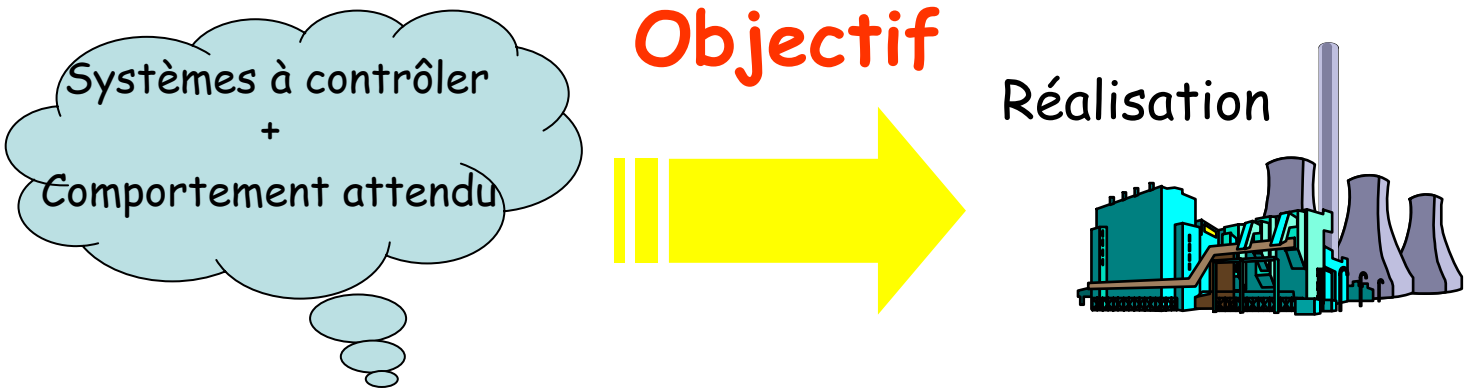
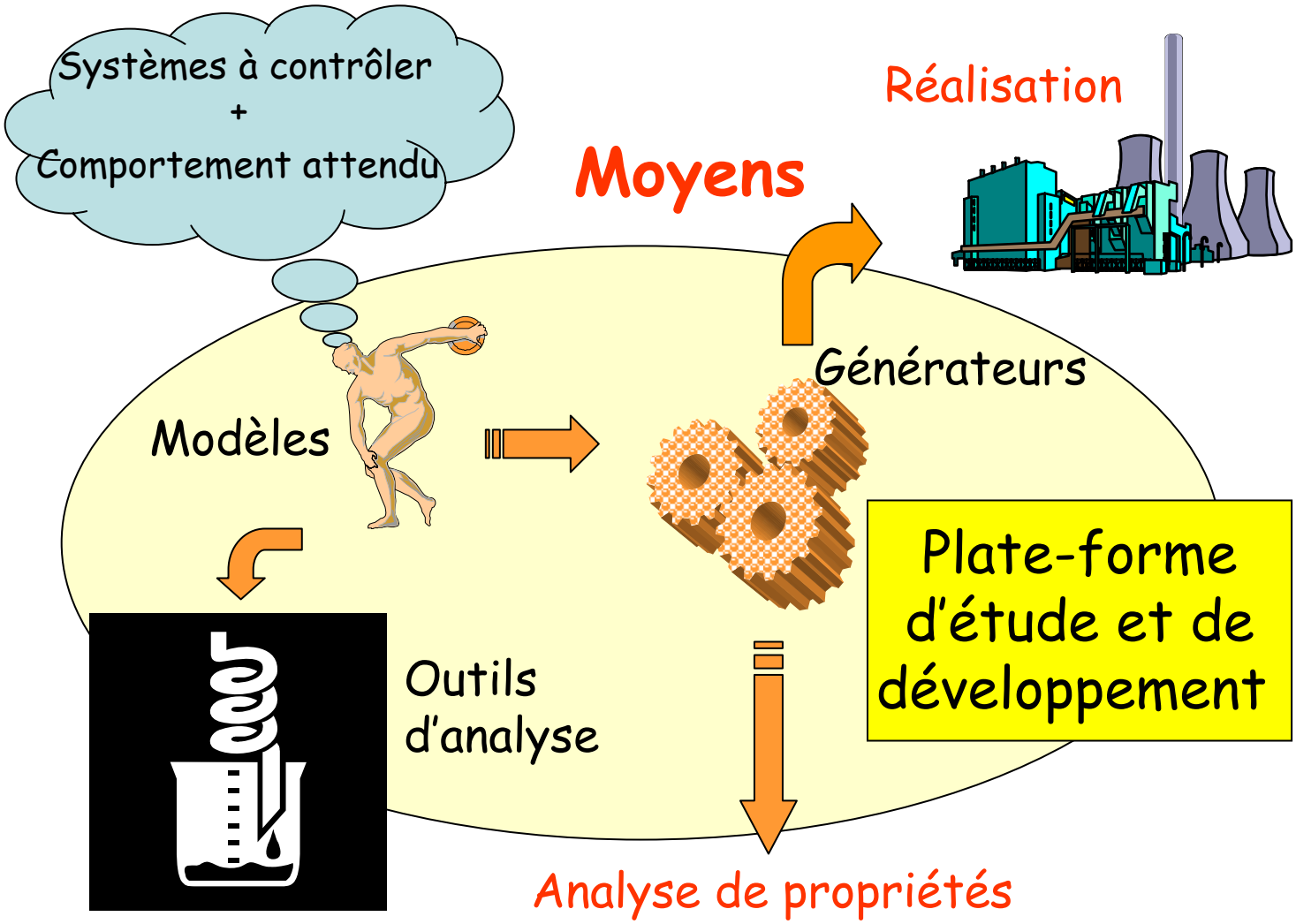
Programmation des Systèmes Réactifs

Chapitre 0 : Préface

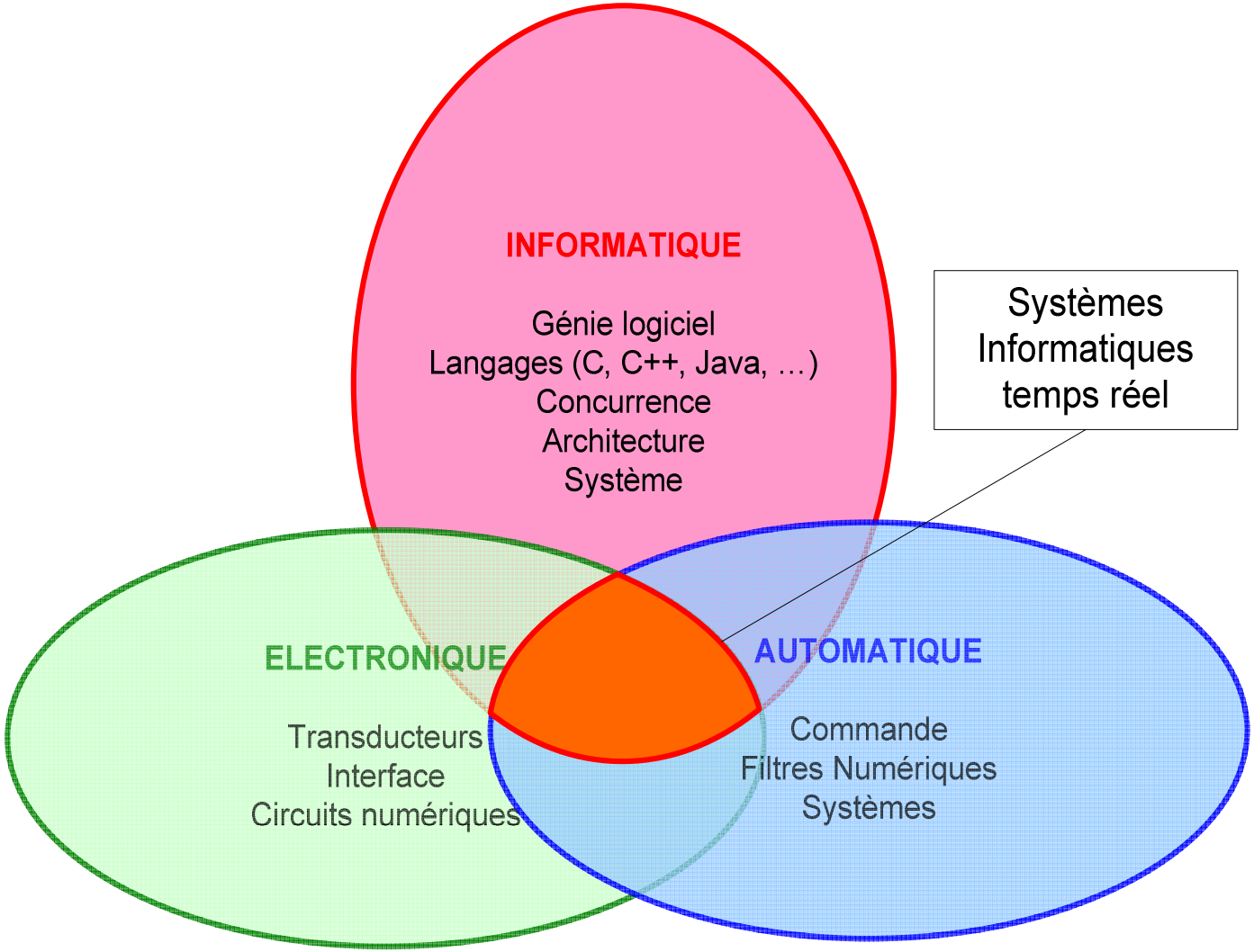
INFORMATIQUE INDUSTRIELLE



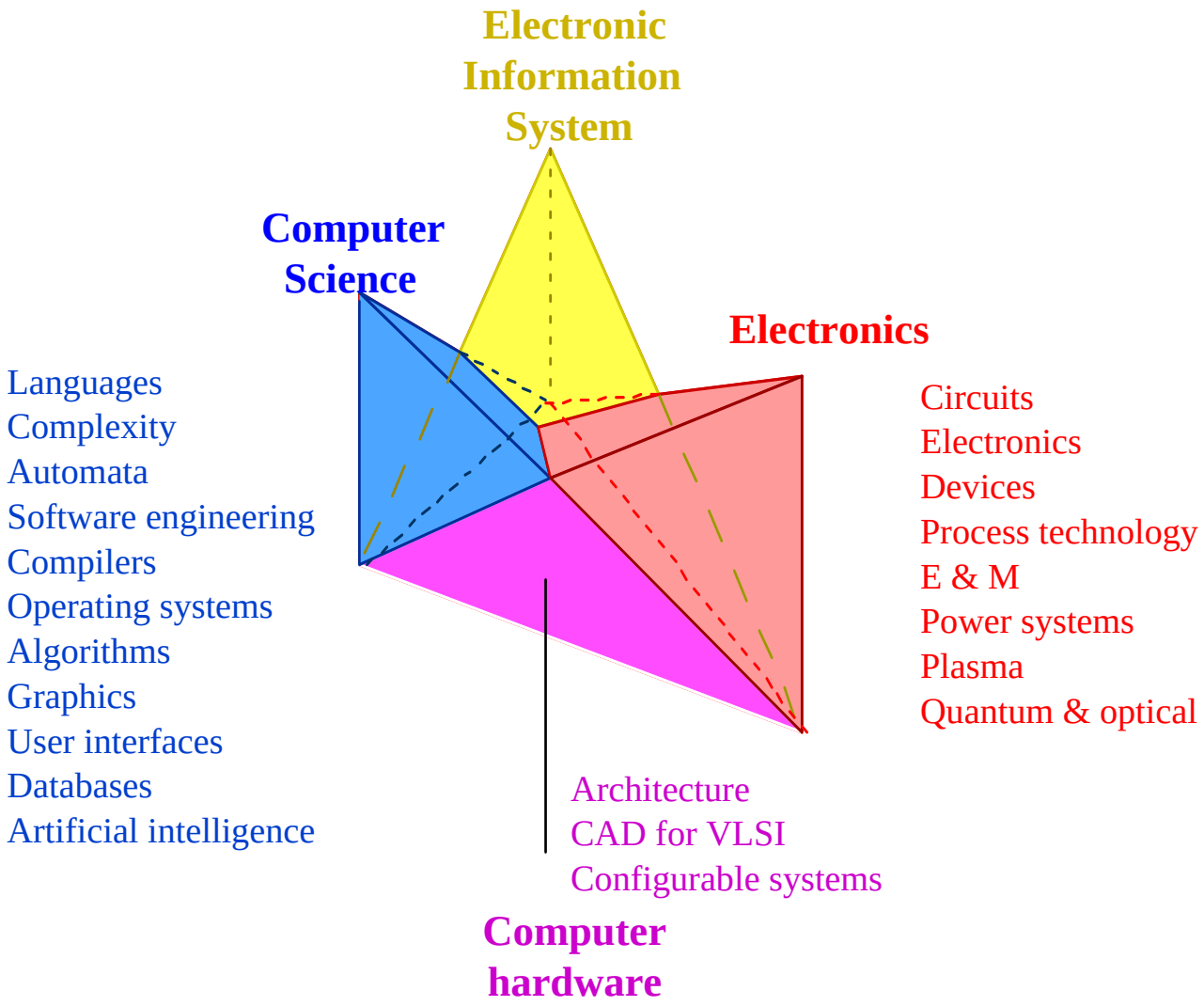
INFORMATIQUE INDUSTRIELLE



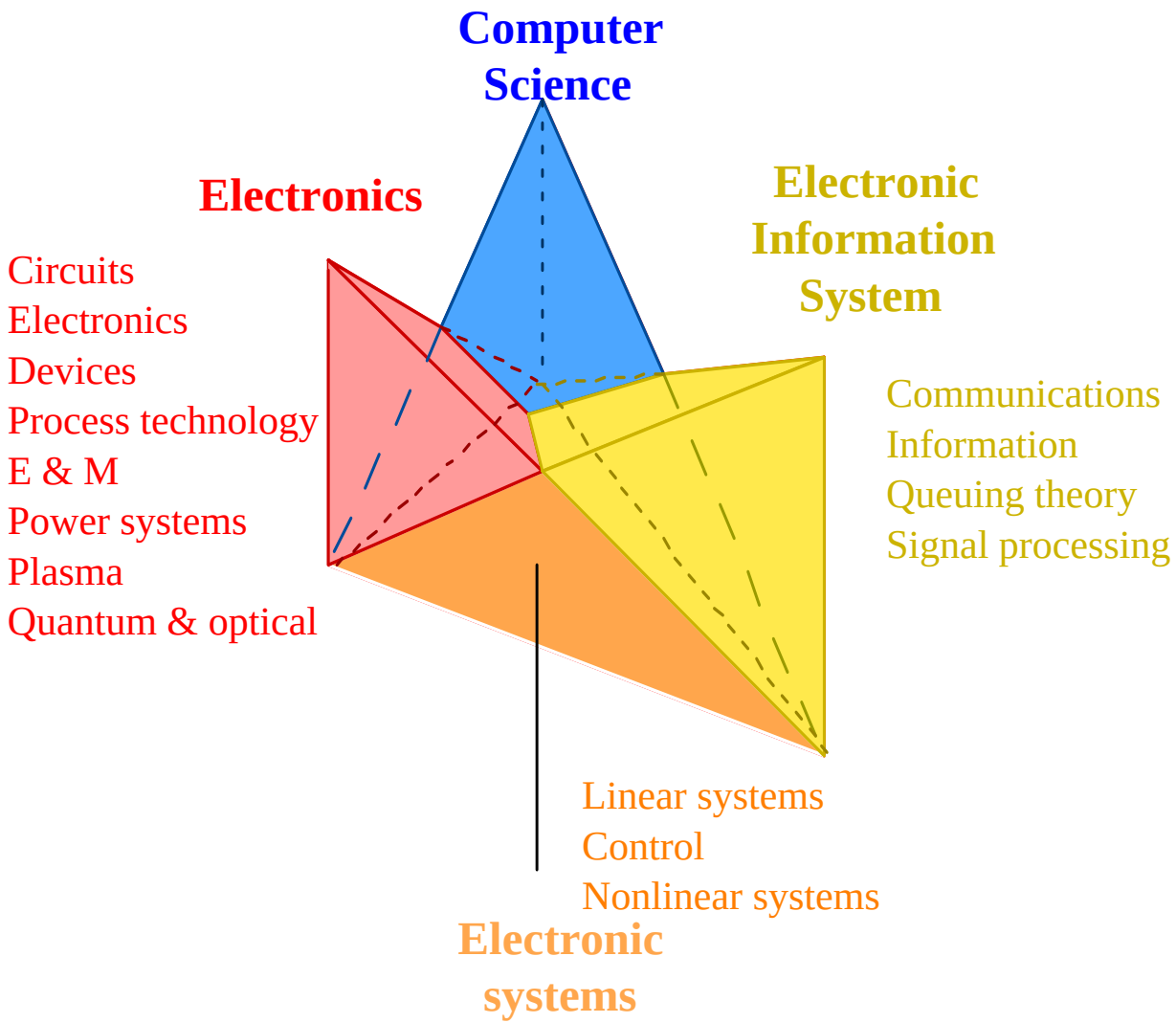
Temps Réel


INFORMATIQUE
Génie logiciel
Langages (C, C++, Java, …)
Concurrence
Architecture
Système
Systèmes
Informatiques
temps réel
ELECTRONIQUE
Transducteurs
Interface
Circuits numériques
AUTOMATIQUE
Commande
Filtres Numériques
Systèmes

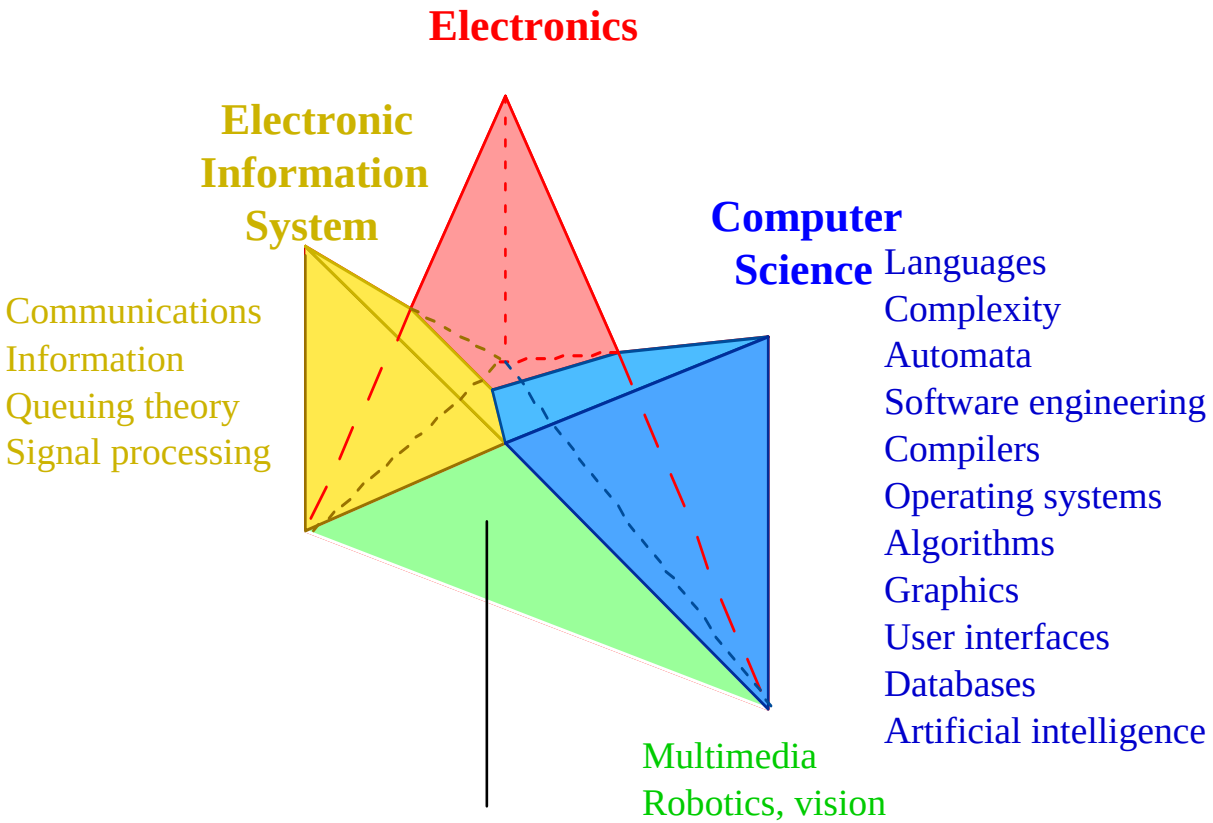
Electronique/informatique (1)



Electronique/informatique (2)



Electronique/informatique (3)



Systèmes réactifs et programmation synchrone

- Introduction aux systèmes temps réel
- Le paradigme synchrone
- Langages déclaratifs synchrones
- Langages impératifs synchrones
- Formalismes graphiques synchrones

INTRODUCTION aux SYSTÈMES TEMPS RÉEL

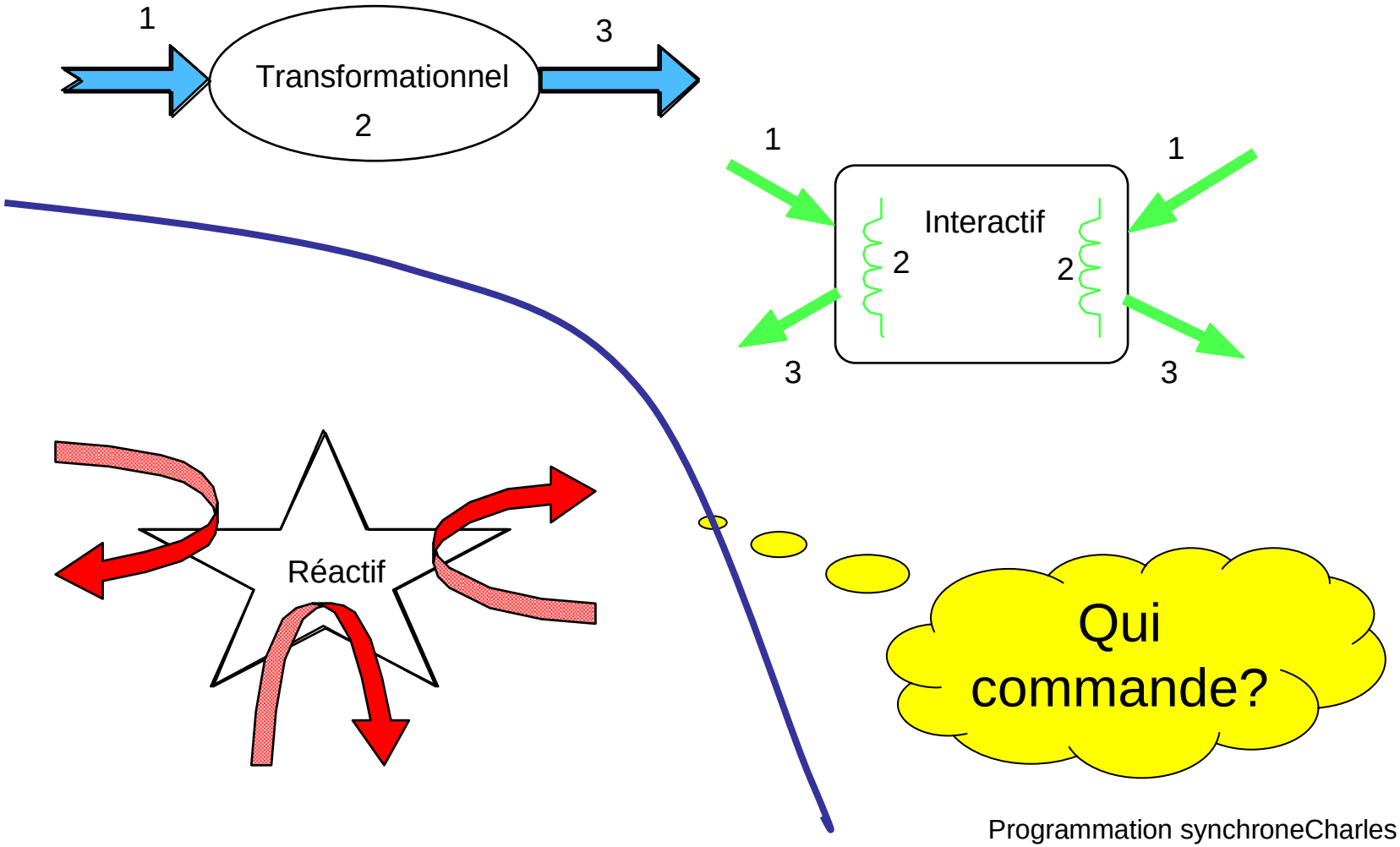
Chapitre 1

Définition

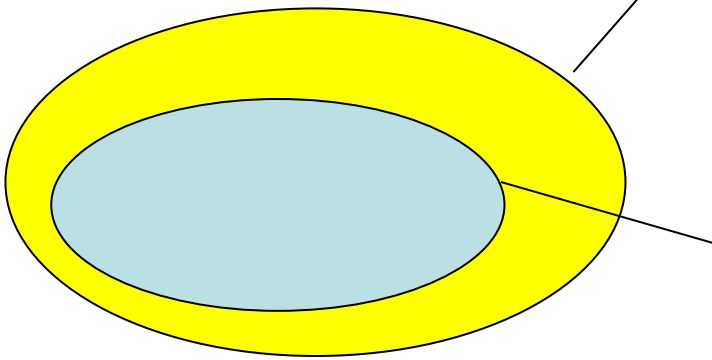
On qualifie de ***temps réel*** une application mettant en œuvre un système informatique dont le **comportement est conditionné** par **l'évolution dynamique** de l'état du **procédé** qui lui est **connecté**.
Ce système informatique est alors chargé de suivre ou de piloter ce procédé en **respectant des contraintes temporelles** définies dans le cahier des charges de l'application.

Jean-Pierre ELLOY, 1991

Transformationnel/Interactif Réactif



Systèmes Réactifs et Temps réel

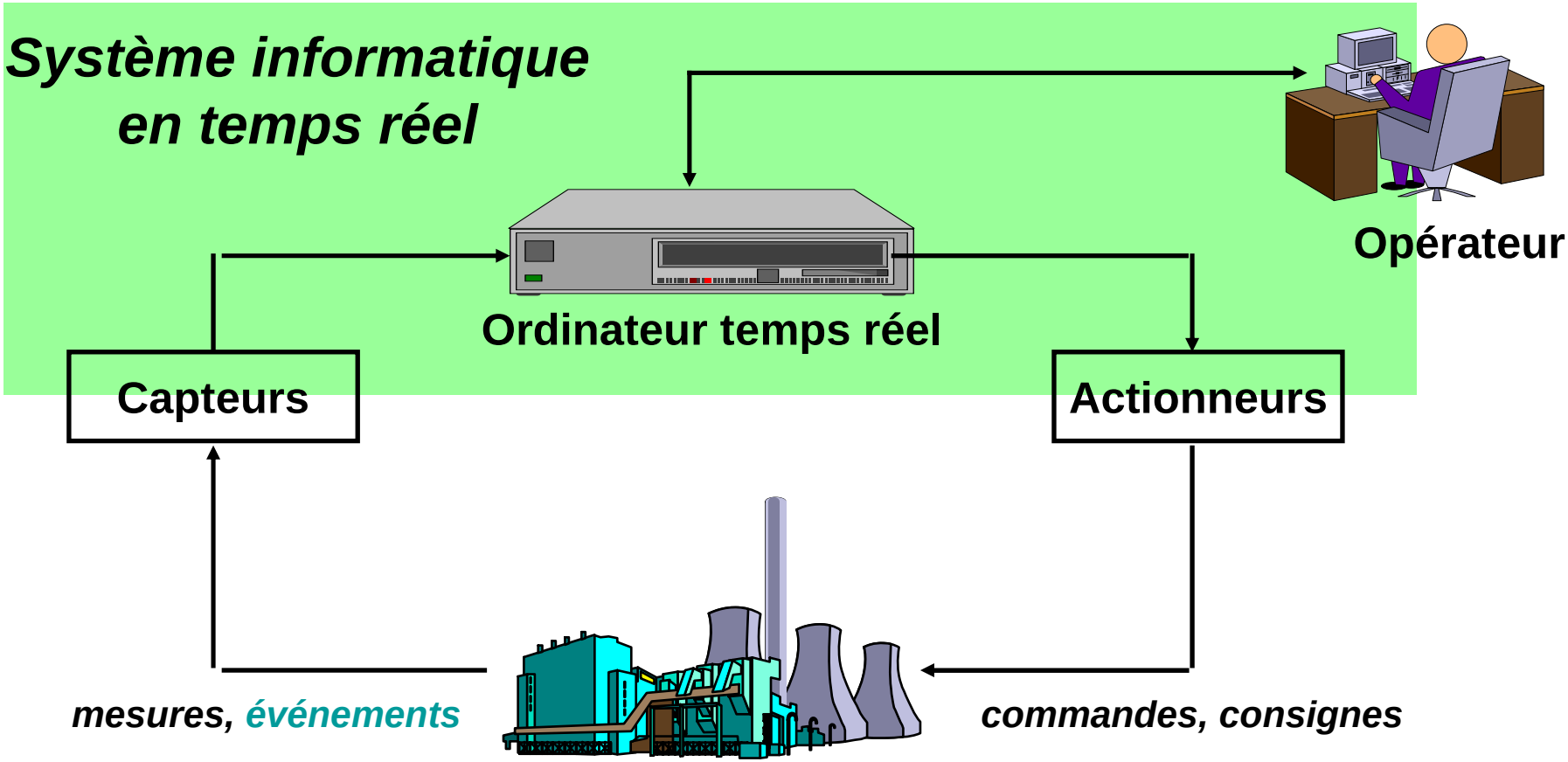


Système réactif= un système qui maintient une interaction permanente avec son environnement

Un système réactif temps réel = Système réactif soumis à contraintes (temporelles) extérieures

- Souci majeur Sûreté (Safety)
- Correction logique essentielle dans tous les cas
- Correction temporelle: une exigence supplémentaire pour les STR

Interactions



Contraintes Temporelles Strictes

Temps réel à **contraintes strictes** « hard real-time »

- Le respect des contraintes temporelles est **indispensable** au bon fonctionnement.
- Le non-respect conduit
  - à des **résultats inexploitables**
  - à des **risques** pour la mission, le système ou son environnement

Contraintes Temporelles Relatives

Temps réel à **contraintes relatives** « soft real-time »

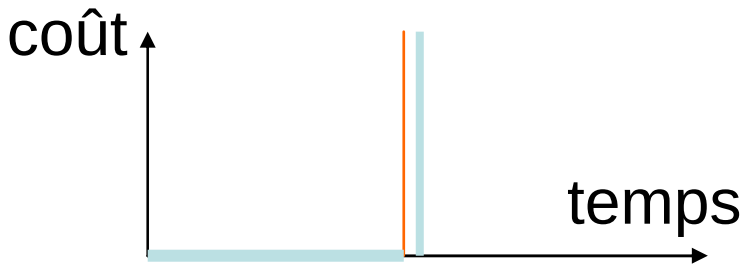
- Le non-respect des contraintes temporelles ne conduit pas à un mauvais fonctionnement
- Il induit juste certaines **nuisances**
  - coût supplémentaire
  - agacement des opérateurs (temps de réponse) imprécisions admissibles…

Temps réel à **contraintes mixtes** « firm real-time »

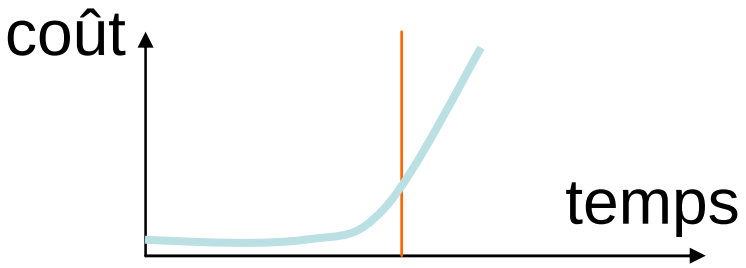
Contraintes strictes/relatives/fermes

Coût du dépassement de l'échéance :

- Résultat sans intérêt
- Dangereux



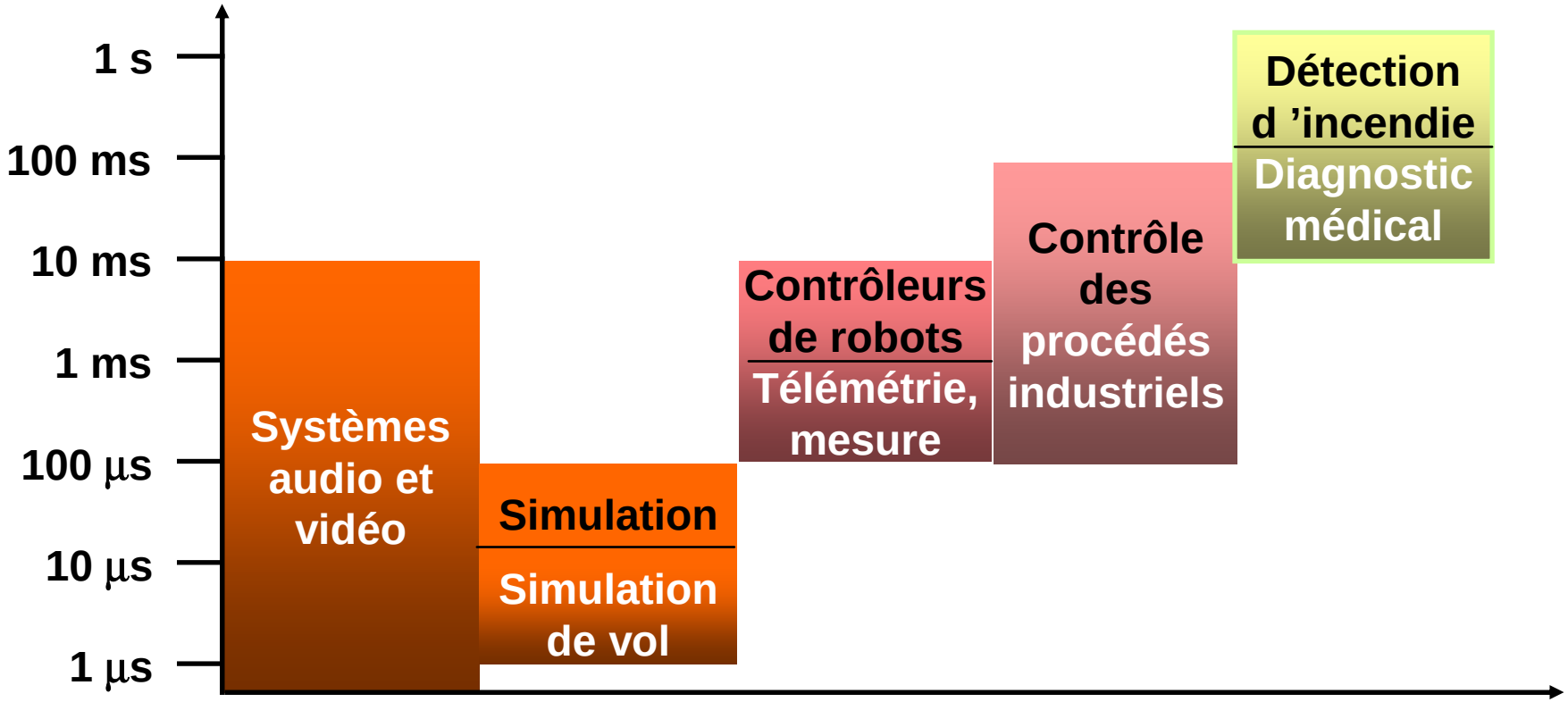
- Croissance du coût avec le retard



Analyse :

- Pire cas
- Performance en moyenne

Contraintes temporelles



(d'après B. Furht et al., Real-Time UNIX Systems, 1991)

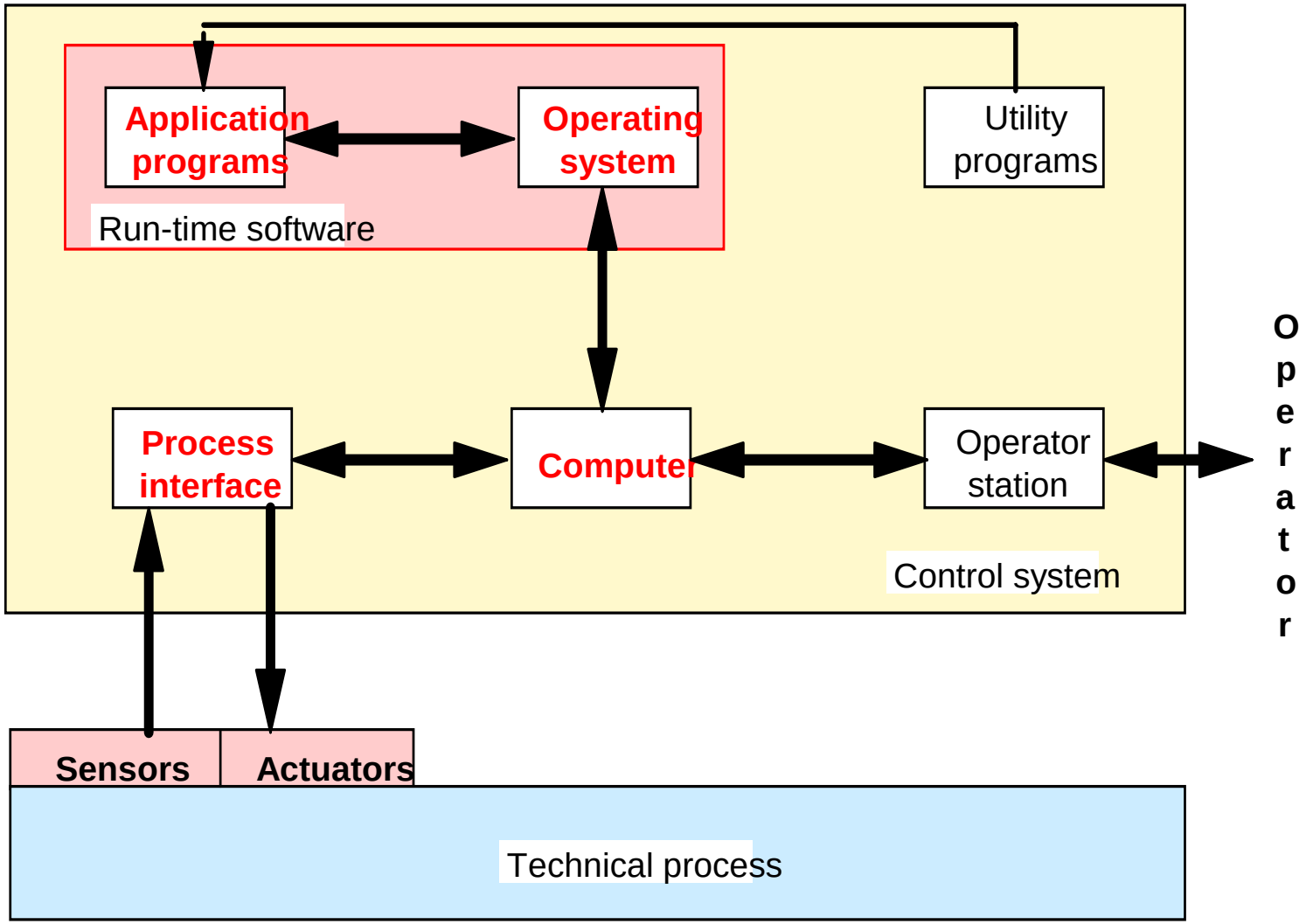
Ce que l’on attend

- Efficacité des opérations
- Facilité des opérations
- Sûreté
- Amélioration des produits
- Réduction des rebuts
- Réduction du travail répétitif fastidieux

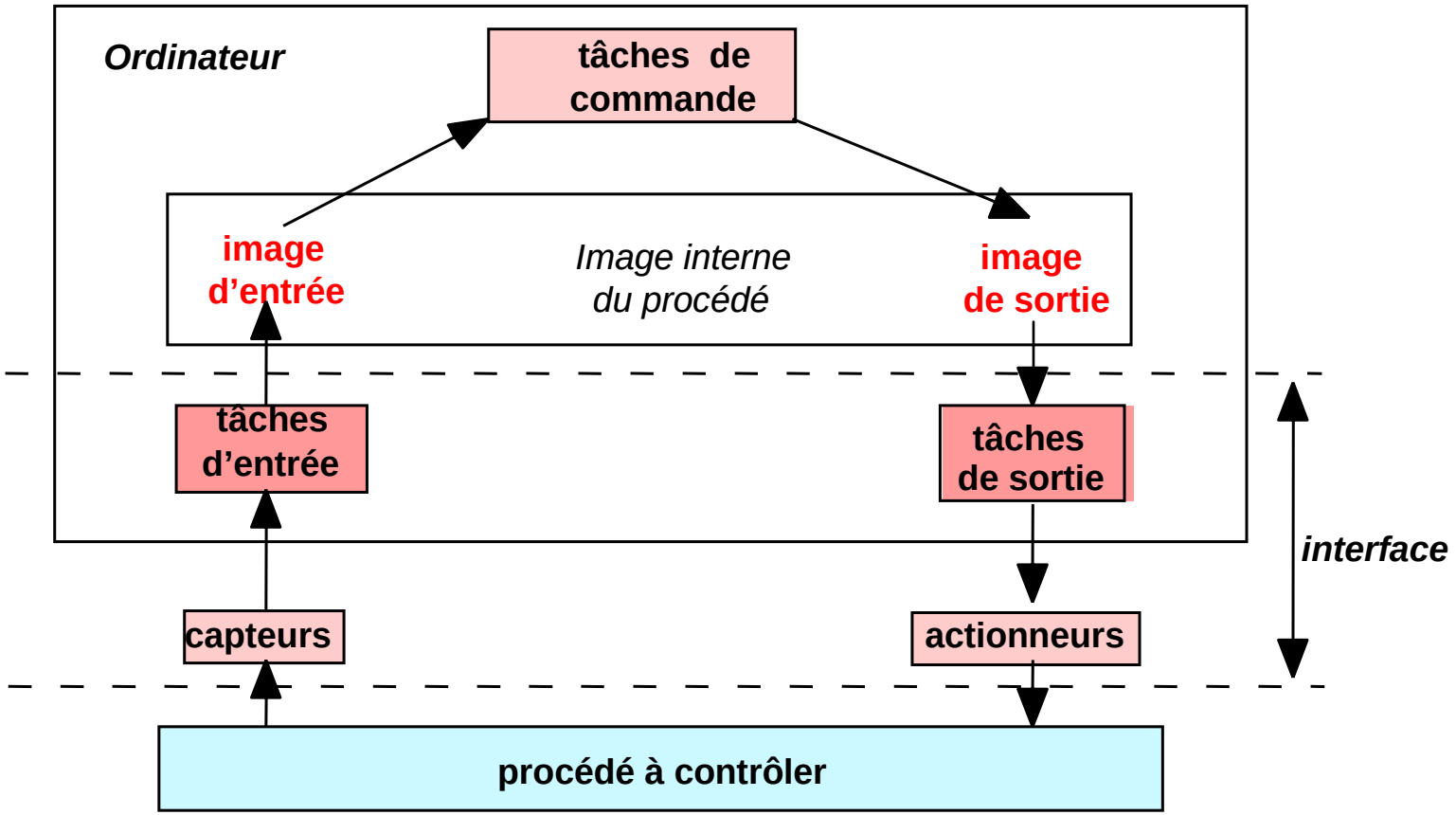
Fonctions à assurer

- Acquisition des données
- Contrôle séquentiel
- Boucle de contrôle
- Supervision
- Analyse des données
- Stockage des données
- Interface homme-machine

Commande en Temps Réel



Interfaces


Ordinateur
tâches de commande
image d'entrée
Image interne du procédé
image de sortie
tâches d'entrée
tâches de sortie
interface
capteurs
actionneurs
procédé à contrôler

Les principaux problèmes

La plupart des systèmes réactifs temps réel sont faits de composants qui évoluent concurremment et communiquent entre eux.

On retrouve donc les problèmes dus au parallélisme (aspects qualitatifs):

- Communication
- Synchronisation
- Organisation du flot de calcul

Il y a en plus les aspects quantitatifs, imposés par les contraintes temporelles, essentiellement liés à la vitesse d'exécution.

Exigences particulières des systèmes temps réel

- Contraintes temporelles *(timing constraints)*
- parallélisme *(concurrency)*
- prévisibilité des comportements *(predictability)*
- sûreté de fonctionnement *(dependability)*

Comment sont-ils programmés?

- Langage impératif séquentiel (C)+RTOS
  - (+) solution la plus répandue
  - (-) ressemble à plusieurs programmes plus ou moins reliés
  - (--) difficultés pour la mise au point et la maintenance
  - (--) peu de possibilités d'analyse automatique
  - (--) le RTOS peut introduire du non déterminisme
- Machine à états finis
  - (+) déterministe
  - (-) ne supporte pas directement la hiérarchie et le parallélisme
  - (--) très sensible aux modifications de la spécification
  - (--) seules les machines de petite taille sont humainement compréhensibles

Comment sont-ils programmés?

- Réseaux de Petri
  - (+) support du parallélisme
  - (+) analyse mathématique
  - (-) peu modulaire
  - (--) non déterministe (en standard)
- Langages de programmation concurrente
  - ( ) ADA, (applications certifiées D.O.D)
  - ( ) Java “temps réel” (standard?)

Ce que l'on voudrait

- Utilisation de techniques modulaires et formelles pour spécifier, implanter et vérifier les programmes
- Avoir un cadre pour traiter tous les systèmes réactifs
- Pouvoir traiter des architectures réparties
- Assurer le déterminisme autant que possible
- S'occuper des problèmes de performance

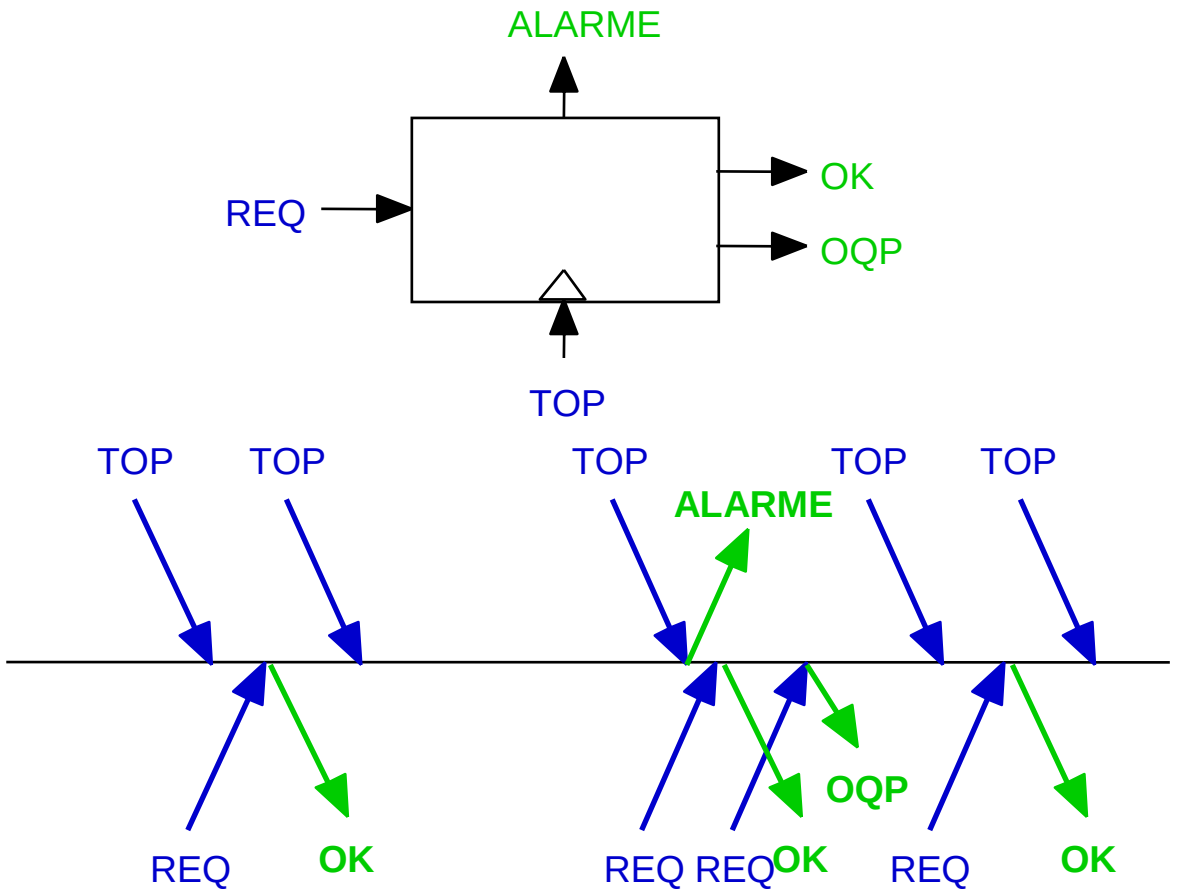
Systèmes Réactifs

Chapitre 2

Systèmes réactifs à Contrôle prépondérant

- Contrôle de processus temps-réel
- Systèmes embarqués
- Supervision de systèmes complexes
- Protocoles de communication *
- Pilotes de périphériques
- Contrôleurs matériels
- interface homme-machine

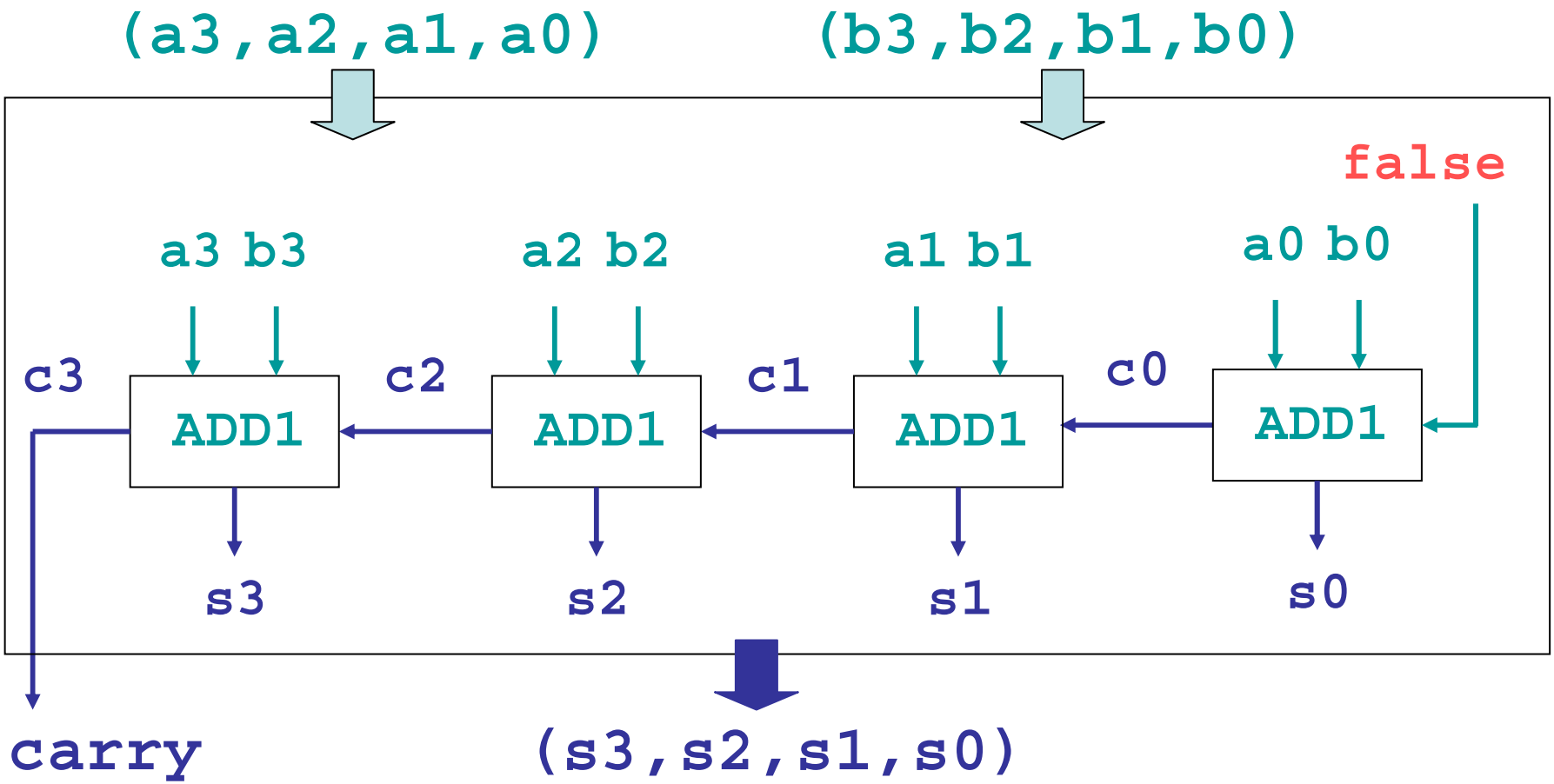
Top/Req


ALARME
REQ
OK
OQP
TOP
TOP
TOP
TOP
TOP
TOP
ALARME
OQP
REQ
OK
REQ REQ
OK
REQ
OK

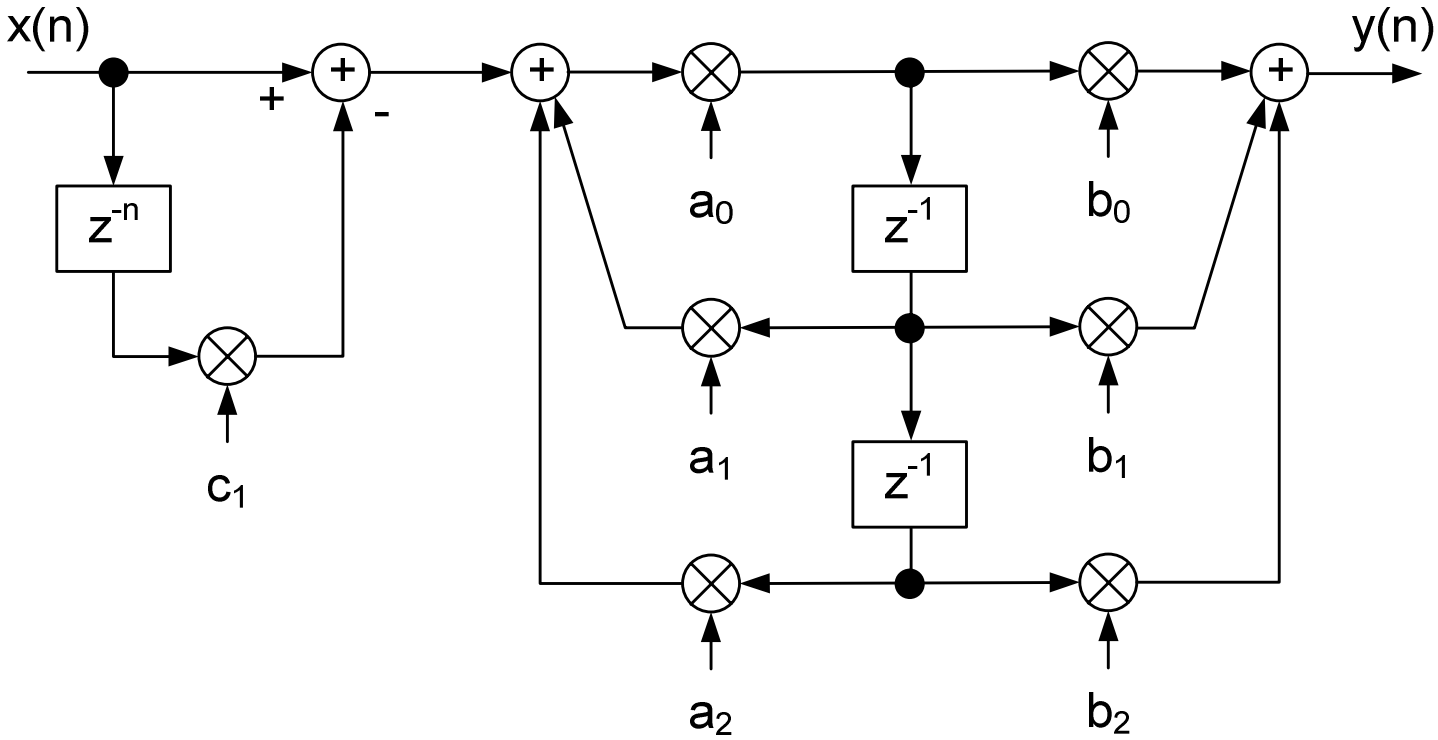
Watchdogs

- Entrées :
  - set
  - reset
    - deadline : integer
- Sorties :
  - alarm

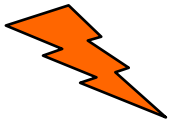
Circuits arithmétiques



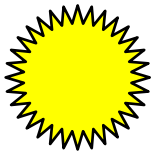
Filtres numériques

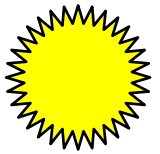


Star Trek



Launch!!

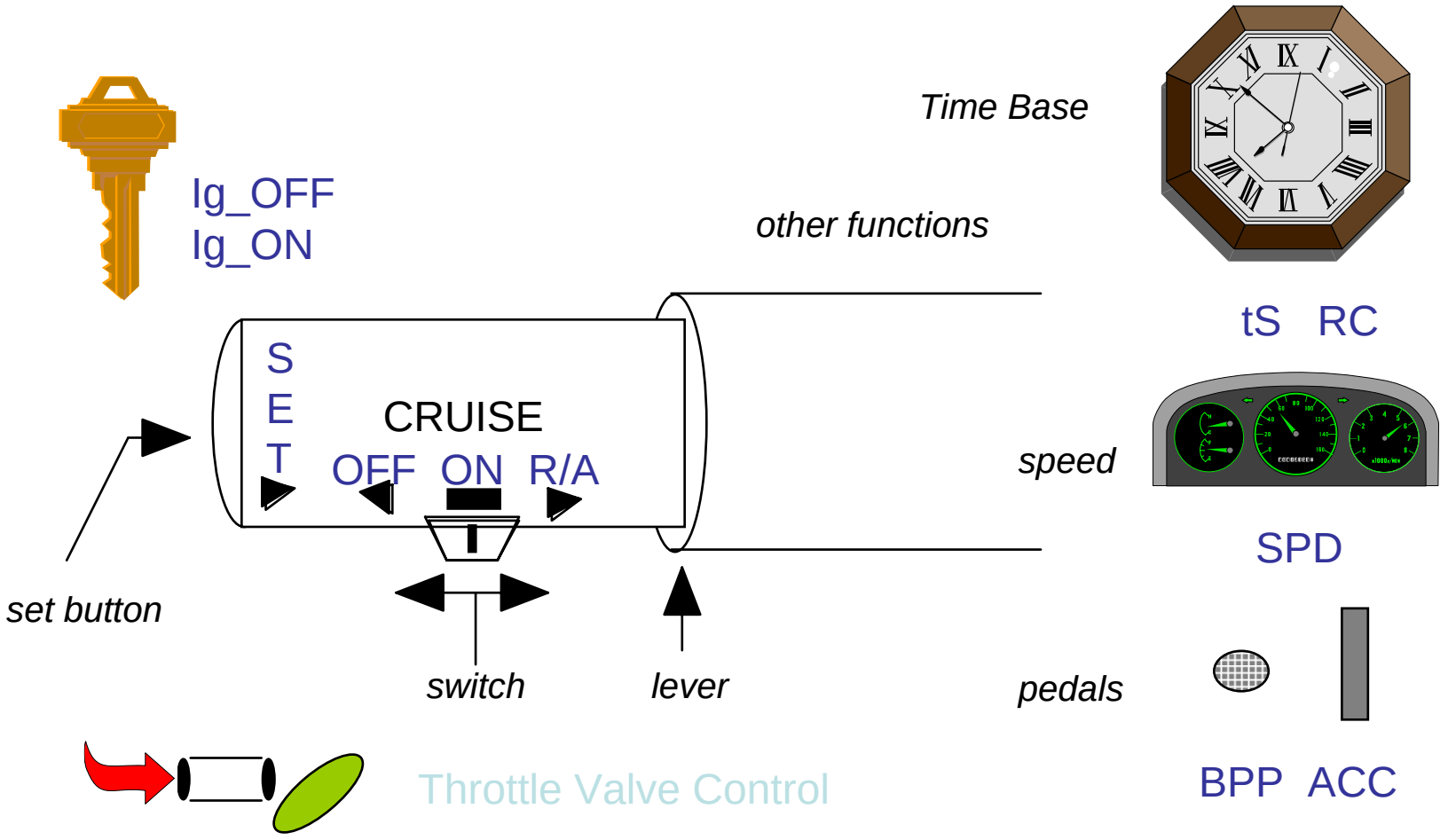




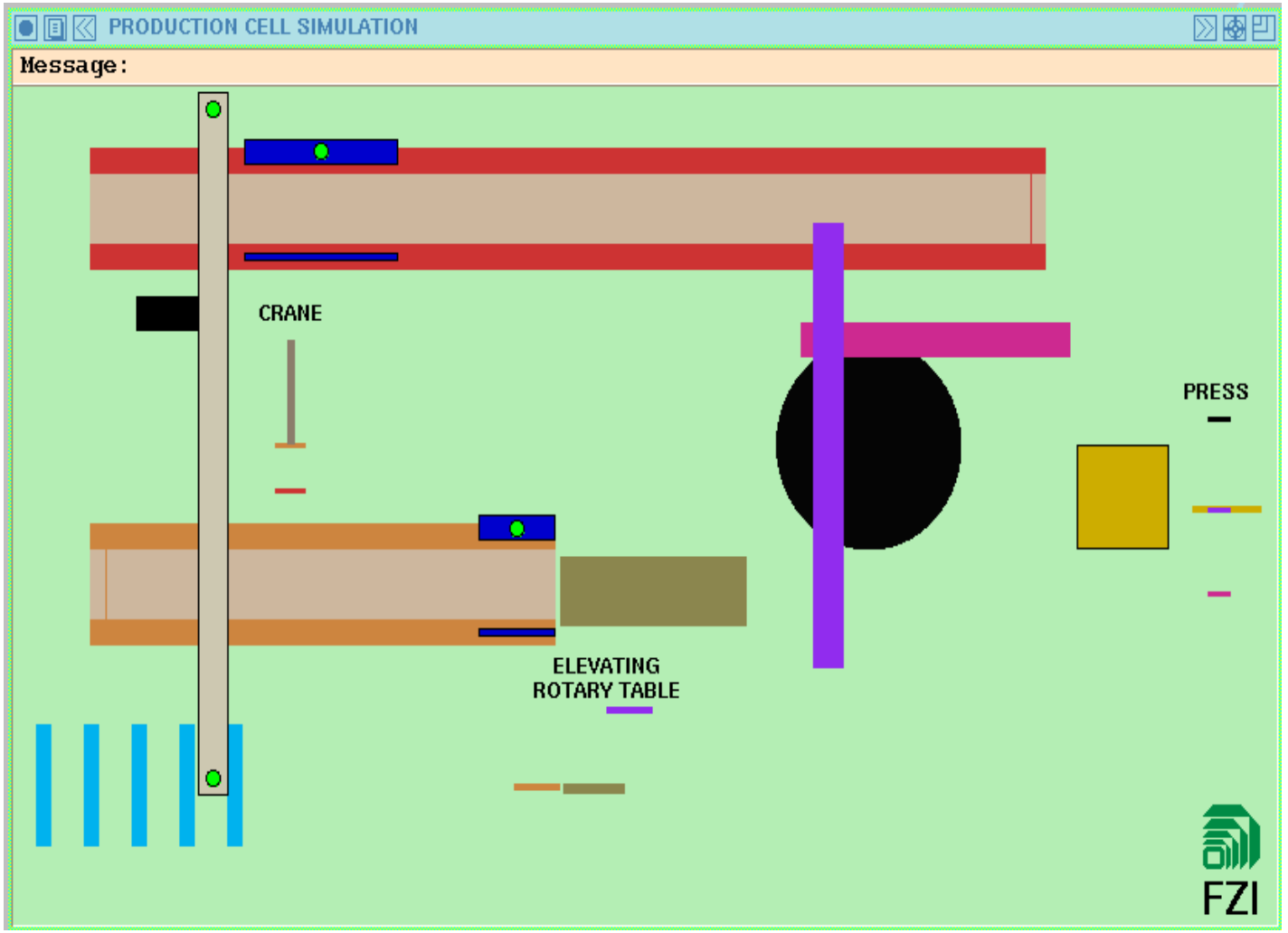
Arm

Arm

Cruise control



Cellule de Production Automatisée



Tk Button: Default Bindings

Tk automatically creates class bindings for buttons that give them default behavior:

- [1] A button activates whenever the mouse passes over it and deactivates whenever the mouse leaves the button. Under Windows, this binding is only active when mouse button 1 has been pressed over the button.
- [2] A button's relief is changed to sunken whenever mouse button 1 is pressed over the button, and the relief is restored to its original value when button 1 is later released.

Tk Button (continued)

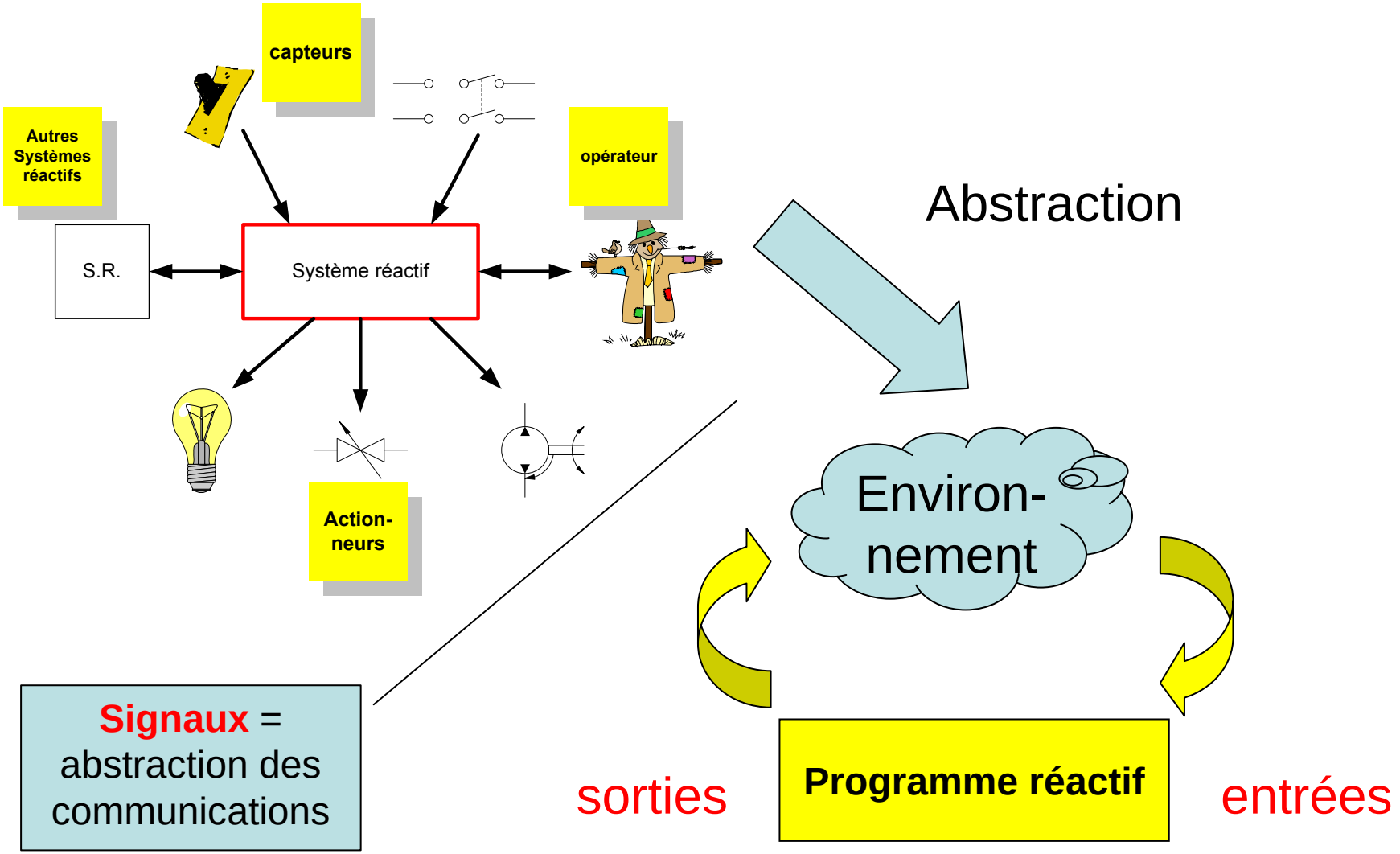
- [3] If mouse button 1 is pressed over a button and later released over the button, the button is invoked. However, if the mouse is not over the button when button 1 is released, then no invocation occurs.

- [4] When a button has the input focus, the space key causes the button to be invoked.

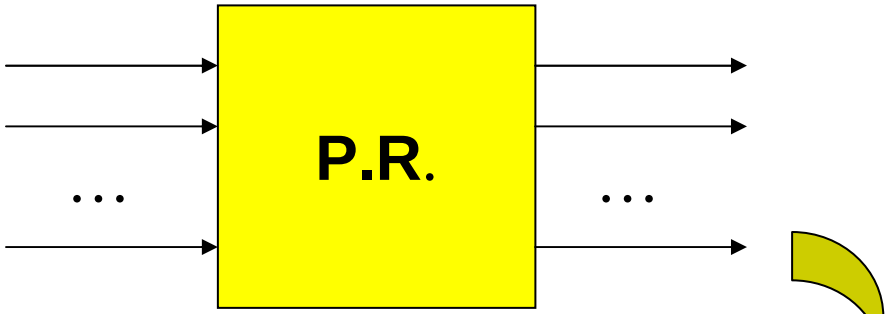
Déterminisme & Réactivité

- **Déterminisme**
  - même séquence d ’entrée $\Rightarrow$
  - même séquence de sortie
- **Réactivité**
  - possibilité de répondre à tout stimuli

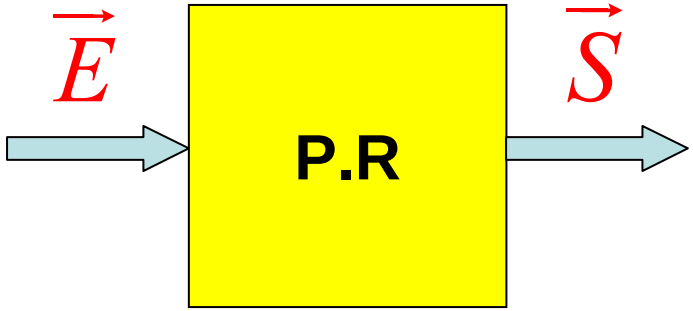
Hypothèses synchrones (1)



Hypothèses synchrones (2)



Travaille sur
Uplets (“vecteurs”)



Hypothèses synchrones (3a)

- Il y a des actions que l'on VEUT disjointes dans le temps, d'autres que l'on voudrait ou qui devraient être simultanées:
- Exemple:
  - **As**: action d'incrémenter un compteur de secondes de 1
  - **Am**: action d'incrémenter un compteur de minutes de 1
  - **Ac**: action composée qui consiste (éventuellement) à agir sur les deux compteurs

Hypothèses synchrones (3b)

Un langage de programmation pour système réactif DOIT être capable d'exprimer de tels comportements et doit être compositionnel.

Soit

- δs la durée nécessaire pour l'action **As**.
- δm la durée nécessaire pour l'action **Am**.
- δc la durée nécessaire pour l'action **Ac**.

Lorsqu'il faut exécuter As et Am, la durée est $\delta = \delta s + \delta m$ qui devrait être égale à δc

Il faut donc que $\delta c = \delta s + \delta m$

Ceci peut être réglé en prenant

- $\delta c = \delta s = \delta m = ?$ — asynchronisme
- ou $\delta c = \delta s = \delta m = 0$ — synchronie

Hypothèses synchrones (3c)

- Considérer la durée des réactions comme nulle est une idéalisation qui permet de donner un sens précis à la simultanéité:
- Quand les compteurs sont à

  Am = 11 et As = 59

  À la prochaine action Ac on doit passer à

  Am = 12 et As = 00

  Et non pas par des situations intermédiaires comme

  Am = 11 et As = 00
- Ceci n'empêche pas qu'il existe un ordre dans les actions : il y a une relation de cause à effet entre l'incrémentation de As lorsqu'il est à 59 et l'incrémentation de Am qui résulte d'une "propagation de retenue".

Hypothèses synchrones (4)

- Une réaction synchrone
  - Exécute un ensemble d'actions (simultanées)
  - Ces exécutions peuvent être partiellement ordonnées (causalités)
  - Pour les raisonnements on considère que toutes les actions exécutées dans un instant ont une durée nulle.
- Le modèle synchrone travaille en fait sur un temps logique, discret. Les instants sont disjoints et totalement ordonnés (on peut les indicer par $\mathbb{N}$).

Hypothèses synchrones (5)

- Dans le même esprit
  - Les communications entre modules synchrones se font dans l'instant : c'est la diffusion instantanée.
  - Conséquence : les entrées sont disponibles au même instant que les sorties (les sorties sont synchrones avec les entrées).
- Cette hypothèse très forte permet, en particulier, de définir une composition parallèle déterministe (e.g., il n'y a pas des phénomènes de course critique)

Hypothèses synchrones (résumé)

- Signaux : support de communication
- Echantillonnage parfait : uplets
- Temps logique : instant, simultanéité
- Durée nulle des réactions
- Perception globale : diffusion instantanée

Synchrone et temps réel (1)

- Le temps logique est bien différent du temps “physique”. Puisqu’on veut utiliser la programmation synchrone pour le temps réel il faut bien lier les deux.
- La façon la plus simple est d’exécuter un programme synchrone cycliquement

```
faire sans fin
   attendre l’activation
   acquérir les entrées
   exécuter les actions
   appliquer les sorties
 fait
```

Exécution du corps de boucle ↔ **instant**

Synchrone et temps réel (2)

- Ce mode d'exécution, lorsque les exécutions sont périodiques, convient très bien pour faire de la commande. Si privilégie le caractère réactif on procède plutôt ainsi

```
faire sans fin
   attendre l'occurrence d'un événement d'entrée
   exécuter les actions
   appliquer les sorties
fait
```

Synchrone et temps réel (3)

- Parmi les signaux d'entrée certains peuvent être liés à un mécanisme de mesure du temps ("horloges").
- Comme les exécutions sont parfaitement déterministes, on peut par une analyse du code engendré, savoir la durée (physique) exacte d'une réaction.
- On peut donc faire, a posteriori, une validation de l'hypothèse de durée nulle. (Bien se souvenir que fonctionner en temps réel n'a de sens que pour un environnement de dynamique donnée).

Synchrone et temps réel (4)

Remarque

- Les circuits numériques utilisent depuis longtemps une approche analogue:
  - Les circuits dits “Synchrones” sont plus faciles à développer que les circuits asynchrones.
  - Il y a des cas où on ne peut pas éviter une certaine dose d’asynchronisme
  - C’est pareil en programmation, la programmation synchrone ne pourra pas tout faire (voir limitations plus tard).

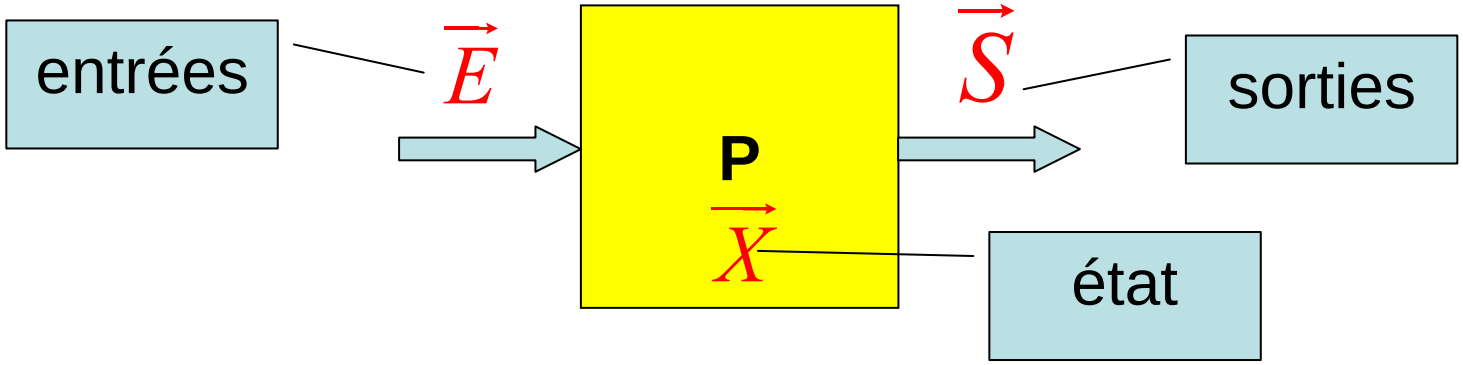
Sémantique (1)

- Les langages synchrones sont des langages spécialisés, pas des langages d'usage général.
- Ils ont été conçus spécialement pour la programmation réactive.
- Ils sont à la fois
  - plus simples (nombre limité de constructeurs)
  - plus propres (sémantique mathématique)

que les langages d'usage général

- Ils sont souvent plus efficaces à l'exécution (compilation sophistiquée)

Sémantique (2)



La signification du programme P (sa sémantique) est donnée par la caractérisation de toutes les correspondances possibles entre séquences d'entrées (E_1;E_2;…) et séquences de sorties (S1;S2;…)

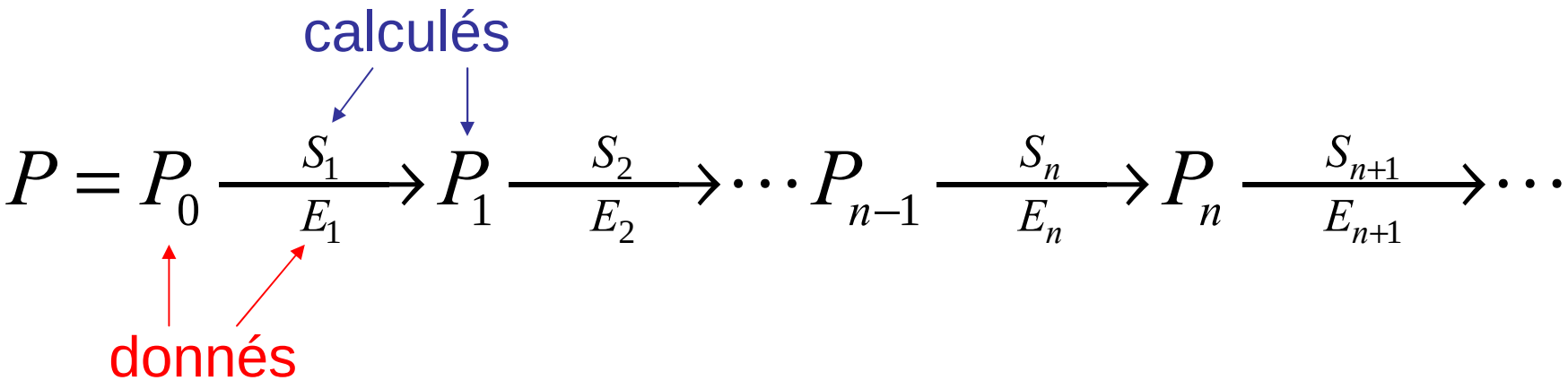
Sémantique (3)

Un programme synchrone étant déterministe, la correspondance est fonctionnelle :

$\overrightarrow{X'} = f(\vec{X}, \vec{E})$ Fonction état suivant

$\vec{S} = g(\vec{X}, \vec{E})$ Fonction de sortie

$\overrightarrow{X_0}$ donné Etat initial



Conclusion

- Langages synchrones = nouvelle approche de la programmation réactive et temps réel.
- Des perspectives intéressantes annoncées …
- Voir la suite du cours pour constater que ce ne sont pas de simples effets d'annonce!