Langages et langages réguliers

Alphabet

Ensemble fini non vide de symboles (lettres) souvent notés A ou Σ

Exemples

- $A := \{a, b\}$ alphabet composé de 2 lettres
- $\Sigma := \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ alphabet de 10 lettres
- $A := \{a, b, c, d, e, \ldots, x, y, z\}$ alphabet de 26 lettres
- $A := \{\circ, \vdash, =\}$ alphabet de 3 lettres

Soit A un alphabet.

Mot sur A

Suite finie de lettres de A:

- ϵ pour le *mot vide* de longueur $|\epsilon| = 0$
- $w = w_1 w_2 \ldots w_n$ mot de longueur $|w| = n$ avec $w_i \in A$

Exemples

- $w = \texttt{abbaba}$ mot sur alphabet $A = \{a, b\}$ de longueur 6
- $w = 1664$ mot sur $\Sigma = \{0, 1, 2, \ldots, 9\}$ de longueur 4

Soit A un alphabet.

Concaténation

Si $u = u_1 u_2 \dots u_m$ et $v = v_1 v_2 \dots\ v_n$ alors
$uv = u_1 u_2 \dots u_m v_1 v_2 \dots\ v_n$ de longueur $m + n$

Mot miroir

Si $u = u_1 u_2 \dots u_n$ alors $u^R = u_n u_{n-1} \dots u_1$

Exemples

- la concaténation de `16` et `64` est `1664`
- la concaténation de $u =$ `orni` et $v =$ `thorynque` est `ornithorynque`
- le miroir de $u =$ `engager` est $u^R =$ `regagne`

Langage

$\mathcal{L}$ langage sur alphabet A est un ensemble de mots sur A

Exemples

Sur $A = \{a, b, c, d, \ldots, z\}$ alphabet:

- $\mathcal{L}_1$ langage des mots de longueur 4: $\texttt{toto} \in \mathcal{L}_1$ et $\texttt{zwpt} \in \mathcal{L}_1$
- $\mathcal{L}_2$ langage des mots du français: $\texttt{ornithorynque} \in \mathcal{L}_2$ et $\texttt{mvtmjsun} \notin \mathcal{L}_2$
- $\mathcal{L}_3$ langage des mots de l'anglais: $\texttt{platypus} \in \mathcal{L}_3$ et $\texttt{mvemjsun} \notin \mathcal{L}_3$
- $\mathcal{L}_4 = \{\texttt{con}, \texttt{per}\}$
- $\mathcal{L}_5 = \{\texttt{fusion}, \texttt{version}, \texttt{sister}\}$

Opérations ensemblistes sur les langages

$\mathcal{L}$ et $\mathcal{L}'$ langages sur A:

- $\mathcal{L} \cup \mathcal{L}'$: mots dans $\mathcal{L}$ ou dans $\mathcal{L}'$
- $\mathcal{L} \cap \mathcal{L}'$: mots dans $\mathcal{L}$ et dans $\mathcal{L}'$
- $\mathcal{L} \setminus \mathcal{L}'$: mots dans $\mathcal{L}$ mais pas dans $\mathcal{L}'$
- $\mathcal{L}^c$: mots sur A pas dans $\mathcal{L}$

Exemples

- $\mathcal{L}_1 \cup \mathcal{L}_2 = \mathcal{L}_1 + \mathcal{L}_2$ mots de longueur 4 ou du français: `ornithorynque`, `test`, `rpoi` mais pas `zaoiruowx`
- $\mathcal{L}_1 \cap \mathcal{L}_2$ mots de longueur 4 du français: `test` mais pas `rpoi` ni `ornithorynque`
- $\mathcal{L}_2 \cap \mathcal{L}_3$ mots français et anglais: `sensible` mais pas `ornithorynque` ni `platypus`
- $\mathcal{L}_2 \setminus \mathcal{L}_3$ mots français mais pas anglais: `ornithorynque` mais pas `sensible` ni `platypus`
- $\mathcal{L}_1^c$ mots de longueur différente de 4: `ornithorynque`, `zpr`, ϵ

Opérations rationnelles sur les langages

$\mathcal{L}$ et $\mathcal{L}'$ langages sur A:

- $\mathcal{L}\mathcal{L}'$ concaténation des langages: mots obtenus en concaténant un mot de $\mathcal{L}$ et un mot de $\mathcal{L}'$
- $\mathcal{L}^n$ concaténation de n mots de $\mathcal{L}$: $\mathcal{L}^1 = \mathcal{L}$, $\mathcal{L}^2 = \mathcal{L}\mathcal{L}$, $\mathcal{L}^0 = \{\epsilon\}$
- $\mathcal{L}^\star$ concaténation de mots de $\mathcal{L}$:

$$\mathcal{L}^\star = \bigcup_{n \geq 0} \mathcal{L}^n$$

Exemples

- $\mathcal{L}_1\mathcal{L}_1$ mots de longueur 8
- $\mathcal{L}_1\mathcal{L}_2$ mots de longueur 4 suivis d'un mot du français: `ultrornithorynque` mais pas `traduction`
- $\mathcal{L}_2\mathcal{L}_2$ concaténation de deux mots du français: `soleil` mais pas `traduction`
- $\mathcal{L}_4\mathcal{L}_5$: `confusion`, `conversion`, `consister`, `perfusion`, `perversion`, `persister`
- $\mathcal{L}_1^\star$ mots de longueur multiple de 4
- $\mathcal{L}_2^\star$ concaténation de mots du français: `soleil`, `traduction` mais pas `ultrornithorynque`
- $A^\star$ ensemble des mots sur A

Quelques propriétés

- $\mathcal{L} + \emptyset = \emptyset + \mathcal{L} = \mathcal{L}$
- $\mathcal{L}\{\epsilon\} = \{\epsilon\}\mathcal{L} = \mathcal{L}$
- $\mathcal{L}\emptyset = \emptyset\mathcal{L} = \emptyset$

Langages rationnels

Langages sur A obtenus à partir:

- des ensembles $\{a\}$ pour $a \in A$ lettre
- ensemble vide $\emptyset$ et $\{\epsilon\}$
- opérateurs union, concaténation et étoile

Exemples

- $\{\mathtt{a}\}^\star\{\mathtt{b}\} + \{\mathtt{c}\} + \{\mathtt{d}\}$ langage rationnel
- $\{\mathtt{b}\}^\star\{\mathtt{b}\} + \{\mathtt{c}\} + \{\epsilon\}$ langage rationnel

En pratique: expressions régulières

- $\mathtt{a}^\star\mathtt{b} + \mathtt{c} + \mathtt{d}$, et $\mathtt{b}^\star\mathtt{b} + \mathtt{c} + \epsilon$
- | utilisé au lieu de $+$

Quelques propriétés

- tester si un mot est dans le langage représenté par une expression régulière est rapide
- créer une expression régulière pour langage rationnel « facile »
- $A(B+C) = AB + AC$ et $(A+B)C = AC + BC$ pour A, B, C trois langages
- lemme d'Arden:

$$X = AX + B \text{ a une unique solution } X = A^{\star}B$$

avec A, B langage et $\epsilon \notin A$

Exemple complet

L langage sur $\{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ où tout `a` est suivi plus tard d'un `b`:

`acbaccb` et `aacb` dans L mais pas `acbacc`

Modélisation

L_b: mots contenant au moins un `b` et où tout `a` est suivi plus tard d'un `b`

$$\begin{cases} L & = & \mathtt{a}L_b + (\mathtt{b}+\mathtt{c})L + \epsilon \\ L_b & = & (\mathtt{a}+\mathtt{c})L_b + \mathtt{b}L \end{cases}$$

Résolution

$$L_b \overset{\text{Arden}}{=\!=\!=} (\mathtt{a}+\mathtt{c})^\star \mathtt{b} L \qquad \ldots \qquad L = (\mathtt{a}(\mathtt{a}+\mathtt{c})^\star\mathtt{b} + \mathtt{b} + \mathtt{c})^\star$$

Automates finis déterministes

Automates finis déterministes

Définitions

Automate fini déterministe

- un alphabet A
- un ensemble fini Q d'états
- un état initial $q_0 \in Q$
- un ensemble d'états finaux $F \subseteq Q$
- une fonction de transition $\delta : Q \times A \rightarrow Q$

Remarques

- $\delta(q, a) = q'$: « de l'état q en lisant a l'automate arrive en q' »
- si δ totale alors automate complet

Exemple et représentation graphique

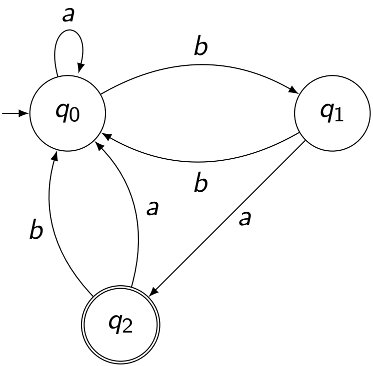
$A := \{a, b\}$

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

q_0 initial

$F := \{q_2\}$

δ	a	b
q_0	q_0	q_1
q_1	q_2	q_0
q_2	q_0	q_0



Automates finis déterministes

Définitions

Mot accepté

$w = w_1 w_2 \dots w_n$ accepté si

- q_0 initial
- $\delta(q_0, w_1) = q_1$
- $\delta(q_1, w_2) = q_2$

  $\vdots$
- $\delta(q_{n-1}, w_n) = q_n$
- q_n final

Langage accepté

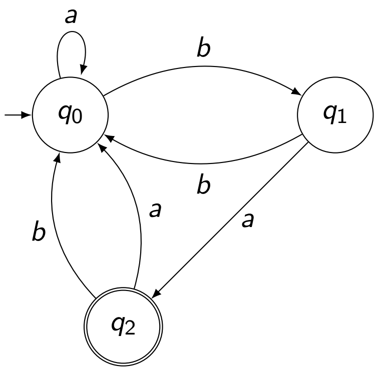
$\mathcal{L}(\mathcal{A})$: ensemble des mots acceptés par automate $\mathcal{A}$

Langages reconnaissables

Ensemble des langages acceptés par des automates

Automates finis déterministes

Exemples



- $abaa \notin \mathcal{L}(\mathcal{A})$:

$$q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_1 \xrightarrow{a} q_2 \xrightarrow{a} q_0 \notin F$$

- $bababa \in \mathcal{L}(\mathcal{A})$:

$$q_0 \xrightarrow{b} q_1 \xrightarrow{a} q_2 \xrightarrow{b} q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_1 \xrightarrow{a} q_2 \in F$$

Automates finis déterministes

Exemples

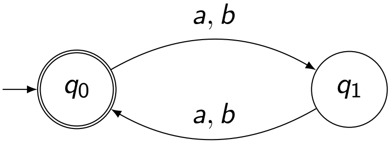
Programme testant l'appartenance d'un mot

```c
int accept(char* w){
    int q;
    for(q = 0; *w != '\0'; w++) {
        if ((q == 0) && (*w == 'a'))
            q = 0;
        else if ((q == 0) && (*w == 'b'))
            q = 1;
        else if ((q == 1) && (*w == 'a'))
            q = 2;
        else if ((q == 1) && (*w == 'b'))
            q = 0;
        else /* q = 2 */
            q = 0;
    }
    return (q == 2);
}
```

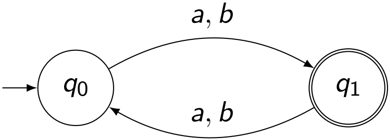
Automates finis déterministes

Exemples

Mots de longueur paire



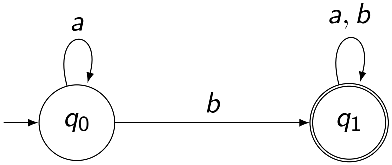
Mots de longueur impaire



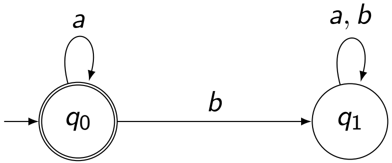
Automates finis déterministes

Exemples

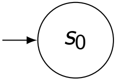
Mots ayant au moins un b



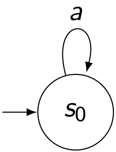
Mots n'ayant pas de b



Mots qui finissent par `b`



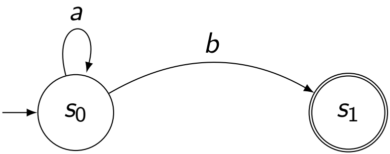
Mots qui finissent par b



Automates finis déterministes

Exemples

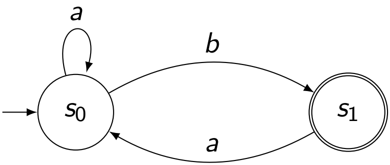
Mots qui finissent par `b`



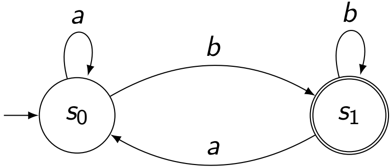
Automates finis déterministes

Exemples

Mots qui finissent par `b`



Mots qui finissent par `b`



Automates finis déterministes

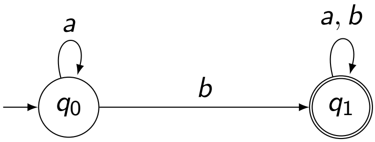
Manipulation d'automates

Complémentaire

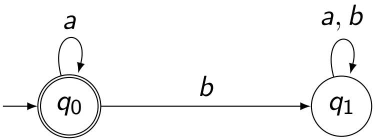
Si $\mathcal{A}$ déterministe **complet** reconnait L alors inverser états finaux/non finaux produit automate qui reconnaît $\overline{L}$

Exemple

- mots ayant au moins un b



- mots n'ayant pas de b

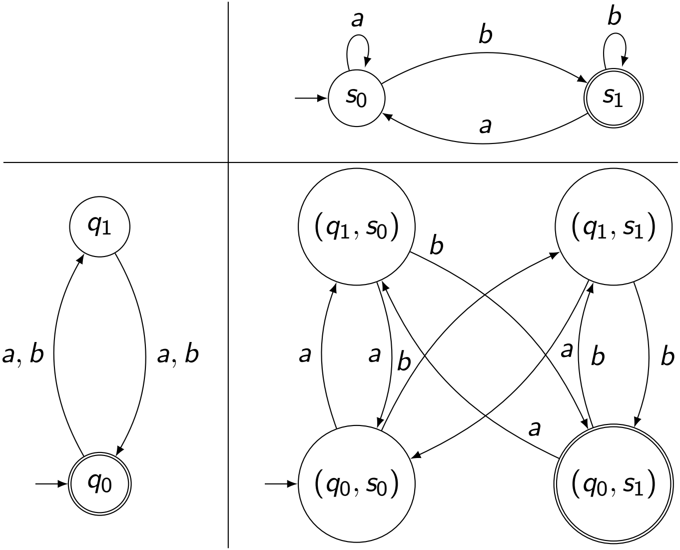


Automate produit et intersection

Si $\mathcal{A}^1$ et $\mathcal{A}^2$ déterministes reconnaissant L_1 et L_2 alors automate produit $\mathcal{A}$ reconnaît $L_1 \cap L_2$ avec:

- $Q = Q^1 \times Q^2$ les états
- (q_0^1, q_0^2) état initial
- $F = F^1 \times F^2$ états finaux
- $$\begin{array}{rcl} \delta : (Q^1 \times Q^2) \times A & \rightarrow & Q^1 \times Q^2 \\ ((q^1, q^2), a) & \mapsto & (\delta^1(q^1, a), \delta^2(q^2, a)) \end{array}$$

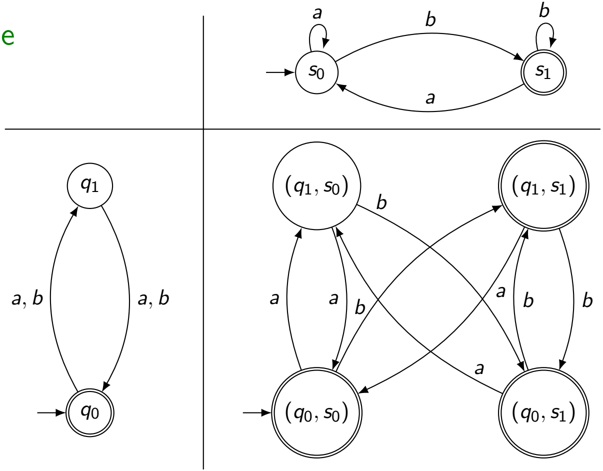
Exemple automate produit



Automate produit et union

La même construction avec $(q^1, q^2) \in F$ si $q^1 \in F^1$ **ou** $q^2 \in F^2$ permet de reconnaître $L_1 \cup L_2$.

Exemple



Automates finis déterministes

Des automates aux regexp

Théorème

Tout langage reconnu par un automate est rationnel.

Principe

$\mathcal{A}$: automate déterministe

L_p: langage des mots acceptés à partir de l'état p

Il suffit de résoudre le système d'équations:

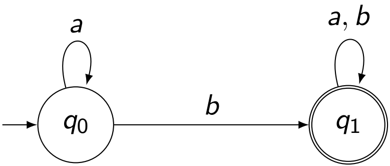
- si q final et $q \xrightarrow{a_1} q_1$, $q \xrightarrow{a_2} q_2$, etc.

$$L_q = a_1 L_{q_1} + a_2 L_{q_2} + \ldots + a_n L_q q_n + \varepsilon$$

- si q non final et $q \xrightarrow{a_1} q_1$, $q \xrightarrow{a_2} q_2$, etc.

$$L_q = a_1 L_{q_1} + a_2 L_{q_2} + \ldots + a_n L_q q_n$$

Exemple 1



$$\begin{cases} L_0 = & aL_0 + bL_1 \\ L_1 = & aL_1 + bL_1 + \varepsilon \end{cases}$$

d'où

$$L_1 = (a+b)^\star$$

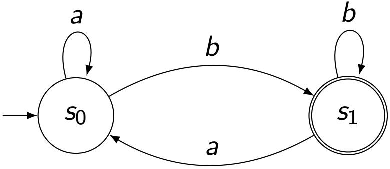
puis

$$L_0 = a^\star b(a+b)^\star$$

Automates finis déterministes

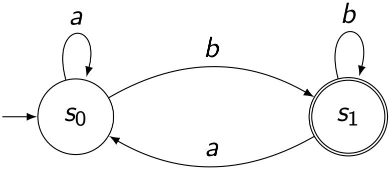
Des automates aux regexp

Exemple 2



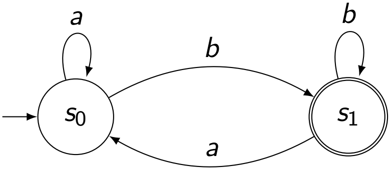
$$\begin{cases} L_0 = \\ \end{cases}$$

Exemple 2



$$\begin{cases} L_0 = & aL_0 + bL_1 \\ L_1 = & \end{cases}$$

Exemple 2

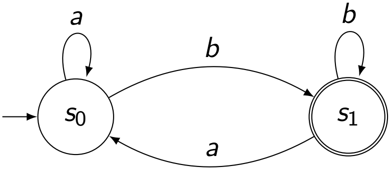


$$\begin{cases} L_0 = & aL_0 + bL_1 \\ L_1 = & aL_0 + bL_1 + \varepsilon \end{cases}$$

Automates finis déterministes

Des automates aux regexp

Exemple 2



$$\begin{cases} L_0 = & aL_0 + bL_1 \\ L_1 = & aL_0 + bL_1 + \varepsilon \end{cases}$$

$$\vdots$$

$$L_0 = (a + bb^\star a)^\star bb^\star$$

ou aussi

$$L_0 = (a + b)^\star b$$

Automates finis non déterministes

Automates finis non déterministes

Définitions

Automate fini non déterministe

- un alphabet A
- un ensemble fini Q d'états
- un **ensemble d'états initiaux** $I \subseteq Q$
- un ensemble d'états finaux $F \subseteq Q$
- une fonction de transition $\delta : Q \times A \to \mathcal{P}(\mathbf{Q})$

Remarques

$\delta(q, a)$: « ensemble des états atteignables depuis q en lisant a »

Exemple

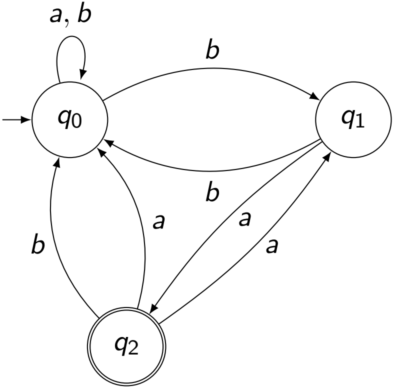
$A := \{a, b\}$

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

$I := \{q_0\}$ initiaux

$F := \{q_2\}$

δ	a	b
q_0	q_0	q_0, q_1
q_1	q_2	q_0
q_2	q_0, q_1	q_0



Automates finis non déterministes

Définitions

Mot accepté

$w = w_1 w_2 \dots w_n$ accepté si

- $q_0 \in I$ un état initial
- $\delta(q_0, w_1) \ \ni \ q_1$
- $\delta(q_1, w_2) \ \ni \ q_2$

  $\vdots$
- $\delta(q_{n-1}, w_n) \ \ni \ q_n$
- $q_n \in F$ un état final

Langage accepté

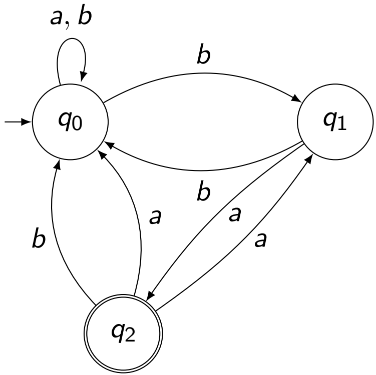
$\mathcal{L}(\mathcal{A})$: ensemble des mots acceptés par automate $\mathcal{A}$

Langages reconnaissables

Ensemble des langages acceptés par des automates non déterministes

Automates finis non déterministes

Définitions



- $abaa \notin \mathcal{L}(\mathcal{A})$:

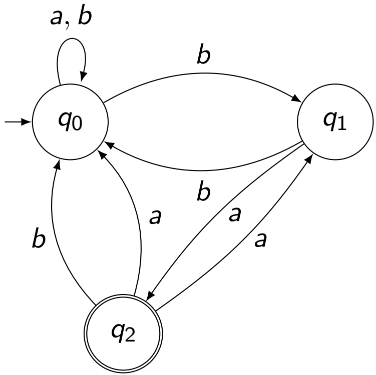
$$q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_0 \xrightarrow{a} q_0 \xrightarrow{a} q_0 \notin F$$

$$q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_1 \xrightarrow{a} q_2 \xrightarrow{a} q_0 \notin F$$

$$q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_1 \xrightarrow{a} q_2 \xrightarrow{a} q_1 \notin F$$

Automates finis non déterministes

Définitions



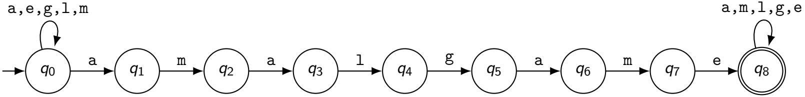
- $baaa \in \mathcal{L}(\mathcal{A})$:

$$q_0 \xrightarrow{b} q_0 \xrightarrow{a} q_0 \xrightarrow{a} q_0 \xrightarrow{a} q_0 \notin F$$

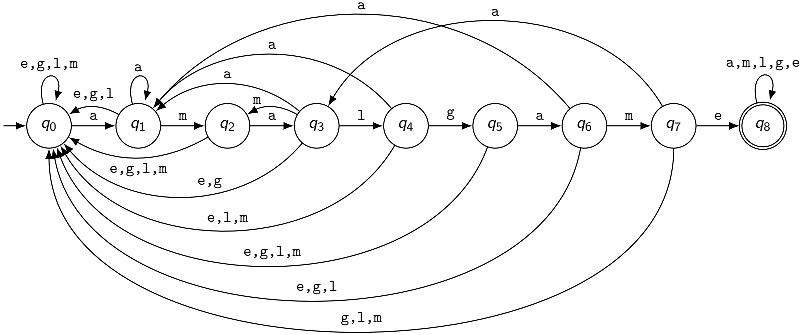
$$q_0 \xrightarrow{b} q_1 \xrightarrow{a} q_2 \xrightarrow{a} q_0 \xrightarrow{a} q_0 \notin F$$

$$q_0 \xrightarrow{b} q_1 \xrightarrow{a} q_2 \xrightarrow{a} q_1 \xrightarrow{a} q_2 \in F$$

Version non déterministe



Version déterministe

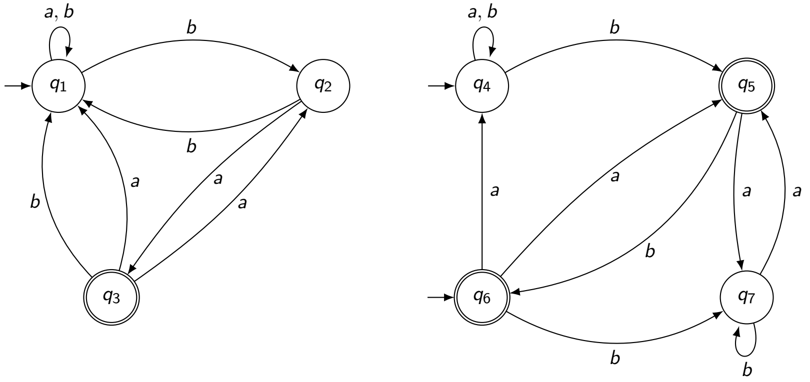


Automate pour l'union

Si $\mathcal{A}_1$ et $\mathcal{A}_2$ automates (non) déterministes reconnaissant L_1 et L_2 alors $\mathcal{A}$ reconnaît $L_1 \cup L_2$ avec:

- $Q = Q_1 \cup Q_2$ les états
- $I = I_1 \cup I_2$ états initiaux
- $F = F_1 \cup F_2$ états finaux
- $\delta = \delta_1 \cup \delta_2$ les transitions

Exemple d'automate pour l'union



Automates finis non déterministes

Constructions d'automates

Automate pour la concaténation

Si $\mathcal{A}_1$ et $\mathcal{A}_2$ automates (non) déterministes reconnaissant L_1 et L_2 alors $\mathcal{A}$ reconnaît $L_1 L_2$ avec:

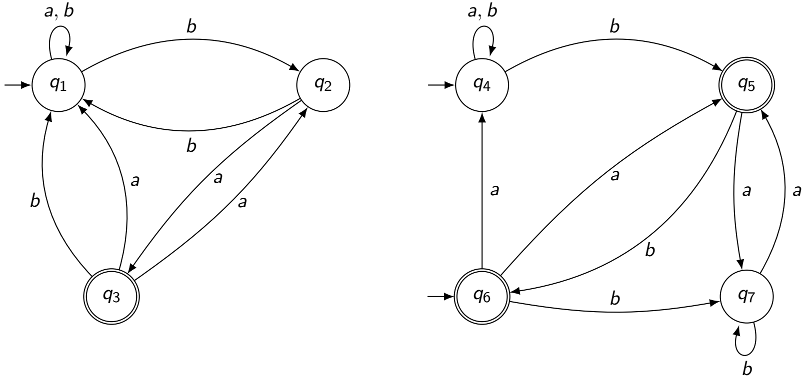
- $Q = Q_1 \cup Q_2$ les états
- $I = I_1$ états initiaux
- δ les transitions: celles de δ_1 et de δ_2 plus

$$\text{si} \left\{ \begin{array}{c} q_i \text{ initial de } \mathcal{A}_2 \\ q_i \xrightarrow{a} q \in \delta_2 \\ q_f \text{ final de } \mathcal{A}_1 \end{array} \right\} \text{ on ajoute } q_f \xrightarrow{a} q$$

les finaux de $\mathcal{A}_1$ se comportent comme les initiaux de $\mathcal{A}_2$

- états finaux
  - si $\varepsilon \notin L_2$: $F = F_2$
  - si $\varepsilon \in L_2$: $F = F_1 \cup F_2$

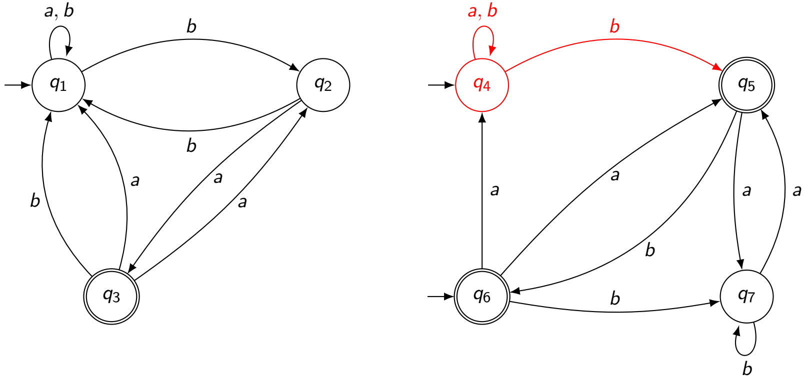
Exemple 1



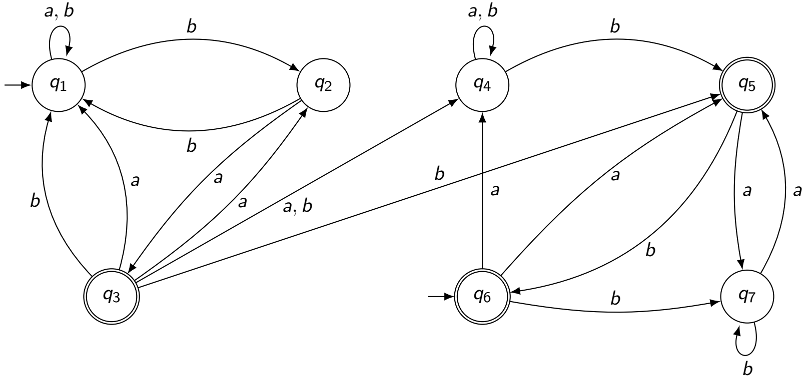
Automates finis non déterministes

Constructions d'automates

Exemple 1



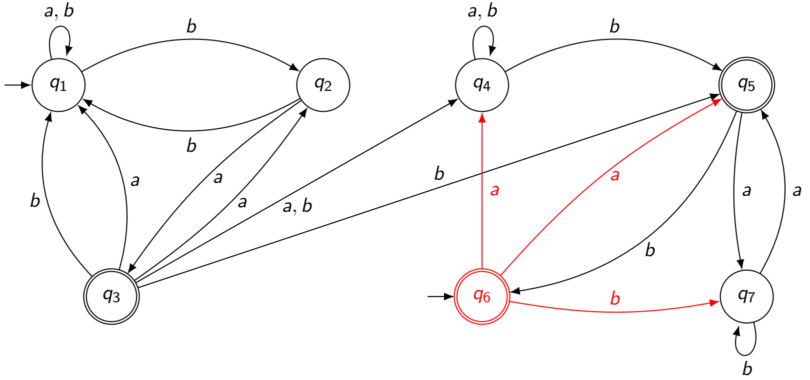
Exemple 1



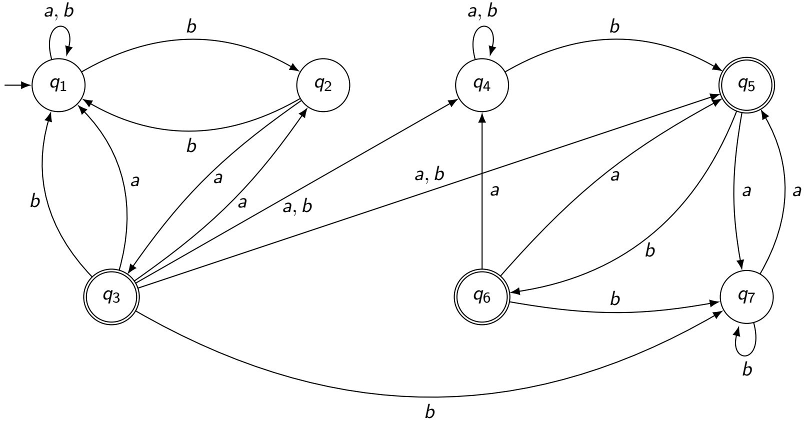
Automates finis non déterministes

Constructions d'automates

Exemple 1



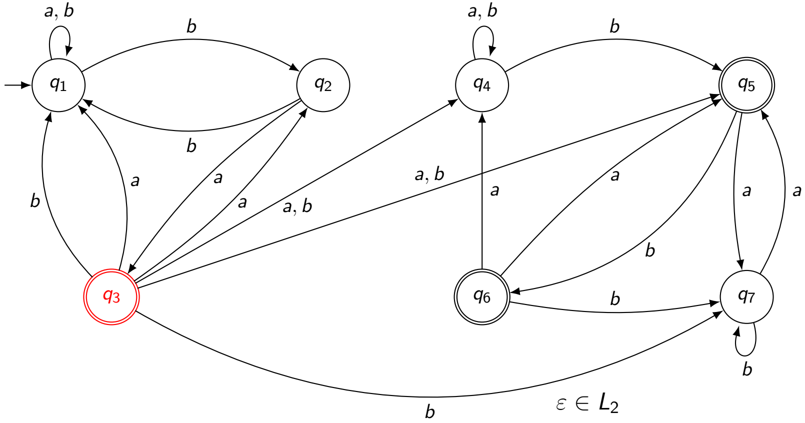
Exemple 1



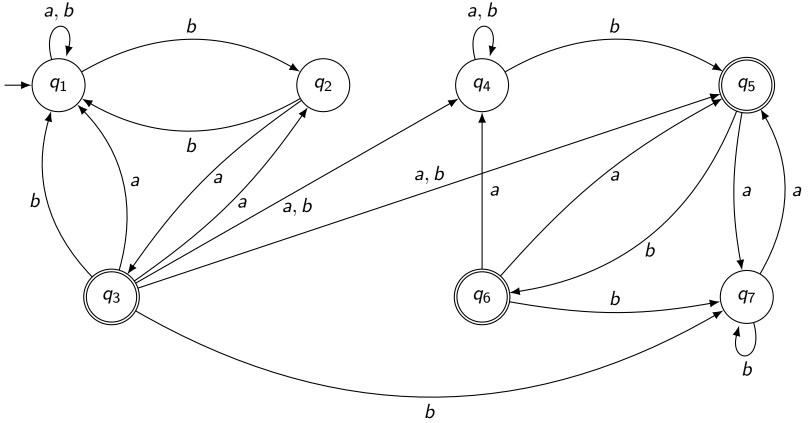
Automates finis non déterministes

Constructions d'automates

Exemple 1



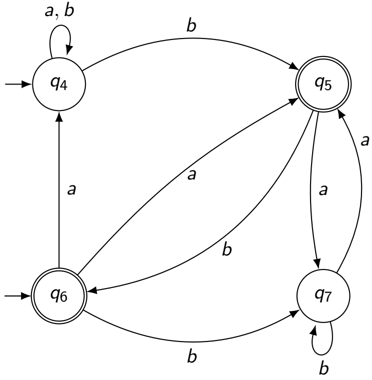
Exemple 1

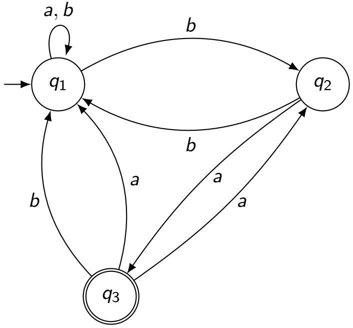


Automates finis non déterministes

Constructions d'automates

Exemple 2





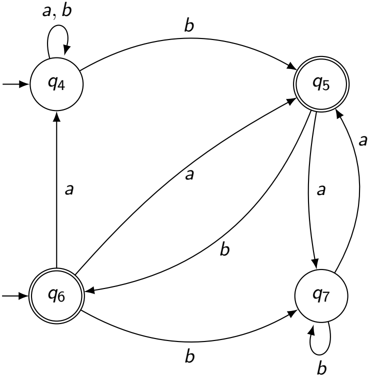
$q_4 \xrightarrow{b} q_5 \xrightarrow{a} q_7 \xrightarrow{a} q_5$

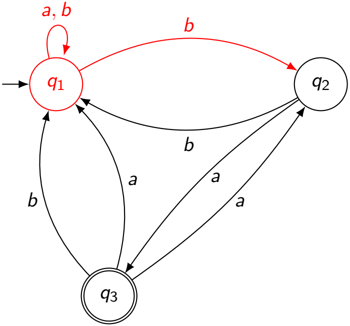
$q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3$

Automates finis non déterministes

Constructions d'automates

Exemple 2

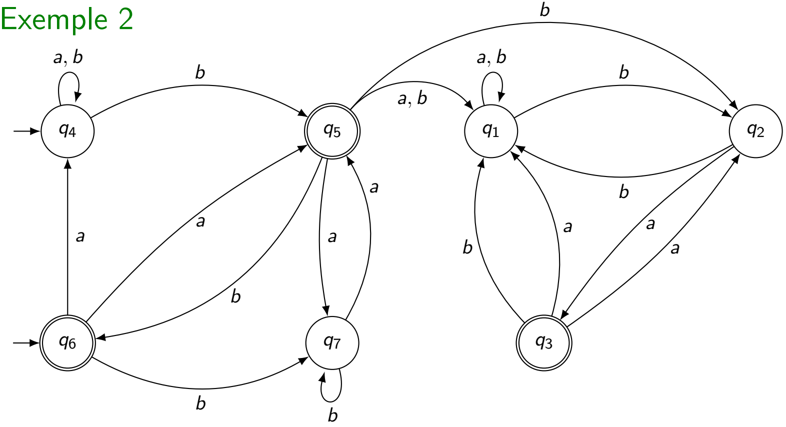




$q_4 \xrightarrow{b} q_5 \xrightarrow{a} q_7 \xrightarrow{a} q_5$

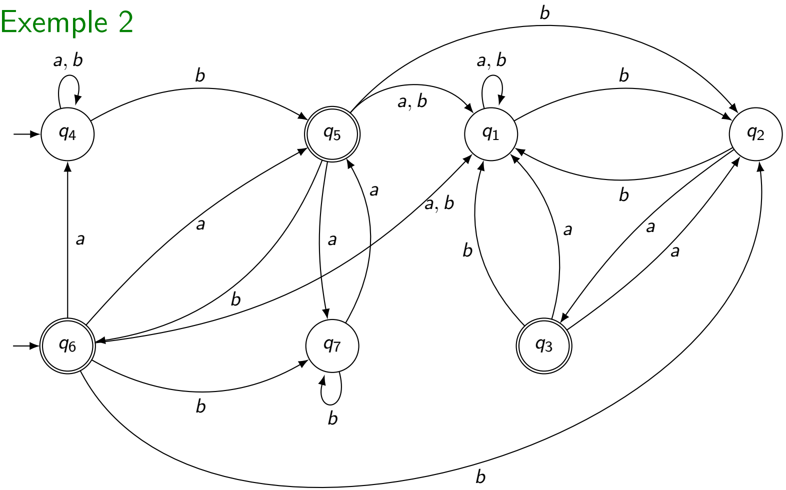
$q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3$

Exemple 2



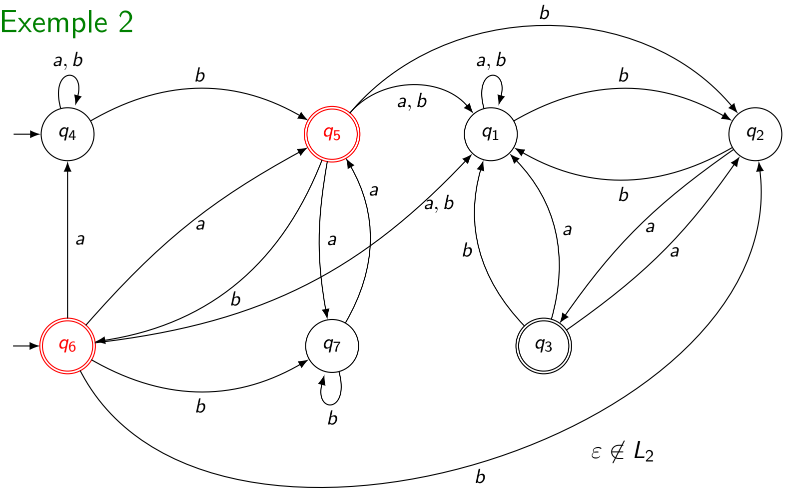
$q_4 \xrightarrow{b} q_5 \xrightarrow{a} q_7 \xrightarrow{a} q_5$ $\qquad$ $q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3$

Exemple 2



$q_4 \xrightarrow{b} q_5 \xrightarrow{a} q_7 \xrightarrow{a} q_5$ $\qquad$ $q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3$

Exemple 2

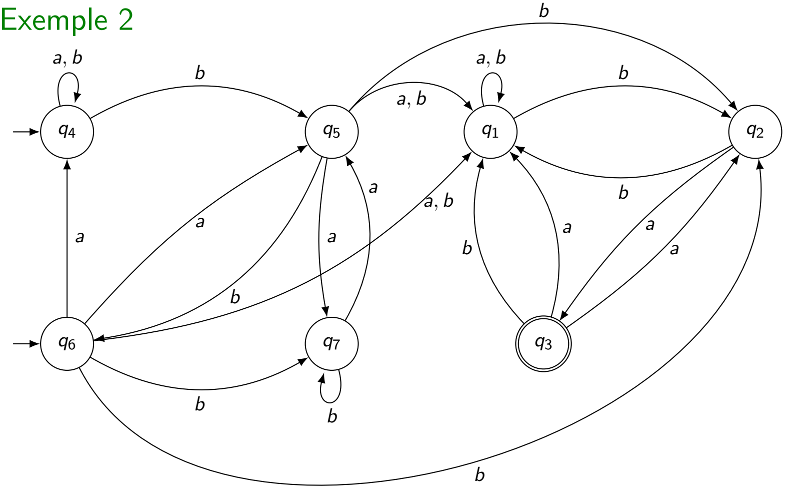


$\varepsilon \notin L_2$

$q_4 \xrightarrow{b} q_5 \xrightarrow{a} q_7 \xrightarrow{a} q_5$

$q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3$

Exemple 2

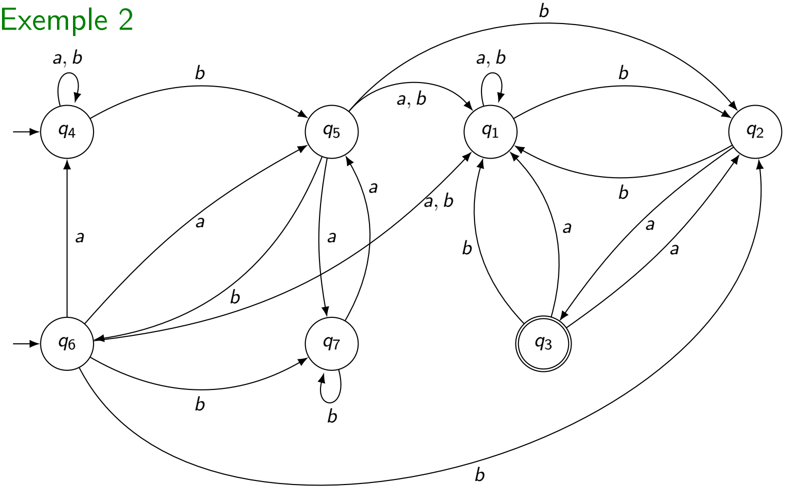


$q_4 \xrightarrow{b} q_5 \xrightarrow{a} q_7 \xrightarrow{a} q_5$ $\qquad$ $q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3$

Automates finis non déterministes

Constructions d'automates

Exemple 2



$q_4 \xrightarrow{b} q_5 \xrightarrow{a} q_7 \xrightarrow{a} q_5$ $\qquad\qquad$ $q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3$

$q_4 \xrightarrow{b} q_5 \xrightarrow{a} q_7 \xrightarrow{a} q_5 \xrightarrow{\qquad\qquad b \qquad\qquad} q_2 \xrightarrow{a} q_3$

Automate pour l'étoile

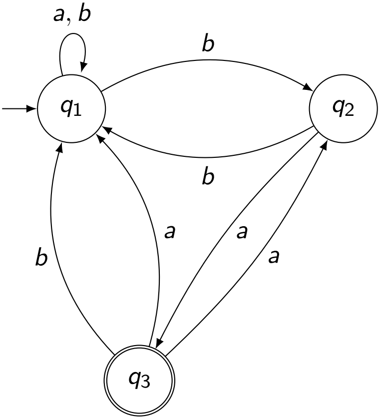
Si $\mathcal{A}_1$ automate (non) déterministe reconnaissant L_1 alors $\mathcal{A}$ reconnaît $L_1^\star$ avec:

- $Q = Q_1 \cup \{q_0\}$ les états
- $I = \{q_0\}$ état initial
- $F = F_1 \cup \{q_0\}$ états finaux
- δ les transitions: celles de δ_1 plus

$$\text{si } \left\{ \begin{array}{c} q_i \text{ initial de } \mathcal{A}_1 \\ q_i \xrightarrow{a} q \in \delta_1 \\ q_f \text{ final de } \mathcal{A}_1 \end{array} \right\} \text{ on ajoute } \left\{ \begin{array}{c} q_f \xrightarrow{a} q \\ q_0 \xrightarrow{a} q \end{array} \right\}$$

les finaux de $\mathcal{A}$ et q_0 se comportent comme les initiaux de $\mathcal{A}_1$

Exemple 1

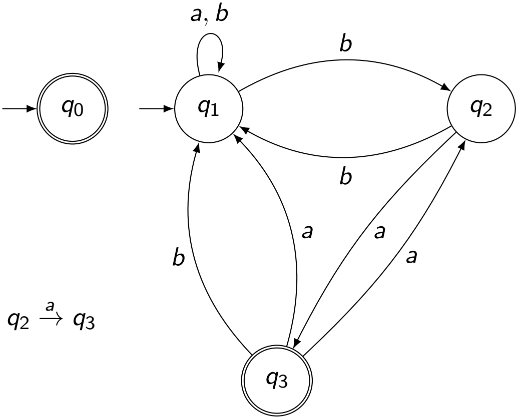


$q_1 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3$

$q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3$

$q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3$

Exemple 1

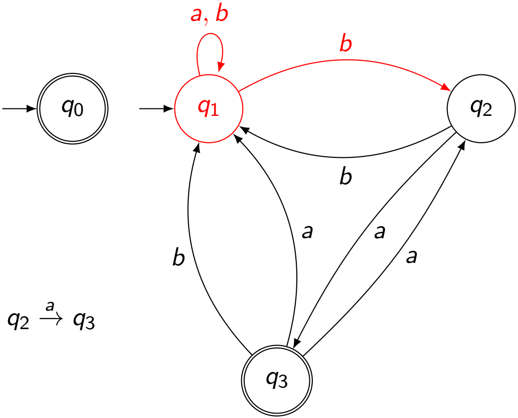


$q_1 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3$

$q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3$

$q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3$

Exemple 1

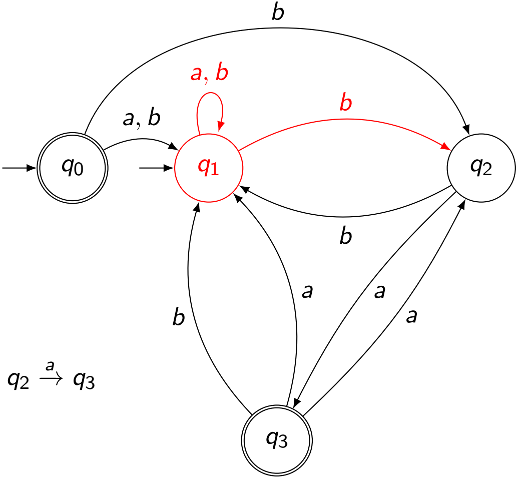


$q_1 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3$

$q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3$

$q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3$

Exemple 1

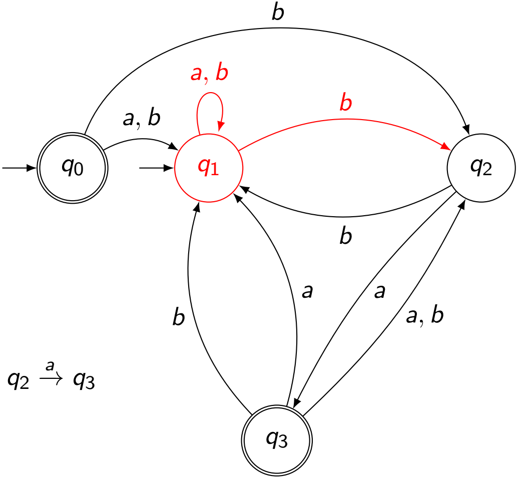


$q_1 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3$

$q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3$

$q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3$

Exemple 1

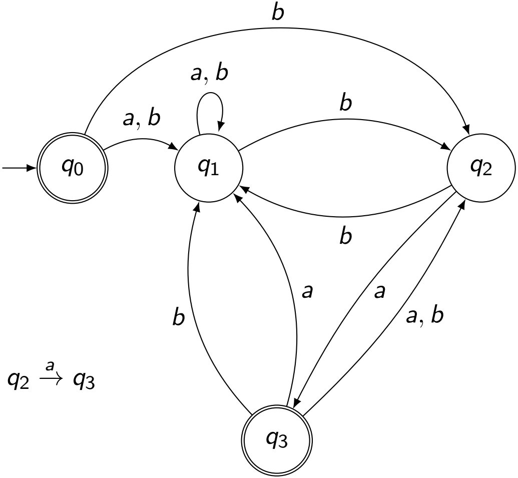


$q_1 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3$

$q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3$

$q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3$

Exemple 1

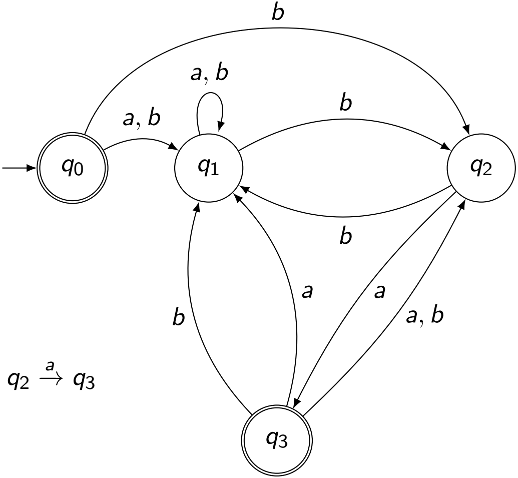


$q_1 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3$

$q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3$

$q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3$

Exemple 1

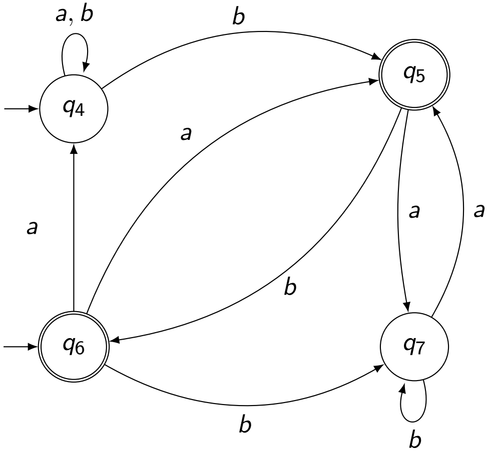


$q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3$

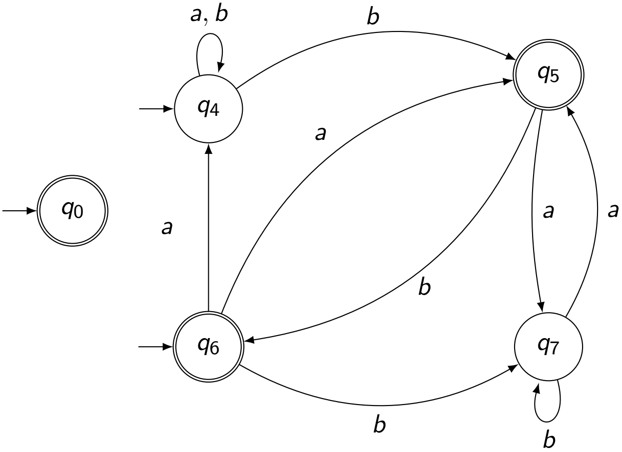
$\xrightarrow{b} q_2 \xrightarrow{a} q_3$

$\xrightarrow{b} q_2 \xrightarrow{a} q_3 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3$

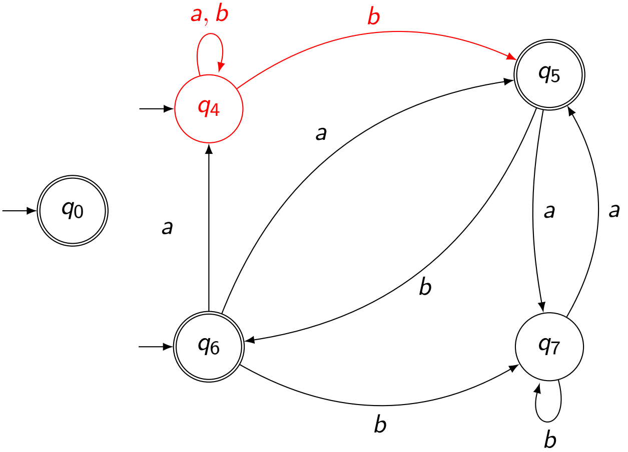
Exemple 2



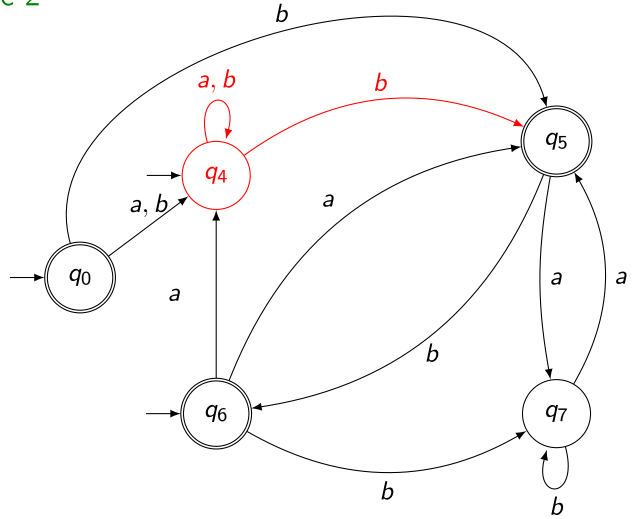
Exemple 2



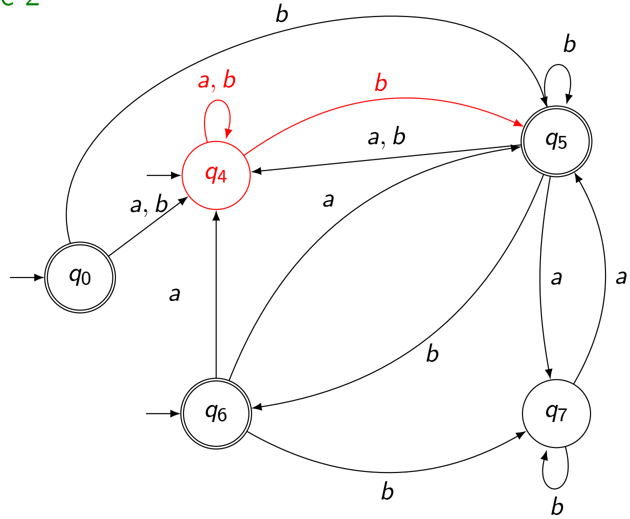
Exemple 2



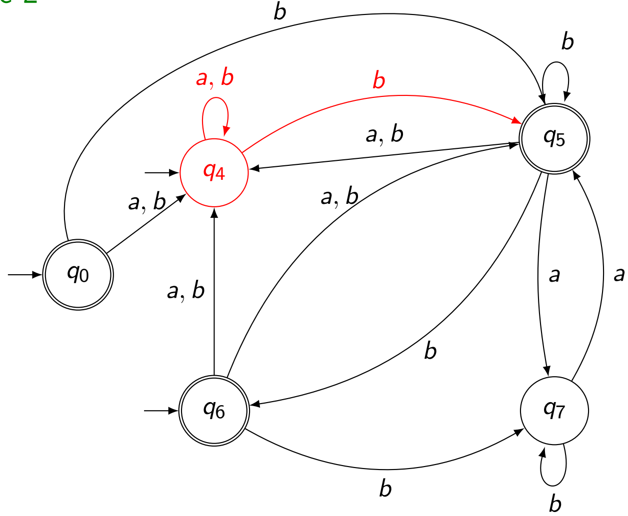
Exemple 2



Exemple 2



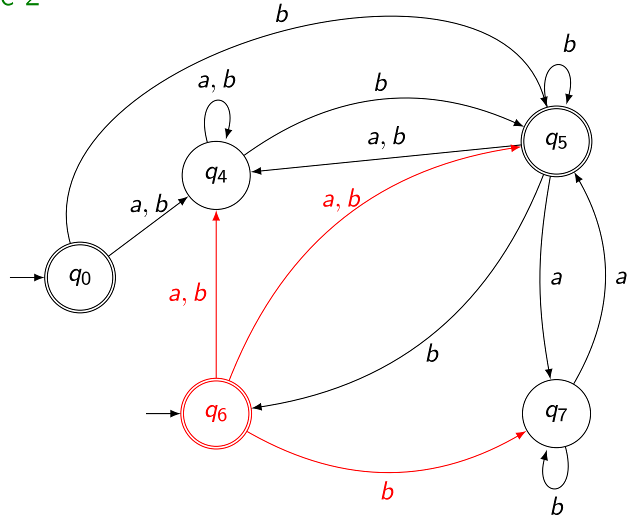
Exemple 2



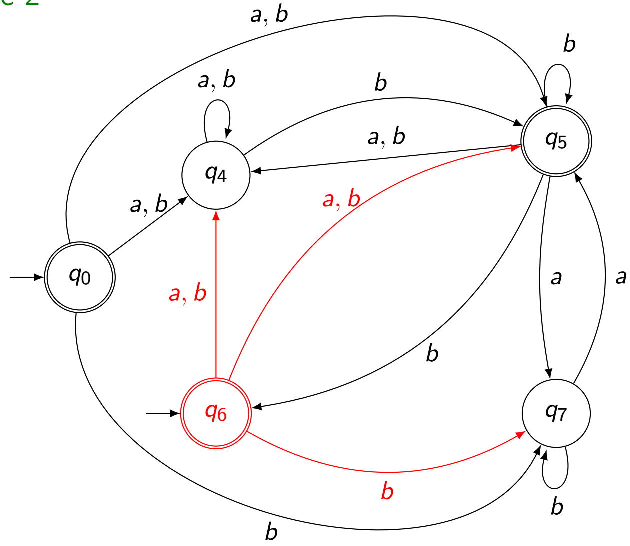
Automates finis non déterministes

Constructions d'automates

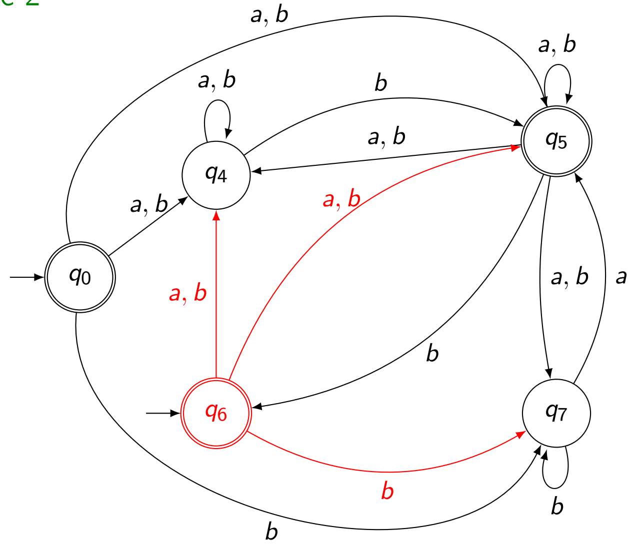
Exemple 2



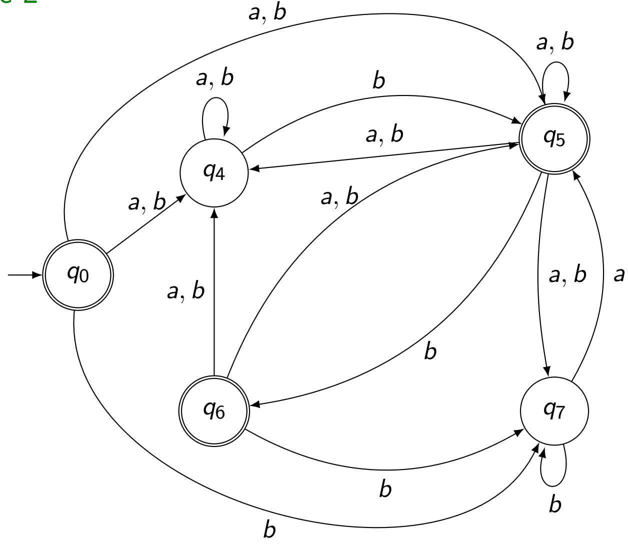
Exemple 2



Exemple 2



Exemple 2



Corollaire

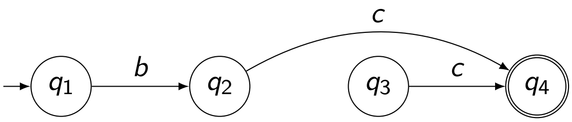
Tout langage rationnel est reconnu par un automate non déterministe.

Exemple: $c(a + bc)^\star$

Corollaire

Tout langage rationnel est reconnu par un automate non déterministe.

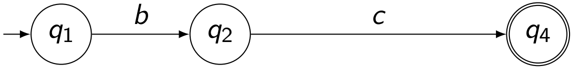
Exemple: $c(a + bc)^{\star}$



Corollaire

Tout langage rationnel est reconnu par un automate non déterministe.

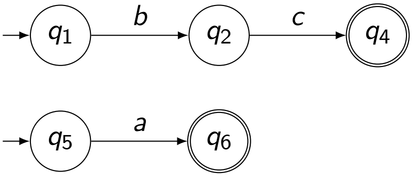
Exemple: $c(a+bc)^\star$



Corollaire

Tout langage rationnel est reconnu par un automate non déterministe.

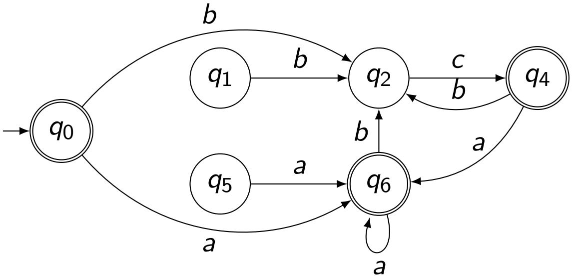
Exemple: $c(a + bc)^{\star}$



Corollaire

Tout langage rationnel est reconnu par un automate non déterministe.

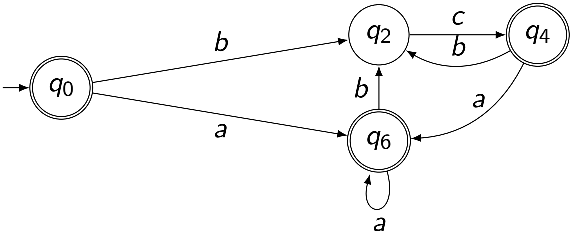
Exemple: $c(a+bc)^\star$



Corollaire

Tout langage rationnel est reconnu par un automate non déterministe.

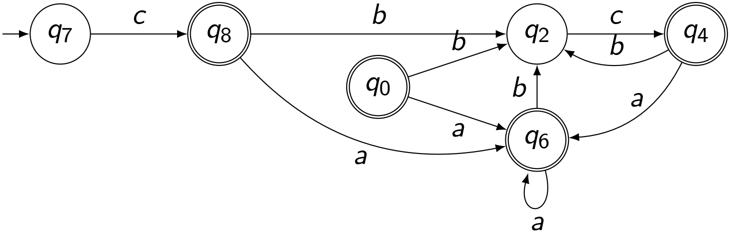
Exemple: $c(a+bc)^\star$



Corollaire

Tout langage rationnel est reconnu par un automate non déterministe.

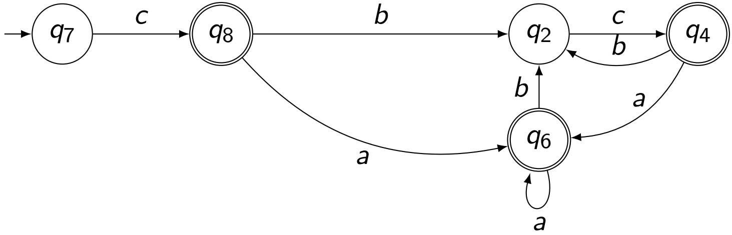
Exemple: $c(a+bc)^\star$



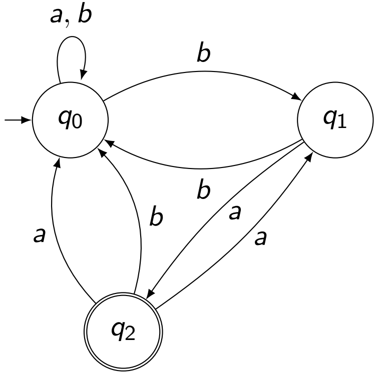
Corollaire

Tout langage rationnel est reconnu par un automate non déterministe.

Exemple: $c(a + bc)^\star$



Principe



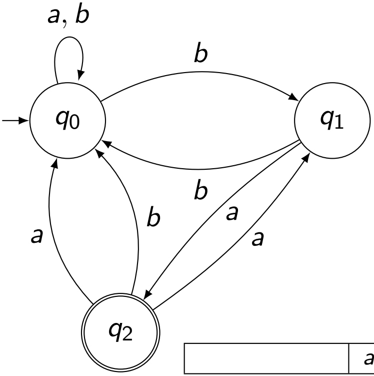
Simuler toutes les exécutions possibles:

$$q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_0, q_1 \xrightarrow{a} q_0, q_2 \xrightarrow{a} q_0, q_1 \xrightarrow{a} q_0, q_2$$
$$\xrightarrow{b} q_0, q_1 \xrightarrow{b} q_0, q_1 \xrightarrow{a} q_0, q_2$$

Méthode

1. $S_0 := I$ état initial
2. pour chaque état S et chaque lettre a, calculer S' ensemble des états atteints depuis les états de S sur a
3. continuer tant que de nouveau états sont découverts
4. les états finaux sont les ensembles contenant au moins un état final

Exemple 1

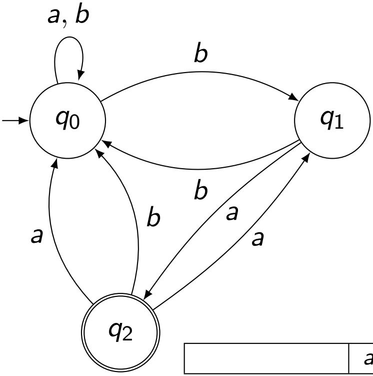


	a	b
$S_0 = \{q_0\}$		

Automates finis non déterministes

Du non-déterminisme au déterminisme

Exemple 1

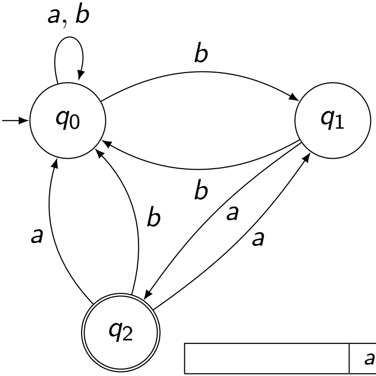


	a	b
$S_0 = \{q_0\}$	$\{q_0\} = S_0$	$\{q_0, q_1\} = S_1$

Automates finis non déterministes

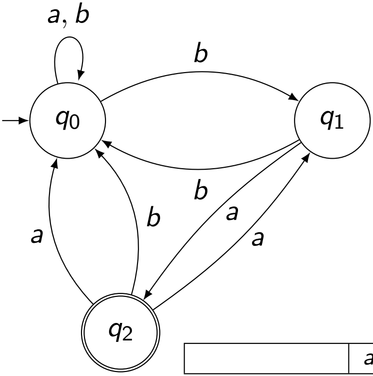
Du non-déterminisme au déterminisme

Exemple 1



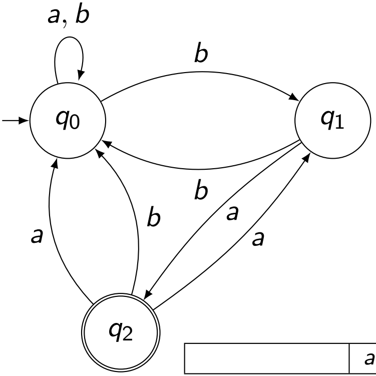
	a	b
$S_0 = \{q_0\}$	$\{q_0\} = S_0$	$\{q_0, q_1\} = S_1$
$S_1 = \{q_0, q_1\}$		

Exemple 1



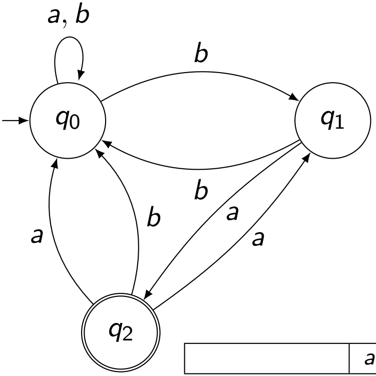
	a	b
$S_0 = \{q_0\}$	$\{q_0\} = S_0$	$\{q_0, q_1\} = S_1$
$S_1 = \{q_0, q_1\}$	$\{q_0, q_2\} = S_2$	

Exemple 1



	a	b
$S_0 = \{q_0\}$	$\{q_0\} = S_0$	$\{q_0, q_1\} = S_1$
$S_1 = \{q_0, q_1\}$	$\{q_0, q_2\} = S_2$	$\{q_0, q_1\} = S_1$
$S_2 = \{q_0, q_2\}$		

Exemple 1



	a	b
$S_0 = \{q_0\}$	$\{q_0\} = S_0$	$\{q_0, q_1\} = S_1$
$S_1 = \{q_0, q_1\}$	$\{q_0, q_2\} = S_2$	$\{q_0, q_1\} = S_1$
$S_2 = \{q_0, q_2\}$	$\{q_0, q_1\} = S_1$	$\{q_0, q_1\} = S_1$

Automates finis non déterministes

Du non-déterminisme au déterminisme

Exemple 2

q_0 initial

q_2 final

	a	b
q_0	q_0, q_2	q_1
q_1	q_0	q_2
q_2	q_0, q_1	

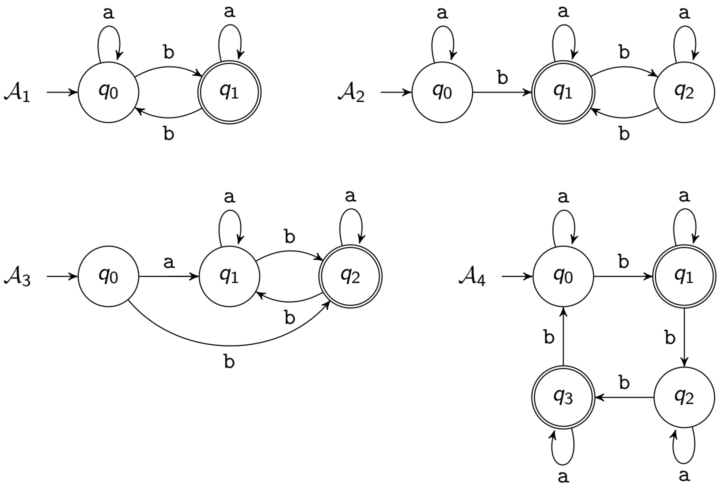
Version déterministe:

	a	b
$S_0 = \{q_0\}$	$\{q_0, q_2\} = S_1$	$\{q_1\} = S_2$
$S_1 = \{q_0, q_2\}$	$\{q_0, q_1, q_2\} = S_3$	$\{q_1\} = S_2$
$S_2 = \{q_1\}$	$\{q_0\} = S_0$	$\{q_2\} = S_4$
$S_3 = \{q_0, q_1, q_2\}$	$\{q_0, q_1, q_2\} = S_3$	$\{q_1, q_2\} = S_5$
$S_4 = \{q_2\}$	$\{q_0, q_1\} = S_6$	$\emptyset = S_7$
$S_5 = \{q_1, q_2\}$	$\{q_0, q_1\} = S_6$	$\{q_2\} = S_4$
$S_6 = \{q_0, q_1\}$	$\{q_0, q_2\} = S_1$	$\{q_1, q_2\} = S_5$
$S_7 = \emptyset$	$\emptyset = S_7$	$\emptyset = S_7$

S_0 initial $\quad S_1, S_3, S_4, S_5$ finaux

Compléments sur les automates

Mots sur $\{a, b\}$ avec nombre impair de b



$\mathcal{A}_1$ **minimal** (complet) en nombre d'états

Rappel

L_q: langage des mots acceptés à partir de l'état q

Principe

Si p et q états tels que $L_p = L_q$ alors p et q peuvent être *fusionnés*

Proposition

Si $\mathcal{A}$ automate **déterministe complet** où:

- $L_q \neq L_{q'}$ pour tout $q \neq q'$
- tous les états sont atteignables (depuis q_0)

alors $\mathcal{A}$ est minimal (complet) en nombre d'états

Problème

Partitionner l'ensemble des états tels que q et q' dans la même partie ssi $L_q = L_{q'}$

Algorithme de Moore

Principe: raffiner une partition

1. Supprimer les états non accessibles
2. Séparer états acceptants (contenant ε) et non acceptants
3. Diviser une partie S si pour une lettre a des états de S arrivent dans des parties différentes
4. Répéter 3 tant qu'il y a des changements

Exemple

	$\downarrow$ 0	1 $\star$	2	3 $\star$	4	5	6
a	1	3	1	3	1	2	4
b	3	0	6	6	5	4	2

Exemple

	$\downarrow$ 0	1⋆	2	3⋆	4	5	6
a	1	3	1	3	1	2	4
b	3	0	6	6	5	4	2

1. Début

A		B				
1	3	0	2	4	5	6

Exemple

	$\downarrow$ 0	1$\star$	2	3$\star$	4	5	6
a	1	3	1	3	1	2	4
b	3	0	6	6	5	4	2

1. Début

A		B				
1	3	0	2	4	5	6

2. Lettre a

	A		B				
	1	3	0	2	4	5	6
a	A	A	A	A	A	B	B

Exemple

	$\downarrow$ 0	1$\star$	2	3$\star$	4	5	6
a	1	3	1	3	1	2	4
b	3	0	6	6	5	4	2

2. Lettre a

	A		B				
	1	3	0	2	4	5	6
a	A	A	A	A	A	B	B

3. Division

A		C			D	
1	3	0	2	4	5	6

Exemple

	$\downarrow$ 0	1 $\star$	2	3 $\star$	4	5	6
a	1	3	1	3	1	2	4
b	3	0	6	6	5	4	2

3. Division

A		C			D	
1	3	0	2	4	5	6

4. Lettre b

	A		C			D	
	1	3	0	2	4	5	6
b	C	D	A	D	D	C	C

Exemple

	↓ 0	1⋆	2	3⋆	4	5	6
a	1	3	1	3	1	2	4
b	3	0	6	6	5	4	2

4. Lettre b

	A		C			D	
	1	3	0	2	4	5	6
b	C	D	A	D	D	C	C

5. Division

E	F	G	H		D	
1	3	0	2	4	5	6

Exemple

	↓ 0	1⋆	2	3⋆	4	5	6
a	1	3	1	3	1	2	4
b	3	0	6	6	5	4	2

5. Division

E	F	G	H		D	
1	3	0	2	4	5	6

6. Lettre a

	E	F	G	H		D	
	1	3	0	2	4	5	6
a	F	F	E	E	E	H	H

Exemple

	$\downarrow$ 0	1$\star$	2	3$\star$	4	5	6
a	1	3	1	3	1	2	4
b	3	0	6	6	5	4	2

6. Lettre `a`

	E	F	G	H		D	
	1	3	0	2	4	5	6
a	F	F	E	E	E	H	H

7. Lettre `b`

	E	F	G	H		D	
	1	3	0	2	4	5	6
b	G	D	F	D	D	H	H

Exemple

	$\downarrow$ 0	1$\star$	2	3$\star$	4	5	6
a	1	3	1	3	1	2	4
b	3	0	6	6	5	4	2

7. Lettre b

	E	F	G	H		D	
	1	3	0	2	4	5	6
b	G	D	F	D	D	H	H

8. Finalement

	$E\star$	$F\star$	$\downarrow G$	H	D
a	F	F	E	E	H
b	G	D	F	D	H

Compléments sur les automates

Fonctionnement de grep et Lex/jflex

Principe

À partir d'une expression régulière:

1. construction d'un automate non-déterministe
2. déterminisation de l'automate
3. minimisation de l'automate
4. utilisation de l'automate pour afficher/traiter les lignes qui correspondent

En pratique

Diverses optimisations et modifications, p. ex.:

- recherche des lignes *contenant un sous-texte* vérifiant une expression régulière
- différentiation des états acceptants (cas de Lex/Flex)
- etc.

Méthode 1

Si L rationnel alors:

- il est reconnu par un automate $\mathcal{A}$ à n états
- soit w dans L avec plus de n lettres
- la reconnaissance de w par $\mathcal{A}$ passe au moins deux fois par le même état q

$$\begin{tikzcd} q_0 \arrow[r, "w_1 \dots w_{i-1}"] & q \arrow[loop above, "w_i \dots w_j"] \arrow[r, "w_{j+1} \dots w_n"] & q_f \end{tikzcd}$$

- on peut alors répéter le sous-mot k fois (on *pompe*)

$$w_1 \dots w_{i-1}(w_i \dots w_j)^k w_{j+1} \dots w_n \quad \in L$$

Méthode 1: exemple 1

$L = \{a^k b^k \mid k \in \mathbb{N}\}$ n'est pas rationnel.

Par l'absurde, supposons L rationnel, alors:

- il est reconnu par un automate $\mathcal{A}$ à n états
- soit $w := a^{n+1} b^{n+1}$ dans L
- la reconnaissance de a^{n+1} par $\mathcal{A}$ passe au moins deux fois par le même état q

$$\begin{tikzcd} q_0 \arrow[r, "a^l"] & q \arrow[loop above, "a^k"] \arrow[r, "a^m b^{n+1}"] & q_f \end{tikzcd}$$

  avec $l + k + m = n + 1$ et $k > 0$
- alors (en pompant 2 fois) on a aussi $a^l a^k a^k a^m b^{n+1} \in L$ avec $l + k + k + m = n + 1 + k > n + 1$, contradiction $\square$

Méthode 1: exemple 2

$L = \{a^k b^l \mid k > l\}$ n'est pas rationnel.

Par l'absurde, supposons L rationnel, alors:

- il est reconnu par un automate $\mathcal{A}$ à n états
- soit $w := a^{n+1} b^n$ dans L
- la reconnaissance de a^{n+1} par $\mathcal{A}$ passe au moins deux fois par le même état q

$$\begin{tikzcd} q_0 \arrow[r, "a^l"] & q \arrow[loop above, "a^k"] \arrow[r, "a^m b^n"] & q_f \end{tikzcd}$$

  avec $l + k + m = n + 1$ et $k > 0$
- alors (en pompant 0 fois) on a aussi $a^l a^m b^n \in L$ avec $l + m \leq n$, contradiction $\square$

Méthode 1: exemple 2 bis

$L = \{a^k b^l \mid k > l\}$ n'est pas rationnel.

Par l'absurde, supposons L rationnel, alors:

- il est reconnu par un automate $\mathcal{A}$ à n états
- soit $w := a^{n+2}b^{n+1}$ dans L
- la reconnaissance de b^{n+1} par $\mathcal{A}$ passe au moins deux fois par le même état q

$$\begin{tikzcd} q_0 \arrow[r, "a^{n+2}b^l"] & q \arrow[loop above, "b^k"] \arrow[r, "b^m"] & q_f \end{tikzcd}$$

avec $l + k + m = n + 1$ et $k > 0$

- alors (en pompant 2 fois) on a aussi $a^{n+2}b^l b^k b^k b^m \in L$ avec $l + k + k + m = n + 1 + k \geq n + 2$, contradiction $\square$

Méthode 2

Si L rationnel alors:

- $\overline{L}$, $L^\star$, etc. rationnels
- soit L' rationnel alors $L \cap L'$, $L \cup L'$, LL', etc. sont rationnels

Méthode 2: exemple 1

$L = \{a^k b^l \mid k \neq l\}$ n'est pas rationnel.

Par l'absurde, supposons L rationnel, alors:

- $\overline{L}$ est rationnel
- soit $L' := a^\star b^\star$ rationnel
- alors $L' \setminus L = L' \cap \overline{L}$ rationnel
- or $L' \setminus L = a^\star b^\star \setminus L = \{a^k b^k \mid k \in \mathbb{N}\}$ qui n'est pas rationnel, contradiction

$\square$

Attention méthode 1 inadaptée

Pomper sur les a n'assure pas de sortir de L, de même sur les b.

Méthode 2: exemple 2

$L = \{a^n b^m \mid n = m$ ou alors $n \times m$ est pair$\}$ n'est pas rationnel.

$n = m$ et n et m impairs, ou bien $m \neq n$ et (n ou m est pair)

Par l'absurde, supposons L rationnel, alors:

- soit $L' := (aa)^\star b^\star + a^\star (bb)^\star$ rationnel
- alors $L \setminus L'$ rationnel
- or $L \setminus L' = \{a^k b^k \mid k$ impair$\}$ qui n'est pas rationnel (par méthode 1 ou 2), contradiction

□

Attention méthode 1 inadaptée

Pomper sur les a d'un mot avec un nombre pair de a peut produire un mot avec un nombre pair de a, donc n'assure pas de sortir de L.

Grammaires (hors contexte)

Grammaires (hors contexte)

Définitions et premières propriétés

Grammaire hors contexte

- un alphabet T des *symboles terminaux*
- un alphabet V des *symboles non-terminaux*
- un symbole de départ S de V
- un ensemble fini de *règles* $X \to w$ où
  - $X \in V$ et
  - w mot sur $V \cup T$

Exemples

$$\begin{array}{lcl} S & \to & ST \\ S & \to & \mathtt{a} \\ T & \to & \mathtt{b}S \end{array}$$

G_1

$$\begin{array}{lcl} S & \to & SS \mid \mathtt{a} \mid T \\ T & \to & TT \mid \mathtt{b}S \end{array}$$

G_2

Réécriture

$u \Rightarrow v$ avec u, v mots sur $V \cup T$ si

- $u = rXt$
- $v = rwt$
- $X \to w$

Exemples

- $\mathtt{a}S \Rightarrow \mathtt{aa}$
- $TT \Rightarrow T\mathtt{b}S$

$$
\begin{array}{ccc}
S & \to & ST \\
S & \to & \mathtt{a} \\
T & \to & \mathtt{b}S
\end{array}
$$

G_1

Dérivation

$u \Rightarrow^n v$ avec u, v mots sur $V \cup T$ s'il existe $u_0, u_1, \ldots, u_n$

- $u = u_0$
- $u_0 \Rightarrow u_1$

  $\vdots$
- $u_{n-1} \Rightarrow u_n$
- $v = u_n$

$u \Rightarrow^\star v$ s'il existe n tel que $u \Rightarrow^n v$

Exemple

- $S \Rightarrow^\star \mathtt{ababa}$ car:

$$\begin{aligned} \underline{S} &\Rightarrow \underline{S}T \Rightarrow \underline{S}TT \\ &\Rightarrow \mathtt{a}T\underline{T} \Rightarrow \mathtt{a}T\mathtt{b}\underline{S} \Rightarrow \mathtt{a}\underline{T}\mathtt{ba} \\ &\Rightarrow \mathtt{ab}\underline{S}\mathtt{ba} \Rightarrow \mathtt{ababa} \end{aligned}$$

$$\boxed{\begin{array}{ccc} S & \to & ST \\ S & \to & \mathtt{a} \\ T & \to & \mathtt{b}S \\ \\ & G_1 & \end{array}}$$

Langage algébrique

- $L(G) = \{u \in T^\star \mid S \Rightarrow^\star u\}$ *langage engendré par G*
- *langage algébrique*: engendré par une grammaire hors-contexte

Exemples

- $\mathtt{ababa} \in L(G_1)$ car $S \Rightarrow^\star \mathtt{ababa}$
- $\mathtt{b} \notin L(G_1)$ car pour tout n

  si $S \Rightarrow^n u$ alors u commence par a ou S

  (par récurrence sur n)

$$\begin{array}{ccc} S & \to & ST \\ S & \to & \mathtt{a} \\ T & \to & \mathtt{b}S \end{array}$$

G_1

Exemple

$$L(G_3) = \{a^n b^n \mid n \in \mathbb{N}\}$$

$$\boxed{\begin{array}{c} S \quad \to \quad aSb \mid \epsilon \\ \\ G_3 \end{array}}$$

car pour tout n

$$S \Rightarrow^n u \quad \Longleftrightarrow \quad u = a^n S b^n \text{ ou } u = a^{n-1} b^{n-1}$$

- $n = 0$

  $\boxed{\Rightarrow}$ $S \Rightarrow^0 S$ donc $u = S = a^0 S b^0$ convient

  $\boxed{\Leftarrow}$
  - si $u = a^0 S b^0 = S$ alors on a bien $S \Rightarrow^0 u$
  - le cas $u = a^{0-1} b^{0-1}$ ne peut pas arriver

Exemple

$$S \to aSb \mid \epsilon$$

G_3

$$L(G_3) = \{a^n b^n \mid n \in \mathbb{N}\}$$

car pour tout n

$$S \Rightarrow^n u \quad \Longleftrightarrow \quad u = a^n S b^n \text{ ou } u = a^{n-1} b^{n-1}$$

- récurrence: soit n tel que la propriété est vérifiée

  $\boxed{\Rightarrow}$ supposons $S \Rightarrow^{n+1} u$,

  alors $S \Rightarrow aSb \Rightarrow^n u$ avec $u = au'b$ et $S \Rightarrow^n u'$,

  donc par hypothèse de récurrence:

  - soit $u' = a^n S b^n$ et alors $u = a^{n+1} S b^{n+1}$,
  - ou bien $u' = a^{n-1} b^{n-1}$ et alors $u = a^n b^n$

Exemple

$$L(G_3) = \{a^n b^n \mid n \in \mathbb{N}\}$$

$$S \rightarrow aSb \mid \epsilon$$

G_3

car pour tout n

$$S \Rightarrow^n u \iff u = a^n S b^n \text{ ou } u = a^{n-1} b^{n-1}$$

- récurrence: soit n tel que la propriété est vérifiée

  $\boxed{\Leftarrow}$
  - si $u = a^{n+1} S b^{n+1}$ alors $S \Rightarrow aSb \Rightarrow^n a^{n+1} S b^{n+1}$
  - si $u = a^n b^n$ alors $S \Rightarrow aSb \Rightarrow^n a^n b^n$

Propriété

Tout langage rationnel est algébrique

Principe: simuler automate avec grammaire

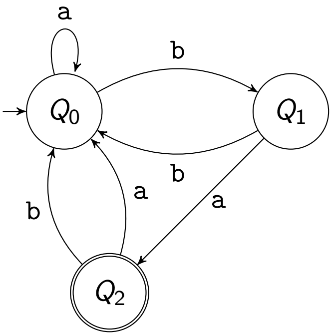
- symboles terminaux: alphabet de l'automate
- symboles non-terminaux: les états de l'automate
- symbole de départ: état intial de l'automate
- règles:
  - pour chaque transition $q \xrightarrow{a} q'$ une règle

    $$q \to aq'$$

  - pour chaque état final q une règle

    $$q \to \epsilon$$

Exemple



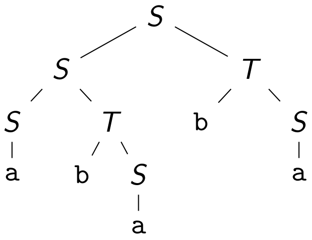
$$
\begin{array}{lcllll}
Q_0 & \rightarrow & \mathtt{a}Q_0 & | & \mathtt{b}Q_1 & \\
Q_1 & \rightarrow & \mathtt{a}Q_2 & | & \mathtt{b}Q_0 & \\
Q_2 & \rightarrow & \mathtt{a}Q_0 & | & \mathtt{b}Q_0 & | \quad \epsilon
\end{array}
$$

$$Q_0 \Rightarrow aQ_0 \Rightarrow abQ_1 \Rightarrow abbQ_0 \Rightarrow abbbQ_1 \Rightarrow abbbaQ_2 \Rightarrow abbba$$

Arbre de dérivation

- racine: symbole de départ
- nœuds internes: symboles non terminaux
- si X nœud interne avec fils $w_1, w_2, \ldots, w_n$ alors $X \to w_1 w_2 \ldots w_n$ règle

Exemples



$$\begin{array}{lcl} S & \to & ST \\ S & \to & \mathtt{a} \\ T & \to & \mathtt{b}S \end{array}$$

G_1

Grammaires (hors contexte)

Arbres de dérivation

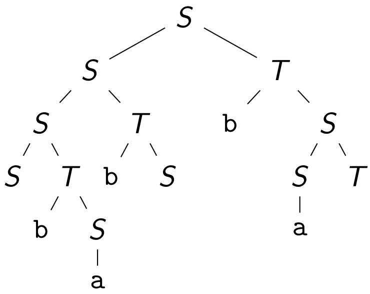
Arbre de dérivation

- racine: symbole de départ
- nœuds internes: symboles non terminaux
- si X nœud interne avec fils $w_1, w_2, \ldots, w_n$ alors $X \to w_1 w_2 \ldots w_n$ règle

Exemples

$$
\begin{array}{lcl}
S & \to & ST \\
S & \to & \mathtt{a} \\
T & \to & \mathtt{b}S
\end{array}
$$

G_1



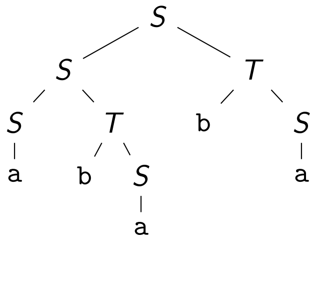
Grammaires (hors contexte)

Arbres de dérivation

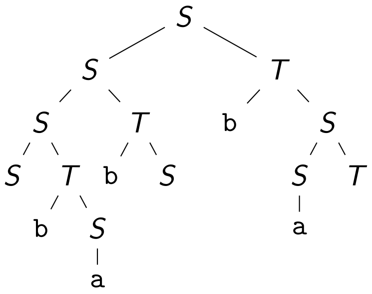
Arbre de dérivation et mot associé

- *mot associé à l'arbre*: feuilles de gauche à droite
- *arbre total*: que des terminaux aux feuilles
- *arbre partiel*: au moins une feuille avec non terminal

Exemples



arbre total pour $\texttt{ababa}$

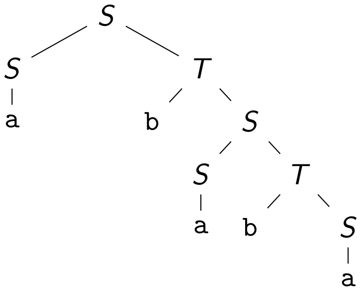


arbre partiel pour $S\texttt{bab}S\texttt{ba}T$

Propriété

$L(G) =$ mots correspondants à des arbres de dérivations totaux

Exemples

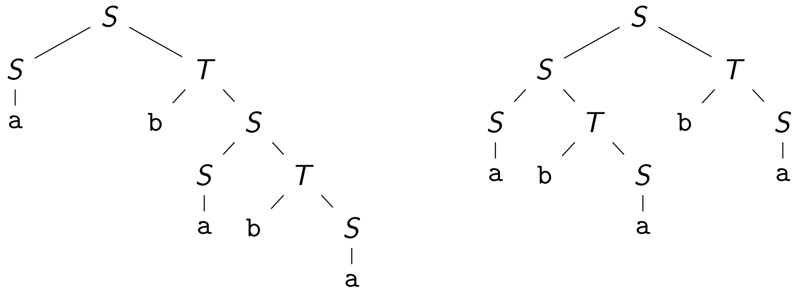


$$
\begin{array}{rcl}
S & \Rightarrow & \underline{S}T \\
 & \Rightarrow & a\underline{T} \\
 & \Rightarrow & ab\underline{S} \\
 & \Rightarrow & ab\underline{S}T \\
 & \Rightarrow & aba\underline{T} \\
 & \Rightarrow & abab\underline{S} \\
 & \Rightarrow & ababa
\end{array}
$$

Grammaire ambigüe

Au moins un mot admet plusieurs arbres de dérivation

Exemple



deux arbres de dérivation pour `ababa` donc G_1 ambigüe

Grammaires (hors contexte)

Algorithme CYK (cas sans ϵ)

Principe en trois étapes pour un mot $w_1 w_2 \ldots w_n$

1. *Transformation de la grammaire*: que des règles de la forme

$$\begin{array}{lcl} X & \rightarrow & YZ \\ X & \rightarrow & Y \\ X & \rightarrow & a \end{array}$$

2. *Calcul des clôtures*

$$Cl(X) : \text{ ensemble des } Y \text{ tels que } Y \Rightarrow^\star X$$

3. *Remplissage table $T[i,j]$*

$$T[i,j] : \text{ ensemble des } X \text{ tels que } X \Rightarrow^\star w_i \ldots w_{i+j}$$

d'où $S \Rightarrow^\star w_1 w_2 \ldots w_n$ ssi $T[1, n-1]$ contient S

Étape 1: Transformation de la grammaire

Que des règles de la forme

$$
\begin{array}{lcl}
X & \rightarrow & YZ \\
X & \rightarrow & Y \\
X & \rightarrow & a
\end{array}
$$

Exemple

$$
\boxed{
\begin{array}{lcl}
S & \rightarrow & TS \mid Ta \\
T & \rightarrow & aU \mid b \\
U & \rightarrow & S \mid UTU \mid cT
\end{array}
}
$$

G_4

Étape 1: Transformation de la grammaire

Que des règles de la forme

$$\begin{array}{lcl} X & \to & YZ \\ X & \to & Y \\ X & \to & a \end{array}$$

Exemple

$$\begin{array}{lcl} S & \to & TS \mid Ta \\ T & \to & aU \mid b \\ U & \to & S \mid UTU \mid cT \end{array}$$

G_4

- $U \to UTU$ devient

$$\begin{array}{lcl} U & \to & RU \\ R & \to & UT \end{array}$$

Grammaires (hors contexte)

Algorithme CYK (cas sans ϵ)

Étape 1: Transformation de la grammaire

Que des règles de la forme

$$\begin{array}{lcl} X & \to & YZ \\ X & \to & Y \\ X & \to & a \end{array}$$

Exemple

$$\begin{array}{lcl} S & \to & TS \mid Ta \\ T & \to & aU \mid b \\ U & \to & S \mid RU \mid cT \\ R & \to & UT \end{array}$$

G'_4

- $S \to Ta$ devient

$$\begin{array}{lcl} S & \to & TA \\ A & \to & a \end{array}$$

- $T \to aU$ devient

$$T \to AU$$

- etc.

Étape 1: Transformation de la grammaire

Que des règles de la forme

$$\begin{array}{lcl} X & \rightarrow & YZ \\ X & \rightarrow & Y \\ X & \rightarrow & a \end{array}$$

Exemple

$$\begin{array}{lcl} S & \rightarrow & TS \mid TA \\ T & \rightarrow & AU \mid b \\ U & \rightarrow & S \mid RU \mid CT \\ R & \rightarrow & UT \\ A & \rightarrow & a \\ C & \rightarrow & c \end{array}$$

G'_4

Initialement:

$$\begin{array}{lcl} S & \rightarrow & TS \mid Ta \\ T & \rightarrow & aU \mid b \\ U & \rightarrow & S \mid UTU \mid cT \end{array}$$

G_4

Étape 2: calcul des clôtures

$$Cl(X): \text{ ensemble des } Y \text{ tels que } Y \Rightarrow^\star X$$

Exemple

$$\begin{array}{lcl} S & \to & TS \mid TA \\ T & \to & AU \mid b \\ U & \to & S \mid RU \mid CT \\ R & \to & UT \\ A & \to & a \\ C & \to & c \end{array}$$

G'_4

X	$Cl(X)$
S	S
T	T
U	U
R	R
A	A
C	C

Étape 2: calcul des clôtures

$$Cl(X): \text{ ensemble des } Y \text{ tels que } Y \Rightarrow^\star X$$

Exemple

$$\begin{array}{lcl}
S & \rightarrow & TS \mid TA \\
T & \rightarrow & AU \mid b \\
\color{red}{U} & \rightarrow & \color{red}{S} \mid RU \mid CT \\
R & \rightarrow & UT \\
A & \rightarrow & a \\
C & \rightarrow & c
\end{array}$$

G'_4

X	$Cl(X)$
S	S
T	T
U	U
R	R
A	A
C	C

Étape 2: calcul des clôtures

$$Cl(X) : \text{ ensemble des } Y \text{ tels que } Y \Rightarrow^{\star} X$$

Exemple

$$\begin{array}{lcl} S & \to & TS \mid TA \\ T & \to & AU \mid b \\ U & \to & S \mid RU \mid CT \\ R & \to & UT \\ A & \to & a \\ C & \to & c \end{array}$$

G'_4

X	$Cl(X)$
S	S, U
T	T
U	U
R	R
A	A
C	C

Étape 3: remplissage table $T[i,j]$

$$T[i,j] : \text{ ensemble des } X \text{ tels que } X \Rightarrow^\star w_i \ldots w_{i+j}$$

Exemple

$$\begin{array}{lcl} S & \rightarrow & TS \mid TA \\ T & \rightarrow & AU \mid b \\ U & \rightarrow & S \mid RU \mid CT \\ R & \rightarrow & UT \\ A & \rightarrow & a \\ C & \rightarrow & c \end{array}$$

G'_4

a	b	a	b	c	b	b	a
A	T	A	T	C	T	T	A

$Cl(S) = \{U, S\}$

i les colonnes, j les lignes

Grammaires (hors contexte)

Algorithme CYK (cas sans ϵ)

Étape 3: remplissage table $T[i,j]$

$T[i,j]$: ensemble des X tels que $X \Rightarrow^\star w_i \dots w_{i+j}$

Exemple

$$
\begin{array}{lcl}
S & \to & TS \mid TA \\
T & \to & AU \mid b \\
U & \to & S \mid RU \mid CT \\
R & \to & UT \\
A & \to & a \\
C & \to & c
\end{array}
$$

G_4'

a	b	a	b	c	b	b	a
A	T	A	T	C	T	T	A
	S			U		S	

$Cl(S) = \{U, S\}$

i les colonnes, j les lignes

Grammaires (hors contexte)

Algorithme CYK (cas sans ϵ)

Étape 3: remplissage table $T[i,j]$

$$T[i,j] : \text{ ensemble des } X \text{ tels que } X \Rightarrow^\star w_i \dots w_{i+j}$$

Exemple

$$\begin{array}{lcl} S & \rightarrow & TS \mid TA \\ T & \rightarrow & AU \mid b \\ U & \rightarrow & S \mid RU \mid CT \\ R & \rightarrow & UT \\ A & \rightarrow & a \\ C & \rightarrow & c \end{array}$$

G'_4

a	b	a	b	c	b	b	a
A	T	A	T	C	T	T	A
	S,U			U		S,U	

$Cl(S) = \{U, S\}$

i les colonnes, j les lignes

Grammaires (hors contexte)

Algorithme CYK (cas sans ϵ)

Étape 3: remplissage table $T[i,j]$

$T[i,j]$: ensemble des X tels que $X \Rightarrow^{\star} w_i \dots w_{i+j}$

Exemple

$$
\begin{array}{lcl}
S & \rightarrow & TS \mid TA \\
T & \rightarrow & AU \mid b \\
U & \rightarrow & S \mid RU \mid CT \\
R & \rightarrow & UT \\
A & \rightarrow & a \\
C & \rightarrow & c
\end{array}
$$

G_4'

a	b	a	b	c	b	b	a
A	T	A	T	C	T	T	A
	S,U			U		S,U	
T	R			R	S		

$Cl(S) = \{U, S\}$

i les colonnes, j les lignes

Grammaires (hors contexte)

Algorithme CYK (cas sans ϵ)

Étape 3: remplissage table $T[i,j]$

$T[i,j]$: ensemble des X tels que $X \Rightarrow^\star w_i \dots w_{i+j}$

Exemple

$$\begin{array}{lcl} S & \rightarrow & TS \mid TA \\ T & \rightarrow & AU \mid b \\ U & \rightarrow & S \mid RU \mid CT \\ R & \rightarrow & UT \\ A & \rightarrow & a \\ C & \rightarrow & c \end{array}$$

G'_4

a	b	a	b	c	b	b	a
A	T	A	T	C	T	T	A
	S,U			U		S,U	
T	R			R	S,U		

$Cl(S) = \{U, S\}$

i les colonnes, j les lignes

Grammaires (hors contexte)

Algorithme CYK (cas sans ϵ)

Étape 3: remplissage table $T[i,j]$

$T[i,j]$: ensemble des X tels que $X \Rightarrow^\star w_i \dots w_{i+j}$

Exemple

$$
\begin{array}{lcl}
S & \to & TS \mid TA \\
T & \to & AU \mid b \\
U & \to & S \mid RU \mid CT \\
R & \to & UT \\
A & \to & a \\
C & \to & c
\end{array}
$$

G'_4

a	b	a	b	c	b	b	a
A	T	A	T	C	T	T	A
	S,U			U		S,U	
T	R			R	S,U		
	U						
T	R						
S,U							

$Cl(S) = \{U, S\}$

i les colonnes, j les lignes

Grammaires (hors contexte)

Algorithme CYK (cas sans ϵ)

Étape 3: remplissage table $T[i,j]$

$T[i,j]$: ensemble des X tels que $X \Rightarrow^\star w_i \dots w_{i+j}$

Exemple

$$
\begin{array}{lcl}
S & \to & TS \mid TA \\
T & \to & AU \mid b \\
U & \to & S \mid RU \mid CT \\
R & \to & UT \\
A & \to & a \\
C & \to & c
\end{array}
$$

G_4'

$Cl(S) = \{U, S\}$

i les colonnes, j les lignes

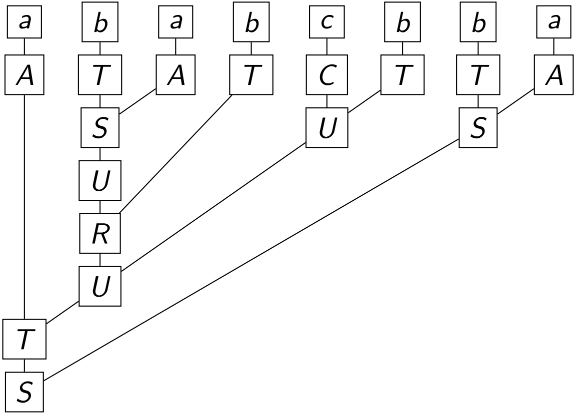
a	b	a	b	c	b	b	a
A	T	A	T	C	T	T	A
	S,U			U		S,U	
T	R			R	S,U		
	U						
T	R						
S,U							

donc $S \Rightarrow^\star ababcbba$

Propriétés

- complexité en $O(n^3)$: $O(n)$ pour chacune des $O(n^2)$ cases
- arbre de dérivation se déduit de la table

Exemple



Automates à pile

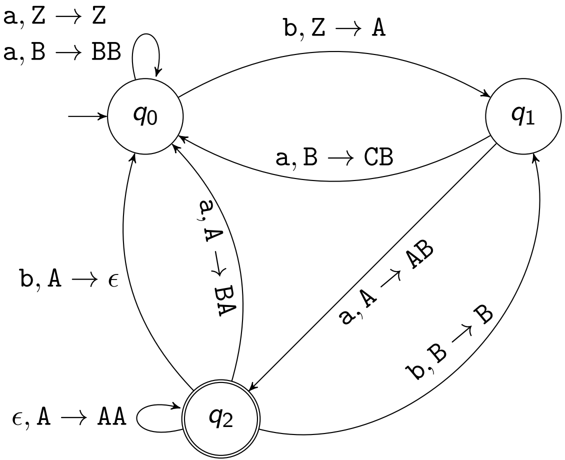
Automate à pile non déterministe

- un alphabet A
- un ensemble fini Q d'états
- un état initial q_0
- un ensemble d'états finaux $F \subseteq Q$
- un alphabet des symboles de la pile Γ
- un symbole de pile initial $\mathtt{Z}$
- une fonction de transition $\delta : Q \times (A \cup \{\epsilon\}) \times \Gamma \to \mathcal{P}(Q \times \Gamma^\star)$

Remarques

- $\delta(q, a, X)$ contient (q', W) signifie: « de l'état q en lisant a avec le symbole X sur la pile, on peut arriver dans l'état q' en remplaçant X par W sur la pile »
- sur l'automate la transition sera notée: $q \xrightarrow{\mathtt{a,X \to W}} q'$

Exemple (représentation graphique)



Exemple (représentation tabulaire)

q_0	a	b	ϵ
Z	q_0, Z	q_1, A	
A			
B	q_0, BB		

q_1	a	b	ϵ
Z			
A	q_2, AB		
B	q_0, CB		

q_2	a	b	ϵ
Z			
A	q_0, BA	q_0, ϵ	q_2, AA
B		q_1, B	

ou bien en un seul tableau:

δ	q_0, Z	q_0, A	q_0, B	q_1, Z	q_1, A	q_1, B	q_2, Z	q_2, A	q_2, B
a	q_0, Z		q_0, BB		q_2, AB	q_0, CB		q_0, BA	
b	q_1, A							q_0, ϵ	q_1, B
ϵ								q_2, AA	

et avec q_0 état initial, q_2 état final et Z symbole de pile initial

Définitions

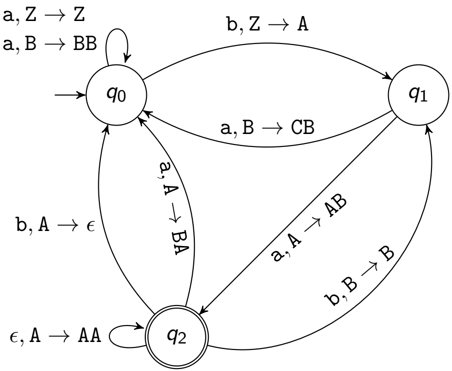
- *configuration:* un état et une pile (q, P)
- *transition:* $(q, P) \overset{a}{\mapsto} (q', P')$
  « de la configuration (q, X) en lisant a on atteint (q', W) »
- *suite de transitions:* $(q, P) \xmapsto{w_1 \dots w_n} (q', P')$ si
  - $(q, P) \overset{w_1}{\mapsto} (q_1, P_1)$
  - $(q_1, P_1) \overset{w_2}{\mapsto} (q_2, P_2)$

    $\vdots$
  - $(q_{n-1}, P_{n-1}) \overset{w_n}{\mapsto} (q_n, P_n)$

Remarque

Si `AAB` pile alors la tête est `A` et le reste est `AB`

Exemple



$$q_0, \mathtt{Z} \overset{\mathtt{a}}{\longmapsto} q_0, \mathtt{Z} \overset{\mathtt{b}}{\longmapsto} q_1, \mathtt{A} \overset{\mathtt{a}}{\longmapsto} q_2, \mathtt{AB} \overset{\epsilon}{\longmapsto} q_2, \mathtt{AAB} \overset{\mathtt{b}}{\longmapsto} q_0, \mathtt{AB}$$

donc

$$q_0, \mathtt{Z} \xmapsto{abab} q_0, \mathtt{AB}$$

Modes d'acceptation

Mot w accepté:

- *par état final* si

$$q_0, \mathrm{Z} \overset{w}{\mapsto} q_f, P$$

- *par pile vide* si

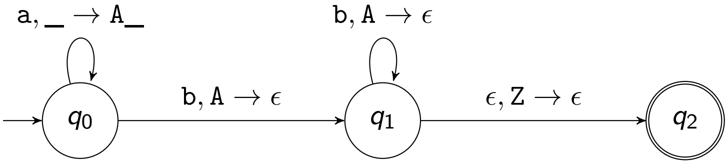
$$q_0, \mathrm{Z} \overset{w}{\mapsto} q, \epsilon$$

- *par état final et pile vide* si

$$q_0, \mathrm{Z} \overset{w}{\mapsto} q_f, \epsilon$$

avec q_f état final, et P pile quelconque

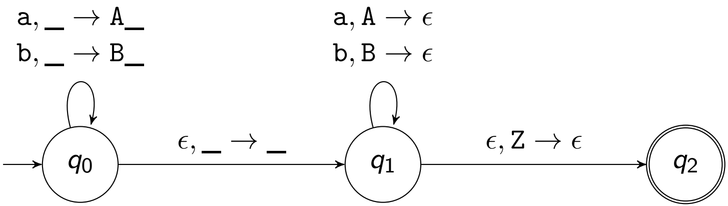
Exemple 1



$$\begin{aligned} q_0, \mathtt{Z} &\overset{a}{\mapsto} q_0, \mathtt{AZ} \overset{a}{\mapsto} q_0, \mathtt{AAZ} \overset{a}{\mapsto} \dots \overset{a}{\mapsto} q_0, \mathtt{A}^n\mathtt{Z} \\ &\overset{b}{\mapsto} q_1, \mathtt{A}^{n-1}\mathtt{Z} \overset{b}{\mapsto} q_1, \mathtt{A}^{n-2}\mathtt{Z} \overset{b}{\mapsto} \dots \overset{b}{\mapsto} q_1, \mathtt{Z} \overset{\epsilon}{\mapsto} q_2, \epsilon \end{aligned}$$

donc $a^n b^n$ reconnu par état final et pile vide

Exemple 2



$$
\begin{aligned}
q_0, \mathtt{Z} &\overset{a}{\vdash} q_0, \mathtt{AZ} \overset{a}{\vdash} q_0, \mathtt{AAZ} \overset{b}{\vdash} q_0, \mathtt{BAAZ} \overset{b}{\vdash} q_0, \mathtt{BBAAZ} \\
&\overset{\epsilon}{\vdash} q_1, \mathtt{BBAAZ} \\
&\overset{b}{\vdash} q_1, \mathtt{BAAZ} \overset{b}{\vdash} q_1, \mathtt{AAZ} \overset{a}{\vdash} q_1, \mathtt{AZ} \overset{a}{\vdash} q_1, \mathtt{Z} \overset{\epsilon}{\vdash} q_2, \epsilon
\end{aligned}
$$

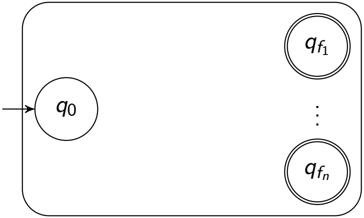
donc *aabbbbaa* reconnu par état final et pile vide

Proposition

Tout langage accepté selon un des trois modes est accepté par un autre mode

Preuve

- état final $\rightarrow$ pile vide



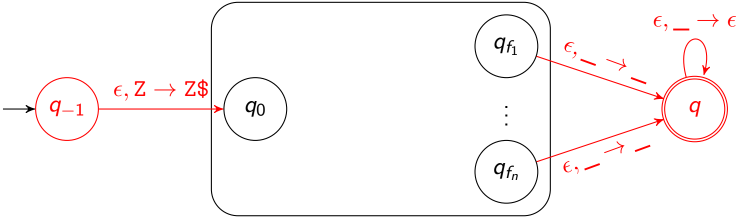
$$q_0, \mathrm{Z} \quad \overset{w}{\vdash\!\!\rightarrow} \quad q_{f_i}, P$$

Proposition

Tout langage accepté selon un des trois modes est accepté par un autre mode

Preuve

- état final $\rightarrow$ pile vide



$$q_{-1}, \mathtt{Z} \overset{\epsilon}{\vdash\!\!\rightarrow} q_0, \mathtt{Z\$} \overset{w}{\vdash\!\!\rightarrow} q_{f_i}, P\$ \overset{\epsilon}{\vdash\!\!\rightarrow} q, P\$ \overset{\epsilon}{\vdash\!\!\rightarrow} q, \epsilon$$

Automates à pile

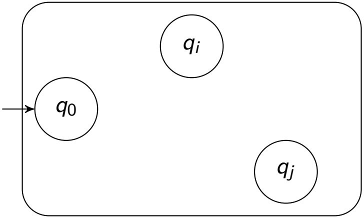
Exemples

Proposition

Tout langage accepté selon un des trois modes est accepté par un autre mode

Preuve

- pile vide → état final



$$q_0, Z \overset{w}{\vdash\!\!\rightarrow} q_i, \epsilon$$

Automates à pile

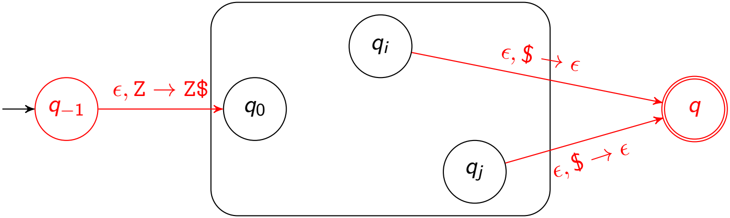
Exemples

Proposition

Tout langage accepté selon un des trois modes est accepté par un autre mode

Preuve

- pile vide $\rightarrow$ état final



$$q_{-1}, \mathrm{Z} \overset{\epsilon}{\mapsto} q_0, \mathrm{Z}\$ \overset{w}{\mapsto} q_i, \$ \overset{\epsilon}{\mapsto} q, \epsilon$$

Proposition

Tout langage accepté selon un des trois modes est accepté par un autre mode

Preuve

- Similaire pour les autres cas

Théorème

Tout langage algébrique est reconnu par un automate à pile.

Preuve

Principe: simuler une dérivation (gauche) dans la pile

Exemple:
$$\begin{array}{lcl} S & \rightarrow & SS \mid \mathtt{a} \mid T \\ T & \rightarrow & TT \mid US \mid \epsilon \\ U & \rightarrow & \mathtt{b} \mid ST \end{array}$$

$$S \rightarrow \underline{S}S \rightarrow \mathtt{a}\underline{S} \rightarrow \mathtt{a}\underline{T} \rightarrow \mathtt{a}\underline{U}S \rightarrow \mathtt{ab}\underline{S} \rightarrow \mathtt{aba}$$

simulé par

$$q, \mathtt{SZ} \overset{\epsilon}{\mapsto} q, \mathtt{SSZ} \overset{a}{\mapsto} q, \mathtt{SZ} \overset{\epsilon}{\mapsto} q, \mathtt{TZ} \overset{\epsilon}{\mapsto} q, \mathtt{USZ} \overset{b}{\mapsto} q, \mathtt{SZ} \overset{a}{\mapsto} q, \mathtt{Z}$$

Théorème

Tout langage algébrique est reconnu par un automate à pile.

Preuve

Principe: simuler une dérivation (gauche) dans la pile

1. Transformation de la grammaire pour n'avoir que

$$A \to \epsilon \qquad A \to B \qquad A \to BC \qquad A \to a$$

2. Automate à trois états: q_0 (initial), q_1, q_F (final)
   - $q_0 \xrightarrow{\epsilon, Z \to SZ} q_1$ pour amorcer
   - $q_1 \xrightarrow{\epsilon, Z \to \epsilon} q_F$ pour terminer
   - $q_1 \xrightarrow{\epsilon, A \to \epsilon} q_1$ pour $A \to \epsilon$
   - $q_1 \xrightarrow{\epsilon, A \to B} q_1$ pour $A \to B$
   - $q_1 \xrightarrow{\epsilon, A \to BC} q_1$ pour $A \to BC$
   - $q_1 \xrightarrow{a, A \to \epsilon} q_1$ pour $A \to a$

Automates à pile

Équivalence automates à pile et grammaires

Théorème

Tout langage reconnu par un automate à pile est algébrique

Preuve

Soit un automate acceptant par état final et pile vide et soit

$$X_p^q : \text{ mots } w \text{ tels que } p, X \overset{w}{\mapsto} q, \epsilon$$

- pour q_0 initial et tout q_F final ajout des règles

$$S \rightarrow Z_{q_0}^{q_F}$$

Automates à pile

Équivalence automates à pile et grammaires

Théorème

Tout langage reconnu par un automate à pile est algébrique

Preuve

Soit un automate acceptant par état final et pile vide et soit

$$X_p^q : \text{ mots } w \text{ tels que } p, X \overset{w}{\mapsto} q, \epsilon$$

- pour toute transition $q \xrightarrow{a, A \to \epsilon} q'$, comme

$$q, A \overset{a}{\mapsto} q', \epsilon$$

  alors ajout des règles

$$A_q^{q'} \to a$$

Théorème

Tout langage reconnu par un automate à pile est algébrique

Preuve

Soit un automate acceptant par état final et pile vide et soit

$$X_p^q : \text{ mots } w \text{ tels que } p, X \overset{w}{\mapsto} q, \epsilon$$

- pour toute transition $q \xrightarrow{a, A \to B} q'$, comme

$$q, A \overset{a}{\mapsto} q', B \overset{w}{\mapsto} p, \epsilon$$

  alors pour tout état p ajout des règles

$$A_q^p \to aB_{q'}^p$$

Automates à pile

Équivalence automates à pile et grammaires

Théorème

Tout langage reconnu par un automate à pile est algébrique

Preuve

Soit un automate acceptant par état final et pile vide et soit

$$X_p^q : \text{ mots } w \text{ tels que } p, X \overset{w}{\vdash\!\!\!\rightarrow} q, \epsilon$$

- pour toute transition $q \xrightarrow{a, A \to BC} q'$, comme

$$q, A \overset{a}{\vdash\!\!\!\rightarrow} q', BC \overset{w_1}{\vdash\!\!\!\longrightarrow} p, C \overset{w_2}{\vdash\!\!\!\longrightarrow} r, \epsilon$$

  alors pour tout état p et tout état r ajout des règles

$$A_q^r \to a B_{q'}^p C_p^r$$

Théorème
Tout langage reconnu par un automate à pile est algébrique

Exemple

$$
\begin{array}{rl|rcl}
 & q_0, \mathtt{Z} & S & \rightarrow & Z_0^2 \\
\xrightarrow{\mathtt{a}} & q_0, \mathtt{AZ} & & \rightarrow & \mathtt{a}A_0^1 Z_1^2 \\
\xrightarrow{\mathtt{b}} & q_0, \mathtt{BAZ} & & \rightarrow & \mathtt{ab}B_0^1 A_1^1 Z_1^2 \\
\xrightarrow{\epsilon} & q_1, \mathtt{BAZ} & & \rightarrow & \mathtt{ab}B_1^1 A_1^1 Z_1^2 \\
\xrightarrow{\mathtt{b}} & q_1, \mathtt{AZ} & & \rightarrow & \mathtt{abb}A_1^1 Z_1^2 \\
\xrightarrow{\mathtt{a}} & q_1, \mathtt{Z} & & \rightarrow & \mathtt{abba}Z_1^2 \\
\xrightarrow{\epsilon} & q_2, \epsilon & & \rightarrow & \mathtt{abba}
\end{array}
$$

Automates à pile

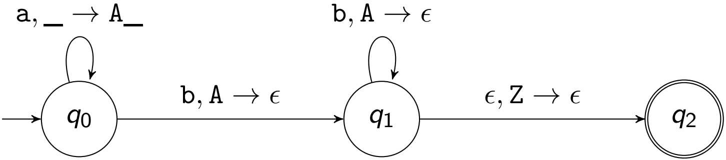
Automates déterministes

Automate à pile déterministe

Pour tout état q:

- si transition sortante sur a, X alors unique
- si transition sortante sur ϵ, X alors unique et aucun transition sur a, X quel que soit a

Exemple déterministe

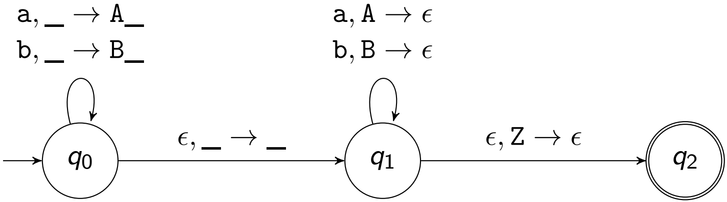


Automate à pile déterministe

Pour tout état q:

- si transition sortante sur a, X alors unique
- si transition sortante sur ϵ, X alors unique et aucun transition sur a, X quel que soit a

Exemple non déterministe



Automate à pile déterministe

Pour tout état q:

- si transition sortante sur a, X alors unique
- si transition sortante sur ϵ, X alors unique et aucun transition sur a, X quel que soit a

Théorème (admis)

Le langage des palindromes ne peut être reconnu par un automate à pile déterministe

Automates à pile déterministes et grammaires LR(0)/LR(1)

$$\boxed{\begin{array}{lcl} S & \rightarrow & TT \mid S\mathtt{c}T \mid \mathtt{a} \\ T & \rightarrow & \mathtt{c}T \mid \mathtt{b}S\mathtt{a} \end{array}}$$

b a a b a c b a a a

Automates à pile déterministes et grammaires LR(0)/LR(1)

Principes analyse ascendante

$$\boxed{\begin{array}{lcl} S & \rightarrow & TT \mid S\mathtt{c}T \mid \mathtt{a} \\ T & \rightarrow & \mathtt{c}T \mid \mathtt{b}S\mathtt{a} \end{array}}$$

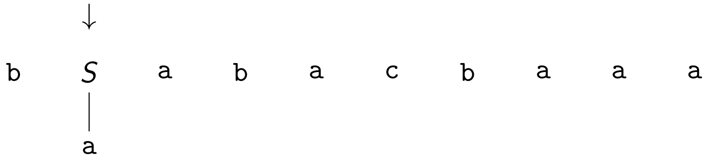
$\downarrow$

$\mathtt{b} \quad \mathtt{a} \quad \mathtt{a} \quad \mathtt{b} \quad \mathtt{a} \quad \mathtt{c} \quad \mathtt{b} \quad \mathtt{a} \quad \mathtt{a} \quad \mathtt{a}$

$$\boxed{\begin{array}{rcl} S & \to & TT \mid S\mathtt{c}T \mid \mathtt{a} \\ T & \to & \mathtt{c}T \mid \mathtt{b}S\mathtt{a} \end{array}}$$

	↓								
b	a	a	b	a	c	b	a	a	a

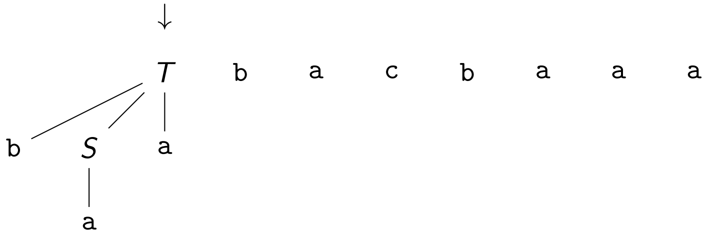
$$\boxed{\begin{array}{lcl} S & \rightarrow & TT \mid S\mathtt{c}T \mid \mathtt{a} \\ T & \rightarrow & \mathtt{c}T \mid \mathtt{b}S\mathtt{a} \end{array}}$$



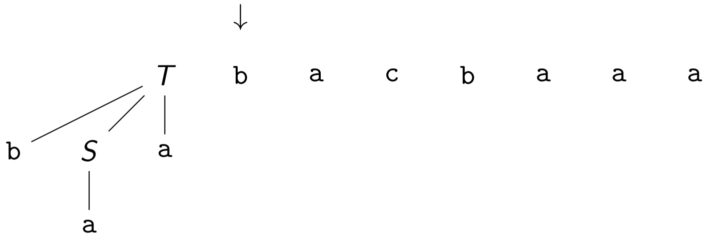
$$\boxed{\begin{array}{lcl} S & \rightarrow & TT \mid S\mathtt{c}T \mid \mathtt{a} \\ T & \rightarrow & \mathtt{c}T \mid \mathtt{b}S\mathtt{a} \end{array}}$$

$$\begin{array}{cccccccccc} & & \downarrow & & & & & & & \\ \mathtt{b} & S & \mathtt{a} & \mathtt{b} & \mathtt{a} & \mathtt{c} & \mathtt{b} & \mathtt{a} & \mathtt{a} & \mathtt{a} \\ & \vert & & & & & & & & \\ & \mathtt{a} & & & & & & & & \end{array}$$

$$\boxed{\begin{array}{lcl} S & \rightarrow & TT \mid S\mathtt{c}T \mid \mathtt{a} \\ T & \rightarrow & \mathtt{c}T \mid \mathtt{b}S\mathtt{a} \end{array}}$$

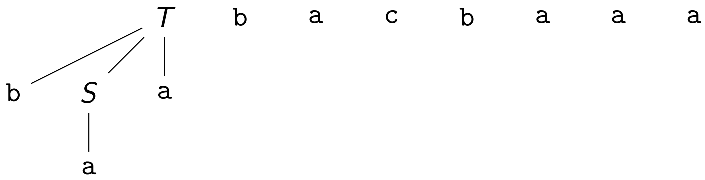


$$\boxed{\begin{array}{lcl} S & \rightarrow & TT \mid S\mathtt{c}T \mid \mathtt{a} \\ T & \rightarrow & \mathtt{c}T \mid \mathtt{b}S\mathtt{a} \end{array}}$$

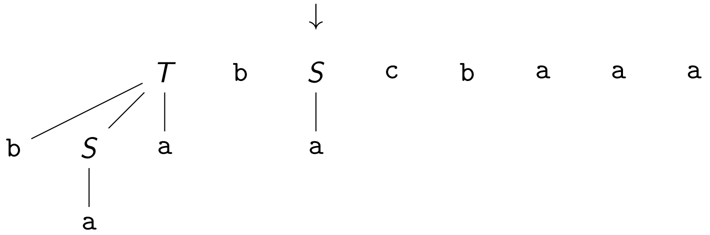


$$\boxed{\begin{array}{rcl} S & \rightarrow & TT \mid S\mathtt{c}T \mid \mathtt{a} \\ T & \rightarrow & \mathtt{c}T \mid \mathtt{b}S\mathtt{a} \end{array}}$$

$$\downarrow$$



$$
\boxed{
\begin{array}{rcl}
S & \rightarrow & TT \mid S\mathtt{c}T \mid \mathtt{a} \\
T & \rightarrow & \mathtt{c}T \mid \mathtt{b}S\mathtt{a}
\end{array}
}
$$

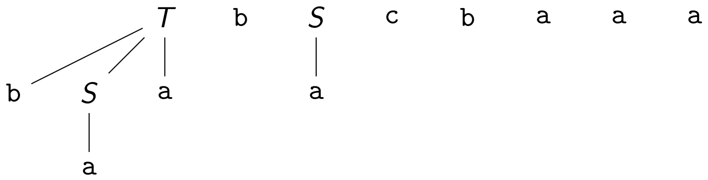


Automates à pile déterministes et grammaires LR(0)/LR(1)

Principes analyse ascendante

$$\boxed{\begin{array}{lcl} S & \rightarrow & TT \mid S\mathtt{c}T \mid \mathtt{a} \\ T & \rightarrow & \mathtt{c}T \mid \mathtt{b}S\mathtt{a} \end{array}}$$

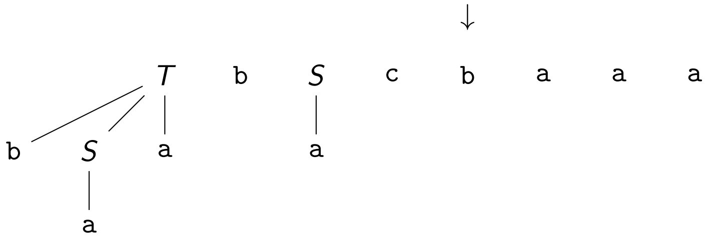
$$\downarrow$$



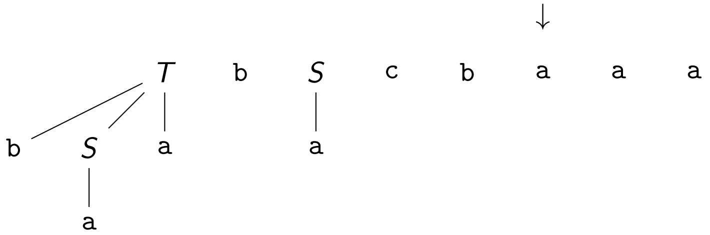
Automates à pile déterministes et grammaires LR(0)/LR(1)

Principes analyse ascendante

$$\boxed{\begin{array}{lcl} S & \rightarrow & TT \mid S\mathtt{c}T \mid \mathtt{a} \\ T & \rightarrow & \mathtt{c}T \mid \mathtt{b}S\mathtt{a} \end{array}}$$



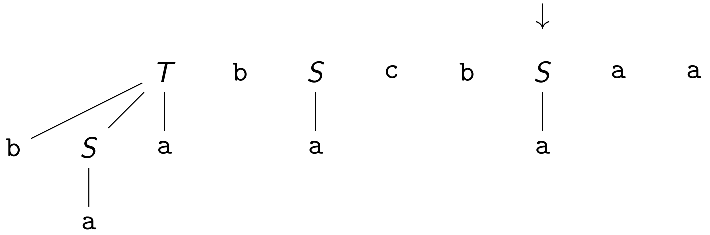
$$
\boxed{
\begin{array}{lcl}
S & \rightarrow & TT \mid S\mathtt{c}T \mid \mathtt{a} \\
T & \rightarrow & \mathtt{c}T \mid \mathtt{b}S\mathtt{a}
\end{array}
}
$$



Automates à pile déterministes et grammaires LR(0)/LR(1)

Principes analyse ascendante

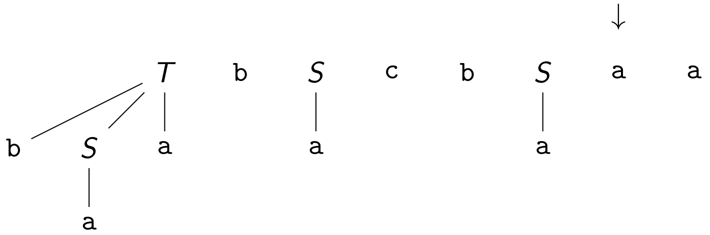
$$\boxed{\begin{array}{lcl} S & \rightarrow & TT \mid S\mathtt{c}T \mid \mathtt{a} \\ T & \rightarrow & \mathtt{c}T \mid \mathtt{b}S\mathtt{a} \end{array}}$$



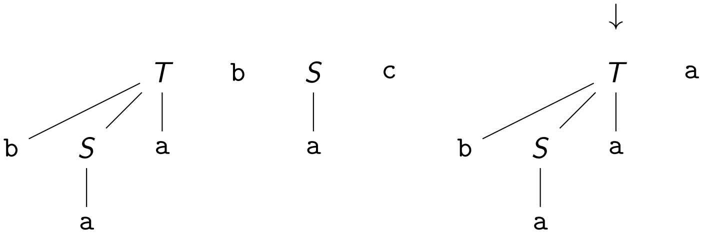
Automates à pile déterministes et grammaires LR(0)/LR(1)

Principes analyse ascendante

$$\boxed{\begin{array}{lcl} S & \rightarrow & TT \mid S\mathtt{c}T \mid \mathtt{a} \\ T & \rightarrow & \mathtt{c}T \mid \mathtt{b}S\mathtt{a} \end{array}}$$



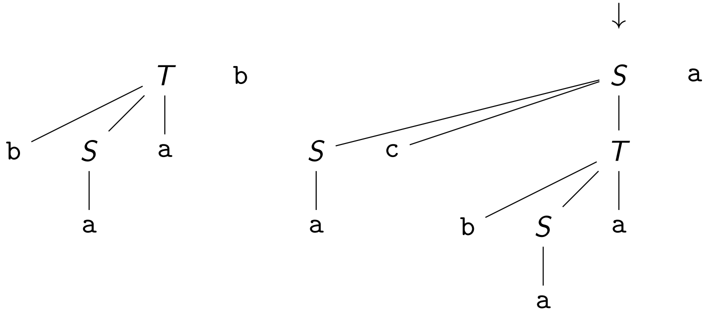
$$\boxed{\begin{array}{lcl} S & \rightarrow & TT \mid S\mathtt{c}T \mid \mathtt{a} \\ T & \rightarrow & \mathtt{c}T \mid \mathtt{b}S\mathtt{a} \end{array}}$$



Automates à pile déterministes et grammaires LR(0)/LR(1)

Principes analyse ascendante

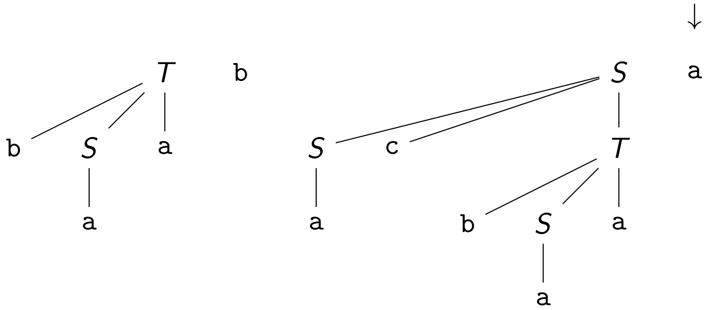
$$\boxed{\begin{array}{lcl} S & \rightarrow & TT \mid S\mathtt{c}T \mid \mathtt{a} \\ T & \rightarrow & \mathtt{c}T \mid \mathtt{b}S\mathtt{a} \end{array}}$$



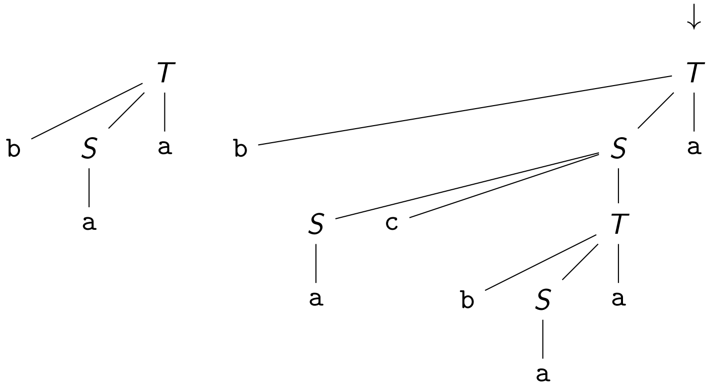
Automates à pile déterministes et grammaires LR(0)/LR(1)

Principes analyse ascendante

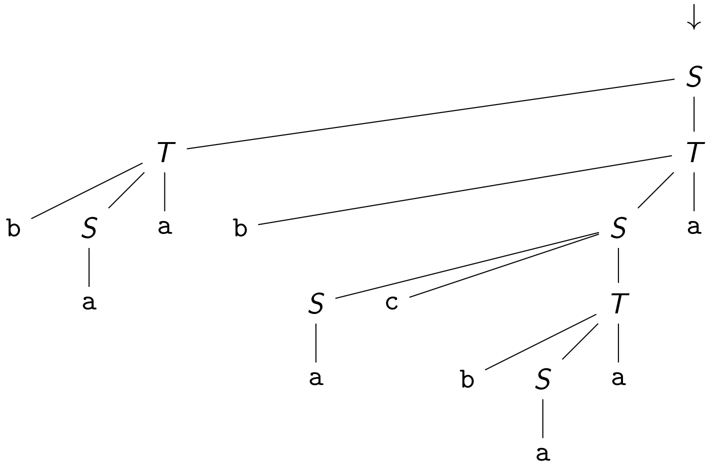
$$\boxed{\begin{array}{lcl} S & \rightarrow & TT \mid S\mathtt{c}T \mid \mathtt{a} \\ T & \rightarrow & \mathtt{c}T \mid \mathtt{b}S\mathtt{a} \end{array}}$$



$$\boxed{\begin{array}{lcl} S & \to & TT \mid S\texttt{c}T \mid \texttt{a} \\ T & \to & \texttt{c}T \mid \texttt{b}S\texttt{a} \end{array}}$$



$$\boxed{\begin{array}{lcl} S & \rightarrow & TT \mid S\mathtt{c}T \mid \mathtt{a} \\ T & \rightarrow & \mathtt{c}T \mid \mathtt{b}S\mathtt{a} \end{array}}$$



Dérivation droite

- $u \Rightarrow_d v$ si u obtenu depuis v par dérivation du non terminal le plus à droite
- $u \Rightarrow_d^\star v$ pour plusieurs dérivations successives

Exemple

L'arbre précédent correspond à la dérivation droite à l'envers:

$$\underline{S} \Rightarrow_d T\underline{T} \Rightarrow_d T\mathtt{b}\underline{S}\mathtt{a} \Rightarrow_d T\mathtt{b}S\mathtt{c}\underline{T}\mathtt{a} \Rightarrow_d T\mathtt{b}S\mathtt{cb}\underline{S}\mathtt{aa}$$

$$\Rightarrow_d T\mathtt{b}\underline{S}\mathtt{cbaaa} \Rightarrow_d \underline{T}\mathtt{bacbaaa} \Rightarrow_d \mathtt{b}\underline{S}\mathtt{abacbaaa}$$

$$\Rightarrow_d \mathtt{baabacbaaa}$$

Poignée

p *poignée* de w si

$$S \Rightarrow_d^\star rAt \Rightarrow_d rpt = w$$

avec $A \to p$ règle, w mot de $(N \cup T)^\star$, t mot de $T^\star$

Exemples

- poignée de $\mathtt{baabacbaaa}$ est $\mathtt{a}$ car $S \to \mathtt{a}$ et

$$S \Rightarrow_d^\star \mathtt{b}\underline{S}\mathtt{abacbaaa} \Rightarrow_d \mathtt{b}\underline{\mathtt{a}}\mathtt{abacbaaa}$$

- poignée de $T\mathtt{b}S\mathtt{cb}S\mathtt{aa}$ est $\mathtt{b}S\mathtt{a}$ car $T \to \mathtt{b}S\mathtt{a}$ et

$$S \Rightarrow_d^\star T\mathtt{b}S\mathtt{c}\underline{T}\mathtt{a} \Rightarrow_d T\mathtt{b}S\mathtt{c}\underline{\mathtt{b}S\mathtt{a}}\mathtt{a}$$

Poignée

p *poignée* de w si

$$S \Rightarrow_d^\star rAt \Rightarrow_d rpt = w$$

avec $A \to p$ règle, w mot de $(N \cup T)^\star$, t mot de $T^\star$

Remarques

- poignée toujours unique pour grammaire non ambigüe
- permet de remonter dérivation droite (réduction)

Grammaires LR(0)/LR(1)

- *LR*: Left to right, Rightmost derivation
- *LR(0)*: poignée déduite sans lire plus loin que la poignée
- *LR(1)*: poignée déduite en lisant une lettre de plus que la poignée

Théorème (admis)

Automates à pile déterministes équivalent aux grammaires LR(1)

Analyseur Shift/Reduce

Dit si un mot donné est accepté par une grammaire donnée:

- `Shift`: lit une lettre supplémentaire du mot d'entrée
- `Reduce`: applique réduction (remontée dans dérivation droite)

Construit y tel que $y \Rightarrow_d^\star u$ avec u lettres lues.

Exemple sur `baabacbaaa` (en colonnes)

$$
\begin{array}{llllll}
\epsilon \xmapsto{\texttt{Shift b}} & \texttt{b} & \xmapsto{\texttt{Shift a}} & T\texttt{ba} & \xmapsto{\texttt{Shift a}} & T\texttt{b}S\texttt{cb}S\texttt{a} \\
\xmapsto{\texttt{Shift a}} & \texttt{ba} & \xmapsto{\texttt{Reduce}} & T\texttt{b}S & \xmapsto{\texttt{Reduce}} & T\texttt{b}S\texttt{c}T \\
\xmapsto{\texttt{Reduce}} & \texttt{b}S & \xmapsto{\texttt{Shift c}} & T\texttt{b}S\texttt{c} & \xmapsto{\texttt{Shift a}} & T\texttt{b}S\texttt{c}T\texttt{a} \\
\xmapsto{\texttt{Shift a}} & \texttt{b}S\texttt{a} & \xmapsto{\texttt{Shift b}} & T\texttt{b}S\texttt{cb} & \xmapsto{\texttt{Reduce}} & T\texttt{b}S\texttt{a} \\
\xmapsto{\texttt{Reduce}} & T & \xmapsto{\texttt{Shift a}} & T\texttt{b}S\texttt{cba} & \xmapsto{\texttt{Reduce}} & TT \\
\xmapsto{\texttt{Shift b}} & T\texttt{b} & \xmapsto{\texttt{Reduce}} & T\texttt{b}S\texttt{cb}S & \xmapsto{\texttt{Reduce}} & S
\end{array}
$$

Item LR(0)

Règle et position dans la règle: $X \to \alpha \bullet \beta$

Item LR(0) valide pour mot

$X \to \alpha \bullet \beta$ valide pour y si

$$S \Rightarrow_d^\star rXt \Rightarrow_d r\alpha\beta t \text{ avec } r\alpha = y$$

Exemples

- $T \to \mathtt{c} \bullet T$ valide pour $T\mathtt{c}$ car

$$S \Rightarrow_d TT \Rightarrow_d T\mathtt{c}T$$

- $T \to \mathtt{b}S \bullet \mathtt{a}$ valide pour $\mathtt{b}S$ car

$$S \Rightarrow_d TT \Rightarrow_d T\mathtt{b}S\mathtt{a} \Rightarrow_d T\mathtt{baa} \Rightarrow_d \mathtt{b}S\mathtt{abaa}$$

Item LR(0)

Règle et position dans la règle: $X \to \alpha \bullet \beta$

Item LR(0) valide pour mot

$X \to \alpha \bullet \beta$ valide pour y si

$$S \Rightarrow_d^\star rXt \Rightarrow_d r\alpha\beta t \text{ avec } r\alpha = y$$

Exemples

- $T \to \mathtt{b}S \bullet \mathtt{a}$ valide pour $\mathtt{b}S$ car

$$S \Rightarrow_d TT \Rightarrow_d T\mathtt{b}S\mathtt{a} \Rightarrow_d T\mathtt{baa} \Rightarrow_d \mathtt{b}S\mathtt{abaa}$$

- $S \to S \bullet \mathtt{c}T$ valide pour $\mathtt{b}S$ car

$$S \Rightarrow_d TT \Rightarrow_d T\mathtt{b}S\mathtt{a} \Rightarrow_d T\mathtt{baa} \Rightarrow_d \mathtt{b}S\mathtt{abaa} \Rightarrow_d \mathtt{b}S\mathtt{c}T\mathtt{abaa}$$

Item LR(0)

Règle et position dans la règle: $X \to \alpha \bullet \beta$

Item LR(0) valide pour mot

$X \to \alpha \bullet \beta$ valide pour y si

$$S \Rightarrow_d^\star rXt \Rightarrow_d r\alpha\beta t \text{ avec } r\alpha = y$$

Utilité

- si $X \to \alpha \bullet a\beta$ valide pour y, opération `Shift` possible
- si $X \to \alpha\bullet$ valide pour y, opération `Reduce` possible

Remarque

Grammaire pas LR(0) *ssi* conflit `Shift`/`Reduce` ou `Reduce`/`Reduce`

Fermeture d'un item LR(0)

Obtenir la *fermeture* de $X \to \alpha \bullet Y\beta$ consiste à ajouter les items $Y \to \bullet\gamma$ pour toutes les règles $Y \to \gamma$ et réitérer si nécessaire

Exemples

$$\boxed{\begin{array}{lcl} S & \to & TT \mid S\mathtt{c}T \mid \mathtt{a} \\ T & \to & \mathtt{c}T \mid \mathtt{b}S\mathtt{a} \end{array}}$$

- fermeture de $S \to T \bullet T$ est

$$\{S \to T \bullet T, T \to \bullet\mathtt{c}T, T \to \bullet\mathtt{b}S\mathtt{a}\}$$

Fermeture d'un item LR(0)

Obtenir la *fermeture* de $X \to \alpha \bullet Y\beta$ consiste à ajouter les items $Y \to \bullet\gamma$ pour toutes les règles $Y \to \gamma$ et réitérer si nécessaire

Exemples

$$\boxed{\begin{array}{lcl} S & \to & TT \mid S\mathtt{c}T \mid \mathtt{a} \\ T & \to & \mathtt{c}T \mid \mathtt{b}S\mathtt{a} \end{array}}$$

- fermeture de $T \to \mathtt{b} \bullet S\mathtt{a}$ est

$$\{\; T \to \mathtt{b} \bullet S\mathtt{a},\; S \to \bullet TT,\; S \to \bullet S\mathtt{c}T,\; S \to \bullet\mathtt{a},\; \}$$

Fermeture d'un item LR(0)

Obtenir la *fermeture* de $X \to \alpha \bullet Y\beta$ consiste à ajouter les items $Y \to \bullet\gamma$ pour toutes les règles $Y \to \gamma$ et réitérer si nécessaire

Exemples

$$\boxed{\begin{array}{lcl} S & \to & TT \mid S\mathtt{c}T \mid \mathtt{a} \\ T & \to & \mathtt{c}T \mid \mathtt{b}S\mathtt{a} \end{array}}$$

- fermeture de $T \to \mathtt{b} \bullet S\mathtt{a}$ est

$$\begin{array}{l} \{\ T \to \mathtt{b} \bullet S\mathtt{a}, \\ \quad S \to \bullet TT, \\ \quad S \to \bullet S\mathtt{c}T, \\ \quad S \to \bullet\mathtt{a}, \\ \quad T \to \bullet\mathtt{c}T, \\ \quad T \to \bullet\mathtt{b}S\mathtt{a}\ \} \end{array}$$

Automates à pile déterministes et grammaires LR(0)/LR(1)

Calcul des items valides

Automate des items valides

Pour tout y, si $E_0 \xrightarrow{y} E$ alors E ensemble des items valides de y

Méthode de construction

- état initial: fermeture des items $S \to \bullet\gamma$
- $E \xrightarrow{x} E'$ avec E' obtenu par:

  si $X \to \alpha \bullet x\beta$ dans E alors $X \to \alpha x \bullet \beta$

  plus la fermeture associée

Exemple

$$\boxed{\begin{array}{lcl} S & \to & TT \mid S\mathtt{c}T \mid \mathtt{a} \\ T & \to & \mathtt{c}T \mid \mathtt{b}S\mathtt{a} \end{array}}$$

- état initial :

$$\begin{array}{|l|} \hline S \to \bullet TT \\ S \to \bullet S\mathtt{c}T \\ S \to \bullet \mathtt{a} \\ \hdashline T \to \bullet \mathtt{c}T \\ T \to \bullet \mathtt{b}S\mathtt{a} \\ \hline \end{array}$$

Exemple

$$\boxed{\begin{array}{lcl} S & \to & TT \mid S\mathtt{c}T \mid \mathtt{a} \\ T & \to & \mathtt{c}T \mid \mathtt{b}S\mathtt{a} \end{array}}$$

- transitions:

$$\boxed{\begin{array}{l} S \to \bullet TT \\ S \to \bullet S\mathtt{c}T \\ S \to \bullet \mathtt{a} \\ \hdashline T \to \bullet \mathtt{c}T \\ T \to \bullet \mathtt{b}S\mathtt{a} \end{array}} \xrightarrow{\quad T \quad} \boxed{\begin{array}{l} S \to T \bullet T \\ \hdashline T \to \bullet \mathtt{c}T \\ T \to \bullet \mathtt{b}S\mathtt{a} \end{array}}$$

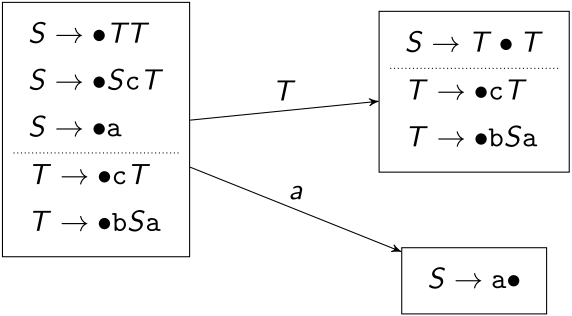
Automates à pile déterministes et grammaires LR(0)/LR(1)

Calcul des items valides

Exemple

$$
\begin{array}{|lcl|}
\hline
S & \rightarrow & TT \mid S\mathtt{c}T \mid \mathtt{a} \\
T & \rightarrow & \mathtt{c}T \mid \mathtt{b}S\mathtt{a} \\
\hline
\end{array}
$$

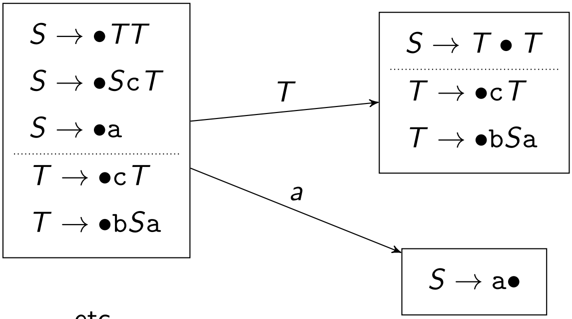
- transitions:



Exemple

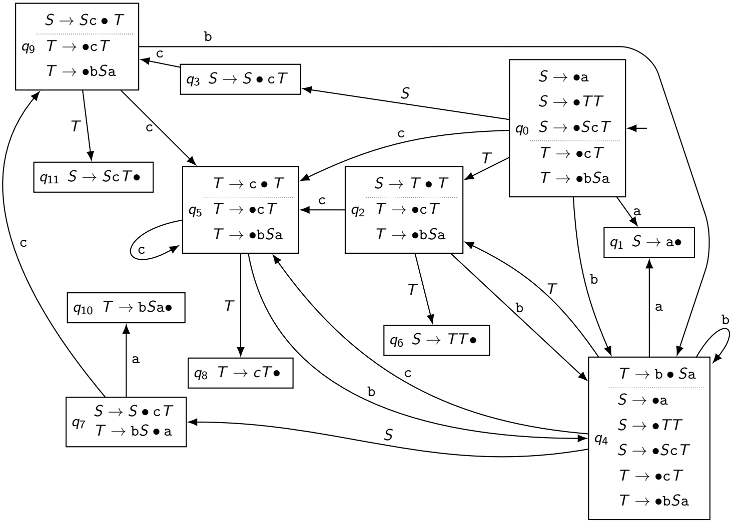
$$
\begin{array}{lcl}
S & \rightarrow & TT \mid S\mathtt{c}T \mid \mathtt{a} \\
T & \rightarrow & \mathtt{c}T \mid \mathtt{b}S\mathtt{a}
\end{array}
$$

- transitions:



etc.

Exemple complet



Analyse d'un mot

1. Au départ mot lu $y = \epsilon$
2. Calcul des items valides pour y (avec automate):
   - si aucun item valide, mot rejeté
   - si un item de la forme $X \to \alpha\bullet$ opération `Reduce`: remplaçement des dernières lettres α de y par X
   - sinon opération `Shift`: ajout prochaine lettre à lire dans y
3. Répétition jusqu'à lecture complète: mot accepté si $y = S$

Exemple

Analyse de acbaa, en ajoutant états dans y:

étape	opération	pile y
1		q_0
2	Shift	q_0 a q_1
3	Reduce $S \to a$	q_0 S q_3
4	Shift	q_0 S q_3 c q_9
5	Shift	q_0 S q_3 c q_9 b q_4
6	Shift	q_0 S q_3 c q_9 b q_4 a q_1
7	Reduce $S \to a$	q_0 S q_3 c q_9 b q_4 S q_7
8	Shift	q_0 S q_3 c q_9 b q_4 S q_7 a q_{10}
9	Reduce $T \to \mathtt{b}S\mathtt{a}$	q_0 S q_3 c q_9 T q_{11}
10	Reduce $S \to S\mathtt{c}T$	q_0 S q_3

Finalement: mot acbaa accepté

Item LR(1)

Règle, position dans la règle et symbole x: $X \to \alpha \bullet \beta, x$

Item LR(1) valide pour mot

$X \to \alpha \bullet \beta, x$ valide pour y si

$$S \Rightarrow_d^\star rXt \Rightarrow_d r\alpha\beta t \text{ avec } r\alpha = y \text{ et } t_1 = x$$

Fermeture d'un item LR(1)

Obtenir la fermeture de $X \to \alpha \bullet Y\beta, x$ consiste à ajouter les items $Y \to \bullet\gamma, y$ pour toutes les règles $Y \to \gamma$ *et tous les symboles y pouvant débuter βx*, et réitérer si nécessaire

Remarques

- automate des items valides similaire (mais plus gros)
- symbole supplémentaire permet de résoudre des conflits

Exemple

$$\boxed{\begin{array}{lcl} S & \rightarrow & \mathtt{a} \mid \mathtt{a}SN \\ N & \rightarrow & \mathtt{b}S \end{array}}$$

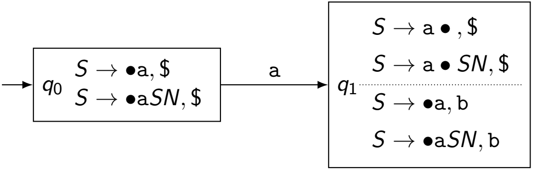
$$\rightarrow q_0 \; \boxed{\begin{array}{l} S \rightarrow \bullet\mathtt{a}, \$ \\ S \rightarrow \bullet\mathtt{a}SN, \$ \end{array}}$$

Automates à pile déterministes et grammaires LR(0)/LR(1)

Items LR(1), LALR(1), SLR(1)

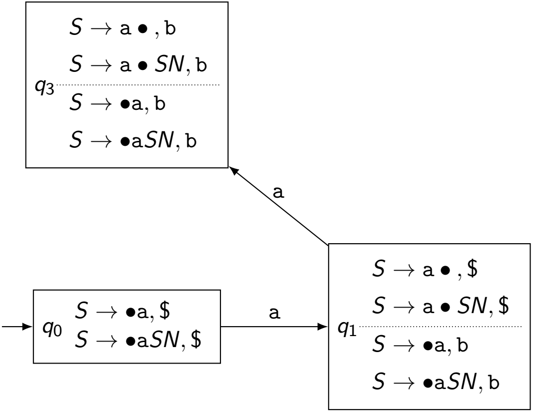
Exemple

$$\begin{array}{|lcl|} S & \to & \mathtt{a} \mid \mathtt{a}SN \\ N & \to & \mathtt{b}S \end{array}$$



Exemple

$$\begin{array}{lcl} S & \rightarrow & \mathtt{a} \mid \mathtt{a}SN \\ N & \rightarrow & \mathtt{b}S \end{array}$$

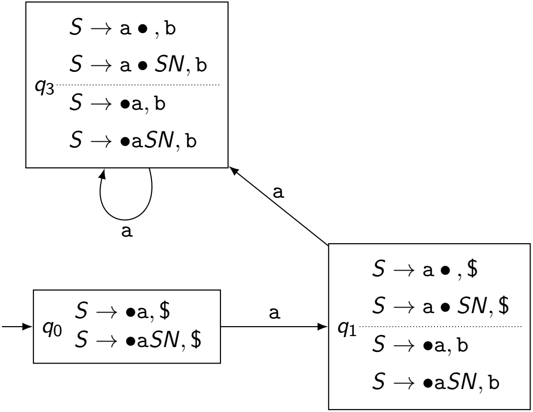


Automates à pile déterministes et grammaires LR(0)/LR(1)

Items LR(1), LALR(1), SLR(1)

Exemple

$$\begin{array}{lcl} S & \rightarrow & \mathtt{a} \mid \mathtt{a}SN \\ N & \rightarrow & \mathtt{b}S \end{array}$$

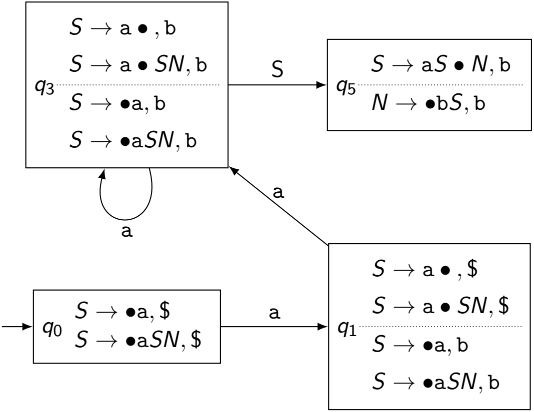


Automates à pile déterministes et grammaires LR(0)/LR(1)

Items LR(1), LALR(1), SLR(1)

Exemple

$$\begin{array}{lcl} S & \to & \mathtt{a} \mid \mathtt{a}SN \\ N & \to & \mathtt{b}S \end{array}$$



Exemple complet