Théorie des langages

Denis Monnerat
monnerat@u-pec.fr

3 avril 2020

IUT de Fontainebleau

Première partie I

Alphabets et langages

Alphabets et langages

Alphabets et langages

Introduction

Alphabet

Alphabet

Un alphabet X est un ensemble (fini) de symboles, appelés des lettres.

Exemples :

- $X = \{0, 1\}$
- $X = \{a, b, c\}$
- $X = \{p, q, \wedge, \vee, \rightarrow\}$

Alphabets et langages

Mot

Mot

Mot

Un mot de longueur k est une suite ordonnée (éventuellement vide) de k lettres.

$$u = (a_1, a_2, ..., a_k)$$

Notation condensée

$$u = a_1 a_2 \cdots a_k$$

Par exemple, *abaaab* et *ccca* sont des mots formés sur l'alphabet $X = \{a, b, c\}$.

- L'ensemble des mots sur l'alphabet X est noté X^*
- Le mot vide, de longueur 0, est noté ε (ou encore Λ)
- On note $|w|$ la longueur d'un mot
- On note $|w|_s$ le nombre d'occurences de la lettre s dans w

Exemple 1

Pour $X = \{a, b, c\}$,

$$X^* = \{\varepsilon, a, b, c, aa, ab, ac, ba, bb, bc, ca, cb, cc, aaa, aab, \ldots\}$$

Exemple 2

avec $X = \{a, b, c\}$

- $|abbac| = 5$
- $|ba| = 2$
- $|\varepsilon| = 0$
- $|abbac|_a = 2$

Alphabets et langages

Opérations sur les mots

Concaténation

Concaténation

On définit sur X^* une loi de composition interne appelée concaténation ou produit. Elle associe à deux mots $a_1 \cdots a_n$ et $b_1 \cdots b_m$ le mot

$$a_1 \cdots a_n b_1 \cdots b_m$$

(de longueur $n + m$).

Propriétés

- le produit est associatif

$$\forall \alpha, \beta, \gamma \in X^*, \quad (\alpha\beta)\gamma = \alpha(\beta\gamma)$$

- ε est l'élément neutre du produit
- $|w.z| = |w| + |z|$

Puissance

Puissance

La puissance d'un mot est définie par $m^0 = \varepsilon$ et $m^{n+1} = m.m^n$.

Par exemple, soit $X = \{a, b\}$ et $w = abb$

- $w^0 = \varepsilon$
- $w^1 = abb$
- $w^2 = w.w = abbabb$
- $w^3 = w.w^2 = w^2.w = abbabbabb$

Égalité

Égalité

Deux mots sont égaux s'ils sont de même longueur et s'ils ont des lettres identiques de positionnements identiques.

Facteurs

Facteurs

Etant donnée trois mots α, β et γ définis sur l'alphabet X,

- β est un facteur (sous-mot) de $\alpha\beta\gamma$.
- α est un préfixe (facteur gauche) de $\alpha\beta$.
- β est un suffixe (facteur droit) de $\alpha\beta$.

Soit $X = \{a, b\}$, et $w = babb$

- les préfixes de w sont ε, b, ba, bab, $babb$
- les suffixes de w sont ε, b, bb, abb, $babb$
- les facteurs sont ε, b, a, ba, ab, bb, bab, abb, $babb$

On parle aussi de préfixes,suffixes et facteurs propres.

Alphabets et langages

Langages

Monoïde

Monoïde

Soit X un alphabet.

$$(X^*, .)$$

est un monoïde (loi associative et élément neutre) libre.

Langages

On note X^* l'ensemble de tous les mots possibles sur l'alphabet X. Un langage L sur X est un ensemble de mots sur X

Langages

L est un langage sur $X \Leftrightarrow L \subset X^* \Leftrightarrow L \in P(X^*)$

Exemples

Sur $X = \{a, b\}$,

- $\emptyset$ est un langage
- $\{\varepsilon\}$ est un langage
- $\{a, ba, ababa\}$ est un langage
- $\{\epsilon, a, a^2, a^3, \ldots\} = \{a^n\}$ est un langage
- $\{\varepsilon, ab, aabb, aaabbb, \ldots\} = \{a^n b^n\}$ est un langage

Exemples de langages

Langage propositionnel

$$X = \{p, q, r, s, \ldots, \wedge, \vee, \neg, \rightarrow, \leftrightarrow, (,)\}$$

- Une formule propositionnelle peut être représentée par un mot de X^*.
- Le langage L des formules propositionnelles bien formées $\subset X^*$.

- $((p \vee q) \rightarrow (p \vee \neg r)) \in L$
- $\neg pq \wedge \vee \notin L$

Question : comment définir un langage, et décider si un mot en fait partie ?

Opérations sur les langages

- l'union de deux langages L_1 et L_2 :

$$L_1 + L_2 = L_1 \cup L_2 = \{x \mid x \in L_1 \vee x \in L_2\}$$

- l'intersection de deux langages L_1 et L_2 :

$$L_1 \cap L_2 = \{x \mid x \in L_1 \wedge x \in L_2\}$$

- le complémentaire d'un langage L_1 :

$$\overline{L_1} = \{x \mid x \notin L_1\}$$

- le produit de deux langages L_1 et L_2 :

$$L_1.L_2 = \{xy \mid x \in L_1,\ y \in L_2\}$$

- l'étoile (fermeture de Kleene) du langage L (i.e le monoïde engendré par L) :

$$L^* = \cup_{n \geq 0} L^n$$

- fermeture positive du langage L :

$$L^+ = \cup_{n \geq 1} L^n$$

Deuxième partie II

Langages Rationnels-Automates

AFD

Représentation algébrique

AFND

Equivalence AFD-AFND

Minimisation

Propriété de cloture

AFD

Automate fini déterministe

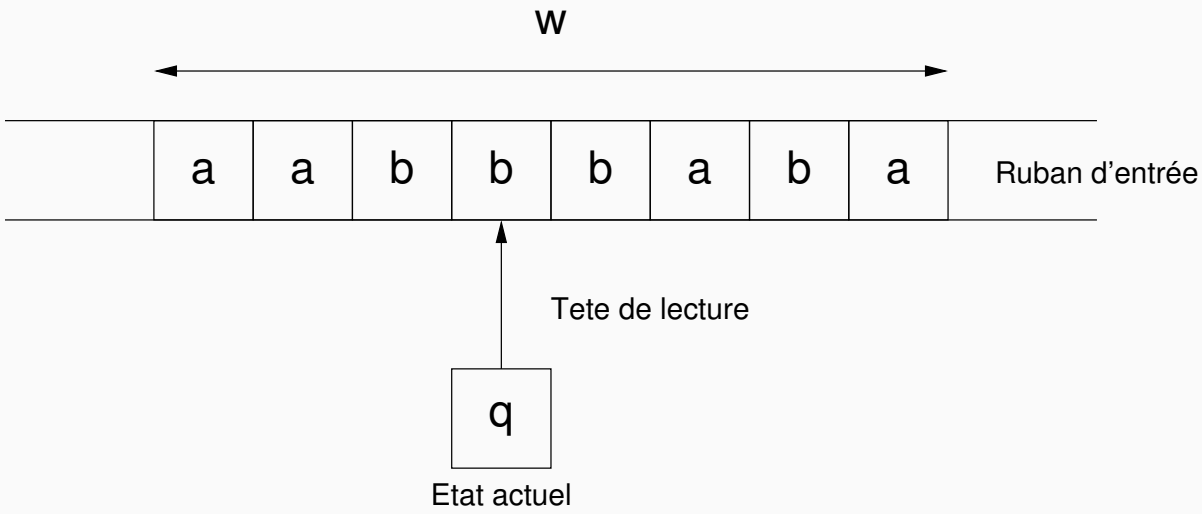
AFD

Un *automate fini déterministe* A (AFD) est un la donnée de :

- X : un alphabet de symboles reconnus.
- Q : un ensemble **fini** d'états.
- $q_0 \in Q$ l'état initial.
- $F \subset Q$ l'ensemble des états terminaux ou acceptants de l'automate.
- $\delta : Q \times X \longrightarrow Q$: une application appelé fonction de transition de l'automate.

On note $A = (X, Q, q_0, F, \delta)$

Remarque : on peut tolérer que la fonction de transition ne soit pas complète (i.e, sur certains états, il manque des transitions). Cela siginifie implicitement que ces transitions manquantes sont des transitions qui ménent à un *état poubelle* absorbant, non acceptant. L'automate est dit émondé.



- w est le mot initial à lire lettre par lettre.
- L'automate est initialement à l'état q_0.
- Considérant le couple actuel (état actuel, symbole lu), l'automate change d'état comme indiqué par la fonction de transition.
- Si après avoir lu le mot en entier, l'automate se trouve dans un état terminal, le mot est accepté, sinon rejeté.

Remarque : on peut étendre la fonction de transition δ à $Q \times X^*$, qu'on notera δ^*, par la définition :

$$\delta^*(q, \epsilon) = q \text{ et } \delta^*(q, ux) = \delta(\delta^*(q, u), x)$$

Cette fonction renvoie l'état après la lecture de tout un mot.

Langage rationnel

Langage reconnu par un AFD

La *langage reconnu* par un AFD $A = (X, Q, q_0, F, \delta)$ est par définition

$$L(A) = \{w \in X^* \mid \delta^*(q_0, w) \in F\}$$

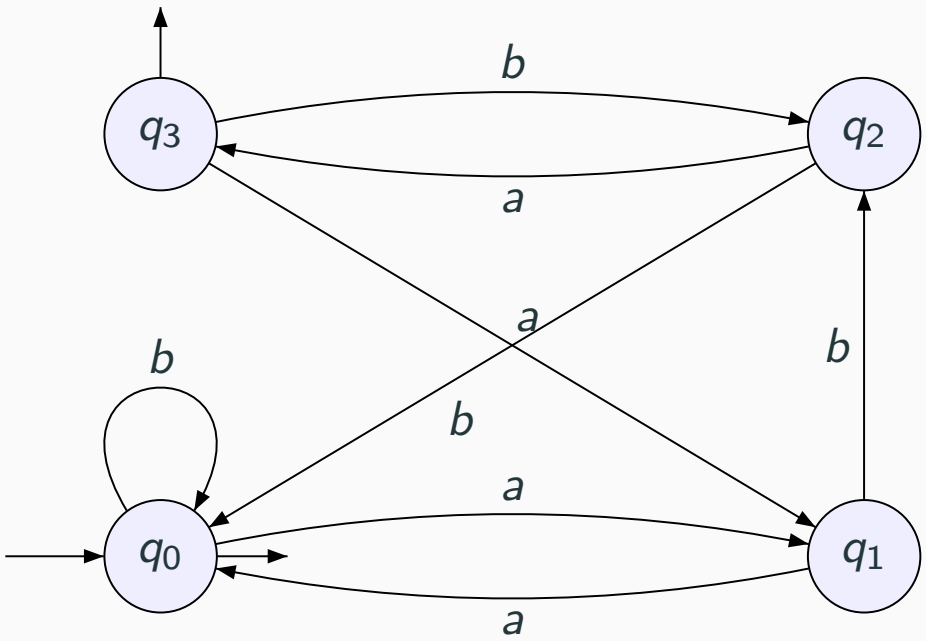
Par définition, un langage L est dit **rationnel**, ou régulier, s'il existe un automate fini déterministe A tel que

$$L = L(A)$$

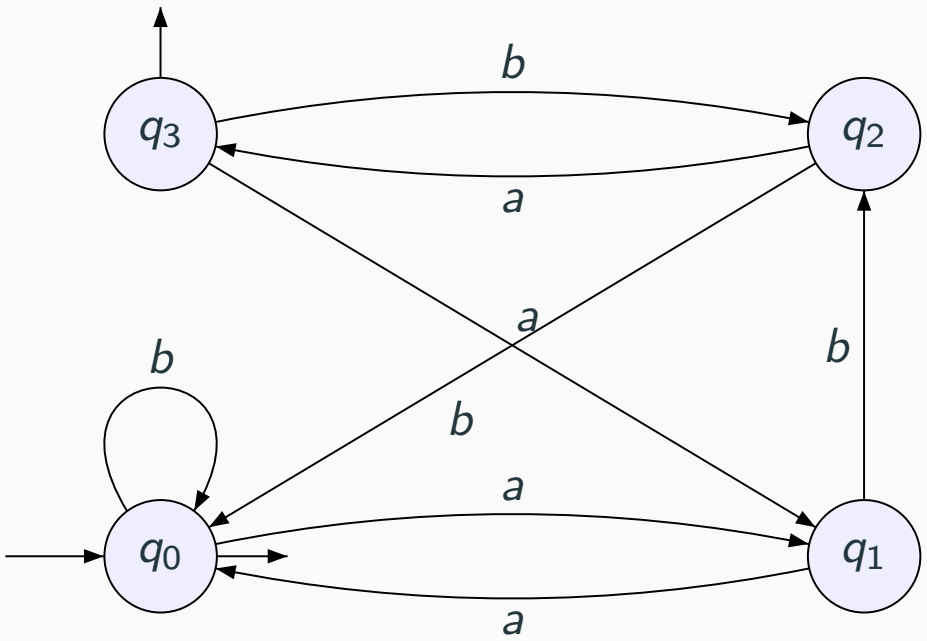
Exemples

- les chaînes se terminant par ab.
- les chaînes sur $\{0, 1\}$ qui représentent un mulitple de 5.
- Les chaînes formés de $6k + 1$ fois le symbole a.

Représentation

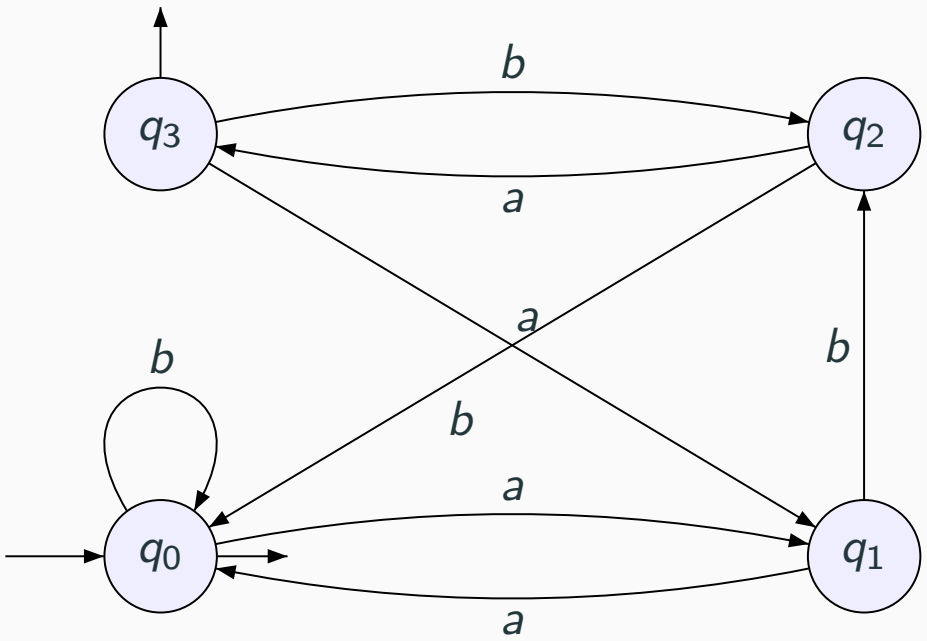


Représentation



δ	a	b
q_0	q_1	q_0
q_1	q_0	q_2
q_2	q_3	q_0
q_3	q_1	q_2

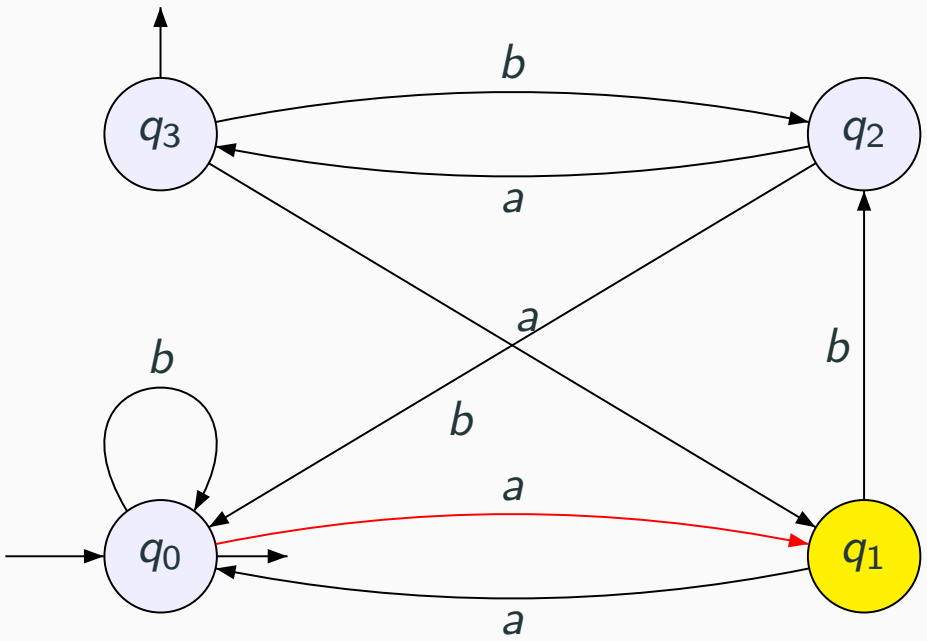
Représentation



δ	a	b
q_0	q_1	q_0
q_1	q_0	q_2
q_2	q_3	q_0
q_3	q_1	q_2

mot en entrée *ababa*

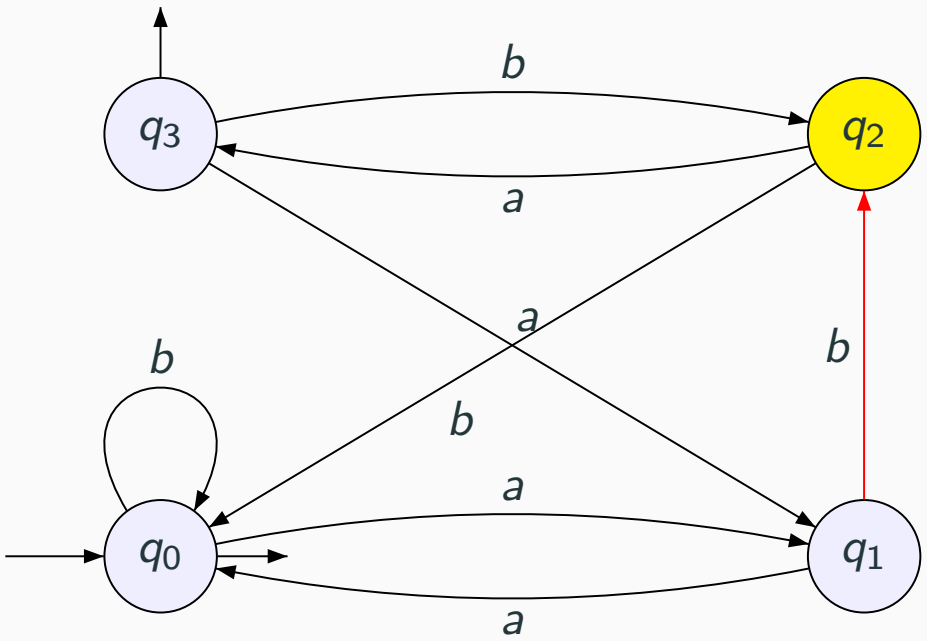
Représentation



δ	a	b
q_0	q_1	q_0
q_1	q_0	q_2
q_2	q_3	q_0
q_3	q_1	q_2

mot en entrée $ababa$

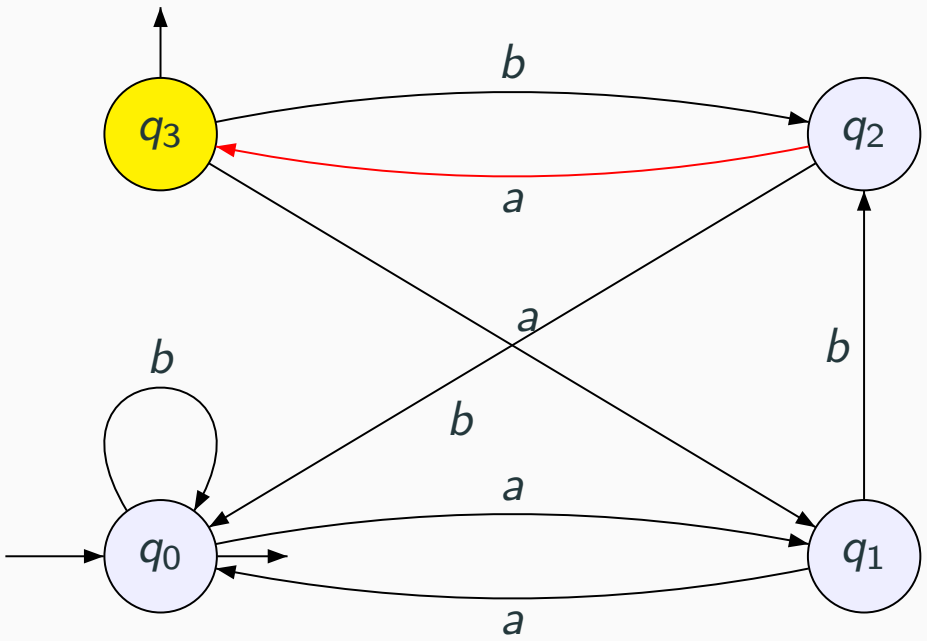
Représentation



δ	a	b
q_0	q_1	q_0
q_1	q_0	q_2
q_2	q_3	q_0
q_3	q_1	q_2

mot en entrée $ababa$

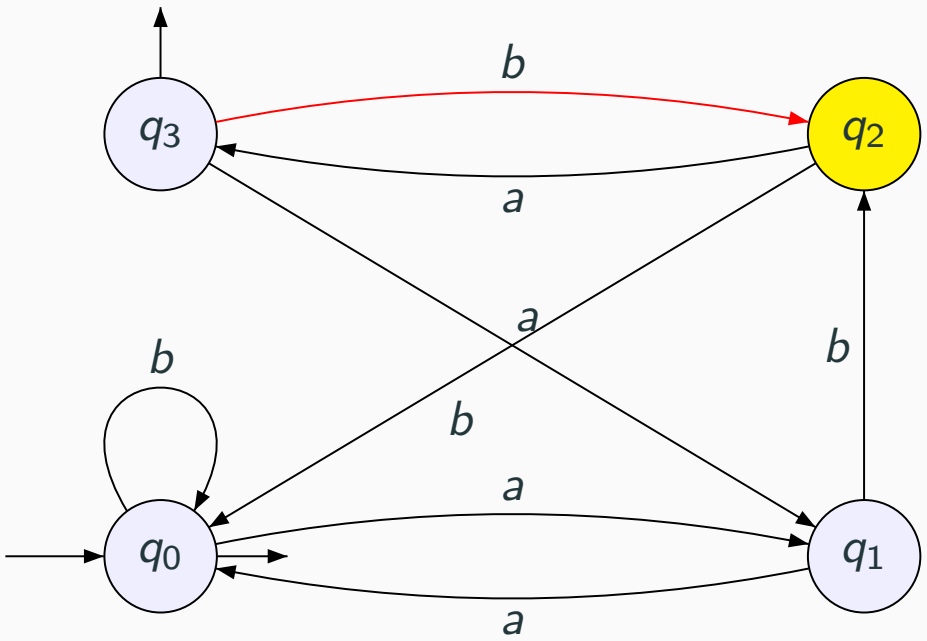
Représentation



δ	a	b
q_0	q_1	q_0
q_1	q_0	q_2
q_2	q_3	q_0
q_3	q_1	q_2

mot en entrée $ababa$

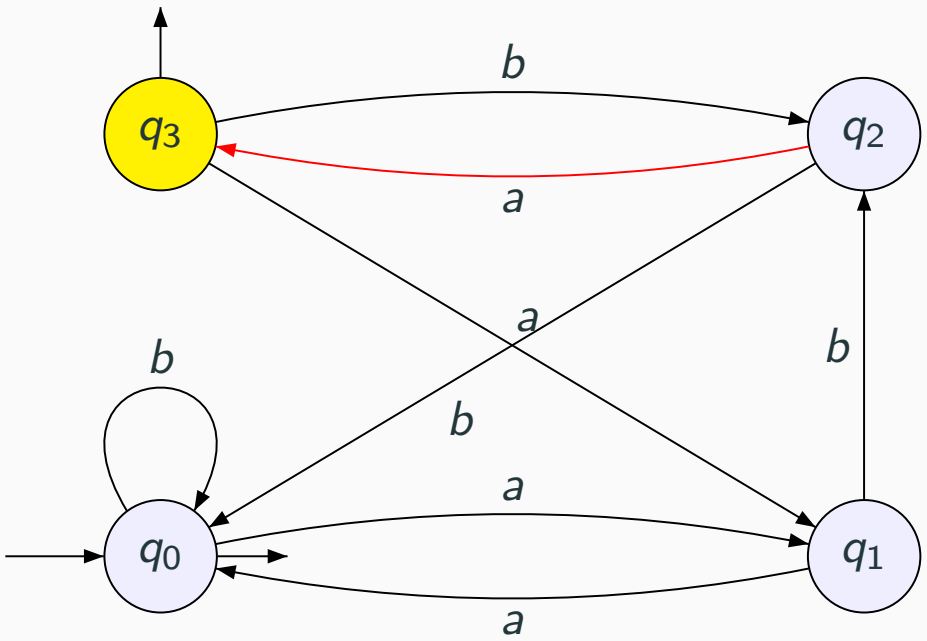
Représentation



δ	a	b
q_0	q_1	q_0
q_1	q_0	q_2
q_2	q_3	q_0
q_3	q_1	q_2

mot en entrée $ababa$

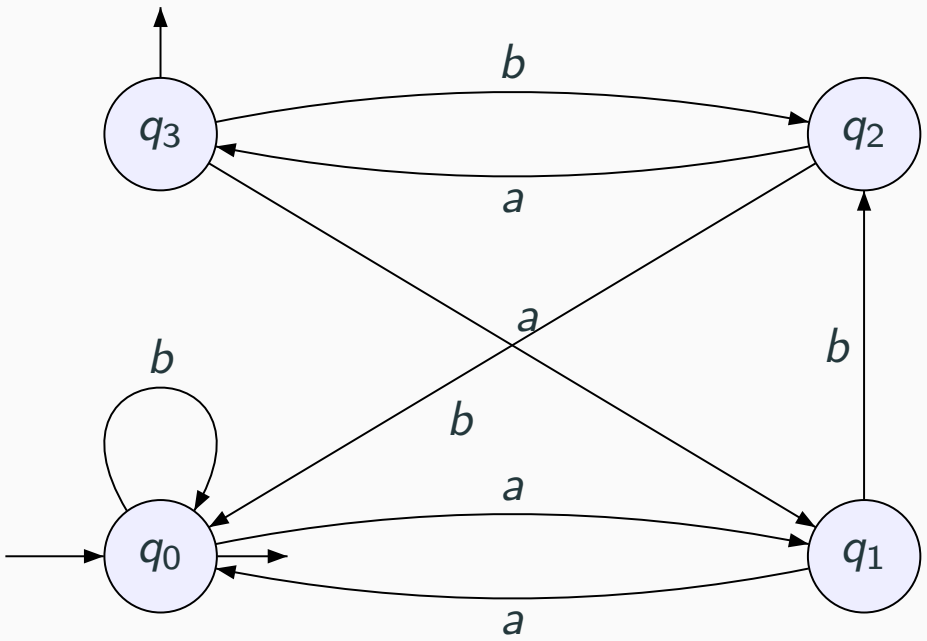
Représentation



δ	a	b
q_0	q_1	q_0
q_1	q_0	q_2
q_2	q_3	q_0
q_3	q_1	q_2

mot en entrée *ababa*

Représentation



δ	a	b
q_0	q_1	q_0
q_1	q_0	q_2
q_2	q_3	q_0
q_3	q_1	q_2

mot en entrée *ababa*

mot accepté, i.e $w = ababa \in L(A)$

Représentation algébrique

Système d'équations définissant un langage

Langage généré à partir d'un état

A chaque état de l'automate, on associe le langage des mots reconnus à partir de cet état

$$L(q) = L_q = \{w \in X^* \mid \delta^*(q, w) \in F\}$$

Remarque : Le langage reconnu par l'automate est L_{q_0}.

Equation associée

L_q est défini par l'équation (la réunion est notée avec le symbole $+$)

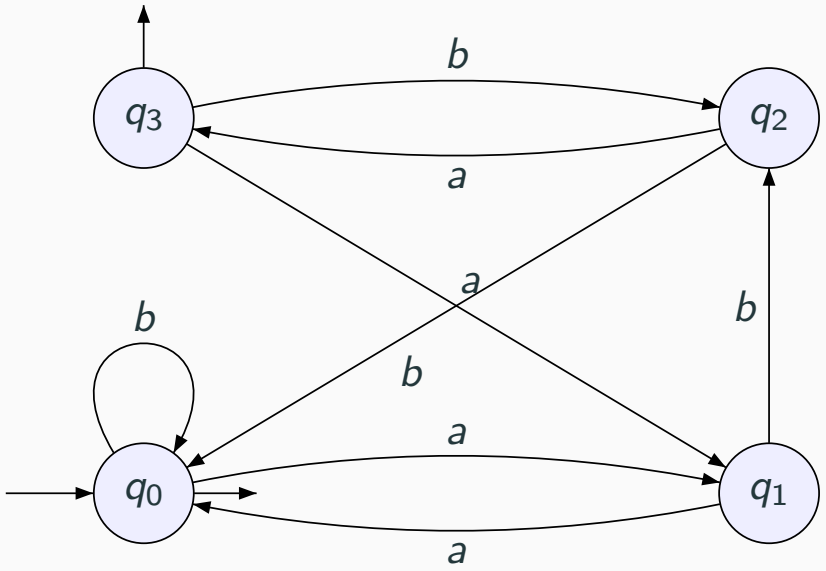
- Si $q \notin F$

$$L(q) = L_q = \sum_{x \in X} x.L(\delta(q, x))$$

- Si $q \in F$

$$L(q) = L_q = \sum_{x \in X} x.L(\delta(q, x)) + \{\varepsilon\} = \sum_{x \in X} x.L(\delta(q, x)) + \varepsilon$$

Exemple



$$\begin{cases} L(q_0) &= b.L(q_0) + a.L(q_1) + \varepsilon \\ L(q_1) &= a.L(q_0) + b.L(q_2) \\ L(q_2) &= b.L(q_0) + a.L(q_3) \\ L(q_3) &= b.L(q_2) + a.L(q_1) + \varepsilon \end{cases}$$

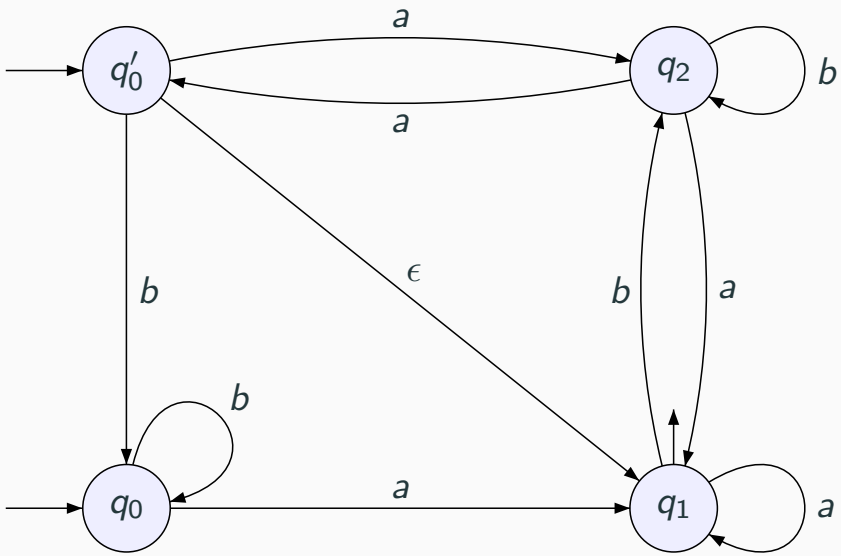
AFND

Automate fini non déterministe

3 types de non déterminisme

- Il peut exister plusieurs états initiaux.
- La fonction de transition est multivaluée ; pour un état q et un symbole s, il y a plusieurs transitions possibles, i.e $\delta(q, s) = \{q_1, \ldots, q_k\}$.
- Il peut exister des transitions ϵ, qui ménent d'un état à un autre sans qu'aucun symbole ne soit consommé.

Exemple



Fonctionnement d'un AFND

- *Configuration* : couple (état, chaîne à analyser) ; décrit l'état instantané de l'automate.
- *Transition* : On peut passer d'une configuration $c = (q, w)$ à $c' = (q', w')$ si :
  - $w = a.w'$ et $q' \in \delta(q, a)$
  - $w = w'$ et $q' \in \delta(q, \epsilon)$

  On note $c \vdash c'$
- Un mot w est accepté s'il existe au moins une séquence du type

$$c_0 = (q_0, w) \vdash c_1 \vdash c_2 \vdash \ldots \vdash c_n = (q_f, \epsilon)$$

avec q_0 initial et q_f final.

Equivalence AFD-AFND

AFD vs AFND

- La classe des langages reconnus par les AFD est la classe des langages rationnelles.
- Il est évident que la classe des langages reconnus par les AFND est plus grandes ($AFD \subset AFND$).
- En fait, c'est la même !

Théorème

Pour tout *AFND* A, il existe un *AFD* B tel que $L(A) = L(B)$.

On va démontrer ce théorème en donnant le moyen de déterminiser un AFND.

Déterminisation

ϵ-fermeture

Soit $A = (Q, X, \delta, Q_0, F)$ un automate fini. Soit $P \subset Q$ un sous ensemble d'états. Son ϵ-fermeture consiste à lui rajouter tous les états que l'on peut atteindre par un nombre fini de ϵ-transition. Pour une définition formelle :

$$\begin{cases} P_0 & = & P \\ P_{i+1} & = & P_i \cup \{q \in Q \mid \exists q' \in P_i,\ q \in \delta(q', \epsilon) \end{cases}$$

Q étant fini, il existe un j tel que $P_{j+1} = P_j$. l'ϵ-fermeture de P est alors ce P_j. On la notera $\epsilon(P)$.

Déterminisation

Soit $A = (Q, X, \delta, Q_0, F)$ un AFND. On construit un *AFD* équivalent $B = (Q', X, \delta', q'_0, F')$ par :

- $Q' = \{\{q1, q_2, \ldots, q_n\} \mid q_i \in Q,\ 1 \leq n \leq |Q|\} = \mathcal{P}(Q)$ ensemble des parties de Q.
- $q_0 = \epsilon(Q_0)$: ϵ-fermeture de Q_0.
- $F' = \{\{q_1, q_2, \ldots, q_n\} \mid \exists j, q_j \in F\}$.
- La fonction de transition δ' est définie de $\mathcal{P}(Q) \times X$ dans $\mathcal{P}(Q)$ par :
  - Pour tout $P = \{q_1, \ldots, q_n\} \in Q'$, et pour tout $x \in X$, on pose :

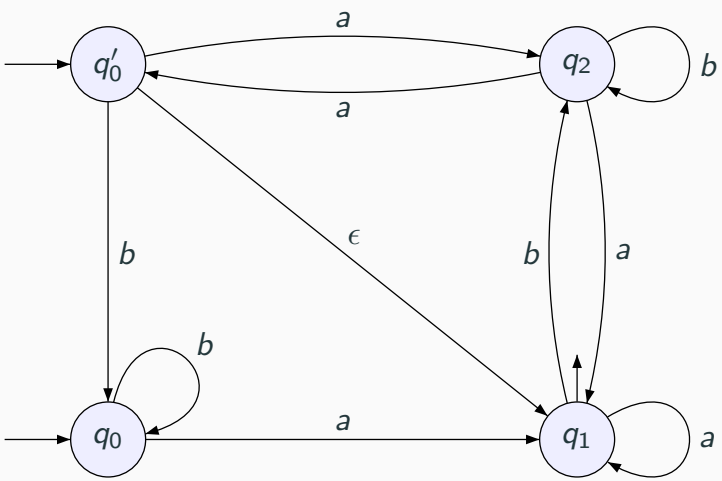
    $$P' = \{q'_1, \ldots, q'_r \mid \forall j, \exists i, q'_j \in \delta(q_i, x)\}$$

    tous les états qu'on atteint en partant des états de P et en consommant le symbole x. On en prend l'ϵ-fermeture.
  - On a alors : $\delta'(\{q_1, \ldots, q_n\}, x) = \epsilon(P')$

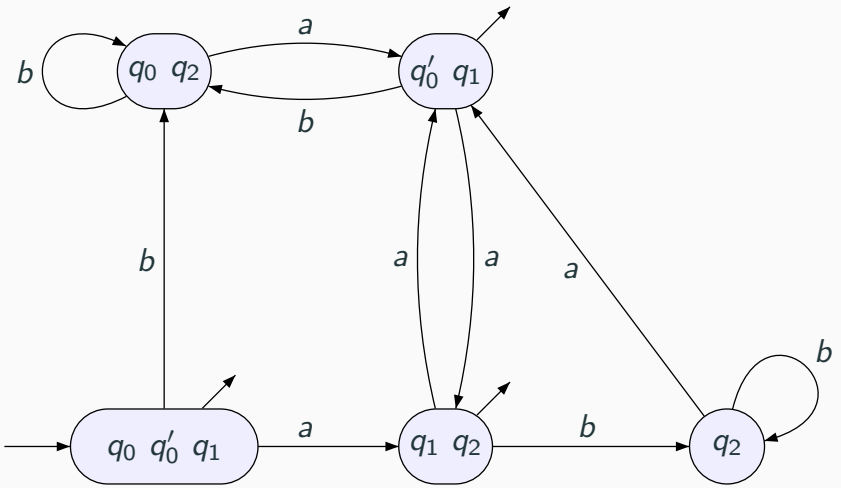
Intuitivement, l'automate ainsi construit simule tous les chemins possibles dans A lors de l'examen d'un mot.

Exemples

AFND



AFD équivalent



Problème : La déterminisation d'un automate conduit souvent à augmenter considérablement le nombre d'états.

Heureusement, on peut toujours minimiser un automate donné, du point de vue de son nombre d'états.

Déterminisation avec les équations

Équations de l'AFND précédent $\begin{cases} L_0 &= b.L_0 + a.L_1 \\ L_{0'} &= a.L_2 + b.L_0 + L_1 \\ L_1 &= a.L_1 + b.L_2 + \varepsilon \\ L_2 &= b.L_2 + a.L_1 + a.L_{0'} \end{cases}$

Le langage reconnu par cet automate est $L_0 + L_0^{'}$

$$L_0 + L_0^{'} = bL_0 + aL_1 + aL_2 + L_1 \quad (1)$$
$$= b(L_0 + L_2) + a(L_1 + L_2) + \varepsilon \quad (2)$$

$$L_0 + L_2 = bL_0 + aL_1 + bL_2 + aL_1 + aL_{0'} \quad (3)$$
$$= b(L_0 + L_2) + a(L_1 + L_{0'}) \quad (4)$$

$$L_1 + L_2 = aL_1 + bL_2 + \varepsilon + bL_2 + aL_1 + aL_{0'} \quad (5)$$
$$= a(L_1 + L_{0'}) + bL_2 + \varepsilon \quad (6)$$

$$L_1 + L_{0'} = aL_2 + bL_0 + aL_1 + bL_2 + \varepsilon \quad (7)$$
$$= a(L_2 + L_1) + b(L_0 + L_2) + \varepsilon \quad (8)$$

$$L_2 = bL_2 + a(L_1 + L_{0'}) \quad (9)$$

On a obtenu le système d'équations

$$\begin{cases} L_0 + L_0^{'} &= b(L_0 + L_2) + a(L_1 + L_2) + \varepsilon \\ L_1 + L_2 &= a(L_1 + L_{0'}) + bL_2 + \varepsilon \\ L_0 + L_2 &= b(L_0 + L_2) + a(L_1 + L_{0'}) \\ L_1 + L_{0'} &= a(L_1 + L_2) + b(L_0 + L_2) + \varepsilon \\ L_2 &= bL_2 + a(L_1 + L_{0'}) \end{cases}$$

que l'on peut directement traduire en AFD.

Minimisation

Minimisation d'un automate

On part d'un AFD, en supprimant ses états inaccesibles. On cherche un automate équivalent (même langage) ayant le moins d'états possibles.

L'idée de la méthode est de regrouper (automate quotient) les différents états par classe d'équivalence (Nérode).

Deux états q et p sont équivalents (inséparables) si les mots reconnus en partant du premier sont exactement les mêmes que ceux en partant du deuxième.

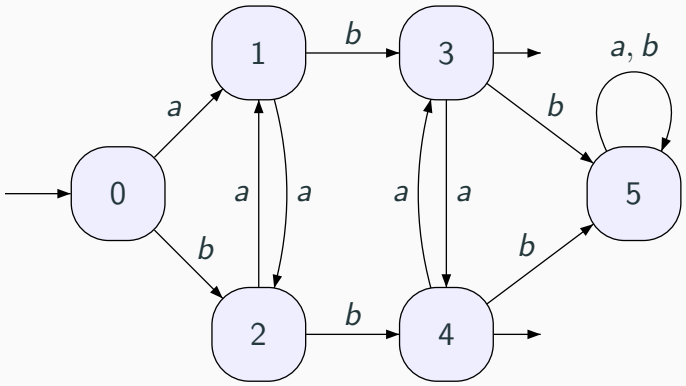
$$\forall u, \delta^*(p, u) \in F \Leftrightarrow \delta^*(q, u) \in F$$

Dans le cas contraire, ils sont dits séparables (on peut trouver un mot qui les sépare), i.e

$$\delta^*(p, u) \in F \wedge \delta^*(q, u) \notin F \text{ ou } \delta^*(p, u) \notin F \wedge \delta^*(q, u) \in F$$

L'algorithme calcule lettre par lettre les mots séparant les états.
L'automate obtenu est minimal. Il est unique (à un renommage des états près).

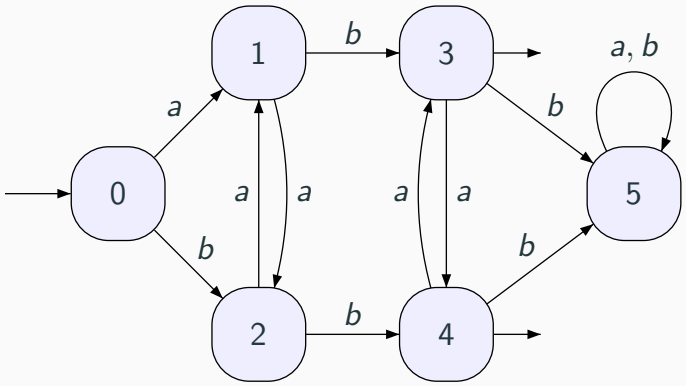
Algorithme de minimisation



- A chaque étape, on essaye de séparer les classes (en cours) déjà calculées.
- On s'arrête quand il n'y a plus de séparation.

0	1	2	5	3	4

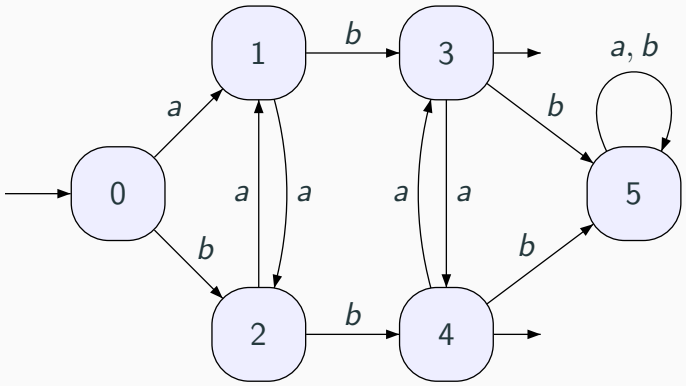
Algorithme de minimisation



	0	1	2	5	3	4	
ε	I	I	I	I	II	II	{0,1,2,5} {3,4}

- A chaque étape, on essaye de séparer les classes (en cours) déjà calculées.
- On s'arrête quand il n'y a plus de séparation.

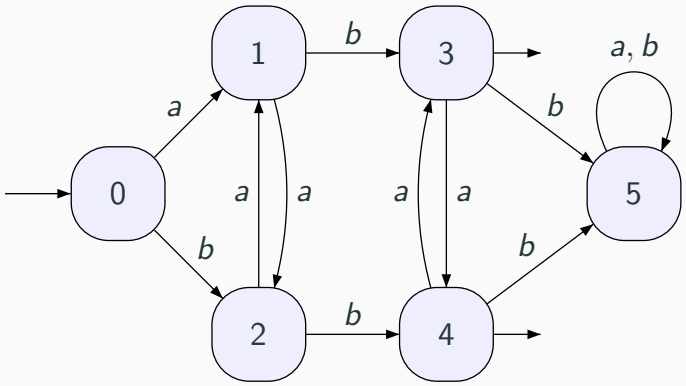
Algorithme de minimisation



	0	1	2	5	3	4	
ε	I	I	I	I	II	II	{0,1,2,5} {3,4}
a	I	I	I	I	II	II	

- A chaque étape, on essaye de séparer les classes (en cours) déjà calculées.
- On s'arrête quand il n'y a plus de séparation.

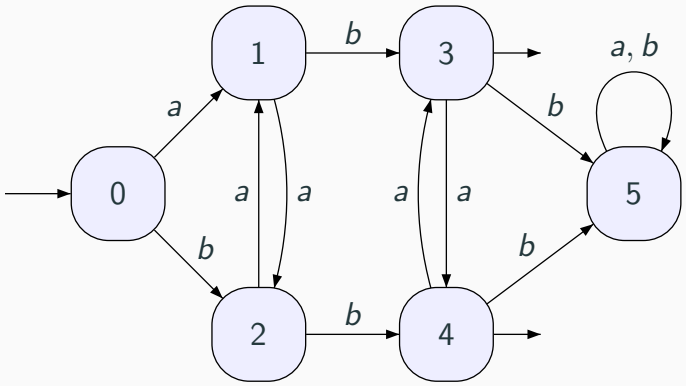
Algorithme de minimisation



- A chaque étape, on essaye de séparer les classes (en cours) déjà calculées.
- On s'arrête quand il n'y a plus de séparation.

	0	1	2	5	3	4	
ε	I	I	I	I	II	II	{0,1,2,5} {3,4}
a	I	I	I	I	II	II	
b	I	II	II	I	I	I	

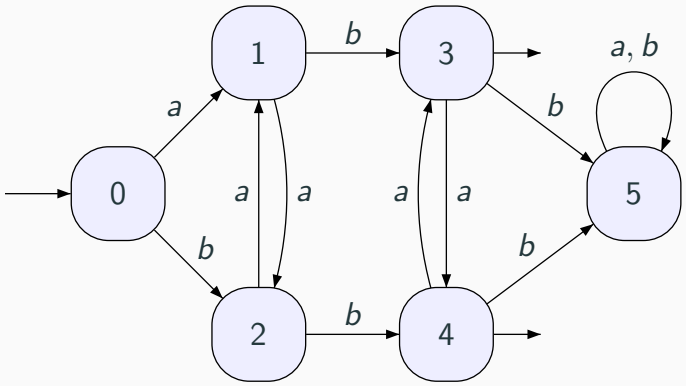
Algorithme de minimisation



- A chaque étape, on essaye de séparer les classes (en cours) déjà calculées.
- On s'arrête quand il n'y a plus de séparation.

	0	1	2	5	3	4	
ε	I	I	I	I	II	II	{0,1,2,5} {3,4}
a	I	I	I	I	II	II	
b	I	II	II	I	I	I	
Sép	I	III	III	I	II	II	{0,5}{1,2}{3,4}

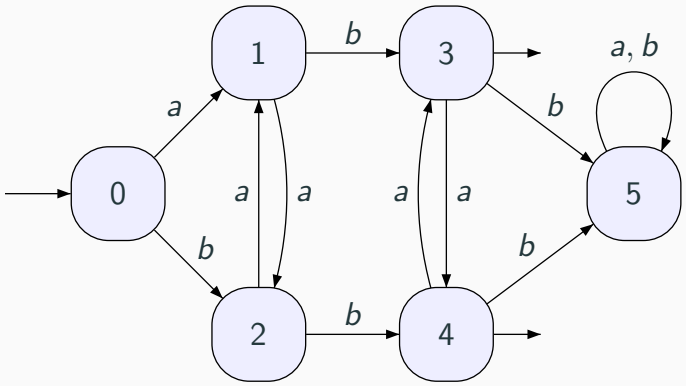
Algorithme de minimisation



- A chaque étape, on essaye de séparer les classes (en cours) déjà calculées.
- On s'arrête quand il n'y a plus de séparation.

	0	1	2	5	3	4	
ε	I	I	I	I	II	II	{0,1,2,5} {3,4}
a	I	I	I	I	II	II	
b	I	II	II	I	I	I	
Sép	I	III	III	I	II	II	{0,5}{1,2}{3,4}
a	III	III	III	I	II	II	

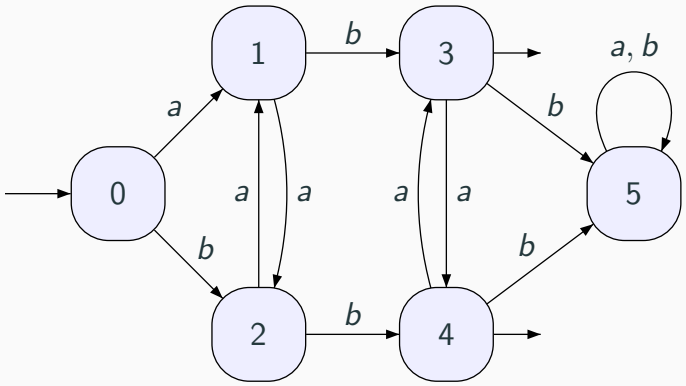
Algorithme de minimisation



- A chaque étape, on essaye de séparer les classes (en cours) déjà calculées.
- On s'arrête quand il n'y a plus de séparation.

	0	1	2	5	3	4	
ε	I	I	I	I	II	II	{0,1,2,5} {3,4}
a	I	I	I	I	II	II	
b	I	II	II	I	I	I	
Sép	I	III	III	I	II	II	{0,5}{1,2}{3,4}
a	III	III	III	I	II	II	
b	III	II	II	I	I	I	

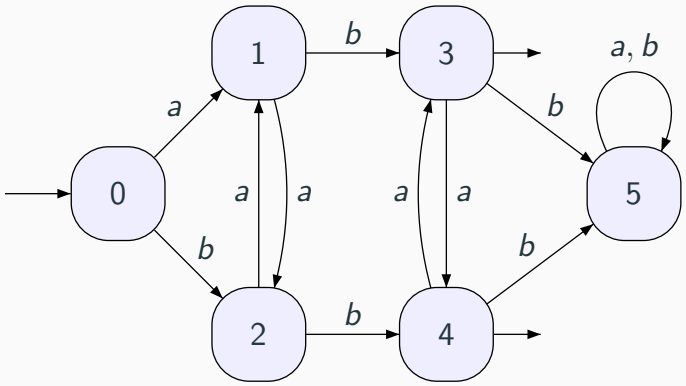
Algorithme de minimisation



- A chaque étape, on essaye de séparer les classes (en cours) déjà calculées.
- On s'arrête quand il n'y a plus de séparation.

	0	1	2	5	3	4	
ε	I	I	I	I	II	II	{0,1,2,5} {3,4}
a	I	I	I	I	II	II	
b	I	II	II	I	I	I	
Sép	I	III	III	I	II	II	{0,5}{1,2}{3,4}
a	III	III	III	I	II	II	
b	III	II	II	I	I	I	
Sép	I	III	III	IV	II	II	{0}{1,2}{3,4}{5}

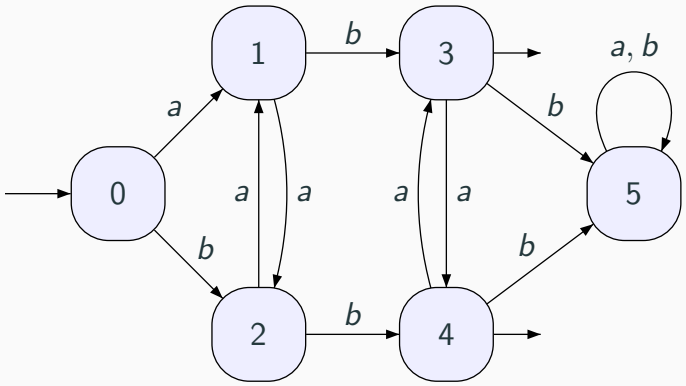
Algorithme de minimisation



- A chaque étape, on essaye de séparer les classes (en cours) déjà calculées.
- On s'arrête quand il n'y a plus de séparation.

	0	1	2	5	3	4	
ε	I	I	I	I	II	II	{0,1,2,5} {3,4}
a	I	I	I	I	II	II	
b	I	II	II	I	I	I	
Sép	I	III	III	I	II	II	{0,5}{1,2}{3,4}
a	III	III	III	I	II	II	
b	III	II	II	I	I	I	
Sép	I	III	III	IV	II	II	{0}{1,2}{3,4}{5}
a	III	III	III	IV	II	II	

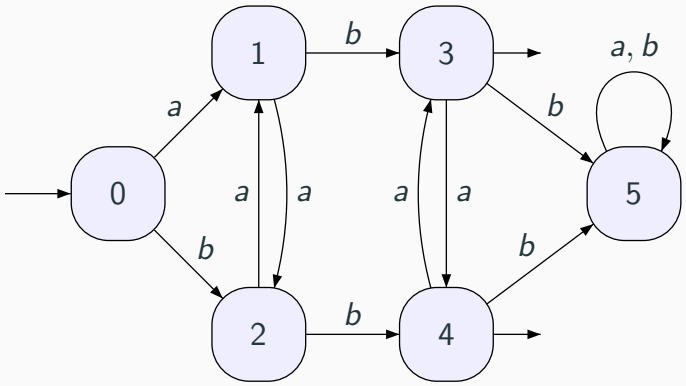
Algorithme de minimisation



- A chaque étape, on essaye de séparer les classes (en cours) déjà calculées.
- On s'arrête quand il n'y a plus de séparation.

	0	1	2	5	3	4	
ε	I	I	I	I	II	II	{0,1,2,5} {3,4}
a	I	I	I	I	II	II	
b	I	II	II	I	I	I	
Sép	I	III	III	I	II	II	{0,5}{1,2}{3,4}
a	III	III	III	I	II	II	
b	III	II	II	I	I	I	
Sép	I	III	III	IV	II	II	{0}{1,2}{3,4}{5}
a	III	III	III	IV	II	II	
b	III	II	II	IV	IV	IV	

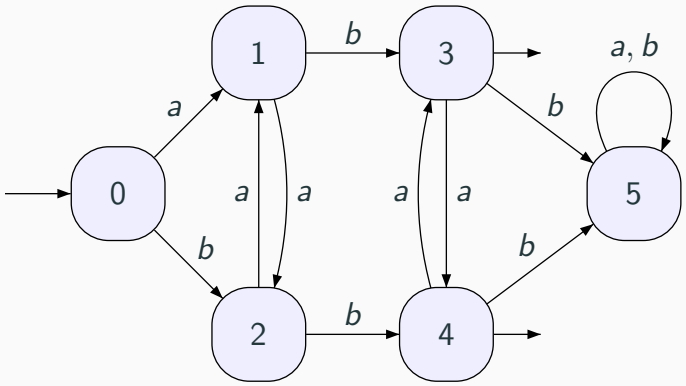
Algorithme de minimisation



- A chaque étape, on essaye de séparer les classes (en cours) déjà calculées.
- On s'arrête quand il n'y a plus de séparation.

	0	1	2	5	3	4	
ε	I	I	I	I	II	II	{0,1,2,5} {3,4}
a	I	I	I	I	II	II	
b	I	II	II	I	I	I	
Sép	I	III	III	I	II	II	{0,5}{1,2}{3,4}
a	III	III	III	I	II	II	
b	III	II	II	I	I	I	
Sép	I	III	III	IV	II	II	{0}{1,2}{3,4}{5}
a	III	III	III	IV	II	II	
b	III	II	II	IV	IV	IV	
Sép	I	III	III	IV	II	II	fin

Algorithme de minimisation

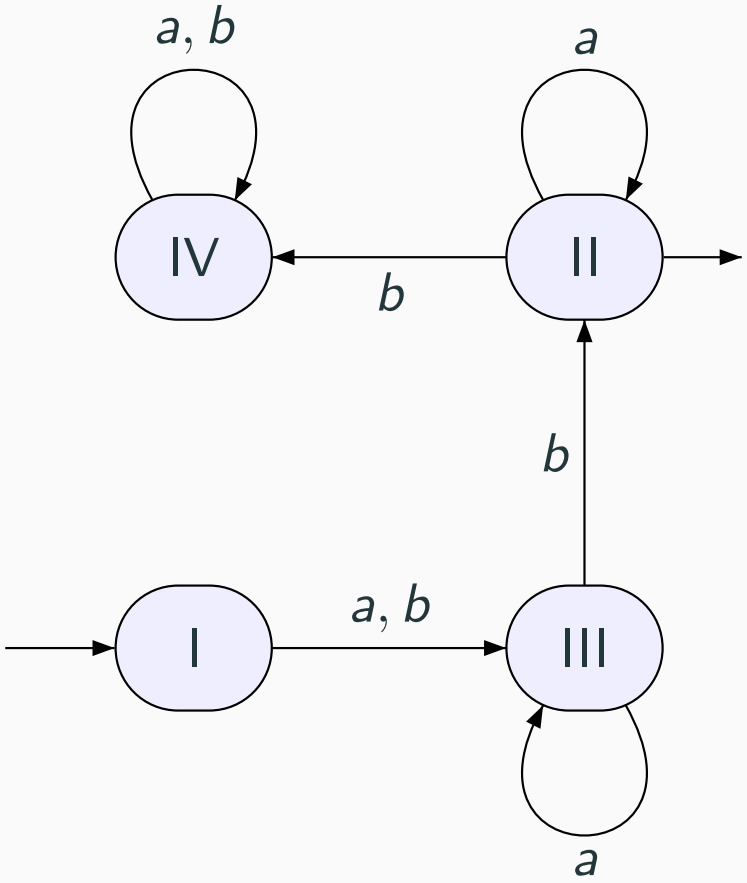


- A chaque étape, on essaye de séparer les classes (en cours) déjà calculées.
- On s'arrête quand il n'y a plus de séparation.

	0	1	2	5	3	4	
ε	I	I	I	I	II	II	{0,1,2,5} {3,4}
a	I	I	I	I	II	II	
b	I	II	II	I	I	I	
Sép	I	III	III	I	II	II	{0,5}{1,2}{3,4}
a	III	III	III	I	II	II	
b	III	II	II	I	I	I	
Sép	I	III	III	IV	II	II	{0}{1,2}{3,4}{5}
a	III	III	III	IV	II	II	
b	III	II	II	IV	IV	IV	
Sép	I	III	III	IV	II	II	fin

Les transitions entre les classes se lisent sur la dernière étape.

On obtient l'automate minimal avec 4 états.



Propriété de cloture

Théorème de Kleene

Théorème de Kleene

Soit X un alphabet fini. L'ensemble des langages rationnelles (RAT) sur X est la plus petite famille de langages contenant les parties de X, et fermée par union, produit et étoile.

Preuve : sens direct

$\emptyset \in RAT$. $\{\varepsilon\} \in RAT$. $\forall a \in X$, $\{a\} \in RAT$. pourquoi ?
Soient L_1 et L_2 deux langages rationnelles respectivement reconnus par les AFD $A_1 = (Q_1, X_1, \delta_1, q_{01}, F_1)$ et $A_2 = (Q_2, X_2, \delta_2, q_{02}, F_2)$ avec $Q_1 \cap Q_2 = \emptyset$.

On construit des AFND qui reconnaissent $L_1 \cup L_2, \overline{L_1}, L_1 \cap L_2, L_1^*$.

- **Union** $L_1 \cup L_2$: on fait la réunion disjointe des deux automates (deux états initiaux).
- **Complément** $\overline{L_1}$: même que A_1, en complémentant ses états terminaux. $A = (Q_1, X_1, \delta_1, q_{01}, Q_1 \setminus F_1)$ convient.

- **Intersection** $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$ ou en faisant le produit de A_1 et A_2 si $X_1 = X_2$ $A_1 \times A_2 = (Q_1 \times Q_2, X_1, \delta, (q_{01}, q_{02}), F_1 \times F_2)$ ou $\delta(q, q') = (\delta_1(q), \delta_2(q'))$
- **Produit** $L_1.L_2$: On construit l'AFND suivant :
  - État initiale : celui de A_1
  - État terminaux : ceux de A_2.
  - Transitions de A_1 et A_2, plus des ε-transitions des états terminaux de A_1 vers l'état initial de A_2.
- **L'étoile** L_1^* : on construit l'AFND suivant à partir de A_1 en ajoutant deux états i (état initial) et t (état terminal), et des ε-transitions :
  - de i vers l'état initial q_{01} de A_1.
  - de tout état final de A_2 vers t.
  - de i vers t et t vers i.

Pour la réciproque, cf la partie sur les expressions régulières.

Tous les langages sont-ils rationnels ?

Non ! Par exemple, $L = \{a^l b^l \mid l \geq 0\}$ n'est pas rationnel.

- Sinon, il existerait un AFD $A = (X, Q, q_0, F, \delta)$ tel que $L(A) = L$.
- Soit $n = |Q|$.
- Considérons tous les mots a^i, avec $1 \leq i \leq n + 1$.
- D'après le principe des tiroirs, il existe $m < p$ tel que $\delta^*(q_0, a^m) = \delta^*(q_0, a^p) = q \notin F$.
- Mais alors $\delta^*(q_0, a^m b^m) = \delta^*(q, b^m) \in F$.
- Et donc $\delta^*(q_0, a^p b^m) = \delta^*(q, b^m) \in F$, contradiction.

Généralisation : Pompage

Soit L un langage rationnel. Il existe une constante n tel que pour tout mot $w \in L$ tel que $|w| \geq n$, on ait $w = x.y.z$ avec :

- $y \neq \epsilon$.
- $|xy| \leq n$
- $x.y^k.z \in L \ \forall k \geq 0$.

Troisième partie III

Expressions régulières

Expression régulières

Expression régulières

Généralités

Exepression régulière

Définition

Soit X un alphabet fini disjoint de $Y = \{(,), +, ., *, \emptyset, \epsilon\}$. On définit inductivement les expressions régulières sur X par :

1. Les lettres de X, $\emptyset$ et ϵ est une *e.r.*
2. si x et y sont des e.r, alors $(x + y)$, $x.y$ et x^* sont des e.r.

Ce qui précéde est une construction syntaxique (algébrique), que l'on interprète en langage rationnel comme suivant :

Interprétation

expression régulière	langage rationnel
$\emptyset$	$\emptyset$: ensemble vide
ϵ	ϵ : mot vide
$a \in X$	$\{a\}$
si r_1 est associé à	L_1
si r_2 est associé à	L_2
$r_1 + r_2$	$L_1 \cup L_2$
$r_1.r_2$	$L_1.L_2$
r_1^*	L_1^*

Remarques

- Le théorème de Kleene affirme que les expressions régulières représentent exactement les langages rationnels.
- On pourra, par abus de langage, identifier une expression régulière avec son langage, et écrire par exemple $r_1 \subset r_2$.

Propriétés algébriques

Avec l'intrerprétation précédente :

- $+$ est idempotent $r + r = r$, commutatif, associatif, d'élément neutre $\emptyset$.
- $.$ est associatif, d'élément neutre ϵ. $\emptyset$ est aborbant $\emptyset.r = \emptyset$.

Quelques propriétés

1. $r^* = \epsilon + r.r^*$
2. $(r_1^* r_2)^*.r_1^* = (r_1 + r_2)^* = r_1^*(r_2 r_1^*)^*$
3. $r_1 = r_1^*$ ssi $r_1 = r_1^2$ et $\epsilon \in r_1$.

Montrons par exemple : $(r_1^* r_2)^* . r_1^* = (r_1 + r_2)*$

- On a $(r_1^* r_2)^* . r_1^* \subset (r_1 + r_2)*$, puisque $(r_1^* r_2)^* . r_1^*$ sont construits à partir de mots de r_1 et r_2.
- Montrons que $(r_1 + r_2)^* \subset (r_1^* r_2)^* . r_1^*$.
  Posons $r_3 = (r_1^* r_2)^* . r_1^*$; Clairement, r_1 et r_2 sont dans r_3, donc $r_1 + r_2 \in r_3$, et donc $(r_1 + r_2)^* \subset r_3^*$.
  Montrons que $r_3^* = r_3$: On a $r_3 = (r_1^* r_2)^* r_1^* = r_1^* (r_2 r_1^*)^*$, donc
  On a $r_3^2 = (r_1^* r_2)^* r_1^* r_1^* (r_2 r_1^*)^* = (r_1^* r_2)^* r_1^* (r_2 r_1^*)^* =$
  $(r_1^* r_2)^* (r_1^* r_2)^* r_1^* = (r_1^* r_2)^* r_1^* = r_3$
  Donc $\forall i \geq 1,\ r_3^i = r_3$.

Expression régulières

er à partir d'un d'automate

Théorème d'Arden

Arden

Soient r_1 et r_2 deux e.r. Alors l'équation

$$x = r_1.x + r_2$$

admet la solution $r_1^*.r_2$. Si $\epsilon \notin r_1$, cette solution est unique.

Preuve :

- $r_1^*.r_2$ est solution, car

$$r_1.(r_1^*.r_2) + r_2 = (r_1.r_1^*)r_2 + r_2 = (r_1.r_1^* + \epsilon)r_2 = r_1^* r_2$$

- Unicité : si $\epsilon \notin r_1$, pour toute solution x_0, on a

$$x_0 = r_1 x_0 + r_2 = r_1(r_1 x_0 + r_2) + r_2 = r_1^2 x_0 + r_1 r_2 + r_2$$

Soit

$$\forall n \geq 1, \quad x_0 = r_1^n x_0 + r_1^{n-1} r_2 + \ldots + r_2$$

Soit w un mot de x_0, et $n = |w|$. On a donc $w \in r_1^{n+1} x_0 + r_1^n r_2 + \ldots + r_2$

Comme $\epsilon \notin r_1$, on a nécessairement

$$w \in r_1^n r_2 + \ldots + r_2 \subset r_1^* r_2$$

D'où $x_0 \subset r_1^* r_2$

On a trivialement $r_1^* r_2 \subset x_0$, puisque $\forall n,\ x_0 = r_1^n x_0 + r_1^{n-1} r_2 + \ldots + r_2$, de sorte que $r_1^{n-1} r_2 + \ldots + r_2 \subset x_0$.

Généralisation

Tout système de n équations à n inconnues du type

$$x_i = r_{i1} x_1 + r_{i2} x_2 + \ldots r_{in} x_n + r_{in+1}$$

avec $\forall 1 \leq i, j \leq n, \quad \epsilon \notin r_{ij}$

admet une *solution unique*.

Preuve : récurrence + **Gauss**.

Un exemple :

$$\begin{cases} x_0 &= ax_1 + \epsilon \\ x_1 &= bx_0 \\ x_2 &= ax_0 + \epsilon \end{cases}$$

Ce système est de la forme générale suivante :

$$\begin{cases} x_0 &= \emptyset x_0 + ax_1 + \emptyset x_2 + \epsilon \\ x_1 &= bx_0 + \emptyset x_1 + \emptyset x_2 + \emptyset \\ x_2 &= ax_0 + \emptyset x_1 + \emptyset x_2 + \epsilon \end{cases}$$

En portant 2 dans 1, il vient

$$x_0 = (ab)^* \epsilon = (ab)^*$$

et donc

$$\begin{aligned} x_1 &= b(ab)^* \\ x_2 &= a(ab)^* + \epsilon \end{aligned}$$

Langage reconnu par un AFD

Idée : Soit $A = (Q, X, \delta, q_0, F)$ un AFD.

- A chaque état $q_i \in Q$, nous associons l'expression régulière x_i qui représente

$$\{w \mid \delta^*(q_i, w) \in F\}$$

, i.e l'ensemble des mots qui, partant de q_i, conduisent à un état final.

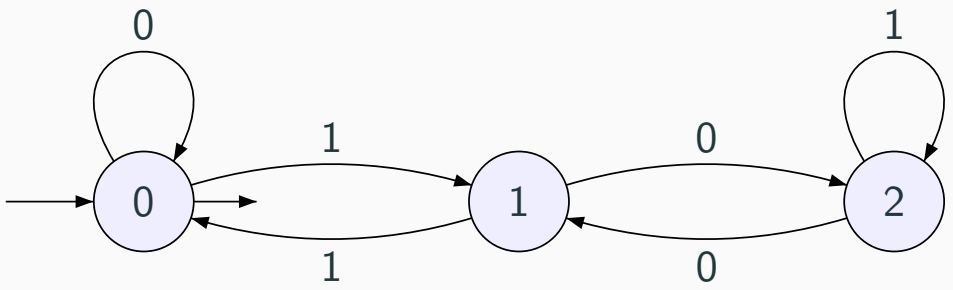
- Cet ensemble contient le mot vide ssi $q_i \in F$.
- Pour chaque état q_i, on a l'équation :

$$x_i = \sum_{a \in X} a.x_{\delta(q_i,a)} \ [+ \epsilon \text{ si } q_i \in F \ ]$$

Le théorème d'Arden donne en particulier

$$x_0 = \{w \in X^* \mid \delta^*(q_0, w) \in F\} = L(A)$$

Un exemple : On considère le langage formé des mots binaires qui sont des multiples de 3. On a donc l'automate suivant :



$$\begin{aligned} x_0 &= 0.x_0 + 1.x_1 + \epsilon \\ x_1 &= 1.x_0 + 0.x_2 \\ x_2 &= 0.x_1 + 1.x_2 \end{aligned}$$

Une solution

- $x_2 = 1^*0x_1$
- $x_1 = 01^*0x_1 + 1x_0 \Rightarrow x_1 = (01^*0)^*1x_0$
- $x_0 = 0x_0 + 1(01^*0)^*1x_0 + \epsilon = (0 + 1(01^*0)^*1)x_0 + \epsilon \Rightarrow x_0 = (0 + 1(01^*0)^*1)^*$

Une autre solution

- On élimine x_1 : $x_0 = 0x_0 + 1(1x_0 + 0x_2) + \epsilon = (0 + 11)x_0 + 10x_2 + \epsilon$ et $x_2 = (00 + 1)x_2 + 1x_0$
- $x_2 = (00 + 1)^*1x_0$
- $x_0 = (0 + 11 + 10(00 + 1)^*1)^*$

Expression régulières

automate à partir d'une e.r

Dérivation

Dérivation

Soit r une e.r sur X^* et $a \in X$. La dérivée (à gauche) de r par rapport à a est

$$D_a(r) = \{w \mid aw \in r\}$$

Exemple :

- $D_a(ab + b) = b$
- $D_a(b) = \emptyset$
- $D_a(a) = \epsilon$

Cette définition s'étend facilement aux mots par :

- $D_\epsilon(r) = r$
- $D_{xy}(r) = D_y(D_x(r))$

Calcul de dérivées

1. $D_a(r_1 + r_2) = D_a(r_1) + D_a(r_2)$
2. $D_a(r_1.r_2) = D_a(r_1).r_2 + \delta(r_1).D_a(r_2)$
   avec $\delta(r) = \epsilon$ si $\epsilon \in r$, $\emptyset$ sinon.
3. $D_a(r_1^*) = D_a(r_1).r_1^*$

1. Evident.
2. Le produit $r_1 r_2$ contient les mots de r_1 suivis de ceux de r_2, plus les mots de r_2 si $\epsilon \in r_2$, d'où le résultat.
3. $r_1^* = \epsilon + r_0.r_0^*$ avec $r_0 = r_1 \setminus \epsilon$. donc
   $D_a(r_1^*) = D_a(r_0)r_0^* + \emptyset = D_a(r_1)r_1^*$.

Automate reconnaissant une e.r

Soit r une e.r sur X. On sait construire un AF(N)D qui la reconnaît grâce au théorème de kleene ; On peut aussi utiliser la dérivation :

- $D_a(r)$ "simule" le reste des mots de r après consammation de a.
- Notons $r_0, r_1, r_2, \ldots, r_n$ les dérivées différentes de r avec $r = r_0 = D_\epsilon(r)$. Elles sont en nombre fini.
- Définissons l'automate fini

$$A = (\{q_0, q_1, q_2, \ldots, q_n\}, X, \delta, F)$$

où :

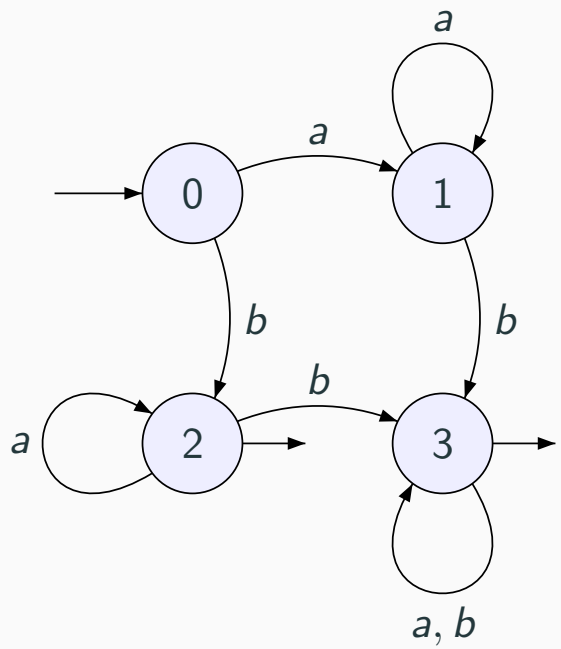
  - q_i est associé à r_i.
  - $q_i \in F$ si $\epsilon \in r_i$
  - $\delta(q_i, a) = q_j$ si $r_j = D_a(r_j)$.

- Cette automate reconnaît bien r par construction.

Un exemple

Soit $r_0 = a^* b(a+b)^* + ba^*$

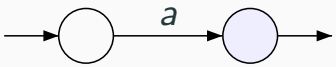
- $D_a(r_0) = a^* b(a+b)^* = r_1$
- $D_b(r_0) = (a+b)^* + a^* = r_2$
- $D_a(r_1) = a^* b(a+b)^* = r_1$
- $D_b(r_1) = (a+b)^* = r_3$
- $D_a(r_2) = (a+b)^* + a^* = r_2$
- $D_b(r_2) = (a+b)^* = r_3$
- $D_a(r_3) = (a+b)^* = r_3$
- $D_b(r_3) = (a+b)^* = r_3$



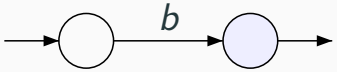
Retour sur Kleene

Un automate pour $(a+b)^*aba$. (construction automatique "idiote")

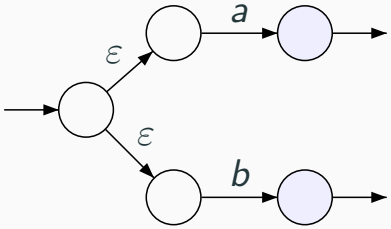
a



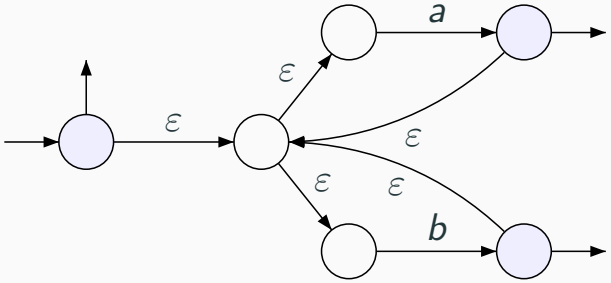
b



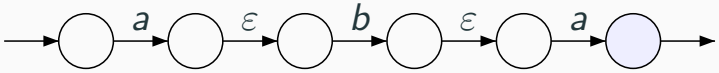
$a+b$



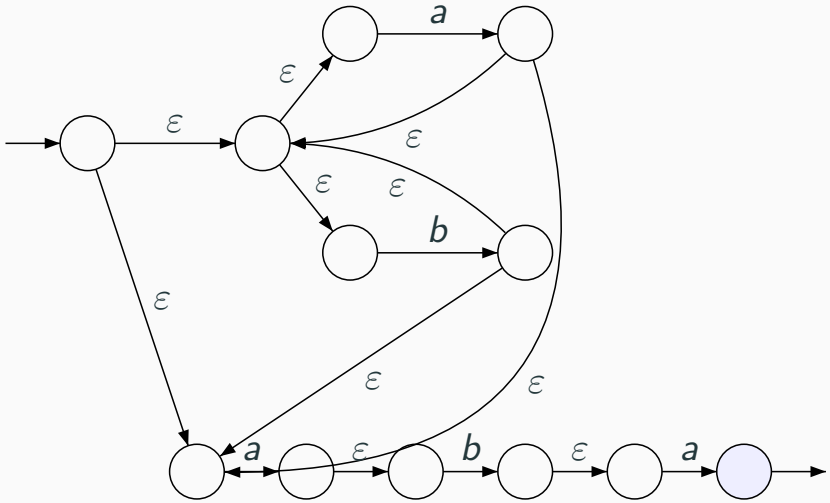
$(a+b)^*$



aba



$(a+b)^*aba$



Quatrième partie IV

Grammaires

Grammaires

Grammaires

Introduction

Principe

Souvenez-vous la "défintion" syntaxique des formes propositionnelles !

- Ensemble de règles de production.
- Règle de réécriture.
- mots du langage : mot obtenu en partant d'un symbole de départ (axiome), et en appliquant les règles un nombre fini de fois.

Grammaire

Grammaire

Formellement, une grammaire est un quadruplet $G = (V, X, P, S)$ où

- V est un ensemble fini appelé alphabet non terminal.
- X disjoint de V est un ensemble appelé alphabet terminal
- $S \in V$ est l'axiome
- $P \subset (V \cup X)^+ \times (V \cup X)^*$ ensemble des règles de production

Remarque :

- Les éléments de P (règles) sont notés $u \to v$ au lieu de (u, v).
- Les éléments de V sont représentés par des lettres majuscules, et ceux de X par des minuscules.

Exemples

$$G_1 \begin{cases} S & \to & aSa \\ S & \to & SbS \\ S & \to & \varepsilon \end{cases}$$

$$G_2 \begin{cases} S & \to & ab \\ S & \to & aASb \\ A & \to & bSb \\ AS & \to & b \\ A & \to & \varepsilon \end{cases}$$

$$G_3 \begin{cases} S & \to & I := E \\ I & \to & a|b|c \\ E & \to & E + E \\ E & \to & E * E \\ E & \to & (E) \\ E & \to & I \end{cases}$$

Dérivation

Dérivation

Soit $G = (V, X, P, S)$ une grammaire. Étant donnés deux mots x et y de $(V \cup X)^*$, on dit que x se dérive en y pour G, qu'on note $x \Rightarrow y$ ssi

- $x = z_1 u z_2$
- $y = z_1 v z_2$
- $u \to v \in P$

Le langage formel engendré par G $(L(G))$ est l'ensemble des mots $w \in X^*$ qu'on obtient par un nombre quelconque de dérivation depuis l'axiome.

Si on note $\overset{*}{\Rightarrow}$ la fermeture réflexive et ransitive de la relation $\Rightarrow$,

$$L(G) = \{w \in X^* \ , | \ S \overset{*}{\Rightarrow} w\}$$

Exemple

La grammaire $G : S \to aS|aSbS|\varepsilon$ (3 règles). On le numérote 1,2 et 3.

Le mot $aaba \in L(G)$

$$S$$

Mais aussi

$$S$$

Exemple

La grammaire $G : S \to aS|aSbS|\varepsilon$ (3 règles). On le numérote 1,2 et 3.

Le mot $aaba \in L(G)$

$$S \xrightarrow{2} aSbS$$

Mais aussi

$$S$$

Exemple

La grammaire $G : S \to aS|aSbS|\varepsilon$ (3 règles). On le numérote 1,2 et 3.

Le mot $aaba \in L(G)$

$$S \overset{2}{\to} aSbS \overset{1}{\to} aSbaS$$

Mais aussi

$$S$$

Exemple

La grammaire $G : S \to aS|aSbS|\varepsilon$ (3 règles). On le numérote 1,2 et 3.

Le mot $aaba \in L(G)$

$$S \xrightarrow{2} aSbS \xrightarrow{1} aSbaS \xrightarrow{3} aSba\varepsilon$$

Mais aussi

$$S$$

Exemple

La grammaire $G : S \to aS|aSbS|\varepsilon$ (3 règles). On le numérote 1,2 et 3.

Le mot $aaba \in L(G)$

$$S \xrightarrow{2} aSbS \xrightarrow{1} aSbaS \xrightarrow{3} aSba\varepsilon \xrightarrow{1} aaSba$$

Mais aussi

$$S$$

Exemple

La grammaire $G : S \to aS|aSbS|\varepsilon$ (3 règles). On le numérote 1,2 et 3.

Le mot $aaba \in L(G)$

$$S \xrightarrow{2} aSbS \xrightarrow{1} aSbaS \xrightarrow{3} aSba\varepsilon \xrightarrow{1} aaSba \xrightarrow{3} aa\varepsilon ba = aaba$$

Mais aussi

$$S$$

Exemple

La grammaire $G : S \to aS|aSbS|\varepsilon$ (3 règles). On le numérote 1,2 et 3.

Le mot $aaba \in L(G)$

$$S \xrightarrow{2} aSbS \xrightarrow{1} aSbaS \xrightarrow{3} aSba\varepsilon \xrightarrow{1} aaSba \xrightarrow{3} aa\varepsilon ba = aaba$$

Mais aussi

$$S \xrightarrow{2} aSbS$$

Exemple

La grammaire $G : S \to aS|aSbS|\varepsilon$ (3 règles). On le numérote 1,2 et 3.

Le mot $aaba \in L(G)$

$$S \xrightarrow{2} aSbS \xrightarrow{1} aSbaS \xrightarrow{3} aSba\varepsilon \xrightarrow{1} aaSba \xrightarrow{3} aa\varepsilon ba = aaba$$

Mais aussi

$$S \xrightarrow{2} aSbS \xrightarrow{1} aaSbS$$

Exemple

La grammaire $G : S \to aS|aSbS|\varepsilon$ (3 règles). On le numérote 1,2 et 3.

Le mot $aaba \in L(G)$

$$S \xrightarrow{2} aSbS \xrightarrow{1} aSbaS \xrightarrow{3} aSba\varepsilon \xrightarrow{1} aaSba \xrightarrow{3} aa\varepsilon ba = aaba$$

Mais aussi

$$S \xrightarrow{2} aSbS \xrightarrow{1} aaSbS \xrightarrow{3} aa\varepsilon bS$$

Exemple

La grammaire $G : S \to aS|aSbS|\varepsilon$ (3 règles). On le numérote 1,2 et 3.

Le mot $aaba \in L(G)$

$$S \xrightarrow{2} aSbS \xrightarrow{1} aSbaS \xrightarrow{3} aSba\varepsilon \xrightarrow{1} aaSba \xrightarrow{3} aa\varepsilon ba = aaba$$

Mais aussi

$$S \xrightarrow{2} aSbS \xrightarrow{1} aaSbS \xrightarrow{3} aa\varepsilon bS \xrightarrow{1} aabaS$$

Exemple

La grammaire $G : S \to aS|aSbS|\varepsilon$ (3 règles). On le numérote 1,2 et 3.

Le mot $aaba \in L(G)$

$$S \xrightarrow{2} aSbS \xrightarrow{1} aSbaS \xrightarrow{3} aSba\varepsilon \xrightarrow{1} aaSba \xrightarrow{3} aa\varepsilon ba = aaba$$

Mais aussi

$$S \xrightarrow{2} aSbS \xrightarrow{1} aaSbS \xrightarrow{3} aa\varepsilon bS \xrightarrow{1} aabaS \xrightarrow{3} aaba\varepsilon = aaba$$

Grammaires

Grammaire context-sensitif/context-free

Grammaire context-free

Définition

$G = (V, X, P, S)$ est algébrique, ou context-free si chaque règle est de la forme

$$A \to w$$

avec $A \in V$ et $w \in (V \cup X)^*$.

Un lanagage engendré par une grammaire algébrique est dit algébrique.

Exemple G définie par $S \to aSb|\varepsilon$ est algébrique. Elle engendre

Grammaire context-free

Définition

$G = (V, X, P, S)$ est algébrique, ou context-free si chaque règle est de la forme

$$A \to w$$

avec $A \in V$ et $w \in (V \cup X)^*$.

Un lanagage engendré par une grammaire algébrique est dit algébrique.

Exemple G définie par $S \to aSb|\varepsilon$ est algébrique. Elle engendre

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

Grammaires

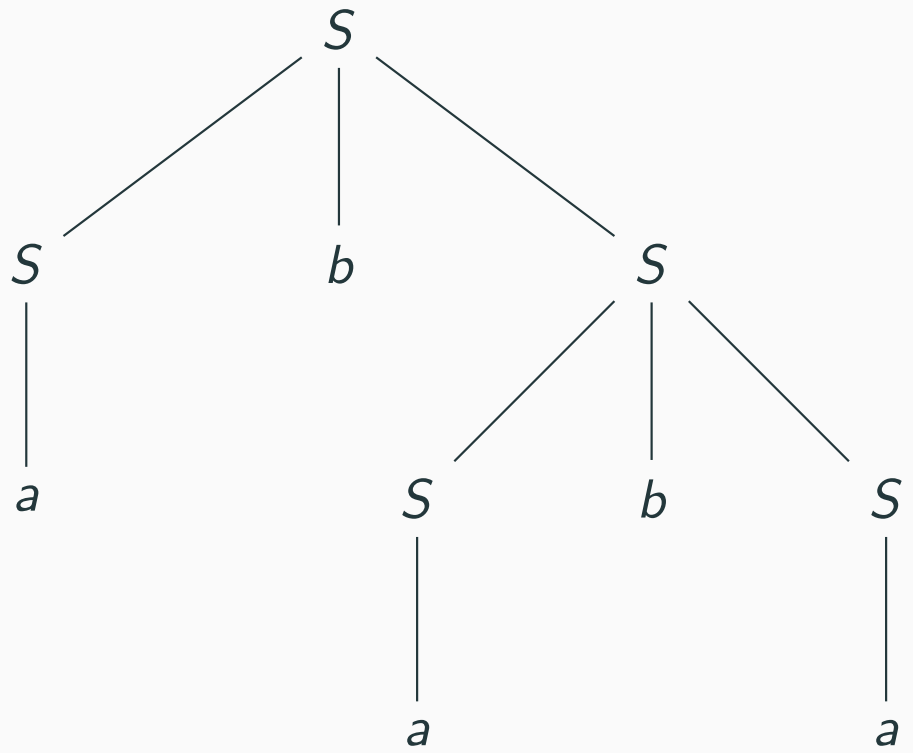
Arbre de dérivation, ambiguïté

Arbre de dérivation

Soit G la grammaire $S \rightarrow SbS|a$

On peut associer à la dérivation

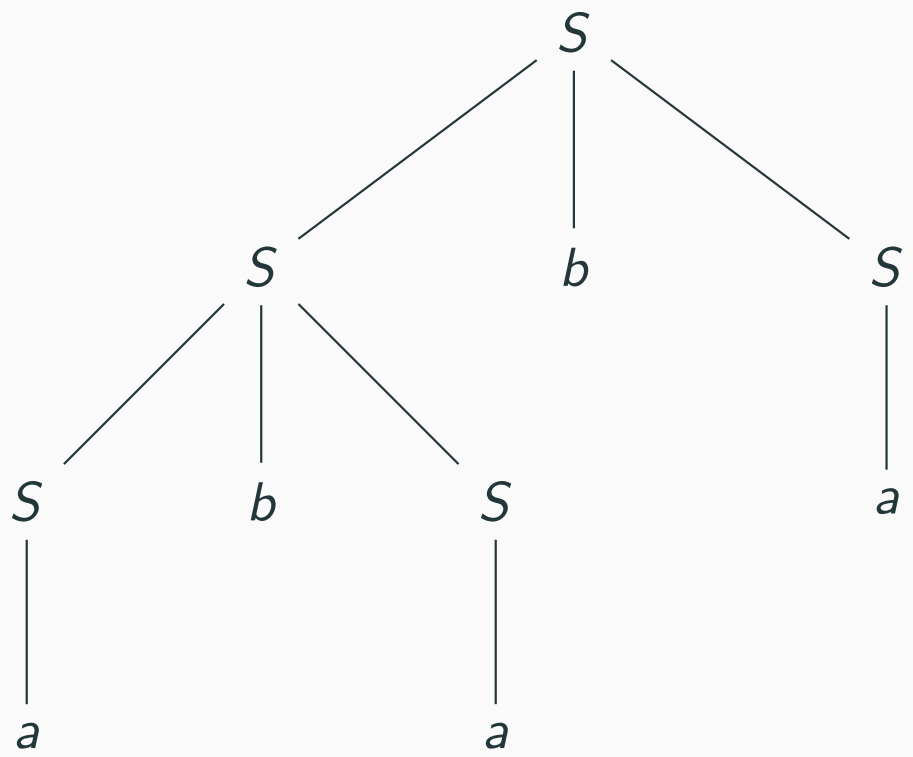
$$S \Rightarrow SbS \Rightarrow SbSbS \Rightarrow SbSba \Rightarrow Sbaba \Rightarrow ababa$$



son arbre de dérivation

Le même mot se dérive aussi

$$S \Rightarrow SbS \Rightarrow SbSbS \Rightarrow SbSba \Rightarrow Sbaba \Rightarrow ababa$$



son arbre de dérivation

Remarque : on peut lever l'ambiguité en consdérant le dérivation la plus à gauche. A chaque étape, on dérive le non-terminal le plus à gauche.

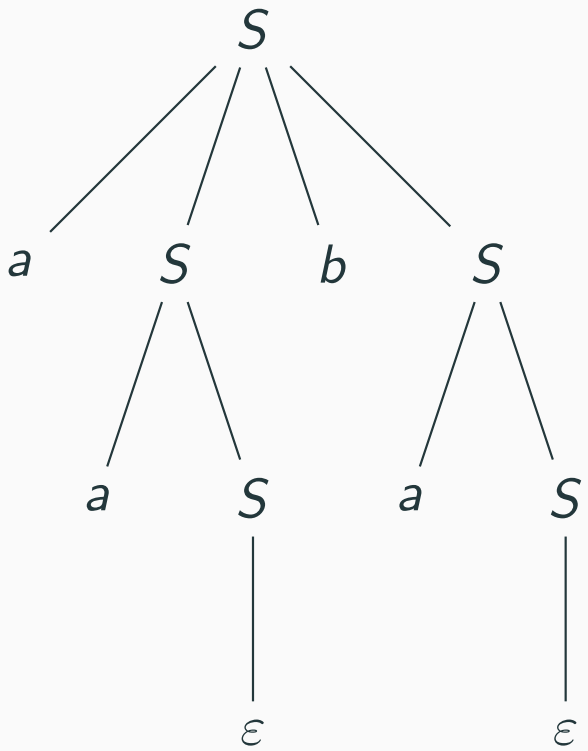
Grammaires ambigües

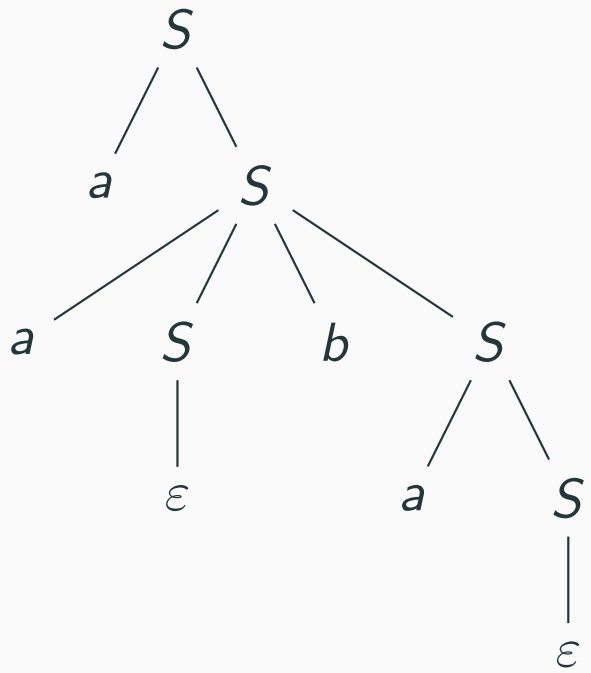
Ambiguité

Une grammaire G est dite ambiguë s'il existe un mot de $L(G)$ qui a au moins 2 dérivation la plus à gauche (donc 2 arbres de dérivation)

La grammaire $G : S \to aS|aSbS|\varepsilon$ (3 règles).

$aaba \in L(G)$





G est ambigüe !

Langage inhéremment ambigus

Il existe des langages algébriques tels que toute grammaire qui l'engendre est ambigüe. On dit que le langage est inhéremment ou intrinsèquement ambigu.

Théorème

$$L = \{a^n b^p c^q \mid n, p, q \in \mathbb{N}\, n = p \text{ ou } p = q\}$$

est intrinsèquement ambigu

Grammaire linéaire à droite

Grammaire linéaire à droite

les règles sont de la forme $A \to xB$ ou $A \to x$, avec $A \in V$ (non terminal) et $x \in X^*$ terminaux.

Théorème

Le langage engendré par une grammaire linéaire G est rationnel.

Preuve : Il suffit de transformer G pour obtenir uniquement des règles du type $A \to uB$, ou alors $A \to \varepsilon$, avec $u \in X$ symbole terminal.

- On transforme $A \to a_1 a_2 \dots a_n$ en $A \to a_1 A_1$, $A_1 \to a_2 A_2$, $\dots$, $A_{n-1} \to a_n A_n$, $A_n \to \varepsilon$.
- On transforme $A \to a_1 a_2 \dots a_n B$ en $A \to a_1 A_1$, $A_1 \to a_2 A_2$, $\dots$, $A_{n-1} \to a_n B$.

Ainsi, toutes les règles sont de la forme $A \to uB$ ou $A \to \varepsilon$, avec $u \in X$.

On construit facilement un automate qui reconnaît $L(G)$:

- on associe à chaque symbole non terminal un état
- si $A \to \varepsilon$, l'état associé à A est acceptant
- l'état initial est l'état associé à S
- $A \to uB$ donne la transition $\delta(A, u) = B$

Evidemment, on peut faire l'inverse, c'est à dire associer à un AFD A une grammaire G linéaire à droite telle que

$$L(A) = L(G)$$