Chapitre 1
Systèmes à événements discrets:

Pr. Abdelouahed TAJER

Plan

- Système à événements discrets
- Modélisation des SED:
  - Langages formels
  - Automates
  - Exemples
- Commande des SED
  - Théorie de supervision
    - Représentation d'un système à événements discrets
    - Concept de supervision
    - Concept de commandabilité
    - Synthèse d'un superviseur pour la commande
- Conclusions

Définitions générales

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)

Système à Evénements Discrets

- système à espace d 'états discret dont les transitions entre états sont associées à l 'occurrence d 'événements discrets asynchrones

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 Di.

Asynchrone

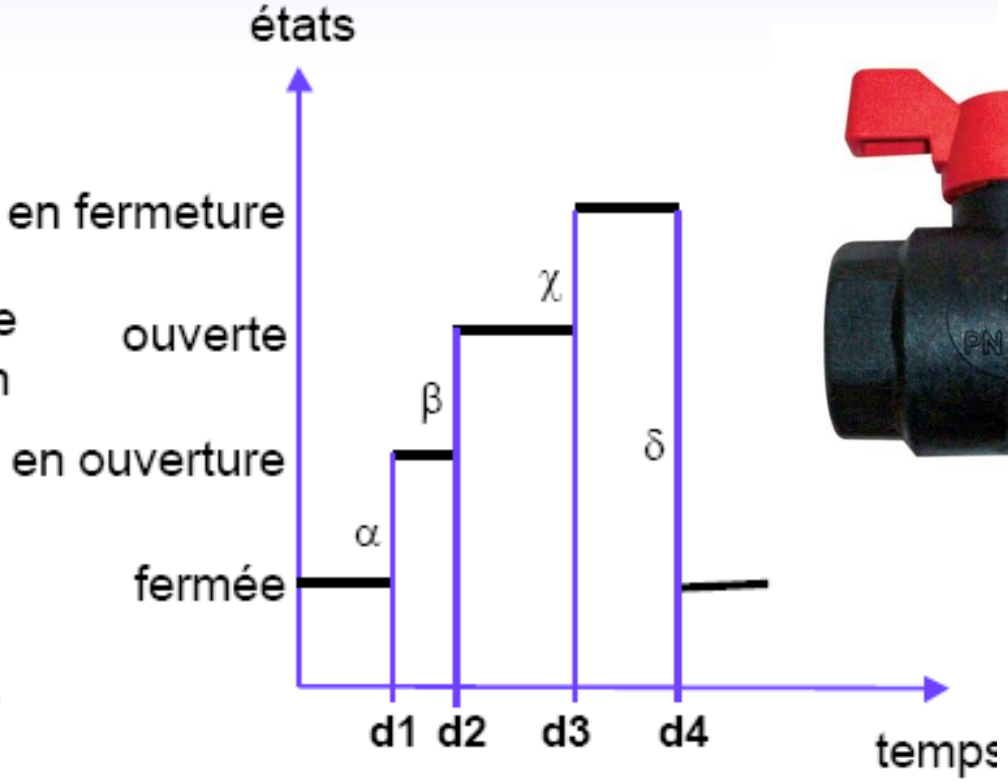
- non cadencé par une horloge

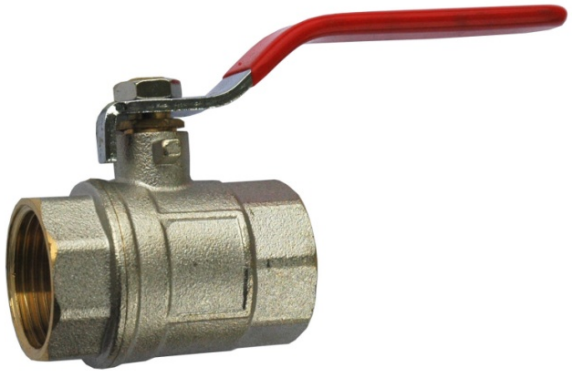
3

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 est représentée par un chronogramme.

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





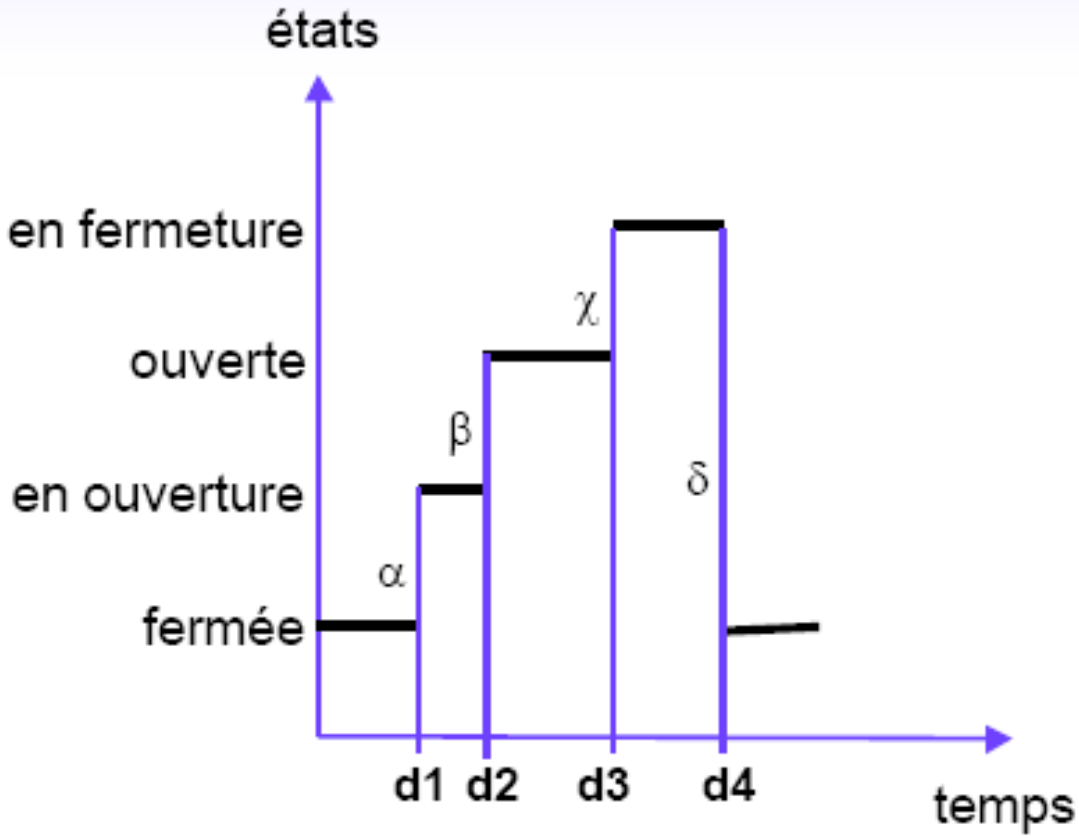
α : événement « début d ’ouverture »
β : événement « fin d ’ouverture »
χ : événement « début de fermeture »
δ : événement « fin de fermeture »

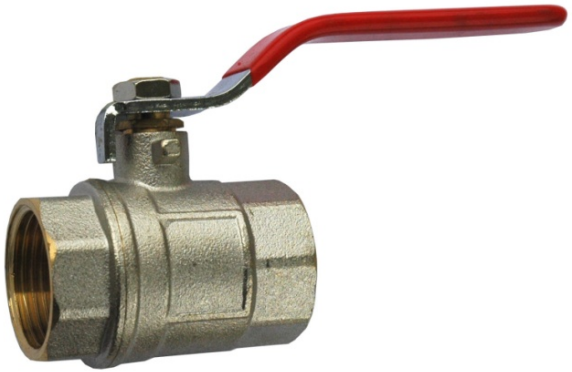
Un exemple basique (2)

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 temporel* formé par la suite ordonnée des couples (α, d1), (β, d2), …
- un *modèle logique* formé de la séquence d 'événements $\{\alpha, \beta, \chi, \delta\}$





Espace des états d'un SED versus espace d 'états d 'un système continu

- **Etat :**
  mémoire minimale du passé d 'un système nécessaire à la détermination de son futur
- **Système Continu**
  - Système dont la dynamique n 'est décrite que par des variables continues du temps
  - l 'état X(ti) d 'un tel système est l 'information nécessaire au temps ti telle que la sortie Y(ti) est uniquement déterminée à partir de cette information X(ti) et de la connaissance du signal d 'excitation u(t) pour t>= ti. L 'espace d 'états est l'ensemble de toutes les valeurs que peut prendre l'état, c'est un *continuum. Un modèle de comportement du système (qui décrit sa* trajectoire dans l 'espace d 'états) est alors :

$$d[X(t)]/dt = f(X(t), u(t), t)\ X(t0) = X0$$
$$Y(t) = g(X(t), u(t), t)$$

- Dans le cas des systèmes linéaires invariants :

à temps continu

$$d[X(t)]/dt = AX(t) + Bu(t)$$
$$X(t0) = X0$$
$$Y(t) = CX(t) + Du(t)$$

à temps discret

$$X(k+1) = FX(k) + Gu(k)$$
$$X(k0) = X0$$
$$Y(k) = CX(k) + Du(t)$$

Système à Evénements Discrets

- SED : système dont la dynamique n'est décrite que par des variables discrètes du temps
  - *l'état du système ne change qu'à certains instants. L'espace d'états est l'ensemble discret de toutes les valeurs que peut prendre l'état du système.*
  - *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.*
  - *Systèmes à espace d'état discret : le vecteur X prend ses valeurs dans un ensemble non dense (Z, N, N*, B, ...). L'espace d'état est Z^n, N^n, B^n, (ou un sous-ensemble non dense), si ce vecteur possède n composantes.*

- Il faut d'autres outils mathématiques que les équations intégro-différentielles pour exprimer l'équation d'état.
  - Théorie des langages réguliers
  - Modélisation par automates à états

Trois niveaux d'abstraction

- **La trajectoire dans l'espace d'états à partir d'un état initial peut être décrite :**
  - uniquement par la logique d'enchaînement des occurrences d'événements; on ne prend en compte que le temps logique
    (SED non temporisés ou logiques)
  - en prenant en compte le temps physique
    (SED temporisés ou temporels)
  - en associant à chaque occurrence d'événement un taux de probabilité
    (SED stochastiques)

- **La suite de ce cours se limite à la première classe.**

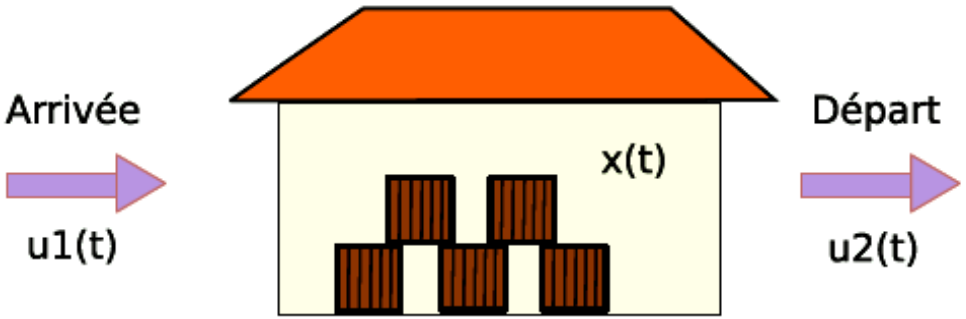
Domaines d'application

- **Nombreux secteurs applicatifs :**
  - Télécommunications
  - Génie logiciel : Bases de Données, Systèmes d'Exploitation
  - Automatisation industrielle
  - Systèmes de production et de transport
  - Systèmes embarqués et mécatroniques

- **Globalement :**
  - Gestion de flux physiques ou informationnels
  - Commande de systèmes physiques (automatisation) ou de traitement de l'information (transactions sur SGBD, systèmes d'exploitation multi-tâches)

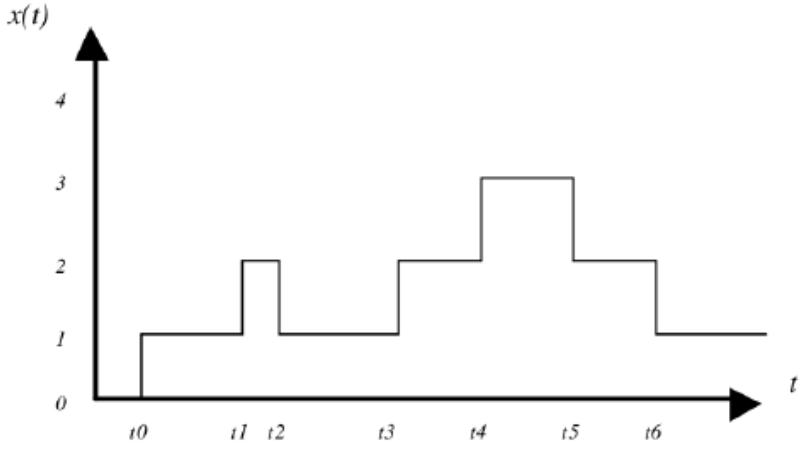
Exemple : Système discret par nature

Entrepôt



- $x(t)$ nombre de caisses dans l'entrepôt à l'instant t
- $u1(t) = 1$ si une caisse arrive au temps t, 0 sinon
- $u2(t) = 1$ si un caisse part au temps t, 0 sinon

Si u1(t) alors x(t) = x(t-1) +1
Si u2(t) alors x(t) = x(t-1) -1

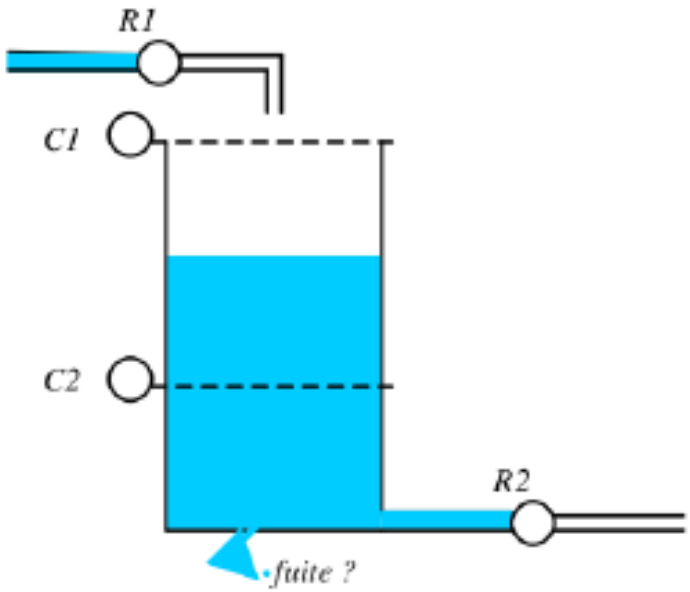


Systeme discretisable (1)

- Système continu mais modèle discret
  - Selon la tâche a effectuer sur le système
  - Modèle qualitatif
    - Exemple : le réservoir

Systeme discretisable (2)

- Récipient



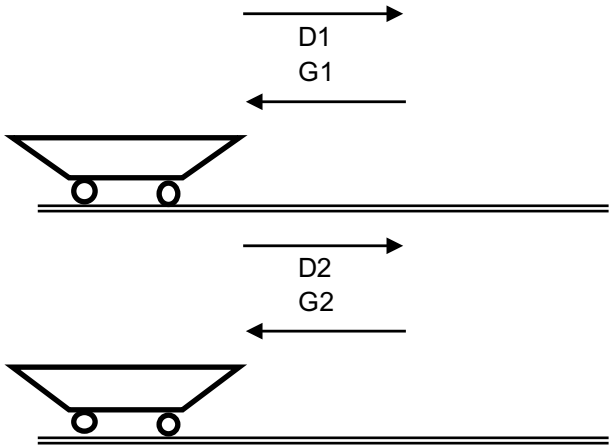
R1 : {ouvert, fermé}
R2 : {ouvert,fermé}
C1 : {niv_atteint,niv_pas_atteint}
C2 : {niv_atteint,niv_pas_atteint}

Espace d 'états discrets :
R1 x R2 x C1 x C2

Exemples d'événements :
Ouverture de R1
Fermeture de R2
Le niveau passe au dessous de C2
Ca déborde!!

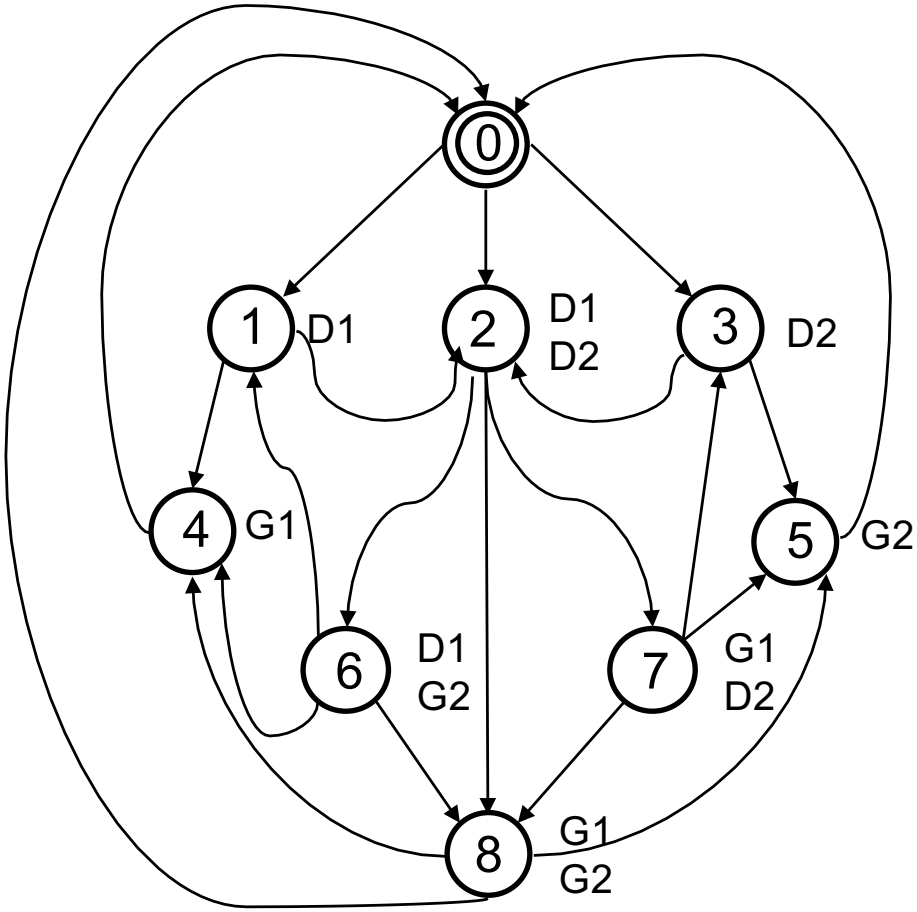
- Détection d 'une fuite :
  - *R1=fermé, R2=fermé, C2=niv_atteint*
  - *événement : « le niveau devient inférieur à C2 »*

Exemple élémentaire : modélisation par automate à états des évolutions d'un système physique



D1 : le chariot 1 se déplace vers la droite
G1 : le chariot 1 se déplace vers la gauche
Idem en 2

- Etiquettes associées aux transitions non représentées
- Automate non déterministe
- Conclusions (exhaustivité, pouvoir d'expression, complexité)



Automate décrivant les mouvements des chariots (vue partielle)

Choix du formalisme de modelisation

- Avant de modéliser :
  - quelle est la nature de mon système ?
    - monolithique, distribue, continu, discret
  - quel est l'objectif de la commande ?
    - Prendre la décision, surveillance..
  - quel est le degré de connaissance de mon système ?
    - connaissance de surface, comportement nominal, comportement de panne
- Choix du formalisme de modélisation
  - Dépend de la réponse aux questions précédentes
- De nombreux formalismes sont possibles
  - **Langage,** Système de transitions (**automate**),, Règles, Réseau de Petri, Grafcet.

Théorie des langages et des automates: Langage formel

Objectif:

Formaliser et/ou modéliser les systèmes à événements discrets

Outils: Langages et automates

• Evénements: ensemble de tous ces symboles **(σ1 , …, σn)** est fini et constitue un alphabet noté Σ.

•Toutes les séquences finies d'événements, ou trace, peuvent alors être représentées par une séquence de symboles ***s* = σ1. σ2 …** appelée chaîne (ou mot) sur l'alphabet **Σ**.

•On appelle alphabet (**ou vocabulaire**), un ensemble fini de symboles noté **Σ**.

Théorie des langages et des automates: Langage formel

Dans le cas d'un **SED**, l'alphabet pourra représenter l'ensemble des événements possibles dans le système. Cet ensemble est composé de tous les événements qui font évoluer le **SED**.

Une *chaîne* (ou *mot* ou *séquence*) définie sur un alphabet Σ est une suite finie d'éléments de Σ **notée *s***. La longueur d'une chaîne *s*, notée |*s*|, représente le nombre de symboles de *s*

(par exemple, |abba| = 4).

- **Evénement → symbole σ ∈ Σ**
- **Trace → séquence *s***
- **Comportement du système → langage *L***

Soit **Σ*** l'ensemble de toutes les chaînes qui peuvent être formés sur les éléments de Σ, y comprises la chaîne vide ε.

Langages formels

- $\sum$: *Ensemble fini des événements générés par un processus* (ex:$\sum$={a,b,c})
- $\sum^*$: *Ensemble de toutes les séquences d'événement générées à partir de* $\sum$

  (ex : $\sum^*$={ε,a,b,c,ab,ac,bc,abc,ca,….}); ε: chaine (mot) vide

- Si v ∈ $\sum^*$, w ∈ $\sum^*$ et v=uw alors ***u est préfixe de v***
- *L* : Ensemble de tous les chemins admissibles L ⊂$\sum^*$ (ex : {a,abc,ab})
  - L: langage défini sur un alphabet $\sum$ tout sous ensemble de $\sum^*$
  - L= Ø le langage vide
- $\overline{L}$: Ensemble des préfixes de L (ex : $\overline{L}$ ={a,ab,abc})
- *Si L =* $\overline{L}$ *alors* ***L est préfixe fermé***
  (ex :L={ab} → $\overline{L}$ *={a,ab} L non fermé L={a,ab}* → $\overline{L}$ *={a,ab} L à préfixe fermé)*
- Comportement d'un SED : Langage est à préfixe fermé sur l'alphabet $\sum$

Opération de base sur les langages

- Soient L1={a, b, ab} un langage défini sur $\sum_1$= {a, b} et L2 = {b, cb} un langage défini sur $\sum_2$= {b, c} . On défini les opérations suivantes.
  - **L'union**

    le langage contenant toutes les chaines contenues soit dans L1 soit dans L2.

    $L1 \cup L2 = \{v/\ v \in L1 \text{ ou } v \in L2\}$

    Ex: $L1 \cup L2 = \{a, b, ab, cb\}$
  - **L'intersection**

    $L1L2 = \{v/\ v \in L1 \text{ et } v \in L2\}$

    Ex: $L1 \cap L2 = \{b\}$
  - La différence

    $L1\text{-}L2 = \{v/\ v \in L1 \text{ et } v \notin L2\}$

    Ex: L1 - L2 ={a, ab}
  - **La concaténation**

    Notée L1.L2 est un langage contenant toutes les chaines formées d'une chaine de L1 suivie

Expressions régulières

- Une **expression régulière** sur un alphabet ∑ est composée d'opérandes et d'opérateurs pris dans l'ensemble {+, ., *}.
- Une **expression régulière** définit un langage sur ∑ quand chaque symbole x de ∑ est identifié au langage {x} et quand les opérateurs +, ., * sont associés respectivement aux opérateurs d'union, de concaténation et de fermeture itérative. Un tel langage est appelé **langage régulier**
- **Opérateur**
  - Les opérateurs sont classés du plus prioritaire au moins prioritaire, dans l'ordre fermeture itérative, concaténation et union (*, ., +)
    - () établies une priorité d'ordre supérieur dans l'analyse de l'expression régulière
    - Ex:

      a.b* définit le langage {a, ab, abb, abbb, …..}

      (a.b)* « » » » » » {∈, ab, abab, ababab, ….}

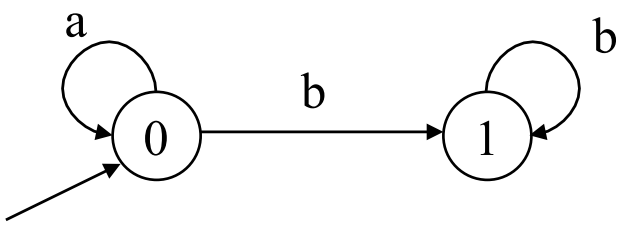
      a+b « » » » » » {a} {b} = {a, b}

      a+b* « » » » » » {a} {∈, b, bb,…} = {a, b, bb,…}

Du langage vers l'automate fini

- Un **automate** est une machine à états
- Un **automate déterministe** ne comporte pas d'états à partir duquel un même événement permet d'atteindre 2 états différents
- Ex: Soit $\sum = \{a, b\}$ et **l'expression régulière** $(a + b)^*b$.

Le langage *L* peut être engendré par l'automate fini de la figure.
(Q, ∑, δ,q0) où Q={0,1} ∑={a,b} q0= 0 δ={(0,a,0);(0,b,1);(1,b,1)}

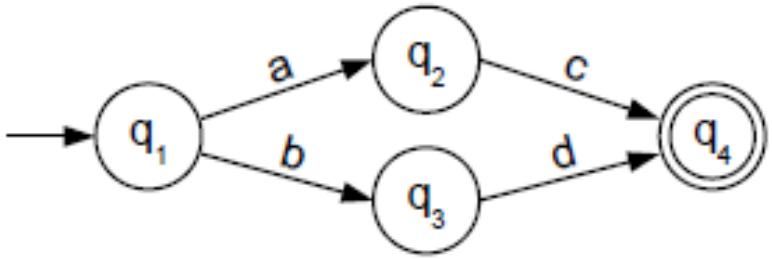


Du langage vers l'automate fini

- Automate à états finis
  - **Finite State Automaton (FSA)**
  - Machine abstraite, issue des travaux de **A. Turing**
- Éléments de l'automate
  - Une ensemble fini d'**états** possibles
  - Un ensemble fini de **symboles** en entrées
  - Une fonction de **transition** entre états
- Représentations
  - **Diagrammes de transition**
  - **Tables de transition**
  - **Notations formelles**

Du langage vers l'automate fini

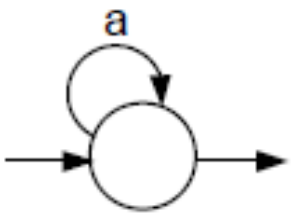
- Diagrammes de transition
  - Représentation « graphique »
  - Graphe orienté étiqueté :
    - **Nœuds** : **états** de l'automate
      - Étiqueté par les noms des **états** (généralement q_x : q_1, q_2, q_3, ...)
      - État **final** : **double cercle**
    - **Arcs orientés** : **transitions** de l'automate
      - Étiqueté par les **symboles** (a, b, c, d, ...)
      - État **initial** : marqué par un **arc sans nœud d'origine**



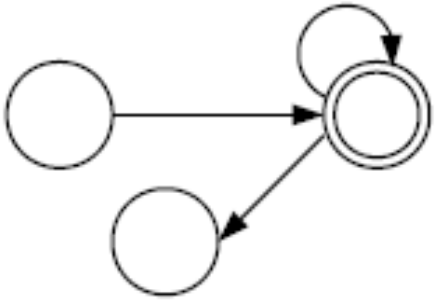
Du langage vers l'automate fini

- Diagrammes de transition (suite)
  - Pour la lisibilité, lorsque deux arcs de même orientation sont possibles entre deux nœuds, ils sont fusionnés (**disjonction**) :

  - Un arc peut « boucler » un état **sur lui-même** :

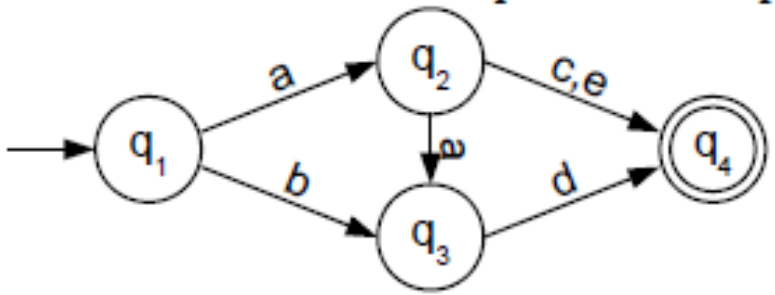


  - Des transitions peuvent **partir de l'état final** :



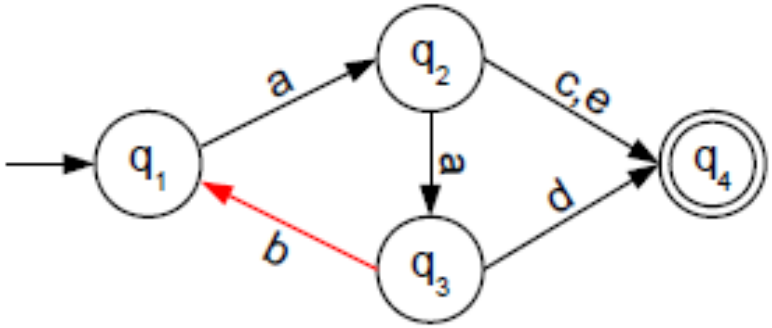
Du langage vers l'automate fini

- Diagrammes de transition (suite)
  - Les **transitions** correspondent à la **concaténation**
  - On « suit » les arcs pour voir quel langage est accepté :



L = {ac, ae, bd, aad }

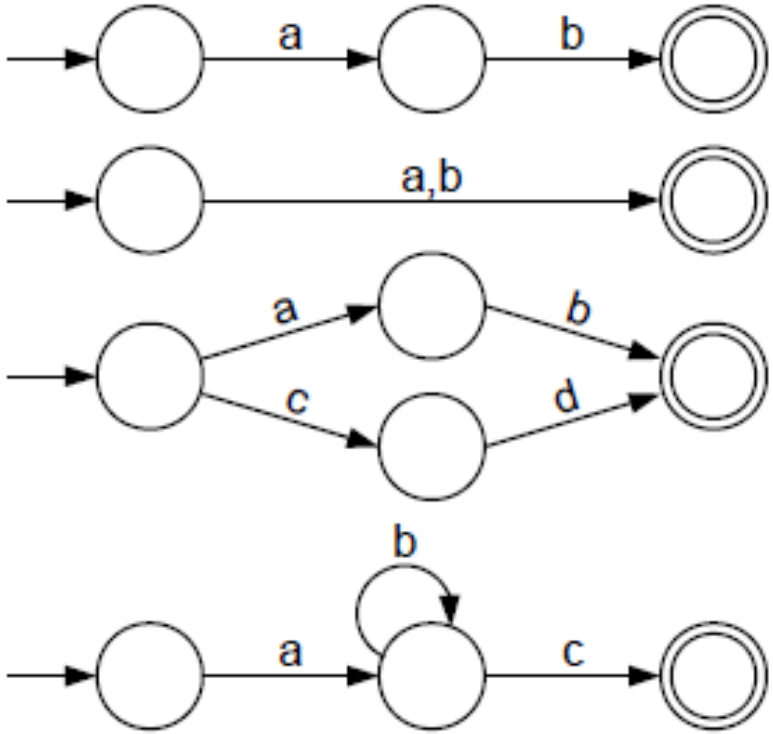
  - Le langage n'est pas forcément fini :



L = {ac, ae, aad, aabac, aabae, aabaad, aabaabac, aabaabae, aabaabaad ... }

Du langage vers l'automate fini

- Diagrammes de transition (suite)
  - Peuvent être mis intuitivement (formellement aussi) en correspondance avec des expressions régulières



R = ab

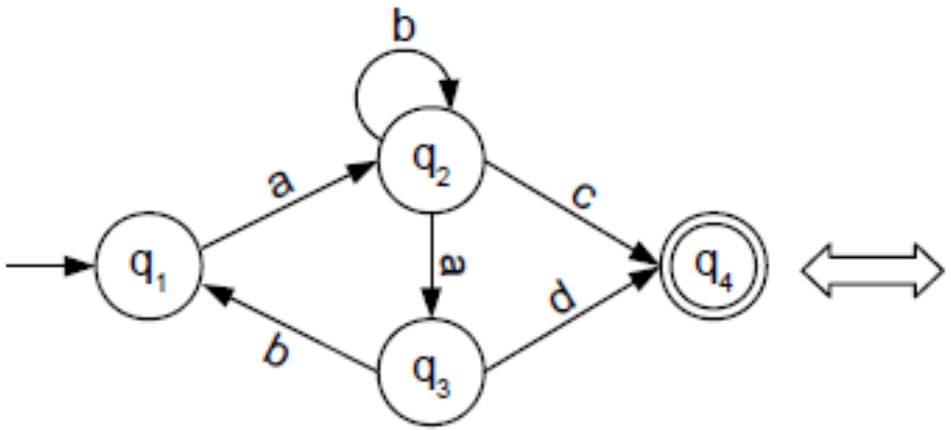
R = a+b

R = ab+cd

R = ab*c

Du langage vers l'automate fini

- Table de transitions
  - **Équivalente** à la représentation par diagramme de transitions
  - **États** en **lignes**, **symboles** en **colonnes**
    - Ligne de l'état **initial** marqué par une **flèche →**
    - Ligne de l'état **final** marqué par une **étoile ***
  - **Contenu** décrit les **transitions** depuis un état / symbole



		a	b	c	d
→	q_1	q_2	∅	∅	∅
	q_2	q_3	q_2	q_4	∅
	q_3	∅	q_1	∅	q_4
*	q_4	∅	∅	∅	∅

Du langage vers l'automate fini

- Notations pour un automate « **déterministe** » **(DFA)**
  - Un ensemble **fini** d'**états**
    - Notation : $Q = \{q_1, q_2, q_3...\}$
  - Un ensemble **fini** de **symboles** (**alphabet**)
    - Notation : $\Sigma = \{a, b, c, d...\}$
  - Une **fonction de transition** qui prend en paramètre un état et un symbole et qui renvoie un état
    - Notation : $\delta(q_i, a) = q_j$ (« signature » $\delta : Q \times \Sigma \rightarrow Q$)
  - Un **état initial**
    - Notation : $q_0 \in Q$
  - Un ensemble d'**états finaux**
    - Notation : $F \subseteq Q$

Du langage vers l'automate fini

- **Automate à états finis**
  - Défini comme le quintuplet : $A = (Q, \Sigma, \delta, q_0, F)$
  - Définition **équivalente** aux diagrammes et tables de transition
  - Repose sur la définition de la **fonction de transition δ**, souvent définie par extension (en listant les cas possibles)
    - La fonction peut renvoyer Ø (par ex. $\delta(q_1, d) = \emptyset$)
  - Définition de la **fonction de transition « étendue »**
    - Fonction qui prend un mot en entrée et utilise δ de l'automate
    - Notation ($\alpha \in \Sigma^*$) : $\overline{\delta}(q_i, \alpha) = q_j$ (« signature » $\overline{\delta} : Q \times \Sigma^* \rightarrow Q$)
    - Définie **récursivement** par **décomposition** du mot w :
      - Si $w = \varepsilon$ alors $\overline{\delta}(q_i, \varepsilon) = \{ q_i \}$
      - Si $w = x.a$ tels que et $x \in \Sigma^*$ et $a \in \Sigma$ alors $\overline{\delta}(q_i, w) = \delta(\overline{\delta}(q_i, x) , a)$
- **Langage reconnu** par un automate déterministe
  - Ensemble des mots tels que la fonction de transition étendue appliquée à l 'état initial et au mot conduit à un état final :
    - $L(A) = \{ w \in \Sigma^* \mid \overline{\delta}(q_0, w) \in F \}$
  - **Théorème de Kleene** : par définition, le langage reconnu par un automate à états finis (NFA) est **régulier** (ou rationnel)

Du langage vers l'automate fini

- Types d'automates
  - Automates à **états finis**
    - **FSA** : **F**inite **S**tate **A**utomata
  - Automates à états finis **déterministes**
    - **DFA** : **D**eterminist **F**inite state **A**utomata
  - Automates à états finis **non-déterministes**
    - **NFA** : **N**on-determinist **F**inite state **A**utomata
  - Automates à états finis **non-déterministes** avec possiblité de transitions ε (mot vide)
    - **ε-NFA** : **N**on-determinist **F**inite state **A**utomata with **ε** transitions
- Tous peuvent être représentés à l'aide des diagrammes de transition, tables de transition, notations formelles

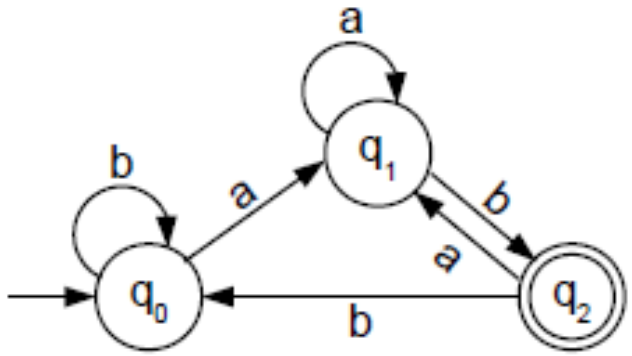
Du langage vers l'automate fini

- **Déterminisme** : requiert que l'automate soit dans un état **unique** (« déterminé ») après avoir lu n symboles
- Automates **non-déterministes** (**NFA**)
  - Non-déterminisme : l'automate peut-être dans **plusieurs états**, « **simultanément** » après avoir lu n symboles
    - **Attention** : un automate non-déterministe n'implique pas que l'on ne « sache » pas dans quels états il est (au contraire)
  - Particulièrement utile lorsque l'on ne sait pas « à l'avance » quel état final l'automate atteindra (**hypothèses**)
    - Facilité de représentation / programmation
      - GPS : une voiture se dirige vers Paris, mais on ne connaît pas la destination finale (Orléans, Strasbourg)
      - Prédiction SMS : un utilisateur tape le début d'un mot, mais plusieurs hypothèses sont possibles pour le mot qu'il veut écrire

Du langage vers l'automate fini

- Différence de représentations **DFA / NFA** (suite)
  - Soit Σ = { a, b }, comment représenter un automate qui reconnaît tous les mots se terminant par ab ?

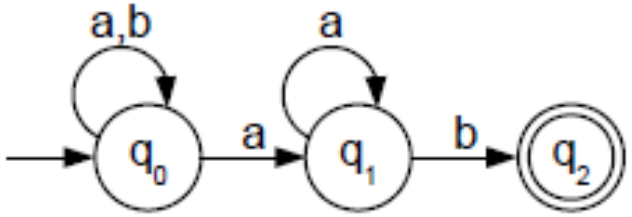
DFA



$\Leftrightarrow$

		a	b
→	q_0	q_1	q_0
	q_1	q_1	q_2
*	q_2	q_1	q_0

NFA

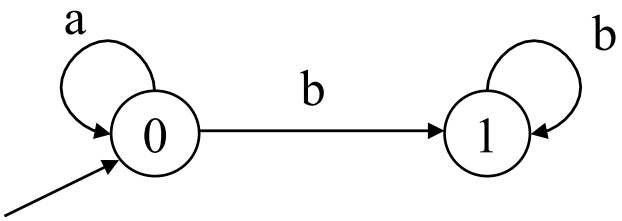


$\Leftrightarrow$

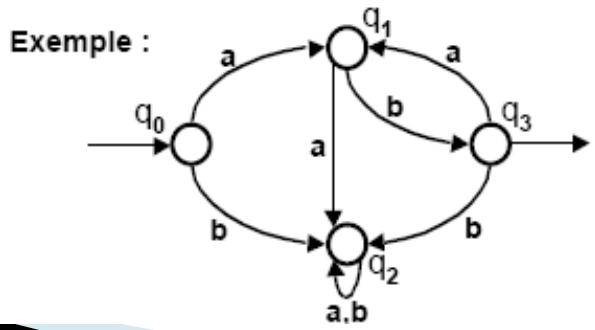
		a	b
→	q_0	$\{ q_0, q_1 \}$	$\{ q_0 \}$
	q_1	{ a }	$\{ q_2 \}$
*	q_2	Ø	Ø

Du langage vers l'automate fini

- Un **automate** est une machine à états
- Un **automate déterministe** ne comporte pas d'états à partir duquel un même événement permet d'atteindre 2 états différents
- Ex: Soit ∑ = {*a*, *b*} et **l'expression régulière** (*a* + *b*)**b*.
  Le langage *L* peut être engendré par l'automate fini de la figure.
  (Q, ∑, δ,q0) où Q={0,1} ∑={a,b} q0= 0 δ={(0,a,0);(0,b,1);(1,b,1)}



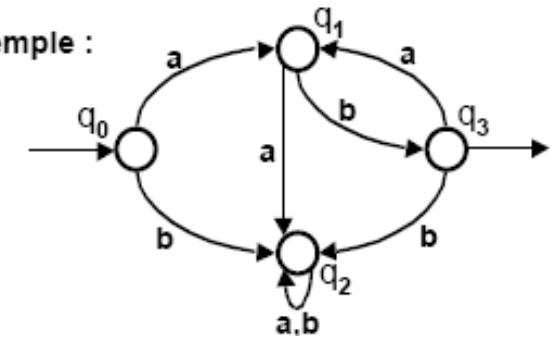
- un automate est un quintuplet : A = (Q, ∑, δ , q0, F) où :
  - Q est l 'ensemble des états
  - ∑ est un alphabet d 'entrée
- δ est la fonction de transition d 'états définie de Qx ∑→ Q qui associe un état de départ et un symbole d 'entrée à un étatd 'arrivée
- q**0 est l 'état initial**
- F est l 'ensemble des états finaux (marqués)(F ⊂ Q)



Q = {q**0, q1, q2, q3 }
∑ = {a,b}
F = {q**3 }**
d est représentée par les arcs associés à des symboles de l 'alphabet
δ (q0, a) = q1
…….

Automate à état fini

Exemple :



- **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

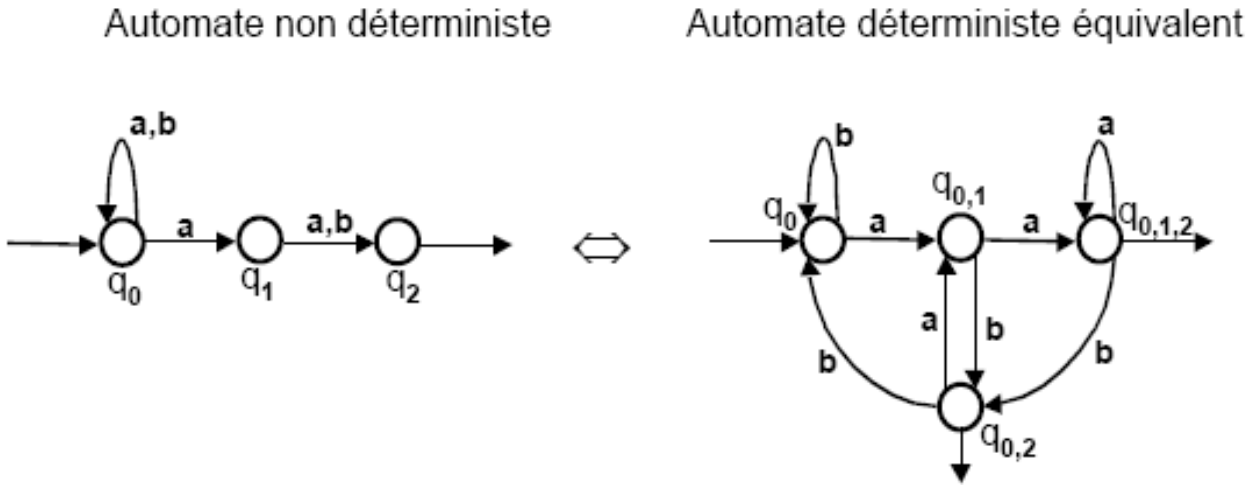
- **Complétude**

  un automate est dit complet lorsque "q Q , " ? ∑ , (q, ?)= q '

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

Automate à état fini

- **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.



Modélisation de SED à l 'aide d 'automates

- Générateurs :
  - Tout Langage régulier peut être présenté par un automate fini

Générateur G= (Q, ∑, ♥ $q_{0,}$ Q_m) avec ♥la fonction partielle de transition d'états car pour tt q∈Q, ♥ Q×∑→Q n'est pas définie que pour certains événements

- **Langage généré et langage marqué**

  Soit un générateur G. L'ensemble des chaines s∈∑* telles que ♥(q_0, s)! définit le langage généré par G, noté L(G). L'ensemble des chaines s∈*L(G) telles que* ♥(q_0, s) ∈Q_m définit le langage marqué par G, noté L_m(G).

- **Accessibilité et co-accessibilité**

  *U*n état q est dit accessible s'il existe une chaine s∈∑* telle que ♥(q_0, s)=q et le générateur lui-même est dit accessible.

  Un état q est dit co-accessible s'il existe une chaine s∈∑* telle que ♥(q, s)∈Q_m . Le Générateur G est dit lui-même co-acessible si ∀q ∈Q, q est co-accessible.

- **Non blocage.**

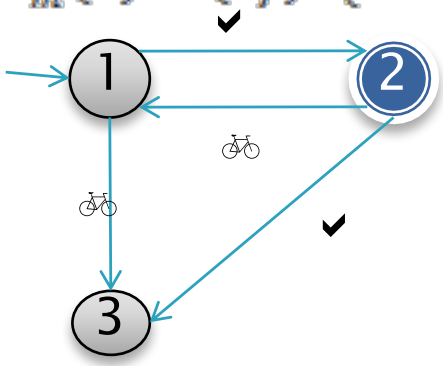
  G est dit non bloquant si chaque état accessible est Co-accessible, ☂ L(G) =$\overline{L}_m$(G), c'est-à-dire que tte chaine générée par G est un préfixe d'une chaine marquée de G.

Modélisation de SED à l 'aide d 'automates

$L(G) = (\alpha\beta)^* (\varepsilon + \alpha + \beta + \alpha\alpha)$

$L_m(G) = (\alpha\beta)^* \alpha$

$L_m(G) = (\alpha\beta)^* (\varepsilon + \alpha)$

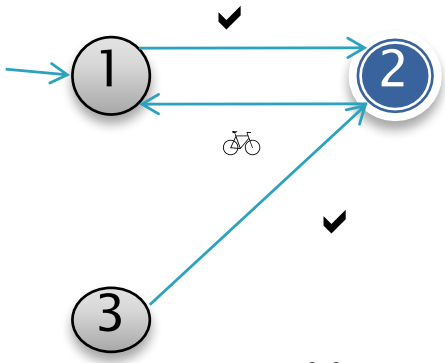


- Accessible
- Non co-accessible
- bloquant

$L(G) = (\alpha\beta)^* (\varepsilon + \alpha)$

$L_m(G) = (\alpha\beta)^* \alpha$

$L_m(G) = (\alpha\beta)^* (\varepsilon + \alpha)$



- Non Accessible
- co-accessible
- Non bloquant

Modélisation de SED à l ’aide d ’automates

- 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

$\Rightarrow$ D ’autres classes de machines d ’états, plus sophistiquées sont adaptées à la modélisation de la commande (Réseaux de Petri, Grafcet, …..)

Exercices:

Exercice 1

Soit A = {a,b,c}. Donner des expressions rationnelles pour chacun

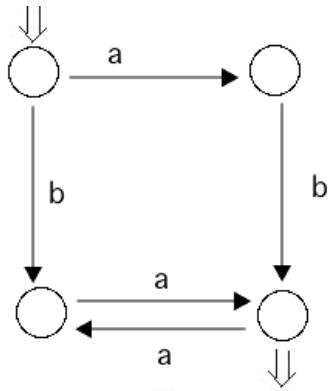
des langages suivants :

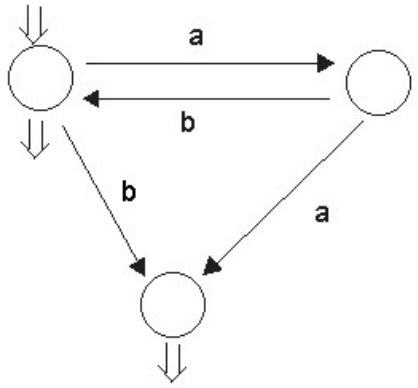
1. L'ensemble des mots de A* qui comptent au moins une fois la lettre a
2. L'ensemble des mots de A* qui ne commencent pas par la lettre a.
3. L'ensemble des mots de A* contenant au moins un a et un b, tels que la première occurrence de a précède la première occurrence de b.

- **Exercice 2**

Donner une expression rationnelle représentant les langages reconnus

- par chacun des automates suivants :





Exercices

- **Exercice 3**

  Donner une expression régulière ainsi qu'un automate déterministe qui représentent les langages suivants :

  a) L = { m | m ∈ {a, b}* et m contient 'ba' comme sous-mot }

  b) L = { m | m ∈ {a, b, c}* et m commence par un 'a' et se termine par 'bc' }

- Donner, pour chaque langage sur A = {a;b;c} donné par les expressions rationnelles suivantes, un automate fini déterministe qui le reconnaisse :

  1. (a+b)*ca*
  2. A*(b+c)
  3. A*aA*bA*

Exercices

- **Exercice 4**

Donner des automates finis déterministes sur {a;b} reconnaissant les langages. . .

1. l'ensemble des mots de longueur paire,
2. l'ensemble des mots se terminant par a,
3. l'ensemble des mots contenant au plus une occurrence de la lettre a,
4. l'ensemble des mots contenant au moins deux occurrences de leur dernière lettre,
5. l'ensemble des mots ayant pour préfixe aba,
6. l'ensemble des mots ayant pour suffixe aba,

.