Théorie des Langages

Cours 5 : Preuves, langages non réguliers

Lionel Rieg

Grenoble INP - Ensimag, 1re année

Preuves de langages d'automates

Comment montrer qu'un automate A est correct ?

- équivalence entre deux automates
- égalité avec un langage connu par ailleurs

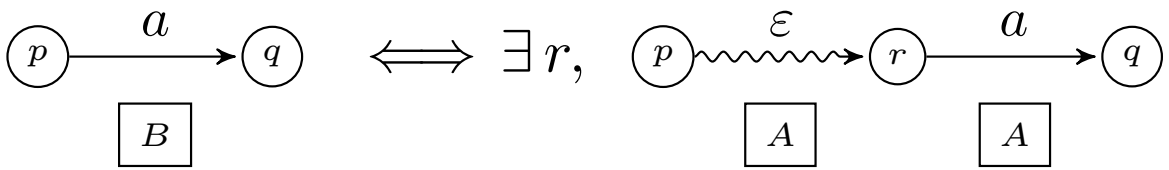
2 méthodes

- On peut caractériser les chemins acceptants
  Il faut savoir caractériser les chemins dans un automate
  $\rightsquigarrow$ intéressant si les automates sont proches (transformation)
- On peut identifier les langages reconnus depuis chaque état de A
  $\rightsquigarrow$ équations entre ces langages données par A (cf. système d'ER)
  $\rightsquigarrow$ preuve par double inclusion

Dans la suite, preuves des transformations d'automates (sauf minimisation)

Rappels sur l'élimination des ε-transistions

1. Calculer $\mathrm{Acc}_\varepsilon(p)$, les états accessibles par ε-transistions
   $\rightsquigarrow$ par itération (cf. cours 2)
2. Construire un automate B équivalent sans ε-transition



- $(p, a, q) \in \delta' \iff a \neq \varepsilon$ et $\exists r \in \mathrm{Acc}_\varepsilon(p)$ tel que $(r, a, q) \in \delta$
- $F' = \{p \in Q \mid \mathrm{Acc}_\varepsilon(p) \cap F \neq \emptyset\}$

Remarques

- *Même Q, V et I, seuls δ et F changent*
- *Par construction, B est sans ε-transition*

Correction de l'élimination des ε-transistions

Théorème

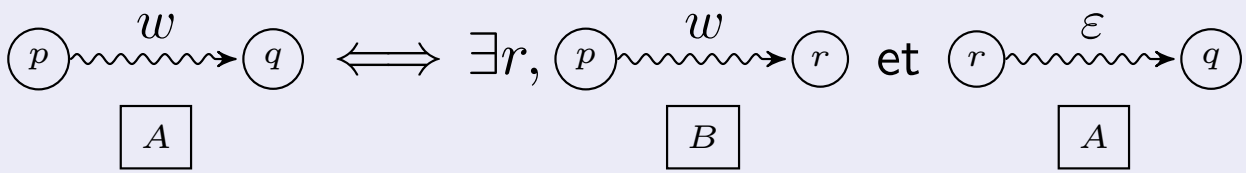
$\forall$ *automate* A, *l'automate* B *défini précédemment est équivalent à* A.

Lemme intermédiaire : caractérisation des chemins

L'automate B vérifie la propriété suivante :

Il existe un chemin de p à q de trace w dans A
si et seulement si
il existe $r \in Q$ tel qu'il existe un chemin de p à r de trace w dans B
et un chemin de r à q de trace ε dans A.

Graphiquement :

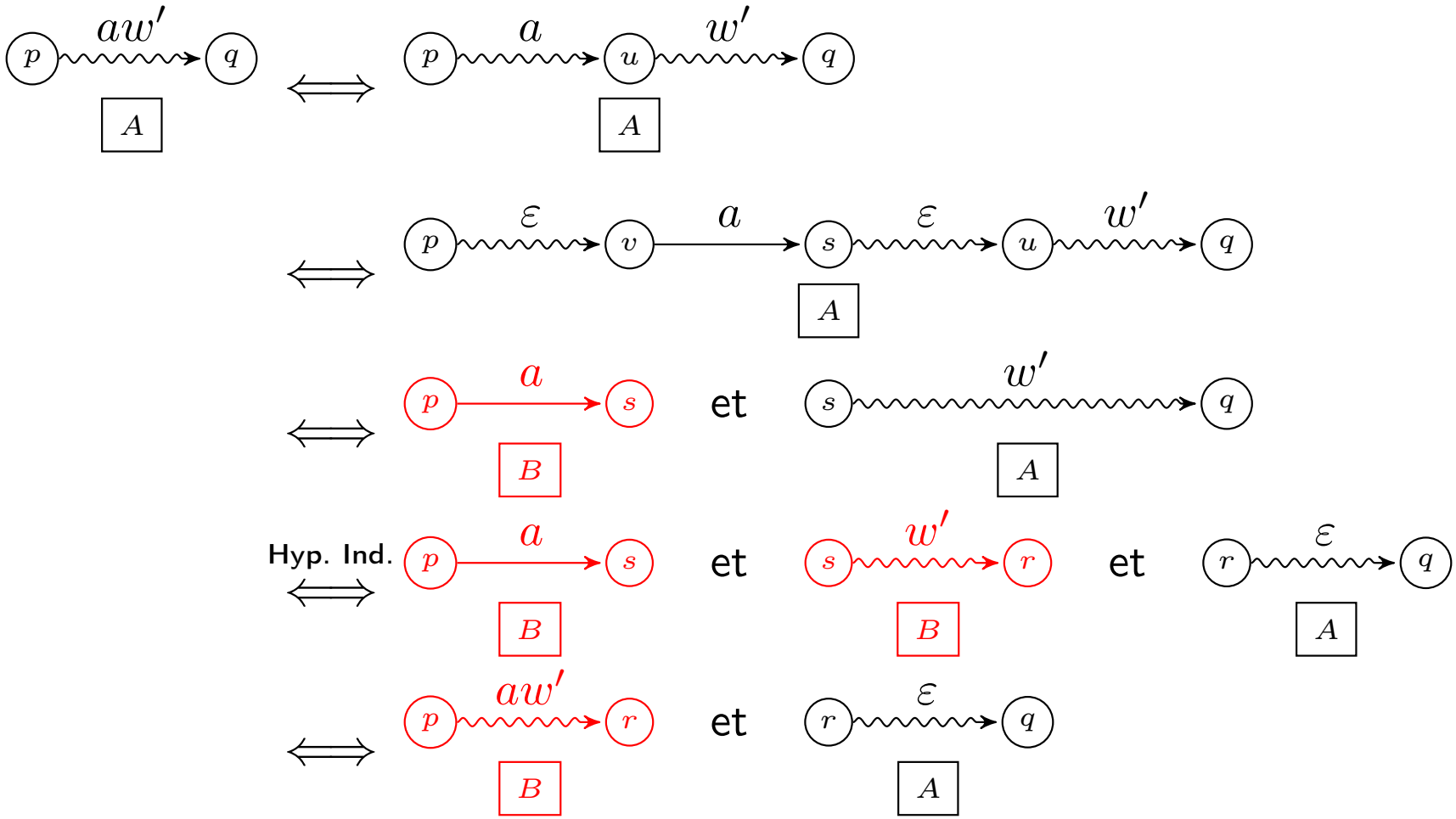


Preuve par induction sur w.

- Base : $w = \varepsilon$. Il suffit de prendre $r \stackrel{\text{def}}{=} p$.

Preuve par induction, suite

- Induction : $w = aw' \quad (a \in V)$



Preuve du théorème

$$
\begin{aligned}
w \in \mathcal{L}(A) \quad &\iff \quad \exists \text{ un chemin de } q_0 \in I \text{ à } q_f \in F \text{ de trace } w \text{ dans } A \\
&\iff \quad \exists r \in Q, \exists \text{ un chemin de } q_0 \in I \text{ à } r \text{ de trace } w \text{ dans } B \\
&\qquad\quad \text{et } \exists \text{ un chemin de } r \text{ à } q_f \in F \text{ de trace } \varepsilon \text{ dans } A \\
&\iff \quad \exists r \in Q, \exists \text{ un chemin de } q_0 \in I \text{ à } r \text{ de trace } w \text{ dans } B \\
&\qquad\quad \text{et } r \in F' \\
&\iff \quad w \in \mathcal{L}(B)
\end{aligned}
$$

Les automates A et B sont bien équivalents.

Rappels sur la déterminisation

Idée : suivre tous les chemins en parallèle

- Entrée : un automate $A = \langle Q, V, \delta_A, I, F_A \rangle$ sans ε-transistions
- Sortie : un automate B déterministe complet équivalent à A

Définition (Automate des parties)

Etant donné un automate $A = \langle Q, V, \delta_A, I, F_A \rangle$ sans ε-transition, on construit l'automate $B = \langle \mathcal{P}(Q), V, \delta_B, \{I\}, F_B \rangle$, où :

- δ_B est défini par

$$\forall P \subseteq Q, \forall a \in V,\ \delta_B(P, a) = \{q \in Q \mid \exists p \in P : (p, a, q) \in \delta_A\}$$

- $F_B = \{P \subseteq Q \mid P \cap F_A \neq \emptyset\}$

Propriété (caractérisation des chemins)

Proposition

Pour tout $w \in V^*$ et pour tout $P \subseteq Q$, on a
$\delta_B^*(P, w) = \{q \in Q \mid \exists p \in P, \exists$ *un chemin dans A de p à q de trace $w\}$.*

Preuve : par induction sur w

- $w = \varepsilon$:
  - $\delta_B^*(P, \varepsilon) = P$
  - $\{q \in Q \mid \exists p \in P, \exists$ un chemin dans A de p à q de trace $\varepsilon\} = P$
    (car A est un automate sans ε-transition)

Propriété (caractérisation des chemins)

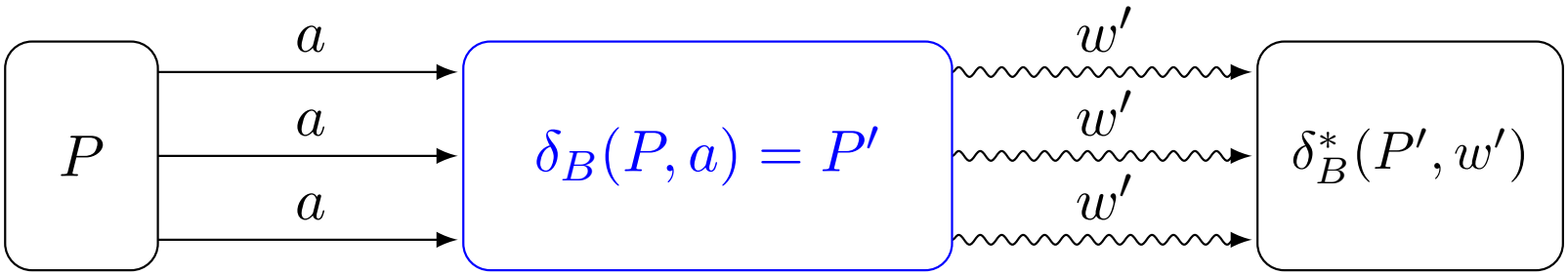
Proposition

Pour tout $w \in V^*$ et pour tout $P \subseteq Q$, on a
$\delta_B^*(P, w) = \{q \in Q \mid \exists p \in P, \exists$ *un chemin dans A de p à q de trace w*$\}$.

Preuve : par induction sur w

- $w = aw'$:
  - Posons $P' \overset{\text{def}}{=} \delta_B(P, a) = \{q \in Q \mid \exists p \in P, (p, a, q) \in \delta_A\}$

$$\begin{aligned} \delta_B^*(P, aw') &= \delta_B^*(\delta_B(P, a), w') = \delta_B^*(P', w') \\ &\overset{\text{Hyp. Ind.}}{=} \{q' \in Q \mid \exists p' \in P', \exists \text{ un chemin de } p' \text{ à } q' \text{ de trace } w'\} \\ &= \{q \in Q \mid \exists p \in P, \exists \text{ un chemin de } p \text{ à } q \text{ de trace } aw'\} \end{aligned}$$



Correction de la déterminisation

Théorème

L'automate B est équivalent à A.

Preuve :

$$
\begin{aligned}
w \in \mathcal{L}(A) &\iff \exists p \in I,\ \exists q \in F_A,\ \exists \text{ un chemin de } p \text{ à } q \text{ de trace } w \\
&\iff \exists q \in \delta_B^*(I, w), q \in F_A \\
&\iff \delta_B^*(I, w) \cap F_A \neq \emptyset \\
&\iff \delta_B^*(I, w) \in F_B \\
&\iff w \in \mathcal{L}(B)
\end{aligned}
$$

Les automates A et B sont bien équivalents.

Langages réguliers et non réguliers

Stabilité des langages réguliers

Théorème

La classe des langages réguliers est fermée par :

- *union, concaténation et concaténation itérée (cf. cours 3)*
- *substitution régulière et homomorphisme*

Substitution régulière

Définition (Substitution régulière)

Soit V et W deux vocabulaires.
Une substitution régulière est une fonction $s : V \to \mathcal{P}(W^*)$
qui à tout $a \in V$ associe un langage régulier $s(a) \subseteq W^*$.

On étend s aux mots par induction : $s^* : V^* \to \mathcal{P}(W^*)$

- $s^*(\varepsilon) = \{\varepsilon\}$
- $s^*(aw) = s(a).s^*(w)$

On étend s^* aux langages : $\overline{s} : \mathcal{P}(V^*) \to \mathcal{P}(W^*)$.

$$\forall L \subseteq V^*,\ \overline{s}(L) = \bigcup_{w \in L} s^*(w)$$

On pourra noter s au lieu de s^* ou $\overline{s}$.

Propriété de fermeture

Exemple

Soient $V = \{a, b\}$ et $W = \{c, d\}$. On pose :

$$\begin{aligned} L &= \left\{ab^i \;\middle|\; i \geqslant 0\right\} = ab^* \\ s(a) &= \left\{c^i \;\middle|\; i \geqslant 0\right\} = c^* \\ s(b) &= \{cd\} = cd \end{aligned}$$

Alors $s(L) = \left\{c^i(cd)^j \;\middle|\; i, j \geqslant 0\right\} = c^*(cd)^*$

Théorème

La classe des langages réguliers est fermée par substitution régulière.

Autrement dit, si L est un langage régulier et s est une substitution régulière, alors $s(L)$ est un langage régulier.

Preuve du théorème

Étapes :

1. On étend les substitutions régulières aux expressions régulières :
$$\forall \text{ ER } E \text{ sur } V,\ s(E) \stackrel{\text{def}}{=} s(\mathcal{L}(E)).$$

2. On prouve que si E est une expression régulière sur V, alors il existe une expression régulière E' sur W telle que $s(E) = \mathcal{L}(E')$.

Lemme intermédiaire sur les mots

Proposition

Soit s une substitution régulière. Pour tous $u, v \in V^*$, on a

$$s^*(u.v) = s^*(u).s^*(v) \ .$$

Preuve : Soit $v \in V^*$. On prouve que pour tout $u \in V^*$ on a $s^*(u.v) = s^*(u).s^*(v)$ par induction structurelle sur u.

Base
$$\begin{aligned} s^*(\varepsilon.v) &= s^*(v) \\ &= \{\varepsilon\}.s^*(v) \\ &= s^*(\varepsilon).s^*(v) \end{aligned}$$

Induction
$$\begin{aligned} s^*(au'.v) &= s^*(a(u'.v)) && \text{(déf. de .)} \\ &= s(a).s^*(u'.v) && \text{(déf. de } s^*\text{)} \\ &= s(a).(s^*(u').s^*(v)) && \text{(Hyp. Ind.)} \\ &= (s(a).s^*(u')).s^*(v) && \text{(assoc. de .)} \\ &= s^*(au').s^*(v) && \text{(déf. de } s^*\text{)} \end{aligned}$$

Lemme intermédiaire sur les ER

Lemme

Soit s une substitution régulière, et soient E et E' des expressions régulières sur V. Alors

$$\begin{aligned} s(E.E') &= s(E).s(E') \\ s(E+E') &= s(E) \cup s(E') \\ s(E^*) &= s(E)^* \end{aligned}$$

Preuve.

- Prouvons que $s(E.E') \subseteq s(E).s(E')$.

  Soit $w \in s(E.E')$. Il existe $v \in E.E'$ tel que $w \in s(v)$.
  Comme $v \in E.E'$, il existe $u \in E$ et $u' \in E'$ tels que $v = u.u'$.
  Donc $w \in s(u.u') = s(u).s(u')$.
  Comme $s(u) \subseteq s(E)$ et $s(u') \subseteq s(E')$, on a le résultat.

  Exercice : Vérifier que $s(E).s(E') \subseteq s(E.E')$.

- Idem pour $E+E'$ et E^*. (preuves plus faciles)

Preuve du théorème

Étapes :

1. On étend les substitutions régulières aux expressions régulières :
$$\forall \text{ ER } E \text{ sur } V,\ s(E) \stackrel{\text{def}}{=} s(\mathcal{L}(E)).$$

2. On prouve que si E est une expression régulière sur V, alors il existe une expression régulière E' sur W telle que $s(E) = \mathcal{L}(E')$.

Preuve de l'étape 2 du théorème.
Par induction structurelle :

Base Pour $E \in \{\emptyset, \epsilon, a\}$, OK : $\emptyset$, $\{\varepsilon\}$ et $s(a)$
($s(a)$ régulier donc représentable par ER)

Induction Pour $E \in \{E_1.E_2,\ E_1 + E_2,\ {E_1}^*\}$, cf lemme précédent

Utilisation des ER comme description inductive des langages réguliers

Homomorphismes

Définition (Homomorphisme)

Une substitution régulière qui à tout $a \in V$ associe un singleton
OU une fonction qui à tout $a \in V$ associe un mot $w \in W^*$.

Exemple

$$\begin{aligned} L &= \left\{ab^i \;\middle|\; i \geqslant 0\right\} = ab^* \\ s(a) &= \{cdc\} = cdc \\ s(b) &= \{dc\} = dc \end{aligned}$$

Alors $s(L) = \left\{cdc(dc)^i \;\middle|\; i \geqslant 0\right\} = cdc(dc)^* = c(dc)^+$

Corollaire

La classe des langages réguliers est fermée par homomorphisme.
(et par homomorphisme inverse)

⚠ Si $h(L) = L'$, on n'a pas forcément $h^{-1}(L') = L$.

Stabilité des langages réguliers

Théorème

La classe des langages réguliers est fermée par :

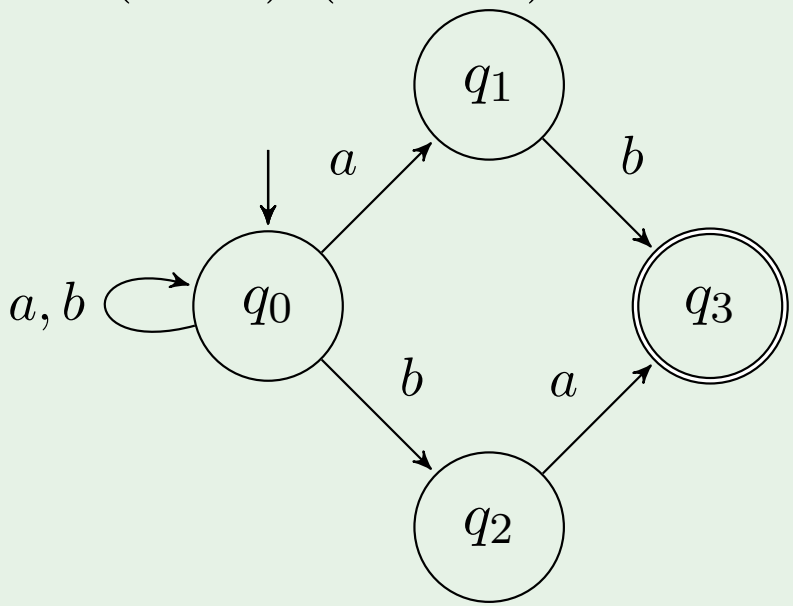
- *union, concaténation et concaténation itérée (cf. cours 3)*
- *substitution régulière et homomorphisme*
- *complémentation*

Complémentation

Question : si L est un langage régulier, peut-on construire un automate qui reconnaît $\overline{L} = V^* \setminus L$?

Exemple

$L = (a+b)^*(ab+ba)$

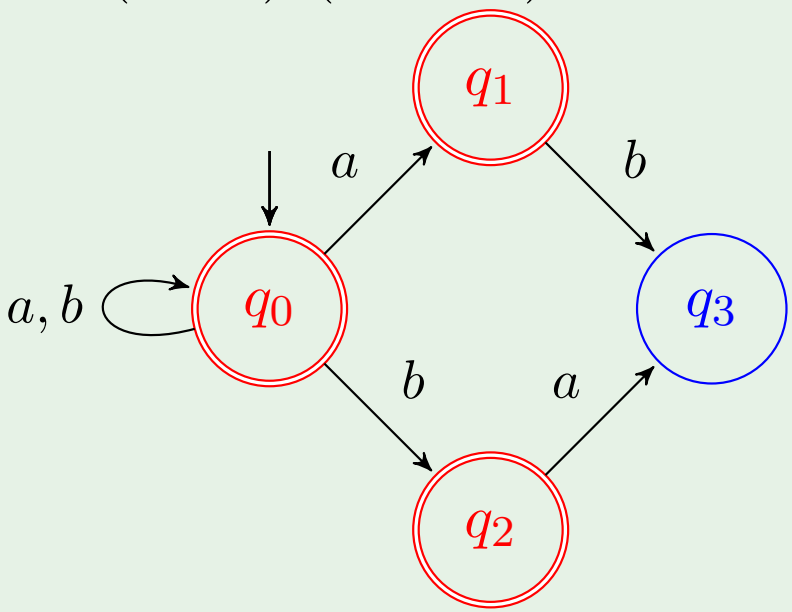


Complémentation

Question : si L est un langage régulier, peut-on construire un automate qui reconnaît $\overline{L} = V^* \setminus L$?

Exemple

$L = (a+b)^*(ab+ba)$



$\mathcal{L}(A') = (a+b)^*$ perdu. . .

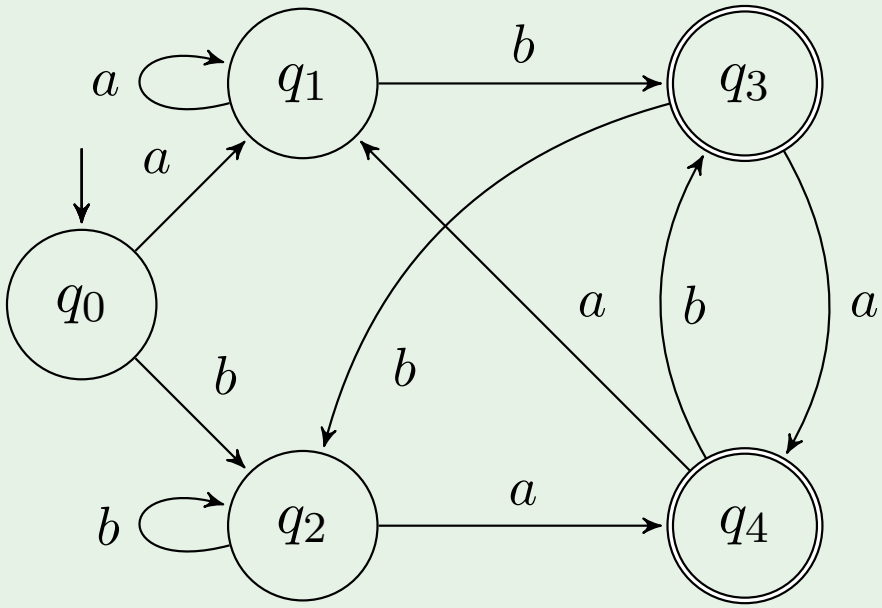
Complémentation (suite)

Problème : on ne peut pas intervertir F et $Q \setminus F$ car
il peut y avoir deux chemins de même trace dans un AFND,
et si l'un mène en F mais pas l'autre on accepte...

Solution : On déterminise...

Exemple

$L = (a+b)^*(ab+ba)$



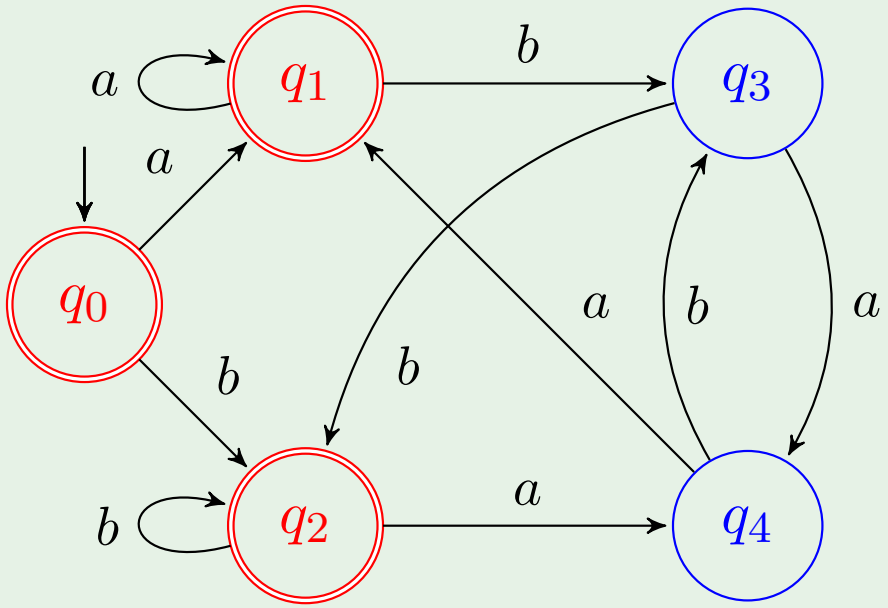
Complémentation (suite)

Problème : on ne peut pas intervertir F et $Q \setminus F$ car
il peut y avoir deux chemins de même trace dans un AFND,
et si l'un mène en F mais pas l'autre on accepte...

Solution : On déterminise...

Exemple

$L = (a+b)^*(ab+ba)$



Gagné !

Exercice : déterminer l'ER associée à cet automate

Remarque : même problème et même solution s'il manque des chemins

Complémentation (fin)

Proposition

La classe des langages réguliers est fermée par complémentation.

Preuve. Soit L un langage régulier et considérons $A = \langle Q, V, \delta, \{q_0\}, F \rangle$ un automate fini déterministe complet tel que $\mathcal{L}(A) = L$.

Posons $A' \stackrel{\text{def}}{=} \langle Q, V, \delta, \{q_0\}, Q \setminus F \rangle$.
A' étant déterministe complet, on a :

$$w \in \mathcal{L}(A') \iff \delta^*(q_0, w) \in Q \setminus F \iff \delta^*(q_0, w) \notin F \iff w \notin \mathcal{L}(A)$$

Stabilité des langages réguliers

Théorème

La classe des langages réguliers est fermée par :

- *union, concaténation et concaténation itérée (cf. cours 3)*
- *substitution régulière et homomorphisme*
- *complémentation*
- *intersection, différence*

Preuve :

$$L \cap M = \overline{\overline{L} \cup \overline{M}}$$
$$L \setminus M = L \cap \overline{M}$$

Attention : les inclusions ne donnent rien !

L régulier et $L \subset M \not\Longrightarrow M$ régulier — penser à $L = \emptyset$

M régulier et $L \subset M \not\Longrightarrow L$ régulier — penser à $M = V^*$

Langages non-réguliers

Comment prouver qu'un langage n'est pas régulier ?

Autrement dit : Étant donné un langage L, comment prouver que pour tout automate fini A, $\mathcal{L}(A) \neq L$?

Idée : Utiliser des propriétés de fermeture

Supposons donné un langage M dont on connaît la non-régularité.

Pour prouver que L n'est pas régulier, on peut procéder par l'absurde :

1. On suppose que L est régulier.
2. On exhibe une série de transformations qui préservent la régularité et qui permettent de passer de L à M.
3. Donc M devrait être régulier.
   Contradiction : l'hypothèse que L était régulier est fausse.

Cette technique de preuve est appelée réduction : on réduit le problème de la régularité de L à celle de M. (*cf.* ROMD)

Exercice

On admet que $M = \{0^p 1^p \mid p \geqslant 0\}$ n'est pas régulier.

Montrer que $L = \{wcw' \mid w, w' \in \{a,b\}^*\ ,\ |w|_a = |w'|_b\}$ n'est pas régulier.

1. Supposons que L est régulier.
2. Alors $L' \stackrel{\text{def}}{=} L \cap a^*cb^* = L \cap \{a^p cb^q \mid p, q \geqslant 0\} = \{a^p cb^p \mid p \geqslant 0\}$ est nécessairement régulier.

2'. Soit l'homomorphisme h défini par :

$$h : \begin{cases} a & \to & 0 \\ b & \to & 1 \\ c & \to & \varepsilon \end{cases}$$

Le langage $h(L')$ est nécessairement régulier.

3. Mais $h(L') = M$, contradiction.

Un premier langage non-régulier

Pour utiliser les propriétés de fermeture, il faut connaître au moins un langage M non-régulier.

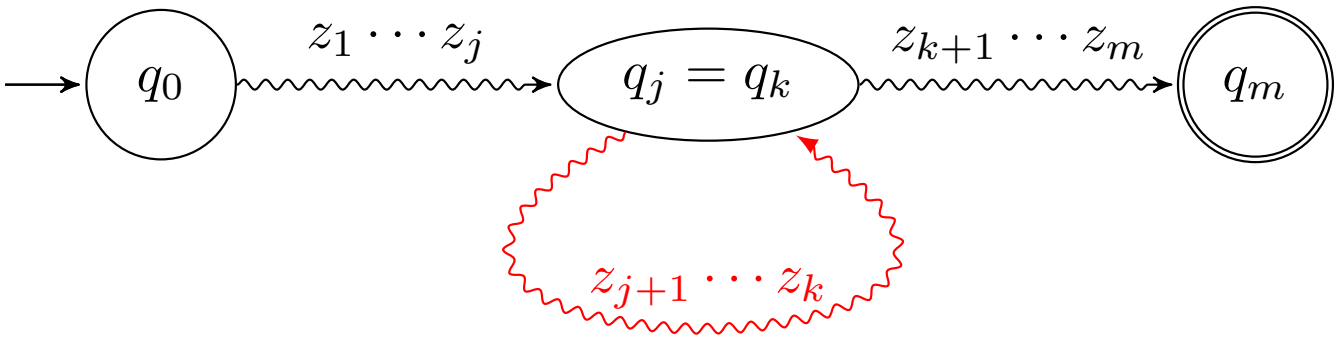
Comment prouver que M n'est pas régulier ?

- En se servant d'une condition nécessaire sur les langages réguliers qui permettra de refaire un raisonnement par l'absurde
- On va supposer qu'il existe un automate fini A tel que $\mathcal{L}(A) = M$ et tenter d'aboutir à une contradiction
- La condition nécessaire la plus standard est donnée par le lemme de l'étoile

  (lemme de pompage, de la pompe, *pumping lemma*)

Principe du lemme de l'étoile

Soit $A = \langle Q, V, \delta, I, F \rangle$ un automate *sans ε-transition*, avec $|Q| = n \geqslant 1$.
Soit $z = z_1 \cdots z_m$ un mot sur V de longueur $m \geqslant n$ reconnu par A.
Il existe donc un chemin $(q_0, z_1, q_1) \cdots (q_{m-1}, z_m, q_m)$ dans A,
avec $q_0 \in I$ et $q_m \in F$.
Il y a $m+1$ états dans ce chemin et n états dans Q, et $m+1 > n$
donc $\exists j, k$ tels que $0 \leqslant j < k \leqslant n$ et $q_j = q_k$



$\forall i \geqslant 0,\ z_1 \cdots z_j (z_{j+1} \cdots z_k)^i z_{k+1} \cdots z_m \in \mathcal{L}(A)$

Énoncé du lemme de l'étoile

Lemme

Soit L un langage régulier. Alors il existe $n \geqslant 1$ tel que pour tout mot z, si $z \in L$ et $|z| \geqslant n$, alors z est de la forme uvw avec :

- $|uv| \leqslant n$
- $|v| \geqslant 1$
- $\forall i \geqslant 0, uv^i w \in L \qquad \iff uv^* w \subseteq L$

Attention

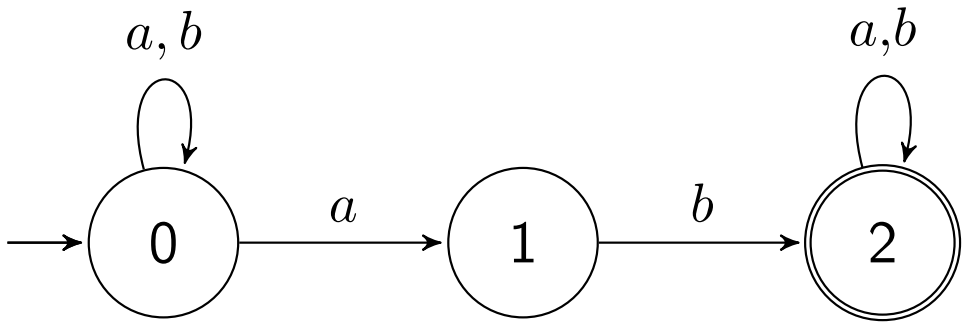
Le lemme de l'étoile est une condition nécessaire mais non suffisante des langages réguliers : il existe des langages non-réguliers qui le satisfont.
Exemple : $\{c^n a^p b^p \mid n, p \in \mathbb{N}\} \cup \{a^n b^p \mid n, p \in \mathbb{N}\}$

Remarques

- *Que se passe-t-il pour les langages finis ?*
- *Des lemmes de l'étoile existent aussi pour d'autres classes de langages.*

Illustrations

- Pour le langage V^*abV^* (les mots qui contiennent ab) :



On a $n = 3$ (nombre d'états de l'automate).

Pour $z = abb$, le chemin acceptant dans l'automate donne $u = ab$, $v = b$, $w = \varepsilon$.

Pour $z = aab$, le chemin acceptant dans l'automate donne $u = \varepsilon$, $v = a$, $w = ab$.

Et pour $z = aabb$? 2 découpages mais un seul vérifie $|uv| \leqslant n$!

- Pour le langage $\{ab\}$: même automate, moins les boucles.
On a $n = 3$ et il n'existe pas de mot $z \in \{ab\}$ tel que $|z| \geqslant 3$.
Le lemme de l'étoile est vrai par vacuité.

Comment se servir du lemme de l'étoile ?

On procède par l'absurde : on suppose que L est régulier et satisfait donc le lemme de l'étoile.

- On considère l'entier n du lemme. (on ne sait rien de sa valeur)
- On choisit un mot $z \in L$ de longueur au moins n. (z dépendra de n)
- Le mot z est décomposé en uvw, où $|uv| \leqslant n$ et $|v| \geqslant 1$.
  (on ne contrôle pas la façon dont z est décomposé, hormis la contrainte sur les longueurs)
- On choisit une valeur de i telle que $uv^iw \notin L$.

On obtient une contradiction : L ne peut pas être régulier.

Remarque

On utilise en fait la contraposée du lemme de l'étoile :

$$L \text{ régulier} \implies \exists n \geqslant 1, \forall z \in L, \dots$$

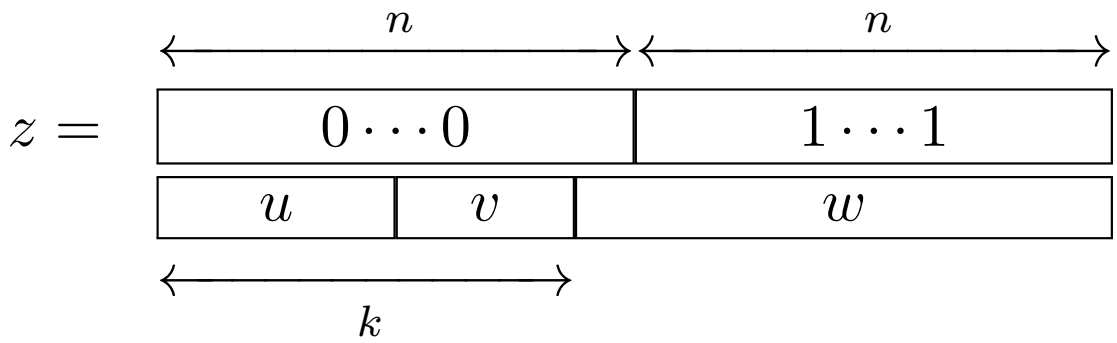
$$\forall n \geqslant 1, \exists z \in L, \dots \implies L \text{ non régulier}$$

Exemple

Montrer que $M = \{0^p 1^p \mid p \geqslant 0\}$ n'est pas régulier.

On suppose que M est régulier et satisfait donc le lemme de l'étoile.

- Soit n l'entier du lemme. (on ne contrôle pas sa valeur)
- On choisit $z = 0^n 1^n \in M$. On a $|z| = 2n \geqslant n$.
- z est décomposé en uvw où $k \overset{\text{def}}{=} |uv| \leqslant n$ et $|v| \geqslant 1$.



Comme $k \leqslant n$, on sait que $uv = 0^k$ sans pour autant connaître k !

- On choisit $i = 0$, on devrait avoir $uv^0w = uw \in M$.
  Mais $uw = 0^{n-|v|}1^n \notin M$, contradiction.

Conclusion : M n'est pas régulier.

Reconnaître des langages non réguliers

Comment étendre les automates pour reconnaître plus que les langages réguliers ?
Qu'est-ce que les AF ne peuvent pas faire (*cf.* $\{a^n b^n \mid n \in \mathbb{N}\}$) ?
compter sans valeur maximale (pas de mémoire non bornée)

- Automate à un compteur (non borné)
  $\rightsquigarrow$ on peut reconnaître $\{a^n b^n \mid n \in \mathbb{N}\}$
  $\rightsquigarrow$ on peut aussi reconnaître les mots bien parenthésés
  (toute parenthèse ouverte est refermée)
  Et si on a plusieurs type de parenthèses : () et [] ?
- Automate à une pile
  $\rightsquigarrow$ on peut reconnaître les mots bien parenthésés avec () et []
  $\rightsquigarrow$ et beaucoup d'autres choses !
- Différence entre compteur et pile ?
  Cela dépend des opérations autorisée sur le compteur
  - Si multiplication/division autorisée : aucune
    $\rightsquigarrow$ on peut représenter une pile par chaque chiffre d'un compteur
  - Si seulement ± 1 (ou $\pm k$) : pile $>$ compteur

Reconnaître des langages non réguliers

Et si on veut aller plus loin qu'une pile ?
un ruban (lire + écrire + marche arrière)

- Automate avec un ruban : machine de Turing

Proposition (Thèse de Church-Turing)

Tout ce qui est calculable l'est par une machine de Turing.

$\rightsquigarrow$ Les machines de Turing sont le modèle concret le plus puissant.

- Il y a des problèmes qu'on ne peut calculer !

théorie de la calculabilité, *cf.* ROMD

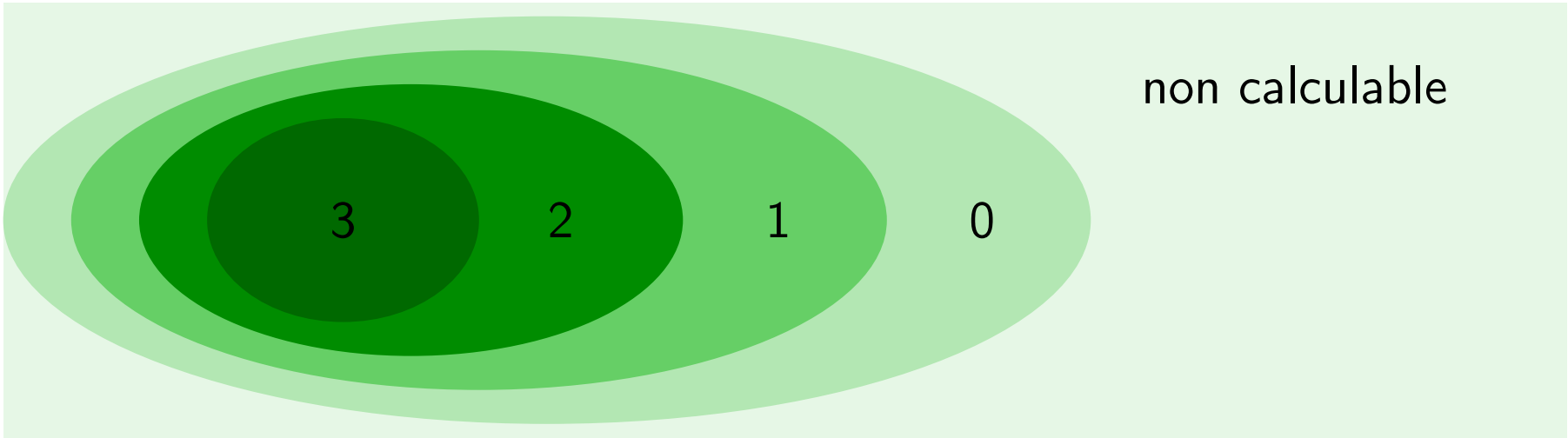
Et entre les deux ?

- Automate à deux piles ? $\iff$ machine de Turing

(avec deux piles on peut simuler un ruban)

Retour sur la hiérarchie de Chomsky

Retour sur la hiérarchie de Chomsky



Type	Langage	Engendré par	Reconnu par
0	Calculatoirement énumérable	Grammaire générale	**Machine de Turing**
1	Sous-contexte	Grammaire sous-contexte	Machine de Turing linéairement bornée
2	Hors-contexte	Grammaire hors-contexte	**Automate à pile**
3	Régulier	Grammaire linéaire à droite	**Automate fini**

Autre généralisation des automates : calcul

Un automate ne fait que décider si un mot appartient ou non à un langage.
Comment le généraliser pour faire du calcul avec un automate ?

Définition (Transducteur)

Un transducteur est un automate fini où chaque transition peut émettre un symbole (ou ε) sur une sortie.

Cela permet de calculer un résultat en reconnaissant un mot.
Exemple : en compilation

- lorsqu'on reconnaît un mot-clé, on indique duquel il s'agit.
- lorsqu'on reconnaît une constante, on calcule sa valeur.