Master Ingénierie Système en EEAPR Spécialité R2
« Productique et Réseaux »

UE Ossature R2-1

« Evaluation des Systèmes à Evénements Discrets »

Les Laboratoires impliqués dans cette spécialité

- **GREEN (Groupe de recherche en électrotechnique et électronique de Nancy)**

- **LIEN (Laboratoire d'instrumentation électronique de Nancy)**

- **LORIA** (Laboratoire Lorrain de Recherche en informatique et ses applications)

- **CRAN** (Centre de Recherche en Automatique de Nancy)
Equipe «Systèmes de production ambiants» (SYMPA)
Equipe «Sûreté de fonctionnement et diagnostic» (SURFDIAG)

- **LURPA** (Laboratoire Universitaire de Recherche en Production Automatisée)
Equipe « Ingénierie des systèmes automatisés (ISA)

Laboratoire **U**niversitaire de **R**echerche en **P**roduction **A**utomatisée

Un laboratoire de Productique

Deux objets de recherche :

- **Les produits mécaniques**

Objectif : Maîtrise de leur qualité géométrique

Equipe « Géométrie tridimensionnelle des pièces et des mécanismes » (Géo3D)

Thèmes : - Dimensionnement et tolérancement géométrique (pièces et assemblages)
- Qualité géométrique des pièces usinées
- Mesure et contrôle par coordonnées

- **Les systèmes de production**

Objectif : Sûreté de fonctionnement de leur système de commande

Equipe « Ingénierie des Systèmes Automatisés » (ISA)

Thèmes : - Modélisation et analyse des systèmes de commande en réseau
- Vérification formelle des contrôleurs logiques
- Synthèse de contrôleurs

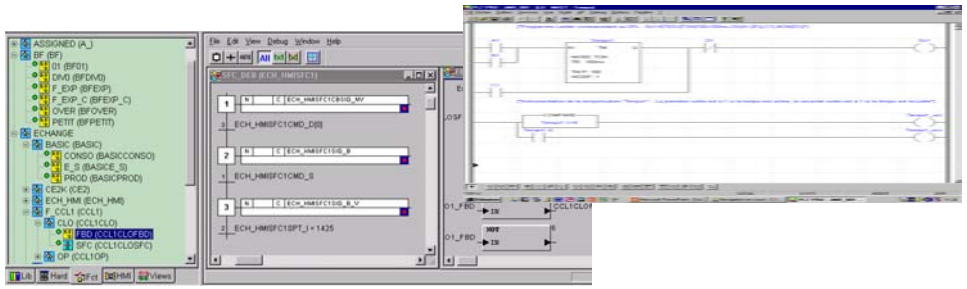
Caractéristiques essentielle des recherches conduites dans l'équipe ISA

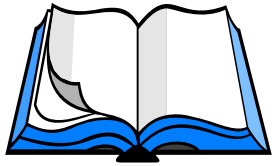
Contribuer à la sûreté de fonctionnement des Systèmes à Evénements Discrets par une recherche technologique s'appuyant sur les modèles et théories de la Commande des SED, c'est-à-dire prenant en compte des difficultés scientifiques inhérents aux besoins industriels

- taille des modèles
- utilisation de langages « métiers », souvent non formels, pour la conception et l'implantation des systèmes de commande
- implantation sur composants d'automatisation (Automates programmables Industriels (API), contrôleurs temps réel, …) distribués sur réseaux de communication multiples
- …

Exemple 1 : Vérification formelle des programmes d'API

Programmes API

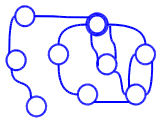




Propriétés extraites du cahier des charges

Formalisation

Formalisation



Automate à états finis

AG(APB→AF ~horn)

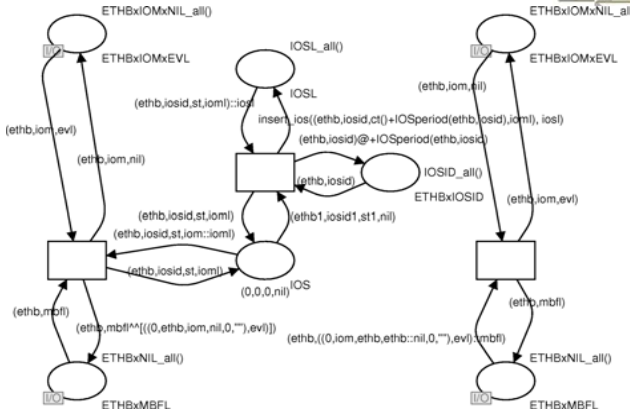
AG(~d1→AF ~lig)

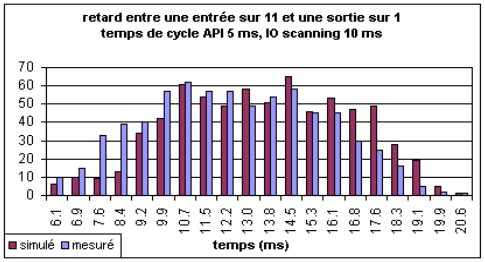
logique temporelle (CTL, TCTL, …)

Processus de model-checking

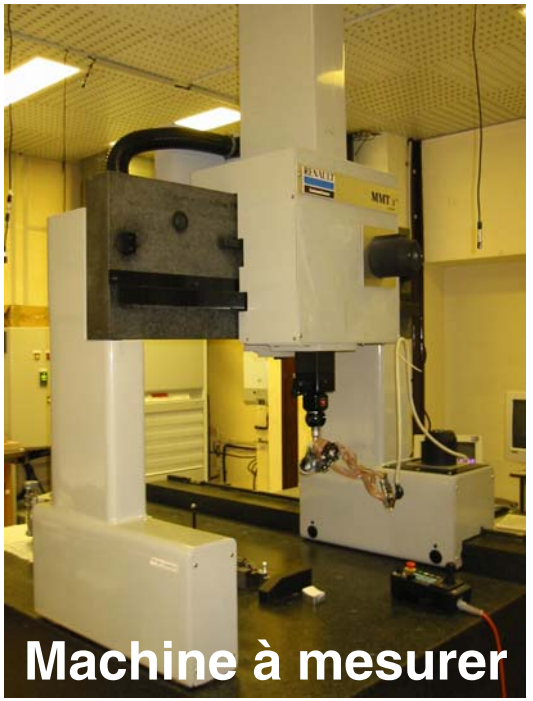
Propriétés vérifiées ou non (et diagnostic)

Exemple 2 : Evaluation des performances temporelles d'architectures de commande via Internet

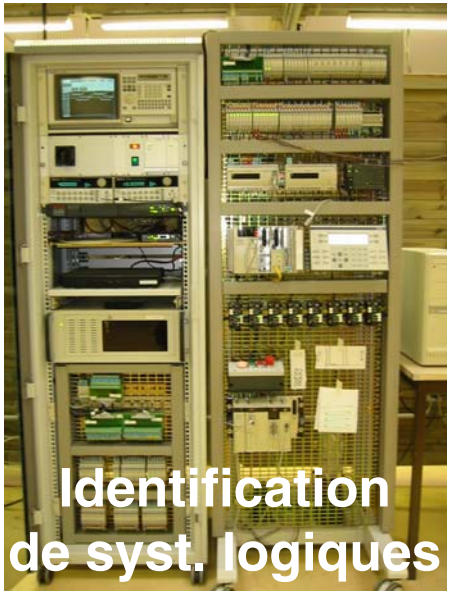





Machines d'usinage


Machine à mesurer


Ensemble de la
plate-forme expérimentale


Identification
de syst. logiques


Commande
via Internet

Contribution de ISA au M2R2

- **Evaluation des SED (R2-1)**

Jean-Jacques Lesage (lesage@lurpa.ens-cachan.fr)

- **Systèmes discrets sûrs de fonctionnement (R2-9)**

Jean-Marc Faure (faure@lurpa.ens-cachan.fr)

En collaboration avec :

Bruno Denis
Olivier De Smet
Jean-Marc Roussel (roussel@lurpa.ens-cachan.fr)
Ryad Zémouri

Les SED dans le MASTER IS EEAPR

- **M1-S1** Bases des SED

Les modèles des SED (automates à états finis, algèbres, RdP, Grafcet, Statecharts, …)

- **M1-S2** Ingénierie des SED

Les méthodes de conception, synthèse, approches multi modèles, intégration, …

- **M2-S3** Validation des SED

Vérification/Validation et Evaluation de performances

Pour 2006 - 2007 :

Aucun étudiant n'a suivi le M1 de ce Master → adaptation du programme (un peu de modèles, un peu de construction de modèles, … dans le but de faire un peu de vérification et d'évaluation de performances)

Master Ingénierie Système en EEAPR Spécialité R2
« Productique et Réseaux »

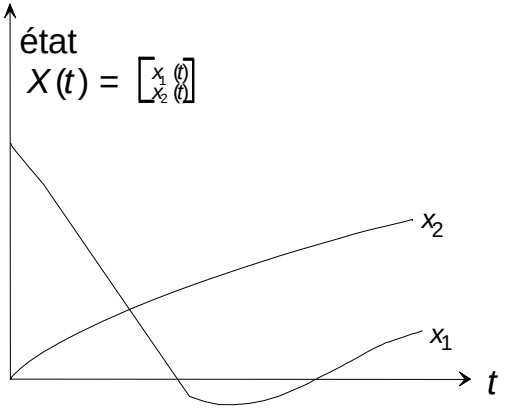
UE Ossature R2-1

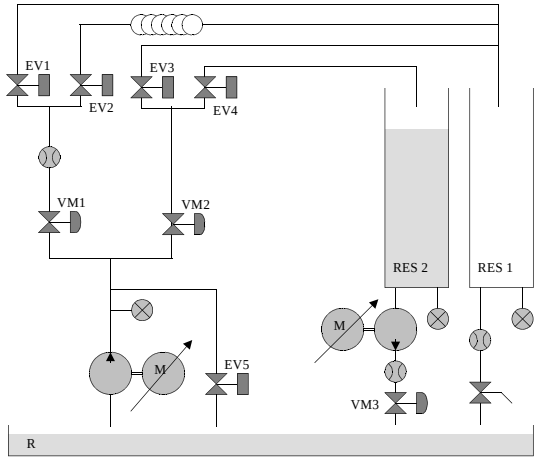
Partie 1 :

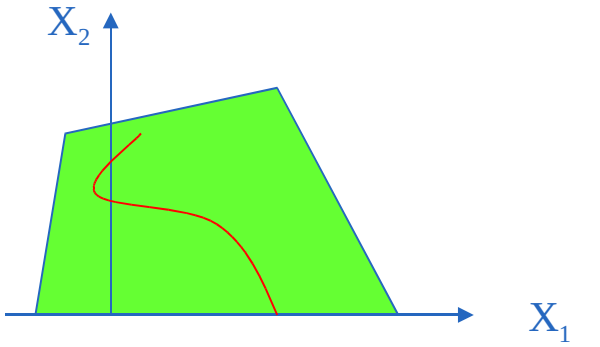
« Base des SED
Langages et Automates »

Notion d'espace d'état

Soit un système *continu* dont l'état est caractérisé à chaque instant par deux *variables à évolution continue dans le temps* (par exemple X_1 est une température et X_2 un débit)





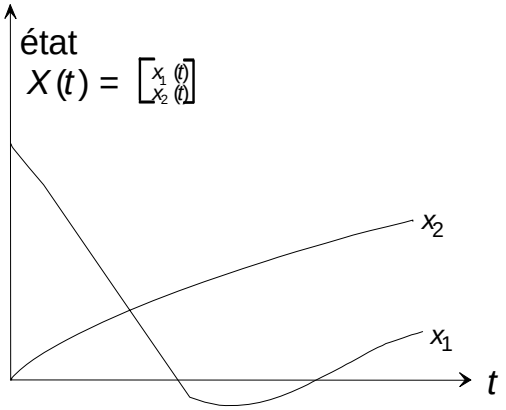


L'*espace d'états* de ce système est le continuum formé de l'ensemble des valeurs de X_1 et X_2.

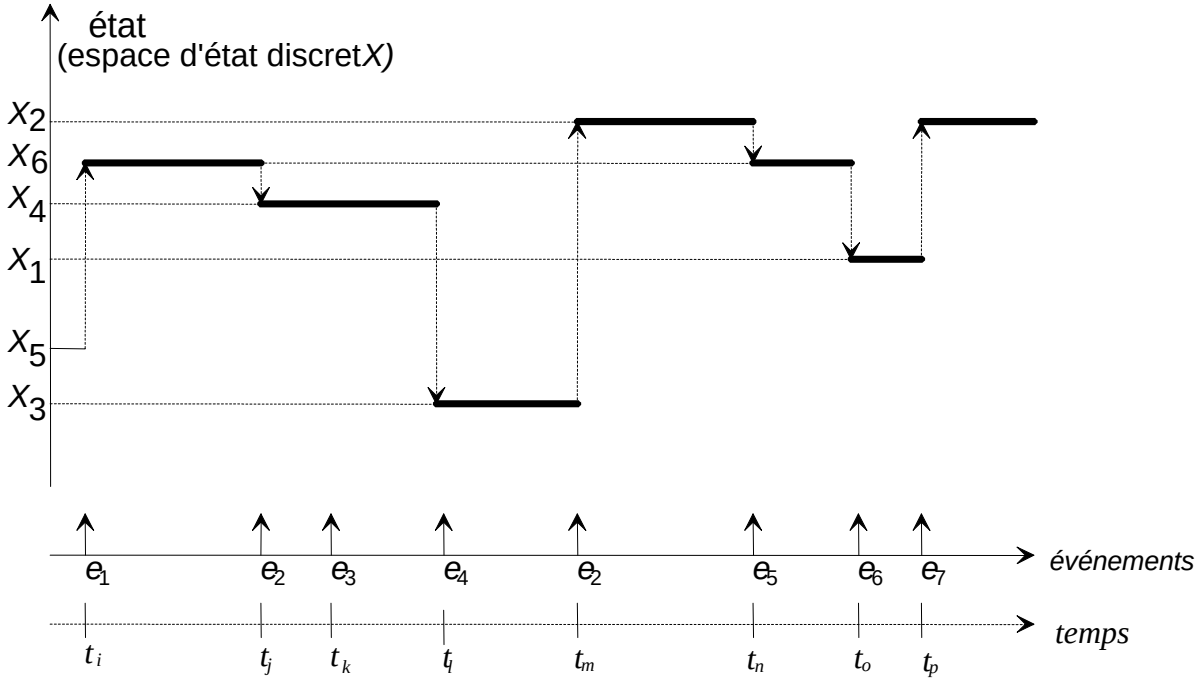
Une *trajectoire* d'état possible de ce système en fonctionnement dans l'espace d'états est donnée ci-dessus

Les SED (1)

Un Système à Evénements Discrets est un *système* à *espace d'état discret* dont les transitions entre états sont associées à l'occurrence d'*événements* discrets *asynchrones*.



Système continu



$X = \{ X_1, X_2, X_3, X_4, X_5, X_6 \}$ ensemble (discret) d'états = espace d'états

$\Sigma = \{ e_1, e_2, e_3, e_4, e_5, e_6, e_7 \}$ ensemble d'événements discret

Les SED (2)

Système

ensemble de composants en interaction accomplissant conjointement une fonction particulière en respectant un ensemble de contraintes

« a combination of components that act together to perform a function not possible with any of the infividual parts » (IEEE standard dictionary of Electrical and Electronic terms) (conseil : pour ce type de définitions consulter http://www.Ifac-control.org ou http://www.ieee.org)

Evénement

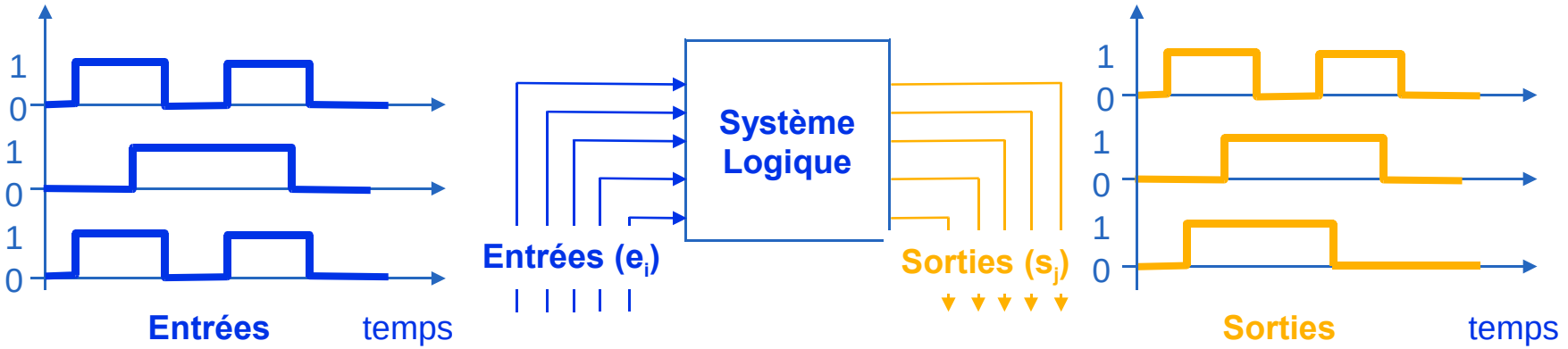
changement de valeur d ’une variable survenant à une date donnée. Pour distinguer n changements de valeur identiques de la même variable, on appelle *occurrence de l ’événement* chaque variation de cette variable à une date d_i

Asynchrone

non cadencé par une horloge

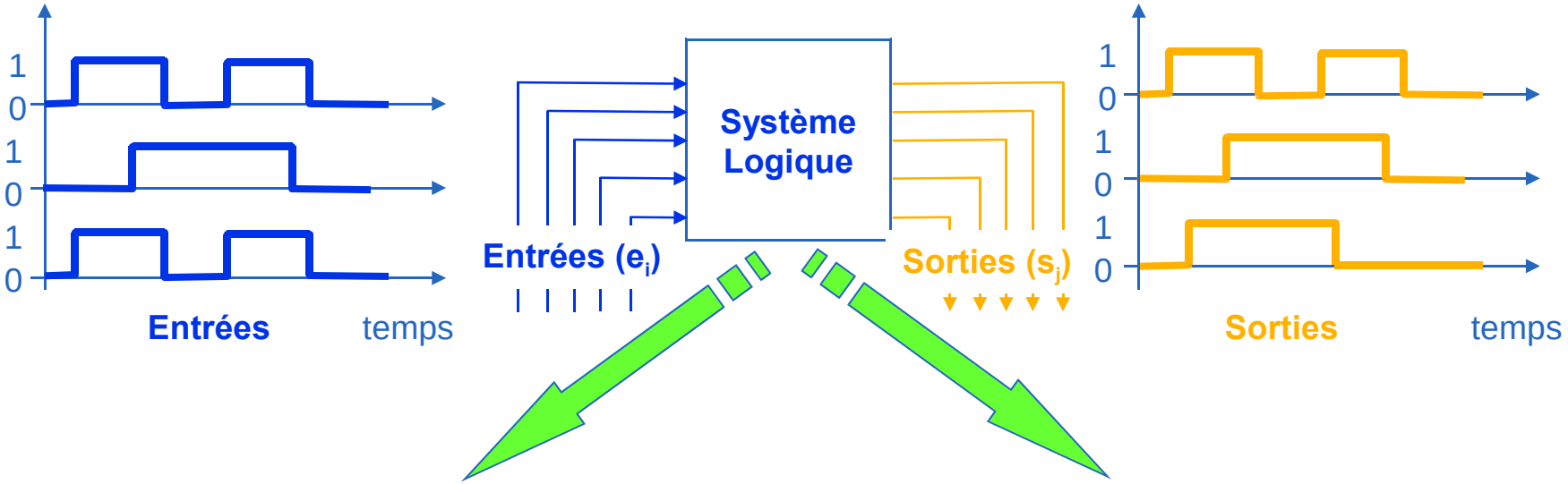
Les Systèmes Logiques

Un Système logique est une sous-classe de SED dont les *entrées et sorties* sont toutes des variables *binaires* (deux états).



La tâche de l'automaticien est de déterminer la *Loi de Commande* qui traduit le comportement attendu du SED (élaboration des évolutions des s_j ,causales des évolutions des e_i).

Comportement des SL : Combinatoire / Séquentiel



Systèmes logiques *combinatoires*

$$\begin{cases} s_1(t) = f_1\,(e_1(t), \ldots, e_n(t)) \\ \vdots \\ s_p(t) = f_p\,(e_1(t), \ldots, e_n(t)) \end{cases}$$

Vrai à chaque instant, donc on peut omettre le temps dans les équations

Systèmes logiques *séquentiels*

$$\begin{cases} s_1(t) = f_1\,(e_1(t), \ldots, e_n(t), Q(t\text{-}1)) \\ \vdots \\ s_p(t) = f_p\,(e_1(t), \ldots, e_n(t), Q(t\text{-}1)) \end{cases}$$

Q(t) est l'*état* du SED

Les SL combinatoire

$$\begin{cases} s_1 = f_1(e_1, \ldots, e_n) \\ \vdots \\ s_p = f_p(e_1, \ldots, e_n) \end{cases}$$

$$\begin{cases} \forall_i, s_i \in \{0,1\} \\ \forall_j, e_j \in \{0,1\} \end{cases}$$

L'*algèbre de Boole* est la base mathématique de modélisation des systèmes combinatoires

Les f_i sont des fonctions combinant les e_j grâce aux opérateurs *ET*, *OU* et *NON* de l'algèbre de Boole.

Outils :

- tables de vérité
- tableaux de karnaugh
- rares méthodes de synthèse

Les SL séquentiels (1)

$$\begin{cases} s_1(t) = g_1\,(e_1(t), \ldots, e_n(t), Q(t\text{-}1)) \\ \quad\vdots \\ s_p(t) = g_p\,(e_1(t), \ldots, e_n(t), Q(t\text{-}1)) \end{cases}$$

Pb : caractériser et modéliser g_i

$$\begin{cases} \forall_i, \forall t,\ s_i(t) \in \{0,1\} \\ \forall_j, \forall t,\ e_j(t) \in \{0,1\} \end{cases}$$

Q (t) est à chaque instant l'*état* du système

Ce système peut se mettre sous la forme :

$$\begin{cases} Q(t) = \varphi\,(e_1(t), \ldots, e_n(t), Q(t\text{-}1)) \\ s_1(t) = f_1\,(e_1(t), \ldots, e_n(t), Q(t)) \\ \quad\vdots \\ s_p(t) = f_p\,(e_1(t), \ldots, e_n(t), Q(t)) \end{cases}$$

Pb : caractériser et modéliser φ et f_i

Les SL séquentiels (2)

$$\begin{cases} Q(t) = \varphi\,(e_1(t), \ldots, e_n(t), Q(t\text{-}1)) \\ s_1(t) = f_1\,(e_1(t), \ldots, e_n(t), Q(t)) \\ \vdots \\ s_p(t) = f_p\,(e_1(t), \ldots, e_n(t), Q(t)) \end{cases}$$

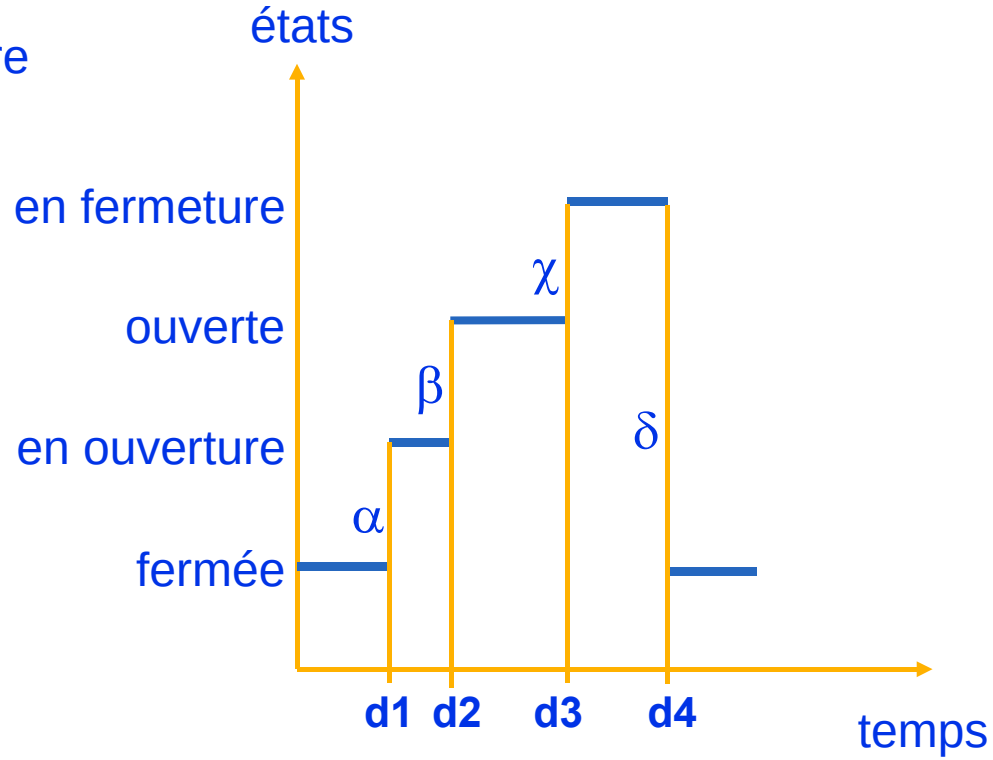
Outils pour exprimer φ :

- équations récurrentes, possibles dans des cas particuliers seulement
- *modèles à états finis*

Un exemple basique (1)

Soit le système “vanne” qui peut être décrit par 4 états : “fermée”, “en ouverture”, “ouverte” et “en fermeture”. Une évolution possible de ce SED (ou trajectoire d’états) est représentée par le chronogramme ci contre.

α, β, χ, et δ sont les quatre événements qui permettent l’évolution de ce SED dans son espace de quatre états possibles.



α : événement « début d ’ouverture »
β : événement « fin d ’ouverture »
χ : événement « début de fermeture »
δ : événement « fin de fermeture »

Un exemple basique (2)

α est l 'événement qui modélise la transition entre l 'état « fermée » et l 'état « en ouverture ».

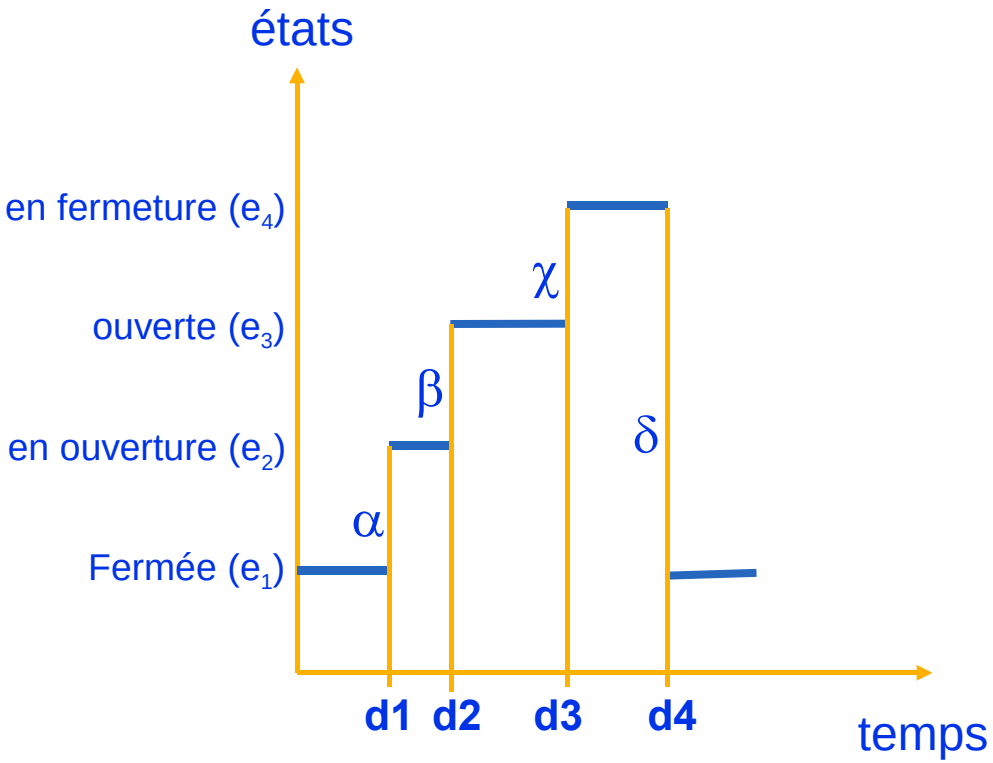
On peut donner deux représentations du comportement de la vanne à partir de son état initial supposé être « fermée » (on parlera également de *trajectoire* du système « vanne ») :

un ***modèle logique*** donné basiquement par la séquence d 'événements

$\{e_1, \alpha \rightarrow e_2, \beta \rightarrow e_3, \chi \rightarrow e_4, \delta \rightarrow e_2\}$

un ***modèle temporel*** donné par la suite ordonnée des couples

$\{e_1, (\alpha, d1) \rightarrow e_2, \ldots \rightarrow e_2\}$

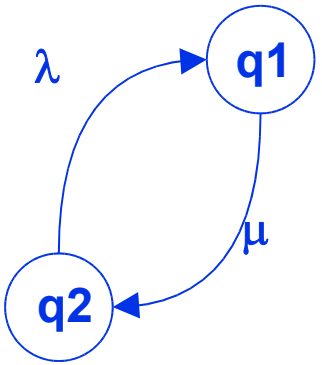


Dans les deux cas, le comportement modélisé est *déterministe*, c'est-à-dire que l'état successeur d'un couple (état, événement) est unique.

Un exemple basique (3)

Tous les états retenus précédemment sont des états de « bon fonctionnement».

Si on s'intéresse maintenant à la représentation de la capacité de la vanne à accomplir sa fonction, on retient un modèle *stochastique*.



q2 : état de bon fonctionnement

q1 : état de panne

λ : taux (probabilité) de panne

μ : taux (probabilité) de remise en marche

En résumé :

La modélisation du comportement des SED nécessite d'utiliser des modèles :

- *Logiques* (*déterministes* ou non)
- *Temporisés*
- *Stochastiques*

Bases de la théorie des langages et des automates (1)

Rappel : notion de modèle

- approximation, vue partielle plus ou moins abstraite de la réalité afin d ’appréhender un système plus simplement. Un modèle est établi selon un point de vue donné pour un objectif donné. Donc, un modèle est partiel et subjectif.

Alphabet

- ensemble fini de symboles. Dans le cas d ’un SED, un alphabet pourra par exemple représenter l ’ensemble des événements d ’entrée possibles d ’un système. On note par exemple $\Sigma = \{a,b\}$

Chaîne ou Mot ou Séquence

- défini sur un alphabet Σ c’est une suite finie d ’éléments de Σ. Par exemple abba est un mot défini sur Σ.

Bases de la théorie des langages et des automates (2)

- la longueur d 'un mot, notée ImotI, représente le nombre de symboles de ce mot (par exemple, Iabbal = 4)
- soit un alphabet Σ. On note Σ^n (où n est un entier naturel) l 'ensemble des mots de longueur n.
  Par exemple, sur $\Sigma = \{a,b\}$: $\Sigma^0 = \{\varepsilon\}$ où ε est le mot vide de symbole
  $\Sigma^1 = \{a,b\}$, $\Sigma^2 = \{aa,bb,ab,ba\}$
- Σ^* est l 'ensemble de tous les mots que l 'on peut construire sur Σ.

$$\Sigma^* = U_{(i>=0)}\ \Sigma^i$$

- dans notre exemple, $\Sigma^* = \{\varepsilon, a, b, aa, bb, ab, ba, aaa, aab, abb, \ldots\}$

Langage

- un langage défini sur un alphabet Σ est un sous ensemble de Σ^* . Par exemple, l 'ensemble des séquences telles que a apparaît en premier et telles que a et b apparaissent alternativement est décrit par le langage
  L = $\{\varepsilon, a, ab, aba, abab, ababa, \ldots\}$. C 'est un langage infini.

Bases de la théorie des langages et des automates (3)

- plusieurs opérations ensemblistes sont définies sur les langages :

l '*union* de L_1 et L_2 est notée $L_1 \cup L_2 = \{w | w \in L_1 \text{ ou } w \in L_2\}$

L '*intersection* de L_1 et L_2 est notée $L_1 \cap L_2 = \{w | w \in L_1 \text{ et } w \in L_2\}$

La *concaténation* de L_1 et L_2 est notée $L_1 . L_2 = \{w | w = xy, x \in L_1 \text{ et } y \in L_2\}$

Automate

- un automate est une machine à états qui permet de traduire la relation causale entrées/sorties d 'un SED en utilisant les bases de la théorie des langages.

Bases de la théorie des langages et des automates (4)

Automate

- un automate est un quintuplet : $A = (Q, \Sigma, \delta, q_0, F)$ où :

  Q est l 'ensemble des états

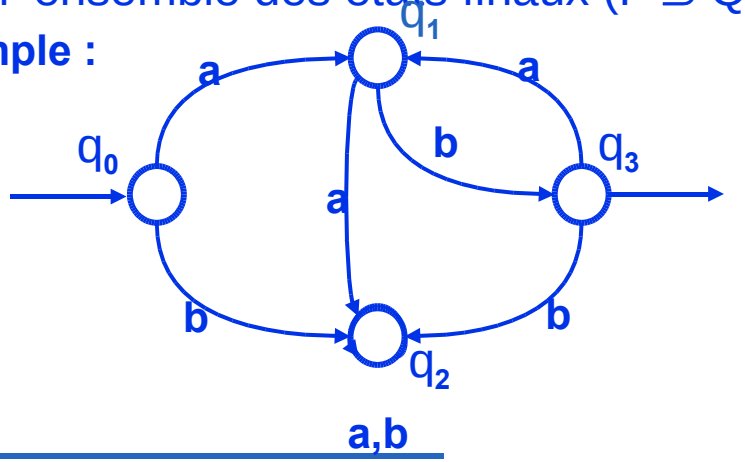
  Σ est un alphabet d 'entrée

  δ est la fonction de transition d 'états définie de $Q \times \Sigma \longrightarrow Q$ qui associe un état de départ et un symbole d 'entrée à un état d 'arrivée

  q_0 est l 'état initial

  F est l 'ensemble des états finaux ($F \supset Q$)

Exemple :



Bases de la théorie des langages et des automates (4)

Automate

- un automate est un quintuplet : $A = (Q, \Sigma, \delta, q_0, F)$ où :

  Q est l 'ensemble des états

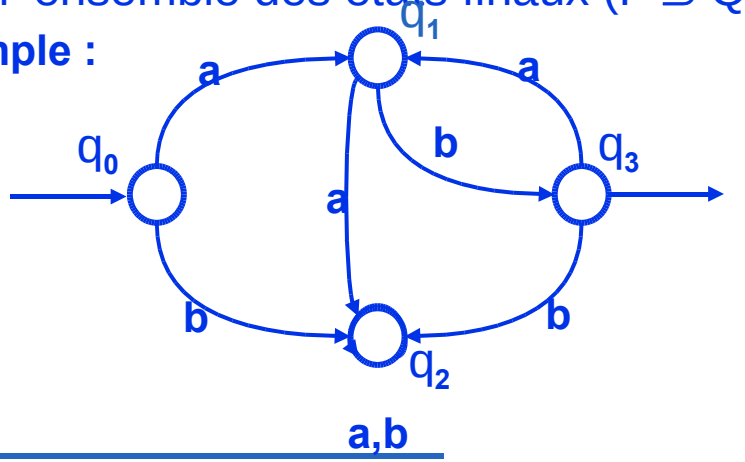
  Σ est un alphabet d 'entrée

  δ est la fonction de transition d 'états définie de $Q \times \Sigma \longrightarrow Q$ qui associe un état de départ et un symbole d 'entrée à un état d 'arrivée

  q_0 est l 'état initial

  F est l 'ensemble des états finaux ($F \supset Q$)

Exemple :



$Q = \{q_0, q_1, q_2, q_3\}$

$\Sigma = \{a,b\}$

$F = \{q_3\}$

δ est représentée par les arcs associés à des symboles de l 'alphabet

$\delta(q_0, a) = q_1$

.....

Bases de la théorie des langages et des automates (5)

Déterminisme

- cet automate est dit *déterministe* dans le sens où depuis tout état, il n 'existe pas deux transitions qui soient associées à un même symbole qui conduiraient à des états différents. Un automate déterministe ne doit donc posséder qu'un seul état initial et ne se trouve à tout instant que dans un seul état

Complétude

- un automate est dit complet lorsque $\forall q \in Q$, $\forall \sigma \in \Sigma$, $\exists q \xrightarrow{\sigma} q'$

Langage accepté par un automate

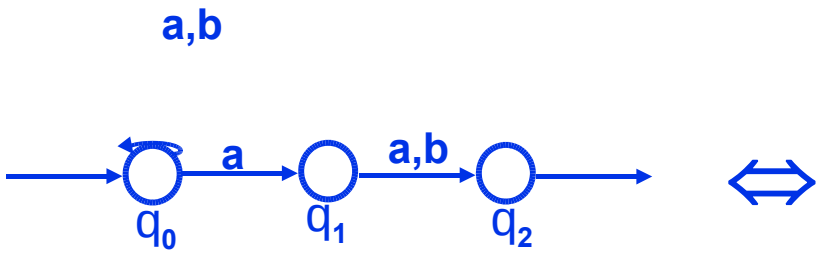
- la séquence ab est reconnue (ou acceptée) par l 'automate précédent ($\exists$ trajectoire d 'états depuis l 'entrée vers la sortie). La séquence aba ne l 'est pas.
- L 'ensemble des séquences acceptées par un automate est le langage accepté.

Bases de la théorie des langages et des automates (6)

Propriété

- on peut transformer tout automate *indéterministe* en automate *déterministe* équivalent (qui reconnaît le même langage). La contre-partie est l ’augmentation du nombre d ’états.

Automate non déterministe Automate déterministe équivalent

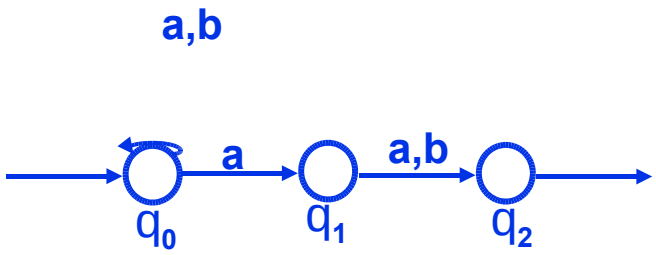


Bases de la théorie des langages et des automates (6)

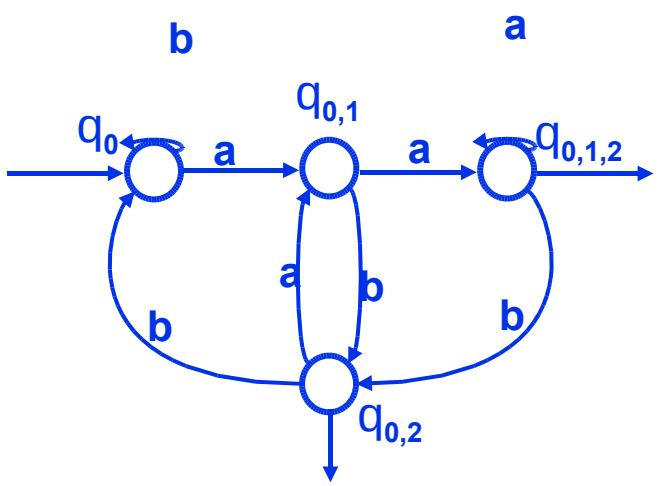
Propriété

- on peut transformer tout automate *indéterministe* en automate *déterministe* équivalent (qui reconnaît le même langage). La contre-partie est l 'augmentation du nombre d 'états.

Automate non déterministe



Automate déterministe équivalent



Bases de la théorie des langages et des automates (7)

Machine de Moore

- c 'est un automate déterministe dans lequel une sortie peut prendre des valeurs (associées aux états) dans un ensemble fini de grandeurs discrètes. La sortie est émise à la fin de la séquence d 'entrée.

- une machine de Moore est un 6-uplet M = (Q, Σ, δ, q_0, Γ, λ) où :

  Q, Σ, δ, q_0 ont la même signification que pour un automate fini déterministe

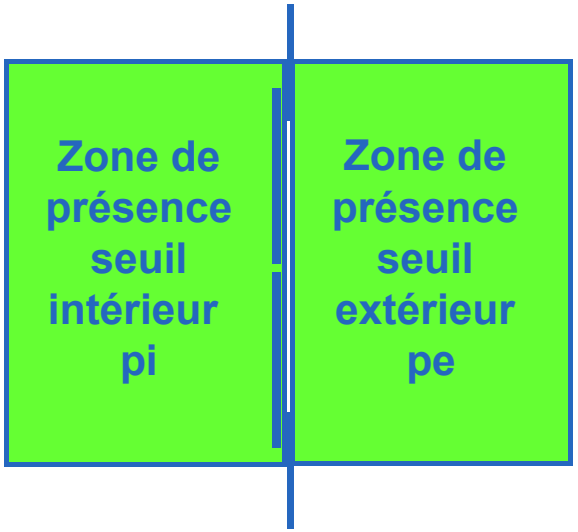
  Γ est un alphabet fini de sortie

  λ est la fonction d 'affectation de sortie λ : Q → Γ

Modélisation de SED à l ’aide d ’automates (1)

Exemple 1 : ouverture automatique d ’une porte

- on considère une porte coulissante (type grand magasin) dont on désire commander l ’ouverture et la fermeture en fonction de deux informations détectant la présence de personnes au seuil intérieur ou extérieur de la porte (détectée par des capteurs optiques ou des tapis sensitifs).

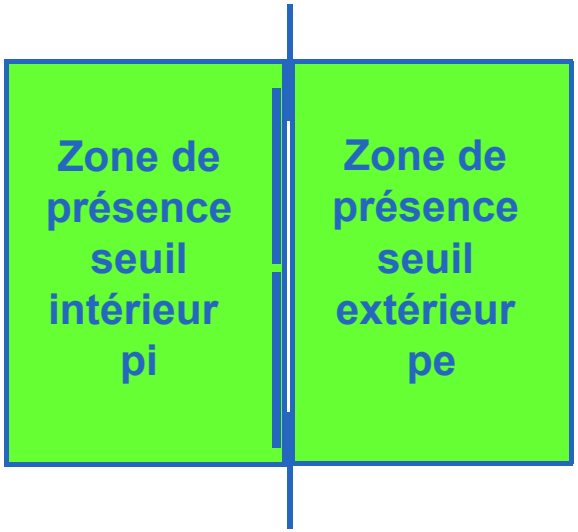


On considérera donc deux sorties à commander : OUV et FER en fonction d ’occurrences d ’événements sur deux entrées : pe et pi.
La porte est supposée être initialement fermée

Modélisation de SED à l ’aide d ’automates (2)

Exemple 1 : ouverture automatique d ’une porte

- séquences d ’entrée possibles
  une personne seule sort : ↑pi ↓pi ↑pe ↓pe
  une personne seule entre : ↑pe ↓pe ↑pi ↓pi
  remarque : à la sortie, ↓pi ↑pe peuvent être simultanés. Dans tous les cas, l ’ouverture doit se faire sur occurrence de ↑pi et la fermeture sur occurrence de ↓pe. Cette séquence peut se réduire à↑pi ↓pe
  (idem pour l ’entrée : ↑pe ↓pi)



- conditions de sécurité :
  si une personne est entrée (↑pe), la fermeture ne devra se faire que si : ↓pi ./pe./pi (soit ↓pi ./pe)
  dans le cas d ’une sortie, la fermeture doit se faire ssi ↓pe./pi

Modélisation de SED à l ’aide d ’automates (3)

Exemple 1 : ouverture automatique d ’une porte

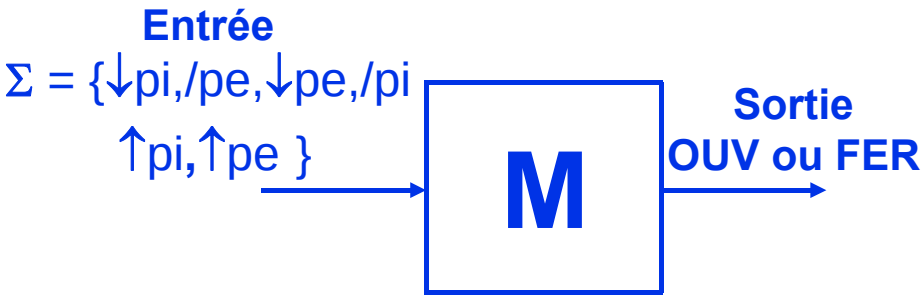
- ↓pi ./pe + ↓pe ./pi est une *expression régulière* dont les opérandes sont des symboles d ’un alphabet (Σ = {↓pi, /pe, ↓pe, /pi}).

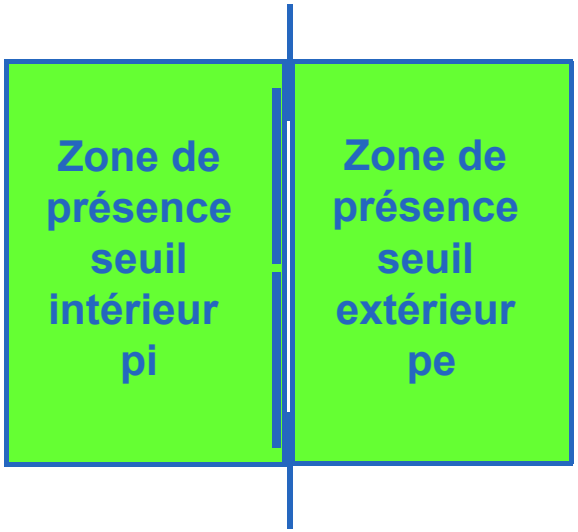
  Si chaque symbole est considéré comme un langage sur Σ (L_1 = {↓pi}, L_2 = {↓pe}, …) les opérateurs +, . sont interprétés comme les opérateurs d ’union et de concaténation sur des langages de Σ. Une expression régulière définit donc un langage sur Σ.

- Par soucis de simplification, on notera les expressions régulières sous leur forme « fonctionnelle » avec opérandes et opérateurs.

Modélisation de SED à l ’aide d ’automates (4)

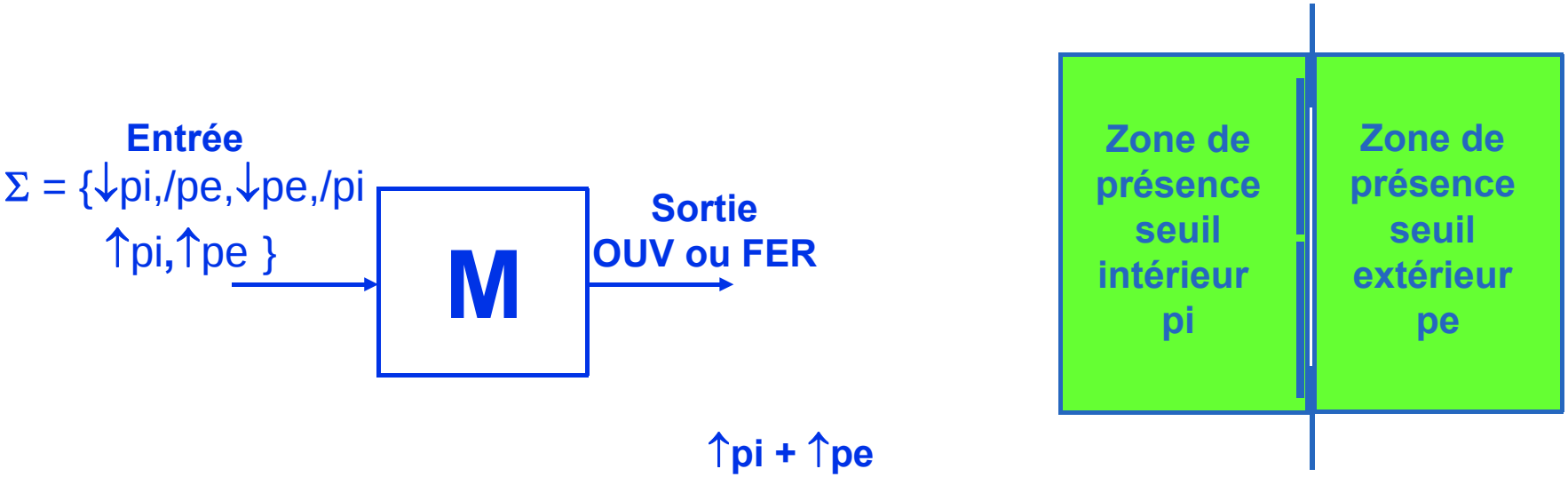
Exemple 1 : ouverture automatique d ’une porte





Modélisation de SED à l ’aide d ’automates (4)

Exemple 1 : ouverture automatique d ’une porte



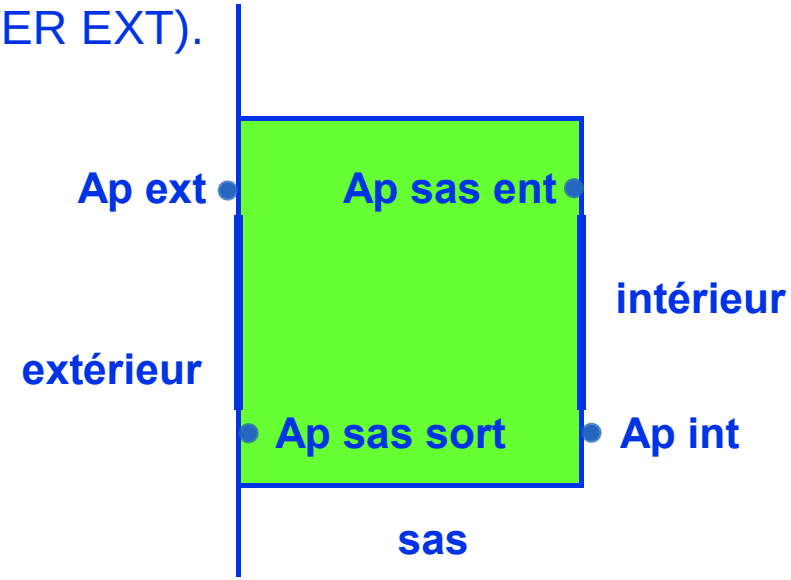
Modélisation de SED à l ’aide d ’automates (5)

Exemple 2 : sas de sécurité d ’une banque

- On considère un sas de sécurité de banque dont on veut gérer le verrouillage et le déverrouillage des portes intérieures et extérieures (VER INT, VER EXT, DEVER INT et DEVER EXT). La manœuvre des portes est manuelle.

- Les clients demandent le déverrouillage des portes par des boutons poussoirs (Ap int, Ap ext, …)

- Deux capteurs de fermeture de porte (int fer, ext fer) permettent le verrouillage.



Modélisation de SED à l ’aide d ’automates (6)

Exemple 2 : sas de sécurité d ’une banque

- pb : trouver les états pertinents pour la commande
- hypothèses : pas d ’état sans sortie affectée
  pas deux états affectant les mêmes sorties

  les états recherchés sont donc tous pertinents par rapport aux sorties
- on envisage la combinatoire des sorties et on cherche les états pertinents

DEVER INT.DEVER EXT

VER INT.VER EXT

	00	01	11	10
00				
01				
11				
10				

Modélisation de SED à l ’aide d ’automates (6)

Exemple 2 : sas de sécurité d ’une banque

DEVER INT.DEVER EXT

VER INT.VER EXT

	00	01	11	10
00				
01				X
11	X			
10		X		

Il ne reste que trois états pertinents, les autres étant incohérents par rapport aux cahier des charges (les deux portes déverrouillées par ex.) ou par rapport au système opérant (une porte à la fois verrouillée et déverrouillée par ex.).

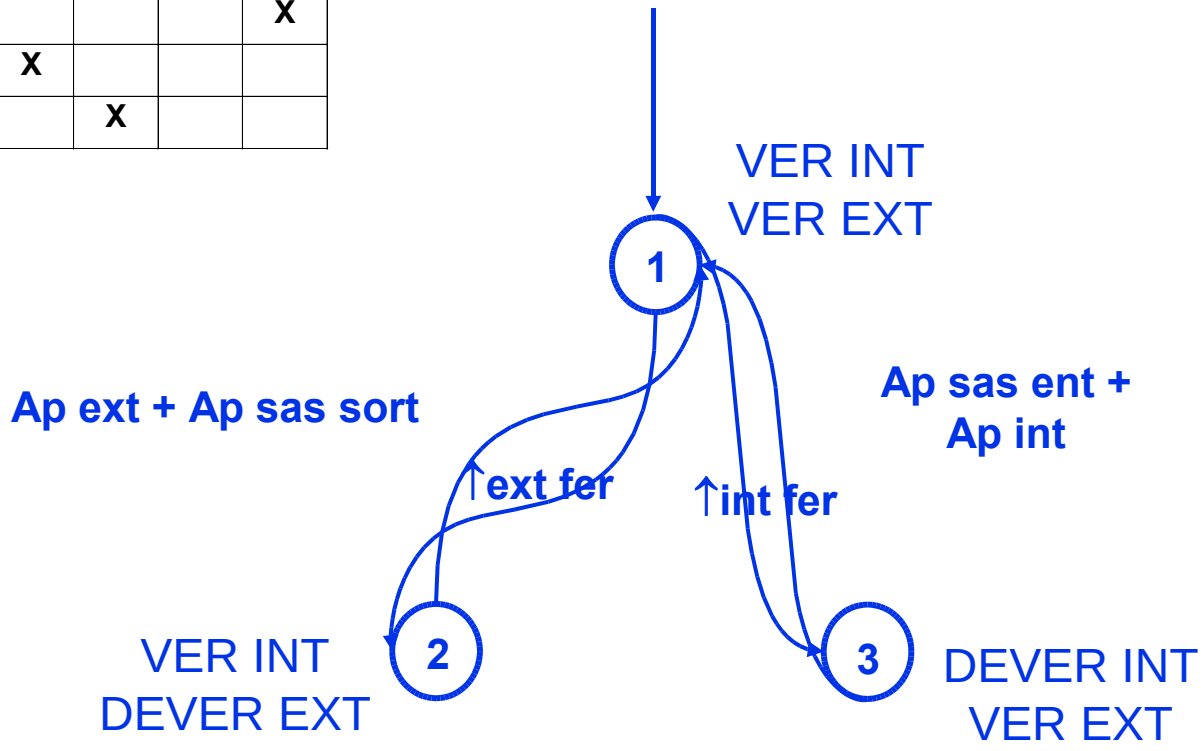
Modélisation de SED à l ’aide d ’automates (7)

VER INT.VER EXT \ DEVER INT.DEVER EXT	00	01	11	10
00				
01				X
11	X			
10		X		

Modélisation de SED à l ’aide d ’automates (7)

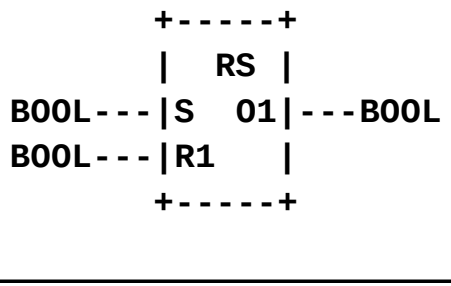
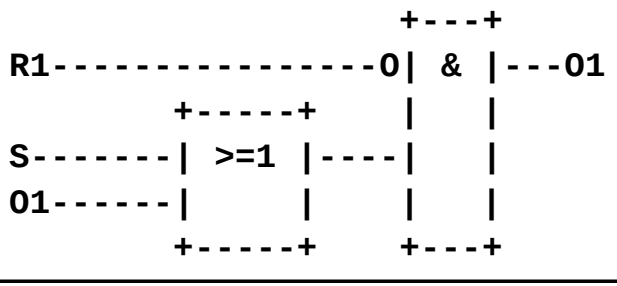
DEVER INT.DEVER EXT

VER INT.VER EXT	00	01	11	10
00				
01				X
11	X			
10		X		



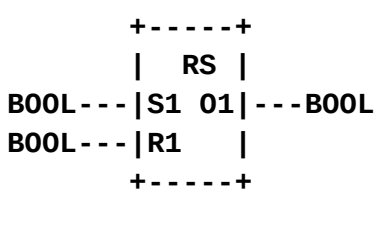
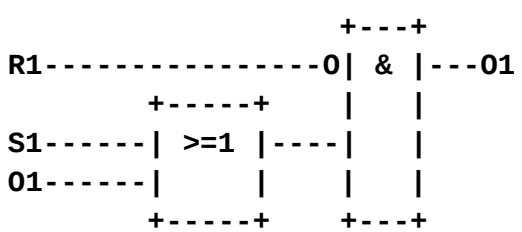
Exemple 3 : application à la vérification de propriété (1)

Définition normalisée du bloc fonctionnel « RS » (IEC 61131-3) :

No.	Graphical form	Function block body
2	**Bistable Function Block (reset dominant)**	
		
The initial state of the output variable 01 shall be the normal default value of zero for Boolean variables.		

le comportement de ce SL est il *combinatoire* ou *séquentiel* ?

Modélisation du bloc fonctionnel « RS » (2)

No.	Graphical form	Function block body
2	Bistable Function Block (reset dominant)	
		
The initial state of the output variable Q1 shall be the normal default value of zero for Boolean variables.		

Exemple : bloc fonctionnel « RS » (2)

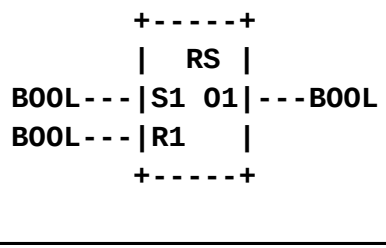
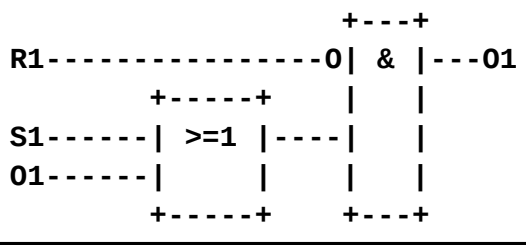
No.	Graphical form	Function block body
2	Bistable Function Block (reset dominant)	
		
The initial state of the output variable Q1 shall be the normal default value of zero for Boolean variables.		

Table de vérité :

R_1	S_1	$O_{1(t)}$
0	0	$O_{1(t-1)}$
0	1	1
1	1	0
1	0	0

OU

R_1	S_1	$O_{1(t-1)}$	$O_{1(t)}$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

le comportement de ce SL est *séquentiel*

Exemple : bloc fonctionnel « RS » (3)

R_1	S_1	$O_{1(t)}$
0	0	$O_{1(t-1)}$
0	1	1
1	1	0
1	0	0

Exemple : bloc fonctionnel « RS » (3)

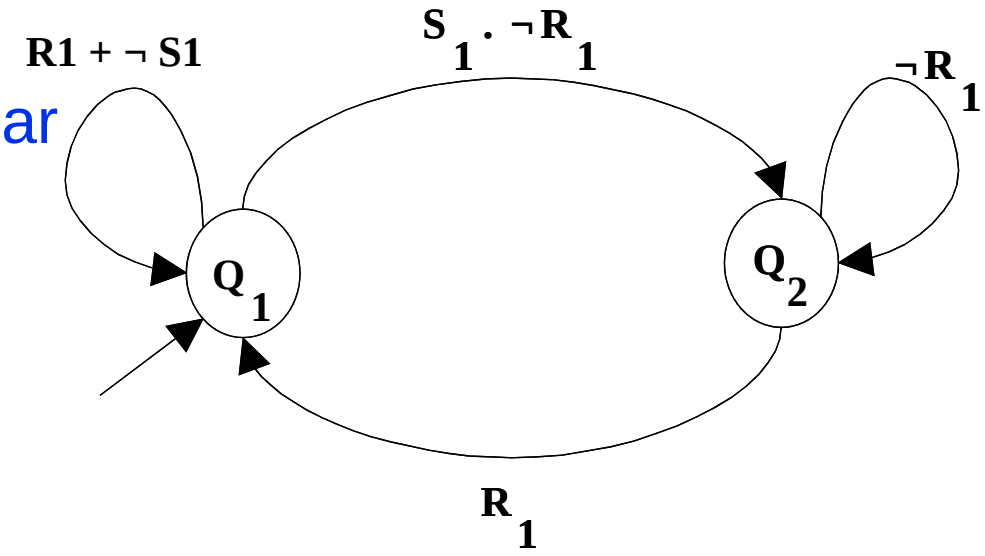
R_1	S_1	$O_{1(t)}$
0	0	$O_{1(t-1)}$
0	1	1
1	1	0
1	0	0

La fonction φ est donnée par un automate à deux états

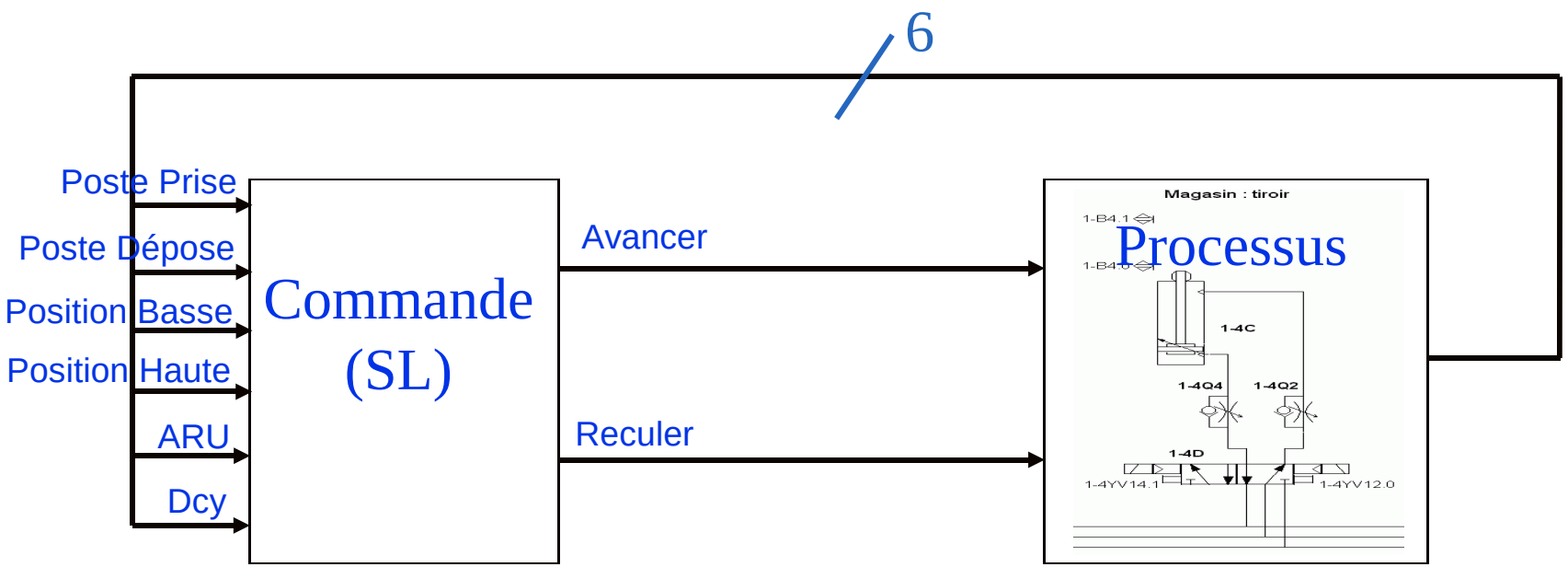
L'équation de la sortie est :

$O_1 = Q_2$

(ou $\neg O_1 = Q_1$)

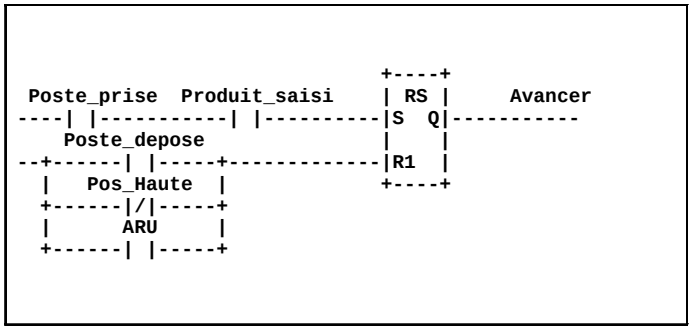


Système : {composants en interaction}
Modélisation modulaire (1)



La complexité des systèmes réels impose d'utiliser des approches de modélisation modulaires

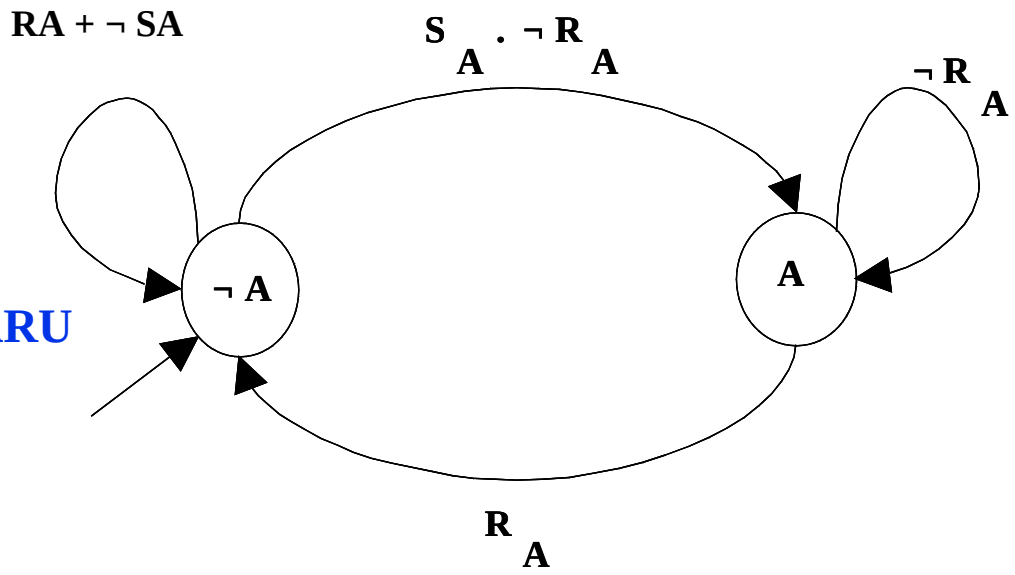
Modélisation modulaire (2)



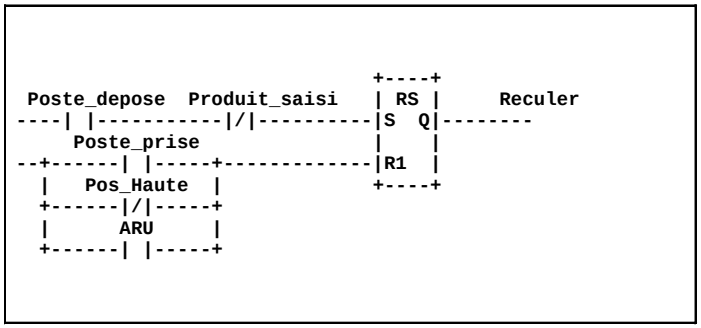
Avec :

S_A = Poste_prise . Produit_saisi

R_A = Poste_depose + ¬Pos_haute + ARU



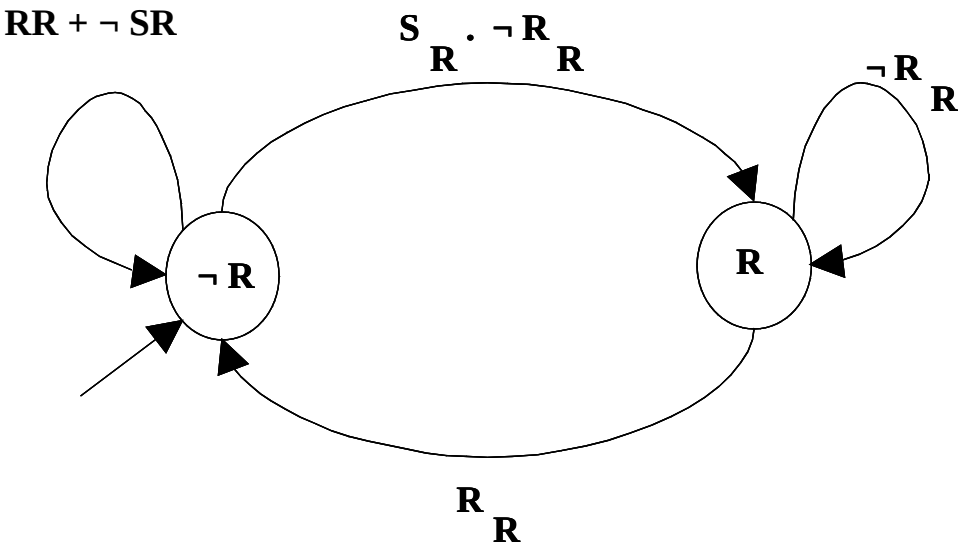
Modélisation modulaire (3)



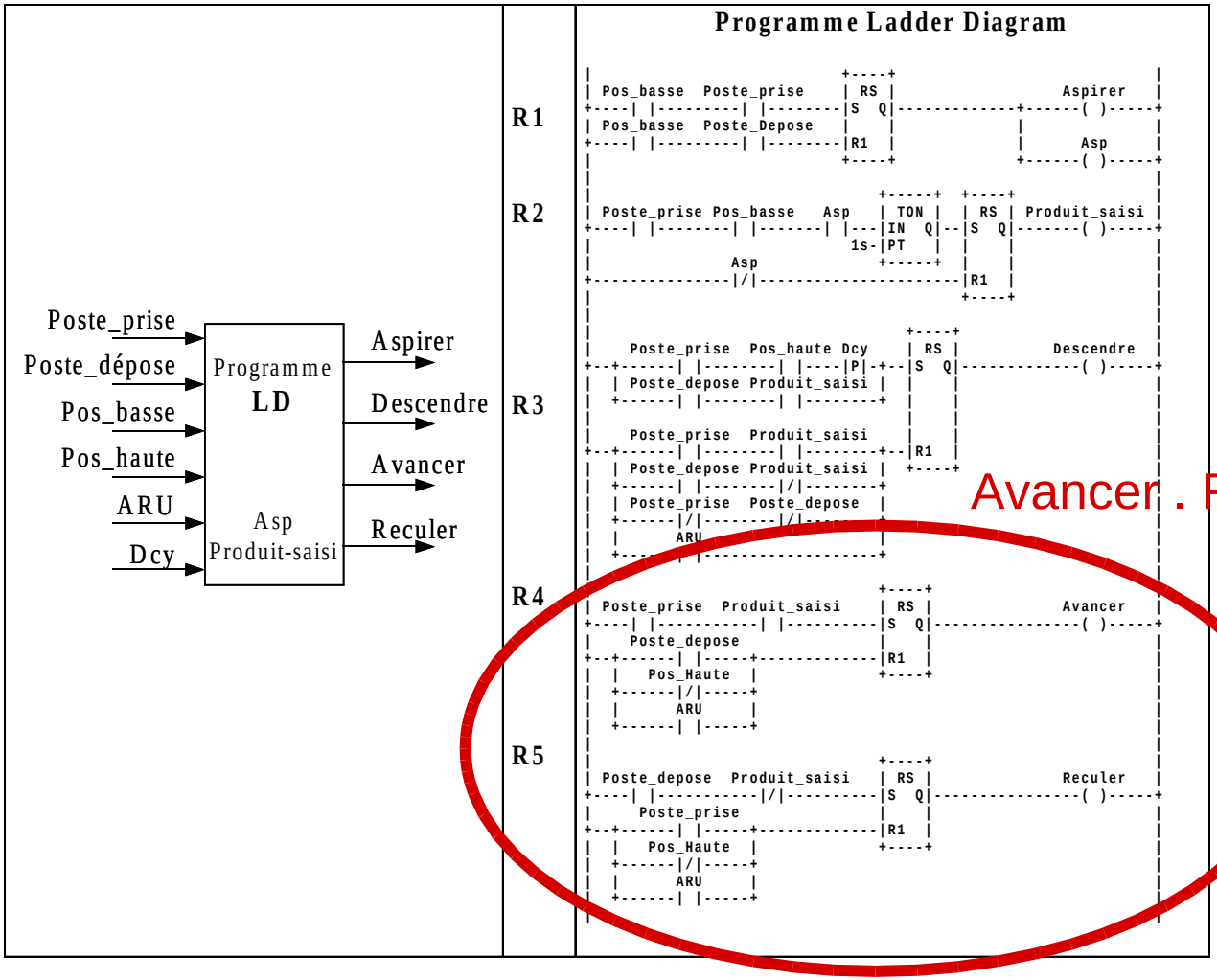
Avec :

S_R = Poste_depose . ¬Produit_saisi

R_R = Poste_prise + ¬Pos_haute + ARU

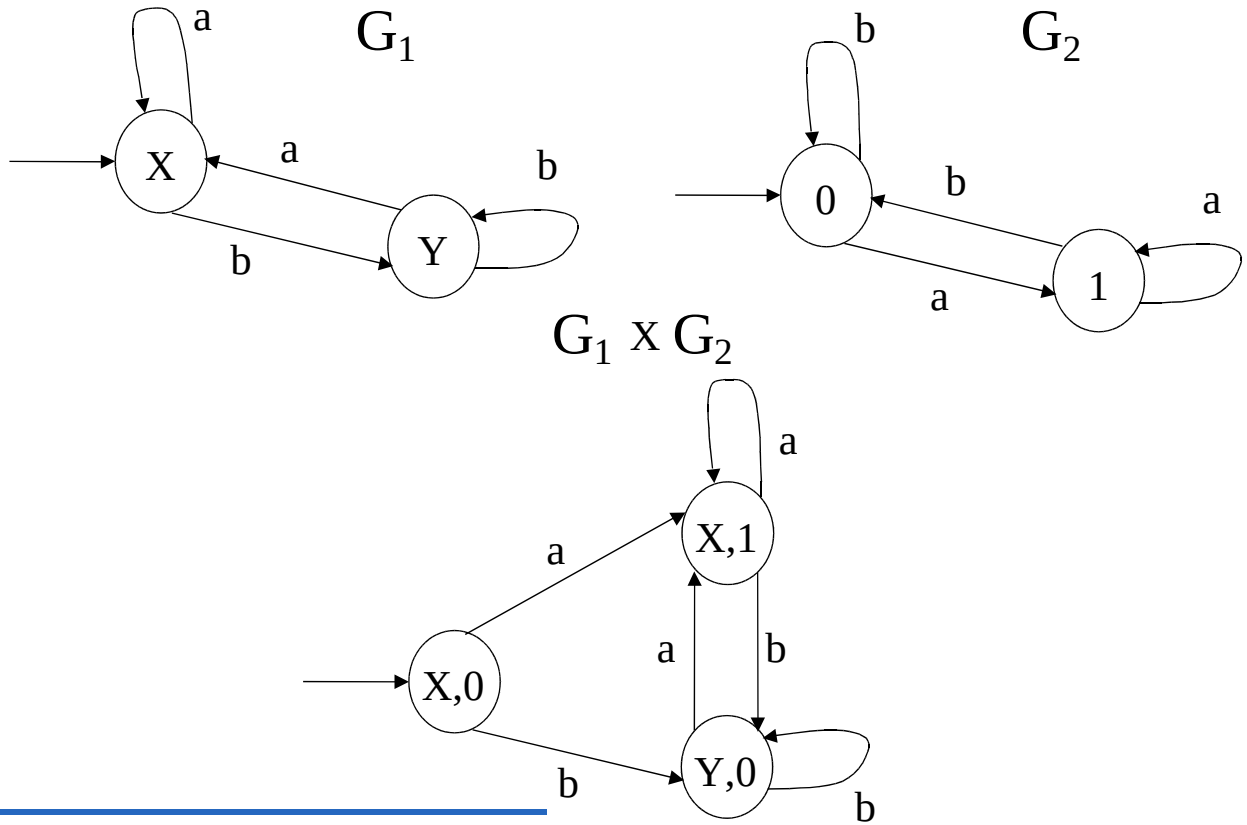


Application à la validation de modèle de comportement ou de programme

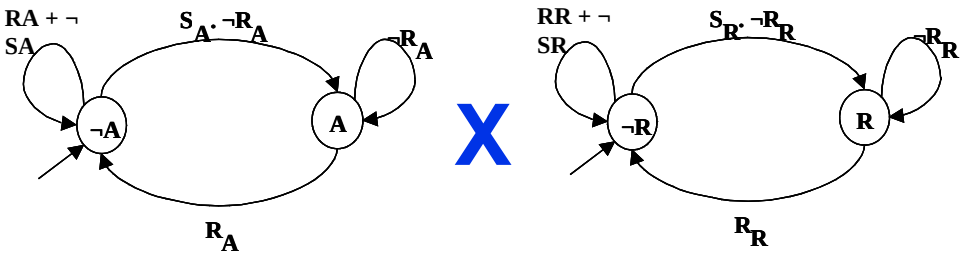


Modélisation modulaire et comportement global

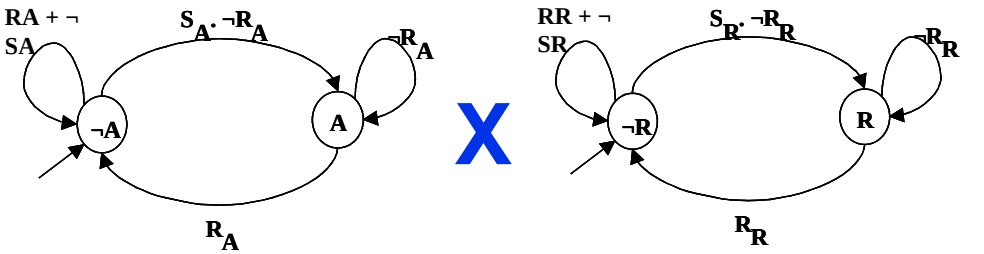
Le comportement de l'ensemble du système (de ses deux composants en parallèle) est obtenu par composition des deux comportements élémentaires



Modélisation modulaire (5)



Modélisation modulaire (6)



$(RA + \neg S\ A) . (RR + \neg S\ R)$ $R_A . S_R . \neg R_R$ $(RA + \neg S\ A) . \neg R_R$

Avec :

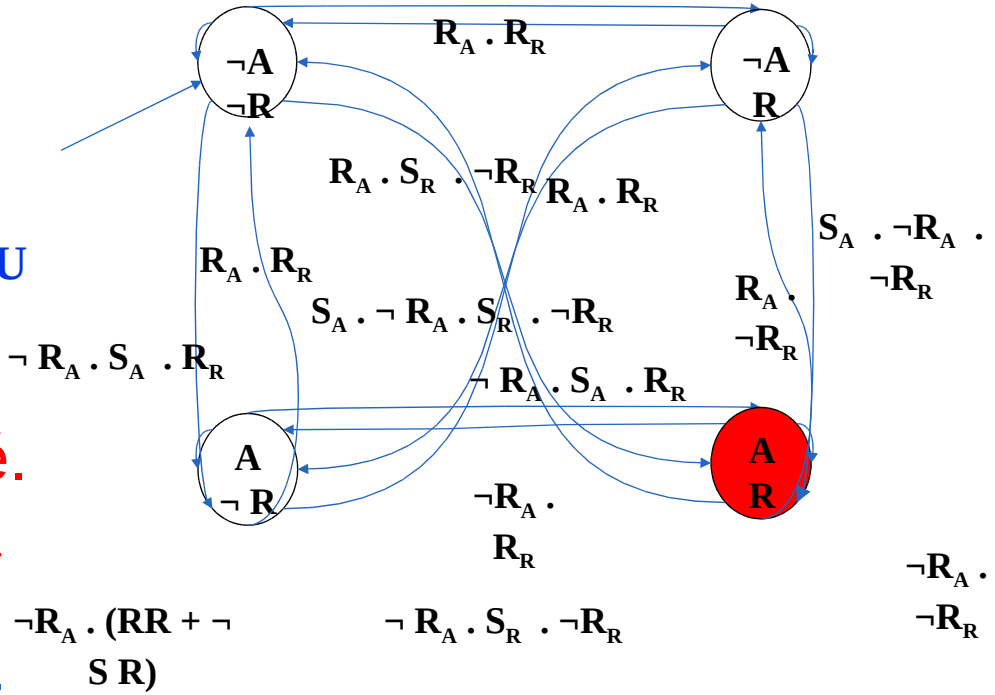
S_A = Poste_prise . Produit_saisi

S_R = Poste_depose . ¬Produit_saisi

R_A = Poste_depose + ¬Pos_haute + ARU

R_R = Poste_prise + ¬Pos_haute + ARU

L'état AR n'est pas souhaité.
S'il est atteignable il faudra l'interdire



Application à la validation de Programme Avancer. Reculer = 0 ? (1)

Atteignabilité de l'état « A,R » ? :

(depuis l'état « ¬A, ¬R ») : S_A . $\neg R_A$. S_R . $\neg R_R = 0$ (1)

(depuis l'état « A, ¬R ») : $\neg R_A$. S_R . $\neg R_R = 0$ (2)

(depuis l'état « ¬A, R ») : S_A . $\neg R_A$. $\neg R_R = 0$ (3)

(9) = 0 car S_A = Poste_prise . Produit_saisi et S_R = Poste_depose . ¬Produit_saisi

(2) = 0 car S_R = Poste_depose . ¬Produit_saisi et $\neg R_A$ = ¬Poste_depose . Pos_haute . ¬ARU

(3) = 0 car S_A = Poste_prise . Produit_saisi et $\neg R_R$ = ¬Poste_prise . Pos_haute . ¬ARU

Conclusion : l'état « A, R » ne peut jamais être atteint (il disparaît de l'automate composé « brut ») et la propriété Avancer.Reculer = 0 est VRAIE

Conclusion

Notions fondamentales des SED

- Espace d'états discrets
- Transition entre états sur occurrence d'événements
- La loi de commande à construire est l'ensemble des trajectoires d'états souhaitées dans l'espace d'états

Limites des automates à états finis

- Un état du système = un état de l'automate (approche monolithique)
- Modèle trop basique pour la modélisation des systèmes complexes
- Des états non souhaités dans le produit d'automates de composants

Besoins

- Expression simple du parallélisme, de la coordination, de la concurrence, de l'alternative, …
- Expression performante des fonct. de sortie $S_i(t) = f_i\ (e_1(t), \ldots, e_n(t), Q(t))$
- Expression du comptage du temps

Modélisation de SED à l ’aide d ’automates (8)

Exemples : conclusions

- les théories de base des langages et des automates finis (machines de Moore) supportent les besoins de modélisation du comportement des SED et procurent une base rigoureuse.

- les automates finis sont des machines d ’états trop rudimentaires pour exprimer des comportements complexes (beaucoup d ’états, parallélisme, concurrence, synchronisation, …)

- les automates finis rendent difficile l ’écriture d ’un modèle

⇒ D ’autres classes de machines d ’états sont mieux adaptées à la modélisation de la commande (Réseaux de Petri, Grafcet, …..)

⇒ une méthode de synthèse nécessite l’utilisation d’une classe d’automates (Ramadge & Wohnam)

⇒ Les automates sont très adaptés la validation des SED