

Langages, Compilation, Automates.

Partie 11: Révision: théorie des langages.

Florian Bridoux

Polytech Nice Sophia

2022-2023

- 1 Langages
- 2 Langages réguliers: AFD, AFI et expressions régulières
- 3 Langages hors-contexte: AP
- 4 Grammaires

- 1 Langages
- 2 Langages réguliers: AFD, AFI et expressions régulières
- 3 Langages hors-contexte: AP
- 4 Grammaires

Définition (Alphabet)

Un **alphabet**, souvent noté Σ , est un ensemble fini non vide d'éléments appelés symboles (ou lettres).

Définition (Mot)

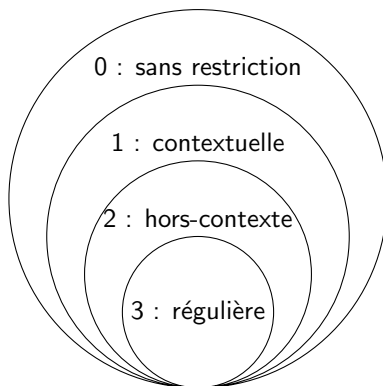
Un **mot** est une suite **finie** de symboles sur un alphabet donné.

- La **longueur** d'un mot u (notée $|u|$) est son nombre de symboles.
- Le **mot vide** (noté ϵ) est le seul mot de longueur nulle.
- Σ^ℓ est l'ensemble des mots de longueur ℓ sur l'alphabet Σ .
- Σ^* est l'ensemble des mots sur l'alphabet Σ .

Définition (Langage)

Un **langage** L sur l'alphabet Σ est un ensemble de mots sur cet alphabet: $L \subseteq \Sigma^*$.

Hiérarchie de Chomsky-Schützenberger



Théorème (Langages réguliers)

Le complémentaire d'un langage régulier est un langage régulier.
L'union ou l'intersection de deux langages réguliers est régulière.

Théorème (Langages hors-contexte)

- L'intersection de deux langages hors-contexte n'est pas nécessairement hors-contexte.
- Le complément d'un langage hors-contexte n'est pas nécessairement hors-contexte.
- L'union de deux langages hors-contexte est hors-contexte.

Théorème (Langages réguliers et langages hors-contexte)

L'intersection (ou l'union) d'un langage hors-contexte et d'un langage régulier est hors-contexte.

1 Langages

2 Langages réguliers: AFD, AFI et expressions régulières

3 Langages hors-contexte: AP

4 Grammaires

Équivalence entre AFD, AFI et expressions régulières

Théorème (Équivalence entre AFD, AFI et expression régulières)

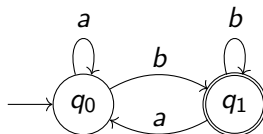
Les AFD, AFI (avec ou sans ϵ -transitions) et les expressions régulières reconnaissent exactement la même famille de langages (les langages réguliers).

Automates finis déterministes (AFD)

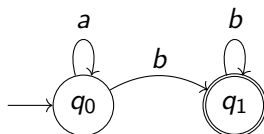
Définition (Automate fini déterministe (AFD))

Un **automate fini déterministe (AFD)** est un quintuplet $(\Sigma, Q, \delta, q_0, F)$ où:

- Σ est un alphabet des symboles d'entrée,
- Q est un ensemble **fini** d'états,
- δ est la fonction de transition: $Q \times \Sigma \rightarrow Q$,
- $q_0 \in Q$ est l'état initial,
- $F \subseteq Q$ est l'ensemble des états d'acceptation.



Automates finis déterministes



Sur le mot *abb*, l'automate va effectuer les transitions suivantes:

$$q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_1 \xrightarrow{b} q_1.$$

L'automate finit dans l'état acceptant q_1 et accepte donc le mot. Donc $L(A)$ ne contient pas le mot *abba*.

Sur le mot *abba*, l'automate va effectuer les transitions suivantes:

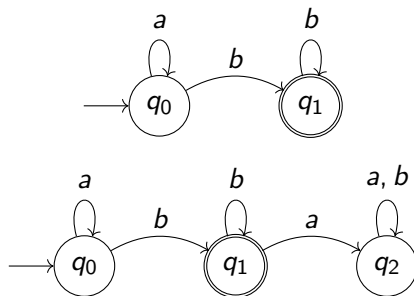
$$q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_1 \xrightarrow{b} q_1 \xrightarrow{a} \perp.$$

L'automate va donc rejeter le mot. Donc $L(A)$ ne contient pas le mot *abba*.

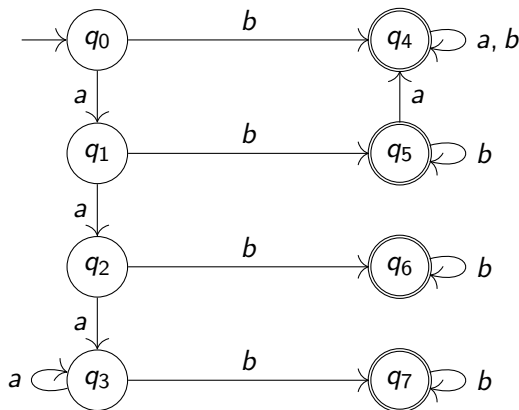
Compléter un automate

Définition (Automate complet)

Un automate $A = (\Sigma, Q, \delta, q_0, F)$ est **complet** si pour tout $q \in Q$ et tout $\ell \in \Sigma$, $\delta(q, \ell)$ est bien défini.

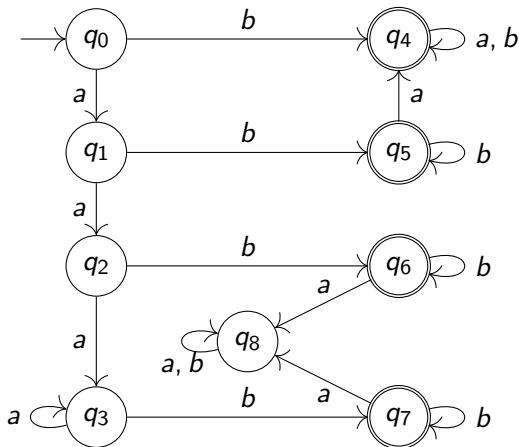


Algorithme de minimisation.



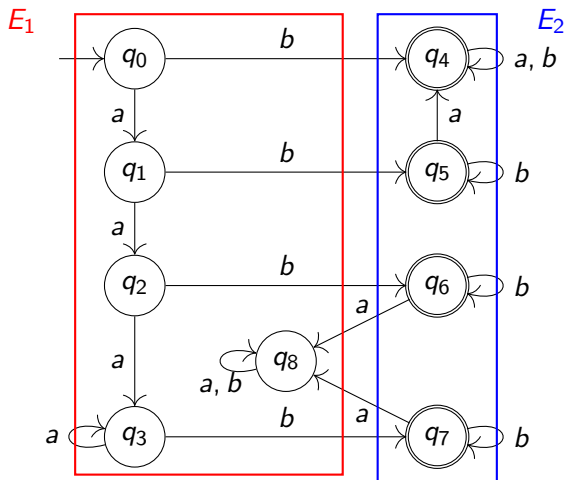
Idée: on va partitionner les états en ensemble d'états qui doivent être fusionnés.

Algorithme de minimisation.



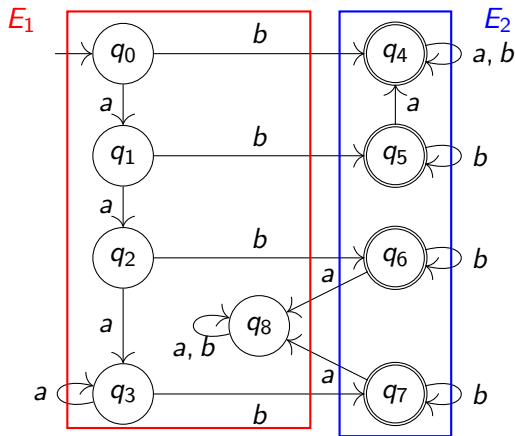
Étape préliminaire: On complète l'automate.

Algorithme de minimisation.



Première étapes: on partitionne en deux groupes: les acceptants et les non acceptants.

Algorithme de minimisation.



$E_1 :$

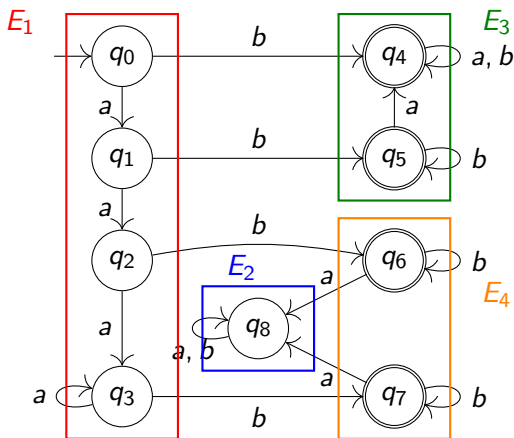
	a	b
q_0	E_1	E_2
q_1	E_1	E_2
q_2	E_1	E_2
q_3	E_1	E_2
q_8	E_1	E_1

$E_2 :$

	a	b
q_4	E_2	E_2
q_5	E_2	E_2
q_6	E_1	E_2
q_7	E_1	E_2

On voit $\{q_0, q_1, q_2, q_3\}$ et $\{q_8\}$ ne peuvent pas être fusionnés.
Même chose pour $\{q_4, q_5\}$ et $\{q_6, q_7\}$.

Algorithme de minimisation.



$E_1 :$

	a	b
q_0	E_1	E_3
q_1	E_1	E_3
q_2	E_1	E_4
q_3	E_1	E_4

$E_2 :$ tout seul

$E_3 :$

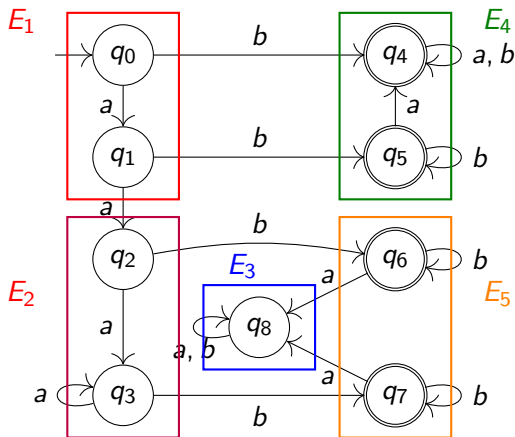
	a	b
q_4	E_3	E_3
q_5	E_3	E_3

$E_4 :$

	a	b
q_6	E_2	E_4
q_7	E_2	E_4

On voit $\{q_0, q_1\}$ et $\{q_2, q_3\}$ ne peuvent pas être fusionnés.

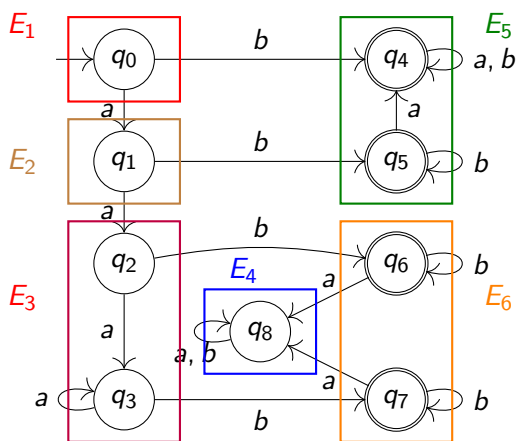
Algorithme de minimisation.



	a	b
$E_1 :$	q_0	E_1
	q_1	E_2
		E_4
	a	b
$E_2 :$	q_2	E_2
	q_3	E_2
		E_5
$E_3 :$	tout seul	
	a	b
$E_4 :$	q_4	E_4
	q_5	E_4
	a	b
$E_5 :$	q_6	E_3
	q_7	E_3
		E_5

On voit q_0 et q_1 ne peuvent pas être fusionnés.

Algorithme de minimisation.



E_1 : tout seul

E_2 : tout seul

	a	b
E_3 :	q_2	E_3
	q_3	E_3

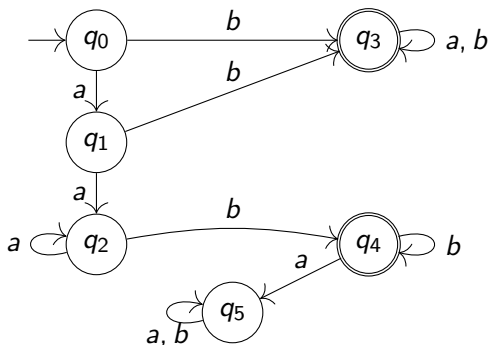
E_4 : tout seul

	a	b
E_5 :	q_4	E_5
	q_5	E_5

	a	b
E_6 :	q_6	E_4
	q_7	E_4

Pas de contradiction: on peut fusionner tous les états qui sont dans un même ensemble.

Algorithme de minimisation.



On obtient l'AFA complet ci-dessus. Si on le souhaite on peut alors retirer l'état poubelle.

Théorème (Unicité)

Le résultat de l'algorithme de minimisation sur un AFD A est le plus petit AFD C tel que $L(C) = L(A)$ (le plus petit étant unique à un renommage des états près).

Traduction: l'algorithme de minimisation nous donne le résultat optimal (qui est unique).

Corollaire

Soit deux AFD A et B avec $L(A) = L(B)$. L'algorithme de minimisation va donner le même AFD C comme résultat pour A et pour B (à un renommage des états près).

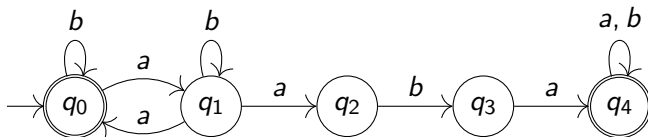
Cet algorithme nous donne donc un moyen de vérifier si deux AFD reconnaissent le même langage.

Automates finis indéterministes (AFI)

Définition (Automate fini indéterministe (AFI))

Un **automate fini indéterministe (AFI)** est un quintuplet $(\Sigma, Q, \delta, Q_0, F)$ où:

- Σ est un alphabet,
- Q est un ensemble **fini** d'états,
- δ est une fonction $Q \times \Sigma \rightarrow P(Q)$ (où $P(Q)$ est l'ensemble des sous-ensembles de Q),
- $Q_0 \subseteq Q$ est l'ensemble des états initiaux,
- $F \subseteq Q$ est l'ensemble des états d'acceptation.

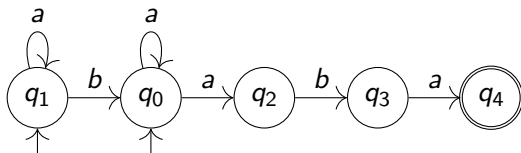


Automates finis indéterministes (AFI)

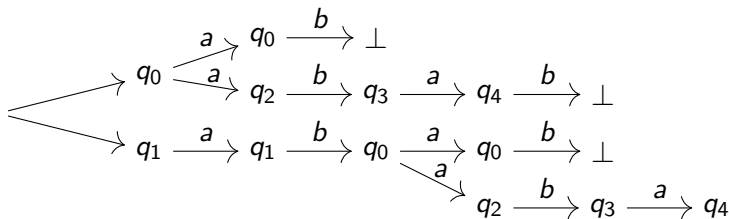
Similairement à un AFD, un AFI A est une machine qui calcule si un mot w appartient au langage $L(A)$ défini par cet automate. La différence est que certaines transitions sont non déterministes: on peut se retrouver dans différents états en ayant lu un même mot.

- Premier choix: dans quel état initial $q_0 \in Q_0$ commencer?
- À chaque lettre ℓ lu (dans un état q_i): vers quel état $q_j \in \delta(q_i, \ell)$ transitionner?
- Si une série de choix permet de finir dans un état acceptant alors $w \in L(A)$, sinon $w \notin L(A)$.

Automates finis indéterministes (AFI)



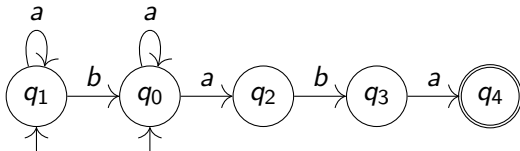
Exécution de la machine sur le mot *ababa*:



La machine peut finir dans l'état acceptant q_4 donc $ababa \in L(A)$.

Détermination d'AFI

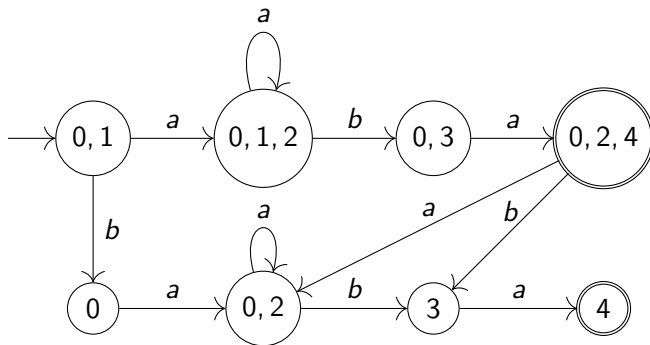
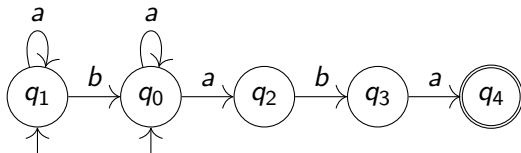
Nous allons déterminer cette AFI:



Idée de l'algorithme de détermination:

- Chaque état de l'AFD va représenter un sous ensemble d'états de l'AFI.
- Après avoir lu un mot w : l'AFI peut se trouver dans l'état q_i
 \Leftrightarrow l'AFD se trouve dans un état Q_j tel que $q_i \in Q_j$.
- En particulier, l'état initial de l'AFD est l'ensemble des états initiaux de l'AFI.
- Et un état Q_j de l'AFD est acceptant ssi $\exists q_i \in Q_j$ tel que q_i est acceptant.

Détermination d'AFI



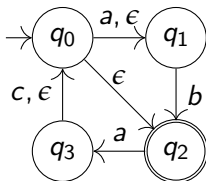
Théorème (Nombre d'états)

Soit A un AFI à $|Q|$ états, alors on peut construire B un AFD à au plus $2^{|Q|}$ états tel que $L(A) = L(B)$.

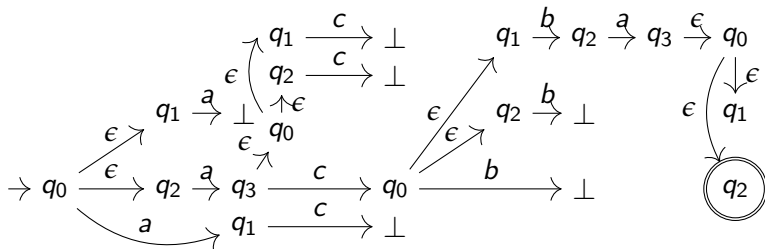
Conclusion: On peut toujours déterminer un AFI, mais le nombre d'états peut augmenter exponentiellement...

AFI avec ϵ -transitions

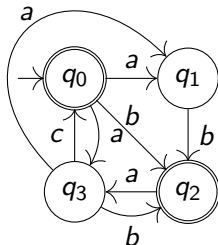
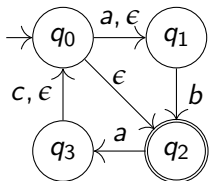
Les ϵ -transitions permettent de passer d'un état à l'autre sans "consommer" de lettres.



Arbre des transitions possibles sur le mot *acba*:



Algorithme pour retirer les ϵ -transitions



Tant que j'ai des ϵ -transitions:

- Je choisis q_i qui a des ϵ -transitions sortantes
- Si, avec des ϵ -transitions je peux atteindre q_j , alors:
 - J'ajoute toutes les (non ϵ) transitions sortantes de q_j sur q_i .
 - Si q_j est acceptant alors je rend q_i acceptant également.
- Je supprime les ϵ -transitions sortantes de q_i .

Définition (Expression régulière)

Les **expressions régulières** sur l'alphabet Σ sont définies récursivement :

- ① (\emptyset est une expression régulière qui décrit l'ensemble vide;)
- ② (ϵ est une expression régulière qui décrit l'ensemble réduit au mot vide $\{\epsilon\}$;)
- ③ pour toute lettre $a \in \Sigma$, a est une expression régulière qui décrit l'ensemble $\{a\}$.
- Si r et s sont deux expressions régulières, alors:
 - ① $r + s$ est une expression régulière qui décrit le langage $L(r) \cup L(s)$,
 - ② rs est une expression régulière qui décrit le langage $L(r)L(s) = \{uv \mid u \in L(r) \text{ et } v \in L(s)\}$;
 - ③ r^* est une expression régulière qui décrit $L(r)^* = \{w_1 \dots w_n \mid w_i \in L(r) \text{ et } n \in \mathbb{N}\}$.
 - ④ (r) est une expression régulière qui décrit $L(r)$.

Expression régulière

Ordre de priorité des opérateurs :

l'étoile $>$ la concaténation $>$ l'union.

Par exemple, $b + ab^*$ se lit $b + (a(b^*))$.

description en français	expression régulière
taille multiple de 2	$((a + b)(a + b))^*$
se termine par a	$(a + b)^*a$
avec le facteur bb	$(a + b)^*bb(a + b)^*$
sans le facteur bb	$(\epsilon + b)(a + ab)^*$

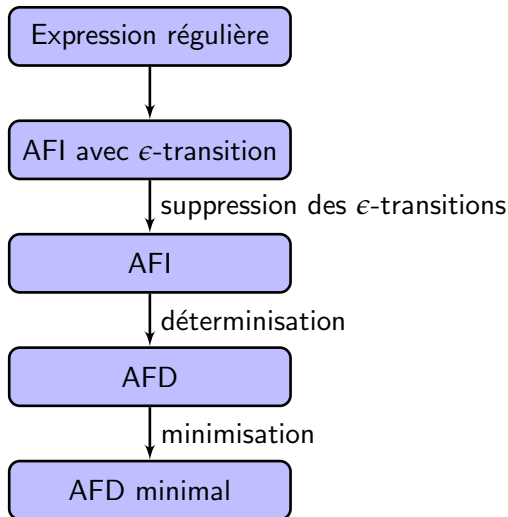
Remarque:

Une expression régulière décrit un unique langage mais un langage peut-être décrit par plusieurs expressions régulières.



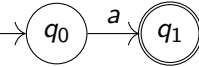
Exemple: $(a + b)(a + b) = aa + ab + ba + bb$.

Transformer une expression régulière en AFD minimal

Étapes de la transformation:

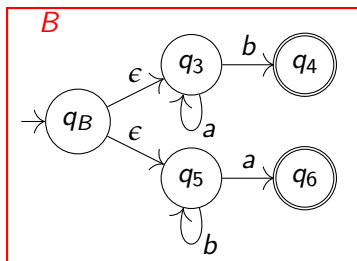
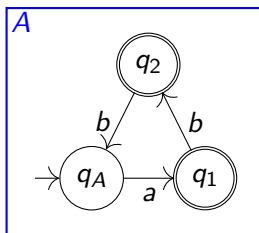


Transformer une expression régulière: brique de base

- Langage vide: 
- Langage $\{\epsilon\}$: 
- Langage $\{a\}$: 

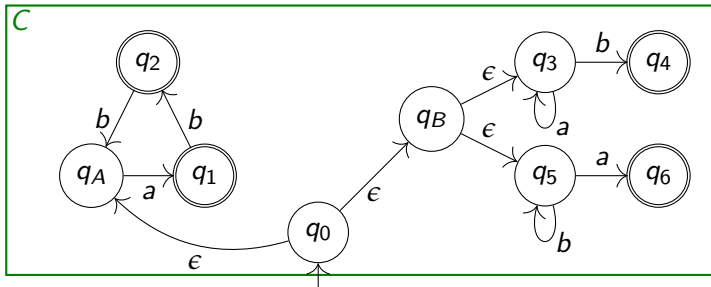
Transformer une expression régulière: union

Supposons que s et r sont deux expressions régulières équivalentes à A et B (avec q_A et q_B leurs deux états initiaux). Comment former C équivalent à $s + r$?



Transformer une expression régulière: union

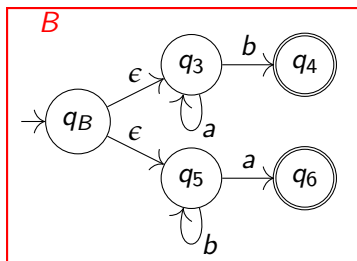
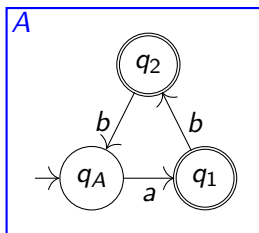
Supposons que s et r sont deux expressions régulières équivalentes à A et B (avec q_A et q_B leurs deux états initiaux). Comment former C équivalent à $s + r$?



- On ajoute un nouvel état initial q_0 .
- On ajoute deux transitions $q_0 \xrightarrow{\epsilon} q_A$ et $q_0 \xrightarrow{\epsilon} q_B$.
- Les états q_A et q_B ne sont plus initiaux.

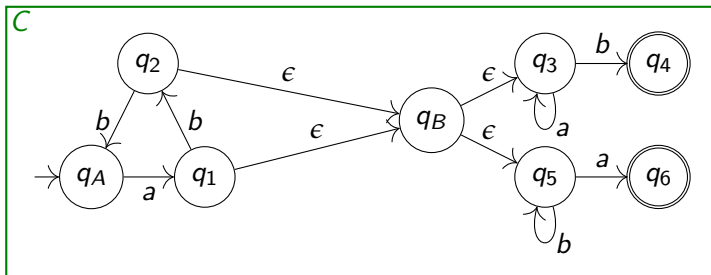
Transformer une expression régulière: concaténation

Supposons que s et r sont deux expressions régulières équivalentes à A et B (avec q_A et q_B leurs deux états initiaux). Comment former C équivalent à sr ?



Transformer une expression régulière: concaténation

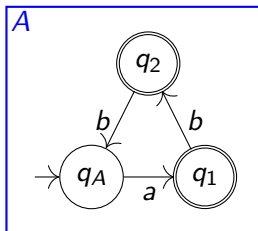
Supposons que s et r sont deux expressions régulières équivalentes à A et B (avec q_A et q_B leurs deux états initiaux). Comment former C équivalent à sr ?



- On ajoute des transitions $q_i \xrightarrow{\epsilon} q_A$ pour tout état acceptant q_i de A .
- q_B n'est plus initial.
- Les états acceptants de q_A ne sont plus acceptants.

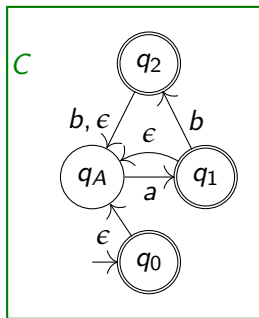
Transformer une expression régulière: étoile

Supposons que s est une expression régulière équivalente à A (avec q_A sont état initial). Comment former C équivalent à s^* ?



Transformer une expression régulière: étoile

Supposons que s est une expression régulière équivalente à A (avec q_A sont état initial). Comment former C équivalent à s^* ?



- On ajoute un nouvel état initial acceptant q_0 .
- On ajoute une transition $q_0 \xrightarrow{\epsilon} q_A$ et une transition $q_i \xrightarrow{\epsilon} q_A$ pour chaque état acceptant q_i .
- L'état q_i n'est plus initial.

Transformer un AFD en expression régulière

Dans l'autre sens, on peut transformer un AFD en expression régulière (on ne va pas le faire ici mais voir le CM3).

Table des matières

- 1 Langages
- 2 Langages réguliers: AFD, AFI et expressions régulières
- 3 Langages hors-contexte: AP
- 4 Grammaires

Automates à pile (AP)

Définition (Automate à pile (AP))

Un **automate à pile (AP)** est un septuplet $(\Sigma, \Gamma, Q, \delta, Z, q_0, F)$ où:

- Σ est un alphabet des symboles d'entrée,
- Γ est un alphabet des symboles de pile,
- Q est un ensemble **fini** d'états,
- δ est la fonction de transition:
 $Q \times (\Sigma \cup \{\epsilon\}) \times (\Gamma \cup \{\epsilon\}) \rightarrow P(Q \times \Gamma^*),$
- $Z \in \Gamma$ symbole initial de la pile (aussi appelé *symbole de fond de pile*)
- $q_0 \in Q$ est l'état initial,
- $F \subseteq Q$ est l'ensemble des états d'acceptation.

Rappel: $P(E)$ désigne l'ensemble des $2^{|E|}$ sous-ensembles de E .

Automates à pile (AP)

Un AP commence dans l'état initial avec une pile vide.

$$\delta : Q \times (\Sigma \cup \{\epsilon\}) \times (\Gamma \cup \{\epsilon\}) \rightarrow P(Q \times \Gamma^*)$$

L'AP commence dans l'état initial q_0 avec une pile contenant uniquement Z . À chaque étape, l'AP:

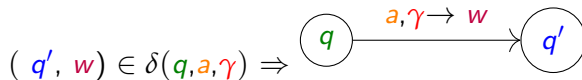
- lit l'état courant $q \in Q$.
- Choisit (indéterministe) de consommer le prochain symbole ($a \in \Sigma$) ou pas de symbole ($a = \epsilon$).
- Choisit (indéterministe) d'ignorer la pile $\gamma = \epsilon$ ou de dépiler le sommet de la pile $\gamma \in \Gamma$.
- Passe dans un état $q' \in Q$ et empile un mot $w \in \Gamma^*$ avec $(q', w) \in \delta(q, a, \gamma)$

L'AP accepte le mot si après avoir lu tout le mot il peut finir dans un état acceptant avec une pile vide.

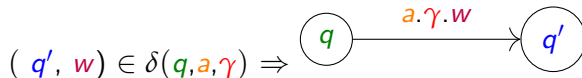
Remarque: on verra plus tard des variations de ce mode d'acceptation.

Représentation des transitions

Classiquement:



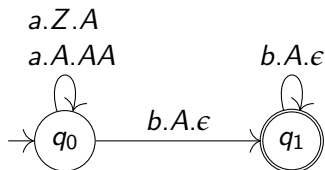
Dans ce cours:



Passage d'une **configuration** à une autre:

$$(q, CBA) \xrightarrow{a.C.ED} (q', EDBA)$$

Exemple: reconnaître le langage $\{a^n b^n \mid n \geq 1\}$



état	mot restant	pile
q_0	$aaabbb$	Z
q_0	$aabbb$	A
q_0	$abbb$	AA
q_0	bbb	AAA
q_1	bb	AA
q_1	b	A
q_1	ϵ	ϵ

On finit dans l'état q_1 (acceptant) avec une pile vide: mot accepté.

Theorem

Un langage L est hors-contexte si et seulement s'il existe un automate à pile A qui le reconnaît ($L(A) = L$).

- Si un langage est engendré par une grammaire hors-contexte alors il existe un automate à pile qui le reconnaît.
- Si un langage est reconnu par un automate à pile alors il est engendré par une grammaire hors-contexte.

Principe:

- 1 Empiler l'axiome S de la grammaire.
- 2 Si le sommet de la pile est un non-terminal N_i alors on le remplace par un la partie droite d'une règle de la forme $N_i \rightarrow \alpha$ de telle sorte que le premier symbole x de α se trouve en sommet de pile.
- 3 Si le sommet de la pile est un terminal x alors on le compare avec le prochain caractère de l'entrée. S'ils sont égaux alors on dépile sinon on "rejette".
- 4 Si la pile est vide et que l'entrée a été totalement lue alors on accepte, sinon on revient à l'étape 2.

Exemple

Grammaire G:

$$\begin{aligned} E &\rightarrow E + T \\ T &\rightarrow T * F \\ F &\rightarrow (E) \mid a \end{aligned}$$


$a.a.\epsilon$
 $+.+. \epsilon$
 $*.*.\epsilon$
 $(.(.\epsilon$
 $).). \epsilon$
 $\epsilon.E.E + T$
 $\epsilon.T.T * F$
 $\epsilon.F.(E)$
 $\epsilon.F.a$

Remarque:

Lorsqu'un non-terminal N_i doit être remplacé au sommet de la pile, il peut l'être par la partie droite d'une règle de la forme $N_i \longrightarrow \alpha$. Plusieurs règles de cette forme peuvent exister dans la grammaire. L'automate correspondant est généralement non déterministe.

Quelques résultats

- On ne le fera pas ici, mais on peut transformer notre automate à pile en une grammaire hors-contexte.
- On peut en déduire que chaque automate à pile est équivalent à un automate à pile à **un seul état**.
- En revanche, il n'y a pas de notion d'automate à pile minimal comme pour les AFD...
- Et en général, savoir si deux automates à pile sont équivalents est un problème indécidable!
- (C'est en revanche décidable pour les automates à pile déterministes...)

Automates à pile déterministe

Informellement, un automate à pile est déterministe quand il n'y a au plus une transition possible dans chaque configuration de l'automate,.

Définition (Automates à pile déterministe)

Un automate à pile est **déterministe** si pour tout $q_i \in Q$, $a \in \Sigma$, $\gamma \in \Gamma$, $|\delta(q_i, a, \gamma) \cup \delta(q_i, \epsilon, \gamma) \cup \delta(q_i, a, \epsilon) \cup \delta(q_i, \epsilon, \epsilon)| \leq 1$.

Contrairement aux AFI qui sont aussi puissant que les AFD (ils reconnaissent les langages réguliers), les AP déterministes sont strictement moins puissant que les AP indéterministes.

Théorème (Palindromes et déterminisme)

Le langage hors-contexte L des palindromes sur l'alphabet $\{a, b\}$ n'est pas reconnu par un AP déterministe.

Table des matières

- 1 Langages
- 2 Langages réguliers: AFD, AFI et expressions régulières
- 3 Langages hors-contexte: AP
- 4 Grammaires

Définition (Grammaires de réécriture)

Une grammaire est un 4-uplet (N, Σ, R, S) où :

- N est un ensemble fini de **symboles non terminaux** (en général représentés par des majuscules).
- Σ est un ensemble fini de **symboles terminaux** avec $N \cap \Sigma = \emptyset$ (en général représentés par des minuscules).
- R est un sous ensemble fini de **règles**.

Une règle de R est de la forme $\alpha \rightarrow \beta$ avec

- $\alpha, \beta \in (N \cup \Sigma)^*$ et ,
- α contient **au moins un non terminal**.

α et β sont appelé respectivement **partie gauche et droite de la règle**.

- S est un élément de N appelé l'**axiome** de la grammaire.

Pour alléger les notations, au lieu d'écrire plusieurs règles:

$$\begin{array}{lcl} \alpha_1 & \rightarrow & \beta_1 \\ \alpha_1 & \rightarrow & \beta_2 \\ \alpha_1 & \rightarrow & \beta_3 \end{array}$$

on écrit simplement:

$$\alpha_1 \rightarrow \beta_1 \mid \beta_2 \mid \beta_3$$

Proto-mots d'une grammaire

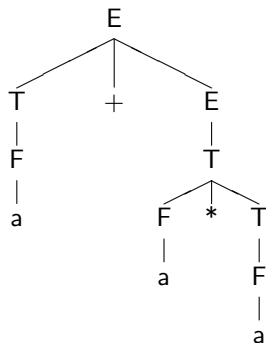
Les **proto-mots** d'une grammaire $G = (N, \Sigma, R, S)$ sont des mots construits sur l'alphabet $\Sigma \cup N$, on les définit récursivement de la façon suivante :

- S est un proto-mot de G
- si $\alpha\beta\gamma$ est un proto-mot de G et $\beta \rightarrow \delta \in R$ alors $\alpha\delta\gamma$ est un proto-mot de G .

Un proto-mot de G ne contenant aucun symbole non-terminal est appelé un mot **engendré** par G . Le **langage engendré par G** , noté $L(G)$ est l'ensemble des mots engendrés par G .

Deux grammaires G et G' sont **équivalentes** si $L(G) = L(G')$.

Arbre de dérivation



Un arbre de dérivation pour $G = (N, \Sigma, R, S)$ est un arbre ordonné et étiqueté dont les étiquettes appartiennent à l'ensemble $N \cup \Sigma \cup \{\varepsilon\}$. Si un nœud de l'arbre est étiqueté par le non terminal A et ses fils sont étiquetés X_1, X_2, \dots, X_n alors la règle $A \rightarrow X_1X_2\dots X_n$ appartient à P .

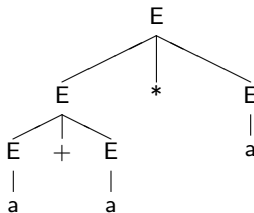
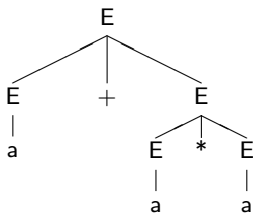
- Un arbre de dérivation indique les règles qui ont été utilisées dans une dérivation, mais pas l'ordre dans lequel elles ont été utilisées.

Ambiguïté

Une grammaire G est **ambiguë** s'il existe au moins un mot m dans $L(G)$ auquel correspond plus d'un arbre de dérivation.

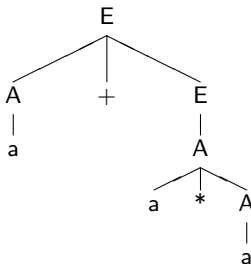
Exemple :

$$E \rightarrow E + E \mid E * E \mid a$$



Exemple :

$$E \rightarrow A + E \mid A, A \rightarrow a * A \mid a$$



Types de règles

Les grammaires peuvent être classées en fonction de la forme de leurs règles de production :

- Une règle est **régulière à gauche** si et seulement si elle est de la forme $A \rightarrow xB$, $A \rightarrow x$ ou $A \rightarrow \epsilon$ avec $A, B \in N$ et $x \in \Sigma^*$.
- Une règle est **régulière à droite** si et seulement si elle est de la forme $A \rightarrow Bx$, $A \rightarrow x$ ou $A \rightarrow \epsilon$ avec $A, B \in N$ et $x \in \Sigma^*$.
- Une règle $A \rightarrow \alpha$ est une règle **hors-contexte** si et seulement si : $A \in N$ et $\alpha \in (N \cup \Sigma)^*$.

Une grammaire est :

- **régulière** si elle est régulière à droite ou régulière à gauche.
Une grammaire est régulière à gauche (*resp.* droite) si **toutes** ses règles sont régulières à gauche (*resp.* droite).
- **hors contexte** si toutes ses règles de production sont hors contexte.

- Les grammaires régulières engendrent exactement les langages réguliers.
- Les grammaires hors-contexte engendrent exactement les langages hors-contextes (aussi appelés *algébriques*).

Exemples de langages réguliers et hors-contextes

Langages réguliers:

$$L_1 = \{m \in \{a, b\}^*\}$$

$$G_1 = (\{S\}, \{a, b\}, \{S \rightarrow aS \mid bS \mid \varepsilon\}, S)$$

$$L_2 = \{m \in \{a, b\}^* \mid |m|_a \bmod 2 = 0\}$$

$$G_2 = (\{S, T\}, \{a, b\}, \left\{ \begin{array}{l} S \rightarrow aT \mid bS \mid \varepsilon, \\ T \rightarrow aS \mid bT \end{array} \right\}, S)$$

Langages hors-contextes:

$$L_1 = \{a^n b^n \mid n \geq 0\}$$

$$G_1 = (\{S\}, \{a, b\}, \{S \rightarrow aSb \mid \varepsilon\}, S)$$

$$L_2 = \{mm^{-1} \mid m \in \{a, b\}^*\} \quad (\text{langage miroir - palindromes paires})$$

$$G_2 = (\{S\}, \{a, b\}, \{S \rightarrow aSa \mid bSb \mid \varepsilon\}, S)$$