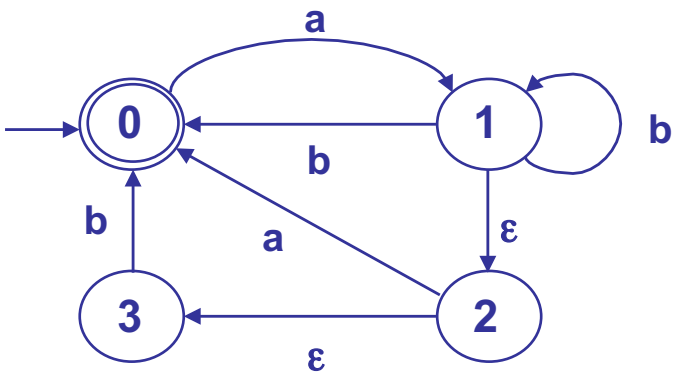
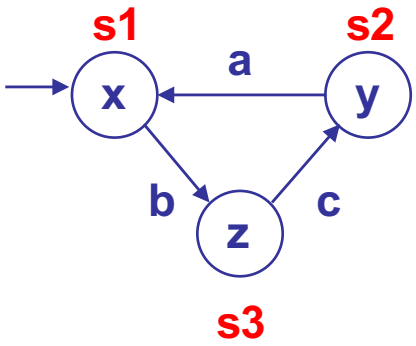
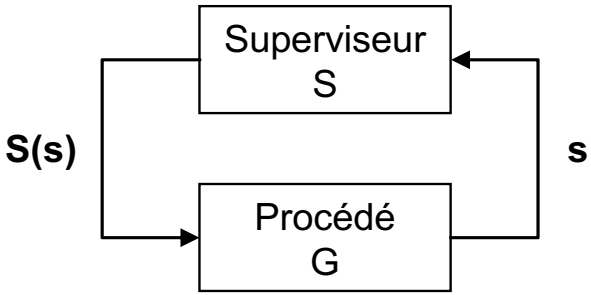
Systèmes à Evénements Discrets

THEORIE DES LANGAGES
AUTOMATES
THEORIE DE LA COMMANDE SUPERVISEE

Jean-François PETIN



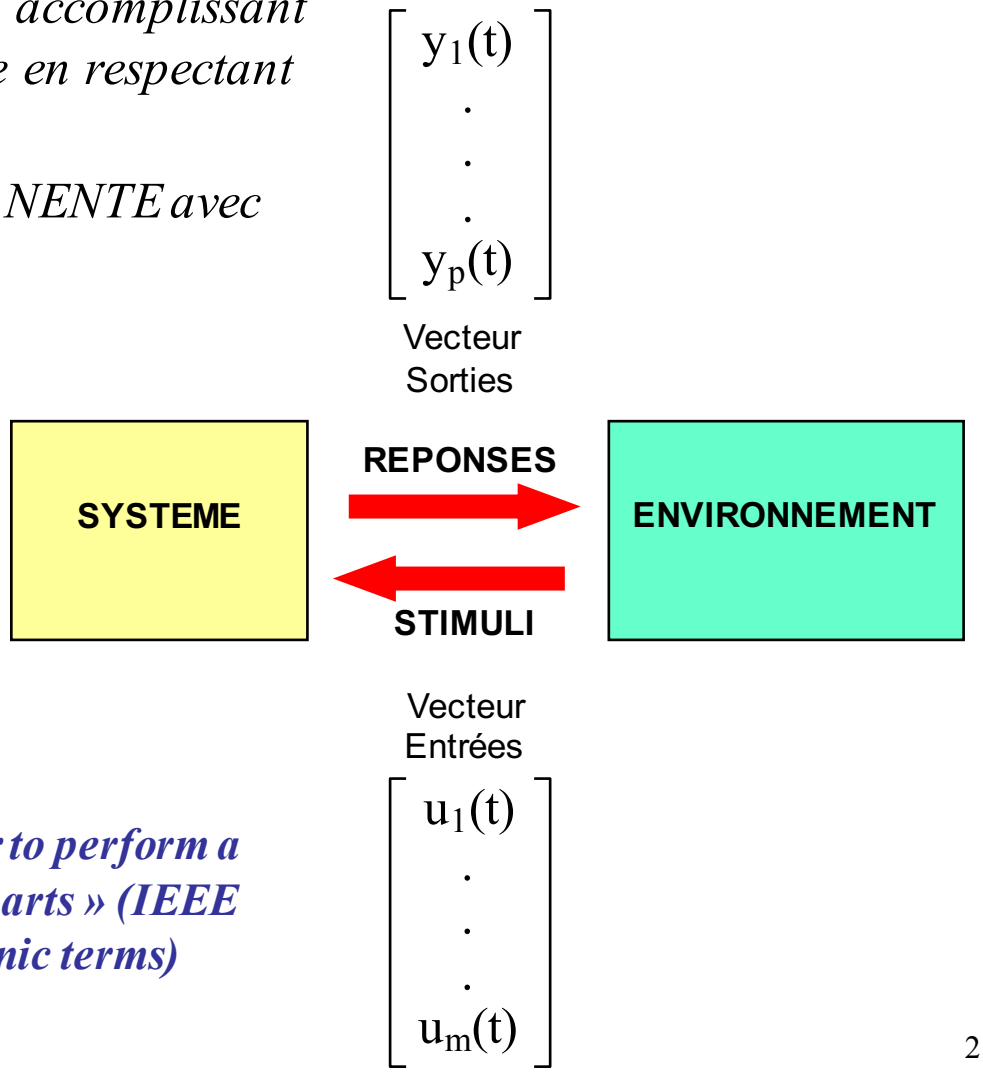




Introduction / Définitions

- Système
  - *ensemble de composants en interaction accomplissant conjointement une fonction particulière en respectant un ensemble de contraintes*
  - *maintient une INTERACTION PERMANENTE avec son environnement*

$$\begin{bmatrix} y_1(t) = g_1(u_1(t) \dots u_m(t)) \\ \vdots \\ y_p(t) = g_p(u_1(t) \dots u_m(t)) \end{bmatrix}$$

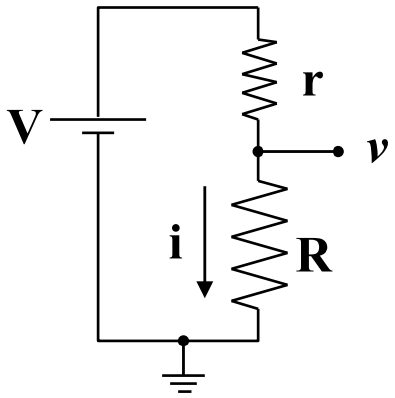


« a combination of components that act together to perform a function not possible with any of the infividual parts » (IEEE standard dictionary of Electrical and Electronic terms)

Introduction / Définitions

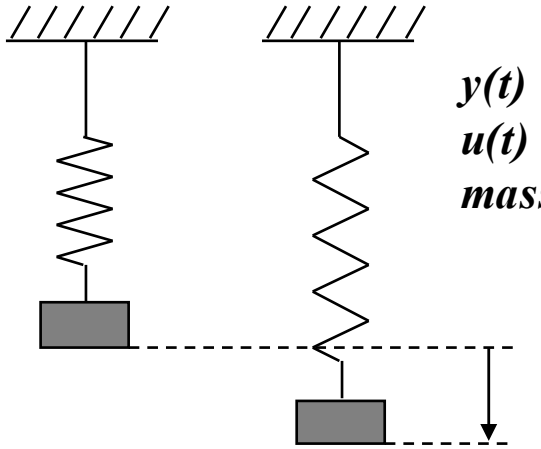
$$\begin{bmatrix} y_1(t) = g_1(u_1(t) \dots u_m(t)) \\ \vdots \\ y_p(t) = g_1(u_1(t) \dots u_m(t)) \end{bmatrix}$$

Exemple 1



$$v = V\left(\frac{R}{R+r}\right)$$

Exemple 2



y(t) déplacement
u(t) écart de la position initiale de la masse par rapport à la position de repos

u(0) = u_0 = y(0)

$$u(t) = \begin{cases} u_0 \text{ pour } t = 0 \\ 0 \text{ sinon} \end{cases}$$

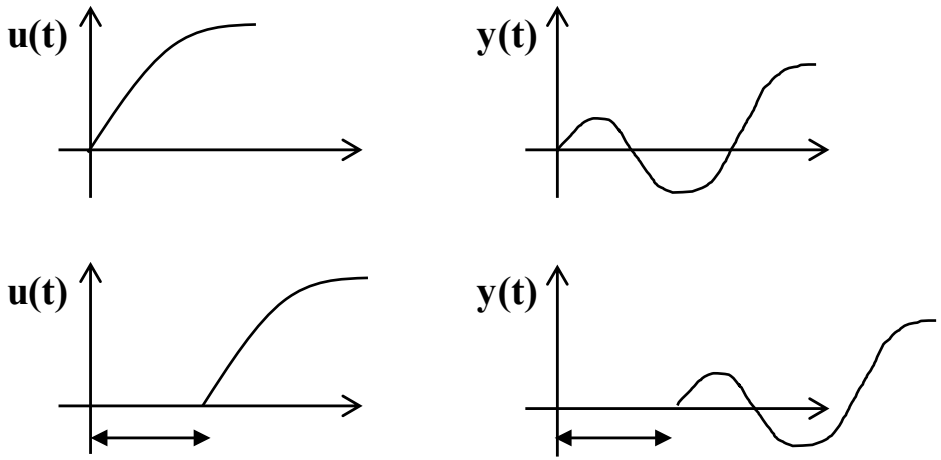
y(t) solution de l'équation différentielle $m\ddot{y} = -ky$

Introduction / Définitions

- Systèmes stationnaires ou invariants

Est-ce que la sortie est toujours la même quand la même entrée est décalée dans le temps ?

Exemple 2: est-ce que les paramètres m (masse) et k (coefficient de raideur du ressort) sont constants au cours du temps ?



Systèmes non stationnaires : $y(t) = g(u(t), t)$

Systèmes stationnaires ou invariants : $y(t) = g(u(t))$

Introduction / Définitions

- SYSTEMES DISCRETS ?
  - Etude des structures et objets mathématiques **énumérables** (comme des nombres entiers) qui ne font pas intervenir la notion de CONTINUITE.
  - Les variables caractérisant le comportement du système, ses entrées ou ses sorties prennent des valeurs distinctes (discrètes)

- Statique / Dynamique
  - Exemple 1 : sortie *v* à l'instant t non dépendante de l'entrée à l'instant $(t - \tau)$
  - Exemple 2 : sortie y à l'instant t dépendante de l'entrée à $t = t_0$, c'est-à-dire de la valeur $u(0) = u_0$

STATIQUE:	***y(t) indépendant de u(τ) pour tout τ < t***
DYNAMIQUE:	***y(t) dépendant des valeurs passées des entrées***
	mémorisation de l'histoire des entrées

Systèmes statiques (systèmes combinatoires) → SYSTEMES LOGIQUES
Systèmes dynamiques (systèmes à états) → SYSTEMES A EVENEMENTS DISCRETS

Algèbre de Boole

ALGEBRE DE BOOLE : Définition

- On appelle **algèbre de Boole** tout quadruplet $(\mathcal{B}, \overline{\ }, +, \cdot)$
  - constitué d'un ensemble $\mathcal{B}$ non vide muni d'au moins deux éléments sur lequel sont définies deux lois de composition interne **binaires** notées $(+, \cdot)$ et une loi de composition interne **unaire** notée $(\overline{\ })$,
  - qui satisfait les neufs axiomes suivants pour tout élément x, y, z de $\mathcal{B}$:

$x + y = y + x$	[1]	$x \cdot y = y \cdot x$	[2]	Commutativité
$x + (y \cdot z) = (x + y) \cdot (x + z)$	[3]	$x \cdot (y + z) = (x \cdot y) + (x \cdot z)$	[4]	Distributivité
$x + 0 = x$	[5]	$x \cdot 1 = x$	[6]	Elément neutre
$x + \overline{x} = 1$	[5]	$x \cdot \overline{x} = 0$	[8]	Complémentaire
$0 \neq 1$	[9]			

- De manière générale, cette définition peut s'appliquer pour des ensembles **infinis**.

Exemple d'algèbre de Boole : algèbre des parties d'un ensemble

L'ensemble support de cette algèbre est l'ensemble $\mathcal{P}(\mathcal{U})$ des sous-ensembles d'un ensemble $\mathcal{U}$ donné. Les éléments neutres sont $\varnothing$ et $\mathcal{U}$. Les trois lois de composition interne sont l'union $(\cup)$, l'intersection $(\cap)$ et le complément à $\mathcal{U}$ d'un sous-ensemble donné $(\neg)$. En s'appuyant sur les définitions de ces trois lois de composition, il est possible de démontrer que $(\mathcal{P}(\mathcal{U}), \cup, \cap, \neg)$ a une structure d'algèbre de Boole.

Algèbre de Boole

Algèbre de Boole des variables booléennes

- On s'intéresse à l'algèbre de Boole définie sur un **ensemble à deux éléments.**
- On appelle **booléen** tout élément de B = {0, 1}. Soit $x \in B$, on note $\bar{x}$ le booléen défini par:

$$\bar{x} = \begin{cases} 1 \text{ si } x = 0 \\ 0 \text{ si } x = 1 \end{cases}$$

- Soient $x \in B$ et $\varepsilon \in \{0, 1\}$. On note x^{ε}, le booléen défini par :

$$x^{\varepsilon} = \begin{cases} x \text{ si } \varepsilon = 1 \\ \bar{x} \text{ si } \varepsilon = 0 \end{cases}$$

- Deux lois de composition interne binaires (**ET logique** $\wedge$ noté et **OU logique** noté $\vee$) et une loi de composition interne unaire (**complément logique** noté $\neg$)

x	y	x ∨ y
0	0	0
0	1	1
1	0	1
1	1	1

x	y	x ∧ y
0	0	0
0	1	0
1	0	0
1	1	1

x	¬ x
0	1
1	0

- $(B, \neg, \vee, \wedge)$ est une algèbre de Boole *(Algèbre de Boole des variables booléennes)*.

Algèbre de Boole

Fonctions booléennes : définition

- On appelle **fonction booléenne d'ordre n**, toute fonction f de B^n dans B qui, à tout n-uplet de variables booléennes $(b_1, \ldots, b_n)$, associe la variable booléenne $f(b_1, \ldots, b_n)$:

$$\begin{aligned} f: \quad & B^n \longrightarrow B \\ & (b_1, \ldots, b_n) \mapsto f(b_1, \ldots, b_n) \end{aligned}$$

- On note F_n l'ensemble des fonctions booléennes d'ordre n et $F = \bigcup_{n=0}^{\infty} F_n$ l'ensemble des fonctions booléennes.

Algèbre de Boole

Fonctions booléennes usuelles

- Fonctions de F_0

  Fonctions constantes 1 et 0

  1 : $\longrightarrow B$, $\mapsto 1$

  0 : $\longrightarrow B$, $\mapsto 0$

- Fonctions de F_1

  Deux fonctions constantes

  1 : $B \longrightarrow B$, $x \mapsto 1$

  0 : $B \longrightarrow B$, $x \mapsto 0$

  Fonctions identité et négation

  Identité : $B \longrightarrow B$, $x \mapsto x$

  Négation : $B \longrightarrow B$, $x \mapsto \neg x$

- Fonctions de F_2
  - Deux fonctions constantes
  - Deux fonctions de projections et deux fonctions de négation des projections

  x_1 : $B^2 \longrightarrow B$, $(x_1, x_2) \mapsto x_1$

  x_2 : $B^2 \longrightarrow B$, $(x_1, x_2) \mapsto x_2$

  Négation de x_1 : $B^2 \longrightarrow B$, $(x_1, x_2) \mapsto \neg x_1$

  Négation de x_2 : $B^2 \longrightarrow B$, $(x_1, x_2) \mapsto \neg x_2$

  - Fonctions « somme » et « produit »

  $\vee : B^2 \longrightarrow B$, $(x_1, x_2) \mapsto x_1 \vee x_2 = \begin{cases} 1 \text{ si } x_1 = 1 \text{ ou } x_2 = 1 \\ 0 \text{ sinon} \end{cases}$

  $\wedge : B^2 \longrightarrow B$, $(x_1, x_2) \mapsto x_1 \wedge x_2 = \begin{cases} 1 \text{ si } x_1 = 1 \text{ et } x_2 = 1 \\ 0 \text{ sinon} \end{cases}$

Algèbre de Boole

Fonctions booléennes usuelles

- Fonctions de F_2 (suite)
  - Fonctions « implique », « est impliqué par », « n'implique pas » et « n'est pas impliqué par »

$\Rightarrow$: $B^2 \longrightarrow B$, $(x_1, x_2) \mapsto \neg x_1 \vee x_2$

$\Leftarrow$: $B^2 \longrightarrow B$, $(x_1, x_2) \mapsto x_1 \vee \neg x_2$

$\not\Rightarrow$: $B^2 \longrightarrow B$, $(x_1, x_2) \mapsto x_1 \wedge \neg x_2$

$\not\Leftarrow$: $B^2 \longrightarrow B$, $(x_1, x_2) \mapsto \neg x_1 \wedge x_2$

  - Fonctions « équivalent »

$\Leftrightarrow$: $B^2 \longrightarrow B$, $(x_1, x_2) \mapsto (x_1 \wedge x_2) \vee (\neg x_1 \wedge \neg x_2)$

  - Fonctions « non OU » (NOR), « non ET » (NAND)

NOR: $B^2 \longrightarrow B$, $(x_1, x_2) \mapsto \neg(x_1 \vee x_2)$

NAND: $B^2 \longrightarrow B$, $(x_1, x_2) \mapsto \neg(x_1 \wedge x_2)$

  - Fonction « OU exclusif » (XOR) notée $\oplus$

$\oplus$: $B^2 \longrightarrow B$, $(x_1, x_2) \mapsto (x_1 \wedge \neg x_2) \vee (\neg x_1 \wedge x_2)$

Algèbre de Boole

Table de vérité

La **Table de Vérité** d'une fonction booléenne $f \in F_n$ est un tableau comportant n+1 colonnes :

- les n premières colonnes représentent la valeur prise par les arguments $x_1, \ldots, x_n$ (entrées)
- la dernière colonne représente l'évaluation de la fonction $f(x_1, \ldots, x_n)$ (sortie).

Le nombre de ligne du tableau est de 2^n lignes.

Exemple

x_1	x_2	x_3	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Définition

Une fonction booléenne ne peut prendre que deux valeurs 0 ou 1, elle est donc connue dès que l'on connaît les n-uplets booléens où elle prend la valeur 1.

Le **support $S_n(f)$** d'une fonction $f \in F_n$ est l'ensemble des n−uplets $(x_1, \ldots, x_n) \in B^n$, tels que $f(x_1, \ldots, x_n) = 1$.

Exemple

x_1	x_2	f
0	0	1
0	1	0
1	0	0
1	1	1

On a donc $S_2(f) = \{(0, 0), (1, 1)\}$.

Algèbre de Boole

Définitions

- On appelle **monôme conjonctif** de F_n tout produit de projections ou de négations de projections, autrement dit tout élément pouvant s'écrire :

$$m = \prod_{i \in I} x_i^{\varepsilon_i} \quad \text{avec } I \subseteq [1, n] \text{ et } \varepsilon \in \{0,1\}$$

Exemple
$m = \overline{x_1} x_2 x_3 \overline{x_4}$

- On appelle **monôme disjonctif** de F_n tout somme de projections ou de négations de projections, autrement dit tout élément pouvant s'écrire :

$$m = \sum_{i \in I} x_i^{\varepsilon_i} \quad \text{avec } I \subseteq [1, n] \text{ et } \varepsilon \in \{0,1\}$$

Exemple
$m = x_1 + \overline{x_2} + x_3 + \overline{x_4}$

- Un monôme (conjonctif ou disjonctif) est dit **canonique** pour une fonction *f* de F_n si chacun des *n* arguments de la fonction *f* intervient exactement une fois dans le monôme.
  - Un monôme conjonctif canonique est appelé MINTERME
  - Un monôme disjonctif canonique est appelé MAXTERME

- En connaissant le support S_n d'une fonction booléenne *f*, on peut de manière automatique, en déduire une écriture algébrique de cette fonction :
  - sous la forme d'une « SOMME DE PRODUITS » (à base de MINTERMEs),
  - sous la forme d'un « PRODUIT DE SOMMES » (à base de MAXTERMEs).

Algèbre de Boole

Première forme normale de Lagrange (ou forme canonique disjonctive)

- Toute fonction booléenne de f de F_n peut s'écrire comme une somme de monômes conjonctifs canoniques de F_n. Cette écriture, appelée première forme normale de Lagrange est déterminée à partir du support de la fonction :

$$f(x_1, x_2, \ldots, x_n) = \sum_{(\varepsilon 1, \varepsilon 2, \ldots, \varepsilon n) \in Sn(f)} x_1^{\varepsilon_1} x_2^{\varepsilon_2} \ldots x_n^{\varepsilon_n}$$

Le **MINTERME** est le **PRODUIT LOGIQUE** (ET) de tous les états logiques des variables indépendantes. Pour une fonction logique donnée, il existe autant de MINTERMEs que de combinaisons possibles de variables indépendantes. Lorsqu'une variable est à l'état logique **1** dans une de ces combinaisons, elle est remplacée par **son nom**, et, quand elle est à l'état logique **0**, elle est remplacée par la **négation de son nom**.

L'expression logique de la fonction est alors la SOMME LOGIQUE (OU) de tous les MINTERMs pour lesquels le résultat de la fonction est **égal à 1**.

a	b	c	f		Minterms
0	0	0	0		$m_0 = \neg a \wedge \neg b \wedge \neg c$
0	0	1	0		$m_1 = \neg a \wedge \neg b \wedge c$
0	1	0	0		$m_2 = \neg a \wedge b \wedge \neg c$
0	1	1	1		$m_3 = \neg a \wedge b \wedge c$
1	0	0	0		$m_4 = a \wedge \neg b \wedge \neg c$
1	0	1	1		$m_5 = a \wedge \neg b \wedge c$
1	1	0	1		$m_6 = a \wedge b \wedge \neg c$
1	1	1	1		$m_7 = a \wedge b \wedge c$

Dans l'exemple, cela donne :

$f = m_3 + m_5 + m_6 + m_7$

$= (\neg a \wedge b \wedge c) \vee (a \wedge \neg b \wedge c) \vee (a \wedge b \wedge \neg c) \vee (a \wedge b \wedge c)$

Algèbre de Boole

Deuxième forme normale de Lagrange (ou forme canonique conjonctive)

- Toute fonction booléenne de f de F_n peut s'écrire comme un produits de monômes disjonctifs canoniques de F_n. Cette écriture, appelée première forme normale de Lagrange est déterminée à partir du support de la fonction :

$$f(x_1, x_2, \ldots, x_n) = \prod_{(\varepsilon 1, \varepsilon 2, \ldots, \varepsilon n) \in \overline{Sn(f)}} \overline{x_1^{\varepsilon_1}} + \overline{x_2^{\varepsilon_2}} + \ldots + \overline{x_n^{\varepsilon_n}}$$

Le **MAXTERME** est une **ADDITION LOGIQUE** (OU) de toutes les variables indépendantes d'une fonction logique. Pour une fonction logique donnée, il existe autant de MAXTERMs que de combinaisons possibles de ces variables dans la table de vérité de la fonction. Lorsque la variable est à l'état logique **0**, elle est remplacée par **son nom**, et, quand elle est à l'état logique **1**, elle est remplacée par la **négation de son nom**.
En fait, le MAXTERME est l'inverse du MINTERME correspondant à la même combinaison de variables.

L'expression logique de la fonction est alors le PRODUIT LOGIQUE (ET) de tous les MAXTERMEs pour lesquels le résultat de la fonction est **égal à 0.**

a	b	c	f		Maxterms
0	0	0	0		$M_0 = a \vee b \vee c$
0	0	1	0		$M_1 = a \vee b \vee \neg c$
0	1	0	0		$M_2 = a \vee \neg b \vee c$
0	1	1	1		$M_3 = a \vee \neg b \vee \neg c$
1	0	0	0		$M_4 = \neg a \vee b \vee c$
1	0	1	1		$M_5 = \neg a \vee b \vee \neg c$
1	1	0	1		$M_6 = \neg a \vee \neg b \vee c$
1	1	1	1		$M_7 = \neg a \vee \neg b \vee \neg c$

Dans l'exemple, cela donne :

$f = M_0.M_1.M_2.M_4$

$= (a \vee b \vee c) \wedge (a \vee b \vee \neg c) \wedge (a \vee \neg b \vee c) \wedge (\neg a \vee b \vee c)$

Algèbre de Boole

Simplification des fonctions booléennes

POSTULATS	
(1) $\overline{0} = 1$	(6) $\overline{1} = 0$
(2) $0 + 0 = 0$	(7) $0 . 0 = 0$
(3) $0 + 1 = 1$	(8) $0 . 1 = 0$
(4) $1 + 0 = 1$	(9) $1 . 0 = 0$
(5) $1 + 1 = 1$	(10) $1 . 1 = 1$
THÉORÈMES POUR UNE SEULE VARIABLE	
(11) $a + 1 = 1$	(15) $a . 1 = a$
(12) $a + 0 = a$	(16) $a . 0 = 0$
(13) $a + a = a$	(17) $a . a = a$
(14) $a + \overline{a} = 1$	(18) $a . \overline{a} = 0$
(19) $\overline{\overline{a}} = a$	

Algèbre de Boole

Simplification des fonctions booléennes

THÉORÈMES POUR PLUSIEURS VARIABLES	
Loi de Commutativité (ou Loi d'Échange)	
(20) $a + b = b + a$	(21) $a \cdot b = b \cdot a$
Loi d'Associativité (ou Loi de Liaison)	
(22) $a + (b + c) = (a + b) + c$	(23) $a \cdot (b \cdot c) = (a \cdot b) \cdot c$
Loi de Distributivité (ou Loi de Répartition)	
(24) $(a + b) \cdot (b + c) = (a.b) + (a.c) + (b.b) + (b.c)$	(25) $a \cdot (b+c) = (a.b) + (a.c)$
LOIS DE MORGAN	
(26) $\overline{a + b} = \overline{a} \cdot \overline{b}$	(27) $\overline{a \cdot b} = \overline{a} + \overline{b}$
LOIS D'ABSORPTION LOGIQUE	
(28) $a + a \cdot b = a$	(29) $a \cdot (a + b) = a$
LOIS D'ADJACENCE LOGIQUE	
(30) $a \cdot b + a \cdot \overline{b} = a$	(31) $(a + b) \cdot (a + \overline{b}) = a$

Simplification des fonctions booléennes

Définitions

- Les méthodes de simplification sont, presque toutes, basées sur une relation d'ordre partiel dans l'ensemble des fonctions booléennes.
- Soient f et g deux fonctions de F_m. Par définition, on dit que que f est plus petite que g si et seulement si :

$$f \leq g \text{ ssi } S_n(f) \subseteq S_n(g)$$

- Relation d'ordre sur les monômes conjonctifs

  Soit m et m' deux monômes conjonctifs :

$$m \leq m' \Leftrightarrow \exists \text{ m''}, \text{ m'' est un monôme conjonctif et m = m'.m''}$$

  En particulier, les plus petits monômes sont les monômes conjonctifs canoniques, le plus grand monôme conjonctif est la constante 1.

- Un monôme m est **maximal** pour une fonction $f \in F_n$ ssi les trois conditions suivantes sont vérifiées :
  - m est un monôme conjonctif
  - m $\subset S_n(f)$ *(que l'on pourra noter* $m < f$*)*
  - $\nexists m' \neq m$ monôme conjonctif tel que $m \leq m' \leq f$

  On note $M(f)$ l'ensemble des monômes maximaux de f.

Simplification des fonctions booléennes

Exemple

x_1	x_2	x_3	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

- $f = x_1x_2 + x_1x_2\overline{x_3} + x_2x_3 + x_3$ définie sur F_3
- f a pour support $S_3(f) = \{(0,0,1), (0,1,1), (1,0,1), (1,1,0), (1,1,1)\}$

- $m = x_2 . x_3$

  On peut trouver $m' = x_3$ tel que $m = m' . x_2$ et $m' \leq f$ *(i.e.* $m' \subseteq S_3(f)$)

  on a donc $m \leq m' \leq f$, d'où $m = x_2 . x_3$ n'est pas maximal
- $m = x_1 . x_2 . \overline{x_3}$

  il existe $m' = x_1x_2$ tel que $m = m' . \overline{x_3}$ et $m' \subseteq S_3(f)$, donc $m \leq m' \leq f$

  donc *m* n'est pas maximal

- $m = x_1 . x_2$

  On ne trouve pas m' tel que $m = m'.m''$ et $m' \leq f$ *(* en effet $x_1 \notin S_3(f)$ et $x_2 \notin S_3(f)$)

  $m = x_1x_2$ est donc un monôme maximal

  De la même manière, $m = x_3$ est maximal puisqu'il n'existe pas de m' tel que $m = m'.m''$.

Simplification des fonctions booléennes

Proposition

- Toute fonction booléenne est la **somme de ses monômes maximaux**, c'est à dire que :

$$\forall f \in F_n, f = \sum_{m \in M(f)} m$$

- L'écriture d'une fonction booléenne à partir de son support grâce aux formes normales de Lagrange utilisent les monômes canoniques. Ces monômes sont justement ceux dont l'écriture est la plus longue et va à l'encontre d'une écriture simplifiée.

- L'objectif est donc de trouver l'ensemble des monômes maximaux. On dit que *f'* est une écriture simplifiée de *f* si et seulement si :
  - *f'* est une somme de monômes maximaux de *f*

$$f' = \sum_{m \in M(f)} m$$

  - Si l'on retire un monôme à *f'*, alors on ne retrouve pas une écriture de *f*

$$\nexists\, m_j \in M(f),\ f = \sum_{m \in M(f) \setminus mj} m$$

Simplification des fonctions booléennes

Exemple

x_1	x_2	x_3	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

- $f = x_1x_2 + x_1x_2\overline{x_3} + x_2x_3 + x_3$ définie sur F_3
- f a pour support $S_3(f) = \{(0,0,1), (0,1,1), (1,0,1), (1,1,0), (1,1,1)\}$

- $m = x_2 . x_3$

  On peut trouver $m' = x_3$ tel que $m = m' . x_2$ et $m' \leq f$ *(i.e.* $m' \subseteq S_3(f)$)

  on a donc $m \leq m' \leq f$, d'où $m = x_2 . x_3$ n'est pas maximal
- De la même manière, $m = x_1 . x_2 . \overline{x_3}$ n'est pas maximal, puisqu'il existe $m' = x_1x_2$ tel que $m = m' . x_3$ et $m' \subseteq S_3(f)$, donc $m \leq m' \leq f$

- $m = x_1 . x_2$

  On ne trouve pas m' tel que $m = m'.m''$ et $m' \leq f$ *(* en effet $x_1 \not\in S_3(f)$ et $x_2 \not\in S_3(f)$)

  $m = x_1x_2$ est donc un monôme maximal

  De la même manière, $m = x_3$ est maximal puisqu'il n'existe pas de m' tel que $m = m'.m''$.

- Ecriture simplifiée de f : $f = x_1x_2 + x_3$

Simplification des fonctions booléennes

SIMPLIFICATION PAR LA MÉTHODE DE KARNAUGH

Les axiomes et propriétés des fonctions de l'Algèbre de Boole permettent la simplification des expressions mais cela reste difficile pour de longues expressions logiques.
La méthode de simplification de KARNAUGH est une méthode qui permet de déterminer simplement l'ensemble des formes simplifiées d'une fonction en en déterminant graphiquement les monômes maximaux.

PRINCIPE DE LA MÉTHODE

Soit $f \in F_4$.

1) On dresse un tableau à card $(B^4) = 16$. Chaque case du tableau correspond à un monôme canonique conjonctif (MINTERME). L'avantage principal de la Table de KARNAUGH par rapport à la Table de Vérité est qu'elle permet une meilleure visualisation des propriétés de l'adjacence logique.
   La disposition des mintermes dans une telle table est telle que chacun des mintermes est voisin de ses mintermes adjacents. Un minterme est dit adjacent à un autre si leurs expressions ne diffèrent que d'une seule variable qui se présente dans la forme « directe » dans l'un et dans la forme « complémentée » dans l'autre.

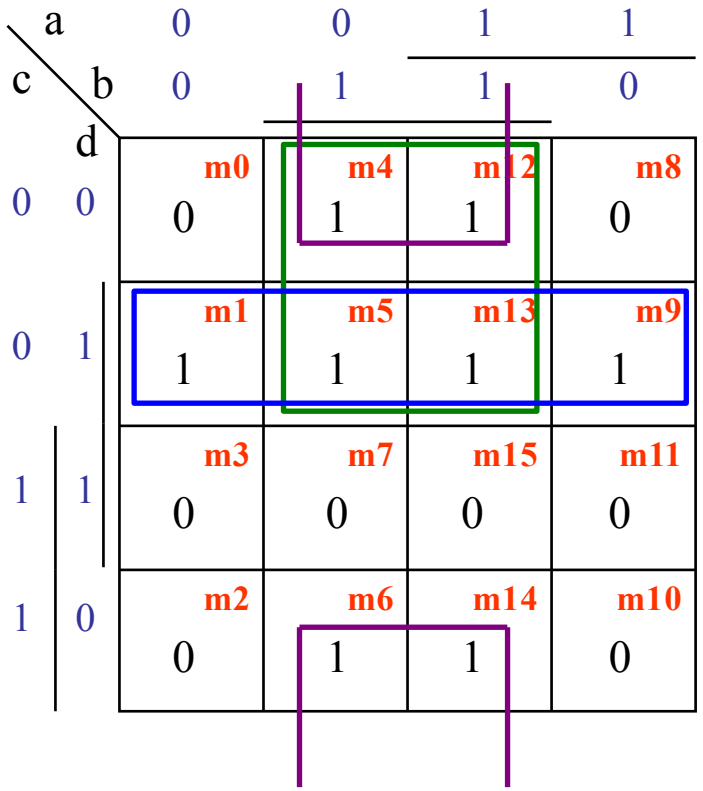
c \ b	a: 0	0	1	1
	b: 0	1	1	0
d				
0	**m0** /a./b./c./d	**m4** /a.b./c./d	**m12** a.b./c./d	**m8** a./b./c./d
1	**m1** /a./b./c.d	**m5** /a.b./c.d	**m13** a.b./c.d	**m9** a./b./c.d
1	**m3** /a./b.c.d	**m7** /a.b.c.d	**m15** a.b.c.d	**m11** a./b.c.d
0	**m2** /a./b.c./d	**m6** /a.b.c./d	**m14** a.b.c./d	**m10** a./b.c./d

Simplification des fonctions booléennes

SIMPLIFICATION PAR LA MÉTHODE DE KARNAUGH

PRINCIPE DE LA MÉTHODE (Suite)

2) pour chaque $(\varepsilon1, \varepsilon2, \varepsilon3, \varepsilon4) \in B^4$, on remplit la case $a^{\varepsilon1}b^{\varepsilon2}c^{\varepsilon3}d^{\varepsilon4}$ avec $f(\varepsilon1, \varepsilon2, \varepsilon3, \varepsilon4)$.
3) on détermine l'ensemble des monômes maximaux de f : on effectue les plus grands regroupements rectangulaires non discontinus possibles de 1, 2, 4, 8 ou 16 cases adjacentes contenant un "1". Pour déterminer l'expression algébrique du monôme correspondant à un regroupement, il suffit de constituer un monôme ne faisant apparaître que les variables qui gardent une valeur constante pour toutes les cases du regroupement en complémentant celles qui valent 0 ($\overline{c}.d$, $\overline{c}.b$, $b.\overline{d}$)
4) on détermine l'ensemble des monômes centraux : on repère parmi les monômes maximaux ceux qui contiennent au moins une case « 1 » qui n'est pas recouverte par un autre monôme maximal ($\overline{c}.d$ et $b.\overline{d}$)
5) on détermine l'ensemble de toutes les combinaisons de monômes maximaux recouvrant toutes les cases « 1 » avec des monômes centraux.



$$f = \overline{c}.d + b.\overline{d}$$

Algèbre de Boole

DESCRIPTION PAR UNE FONCTION BOOLEENNE

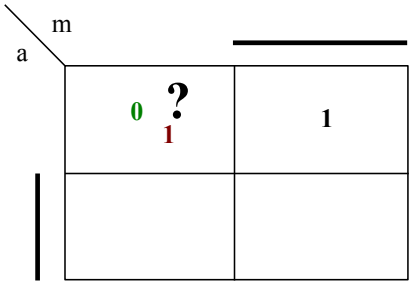
$Y = f(x1, x2, ..., xn)$ avec ***x1, …, xn*** entrées du système, ***Y*** sa sortie et ***f*** une fonction booléenne

LIMITES

Exemple

Un moteur électrique M est commandé par 2 boutons-poussoirs : m (marche) et a (arrêt). Le fonctionnement est le suivant :

1) à l'arrêt, m et a ne sont pas enclenchés,
2) on appuie sur m seulement, le moteur M tourne,
3) on relâche m, le moteur M continue de tourner,
4) on appuie sur a, le moteur M s'arrête,
5) on relâche a, le moteur M reste arrêté.



Question : $M = f(m,a)$?

Réponse : La variable de sortie M (moteur) est bien fonction des deux variables d'entrées *m* et *a*.

Néanmoins lorsque m = 0, a = 0 , M = 0 ou 1 ?

L'état courant du système dépend de son état antérieur (son « passé »), c'est à dire de la séquence des entrées l'ayant amené dans cet état antérieur.

→ Système à Evénements Discrets

- Etat
  - Mémoire minimale nécessaire pour connaître le comportement futur d'un système

Exemple 2

$u(t)$, $y(t_1)$ pour $t_1 > t_0$ étant connus est-il possible de déterminer $y(t_1 + \tau)$ pour $\tau > 0$) ?

$$\begin{cases} y(t_1 + \tau) > y(t_1) \text{ si la masse descend à l'instant } t_1 \\ y(t_1 + \tau) < y(t_1) \text{ si la masse monte à l'instant } t_1 \\ y(t_1 + \tau) = y(t_1) \text{ si la masse est au repos à l'instant } t_1 \end{cases}$$

Information manquante : vitesse de la masse $\dot{y}(t_1)$

L'état d'un système au temps t_0 est l'information requise à t_0 telle que la sortie $y(t)$ pour $t \geq t_0$ est déterminée de manière unique à partir de cette information et de l'entrée $u(t)$, $t \geq t_0$

ESPACE D'ETAT: **ensemble des valeurs possibles pour les états $x_i(t)$**

EQUATIONS D'ETAT: **ensemble des équations nécessaires pour définir l'état $x(t)$, $\forall\, t > t_0$, à partir de $x(t_0)$ et $u(t)$,**

- Espace d'état continu
  - l'espace d'état est un continuum constitué par un vecteur de dimension n de nombres réels (ou complexes)
  - exemple 2: vitesse du ressort
  - systèmes décrits par des équations différentielles

$$\begin{cases} \dot{x}(t) = f(x(t),u(t),t), & x(t_0) = x_0 \\ y(t) = g(x(t),u(t),t) \end{cases}$$

Systèmes linéaires

$[g(a_1u_1 + a_2u_2) = a_1g(u_1) + a_2g(u_2)]$

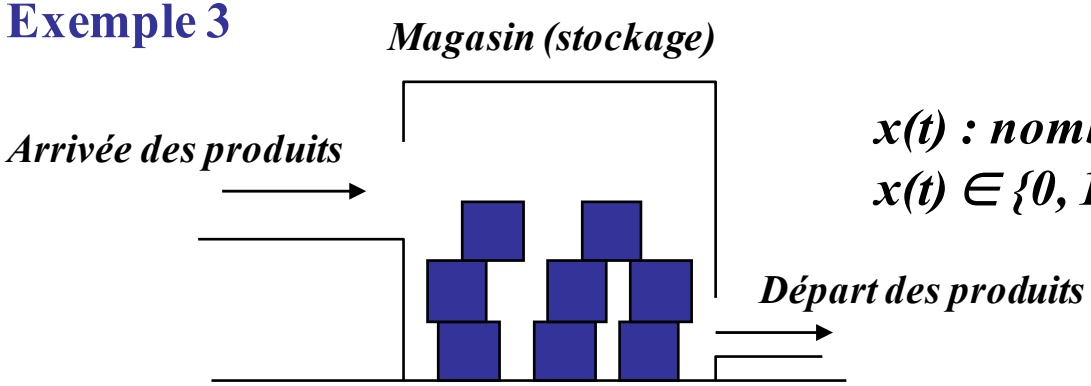
$$\begin{cases} \dot{x}(t) = A(t)x(t) + B(t)u(t) \\ y(t) = C(t)x(t) + D(t)u(t) \end{cases}$$

Systèmes linéaires invariants

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t) \\ y(t) = Cx(t) + Du(t) \end{cases}$$

- Espace d'état discret
  - l'espace d'état est un ensemble discret
  - exemples: {ON, OFF}, {montée, descente, repos}

Exemple 3

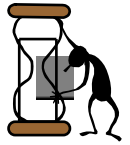


x(t) : nombre de produits dans le magasin
x(t) ∈ {0, 1, 2, … n}

  - transitions entre états

« Si quelque chose de particulier arrive et que le système est dans l'état x_1, alors le nouvel état devient x_2 »

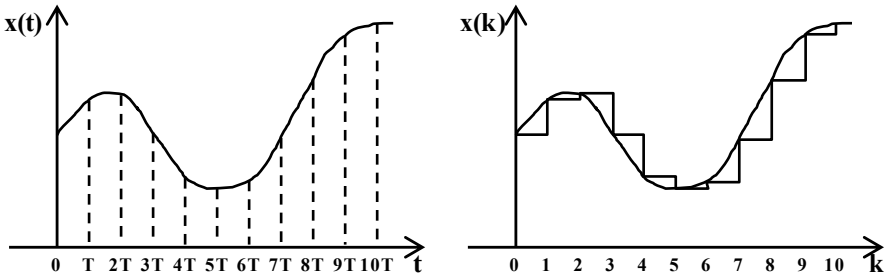
- Modélisation du temps



  - t : variable continue *(correspond à une vision physique et intuitive du temps dans le monde réel)*
  - t : variable discrète *(les entrées u(t), les états x(t) et les sorties y(t) ne sont définies qu'à des instants précis)*

SYSTÈME À TEMPS DISCRET

  - temps défini comme une séquence d'intervalles égaux
    - $t_0 < t_1 < \ldots < t_k$
    - $t_{k+1} - t_k$: T (période d'échantillonnage)



$$\dot{x}(t) = Ax(t) + Bu(t)$$
$$y(t) = Cx(t) + Du(t)$$

$$x(k+1) = Fx(k) + Gu(k)$$
$$y(k) = Cx(k) + Du(k)$$

La discrétisation du temps n'implique pas la discrétisation des états

Systèmes à Evénements Discrets

- **Evénement**
  - *On appelle EVENEMENT un changement de valeur d 'une variable survenant à une date donnée. Pour distinguer n changements de valeur identique de la même variable, on appelle occurrence de l 'événement chaque variation de cette variable à une date Di.*
  - Exemples: appui sur un bouton, arrivée d'un produit (cf exemple 3)

- Temps discret vs Evénement
  - Espace d'état continu
    - L'état change continûment avec le temps
    - Vrai à temps continu mais aussi à temps discret *(à chaque k+1, on s'attend à voir changer l'état)*

      ⇒ ***Variable de temps (t ou k) est une variable indépendante***

  - Espace d'état discret
    - L'état ne change qu'à certains points dans le temps
    - Horloge : à chaque « coup » d'horloge, il doit y avoir une occurrence d'un événement

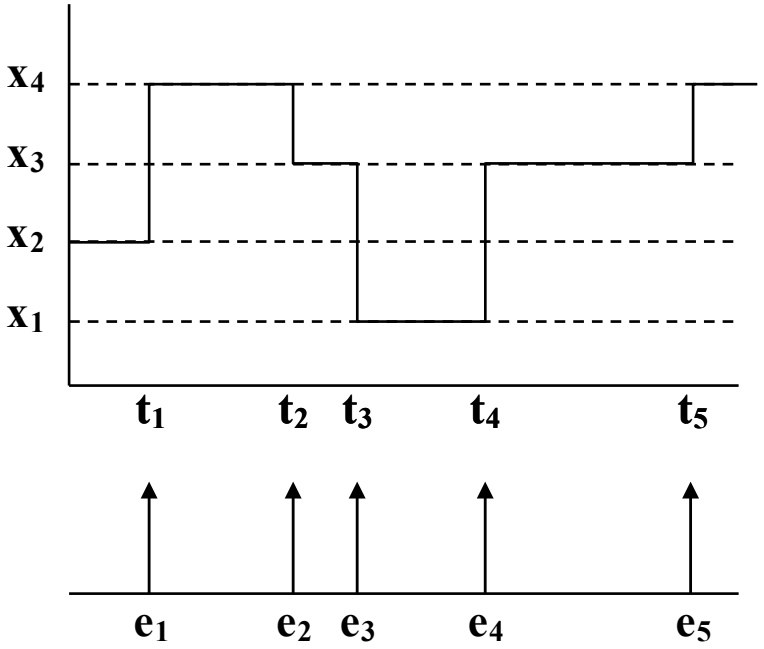
      ⇒ ***Transitions d'états SYNCHRONISÉES par l'horloge***

    - Occurrence d'événements peut survenir à n'importe quel instant

      ⇒ ***Transitions d'états associées à l'apparition d'événements ASYNCHRONES***

DEFINITION D'UN S.E.D. : Système à espace d'état discret dont les transitions entre états sont associées à l'occurrence d'événements discrets asynchrones

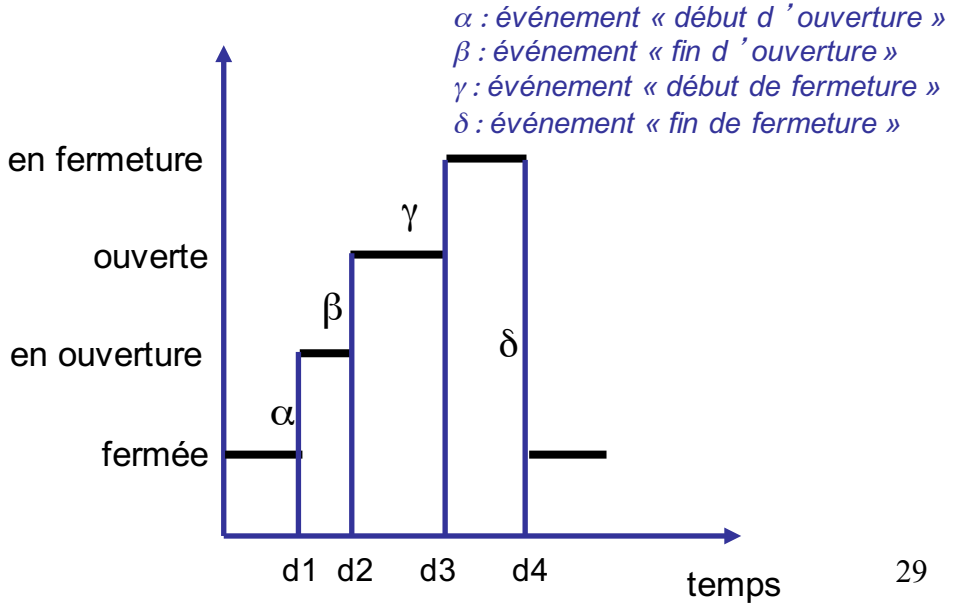
$X=\{x_1, x_2, x_3, x_4\}$



Exemple

Vanne décrite par 4 états : "fermée", "en ouverture", "ouverte" et "en fermeture".

α, β, γ, et δ sont les quatre événements qui permettent l'évolution de ce SED dans son espace d'état.

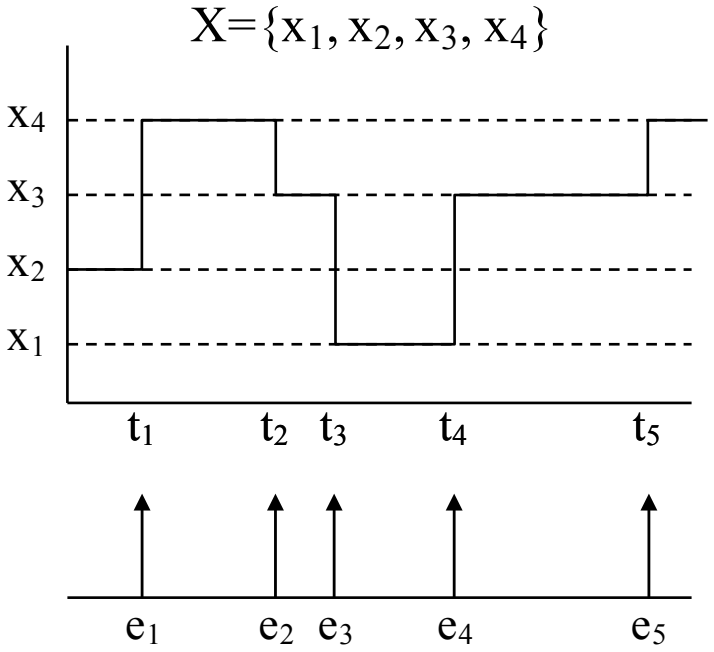


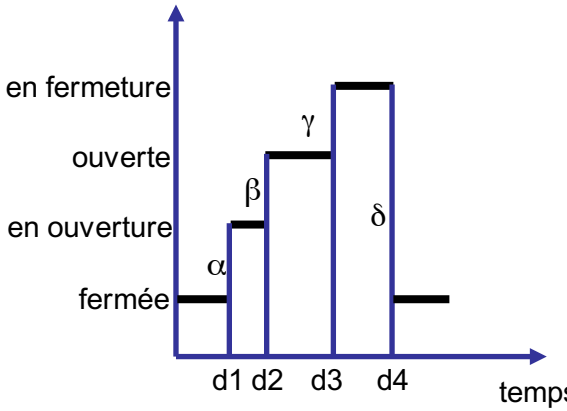
Systèmes à Evénements Discrets

Le modèle de comportement du système ne peut alors plus être une équation différentielle ou aux différences finies. Il doit décrire la trajectoire des états du système comme une fonction d'une séquence d'événements d'entrée.

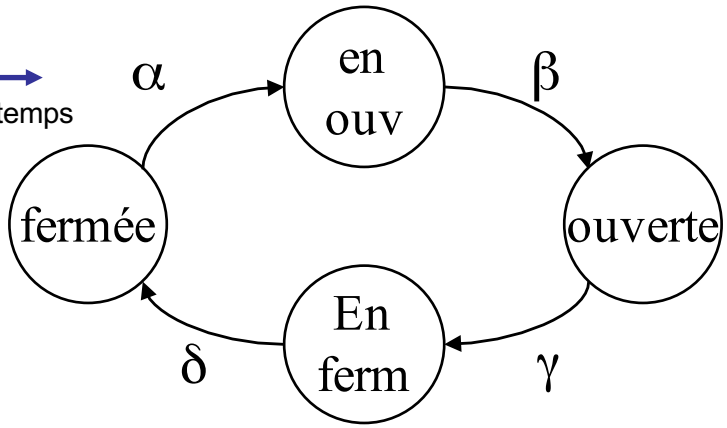
On peut donner deux représentations du comportement de la vanne à partir de son état initial supposé être « fermée » (on parlera également de *trajectoire* du système « vanne ») :

- un ***modèle temporel*** formé par la suite ordonnée des couples (e1, t1), (e2, t2), …
- un ***modèle logique*** basé sur les séquences d'événements {e1, e2, e3, e4}





Modèle logique



Exemples de Systèmes à Evénements Discrets

- **Réseaux de communication** : protocoles
- **Systèmes manufacturiers** : flux de production, commande des systèmes de production
- **Systèmes embarqués** : automobile, avionique, ferroviaire
- **Systèmes hybrides** : continus / discrets

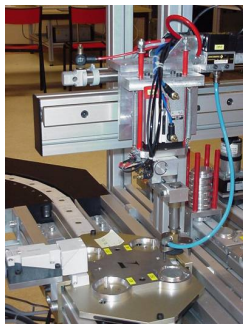
Aéronautique

Ferroviaire

Automobile

Militaire

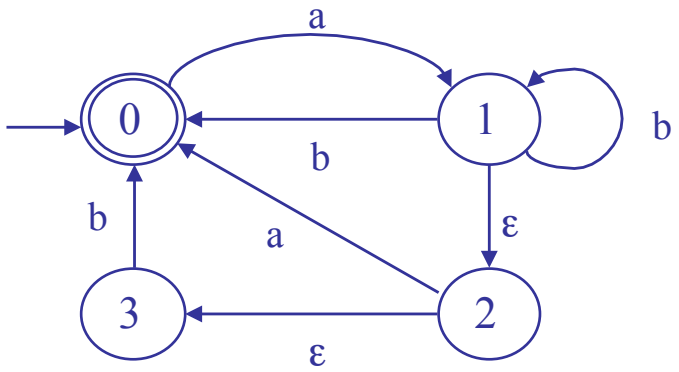
Spatial

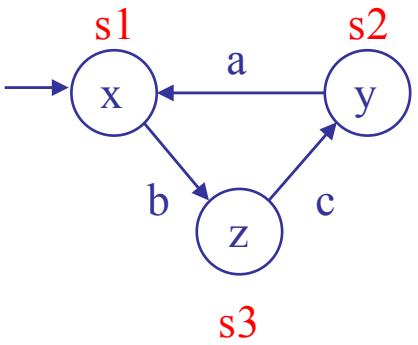


Système de production manufacturier

Energie

Théorie des langages
Automates à états finis





Théorie des Langages

- Un **alphabet** est un ensemble **fini** de symboles. Dans le cas d'un SED, un alphabet pourra représenter l 'ensemble des événements d'entrée possibles d'un système

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

- Un **mot** (ou chaine ou séquence) est une séquence **finie** d'éléments d'un alphabet Σ

  Exemple: *abba* est un mot défini sur Σ.

  - ε est le mot vide
  - Σ^* représente l'ensemble de tous les mots que l'on peut construire sur Σ (ε inclus)

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

- Un **langage** est un ensemble L de mots définis sur l'alphabet Σ.
  - C'est un sous-ensemble de Σ^*
  - Exemple $\Sigma = \{a, b, g\}$,
    - L1 = {ε, a, ab}
    - L2 = {ensemble des mots de longueur 3 commençant par a}

      → **langage fini** (9 mots) : {aaa, aab, aag, aba, abb, abg, aga, agb, agg}
    - L3 = {ensemble des mots commençant par a} → **langage infini**

Opérations sur les langages

- Soit deux langages L1, L2 $\subseteq \Sigma^*$

- **UNION** $L_1 \cup L_2 = \{w : w \in L_1 \text{ ou } w \in L_2\}$

- **INTERSECTION** $L_1 \cap L_2 = \{w : w \in L_1 \text{ et } w \in L_2\}$

- **CONCATÉNATION** $L_1.L_2 = \{s \in \Sigma^* : (s=s_1s_2) \text{ et } (s_1 \in L_1) \text{ et } (s_2 \in L_2)\}$

- **FERMETURE ITERATIVE** $L^* = \{\varepsilon\} \cup L \cup LL \cup LLL \cup \ldots$
  (fermeture de Kleene) $= \{\varepsilon\} \cup L^1 \cup L^2 \cup L^3 \cup \ldots L^n \cup \ldots$

- **FERMETURE PREFIXIELLE**
  - **Préfixe d'un mot** u, v $\in \Sigma^*$, u est préfixe de v si $\exists$ w $\in \Sigma^*$: v = uw
    $\forall$ s $\in \Sigma^*$, ε et s sont des préfixes de s
  - **Fermeture préfixielle** L $\subseteq \Sigma^*$, $\overline{L} = \{u \in \Sigma^* : \exists v \in \Sigma^* (uv \in L)\}$
    Si $\overline{L}$ = L alors L est dit préfixe-clos ou fermé

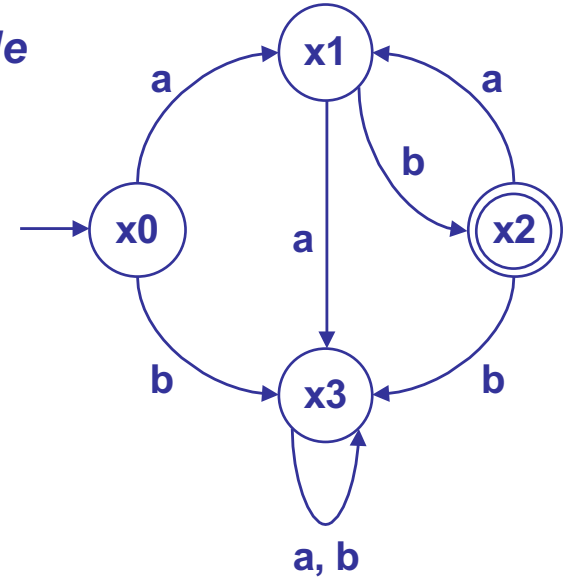
Opérations sur les langages : exemples

- $\Sigma = \{a,d\}$, L1 = {a, ad}, L2 = {aa}
  - **UNION** $L_1 \cup L_2 = \{a, aa, ad\}$
  - **INTERSECTION** $L_1 \cap L_2 = \emptyset$
  - **CONCATÉNATION** $L_1.L_2 = \{aaa, adaa\}$
  - **FERMETURE ITERATIVE** $L1^* = \{\varepsilon\} \cup L^1 \cup L^2 \cup L^3 \cup \ldots L^n \cup \ldots$
    $L1^* = \{\varepsilon\} \cup \{a, ad\} \cup \{aa, aad, ada, adad\} \cup \ldots$
  - **FERMETURE PREFIXIELLE** $\overline{L2} = \{\varepsilon, a, aa\}$
- $\Sigma = \{a,b,g\}$, L1={ε, a, abb}, L2={ε, g}
  - **CONCATÉNATION** $L1.L2 = \{\varepsilon, g, a, ag, abb, abbg\}$
  - **FERMETURE ITERATIVE** $L1^* = \{\varepsilon, a, abb, aa, aabb, abba, abbabb, \ldots\}$,
    $L2^* = \{\varepsilon, g, gg, ggg, gggg, \ldots\}$
  - **FERMETURE PREFIXIELLE** $\overline{L1} = \{\varepsilon, a, ab, abb\}$, $\overline{L2} = \{\varepsilon, g\}$

Automates

- Un automate est un quintuplet $G = (X, \Sigma, f, x_0, X_m)$ où :
  - X est l'ensemble des états
  - Σ est un alphabet d 'entrée
  - f est la fonction de transition d'états définie de $X \times \Sigma \rightarrow X$ qui associe un état de départ et un symbole d'entrée à un état d'arrivée
  - x_0 est l'état initial
  - X_m est l'ensemble des états finaux ou états marqués ($X_m \subseteq X$)

Exemple



- $X = \{x_0, x_1, x_2, x_3\}$
- $\Sigma = \{a,b\}$
- f est représentée par les arcs associés à des symboles de l'alphabet $f(x_0, a) = x_1$
-x_0 état initial
-$X_m = \{x_2\}$

Langages générés

- Le domaine de f est souvent étendu de X x Σ à X x Σ*
  - f(x, ε) = x et f(x, se) = f(f(x,s), e) pour s ∈ Σ* et e ∈ Σ

- ***Langage généré***

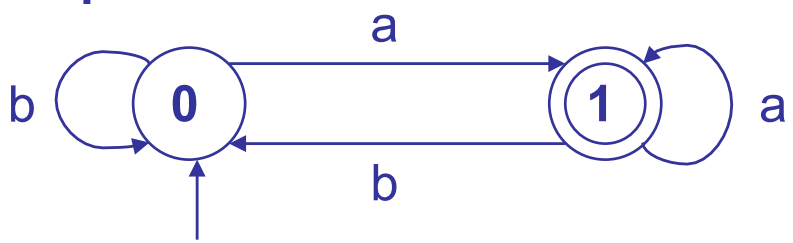
$$\mathcal{L}(G) = \{s \in \Sigma^* : f(x_0, s) \text{ est défini}\}$$

  - $\mathcal{L}(G)$ est préfixe-clos par définition: un chemin depuis x0 est possible si tous ses préfixes sont aussi possibles
  - Si f est une fonction totale, $\mathcal{L}(G) = \Sigma^*$

- ***Langage marqué***

$$\mathcal{L}_m(G) = \{s \in \mathcal{L}(G) : f(x_0, s) \in X_m\}$$

Exemple



- $\mathcal{L}(G) = \Sigma^*$
- $\mathcal{L}_m(G) =$ {a, aa, ba, aaa, aba, bba, …}
  c'est-à-dire tous les mots finissant par a

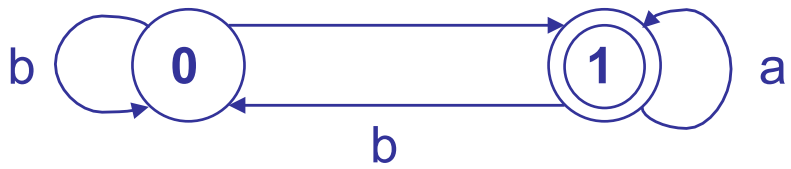
Langages réguliers

- Les **expressions régulières** sont définies à partir d'expressions élémentaires et des opérations d'union, de concaténation et de fermeture itérative.
  - Tout événement est une expression régulière.
  - Ø et ε sont des expressions régulières.
  - Si s et r sont des expressions régulières, alors s+r, s.r, s* et r* (union, concaténation, fermeture de Kleene) sont des expressions régulières.

- **Expressions régulières** et **langages**

L'expression régulière b + a.(b + c) correspond au langage L = {b, ab, ac}

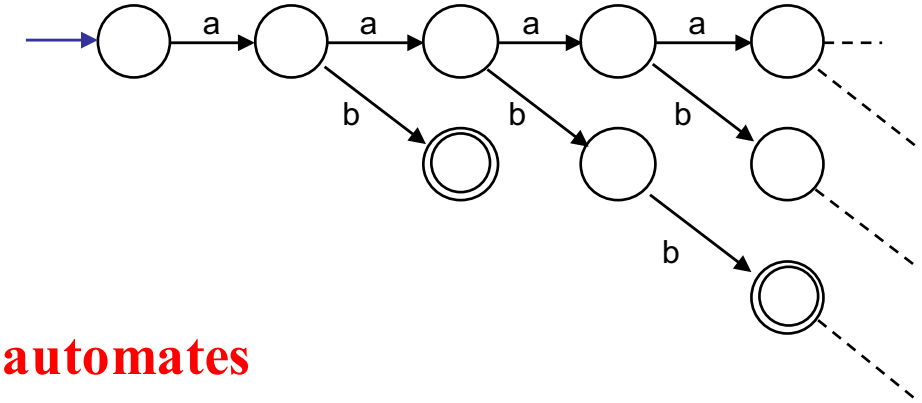
- Un **langage régulier** est un langage marqué par un automate fini (**théorème de Kleene**)
  - L'automate est fini mais le langage peut être fini ou infini
  - L'expression régulière est une notation plus concise
  - L'automate permet une manipulation plus aisée



- $\mathcal{L}(G) = \Sigma^*$
- $\mathcal{L}_m(G) = \{a, aa, ba, aaa, aba, bba, \ldots\}$
  c'est-à-dire tous les mots finissant par a

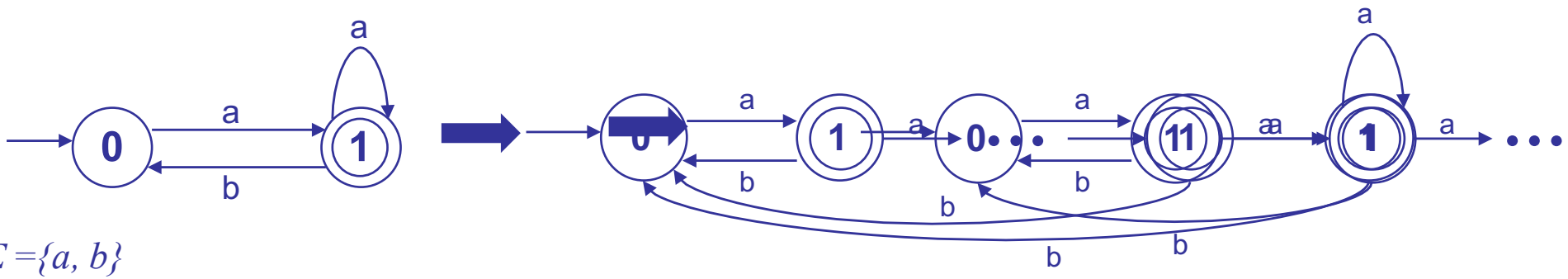
Propriétés

- Tous les langages ne peuvent pas être représentés par des automates à états finis
  - L = {ε, ab, aabb, aaabbb, …} = {$a^n b^n$: n>0} sur l'ensemble Σ = {a,b}



- **Équivalence entre automates**
  - Plusieurs solutions pour construire un automate générant et marquant un même langage
  - Deux automates sont équivalents s'ils génèrent et marquent les mêmes langages :

    G1 et G2 sont équivalents si L(G1) = L(G2) et Lm(G1) = Lm(G2)



- *Σ ={a, b}*
- *ℒ (G) est l'ensemble des mots de Σ* commençant par l'événement a et ne comportant plusieurs occurrences successives de b*
- *$ℒ_m$ (G) est le sous-ensemble de ℒ (G) composé de mots terminant par a*

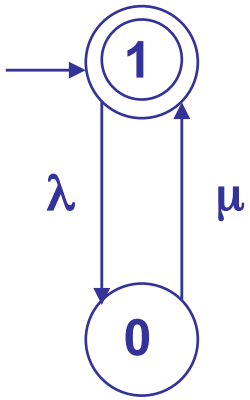
- **Lemmes d'Arden**

  Soit Σ un alphabet et A, B deux langages sur Σ, l'équation X = XA + B où X est un langage sur Σ :

  – si $\varepsilon \notin A$, admet une solution unique $X = B.A^*$

  – si $\varepsilon \in A$, admet un ensemble de solutions $X = \{L.A^* , B \subset L\}$

- **Exemple d'une machine soumise à des pannes**



– 1 : état de marche (état initial et état marqué)
– 0 : état de panne
– λ : panne de machine (événement)
– μ : réparation de machine (événement)
– f(1, λ) = 0, f(1, μ) non définie
– f(0, μ) = 1, f(0, λ) non définie
– $x_0 = 1$ et $x_m = 1$

$L_1 = L_0 \mu + \varepsilon$

$L_0 = L_1 \lambda$

Par substitution $L_1 = L_1\lambda\mu + \varepsilon$

*Règles d'Arden : X = XA + B a pour solution X= BA**

$L_1 = (\lambda\mu)^*$ et $L_0 = (\lambda\mu)^*\lambda$ d'où

$L(G) = L1 + L0 = (\lambda\mu)^*(\varepsilon + \lambda)$

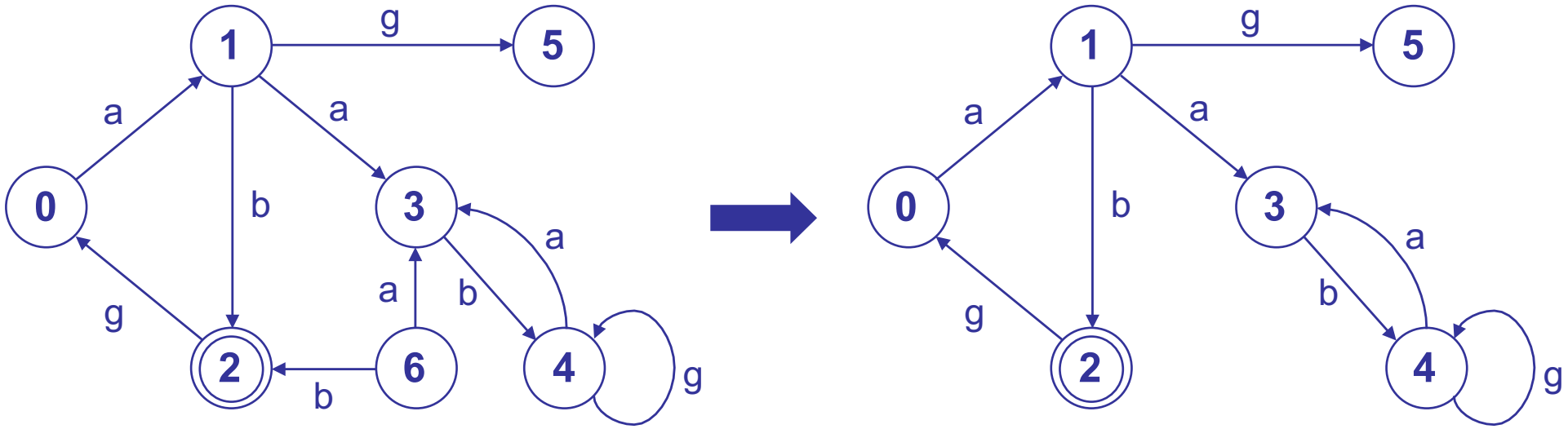
$L_m(G) = L1 = (\lambda\mu)^*$

Opérations sur automates

- Accessibilité
  - L'ensemble des états accessibles est défini par:

$$X_{ac} = \{x \in X : \exists s \in \Sigma^* \ (f(x_0, s) = x)\}$$

  - $Ac(G) = \{X_{ac}, \Sigma, f_{ac}, x_0, X_{ac,m}\}$ avec $f_{ac} = f \mid X_{ac} \times \Sigma \rightarrow X_{ac}$ et $X_{ac,m} = X_m \cap X_{ac}$
  - Les langages $\mathcal{L}(G))$ et $\mathcal{L}_m(G)$ ne sont pas affectés par cette opération

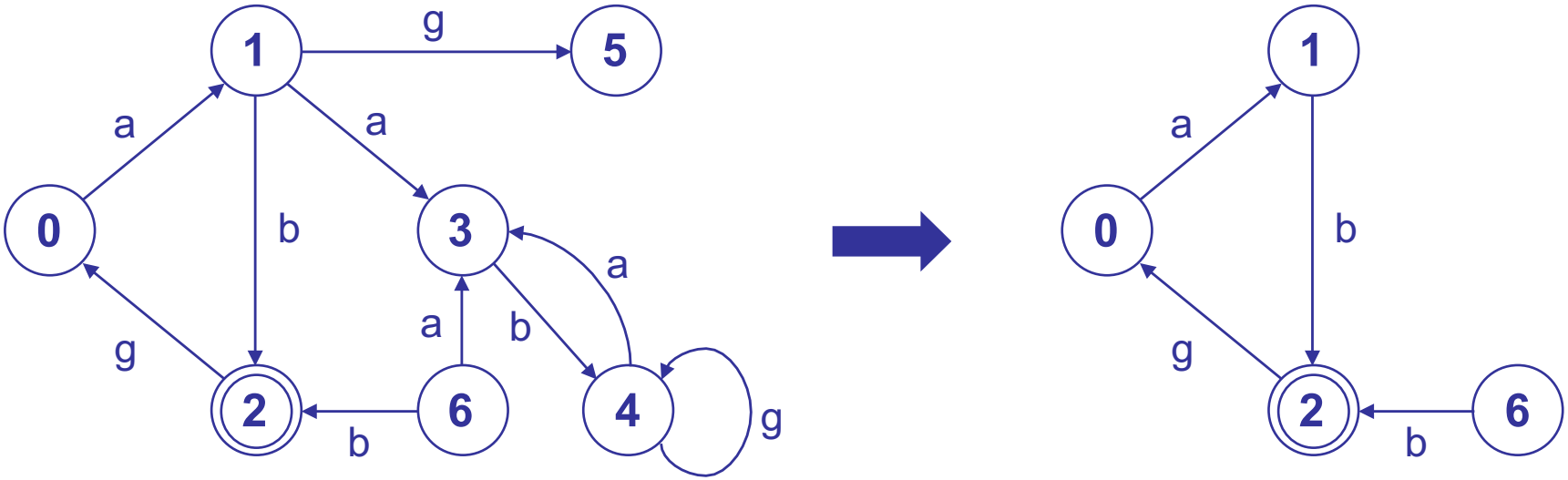


Opérations sur automates

- Co-accessibilité
  - Un état x est dit co-accessible s'il existe une chaîne conduisant à un état marqué qui passe par x: il existe un chemin depuis x vers un état marqué.

$$X_{coac} = \{x \in X : \exists\, s \in \Sigma^* \ (f(x, s) \in X_m)\}$$

  - $CoAc(G) = \{X_{coac}, \Sigma, f_{coac}, x_0, X_m\}$ avec $f_{coac} = f \mid X_{coac} \times \Sigma \rightarrow X_{coac}$
  - Le langage $\mathcal{L}_m(G)$ n'est pas affecté par cette opération *(les états supprimés ne se trouvent pas sur un chemin de x_0 vers un état marqué)*
  - Le langage $\mathcal{L}(G)$ est restreint à $\mathcal{L}(Coac(G)) = \overline{\mathcal{L}_m(Coac(G))}$

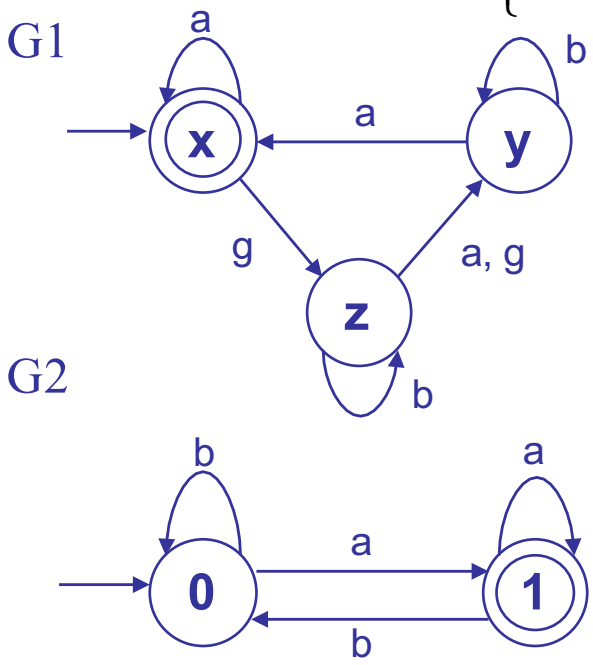


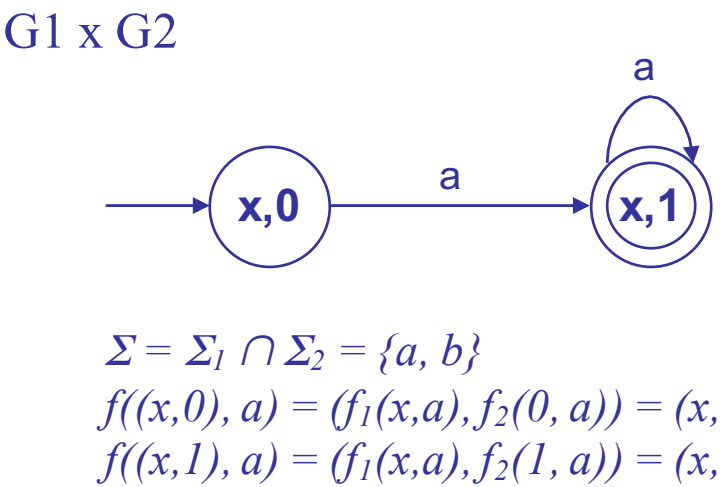
Opérations sur automates

- Produit synchrone
  - Le produit des automates G1 et G2 est l'automate:

$$G1 \times G2 = Ac(X_1 \times X_2,\ \Sigma_1 \cap \Sigma_2,\ f,\ (x_{01},x_{02}),\ X_{m1} \times X_{m2})$$

Où $f((x_1,x_2), e) := \begin{cases} (f_1(x_1,e), f_2(x_2, e)) \text{ si } f_1(x1,e) \text{ et } f_2(x2, e) \text{ sont tous deux définis} \\ \text{indéfini sinon} \end{cases}$





$\Sigma = \Sigma_1 \cap \Sigma_2 = \{a, b\}$
$f((x,0), a) = (f_1(x,a), f_2(0, a)) = (x, 1)$
$f((x,1), a) = (f_1(x,a), f_2(1, a)) = (x, 1)$

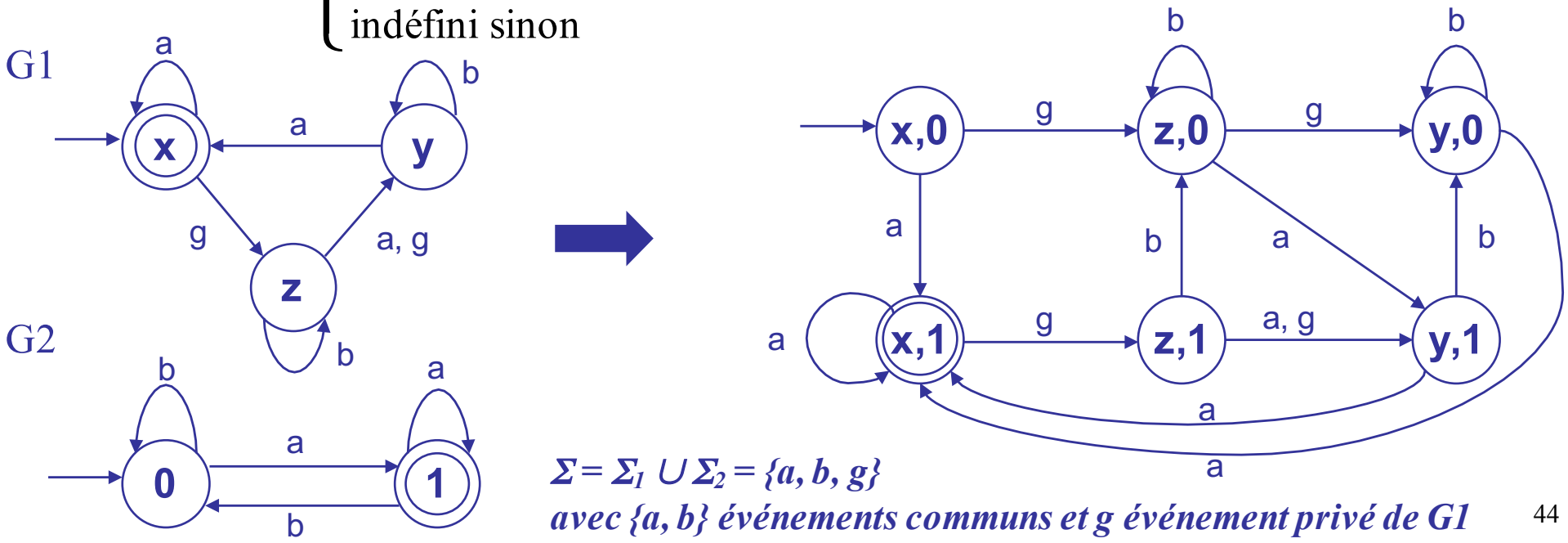
Opérations sur automates

- Composition synchrone
  - La composition synchrone de deux automates G1 et G2 est l'automate:

$$G1 \,||\, G2 = Ac(X_1 \times X_2,\ \Sigma_1 \cup \Sigma_2,\ f,\ (x_{01},x_{02}),\ X_{m1} \times X_{m2})$$

Où $f((x_1,x_2), e) :=$

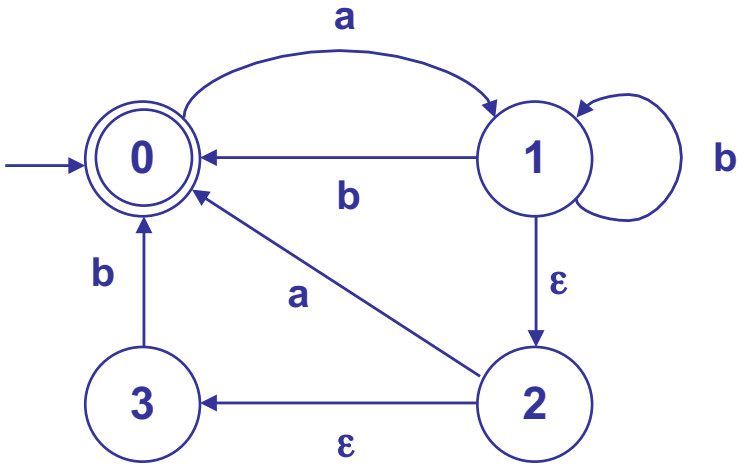
- $(f_1(x_1,e), f_2(x_2, e))$ si $f_1(x1,e)$ et $f_2(x2, e)$ sont tous deux définis
- $(f_1(x_1,e), x_2)$ si $f_1(x_1,e)$ est défini pour $e \in \Sigma_1 \setminus \Sigma_2$ *(événements privés)*
- $(x_1, f_2(x_2,e))$ si $f_2(x_2,e)$ est défini pour $e \in \Sigma_2 \setminus \Sigma_1$ *(événements privés)*
- indéfini sinon



$\Sigma = \Sigma_1 \cup \Sigma_2 = \{a, b, g\}$
avec {a, b} événements communs et g événement privé de G1

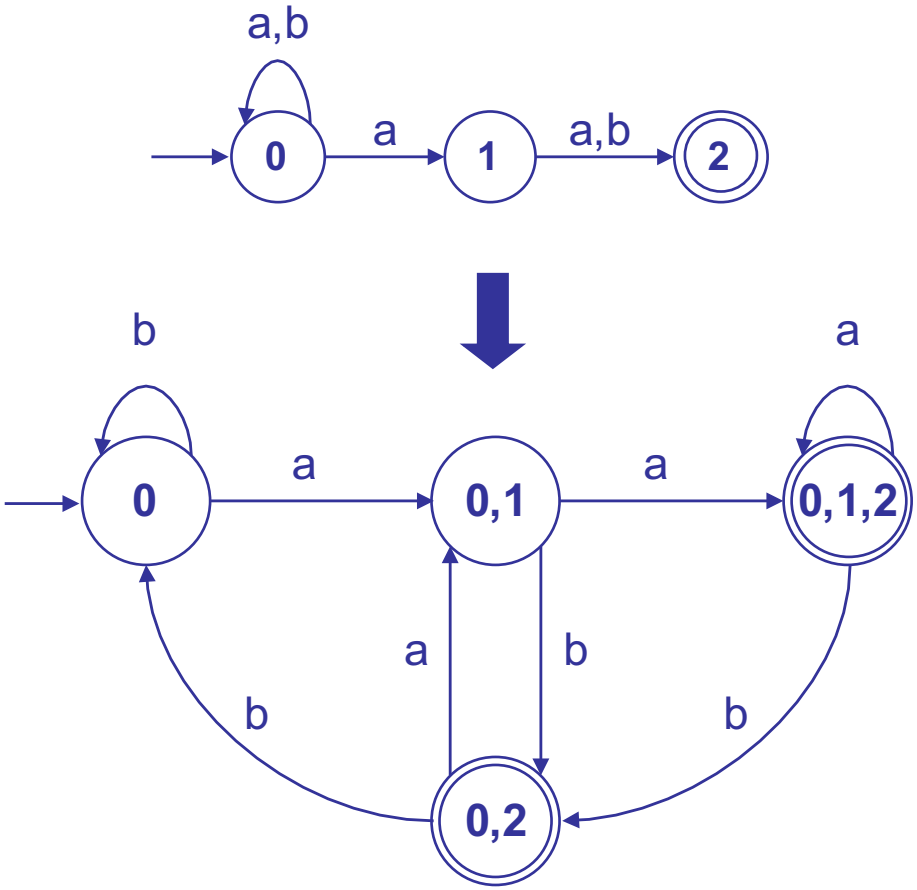
Déterminisme des automates

- Déterministes
  - La fonction f est définie sur $X \times \Sigma \rightarrow X$
  - Un état ne possède qu'au plus une image par f
- Non-déterministes
  - $(X, \Sigma \cup \{\varepsilon\}, f_{nd}, x_0, X_m)$
  - La fonction f_{nd} est définie sur $X \times \Sigma \cup \{\varepsilon\} \rightarrow 2^X$ où 2^X est l'ensemble des sous-ensembles de X
  - Un état peut posséder plusieurs images et une transition entre états peut être associée au mot vide ε



Déterminisme des automates

- Transformation non-déterministe/déterministe
  - *tout automate indéterministe peut être transformé en automate déterministe équivalent (appelé « observateur »)*



Etape 1: on examine l'état 0
- a conduit à l'état 0 ou 1
- b conduit à l'état 0

Etape 2: on examine l'état (0,1)
- a conduit à l'état 0 ou 1 ou 2
- b conduit à l'état 0 ou 2

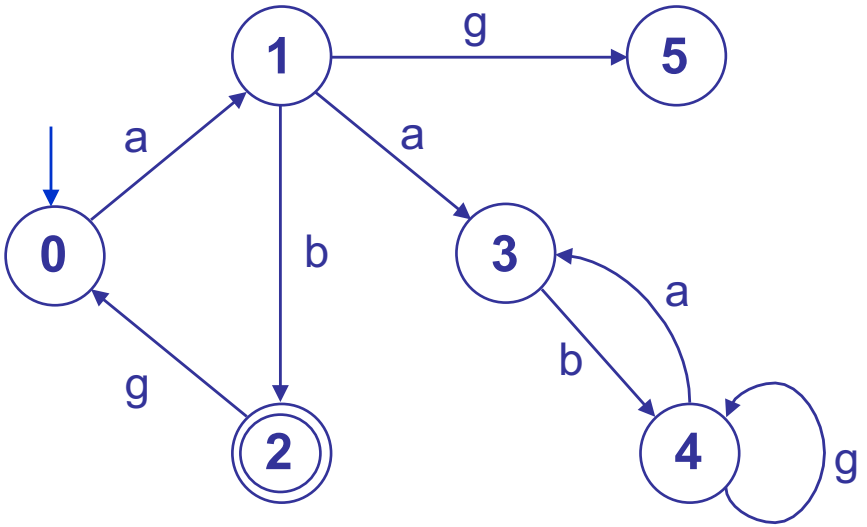
Etape 3: on examine l'état (0,1,2)
- a conduit à l'état 0 ou 1 ou 2
- b conduit à l'état 0 ou 2

Etape 4: on examine l'état (0,2)
- a conduit à l'état 0 ou 1
- b conduit à l'état 0

Blocage des automates

- Deadlock
  - L'automate arrive dans un état x à partir duquel aucun événement de Σ ne peut être sensibilisé
  - $\overline{\mathcal{L}_m(G)} \subset \mathcal{L}(G)$: toute chaîne de $\mathcal{L}(G)$ qui termine sur l'état x ne peut être un préfixe d'une chaîne de $\mathcal{L}_m(G)$
- Livelock
  - L'automate arrive dans un groupe d'états non marqués dont on ne peut ressortir (groupe absorbant)
  - $\overline{\mathcal{L}_m(G)} \subset \mathcal{L}(G)$: toute chaîne de $\mathcal{L}(G)$ qui atteint le groupe d'états absorbant ne peut être un préfixe d'une chaîne de $\mathcal{L}_m(G)$

Par définition, $\overline{\mathcal{L}_m(G)} \subseteq \mathcal{L}(G)$,
l'automate est dit:
- non bloquant si $\overline{\mathcal{L}_m(G)} = \mathcal{L}(G)$
- bloquant si $\overline{\mathcal{L}_m(G)} \subset \mathcal{L}(G)$

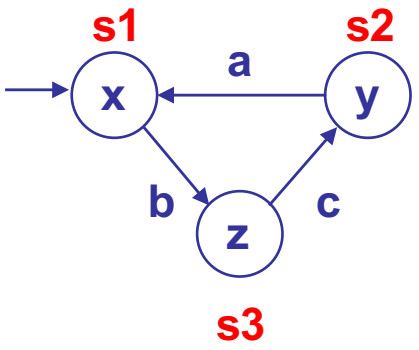


Automates particuliers

- Notions d'événements d'entrées et de sorties
- Machine de Moore
  - Une fonction de sortie assigne un événement de sortie à chaque état

$$X_{t+1} = f(X_t, E_t)$$

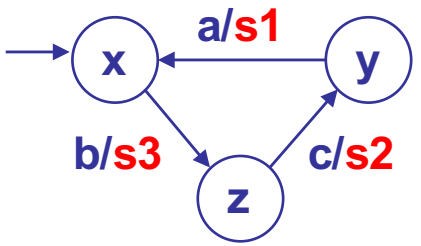
$$Y_t = g(X_t)$$



- Machine de Mealy
  - Transitions associées à des événements de la forme entrée/sortie
  - *Interprétation d'une transition ei/eo depuis l'état x: lorsque l'automate est dans l'état x et qu'il « reçoit » l'événement ei, il évolue vers l'état y et « émet » la sortie eo.*

$$X_{t+1} = f(X_t, E_t)$$

$$Y_t = g(X_t, E_t)$$



SED Pourquoi ?

- **Exemples de Systèmes à Evénements Discrets**
  - **Réseaux de communication** : protocoles
  - **Systèmes manufacturiers** : flux de production, commande des systèmes de production
  - **Systèmes embarqués** : automobile, avionique, ferroviaire
  - **Systèmes hybrides** : continus / discrets


Aéronautique

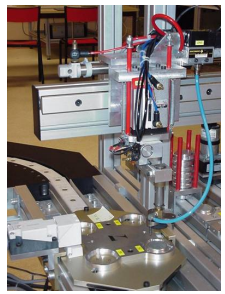

Ferroviaire


Automobile


Militaire


Spatial

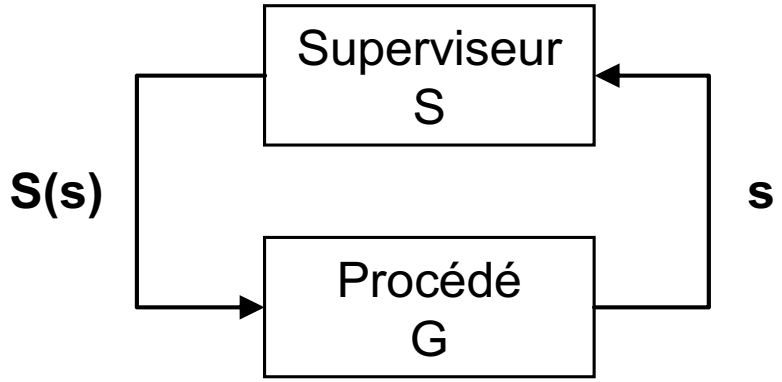

Energie

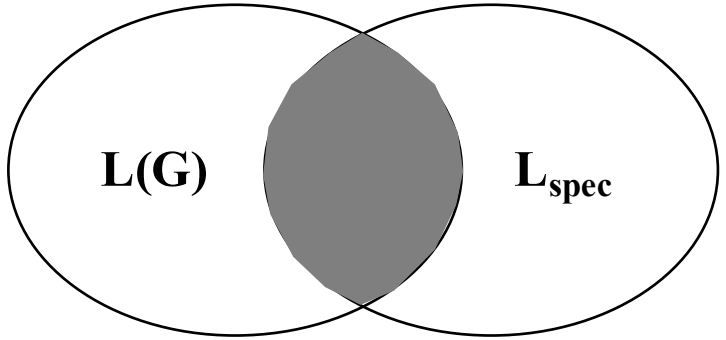

Système de production manufacturier

- **Langages et automates**
  - Opérations de composition/produit sur les automates
    - Systèmes complexes de trop grande taille impossible à modéliser « d'un seul coup »
    - Petits modèles correspondant chacun à un petit sous-système du système complet puis composition pour obtenir le modèle global
  - Identification des langages générés ou marqués : recherche de séquences
    - Trajectoires à suivre pour atteindre un état désiré (conduite de systèmes complexe)
    - Séquences critiques conduisant à un état redouté tel qu'une défaillance

Théorie de la commande supervisée

- Ramadge & Wonham (1987)

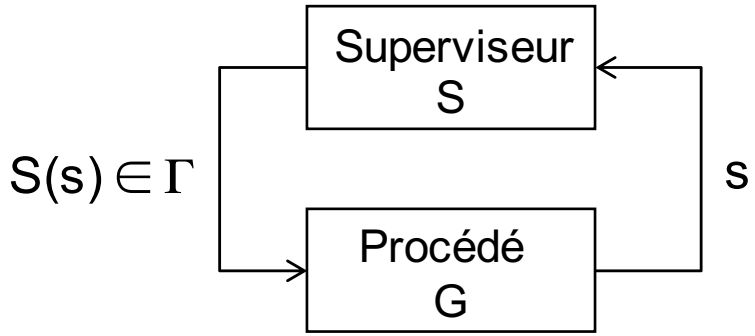




  - Procédé
    - automate G,
    - langage L(G) = comportement non contrôlé
  - Spécification S, langage L_{spec}
  - Objectif du superviseur :
    - Interdire tous les fonctionnements non désirés
    - Garantir au SED le comportement le plus libre possible dans le périmètre du fonctionnement désiré

Superviseur (1/2)

- Un **superviseur** est une fonction
  - $f : L(G) \rightarrow \Gamma \subseteq 2^{\Sigma}$ *(Γ est l'ensemble des commandes admissibles et 2^{Σ} est l'ensemble des sous-ensemble de Σ)*
  - Il autorise ou interdit l'occurrence des événements de manière à satisfaire les spécifications en fonction de la séquence des événements générés par le procédé.



  - Compte tenu de la difficulté à représenter la fonction S (il est difficile en pratique de lister S(s) pour tout $s \in L(G)$), le superviseur peut être « *réalisé* » par un automate à état fini correspondant au langage L(S/G) généré par le système en boucle fermée.

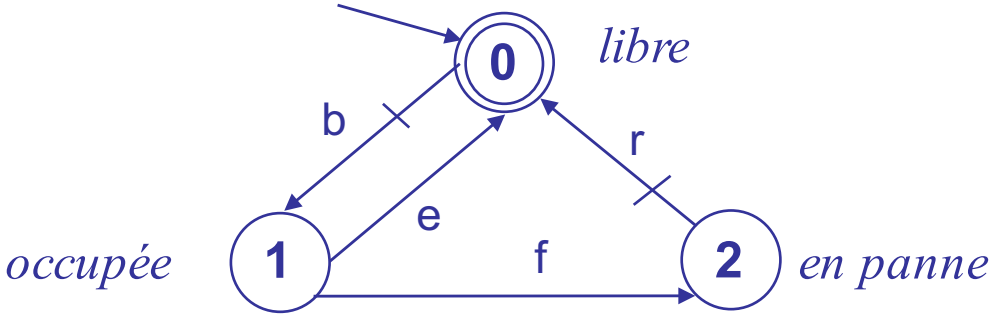
Superviseur (2/2)

- Langage généré par S/G, noté L(S/G): étant donné un procédé G et un superviseur S, le comportement du système en boucle fermée peut être décrit par un automate dont le langage est défini de matière récursive par :
  - $\varepsilon \in L(S/G)$
  - $[(s \in L(S/G)) \,\&\, (s\sigma \in L(G)) \,\&\, (\sigma \in S(s))] \Leftrightarrow [s\sigma \in L(S/G)]$

- L'alphabet Σ est partitionné en deux sous-ensembles
  - Σ_c est l'ensemble des événements contrôlables (c'est-à-dire que le superviseur peut les autoriser ou les interdire)
  - Σ_u est l'ensemble des événements incontrôlables (c'est-à-dire que le superviseur ne peut faire aucune action sur ces événements)

Exemple

- Machine soumise à des pannes
- Spécification = au plus une défaillance



Séquence	Etat	Evts de G	Evts autorisés par S
ε	0	b	b
b	1	e,f	e, f
(be)*	0	b	b
(be)*b	1	e,f	e, f
(be)*bf	2	r	r
(be*)bfr	0	b	-

Contrôlabilité (1/4)

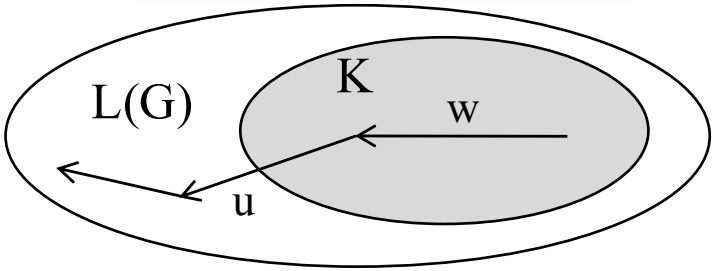
- Etant donné un procédé **G**, quels sont les fonctionnements décrits par un langage **K** avec **K ⊆ L(G)** qui peuvent être accomplis grâce à la supervision ?

- Le concept de contrôlabilité répond à cette question et son étude constitue un préalable à la synthèse d'un superviseur.

- *Définition*: K est **admissible** par L(G) si K ⊆ L(G). Si une spécification L_{spec} donnée n'est pas incluse dans L(G), seul K = L(G) x L_{spec} devra être considéré comme spécification.

- *Définition*: K est **contrôlable** par rapport à L(G) et Σ_u (événements incontrôlables) si

$$\overline{K}\,\Sigma_u \cap L(G) \subseteq \overline{K}$$



w : préfixe de K
$u \in \Sigma_u$
K n'est pas contrôlable

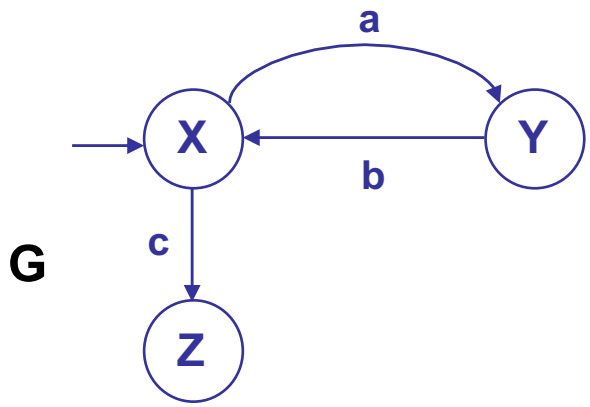
Contrôlabilité (2/4)

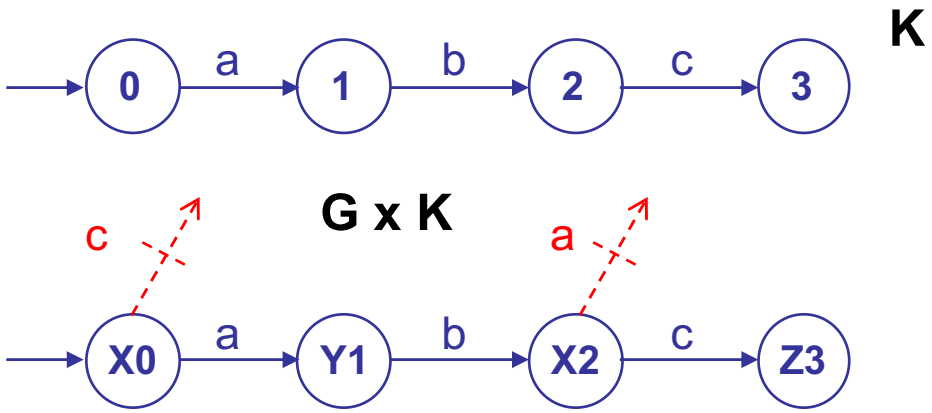
- Exemple : L(G)= (ab)*(ε + a + c); K = abc
  - K est admissible puisque K ⊆ L(G)
  - $\overline{K}$ = (ε + a + ab + abc)
    - K est contrôlable si Σ_u = (b) car

      (ε + a + ab + abc) b ∩ (ab)*(ε + a + c) = ab
    - K n'est pas contrôlable si Σ_u = (c) car

      (ε + a + ab + abc) c ∩ (ab)*(ε + a + c) = c + abc

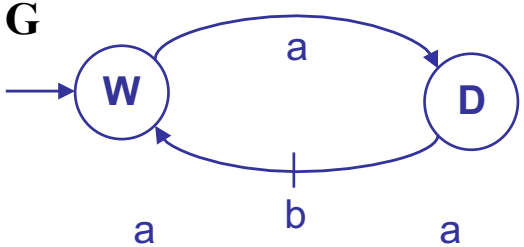
- Vérification de la contrôlabilité de K par rapport à G à partir de l'automate
  - Calcul du produit G x K et vérifier que K = G x K
  - Vérifier qu'à chaque état de G x K les événements e admis dans G mais interdits dans K sont contrôlables



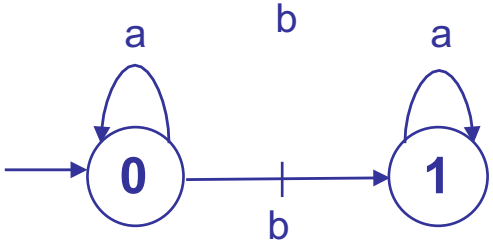


Contrôlabilité : exemple (3/4)

Procédé G



Spec

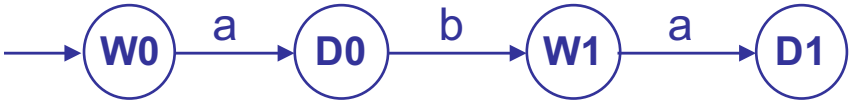


- *L_{spec} admissible pour L(G) ?*

$L_{spec} \not\subset L(G)$: contre-exemple $aa \in L_{spec}$ mais $aa \notin L(G)$
Donc L_{spec} non admissible pour L(G)

- *Quelles spécifications prendre ?*

$K = L(G) \times L_{spec}$ devra être considéré comme spécification



- *Contrôlabilité de K par rapport à L(G) et Σ_u= (a) ?*

$K = a + ab + aba$ $\qquad\qquad$ $\overline{K} = \varepsilon + a + ab + aba$

$L(G) = (ab)^* (\varepsilon + a)$

$\overline{K}\,\Sigma_u \cap L(G) = (\varepsilon + a + ab + aba)\,a \cap (ab)^* (\varepsilon + a) = a + aba$

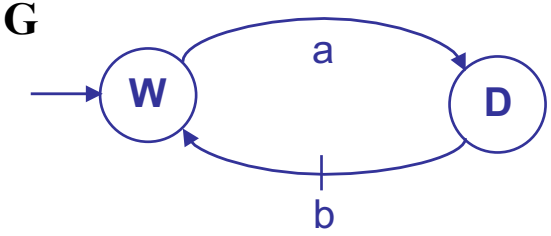
Donc $\overline{K}\,\Sigma_u \cap L(G) \subseteq \overline{K}$ ce qui signifie que K est contrôlable par rapport à L(G) et Σ_u

En fait, il suffit d'interdire b (événement contrôlable) dans D1

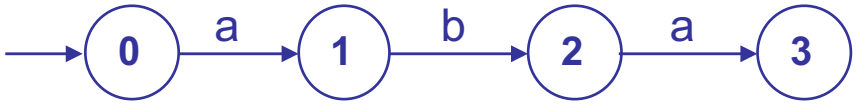
Contrôlabilité : exemple (4/4)

- *Vérification de la contrôlabilité de K par rapport à L(G) et Σ_u= (a) à l'aide de l'automate ?*

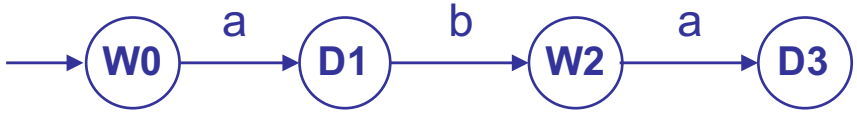
G



K



G x K = G || K= K

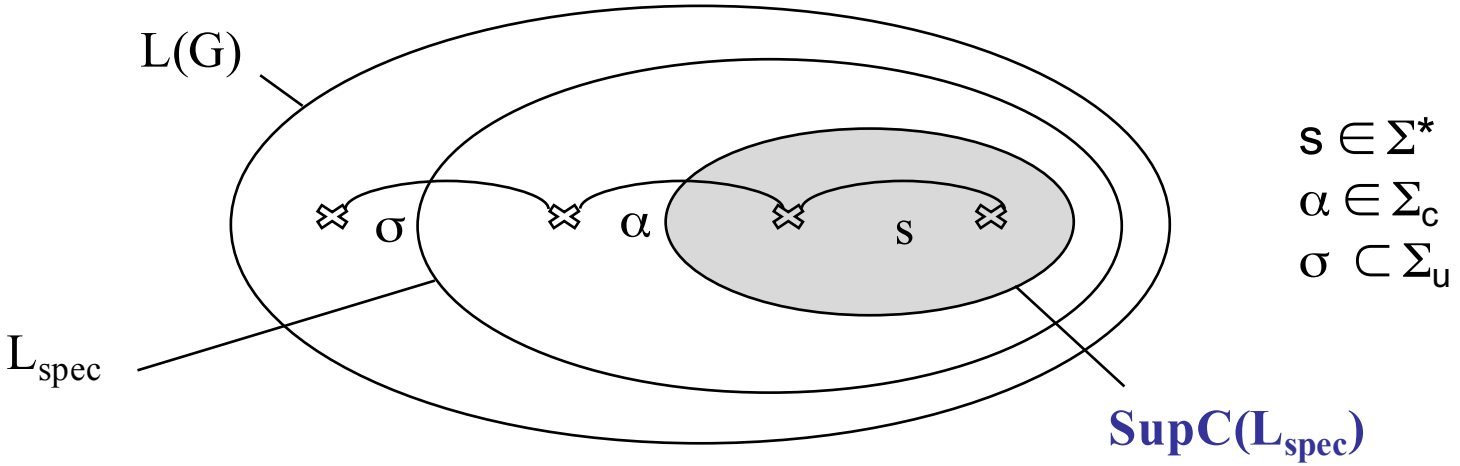


- En W0, D1, W2 il n'existe aucun événement admis dans G et interdit dans K
- En D3, b est admis dans G mais interdit dans K ⇒ K est contrôlable si b est contrôlable

Synthèse de la commande

- Cas 1 : Si L_{spec} est contrôlable par rapport à L(G) alors L_{spec} est le superviseur recherché

- Cas 2 : Si L_{spec} est incontrôlable par rapport à L(G) alors le superviseur est le plus grand langage contrôlable inclus dans L_{spec} noté $SupC(L_{spec})$ (*appelé aussi suprême contrôlable*)



Cas 1 : L_{spec} contrôlable

- **Théorème de contrôlabilité**

  Pour $K \subseteq L(G)$, non vide, il existe un superviseur S, tel que $L(S/G) = K$ si et seulement si:

  1. K est préfixe-clôt
  2. K est contrôlable par rapport à L(G) et Σ_u

  Cette première propriété concerne le langage généré par G sans tenir compte des états marqués

- **Théorème de contrôlabilité non bloquante**

  Pour $K \subseteq L(G)$, non vide, il existe un superviseur S, tel que $L(S/G) = K$ si et seulement si:

  1. K est clôt par rapport à $L_m(G)$, i.e. $K = L_m(G) \cap \overline{K}$
  2. K est contrôlable par rapport à L(G) et Σ_u

Cas 2 : L_{spec} non contrôlable

- Définition

  Soit C(K) l'ensemble des sous-langages contrôlables de K par rapport à L(G) et Σ_u:

  $$C(K) = \{K' : K' \subseteq K, \overline{K'}\ \Sigma_u \cap L(G) \subseteq \overline{K'}\ \}$$

  Propriété : l'union de deux sous-langages contrôlables est contrôlable

- Définition

  Soit le plus grand langage contrôlable de K, aussi appelé le langage suprême contrôlable

  $$SupC(K) = \sup (K' : K' \in C(K)\}$$

  Théorème : SupC(K) existe et est unique

- Principes

  – Rappel : $K \subseteq L(G)$ est contrôlable si pour tout préfixe w de K et $\sigma \in \Sigma_u$ tel que $w\sigma \in L(G)$, $w\sigma$ est un préfixe de K

  – Le suprême contrôlable est obtenu en enlevant des séquences de K, celles pour lesquelles il existe un préfixe w et $\sigma \in \Sigma_u$ tels que $w\sigma \in (L(G) - K)$. Une telle séquence est appelée préfixe non contrôlable

Algorithme de KUMAR

- Hypothèses
  - L(G) et Lspec sont réguliers

- **Pas 1** : Construire le produit synchrone G x K des automates G et K

- **Pas 2** : Rechercher les états défendus de G x K

  Un état défendu (x, q) est tel qu'il existe un événement non contrôlable σ où (x, σ) est admis dans G et (q, σ) interdit dans K.

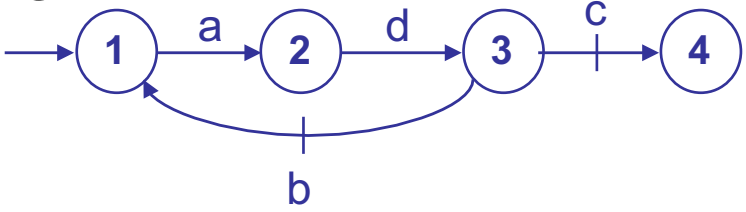
- **Pas 3** : Rechercher les états faiblement défendus du composé G x K

  Un état faiblement défendu (x, q) n'est pas un état défendu mais il existe un mot d'événements non contrôlables dans G x K qui conduit à un état défendu.

- **Pas 4** : Supprimer, dans G x K, les états défendus, les états faiblement défendus, les états non accessibles ainsi que toutes les transitions liées à ces états

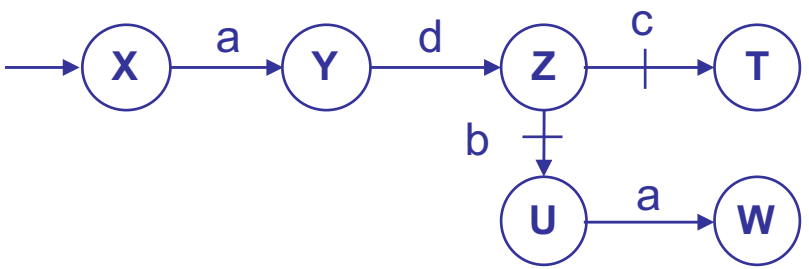
Algorithme de Kumar : exemple 1

G

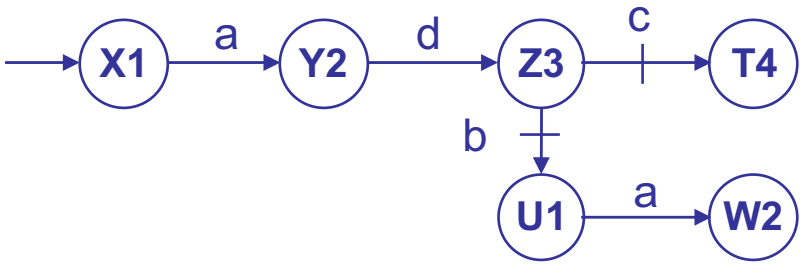


$\Sigma_c = \{b, c\}$
$\Sigma_u = \{a, d\}$

$L_{spec} = \varepsilon + a + ad + adc + adb + adba$

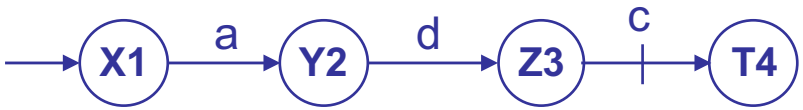


G x L_{spec}



- Etats défendus
  - On cherche des événements non contrôlables admis dans L(G) mais interdits dans L_{spec}
  - W2 : *d* non contrôlable et *d* admis dans G (état 2) mais *d* interdit dans L_{spec} (état W) de K, d'où W2 est un ETAT DEFENDU

S/G

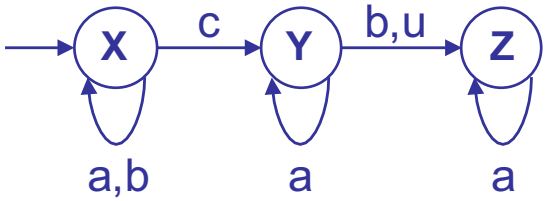


- Etats faiblement défendus
  - f(U1, a) = W2
  - a non contrôlable
  - W2 état défendu

} U1 est un ETAT FAIBLEMENT DEFENDU

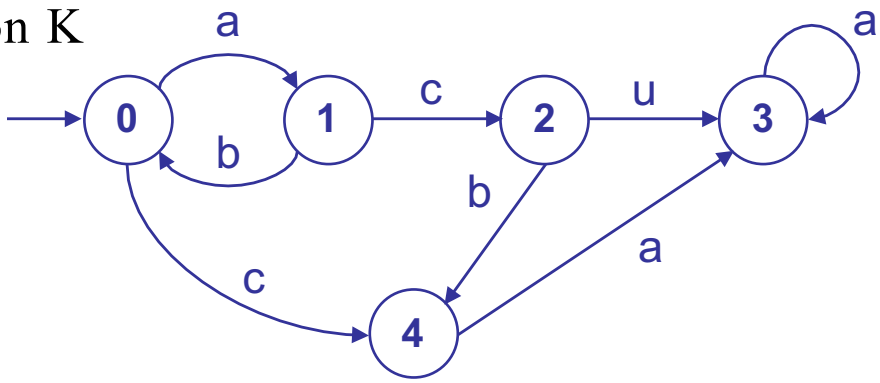
Algorithme de Kumar : exemple 2

Procédé G

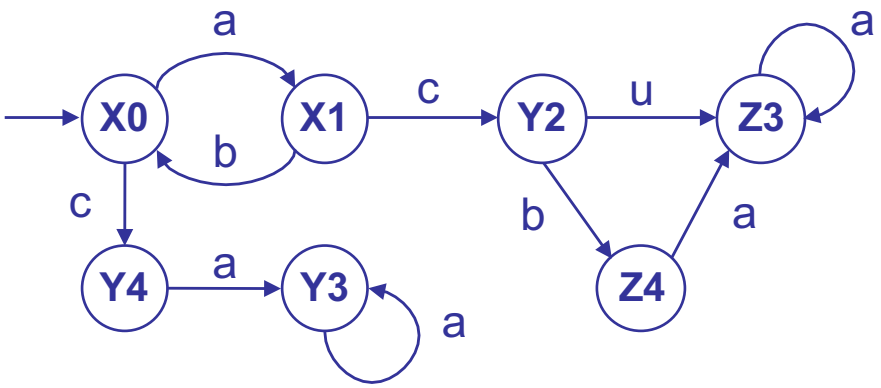


$\Sigma_c = \{a, b, c\}$
$\Sigma_u = \{u\}$

Spécification K

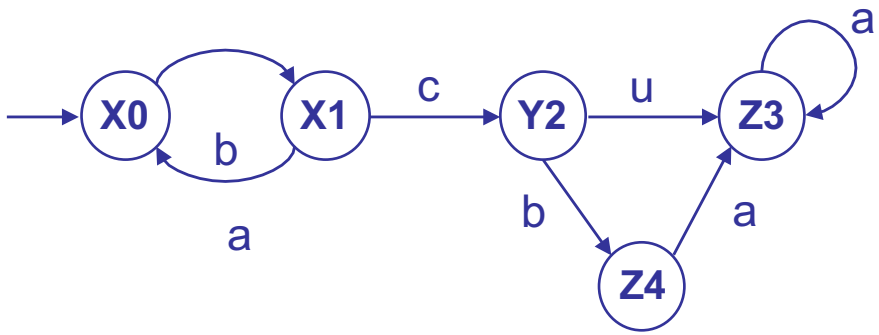


- Vérifier que $L(K) \subseteq L(G)$
- G x K



- Etats défendus
  - On cherche des événements non contrôlables admis dans L(G) mais interdits dans L_{spec}
  - Y3, Y4 : *u* non contrôlable et *u* admis dans G (état Y) mais *u* interdit dans L_{spec} (états 3 et 4) de K,
  - Y3 et Y4 sont donc des ETATS DEFENDUS
- Pas d'états faiblement défendus

S/G



Conclusion

- Modèle efficace pour l'analyse et la commande des SED
  - Sémantique fondée sur la théorie des langages
  - Mécanismes
    - pour la validation: Accessibilité, co-accessibilité, préfixe clôture, …
    - pour la transformation de modèle : produit, composition, …
    - pour la synthèse de la commande des SED
  - Applications industrielles : Modélisation ou Synthèse de la commande des SED (*critiques*), synthèse de systèmes électroniques, …


Aéronautique


Ferroviaire


Automobile


Energie


Militaire


Spatial

- Limites pour la modélisation de systèmes complexes
  - Structures simples : un seul état actif, pas de parallélisme, …
  - Pas de hiérarchie
  - Explosion combinatoire du nombre d'état (composition synchrone)