

Les grammaires

Introduction:

- ▶ Une grammaire est l'ensemble des règles à suivre pour parler et écrire correctement une langue.
- ▶ Ce même principe s'applique également à la théorie des langages, où en suivant les **règles de production** d'une grammaire, il est possible de créer un langage spécifique.

Grammaire

Définitions:

- ▶ Une grammaire est un ensemble de règles de production qui sont utilisées pour engendrer un langage
- ▶ Ensemble de règles pour générer les mots du langage sont sous la forme de règles de réécriture
 - Remplacer une séquence de symboles par une autre séquence
- ▶ Mots générés = mots obtenus à partir d'un symbole spécial appelé symbole de départ ou axiome

Grammaire

Exemple: Considérons la phrase suivante :

La vieille dame regarde la petite fille

- ▶ Peut-on construire une grammaire qui permet de générer cette phrase?
- ▶ Alphabet : $A = \{ \text{la, vieille, petite, dame, fille, regarde} \}$
- ▶ Structure de la phrase :
 - ▶ Un groupe sujet (article, adjectif, nom)
 - ▶ Un verbe
 - ▶ Un groupe complément d'objet (article, adjectif, nom)

Grammaire

Exemple:

► Règles de production:

1. $\langle \text{Phrase} \rangle \rightarrow \langle \text{Sujet} \rangle \langle \text{Verbe} \rangle \langle \text{Complément} \rangle$
2. $\langle \text{Sujet} \rangle \rightarrow \langle \text{Groupe Nominal} \rangle$
3. $\langle \text{Complément} \rangle \rightarrow \langle \text{Groupe Nominal} \rangle$
4. $\langle \text{Groupe Nominal} \rangle \rightarrow \langle \text{Article} \rangle \langle \text{Nom} \rangle$
5. $\langle \text{Groupe Nominal} \rangle \rightarrow \langle \text{Article} \rangle \langle \text{Adjectif} \rangle \langle \text{Nom} \rangle$
6. $\langle \text{Article} \rangle \rightarrow \text{la}$
7. $\langle \text{Nom} \rangle \rightarrow \text{dame} \mid \text{fille}$
8. $\langle \text{Adjectif} \rangle \rightarrow \text{vieille} \mid \text{petite}$
9. $\langle \text{Verbe} \rangle \rightarrow \text{regarde}$

Grammaire

► Définition formelle:

Une grammaire **G** est un quadruplet **(N,T,P,S)** tels que:

- **N**: ensemble fini de symboles non terminaux
- **T**: ensemble fini de symboles terminaux
- $N \cap T = \emptyset$
- **S**: symbole non terminal appelé axiome (point de départ de la dérivation) ;
- **P**: ensemble fini de règles de production de la forme:
 $\alpha \rightarrow \beta$ tel que $\alpha \in (N \cup T)^+$ et $\beta \in (N \cup T)^*$

La notation $\alpha \rightarrow \beta$ est appelée une **dérivation** et signifie que α peut être remplacé par β .

Grammaire

► **Dérivation directe:**

Un mot w' dérive directement d'un mot w qu'on note $w \Rightarrow w'$ si une règle G est appliquée une fois pour passer de w à w' .

Càd: il existe une règle $\alpha \rightarrow \beta$ dans P telle que:

$w = u\alpha v$ et $w' = u\beta v$ avec $u, v \in (N \cup T)^*$

► **Dérivation au sens général:**

Un mot w' dérive d'un mot w qu'on note $w \rightarrow^* w'$, si on applique n fois les règles de G pour passer de w vers w' tel que $n \geq 0$.

Grammaire

► Exemple:

soit la grammaire: $G=(\{S\}, \{a, b\}, \{S \rightarrow aSb, S \rightarrow \epsilon\}, S)$, nous avons:

- aSb dérive directement de S et on écrit: $S \Rightarrow aSb$ car il existe une règle $S \rightarrow aSb \in P$
- ab dérive directement de aSb et on écrit: $aSb \Rightarrow ab$ car il existe une règle $S \rightarrow \epsilon \in P$
- $S \rightarrow aSb \rightarrow ab$ est une dérivation de longueur 2
- $S \rightarrow aSb \rightarrow aaSbb \rightarrow aabb$ est une dérivation de longueur 3
- $S \rightarrow aSb \rightarrow \dots \rightarrow a^n b^n$ est une dérivation de longueur $n+1$

Langage engendré par une grammaire

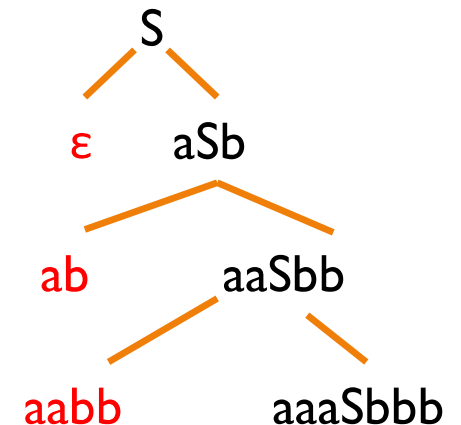
- **Définition:** Un langage engendré par une grammaire $G=(N,T,P,S)$ est l'ensemble des mots obtenus en appliquant des séquences de dérivations à partir de l'axiome S .

On note: $L(G)=\{w \in T^*/S \Rightarrow^*_G w\}$

- **Exemples:**

Pour la grammaire: $G=(\{S\}, \{a, b\}, \{S \rightarrow aSb, S \rightarrow \epsilon\}, S)$:

- Mot minimal: ϵ
- La forme générale: $L(G)=\{a^n b^n / n \geq 0\}$



- Remarque: Une grammaire définit un seul langage par contre un même langage peut être engendré par plusieurs grammaires différentes.

Grammaires équivalentes

- ▶ Deux grammaires sont équivalentes si elles engendrent le même langage.

G équivalente à $G' \Leftrightarrow L(G)=L(G')$

- ▶ **Exemples:**

- ▶ $G=(\{S,A,B\}, \{a,b\}, \{S \rightarrow aS \mid ABb, A \rightarrow Aa \mid a, B \rightarrow b\}, S)$

- ▶ $G'=(\{S,A,B\}, \{a,b\}, \{S \rightarrow aS \mid aA, A \rightarrow aA \mid bB, B \rightarrow b\}, S)$

- ▶ \Rightarrow On trouve que $L(G)=L(G')=\{a^k b^2 \mid k \geq 1\}$

Classification des grammaires

► **Hiérarchie de Chomsky**

Selon la classification de Chomsky, les grammaires sont regroupées en quatre types en fonction de la forme de leurs règles de production:

- Grammaire Syntagmatique -Type 0
- Grammaire Monotone – Type I
- Grammaire Algébrique (hors contexte) – Type 2
- Grammaire Régulière – Type 3

Classification des grammaires

► **Grammaire Syntagmatique -Type 0**

Dite aussi grammaire sans restriction (grammaire générale): si toutes ses règles sont de la forme générale suivante:

$\alpha \rightarrow \beta$ tel que $\alpha \in (N \cup T)^+$ et $\beta \in (N \cup T)^*$

► **Exemple:**

$G = (\{S\}, \{a, b, c\}, \{S \rightarrow aS \mid Sb \mid c, aSb \rightarrow Sa \mid bS\}, S)$

Classification des grammaires

- ▶ **Grammaire monotone-Type I:**

si toutes ses règles sont de la forme:

$\alpha \rightarrow \beta$ avec $|\alpha| \leq |\beta|$ tel que $\alpha \in (N \cup T)^+$ et $\beta \in (N \cup T)^*$

Exception: axiome $\rightarrow \varepsilon$ peut appartenir à P

- ▶ **Exemple:**

$G = (\{S, R, T\}, \{a, b, c\}, \{S \rightarrow \varepsilon \mid abc, R \rightarrow aRTb \mid aTb, Tb \rightarrow bT, Tc \rightarrow cc\}, S)$

Classification des grammaires

- ▶ **Grammaire algébrique –type2:**

si toutes ses règles sont de la forme:

$A \rightarrow \beta$ avec $A \in N$ et $\beta \in (N \cup T)^*$

- ▶ **Exemple:**

- ▶ $G = (\{S\}, \{a, b\}, \{S \rightarrow aSb \mid \varepsilon\}, S)$

Mots: $\varepsilon, ab, aabb, aaabbb, \dots, a^n b^n$

- ▶ $G = (\{S\}, \{0, 1\}, \{S \rightarrow 0S0 \mid 1S1 \mid \varepsilon \mid 0 \mid 1\}, S)$

Mots: 00, 11, 010, 101, 000, 111, 101101, ...

C'est la grammaire des Palindromes

Classification des grammaires

▶ **Grammaire Algebrique (hors contexte)- Type 3**

Dite aussi grammaire régulière si elle est régulière à gauche ou bien à droite

- ▶ Une grammaire G dite régulière à gauche si toutes ses règles sont de la forme:

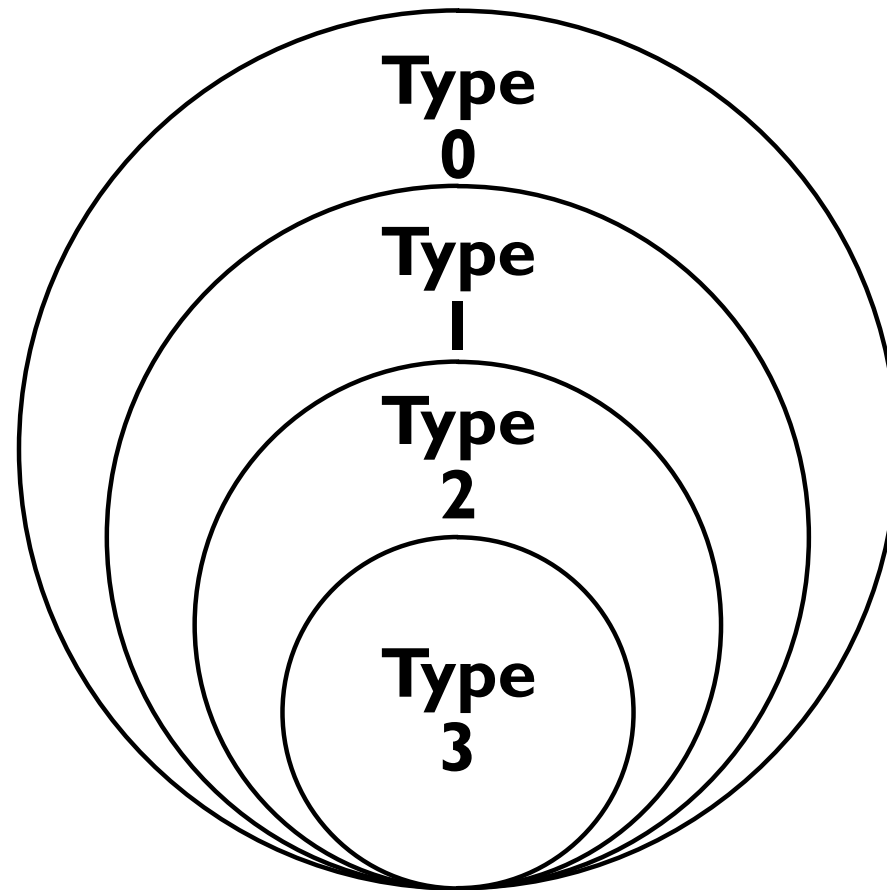
$A \rightarrow Bw$ ou $A \rightarrow w$ avec $A, B \in N$ et $w \in T^*$

- ▶ Une grammaire G dite régulière à droite si toutes ses règles sont de la forme:

$A \rightarrow wB$ ou $A \rightarrow w$ avec $A, B \in N$ et $w \in T^*$

Classification des grammaires

- Il existe une relation d'inclusion entre les types de grammaires selon la figure suivante :



Classification des grammaires

- ▶ Le type retenu pour une grammaire est le plus petit qui satisfait les conditions.
- ▶ Pour trouver la classe d'un langage on procède cependant comme suit :
 - ▶ Chercher une grammaire de type 3 qui le génère, si elle existe, le langage est de type 3 (ou **régulier**)
 - ▶ Sinon, chercher une grammaire de type 2 qui le génère, si elle existe, le langage est de type 2 (ou **algébrique**)
 - ▶ Sinon, chercher une grammaire de type 1 qui le génère, si elle existe, le langage est de type 1 (ou **contextuel**)
 - ▶ Sinon, le langage est de type 0.

Les automates

- ▶ Les grammaires représentent un moyen qui permet de décrire un langage d'une manière inductive.
- ▶ Elles montrent comment les mots du langage sont construits.
- ▶ Considérons à présent un langage L , on se propose de répondre à la question **si $w \in L$?**. On peut y répondre de plusieurs façons:
 - ▶ On peut vérifier l'existence de w dans la liste des mots de L (impossible à réaliser si le langage est infini).
 - ▶ Si L est défini par compréhension, on peut alors vérifier si w respecte la propriété du langage.
 - ▶ Si L est défini par une grammaire, on vérifie l'existence d'une chaîne de dérivation pour w

Les automates

- ▶ Il existe en réalité un autre moyen permettant de répondre à cette question : **les automates**.
- ▶ Un automate est une machine qui, après avoir exécuté un certain nombre d'opérations sur le mot, peut répondre à cette question par **oui** ou **non**.
- ▶ **Définition** : Un automate est une machine abstraite qui permet de lire un mot et de répondre à la question : "un mot w appartient-il à un langage L ?" par oui ou non.

Les automates

- ▶ **Un automate est composé de :**
 - ▶ Une bande en entrée finie ou infinie sur laquelle sera inscrit le mot à lire ;
 - ▶ Un organe de commande qui permet de gérer un ensemble fini de pas d'exécution ;
 - ▶ Eventuellement, une mémoire auxiliaire de stockage.
- ▶ **un automate contient au minimum :**
 - ▶ Un alphabet pour les mots en entrée noté X ;
 - ▶ Un ensemble non vide d'états noté Q ;
 - ▶ Un état initial noté $q_0 \in Q$;
 - ▶ Un ensemble non vide d'états finaux $q_f \in Q$;
 - ▶ Une fonction de transition (permet de changer d'état) notée δ .

Les automates

- ▶ Configuration d'un automate
- ▶ Le fonctionnement d'un automate sur un mot se fait à travers un ensemble de configurations.
- ▶ Définition: On appelle configuration d'un automate en fonctionnement les valeurs de ses différents composants, à savoir la position de la tête L/E, l'état de l'automate et éventuellement le contenu de la mémoire auxiliaire (lorsqu'elle existe).
- ▶ Il existe deux configurations spéciales appelées configuration initiale et configuration finale.

Les automates

- ▶ **Définition:** La configuration initiale est celle qui correspond à l'état initial q_0 et où la tête de L/E est positionnée sur le premier symbole du mot à lire.
- ▶ **Définition:** Une configuration finale est celle qui correspond à un des états finaux q_f et où le mot a été entièrement lu.
- ▶ On dit qu'un mot est reconnu par un automate si, à partir d'une configuration initiale, on arrive à une configuration finale à travers une succession de configurations intermédiaires .
- ▶ On dit qu'un langage est reconnu par un automate lorsque tous les mots de ce langage sont reconnus par l'automate.

Classification des automates

- ▶ Comme les grammaires, les automates peuvent être classés en 4 classes selon la hiérarchie de Chomsky:
 - ▶ Type 3 ou automate à états fini (AEF) : il reconnaît les langages de type 3. Sa structure est la suivante :
 - ▶ bande en entrée finie ;
 - ▶ sens de lecture de gauche à droite ;
 - ▶ Pas d'écriture sur la bande et pas de mémoire auxiliaire.
 - ▶ Type 2 ou automate à pile : il reconnaît les langages de type 2. Sa structure est similaire à l'AEF mais dispose en plus d'une mémoire organisée sous forme d'une pile infinie ;

Classification des automates

- ▶ Type I ou automate à bornes linéaires (ABL) : il reconnaît les langages de type I. Sa structure est la suivante :
 - ▶ Bande en entrée finie accessible en lecture/écriture ;
 - ▶ Lecture dans les deux sens ;
 - ▶ Pas de mémoire auxiliaire.
- ▶ Type 0 ou machine de Turing : il reconnaît les langages de type 0. Sa structure est la même que l'ABL mais la bande en entrée est infinie.

Les automates

Grammaire	Langage	Automate
Type 0	Récurivement énumérable	Machine de Turing
Type 1	Contextuel	Machine de Turing à borne linéaire
Type 2	Algébrique	Automate à pile
Type 3	Régulier ou rationnel	Automate à états fini

Automates à Etats Finis

Définition: Un automate d'états finis est défini par un quintuple $A=(X, Q, I, \delta, F)$ avec:

- ▶ **X**: l'alphabet
- ▶ **Q**: L'ensemble fini des états
- ▶ **I**: L'ensemble des états initiaux ($I \subseteq Q$)
- ▶ **F**: L'ensemble des états finaux ($F \subseteq Q$)
- ▶ **δ** : La relation de transition définie par l'ensemble fini de transitions de la forme (i, a, j) où i et j sont des états et a est un symbole appartenant à X .
- ▶ On la note $\delta(i, a)=j$ qui signifie la transition de l'état i vers l'état j en lisant le symbole a .

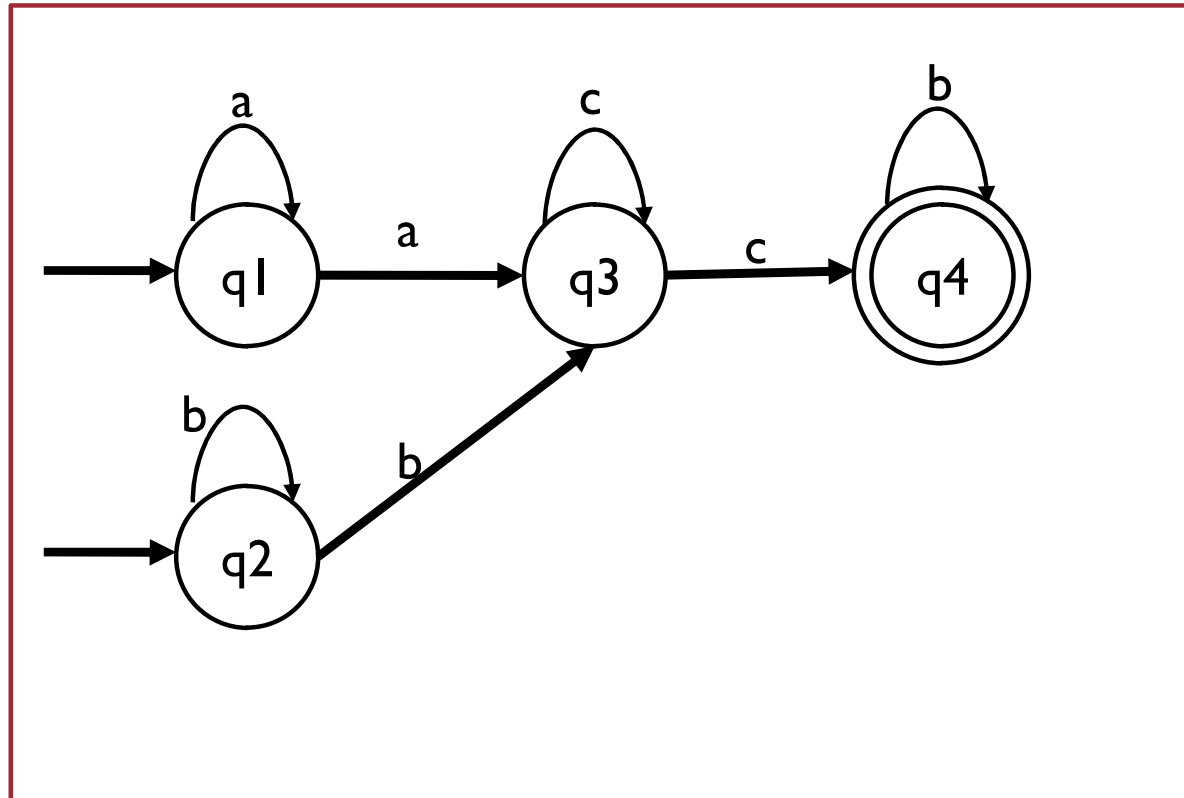
Automates à Etats Finis

Représentation Graphique: Elle consiste à représenter un automate d'états finis par un graphe orienté comme suit:

- ▶ Chaque état est schématisé par un rond (un nœud).
- ▶ Chaque état initial est précédé d'une flèche
- ▶ Chaque état final est entouré d'un cercle
- ▶ Pour chaque transition $\delta(q_i, a)=q_j$ on raccorde le nœud q_i au nœud q_j par un arc étiqueté par le symbole a .

Automates à Etats Finis

- ▶ **Représentation graphique: exemple**
- ▶ Soit l'automate d'états finis $A=(X, Q, I, \delta, F)$ avec:
 - ▶ $X=\{a,b,c\}$
 - ▶ $Q=(q1,q2,q3,q4)$
 - ▶ $I=(q1,q2)$
 - ▶ $F=\{q4\}$
 - ▶ Les transitions:
 - ▶ $\delta(q1,a)=\{q1,q3\}$
 - ▶ $\delta(q2,b)=\{q2,q3\}$
 - ▶ $\delta(q3,c)=\{q3,q4\}$
 - ▶ $\delta(q4,b)=\{q4\}$



Automates à Etats Finis

► Représentation Matricielle: Exemple

► Soit l'automate d'états finis $A=(X, Q, I, \delta, F)$ avec:

- $X=\{a,b,c\}$
- $Q=(q1,q2,q3,q4)$
- $I=(q1,q2)$
- $F=\{q4\}$
- Les transitions:
 - $\delta(q1,a)=\{q1,q3\}$
 - $\delta(q2,b)=\{q2,q3\}$
 - $\delta(q3,c)=\{q3,q4\}$
 - $\delta(q4,b)=\{q4\}$

	a	b	c
→q1	{q1,q3}	-	-
→q2	-	{q2,q3}	-
q3	-	-	{q3,q4}
q4	-	q4	-

Automates à Etats Finis

Langage accepté par un automate: Définition

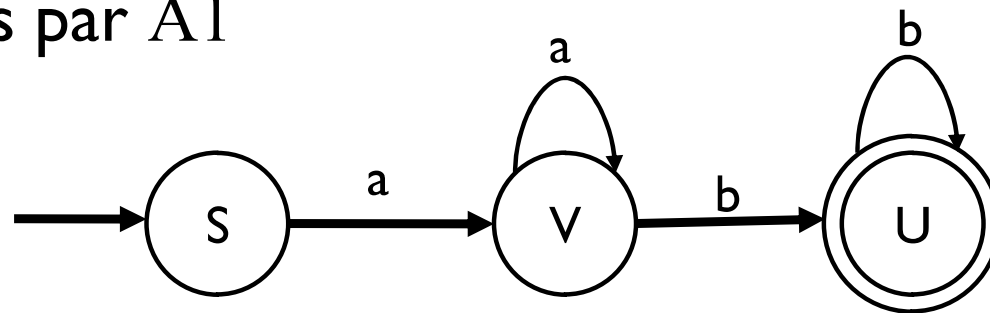
Soit l'automate d'états finis $A=(X, Q, I, \delta, F)$, on dit que:

- ▶ Tout mot fini $w=x_0.x_1.x_2.x_3....x_n$ de X est accepté par un automate A si et seulement si il existe une séquence $q_0.q_1.q_2....q_n.q_{n+1}$ de Q telle que:
 - ▶ Pour tout $0 \leq i \leq n$, $(q_i, x_i, q_{i+1}) \in \delta$.
 - ▶ q_{n+1} appartient à l'ensemble F
 - ▶ Le langage accepté (ou reconnu) par un automate A , noté $L(A)$ est constitué de l'ensemble des mots acceptés par A .
 - ▶ Le langage accepté par un automate d'état fini est un langage **régulier**.

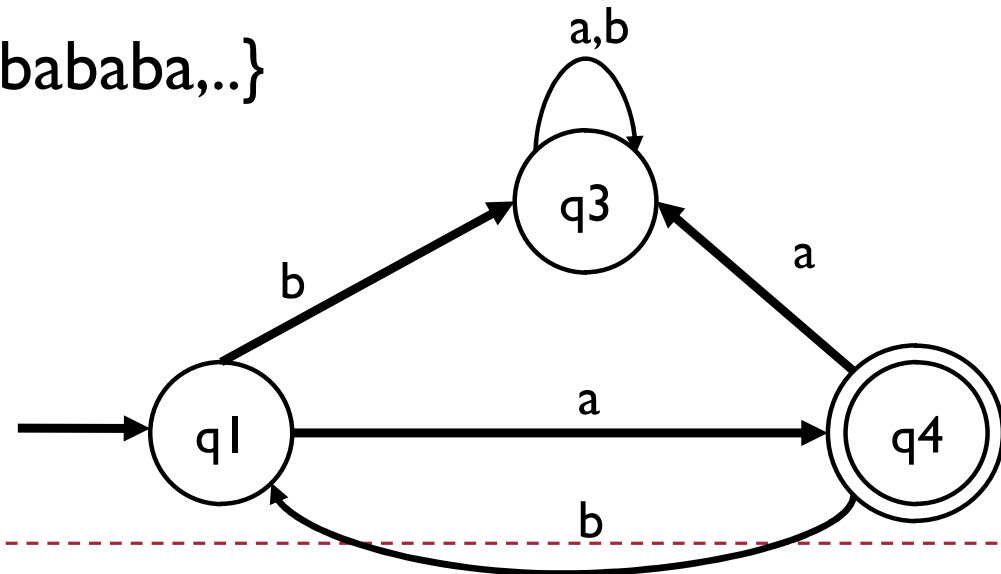
Automates à Etats Finis

► Langage accepté par un automate: Exemples:

1. Le langage accepté par cet automate est:
 $L(A1) = \{a^n b^p / n > 0, p > 0\}$. Les mots bbab et aba ne sont pas acceptés par A1



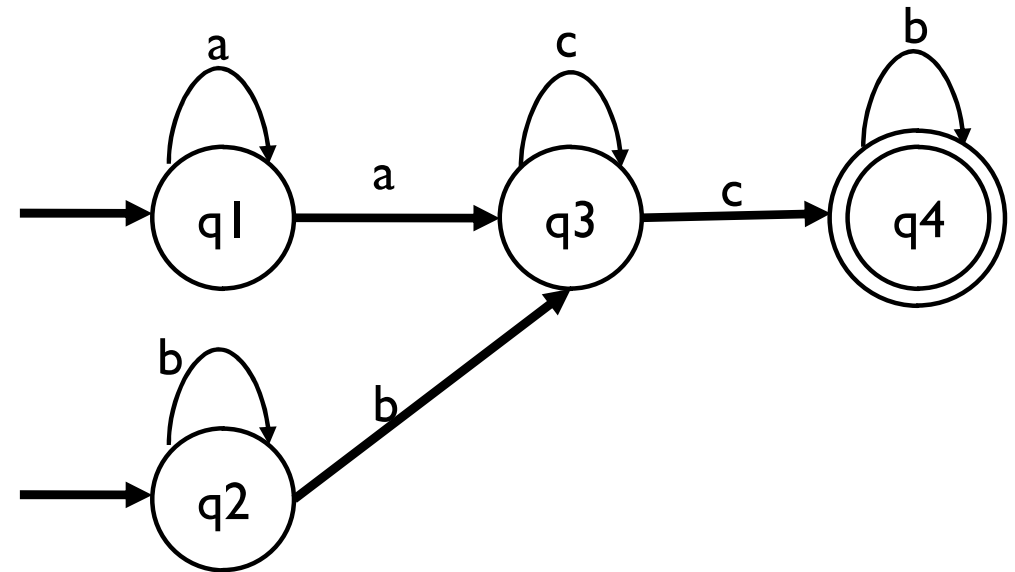
2. $L(A2) = \{a, aba, ababa, ababababa, \dots\}$



Automates Finis Non Déterministes

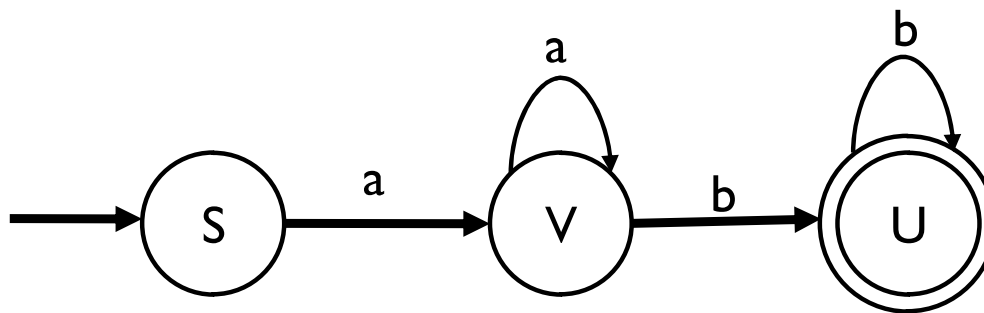
- ▶ Un automate est dit indéterministe, car:
 1. Il peut y avoir plusieurs états initiaux
 2. Étant donné un état $\delta_i \in \delta$ et un symbole $a \in X$, il peut exister plusieurs transitions possibles

Dans la représentation graphique, ce non déterminisme correspond au cas où il y a plusieurs arcs étiquetés par un même symbole terminal qui partent du même sommet.



Automates Finis Déterministes

- ▶ Un automate est dit déterministe, si:
 1. Il a un seul état initial
 2. Étant donné un état $\delta_i \in \delta$ et un symbole $a \in X$, il existe une seule transition possible



Déterminisation d'un AFN

► **Objectif:**

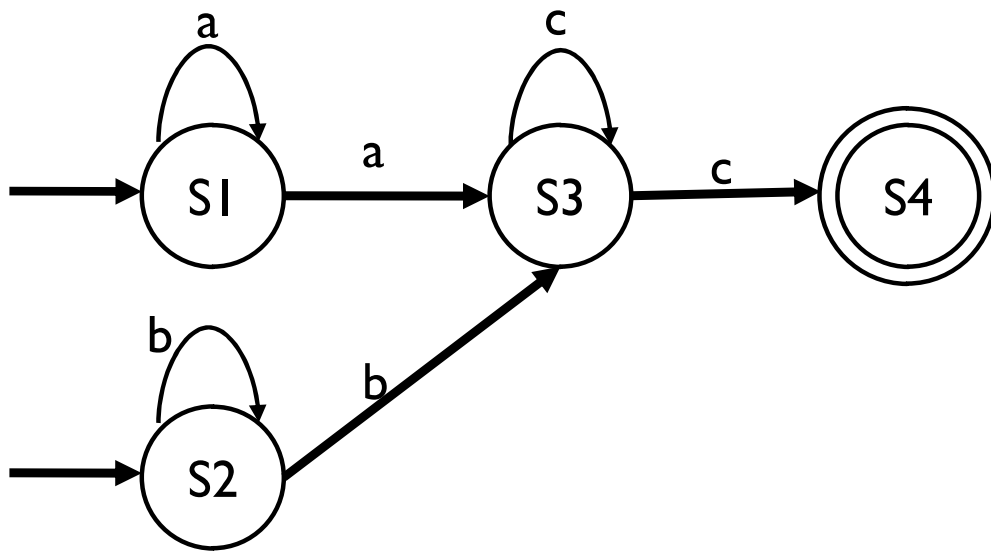
- Pour déterminer si un mot u de longueur n est accepté, un AFD effectue exactement n transitions, tandis qu'un AFN en effectue de l'ordre de 2^n . L'exécution d'un AFD est donc nettement plus efficace que celle d'un AFN.
- En contrepartie, on peut se demander si les AFN sont plus généraux, c'est-à-dire s'ils acceptent plus de langages que les AFD. La réponse, négative, est donnée par ce théorème:

► **Théorème (Rabin-Scott):**

Tout langage reconnu par un AFN peut être reconnu par un AFD.

Déterminisation d'un AFN

- ▶ En pratique: prenons l'automate suivant:



	a	b	c
→S1	{S1,S3}	-	-
→S2	-	{S2,S3}	-
S3	-	-	{S3,S4}
S4	-	-	-

- ▶ On construit ensuite les états de l'AFD et leurs fonctions de transition