règle de base: F(X,Y) ⇒ même horloge pour X,Y et pour le résultat
  - Règles spécifiques (opérateurs temporels)



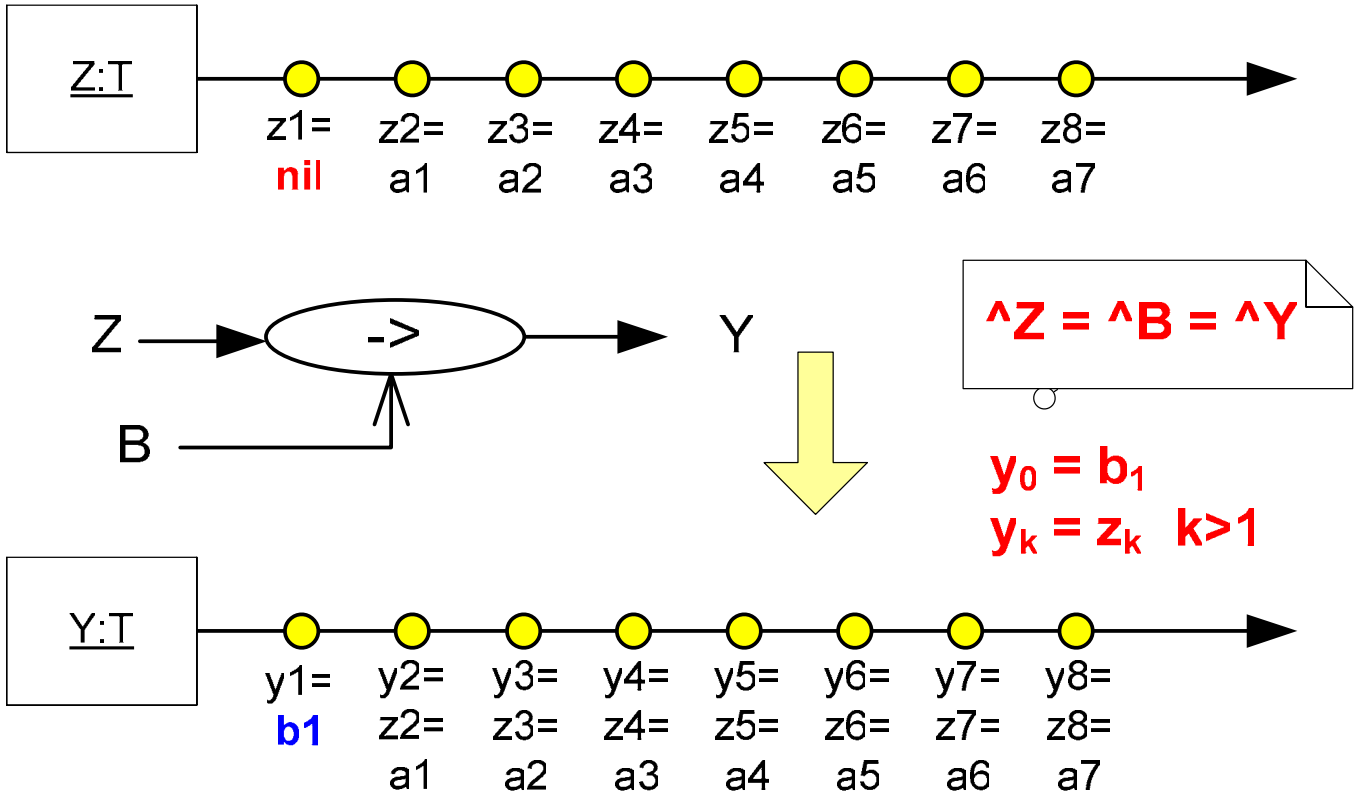


Quelques opérations "réactives"

Illustration au travers du langage LUSTRE

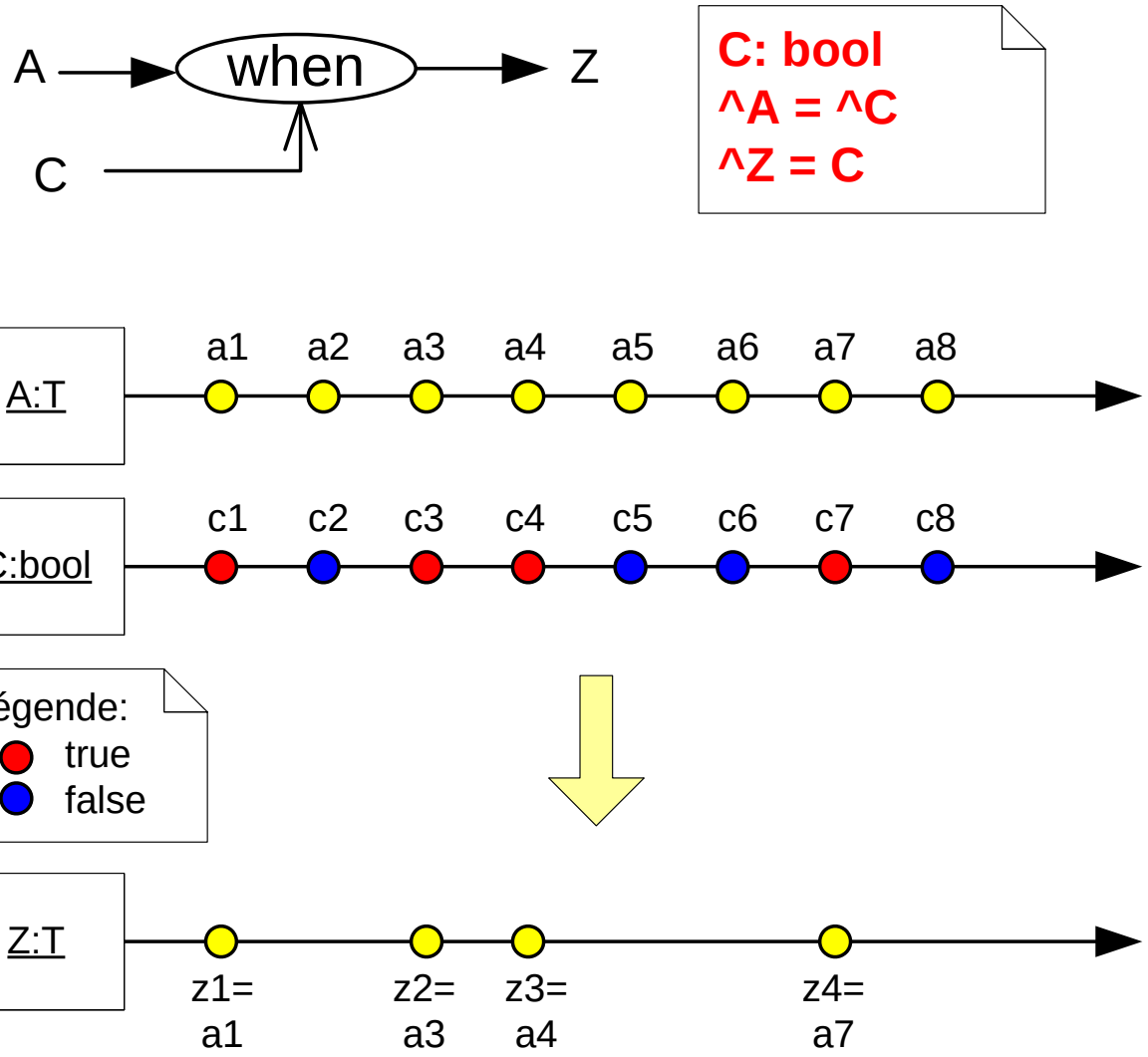


Quelques opérations "réactives"

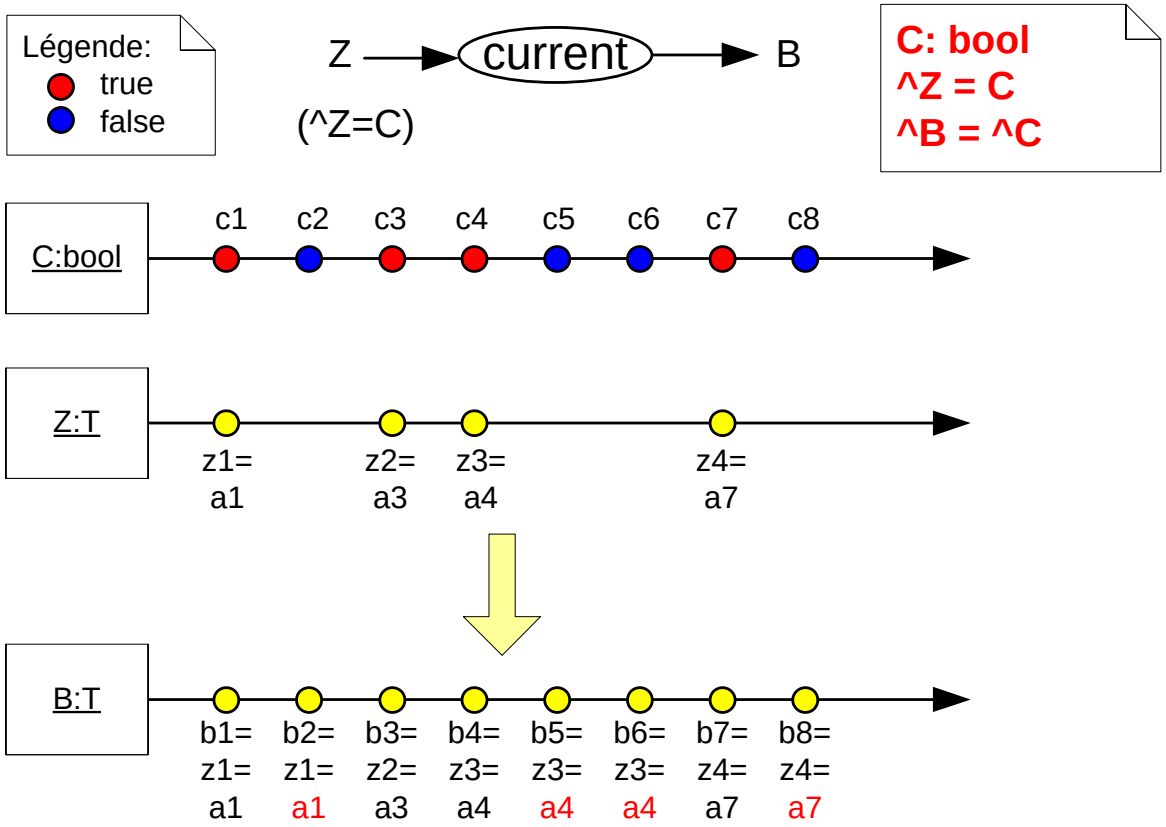


Lucid synchrone & SCADE : **fby = -> pre**

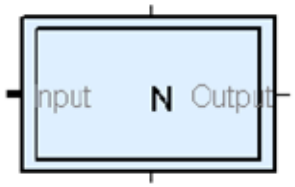
Quelques opérations "réactives"



Quelques opérations "réactives"



SCADE: **condact**



n inputs, m outputs, initial values,
Boolean activation condition
N: any operator

Calcul d'horloge en Lustre

- En LUSTRE: très simple – mono-horloge
  - horloge de base
  - hiérarchie (when) ⇒ arbre d'horloges
- Règles:
  - Les entrées sont sur l'horloge de base.
  - Les constantes sont sur l'horloge de base.
  - Pour tout opérateur op, l'expression (X op Y) a pour horloge H si X *et* Y ont pour horloge H.
  - (X **when** H) est correct si X et H sont sur la même horloge. L'horloge de (X **when** H) est alors H.
  - L'horloge de (**current** X) est l'horloge de l'horloge de X.

Calcul d'horloge en Signal

- En SIGNAL: plus complexe – multi-horloge
  - Horloge comme Type
  - Partiellement ordonnées
  - Créations de nouvelles horloges

    Z := X default Y ⇒ union d'horloge

    Synchronisations et opérations sur horloges

Opérateurs mono-horloge

Z := X **op** Y	$Z_k = X_k$ op Y_k	^Z = ^X = ^Y	X → Z & Y → Z
Z := X**$**1	$Z_k = X_{k-1}$	^Z = ^X	

Opérateurs multi-horloges

Z := X **when** B	$Z_\tau = X_\tau$ when B_τ = true = ⊥ otherwise	^Z = ^X when B	B → Z & X → Z when B
Z := X **default** Y	$Z_\tau = X_\tau$ when $X_\tau \neq \perp$ = Y_τ otherwise	^Z = ^X ∪ ^Y	X → Z & Y → Z when (^Y\^X)

Syntaxe

Sémantique

Contraintes d'horloges

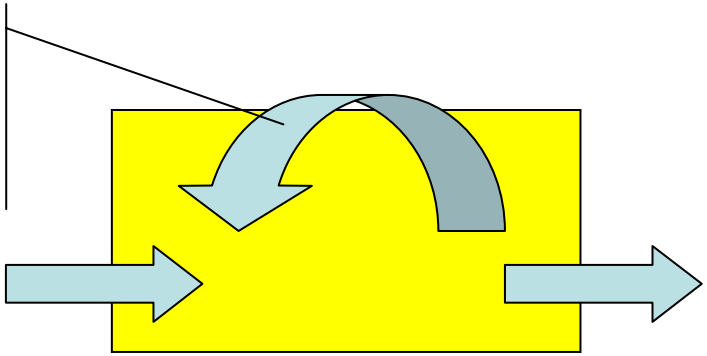
Dépendances causales

Calcul d'horloge en Lucid Synchrone

- Langage fonctionnel : Lustre ++
- Flot de données synchrone + ordre supérieur
- Système de type polymorphique permettant d'inférer automatiquement les types
- Les horloges sont considérées comme des types
- Le calcul d'horloge permet de vérifier la correction temporelle du programme

Causalité

Possibilité de "feedback" instantané



$$\mathbb{X}' = \lambda(\mathbb{X}, \mathbb{I}, \mathbb{O})$$

$$\mathbb{O} = \omega(\mathbb{X}, \mathbb{I}, \mathbb{O})$$

- Cycles
  - LUSTRE: interdits (il faut dans pre dans les rebouclages)
  - SIGNAL: calcul d'horloge
  - ESTEREL:
    - Boucles instantanées: rejets
    - Cycles statiques (syntaxiques)
    - Cycles dynamiques (exploration de l'espace d'états en V5)

Causalité (2)

- Problème délicat en ESTEREL
- Absence de solution (non réactif)
  - On ne peut pas déterminer l'état d'un signal
  - Présence: present S else emit S end
  - Valeur: emit S(?S+1)
- Solutions multiples (non déterministe)
  - Plusieurs solutions sont logiquement acceptables
  - Présence: present S then emit S end
  - Value: emit S(?S)
- Causalité constructive (non respect de la causalité)
  - La cohérence logique ne suffit pas.
  - Si Non respect de la sémantique de la séquence ⇒ rejet
  - Construire la preuve de la présence ou de l'absence d'un signal.

Évolutions

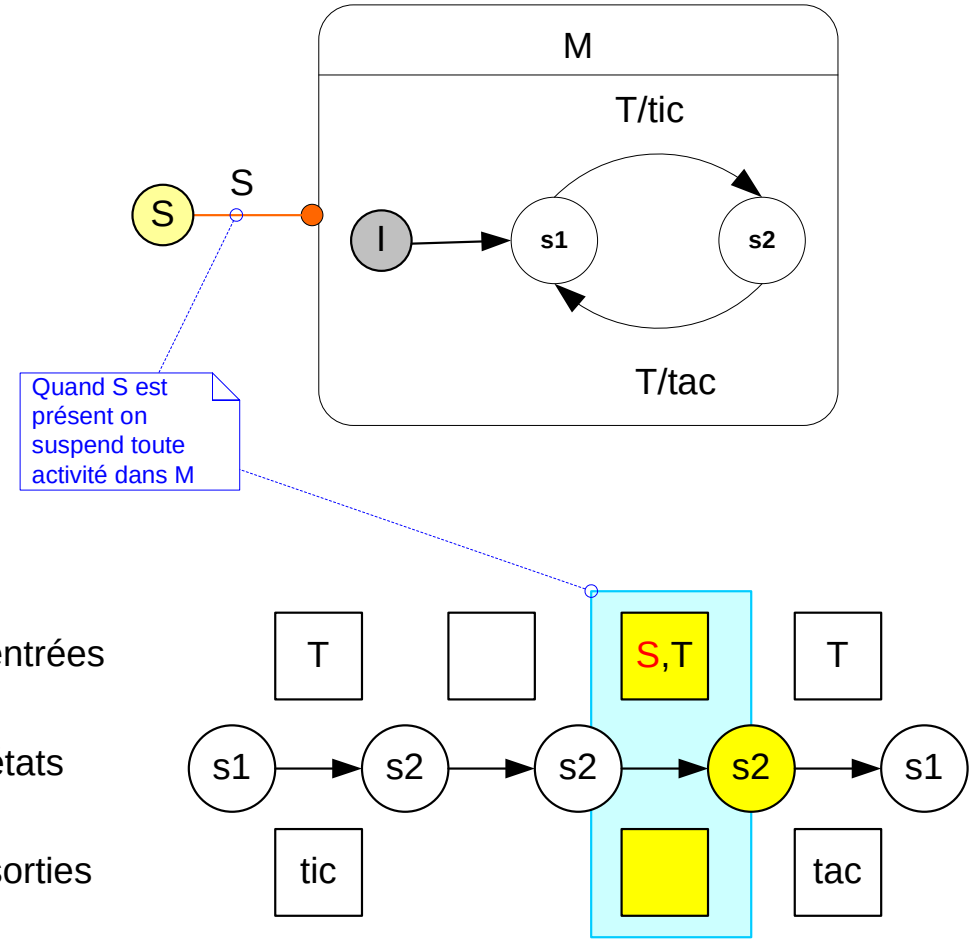
Souvent motivées par les applications (industrielles)

Impératif + déclaratif

- Signal: possibilités de machines à états, mais pas très commode
- Lustre: pas naturel
- Esterel v7: possibilités d'écrire des équations
  - dans les états d'un syncChart
  - dans des instructions **`sustain`**
- Inversement SCADE qui était basé sur Lustre admet des descriptions par machines à états

Esterel: Weak Suspend

suspend (normal)



Esterel: Weak Suspend

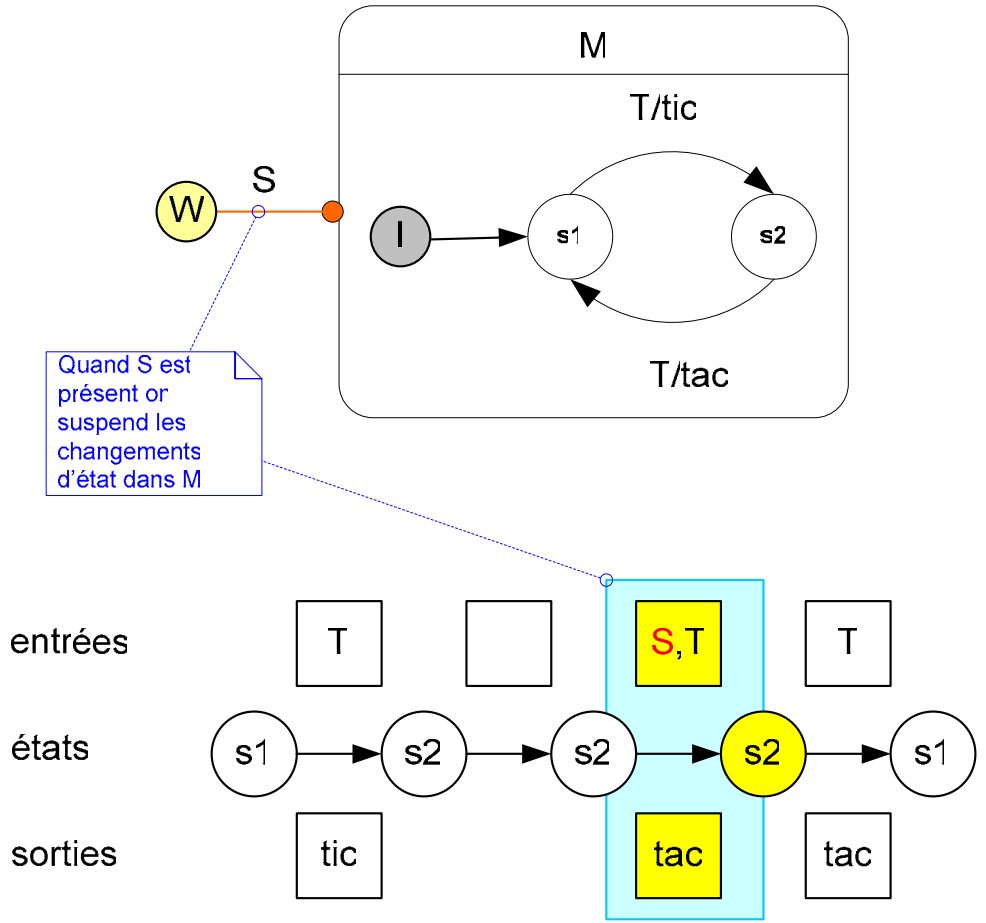
suspend (faible)

Pourquoi cette nouveauté?

Circuits : combinatoire / séquentiel

Difficulté : scoping

Utilisation : multiClock

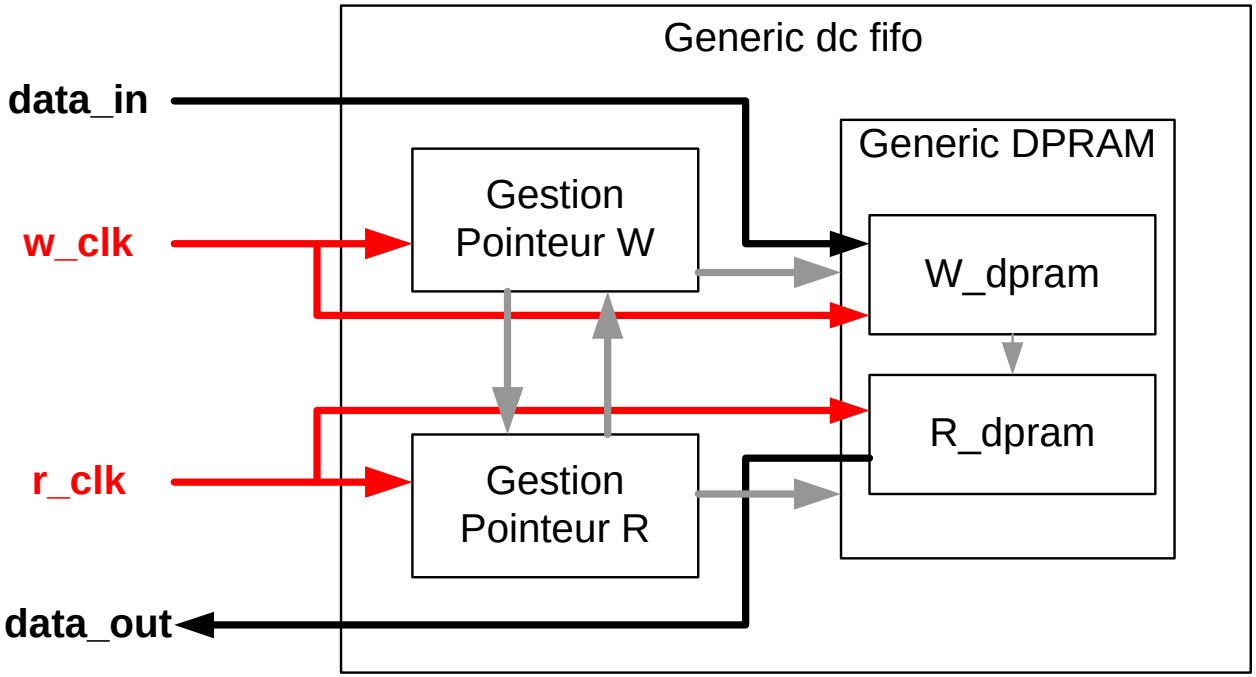


Esterel : Multiclock

- Motivé par applications circuits
- GALS (Globally Asynchronous, Locally Synchronous)

Exemple:

FIFO double clock



Lucid synchrone

- Extension récente*: n-synchronisme (horloges périodiques, insertion de buffers)

* Rapport de recherche INRIA n°5603, 2005

FIFO “synchrones” avec possibilité d’écriture/lecture simultanées

Où se documenter?

- André C., « Comparaison des styles de programmation de langages synchrones », Technical Report n◦ ISRN I3S/RR-2005-13-FR, I3S, Sophia-Antipolis, (F), Juin, 2005.
- Benveniste A., Caspi P., Edwards S. A., Halbwachs N., Le Guernic P., de Simone R., « The Synchronous Languages 12 Years Later », *Proceedings of the IEEE*, vol. 91, n◦ 1, p. 64-83, 2003.
- Zaffalon L. « Programmation synchrone de systèmes réactifs avec Esterel et les SyncCharts », Presses Polytechniques et Universitaires Romandes, 2005

Version étendue du papier contenu dans les actes de MSR05

La référence, avec de nombreuses indirections

Un des rares livres avec exercices