LF – Théorie des langages formels

Sylvain Brandel
2025 – 2026
sylvain.brandel@univ-lyon1.fr

Partie 2

LANGAGES RATIONNELS

Automates à états finis
Caractérisation, minimisation, rationalité

Automates à états finis

- Automate ou Machine
  - *Ang. Automaton / Machine*
- Automates ou Machines
  - *Ang. Automata / Machines …*

- Automates (à états) finis déterministes
  - *Ang. Deterministic Finite Automata (DFA)*
- Automates (à états) finis non déterministes
  - *Ang. Nondeterministic Finite Automata (NFA)*
- Automates à pile
  - *Ang. Pushdown Automata (PDA)*
- Machines de Turing
  - *Ang. Turing Machines*

Automates à états finis

- Exemple concret
  - Machine à café dans le hall du déambu (pas sans contact)
  - Barrière de péage du périph' (pas liber-t …)
  - Laverie (pas sans contact)
  - …

Automates finis déterministes

- Simulation d'une machine très simple :
  - Mémorisation d'un état
  - Programme sous forme de graphe étiqueté indiquant les transitions possibles

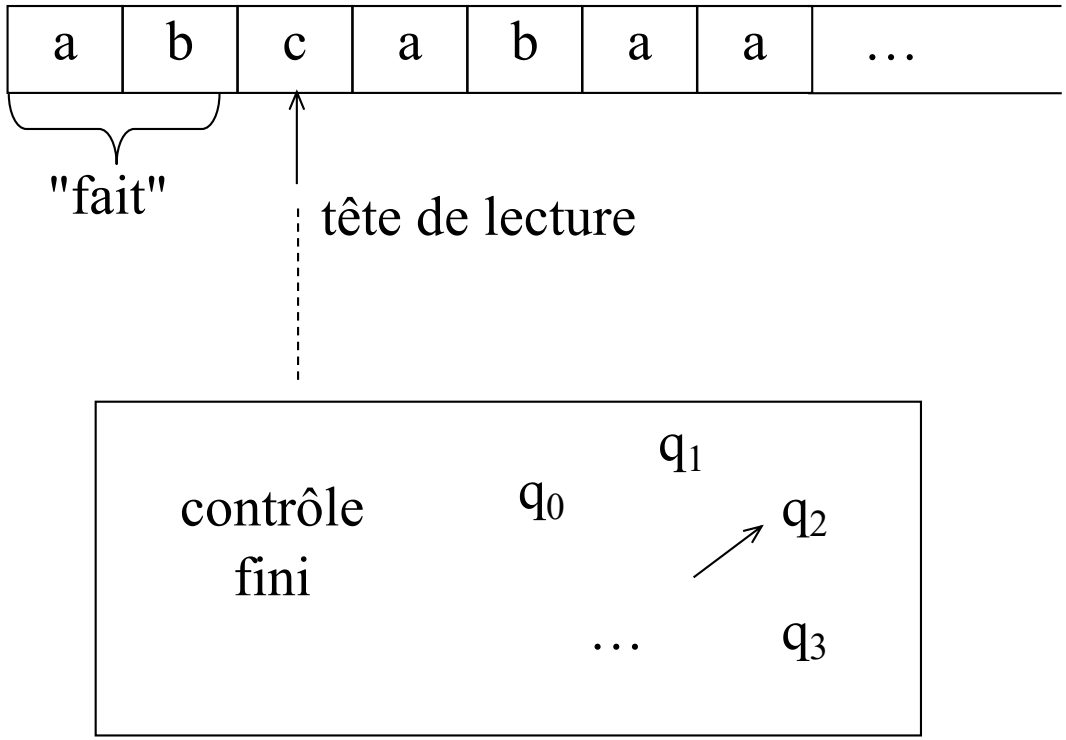
- Cette machine lit un mot en entrée

- Ce mot décrit une suite d'actions et progresse d'état en état

- Si le dernier état est un état acceptant et que le mot a été entièrement lu, on dit que le mot est accepté

$\Rightarrow$ Un automate permet de reconnaître un langage

Automates finis déterministes



- Un état dépend uniquement
  - De l'état précédent
  - Du symbole lu

Automates finis déterministes

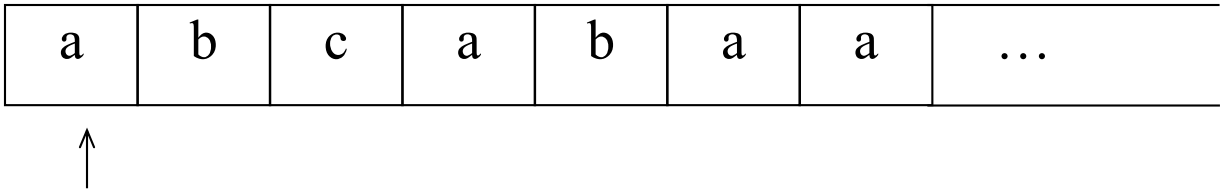
- Un automate déterministe fini est le quintuplet $M = (K, \Sigma, \delta, s, F)$ où :
  - K : ensemble fini (non vide) d'états
  - Σ : alphabet (ensemble non vide de symboles)
  - δ : fonction de transition : $K \times \Sigma \rightarrow K$

    $\delta(q, \sigma) = q'$ (q' : état de l'automate après avoir lu la lettre σ dans l'état q)
  - s : état initial : $s \in K$
  - F : ensemble des états finaux (acceptants) : $F \subset K$

- Si δ est une application, alors l'automate est complet

Automates finis déterministes

- Exécution



La machine

- lit a (qui est ensuite oublié),
- passe dans l'état $\delta(s, a)$ et avance la tête de lecture,
- répète cette étape jusqu'à ce que tout le mot soit lu ou plus de transition applicable

- La partie déjà lue du mot ne peut pas influencer le comportement à venir de l'automate

→ d'où la notion de configuration

Automates finis déterministes

- Configuration
  - état dans lequel est l'automate
  - mot qui lui reste à lire (partie droite du mot initial)

- Formellement

  une configuration est un élément quelconque de $K \times \Sigma^*$

- Exemple

  sur l'exemple précédent, la configuration est $(q_2, cabaa)$

Automates finis déterministes

- Le fonctionnement d'un automate est décrit par le passage d'une configuration C_1 à une configuration C_2

- C_2 est déterminée
  - en lisant un caractère
  - et en appliquant la fonction de transition

- Exemple
  - $(q_2, cabaa)$ → $(q_3, abaa)$ si $\delta(q_2, c) = q_3$

Automates finis déterministes

- Un automate M détermine une relation binaire $\vdash_M$ entre configurations définie par :
  - $\vdash_M \subset (K \times \Sigma^*)^2$
  - $(q, w) \vdash_M (q', w')$ ssi $\exists\, \sigma \in \Sigma$ tel que $w = \sigma w'$ et $\delta(q, \sigma) = q'$

- On dit alors qu'on passe de (q, w) à (q', w') en une étape

Automates finis déterministes

- On note $\vdash_M^*$ la fermeture transitive réflexive de $\vdash_M$:

  $(q, w) \vdash_M^* (q', w')$ signifie qu'on passe de (q, w) à (q', w') en zéro, une ou plusieurs étapes

- Un mot w est accepté par M ssi $(s, w) \vdash_M^* (q, \varepsilon)$, avec $q \in F$

- Le langage accepté par M est l'ensemble de tous les mots acceptés par M

  Ce langage est noté $L(M)$

Automates finis déterministes

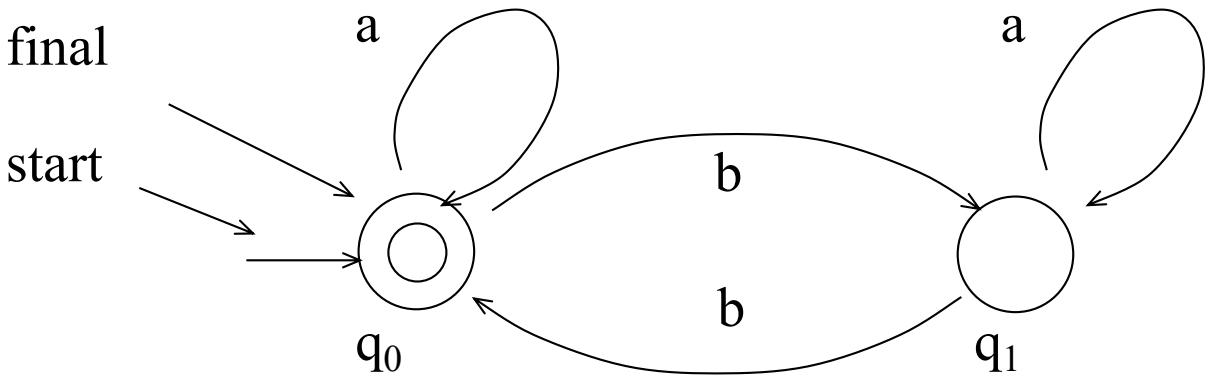
- Exemple $M = (K, \Sigma, \delta, s, F)$
  - K = {q_0, q_1}
  - Σ = {a, b}
  - s = q_0
  - F = {q_0}

δ :

q	σ	δ(q, σ)
q_0	a	q_0
q_0	b	q_1
q_1	a	q_1
q_1	b	q_0

$\Leftrightarrow$

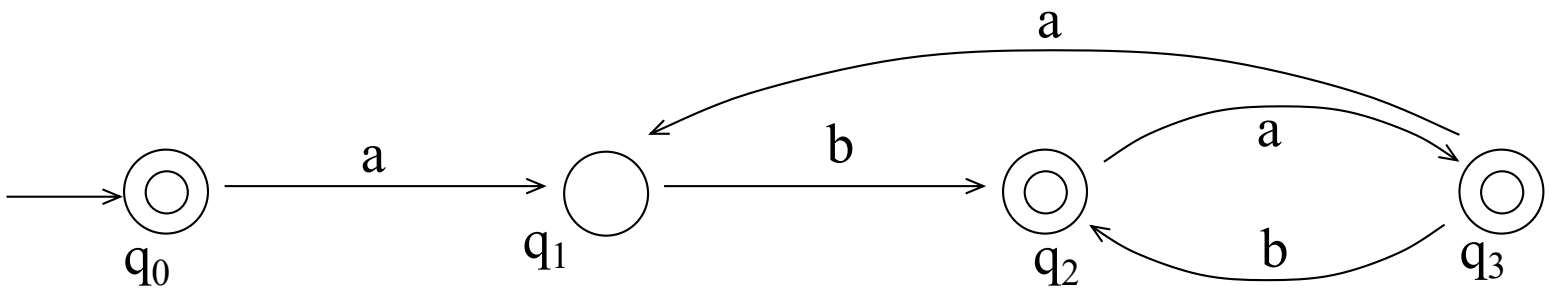
	a	b
q_0	q_0	q_1
q_1	q_1	q_0



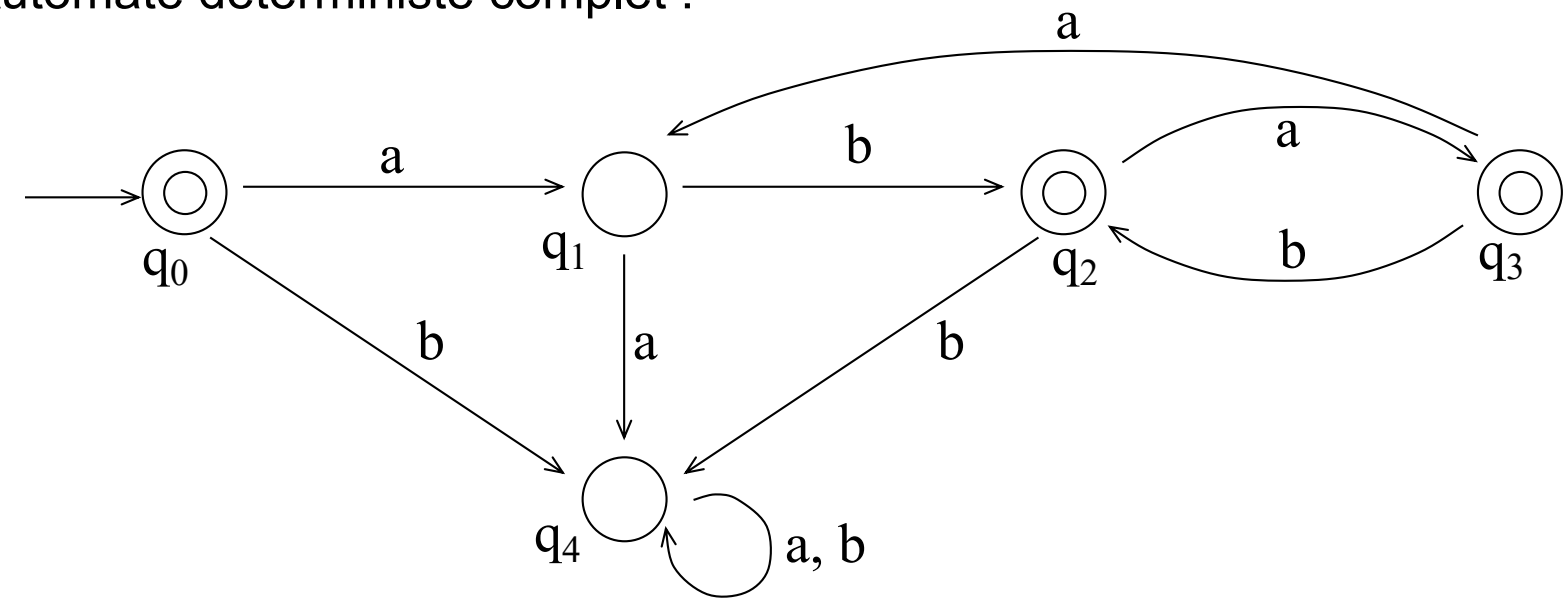
Automates finis déterministes

- $L = (ab \cup aba)^*$

  Automate déterministe non complet :



  Automate déterministe complet :



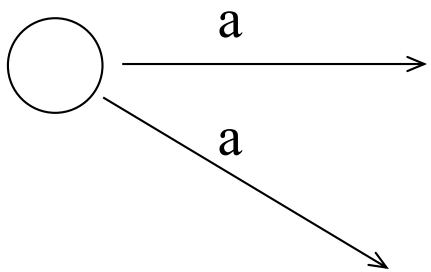
Automates finis non déterministes

- Remplacer la fonction $\vdash_M$ (ou δ) par une relation

- Une relation, c'est beaucoup plus général qu'une fonction

  → on a ainsi une classe plus large d'automates

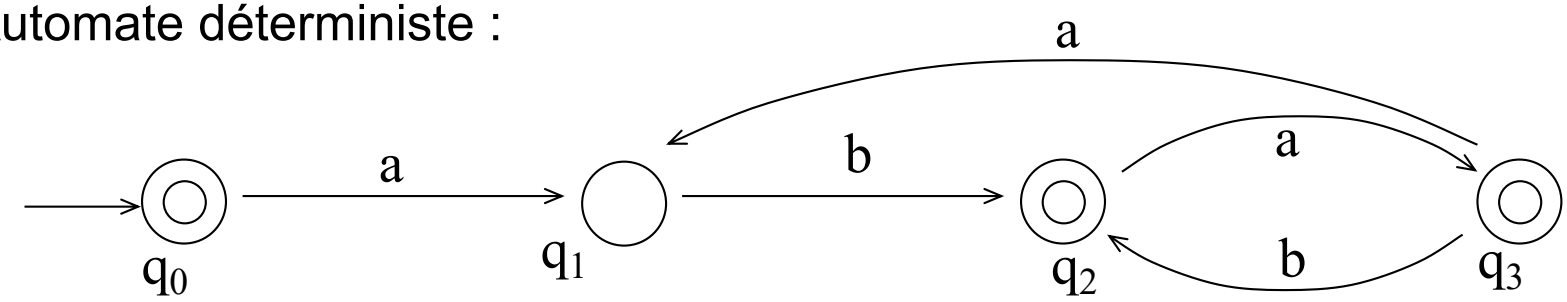
$\Rightarrow$ Dans un état donné, on pourra avoir :



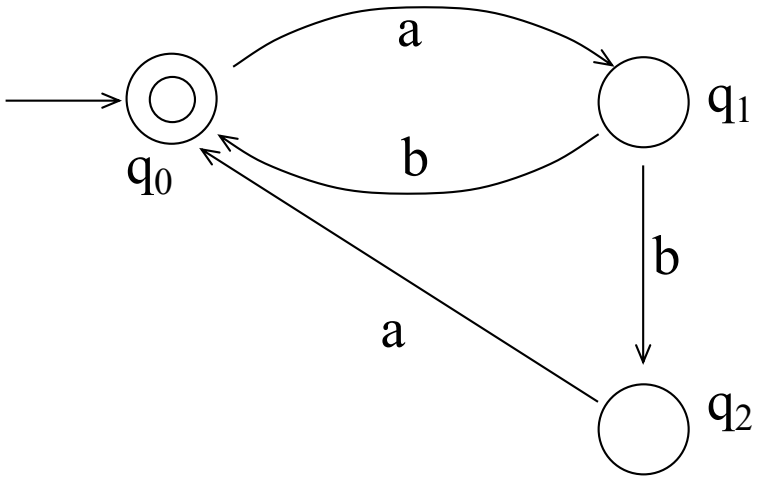
Automates finis non déterministes

- $L = (ab \cup aba)^*$

  Automate déterministe :



  Automate non déterministe :



Automates finis non déterministes

- Un automate non déterministe fini est le quintuplet $M = (K, \Sigma, \Delta, s, F)$ où
  - K : ensemble fini (non vide) d'états
  - Σ : alphabet (ensemble non vide de symboles)
  - Δ : relation de transition : $K \times \Sigma \times K$

    $(q, \sigma, q') \in \Delta$: σ –transition ($\sigma \in \Sigma$)
  - s : état initial : $s \in K$
  - F : ensemble des états finaux (acceptants) : $F \subset K$

- hormis Δ, le reste est identique à la formulation déterministe

- Δ peut-être définie comme une application $K \times \Sigma \rightarrow P(K)$

Automates finis non déterministes

- Une configuration est un élément de K $\times \Sigma^*$.

- Un automate M détermine une relation binaire $\vdash_M$ entre configurations définie par :
  - $\vdash_M \subset (K \times \Sigma^*)^2$
  - $(q, w) \vdash_M (q', w')$ ssi $\exists\, \sigma \in \Sigma$ tel que $w = \sigma w'$ et $(q, \sigma, q') \in \Delta$

Automates finis non déterministes

- $\vdash_M$ est une relation et non plus une fonction (automates déterministes)

  Pour une configuration (q, w), il peut y avoir plusieurs configurations (q', w') (ou aucune) tel que $(q, w) \vdash_M (q', w')$

- On note comme avant $\vdash_M^*$ la fermeture transitive réflexive de $\vdash_M$

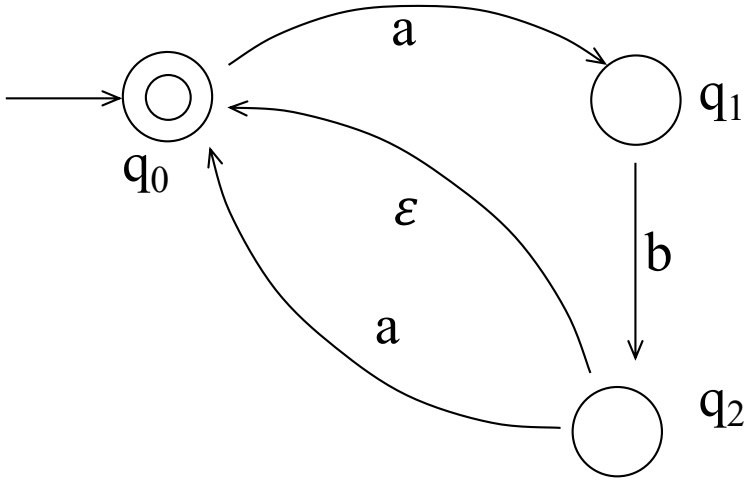
- Un mot w est accepté par M ssi $(s, w) \vdash_M^* (q, \varepsilon)$, avec $q \in F$

- $L(M)$ est le langage de tous les mots acceptés par M

Automates finis non déterministes avec transitions spontanées

- Ajout de transitions vides
  - Il est possible d'étiqueter une flèche par le symbole ε.

- Autre formulation encore plus intuitive (?) de l'automate précédent :



Automates finis non déterministes avec transitions spontanées

- Un automate non déterministe fini avec transitions spontanées est le quintuplet $M = (K, \Sigma, \Delta, s, F)$ où :
  - K : ensemble fini (non vide) d'états
  - Σ : alphabet (ensemble non vide de symboles)
  - Δ : relation de transition : $K \times (\Sigma \cup \{\varepsilon\}) \times K$

    $(q, \sigma, q') \in \Delta$: σ –transition ($\sigma \in \Sigma$)
  - s : état initial : $s \in K$
  - F : ensemble des états finaux (acceptants) : $F \subset K$

- hormis la relation, la définition est identique à la formulation déterministe et à la formulation non déterministe

Automates finis non déterministes avec transitions spontanées

- Si $(q, \varepsilon, q') \in \Delta$: on a une ε –transition (transition spontanée)
  → On passe de q à q' sans lire de symbole dans le mot courant.

- Une configuration est un élément de $K \times \Sigma^*$

- M décrit une relation binaire entre configurations qu'on note $\vdash_M$:

  $(q, w) \vdash_M (q', w')$

  ssi il existe un mot d'au plus une lettre $u \in \Sigma \cup \{\varepsilon\}$

  tel que $w = uw'$

  et $(q, u, q') \in \Delta$

Automates finis non déterministes avec transitions spontanées

- $\vdash_M$ est une relation et non plus une fonction (automates déterministes)
  - (q, ε) peut être en relation avec une autre configuration (après une ε –transition)
  - pour une configuration (q, w), il peut y avoir plusieurs configurations $\vdash_M$ (ou aucune) telles que $(q, w) \vdash_M (q', w')$

- On note comme avant $\vdash_M^*$ la fermeture transitive réflexive de $\vdash_M$

- Un mot w est accepté par M ssi $(s, w) \vdash_M^* (q, \varepsilon)$, avec $q \in F$

- $L(M)$ est le langage de tous les mots acceptés par M

Et maintenant ?

- Déterministe ou non déterministe ?
  - Même classe de langages reconnus
  - Donc équivalents

- Langages rationnels et langages reconnus par les automates finis ?
  - Même classe de langages

- Un langage est-il rationnel ?
  - Preuve de rationalité

- Un automate est-il minimal ?
  - Automate standard

Elimination du non-déterminisme

- Définition
  - 2 automates finis M et M' (déterministes ou non) sont équivalents
    ssi $\mathrm{L}(M) = L(M')$

- Théorème

  Pour tout automate non déterministe,

  il existe un automate déterministe équivalent,

  et il existe un algorithme pour le calculer

  Cet algorithme est appelé déterminisation d'un automate

Elimination du non-déterminisme
Preuve

- Soit $M = (K, \Sigma, \Delta, \mathrm{s}, \mathrm{F})$ un automate non déterministe

- Problème : déterminer $M' = (K', \Sigma, \delta', s', \mathrm{F}')$ déterministe

- Démarche
  1) Méthode pour construire M'
  2) Montrer
     - M' déterministe (par construction : trivial)
     - M' équivalent à M

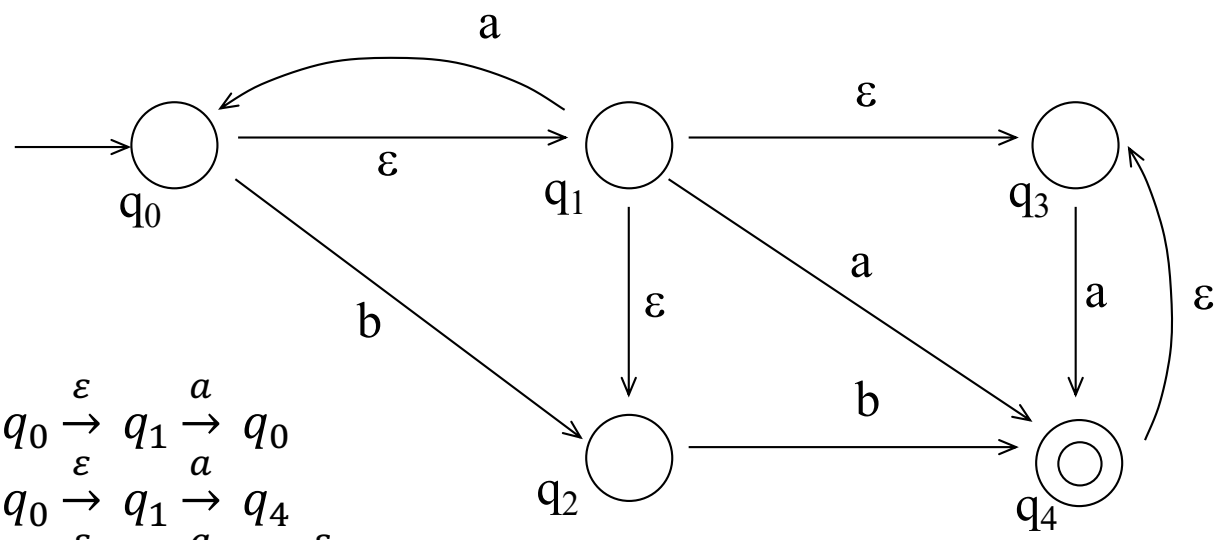
Elimination du non-déterminisme
Preuve

- Intuition
  - Pour toute lettre σ de Σ, on considère l'ensemble des états qu'on peut atteindre en lisant σ
  - On rassemble ces états et on considère des états étiquetés par des parties de K $(P(K))$
  - L'état initial de M' est l'ensemble des états atteignables en ne lisant aucune lettre
  - Les états finaux de M' sont les parties (atteignables) de $P(K)$ contenant au moins un état final de M

Elimination du non-déterminisme

Preuve

- Exemple



$q_0 \xrightarrow{a}$

q_0	$q_0 \xrightarrow{\varepsilon} q_1 \xrightarrow{a} q_0$
q_4	$q_0 \xrightarrow{\varepsilon} q_1 \xrightarrow{a} q_4$
q_3	$q_0 \xrightarrow{\varepsilon} q_1 \xrightarrow{a} q_4 \xrightarrow{\varepsilon} q_3$
q_1	$q_0 \xrightarrow{\varepsilon} q_1 \xrightarrow{a} q_0 \xrightarrow{\varepsilon} q_1$
q_2	$q_0 \xrightarrow{\varepsilon} q_1 \xrightarrow{a} q_0 \xrightarrow{\varepsilon} q_1 \xrightarrow{\varepsilon} q_2$

$q_0 \xrightarrow{b}$

q_2	$q_0 \xrightarrow{b} q_2$
q_4	$q_0 \xrightarrow{\varepsilon} q_1 \xrightarrow{\varepsilon} q_2 \xrightarrow{b} q_4$
q_3	$q_0 \xrightarrow{\varepsilon} q_1 \xrightarrow{\varepsilon} q_2 \xrightarrow{b} q_4 \xrightarrow{\varepsilon} q_3$

état initial : $\{q_0, q_1, q_2, q_3\}$

Elimination du non-déterminisme
Preuve

- Prise en compte des ε-transitions : ε-clôture
- Soit $q \in K$, on note $E(q)$ l'ensemble des états de M atteignables sans lire aucune lettre :

$$E(q) = \{p \in K : (q, \varepsilon) \vdash_M^* (p, \varepsilon)\}$$

$E(q)$ est la clôture de $\{q\}$ par la relation binaire $\{(p, r) \mid (p, \varepsilon, r) \in \Delta\}$

- Construction de $E(q)$

$E(q) := \{q\}$
tant que il existe une transition $(p, \varepsilon, r) \in \Delta$ avec $p \in E(q)$ et $r \notin E(q)$
$E(q) := E(q) \cup \{r\}$

Elimination du non-déterminisme
Preuve

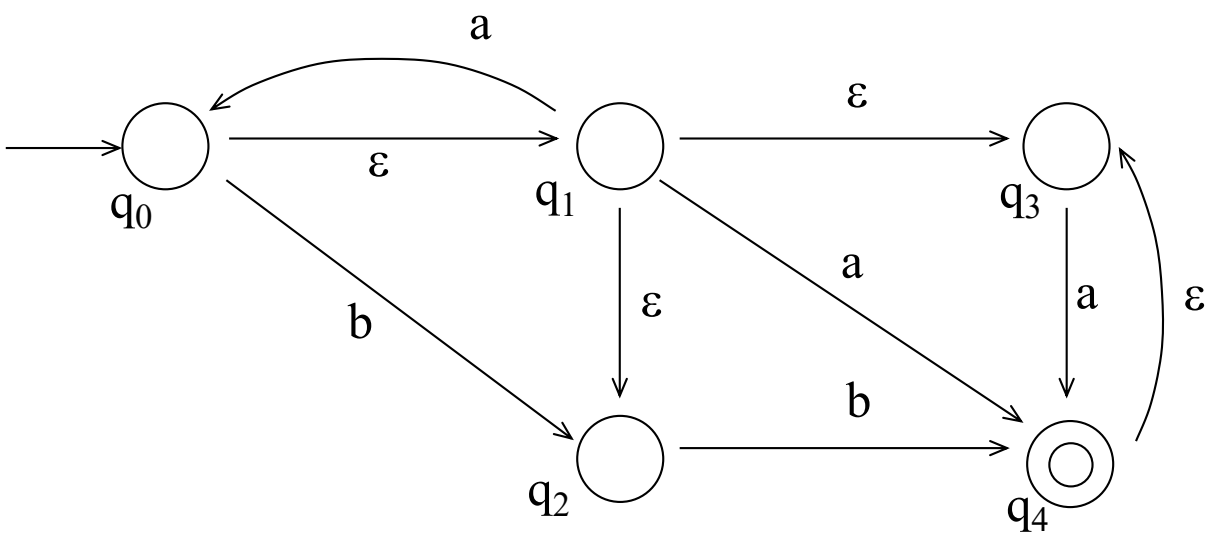
- On obtient donc $M' = (K', \Sigma, \delta', s', F')$ déterministe
  - $K' = P(K)$
  - $s' = E(s)$
  - $F' = \{Q \subset K : Q \cap F \neq \emptyset\}$
  - $\delta' = P(K) \times \Sigma \rightarrow P(K)$

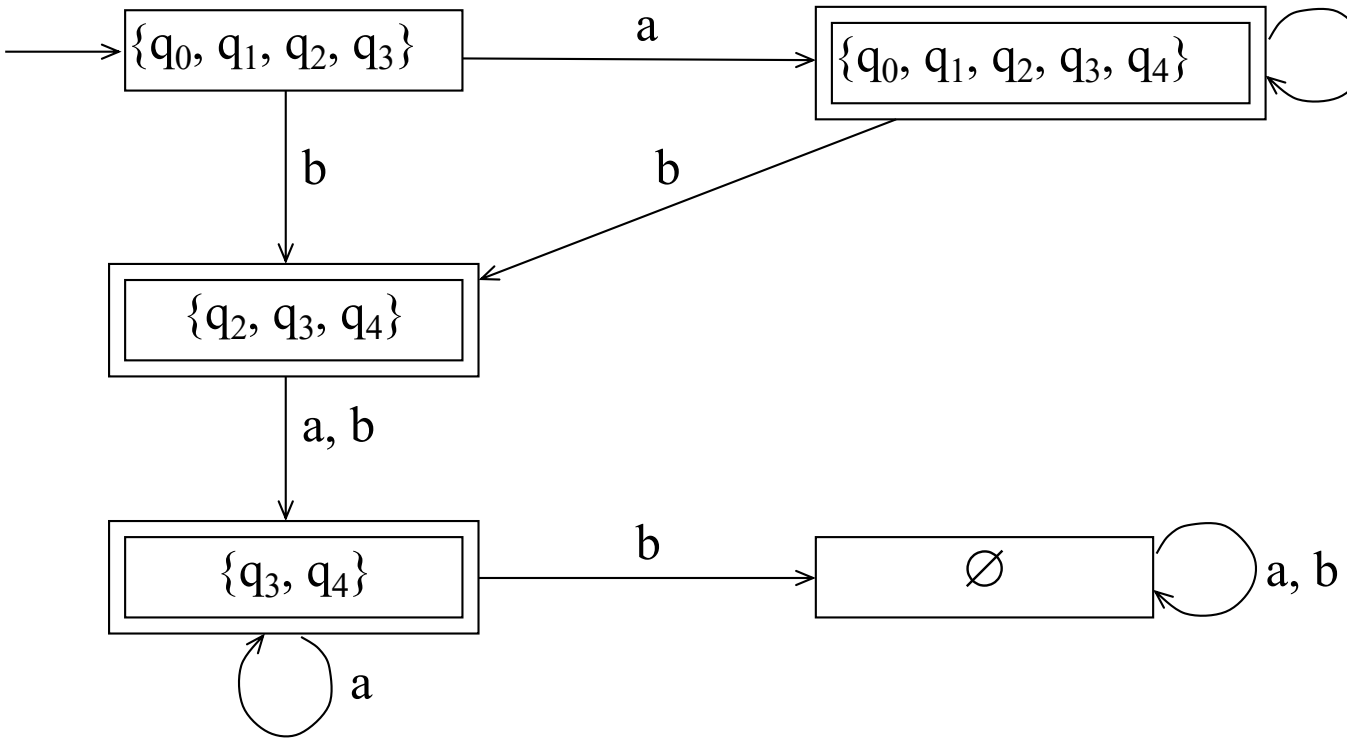
$$\forall Q \subset K, \forall a \in \Sigma, \quad \delta'(Q, a) = \cup \{E(p) \mid q \in Q : (q, a, p) \in \Delta\}$$

$\delta'(Q, a)$: ensemble de tous les états de M dans lesquels M peut aller en lisant a (y compris ε)

Elimination du non-déterminisme

Exemple



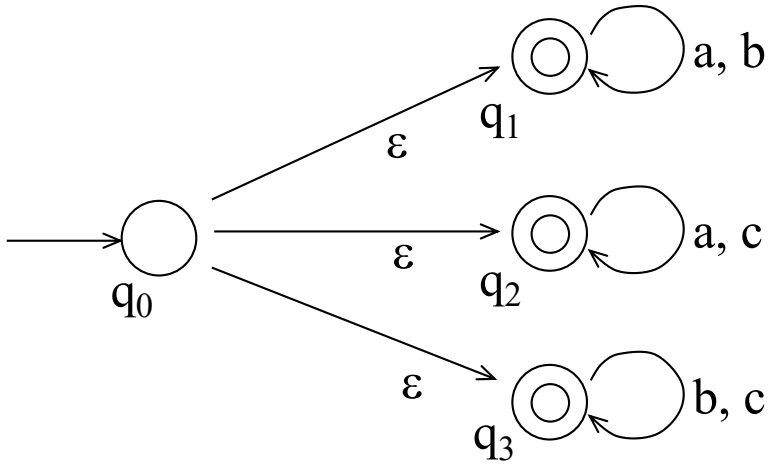


Elimination du non-déterminisme

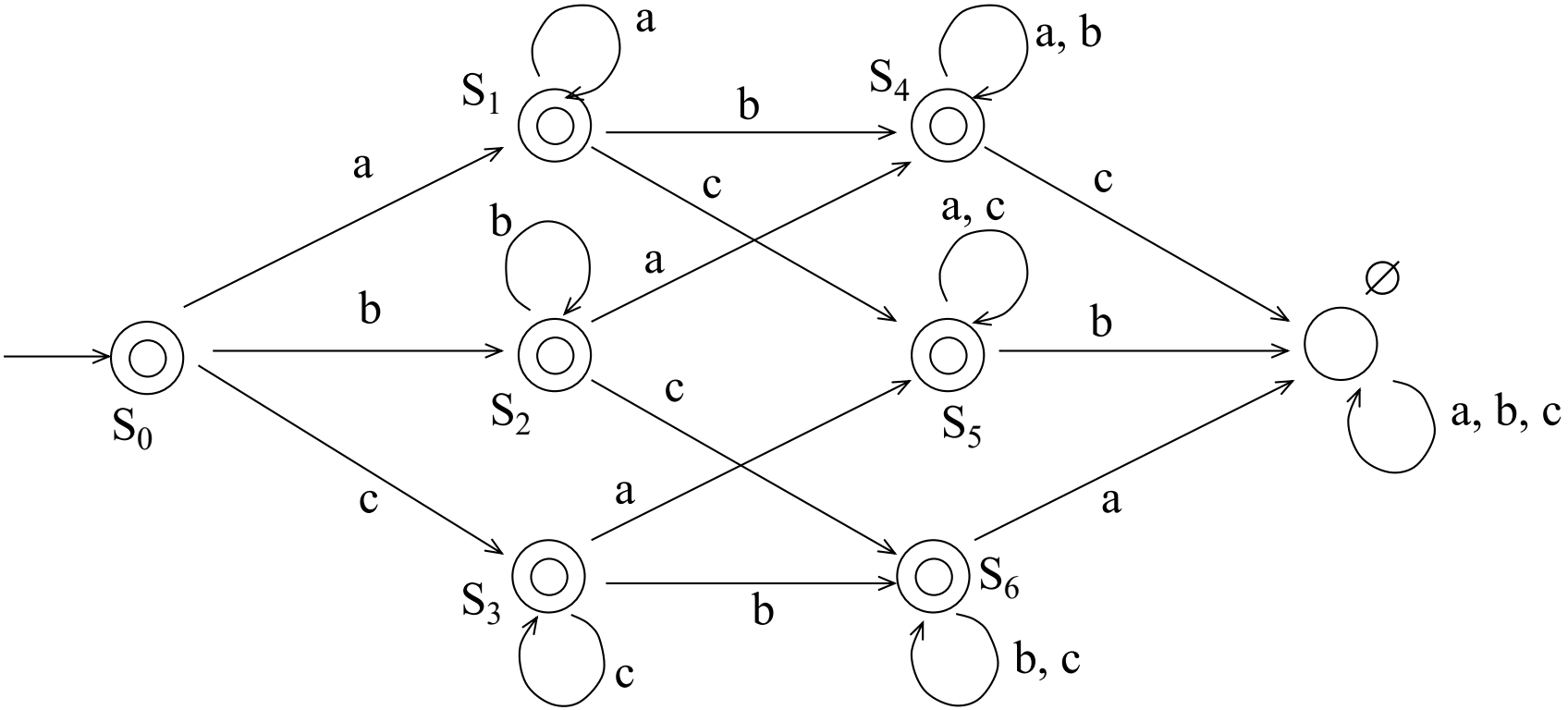
Autre exemple

$\Sigma = \{a, b, c\}$

$L(M) = (a \cup b)^* \cup (b \cup c)^* \cup (a \cup c)^*$



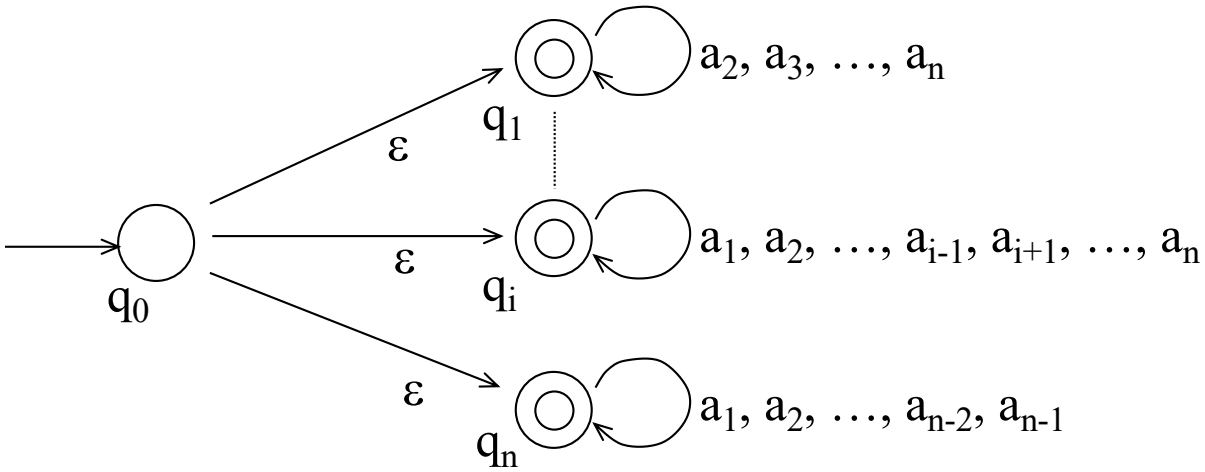
4 états



8 états

Elimination du non-déterminisme

Généralisation



$\Sigma = \{a_1, a_2, \ldots, an\}$

$\Sigma_1 = \Sigma - \{a_1\}$

…

$\Sigma_i = \Sigma - \{a_i\}$

→ $L(M) = \cup_{i=1}^{n} \Sigma_i^*$

Lien avec les langages rationnels
Stabilité

- Propriété de stabilité des langages reconnus par des automates finis.

- Théorème

  La classe des langages acceptés par les automates finis est stable par :

  - *Union*
  - *Concaténation*
  - *Fermeture itérative*
  - *Complément*
  - *Intersection*

  ordre de la démonstration

- Preuve constructive
  - Construction de l'automate pour chaque opération

Lien avec les langages rationnels
Stabilité

- Union

$L_1 = L(M_1)$ $\quad M_1 = (K_1, \Sigma, \Delta_1, s_1, F_1)$ → Même Σ

$L_2 = L(M_2)$ $\quad M_2 = (K_2, \Sigma, \Delta_2, s_2, F_2)$ → Hyp. $K_1 \cap K_2 = \varnothing$

Posons s tel que $s \notin K_1$ et $s \notin K_2$.

→ $M_\cup : (\{s\} \cup K_1 \cup K_2, \Sigma, \Delta_1 \cup \Delta_2 \cup \{(s, \varepsilon, s_1), (s, \varepsilon, s_2)\}, s, F_1 \cup F_2)$

$w \in L(M_\cup) \Leftrightarrow (s, w) \vdash_M (s_1, w) \vdash_M^* (f_1, \varepsilon)\ (f_1 \in F_1)$ ← L_1

ou

$(s, w) \vdash_M (s_2, w) \vdash_M^* (f_2, \varepsilon)\ (f_2 \in F_2)$ ← L_2

$\Leftrightarrow w \in L_1 \cup L_2$

→ $L(M_\cup) = L_1 \cup L_2$

Lien avec les langages rationnels
Stabilité

- Concaténation

$L_1 = L(M_1)$ $\quad M_1 = (K_1, \Sigma, \Delta_1, s_1, F_1)$ → Même Σ

$L_2 = L(M_2)$ $\quad M_2 = (K_2, \Sigma, \Delta_2, s_2, F_2)$ → Hyp. $K_1 \cap K_2 = \varnothing$

→ $M_c : \ (K_1 \cup K_2, \Sigma, \Delta_1 \cup \Delta_2 \cup \{(f_i, \varepsilon, s_2) \mid f_i \in F_1\}, s_1, F_2)$

→ $L(M_c) = L_1 L_2$

Lien avec les langages rationnels
Stabilité

- Etoile de Kleene

$$L_1 = L(M_1) \qquad M_1 = (K_1, \Sigma, \Delta_1, s_1, F_1)$$

Posons s tel que $s \notin K_1$

→ $M_K : (K_1 \cup \{s\}, \Sigma, \Delta_1 \cup \{(s, \varepsilon, s_1)\} \cup \{(f_i, \varepsilon, s_1) \mid f_i \in F_1\}, s, F_1 \cup \{s\})$

→ $L(M_K) = {L_1}^*$

Lien avec les langages rationnels
Stabilité

- Complément

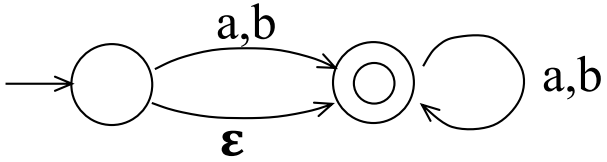
$L_1 = L(M_1) \qquad M_1 = (K_1, \Sigma, \Delta_1, s_1, F_1)$

→ $M_\neg : (K_1, \Sigma, \Delta_1, s_1, \neg F_1)\ avec\ \neg F_1 = K_1 - F_1$

→ $L(M_\neg) = \neg L_1$

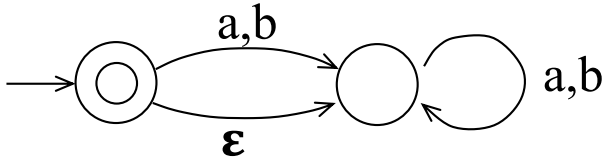
Attention M_1 doit être déterministe et complet

ex non det. :



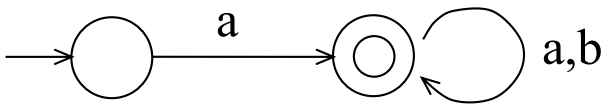
L(M) = (a ∪ b)*

➔



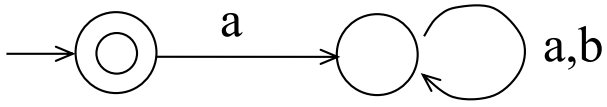
L(M¬) = {ε}

ex non complet :



L(M) = a (a ∪ b)*

➔



L(M¬) = {ε}

Lien avec les langages rationnels
Stabilité

- Intersection

$L_1 = L(M_1)$ $\quad M_1 = (K_1, \Sigma, \Delta_1, s_1, F_1)$ → Même Σ

$L_2 = L(M_2)$ $\quad M_2 = (K_2, \Sigma, \Delta_2, s_2, F_2)$ → Hyp. $K_1 \cap K_2 = \varnothing$

- Deux méthodes :

  - $L(M_1) \cap L(M_2) = \overline{\overline{L(M_1)} \cup \overline{L(M_2)}}$

  - Automate produit, avec M_1 et M_2 déterministes et complets
    - $M_\cap : (K_1 \times K_2, \Sigma, \{((p_1, p_2), \sigma, (q_1, q_2)) \mid (p_1, \sigma, q_1) \in \Delta_1 \; et \; (p_2, \sigma, q_2) \in \Delta_2\}, (s_1, s_2), F_1 \times F_2)$
    - Quadratique en le nombre d'états

→ $L(M_\cap) = L_1 \cap L_2$

Lien avec les langages rationnels
Stabilité

- Retour sur l'union

$L_1 = L(M_1)$ $\quad M_1 = (K_1, \Sigma, \Delta_1, s_1, F_1)$ → Même Σ

$L_2 = L(M_2)$ $\quad M_2 = (K_2, \Sigma, \Delta_2, s_2, F_2)$ → Hyp. $K_1 \cap K_2 = \varnothing$

Posons s tel que $s \notin K_1$ et $s \notin K_2$.

→ $M_\cup : (\{s\} \cup K_1 \cup K_2, \Sigma, \Delta_1 \cup \Delta_2 \cup \{(s, \varepsilon, s_1), (s, \varepsilon, s_2)\}, s, F_1 \cup F_2)$

- Autre méthode :
  - Automate produit, avec M_1 et M_2 déterministes et complets
    - $M_\cup : (K_1 \times K_2, \Sigma, \{((p_1, p_2), \sigma, (q_1, q_2)) \mid (p_1, \sigma, q_1) \in \Delta_1 \; et \; (p_2, \sigma, q_2) \in \Delta_2\}, (s_1, s_2), F_1 \times K_2 \cup K_1 \times F_2)$
    - Quadratique en le nombre d'états

→ $L(M_\cup) = L_1 \cup L_2$

Lien avec les langages rationnels
Stabilité

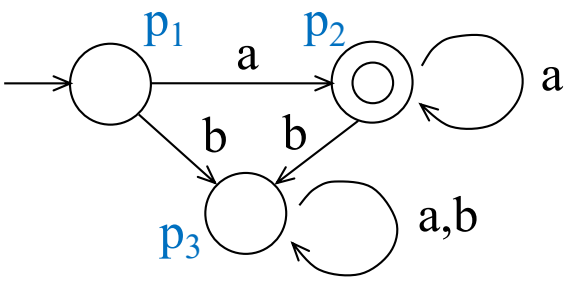
- Automate produit – intersection

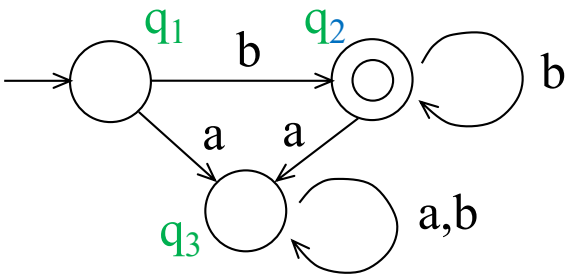
$L_1 = L(M_1)$ $\quad M_1 = (K_1, \Sigma, \Delta_1, s_1, F_1)$ → Même Σ

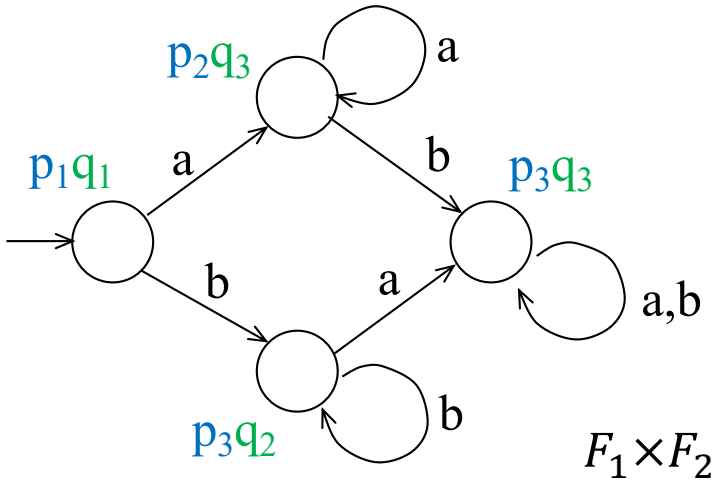
$L_2 = L(M_2)$ $\quad M_2 = (K_2, \Sigma, \Delta_2, s_2, F_2)$ → Hyp. $K_1 \cap K_2 = \varnothing$

– Automate produit, avec M_1 et M_2 déterministes et complets

- $M_\cap$: $(K_1 \times K_2, \Sigma, \{((p_1, p_2), \sigma, (q_1, q_2)) \mid (p_1, \sigma, q_1) \in \Delta_1 \; et \; (p_2, \sigma, q_2) \in \Delta_2\}, (s_1, s_2), F_1 \times F_2)$







$F_1 \times F_2 = \emptyset$

Lien avec les langages rationnels
Stabilité

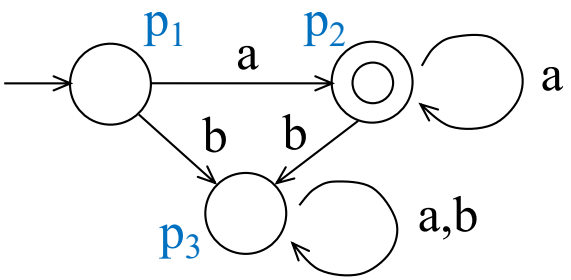
- Automate produit – union

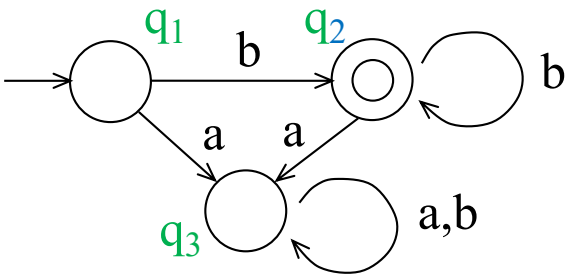
$L_1 = L(M_1)$ $\quad M_1 = (K_1, \Sigma, \Delta_1, s_1, F_1)$ → Même Σ

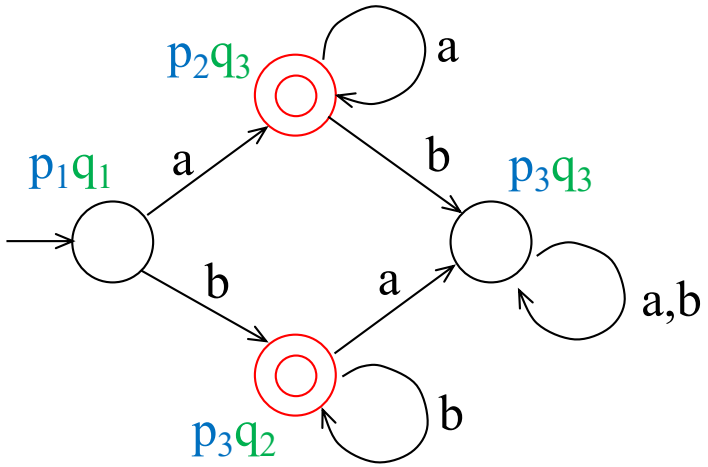
$L_2 = L(M_2)$ $\quad M_2 = (K_2, \Sigma, \Delta_2, s_2, F_2)$ → Hyp. $K_1 \cap K_2 = \varnothing$

– Automate produit, avec M_1 et M_2 déterministes et complets

- $M_\cup : (K_1 \times K_2, \Sigma, \{((p_1, p_2), \sigma, (q_1, q_2)) \mid (p_1, \sigma, q_1) \in \Delta_1 \; et \; (p_2, \sigma, q_2) \in \Delta_2\}, (s_1, s_2), F_1 \times K_2 \cup K_1 \times F_2)$







Lien avec les langages rationnels

Inclusion des classes de langages

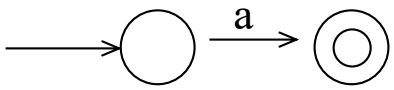
- Théorème

  La classe des langages acceptés par les automates finis contient les langages rationnels.

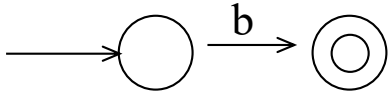
- Exemple

  $(ab \cup aba)^*$

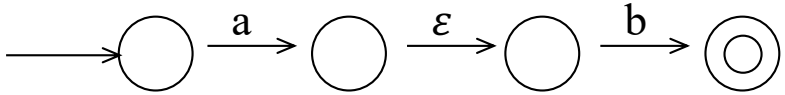
a



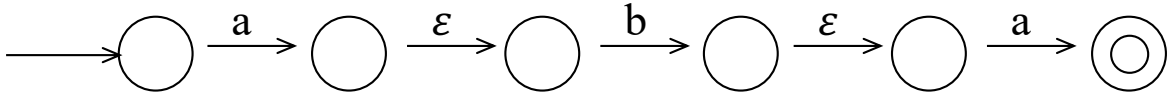
b



ab



aba

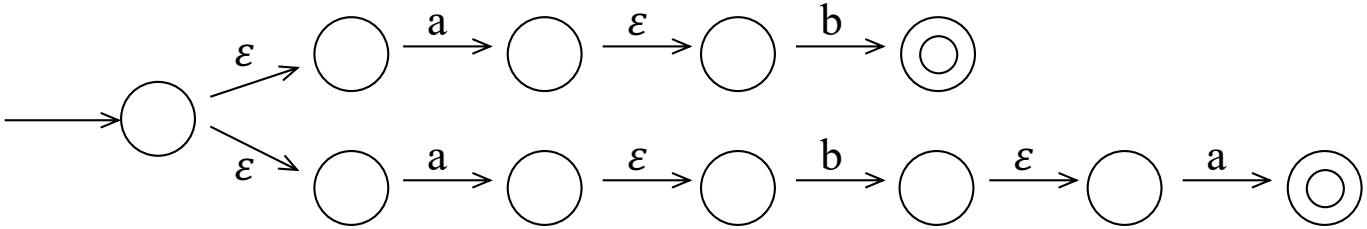


Lien avec les langages rationnels
Inclusion des classes de langages

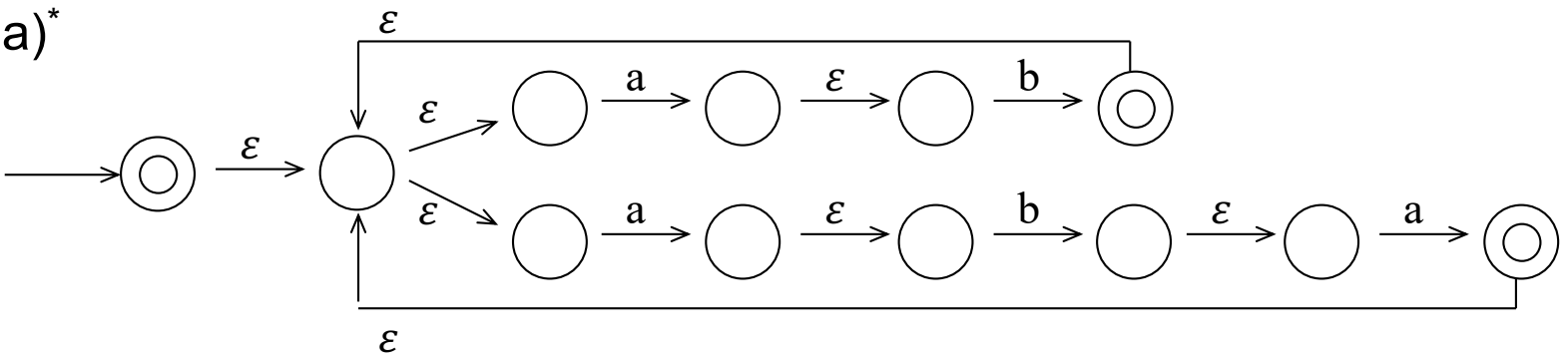
- Exemple

  $(ab \cup aba)^*$

  $ab \cup aba$



  $(ab \cup aba)^*$



Lien avec les langages rationnels

Caractérisation des langages rationnels

- Théorème

  Un langage est rationnel ssi il est accepté par un automate fini.

- Preuve

  On suppose qu'on a numéroté (ordonné) les états.

  Soit $M = (K, \Sigma, \Delta\, s, F)$ un automate fini (déterministe ou non). $|K| = n$.

  $$K = \{q_1, q_2, \dots, q_n\}, \quad s = q_1$$

  Le langage reconnu par M est la réunion de tous les langages reconnus en parcourant tous les chemins possibles dans le graphe.

  → À chaque chemin allant de s à f $(\in F)$, on associe le langage trouvé.

Lien avec les langages rationnels

Caractérisation des langages rationnels

- On pose $R(i,j,k)$ = l'ensemble des mots obtenus par lecture de l'automate M
  - en partant de l'état q_i,
  - en arrivant dans l'état q_j (avec le mot vide),
  - en ne passant que par des états intermédiaires dont le numéro est $\leq k$.

- $R(i,j,k)$ est un langage

- $R(i,j,k) = \{w \mid (q_i, w) \vdash_M^* (q_j, \varepsilon)$ *sans passer par des états intermédiaires dont le numéro est* $> k\}$

$$R(i,j,n) = \{w \mid (q_i, w) \vdash_M^* (q_j, \varepsilon)\}$$

$$L(M) = \bigcup_{i \mid q_i \in F} R(1,i,n)$$

Lien avec les langages rationnels

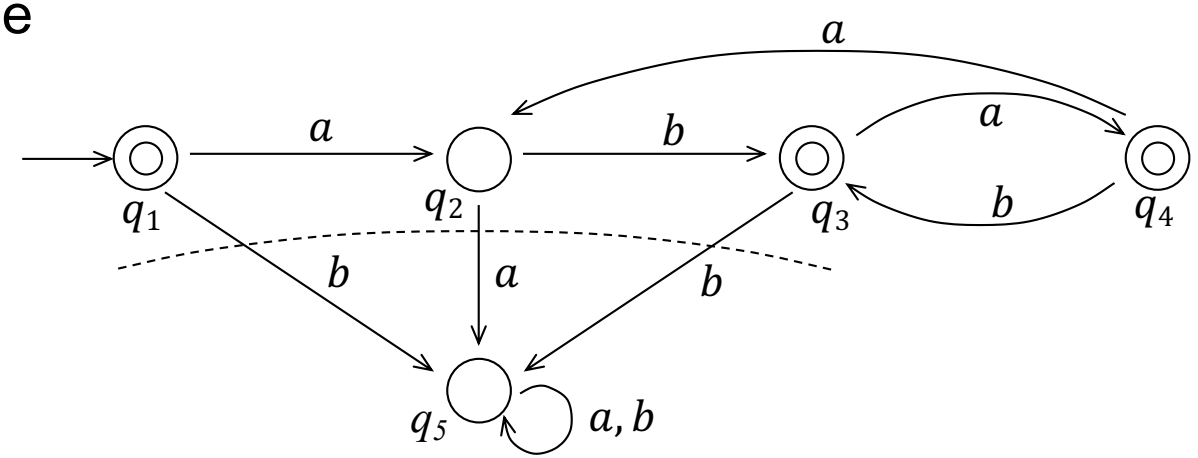
Caractérisation des langages rationnels

- $R(i,j,k)$ est un langage rationnel dont on peut calculer la représentation par récurrence sur k.

- Preuve

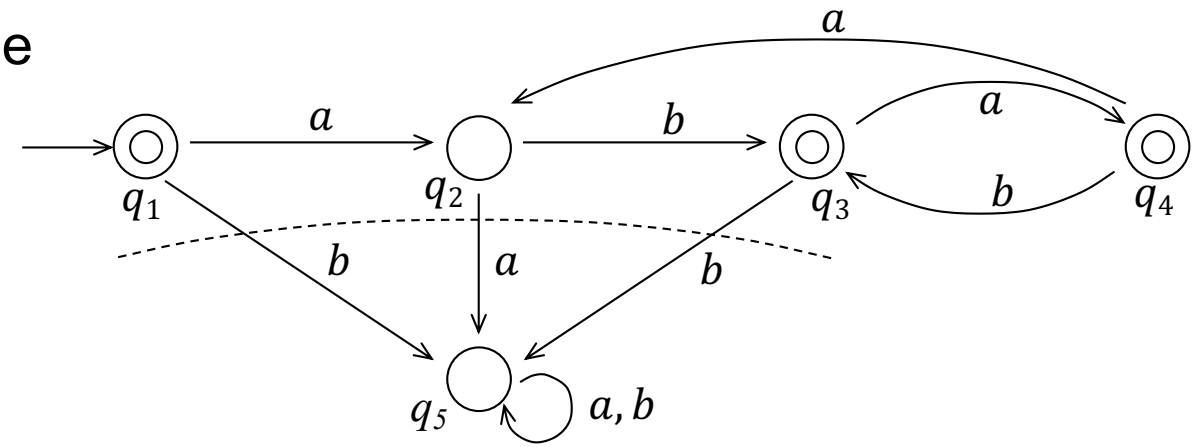
$$R(i,j,k) = R(i,j,k-1) \cup R(i,k,k-1).\left(R(k,k,k-1)\right)^*.R(k,j,k-1)$$

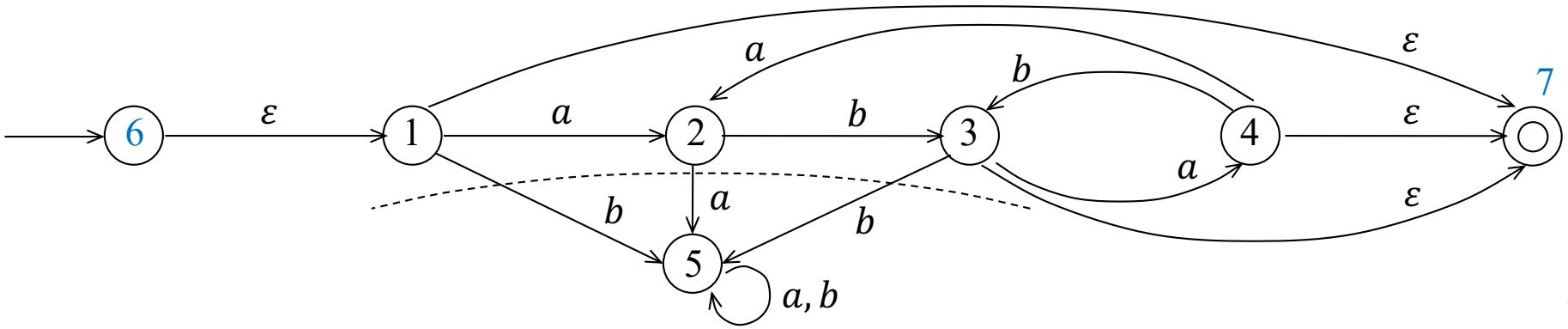
- Exemple



Lien avec les langages rationnels
Caractérisation des langages rationnels

- Forme spéciale de M ($L(M)$ inchangé) :
  - Un seul état final f
  - Si $(p, \sigma, q) \in \Delta$, alors $p \neq f$ et $q \neq s$ (pas de retour de f ou vers s)
- $s = q_{n-1}$ et $f = q_n$, alors $L(M) = R(n-1, n, n)$
- Exemple





Lien avec les langages rationnels
Caractérisation des langages rationnels

- Exemple
  - $R(i,j,0)$ → étiquettes sur les flèches.
  - Principe : calculer $R(6,7,7)$

$R(6,7,7)$ = $R(6,7,6) \cup R(6,6,6)\, R(6,6,6)^* \, R(6,7,6)$

= $R(6,7,5) \cup R(6,5,5)\, R(5,5,5)^* \, R(5,7,5) \cup \ldots$

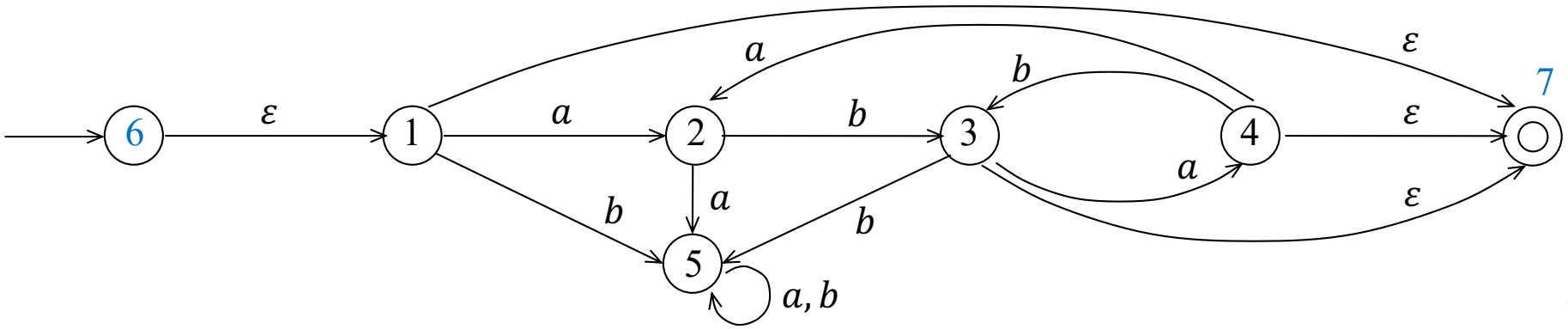
= …

Fastidieux ⇒ on calcule les $R(i,j,k)$ de proche en proche.

→ On supprime q_1 (ce qui revient à calculer $R(i,j,1)$)

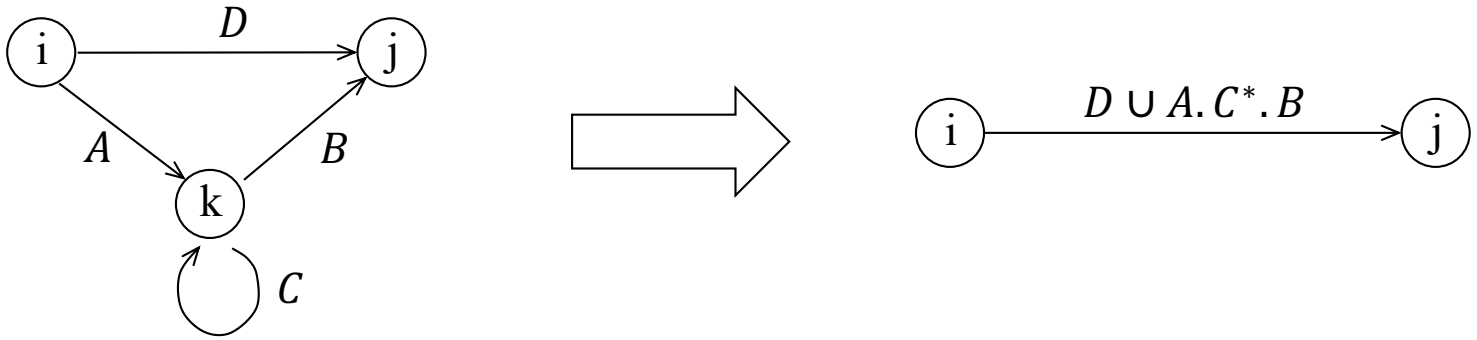
→ On supprime q_2 (ce qui revient à calculer $R(i,j,2)$)

…



Lien avec les langages rationnels
Caractérisation des langages rationnels

- Règles de suppression d'un état q_k
  - Pour chaque paire d'états $q_i \neq q_k$ et $q_j \neq q_k$ pour lesquels il y a une flèche de q_i à q_k étiquetée par A et une flèche de q_k à q_j étiquetée par B
    - On ajoute une flèche de q_i à q_j étiquetée par $A.C^*.B$, où C est l'étiquette de la flèche de q_k à q_k ; s'il n'y a pas de flèche de q_k à q_k, $C = \emptyset$, donc $C^* = \{\varepsilon\}$, la flèche de q_i à q_j est donc étiquetée par $A.B$
    - S'il y a déjà une flèche de q_i à q_j étiquetée par D, l'étiquette devient $D \cup A.C^*.B$
  - Pour chaque paire d'états $q_i \neq q_k$ et $q_j \neq q_k$ pour lesquels il y a une flèche de q_i à q_j étiquetée par D et pas de flèche de q_i à q_k ou de q_k à q_j
    - L'étiquette de q_i à q_j reste D
  - On supprime q_k et toutes les flèches entrantes et sortantes

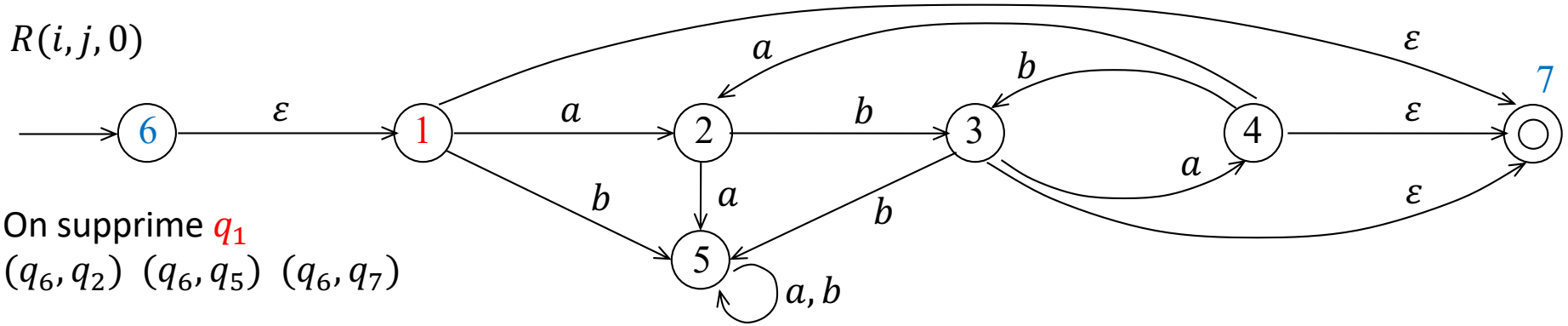


Lien avec les langages rationnels

Caractérisation des langages rationnels

- Exemple

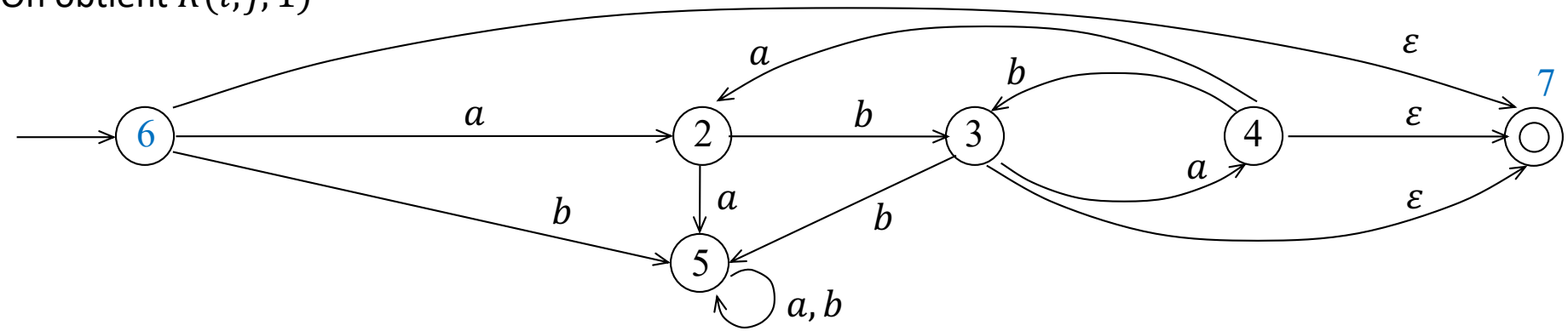
$R(i,j,0)$



On supprime q_1

(q_6, q_2) (q_6, q_5) (q_6, q_7)

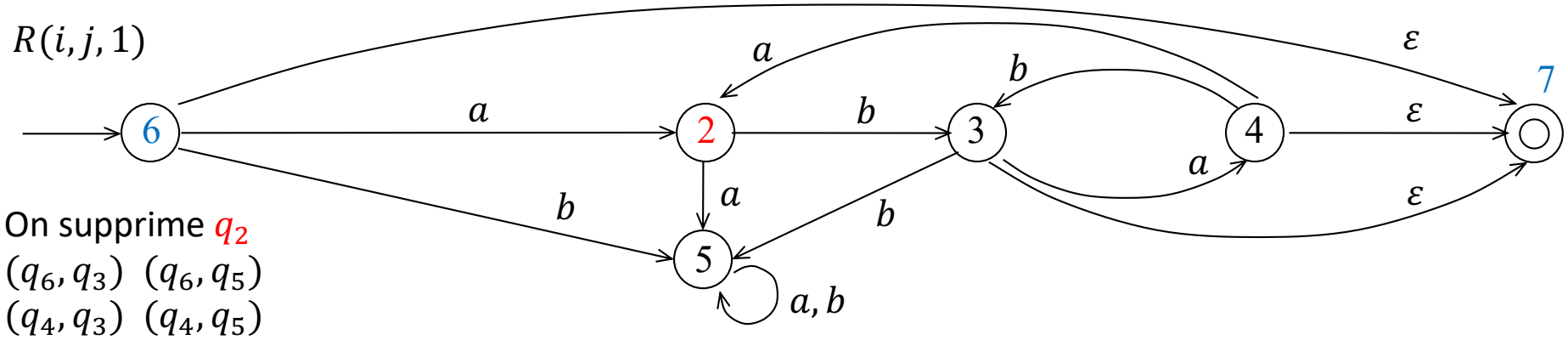
On obtient $R(i,j,1)$



Lien avec les langages rationnels

Caractérisation des langages rationnels

- Exemple

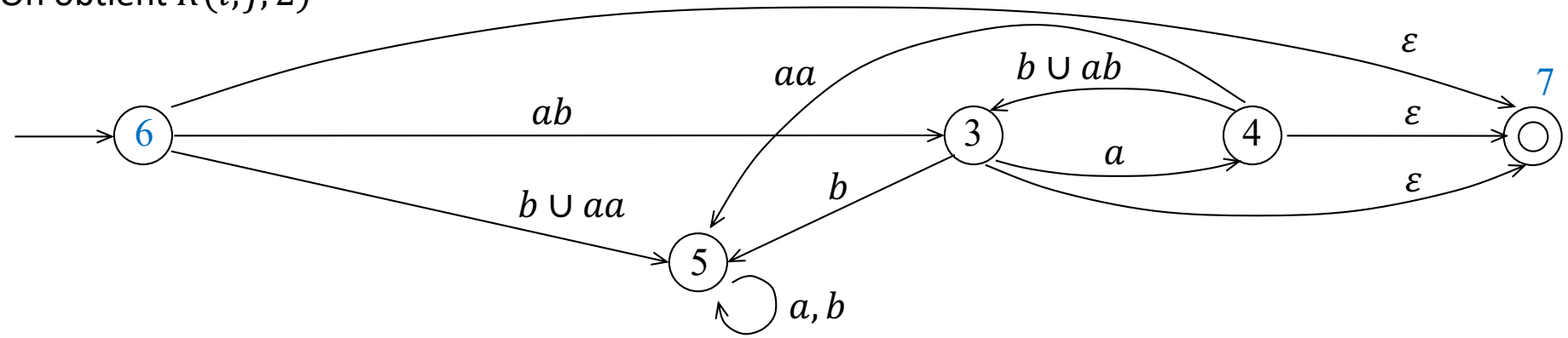


On supprime q_2

(q_6, q_3) (q_6, q_5)

(q_4, q_3) (q_4, q_5)

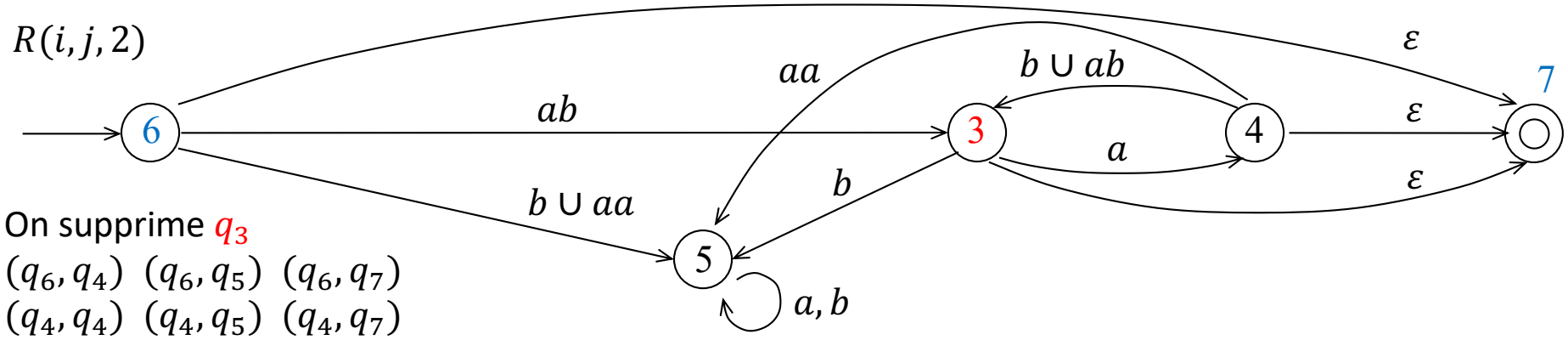
On obtient $R(i,j,2)$



Lien avec les langages rationnels

Caractérisation des langages rationnels

- Exemple

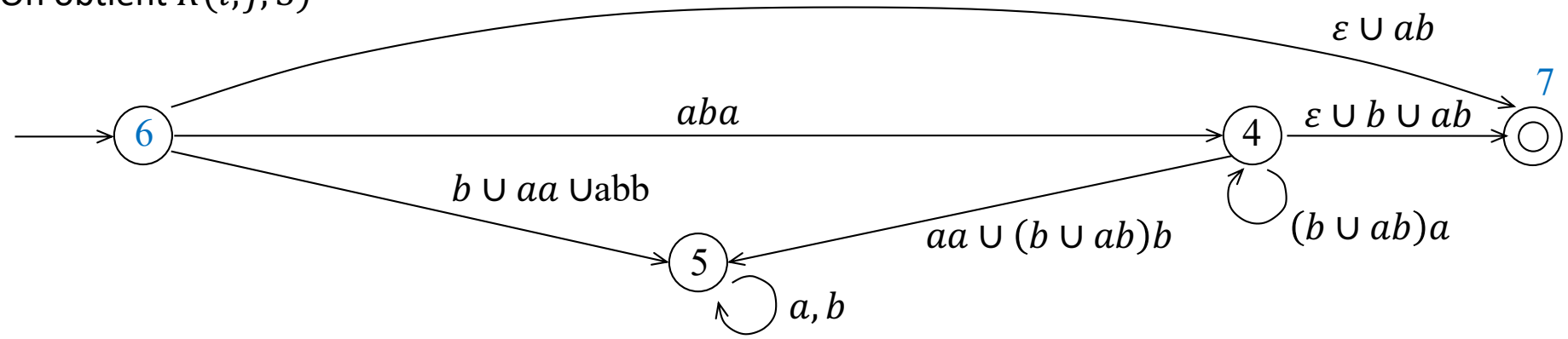


On supprime q_3

(q_6, q_4) (q_6, q_5) (q_6, q_7)

(q_4, q_4) (q_4, q_5) (q_4, q_7)

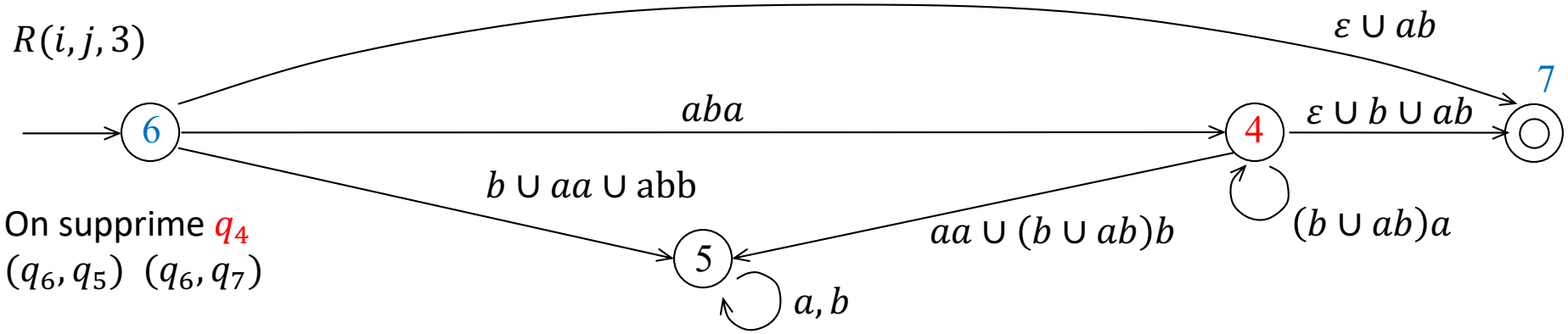
On obtient $R(i,j,3)$



Lien avec les langages rationnels

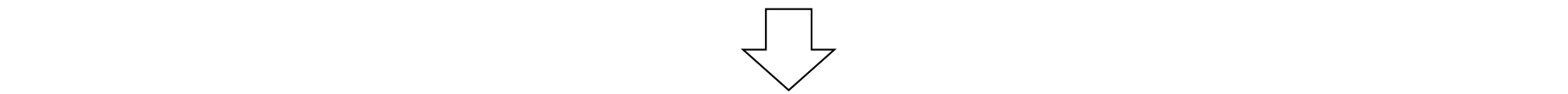
Caractérisation des langages rationnels

- Exemple

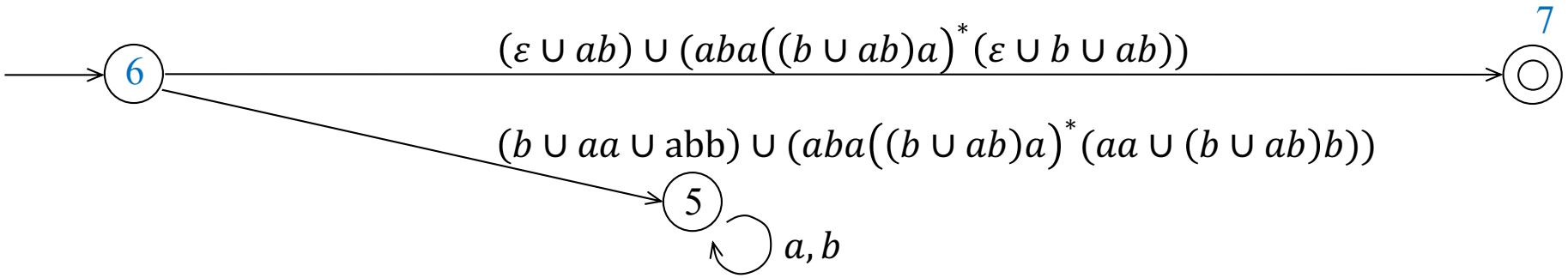


On supprime q_4

(q_6, q_5) (q_6, q_7)



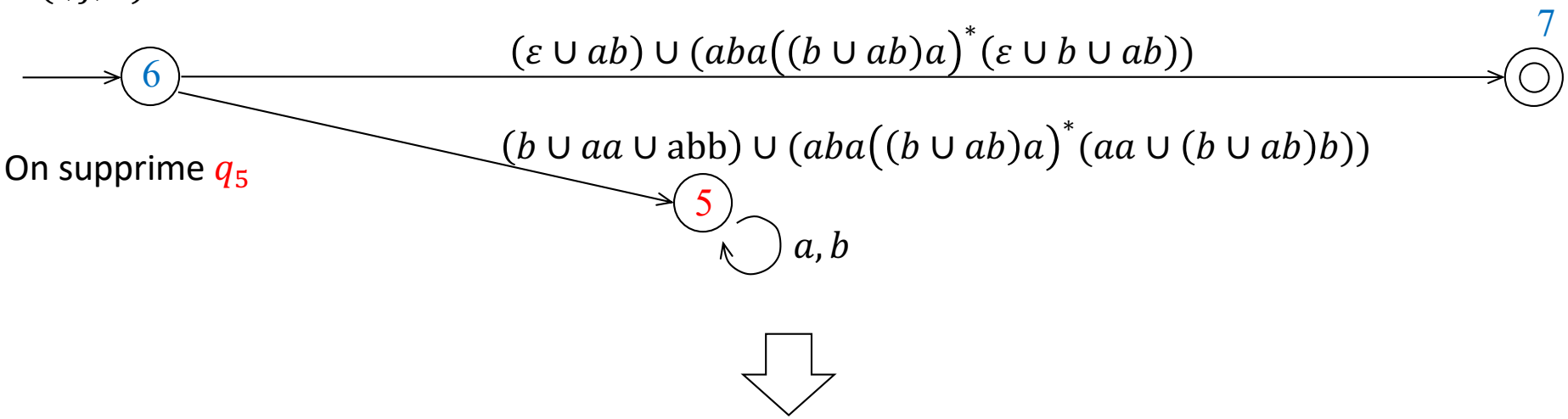
On obtient $R(i,j,4)$



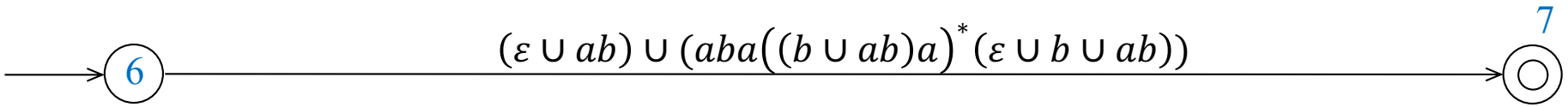
Lien avec les langages rationnels
Caractérisation des langages rationnels

- Exemple

$R(i,j,4)$



On obtient $R(i,j,5)$



Donc $R(6,7,5) = R(6,7,7) = L(M) = \varepsilon \cup ab \cup aba(ba \cup aba)^*(\varepsilon \cup b \cup ab)$

Minimisation des états

Théorème de Myhill – Nerode

Contexte à droite relativement à un langage

- Définition

  Soit $L \subseteq \Sigma^*$ un langage

  On définit pour un mot $u \in \Sigma^*$ son contexte à droite relativement à L :

$$R_L(u) = \{ z \in \Sigma^* \mid uz \in L \}$$

Minimisation des états

Théorème de Myhill – Nerode

Equivalence des mots suivant un langage

- Définition

  Soit $L \subseteq \Sigma^*$ un langage et $x, y \in \Sigma^*$ deux mots.

  On dit que x et y sont équivalents suivant L, et on note $x \approx_L y$,

  si pour tout mot z de Σ^* :

  $xz \in L$ ssi $yz \in L$

- Propriété

  $\approx_L$ est une relation d'équivalence

  On note $[w]_L$ la classe d'équivalence du mot w.

Minimisation des états

Théorème de Myhill – Nerode

Equivalence des mots suivant un langage

- Exemple $L = (ab \cup ba)^*$

  Chercher les classes suivant $\approx_L$ dans Σ^*

  - $[\varepsilon] = L$
  - $[a] = La$
  - $[b] = Lb$
  - $[aa] = L\,(aa \cup bb)\,\Sigma^*$
  - $[bb] = [aa]$
  - $[bba] = [aa]$
  - $[aaa] = [aa]$
  - $[ab] = L = [ba] = [\varepsilon]$ car $ab \in L$

  ...

  On montre facilement par récurrence qu'on a toutes les classes

  - On obtient 4 classes d'équivalence :

    $\Sigma^* = L \cup La \cup Lb \cup L\,(aa \cup bb)\,\Sigma^*$

Minimisation des états

Théorème de Myhill – Nerode

Equivalence des mots suivant un automate

- Définition

  Soit M = (K, Σ, δ, s, F) un automate déterministe (fini), et x, y $\in \Sigma^*$ deux mots.

  On dit que x et y sont équivalents relativement à M, on note $x \sim_M y$,

  ssi il existe un état q de K tel que :

  $$(s, x) \vdash_M^* (q, \varepsilon) \text{ et } (s, y) \vdash_M^* (q, \varepsilon)$$

- M étant déterministe, si on note $q_M(x)$ l'état auquel on parvient dans M en lisant x, on a :

  $$x \sim_M y \text{ ssi } q_M(x) = q_M(y)$$

Minimisation des états

Théorème de Myhill – Nerode

Equivalence des mots suivant un automate

- Propriétés
  - $\sim_M$ est une relation d'équivalence.
  - On note E_q la classe d'équivalence des mots x tels que $q_M(x) = q$

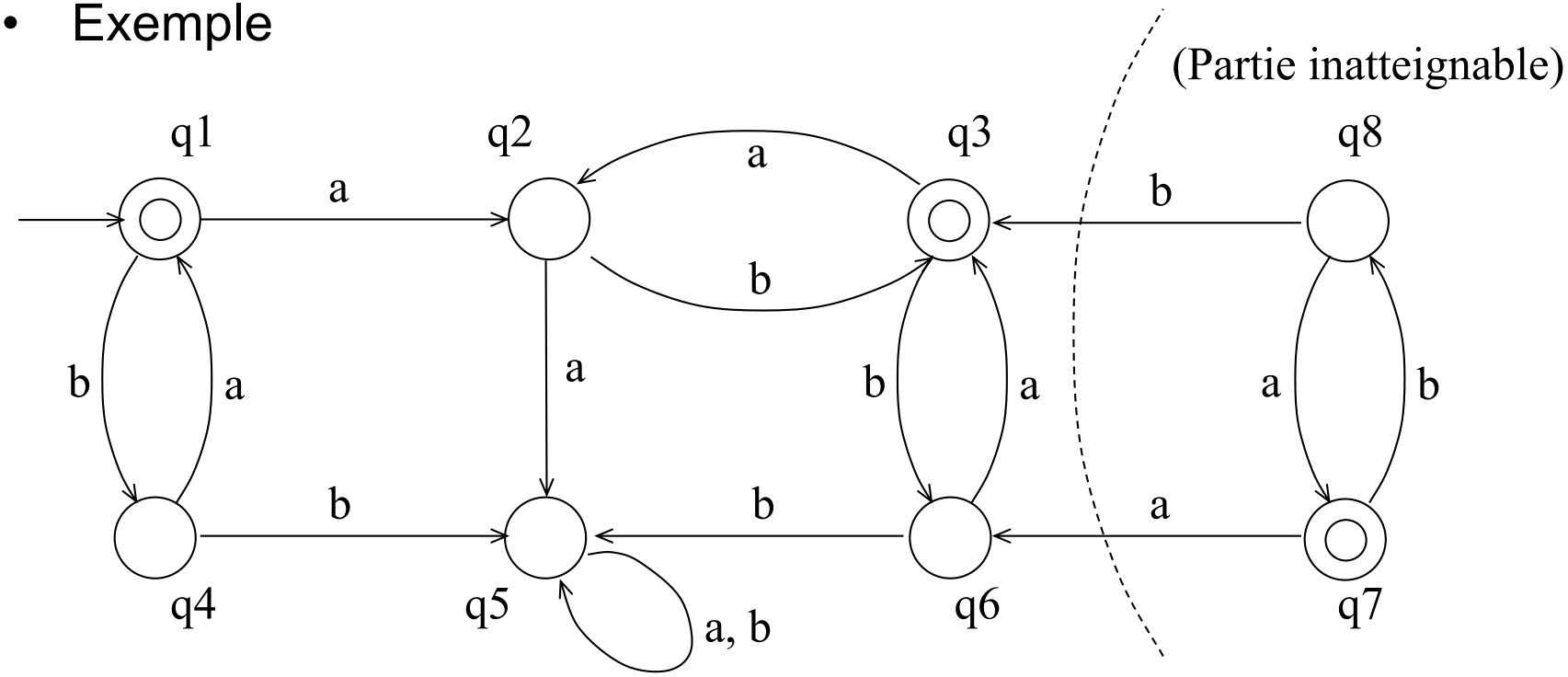
    $E_q = \varnothing$ si l'état est inatteignable
  - Il y a autant de classes d'équivalence que d'états atteignables (accessibles).

Minimisation des états

Théorème de Myhill – Nerode

Equivalence des mots suivant un automate

- Exemple



[ε] = $E_{q1} \cup E_{q3}$
[a] = E_{q2}
[b] = $E_{q4} \cup E_{q6}$
[aa] = E_{q5}

Minimisation des états

Théorème de Myhill – Nerode

Equivalence des mots suivant un automate

- $E_q = L(M^q)$ avec $M^q = (K, \Sigma, \delta, s, \{q\})$
- Théorème

Pour tout automate fini déterministe M et deux mots x, y $\in \Sigma^*$, on a :

si $x \sim_M y$ alors $x \approx_{L(M)} y$

(on dit que $\sim_M$ raffine $\approx_{L(M)}$)

Les classes de $\sim_M$ sont plus petites (et incluses) dans les classes de $\approx_{L(M)}$

Minimisation des états
Théorème de Myhill – Nerode

Automate standard

- Théorème de Myhill – Nerode

Soit $L \subseteq \Sigma^*$ un langage rationnel.
Il existe un automate déterministe ayant $|\ \Sigma^* / \approx_L\ |$ états acceptant L.
(C'est-à-dire autant d'états que le nombre de classes d'équivalence suivant $\approx_L$.)

- Cet automate a le plus petit nombre d'états possibles
- Il n'y a qu'un seul automate vérifiant cela
- On appelle cet automate l'automate standard de L
  (ou automate minimal de L).

Minimisation des états

Théorème de Myhill – Nerode

Automate standard

- Preuve (constructive) :

M = (K, Σ, δ, s, F) automate standard d'un langage L, avec

- K = { [x], x $\in$ Σ^* }
- s = [ε]
- F = { [x], x $\in$ L }
- δ : définie par δ([x], a) = [xa]

Minimisation des états

Théorème de Myhill – Nerode

Automate standard

- K est fini car L est reconnu par un automate déterministe M'

  $| \Sigma^* / \sim_{M'} | \geq | \Sigma^* / \approx_L |$

- δ bien définie : si [x] = [y] alors [xa] = [ya]

  (clair car : $x \approx_L y \Rightarrow xa \approx_L ya$)

- L = L(M)

  (i) on montre d'abord que $([x], y) \vdash_M^* ([xy], \varepsilon)$ (par induction sur |y|)

  (ii) $\forall x \in \Sigma^*$, $x \in L(M)$ ssi $([\varepsilon], x) \vdash_M^* ([x], \varepsilon)$ et $[x] \in F$ ie $x \in L$

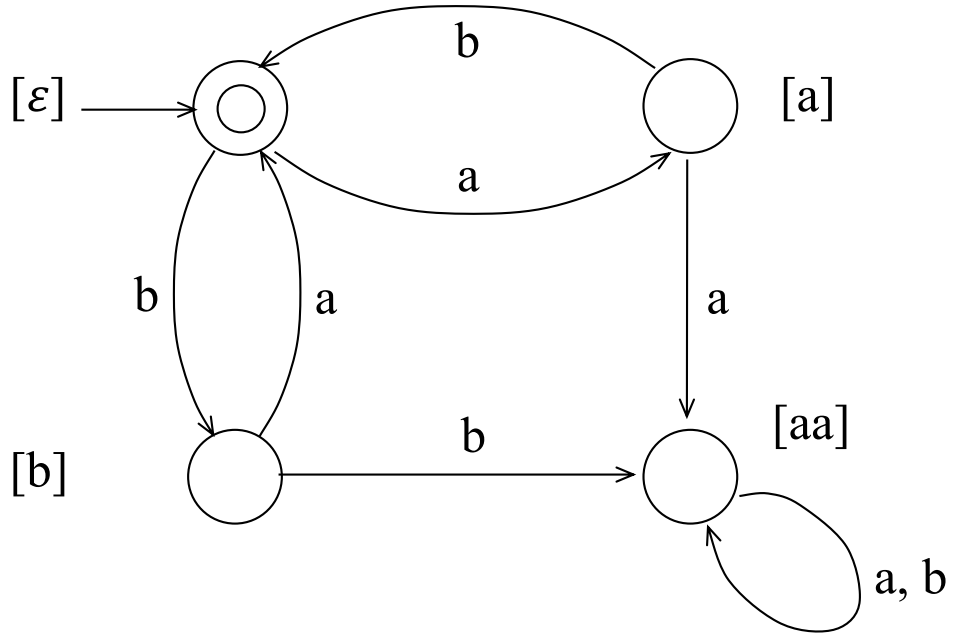
  (par définition de F)

Minimisation des états

Théorème de Myhill – Nerode

Automate standard

- Exemple



Minimisation des états
Théorème de Myhill – Nerode

Corollaire (Myhill – Nerode)

- Théorème

  L est rationnel ssi $\approx_L$ a un nombre fini de classes d'équivalence.

Minimisation des états

Minimisation d'un automate donné

- Définition

  Soit M = (K, Σ, δ, s, F) un automate fini déterministe.

  On note M(q), pour q $\in$ K, l'automate (K, Σ, δ, q, F).

  (Avec cette notation, M = M(s).)

  On dit que p et q sont deux états équivalents de M, et on note p $\equiv$ q,

  ssi L(M(p)) = L(M(q)).

- p $\equiv$ q ssi $\forall$ w $\in \Sigma^*$,
  - soit (p, w) $\vdash_M^*$ (f_1, ε) et (q, w) $\vdash_M^*$ (f_2, ε) avec f_1, $f_2 \in$ F
  - soit (p, w) $\vdash_M^*$ (g_1, ε) et (q, w) $\vdash_M^*$ (g_2, ε) avec g_1, $g_2 \notin$ F

  classe d'équivalence des mots x tels que $q_M(x)$ = p

Minimisation des états

Minimisation d'un automate donné

- Propriété

  $M = (K, \Sigma, \delta, s, F)$ sans états inatteignables ($E_p \neq \varnothing \ \forall \ p \in K$).

  Soient p et q $\in$ K. $\quad (\exists\ v \in \Sigma^* \mid E_p \subset [v] \text{ et } E_q \subset [v]) \Leftrightarrow p \equiv q$

- Corollaire

  $\equiv$ sur K induit une relation d'équivalence sur $\Sigma^* / \sim_M$ (notée $\approx$)
  définie par $E_p \approx E_q$ ssi $p \equiv q$

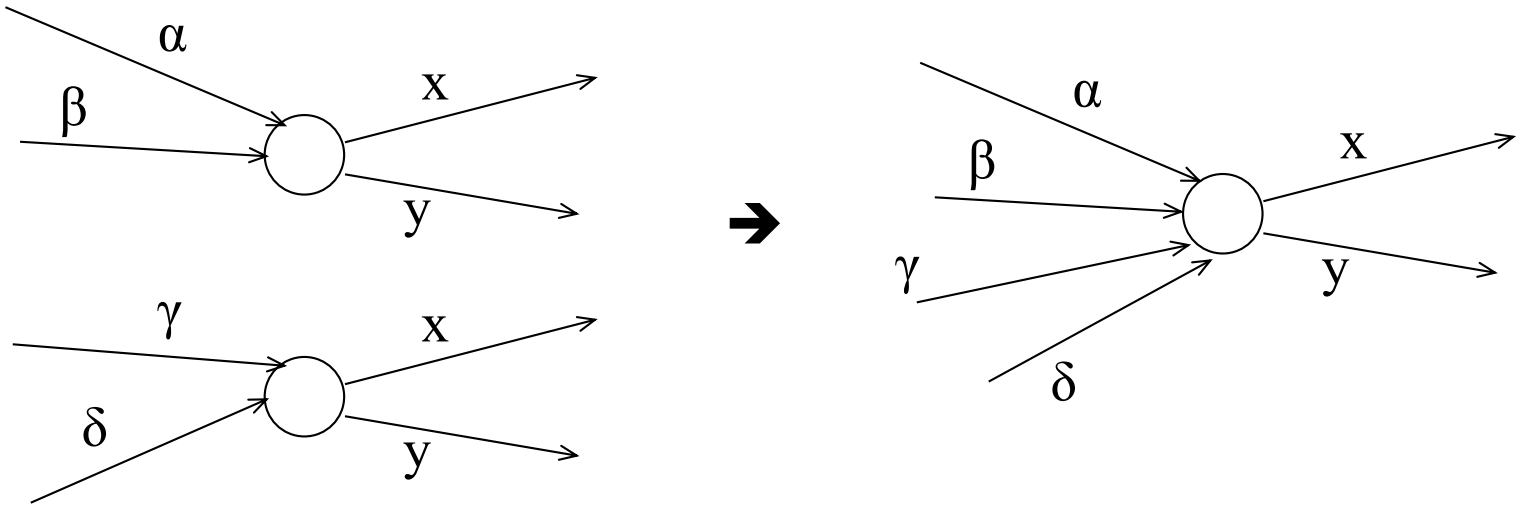
  Chaque classe d'équivalence correspond à une classe de la forme [v].

Minimisation des états

Minimisation d'un automate donné

Deux états équivalents sont ceux qu'on doit fusionner pour obtenir l'automate (minimal) standard.

On garde les flèches entrantes et on oublie les flèches sortantes de l'un des états (qui sont étiquetées par toutes les lettres de Σ)



Minimisation des états

Minimisation d'un automate donné

- Concrètement

  On calcule $\equiv$ puis on fait la fusion.

  $\equiv$ est calculée comme la limite de $(\equiv_i)_{i \in N}$.

  $\equiv_i$ définie par :

  $p \equiv_i q$ ssi pour tout mot w de longueur $\leq i$ tel que
  $(p, w) \vdash_M^* (f_1, \varepsilon)$ et $(q, w) \vdash_M^* (f_2, \varepsilon)$

  on a $f_1 \in F \Leftrightarrow f_2 \in F$

  (ou $L(M(p)) \cap (\cup_{k=0}^{i} \Sigma^k) = L(M(q)) \cap (\cup_{k=0}^{i} \Sigma^k)$)

$\forall i \in N$: $p \equiv q \Leftrightarrow p \equiv_i q$

et $p \equiv_i q \Leftrightarrow p \equiv_{i-1} q$

Minimisation des états

Minimisation d'un automate donné

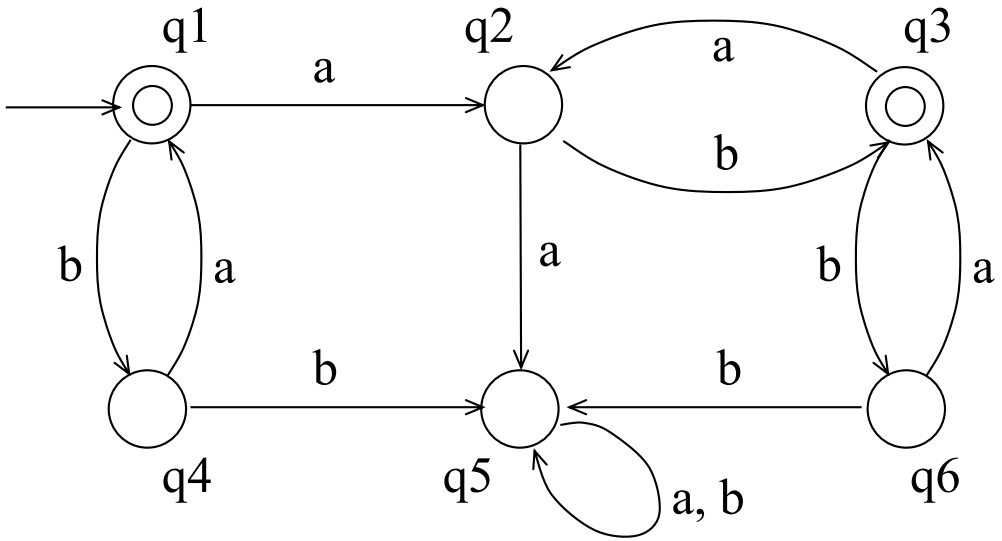
- Propriété

  Pour tout couple $(p, q) \in K^2$ et $n \geq 1$ on a :

  $p \equiv_n q$ ssi $p \equiv_{n-1} q$

  et $\forall \sigma \in \Sigma, \delta(p, \sigma) \equiv_{n-1} \delta(q, \sigma)$

- Exemple



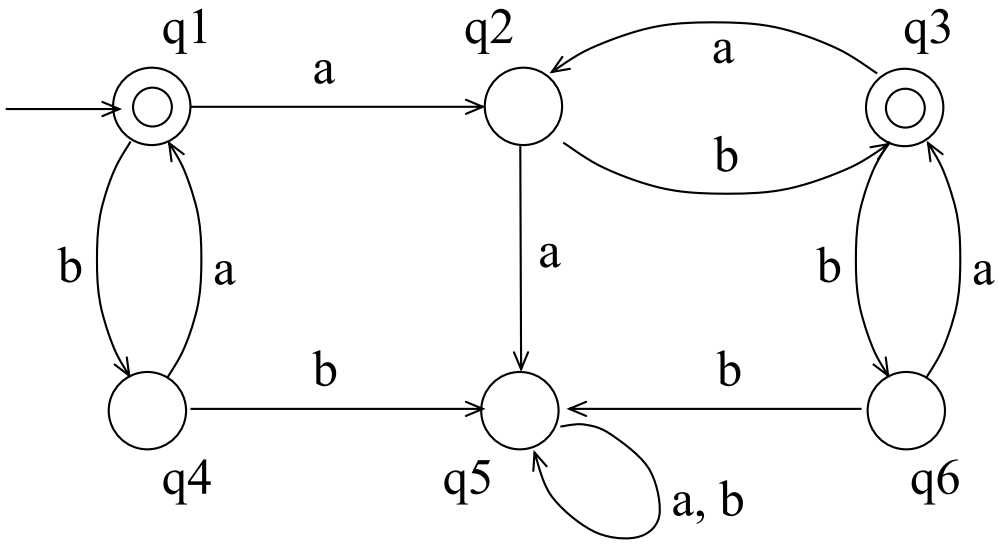
$q_1 \equiv_0 q_3$ car $q_1 \in F$ et $q_3 \in F$

$q_2 \equiv_0 q_4$ car $q_2 \notin F$ et $q_4 \notin F$

Minimisation des états

Minimisation d'un automate donné

- Exemple



$q_1 \equiv_1 q_3$ car $q_1 \equiv_0 q_3$ et $\delta(q_1, a) \equiv_0 \delta(q_3, a)$

$(\delta(q_1, a) = q_2$ et $\delta(q_3, a) = q_2$ et $\{q_2\})$

$\delta(q_1, b) \equiv_0 \delta(q_3, b)$

$(\delta(q_1, b) = q_4$ et $\delta(q_3, b) = q_6$ et $\{q_4, q_6\})$

$q_2 \not\equiv_1 q_4$ car $\delta(q_2, a) \not\equiv_0 \delta(q_4, a)$

$(\delta(q_2, a) = q_5$ et $\delta(q_4, a) = q_1$ et $\{q_1, q_3\}$ et $\{q_5\})$

Minimisation des états

Minimisation d'un automate donné

- Exemple

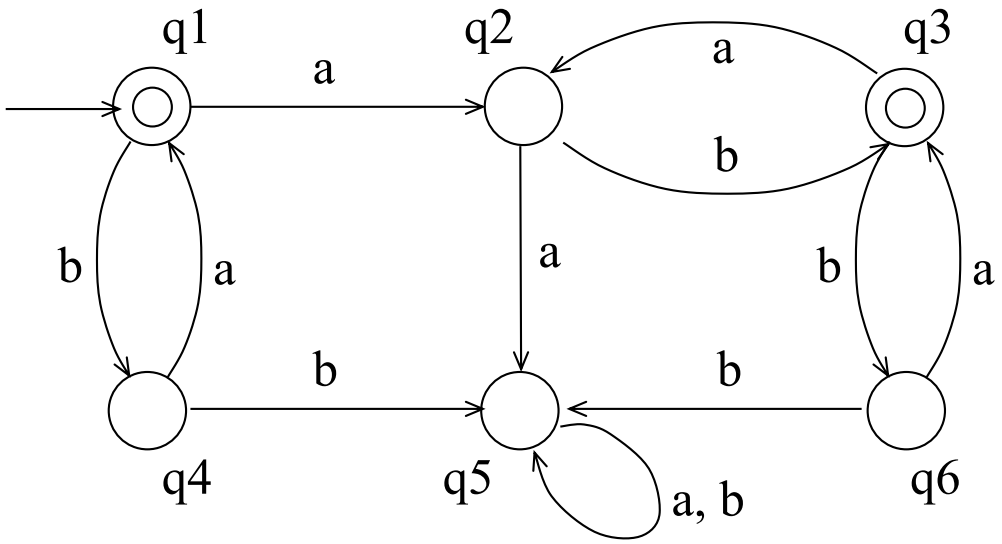
$\equiv_0$: $\{q_1, q_3\}, \{q_2, q_4, q_5, q_6\}$

$\equiv_1$: $\{q_1, q_3\}, \{q_2\}, \{q_4, q_6\}, \{q_5\}$

$\equiv_2$: $\{q_1, q_3\}, \{q_2\}, \{q_4, q_6\}, \{q_5\}$

…

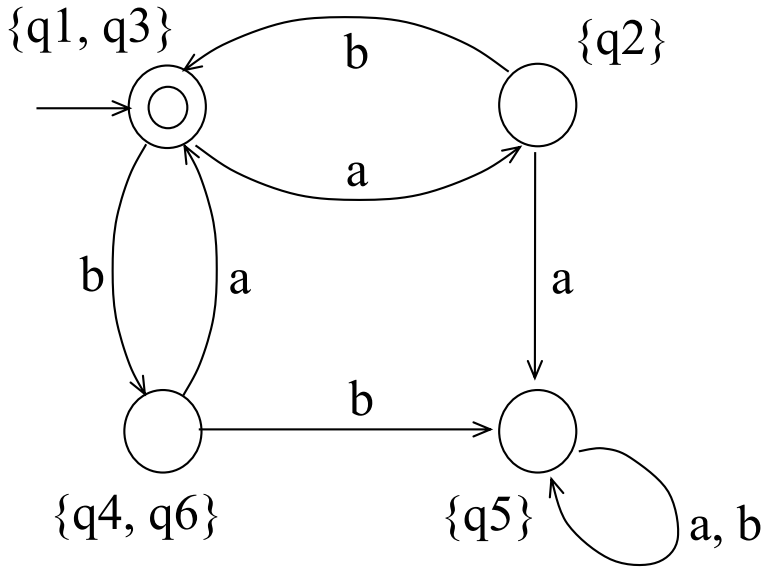
(Ça ne change plus)



- Finalement :

$q_1 \equiv q_3$

$q_4 \equiv q_6$



Langages rationnels

- Langage : ensemble de mots sur Σ
  - Éléments de base
    - L'ensemble $\varnothing$
    - Le mot vide ε
    - Les singletons sur Σ
  - Opérations
    - La concaténation de langages
    - La réunion de deux langages
    - L'intersection de deux langages
    - La fermeture de Kleene

- Langages rationnels
  - Représentation finie :
    - Éléments de base,
    - Concaténation,
    - Union,
    - Fermeture de Kleene

Langages rationnels

- Expressions régulières sur Σ : *plus petit* ensemble E tel que
  - $\emptyset \in E$
  - $\varepsilon \in E$
  - Si $\sigma \in \Sigma$, alors $\sigma \in E$
  - Si $e_1, e_2 \in E$, alors $e_1 + e_2 \in E$ et $e_1 . e_2 \in E$
  - Si $\mathrm{e} \in E$, alors $(\mathrm{e}) \in E$, $e^* \in E$ et $e^+ \in E$

- Priorité : $^{*/+}$ > . > +

- Exemple
  - Σ = {a, b} (a + a . b)* . a* + ε

Langages rationnels

- Langage représenté :
  - $[\![\emptyset]\!] = \emptyset$
  - $[\![\sigma]\!] = \{\sigma\}$ pour $\sigma \in \Sigma$
  - $[\![\varepsilon]\!] = \{\varepsilon\}$
  - $[\![e_1 + e_2]\!] = [\![e_1]\!] \cup [\![e_2]\!]$
  - $[\![e_1 . e_2]\!] = \{ w_1 w_2 \mid w_1 \in [\![e_1]\!] \text{ et } w_2 \in [\![e_2]\!] \}$
  - $[\![e^*]\!] = \bigcup_{n=0}^{\infty} [\![e^n]\!]$ où $e^0 = \{\varepsilon\}, e^1 = e, e^{n+1} = e . e^n$
  - $[\![e^+]\!] = \bigcup_{n=1}^{\infty} [\![e^n]\!]$ où $e^1 = e, e^{n+1} = e . e^n$

Langages rationnels

- Exemples
  - $\Sigma = \{a, b, c\}$ $(a \,.\, a^* \,.\, c + (b + c))^* \,.\, a^*$ Mots ne contenant pas ab
  - $\Sigma = \{0, 1\}$ $0 + 1 \,.\, (0 + 1)^*$ Entiers en binaire
  - $\Sigma = \{0, 1\}$ $0 + 1 \,.\, (0 + 1)^* \,.\, 0$ Entiers en binaire pairs
  - $\Sigma = \{0, 1\}$ $0^* + (((0^* \,.\, (1 + (1.1))) \,.\, ((0 \,.\, 0^*) \,.\, (1 + (1 \,.\, 1)))^*) \,.\, 0^*)$ Mots ne contenant pas 111

Langages rationnels

- Système d'équations linéaires gauche :

$$\begin{cases} X_1 = e_1^1 X_1 + \cdots + e_n^1 X_n + f^1 \\ \vdots \\ X_n = e_1^n X_1 + \cdots + e_n^n X_n + f^n \end{cases}$$

avec e_i^j expressions régulières $\in \Sigma \cup \{\varepsilon\} \cup \emptyset$, f^i quelconque (langage)

- Solution

$$X_1, \cdots, X_n \; si \; X_1 = [\![e_1^1]\!] . X_1 \cup \cdots \cup [\![e_n^1]\!] . X_n \cup [\![f^1]\!]$$
$$\vdots$$
$$X_n = [\![e_1^n]\!] . X_1 \cup \cdots \cup [\![e_n^n]\!] . X_n \cup [\![f^n]\!]$$

Langages rationnels

- Exemple

$$\begin{cases} X_1 = aX_2 + bX_3 + \varepsilon \\ X_2 = aX_1 + bX_4 \\ X_3 = bX_1 + aX_4 \\ X_4 = bX_2 + a\,X_3 \end{cases}$$

X_1 = { mots ayant un nombre pair de a et pair de b }
X_2 = { mots ayant un nombre impair de a et pair de b }
X_3 = { mots ayant un nombre pair de a et impair de b }
X_4 = { mots ayant un nombre impair de a et impair de b }

Langages rationnels

- Soit L un langage.

  L rationnel si ∃ S tq (**L**, L_1, L_2, …) est solution minimale de S

- Lemme

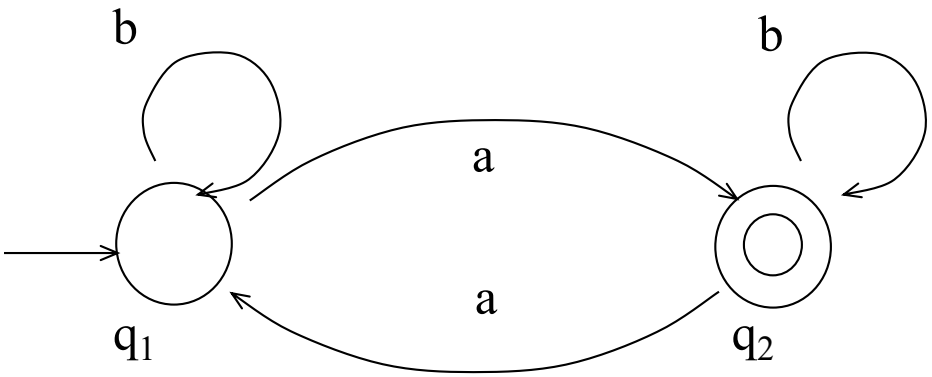
$X = eX + f$	avec e, f : expressions régulières	
$e^*.f$	est solution minimale de $X = eX + f$	si $\varepsilon \in e$
	est solution unique de $X = eX + f$	si $\varepsilon \notin e$

Preuve

- $e^*.f$ solution
- $e^*.f$ solution minimale
- $e^*.f$ solution unique si $\varepsilon \notin e$

Langages rationnels

- Exemple



$$\begin{cases} X_1 = & bX_1 + aX_2 \\ X_2 = & bX_2 + aX_1 + \varepsilon \end{cases}$$

$$X_1 = b^*a\,(b + ab^*a)^*$$
$$X_2 = (b + ab^*a)^*$$

Langages rationnels

- Exemple

$$\begin{cases} X_1 = & 1X_1 + 1X_2 + \varepsilon \\ X_2 = & 0X_1 \\ X_3 = & 0X_2 + 0X_3 + 1X_3 \end{cases}$$

X_1 = (1 + 10)*

X_2 = 0(1 + 10)*

X_3 = (0 + 1)* 00 (1 + 10)*

Rationalité

- Montrer qu'un langage est rationnel

(1) Stabilité (rappel : la classe des langages acceptés par un automate est stable par union, concaténation, fermeture itérative, complément, intersection)

(2) Caractérisation (rappel : un langage est rationnel ssi il est accepté par un automate)

(3) Un langage est rationnel ssi il peut être décrit par une expression régulière.

(4) Un langage L est rationnel ssi $\approx_L$ a un nombre fini de classes d'équivalence.

(2) et (3) → équivalence entre automate et ER

→ il existe des algorithmes

automate → ER

ER → automate

Rationalité

- Montrer qu'un langage est rationnel

  – À partir de (1) : utiliser les propriétés de stabilité

  → décomposer le langage en sous ensembles par union, intersection, concaténation,

  et montrer que ces sous ensembles sont rationnels.

  – À partir de (2) : construire un automate acceptant ce langage

  (on peut éventuellement déterminiser / minimiser cet automate)

  – À partir de (3) : construire une expression régulière décrivant ce langage

  – À partir de (4) : déterminer les classes d'équivalence de la relation $\approx_L$ et montrer que leur nombre est fini

Non rationalité

- Il existe des langages non rationnels
  - L'ensemble des expressions régulières est dénombrable
  - L'ensemble des langages est non dénombrable

- Tout langage fini est rationnel (il peut être décrit par une ER composée de l'union de tous les mots du langage)

→ La question de non rationalité ne se pose que pour les langages infinis

- Montrer la non rationalité
  - Stabilité et raisonnement par l'absurde
  - Lemme de l'étoile

Non rationalité

Propriétés de stabilité

- Pour montrer que L est non rationnel :

  on pose l'hypothèse que L est rationnel

  et on détermine L_0 non rationnel et L_1 rationnel tels que

  $$L_0 = L \ \theta \ L_1 \quad (\theta \in \{\cap, \cup, .\}$$

- L supposé rationnel
- L_1 rationnel $L \ \theta \ L_1$ rationnel (stabilité de la classe des langages rationnels par θ)
- Or $L \ \theta \ L_1 = L_0$ avec L_0 connu (démontré) non rationnel

  ➔ Contradiction

  → l'hypothèse (L rationnel) est fausse

Non rationalité

Propriétés de stabilité

- Exemple : L = { w ∈ {a,b}* | w contient autant de a que de b }
  Montrons que L est non rationnel

Supposons que L est rationnel.

Soit $L_1 = a^*b^*$.

L_1 rationnel (parce que décrit par une expression rationnelle).

Donc $L \cap L_1$ rationnel (par stabilité de la classe des langages rationnels par $\cap$).

Or $L \cap L_1 = L_0 = \{a^n b^n \mid n \geq 0\}$.

L_0 non rationnel (démontré plus loin).

Donc contradiction.

Donc l'hypothèse (L rationnel) est fausse.

Donc L non rationnel.

Non rationalité

Lemme de l'étoile

- Théorème Lemme de l'étoile

Soit L un langage rationnel infini accepté par un automate déterministe M à k états.

Soit z un mot quelconque de L tel que $|z| \geq k$.

Alors z peut être décomposé en uvw

avec $|uv| \leq k$, $|v| \neq 0$ *et* $uv^iw \in L$, $\forall i \geq 0$.

Non rationalité

Lemme de l'étoile

- Exemple : Montrons que $L = \{a^n b^n \mid n \geq 0\}$ est non rationnel.

Supposons que L est rationnel.
L est reconnu par un automate M à k états.
D'après le lemme de l'étoile, $\forall\ z \in L, |z| \geq k, \exists\ u, v, w \in \Sigma^*$ tels que

$$z = uvw,\ |uv| \leq k,\ |v| > 0 \text{ et } \forall\ i \geq 0,\ uv^i w \in L$$

Soit $z_0 = a^k b^k$.
On a bien $z_0 \in L$ et $|z_0| = 2k \geq k$.
Toutes les décompositions possibles $z_0 = uvw$ telles que $|uv| \leq k$, $|v| > 0$ sont de la forme $u = a^p$, $v = a^q$, $w = a^r b^k$ avec $q > 0$ et $p+q+r = k$.

Or $uv^i w = a^p a^{qi} a^r b^k = a^{p + qi + r} b^k$
On a $\forall\ i \neq 1,\ p + qi + r \neq k$
Donc $\forall\ i \neq 1,\ uv^i w \notin L$
Donc contradiction dans la propriété
Donc l'hypothèse (L rationnel) est fausse
Donc L non rationnel

Complexité d'algorithmes pour les automates

- Théorèmes

(i) Il existe un algorithme exponentiel (en le nombre d'états) — *déterminisation*

Entrée : un automate fini non déterministe

Sortie : un automate fini déterministe équivalent

(ii) Il existe un algorithme polynomial (en fonction de la taille de l'expression ou du nombre d'opérateurs) — *ER → automate*

Entrée : une expression régulière

Sortie : un automate non déterministe équivalent

(iii) Il existe un algorithme exponentiel (en fonction du nombre d'états) — *automate → ER*

Entrée : un automate non déterministe

Sortie : une expression régulière équivalente

(le nombre des R(i, j, k) est multipliée par 4 à chaque incrément de k)

$R(i, j, k) = R(i, j, k-1) \cup R(i, k, k-1)\, R(k, k, k-1)^*\, R(k, j, k-1)$

Complexité d'algorithmes pour les automates

(iv) Il existe un algorithme polynomial (en fonction du nombre d'états) — *minimisation*

Entrée : un automate déterministe

Sortie : l'automate déterministe minimal (standard) équivalent

(v) Il existe un algorithme polynomial pour décider si deux automates déterministes sont équivalents (passe par l'automate standard) — *équivalence*

Entrée : deux automates déterministes

Sortie : vrai s'ils sont équivalents, faux sinon

(vi) Il existe un algorithme exponentiel pour déterminer si deux automates non déterministes son équivalents — *équivalence*

Entrée : deux automates non déterministes

Sortie : vrai s'ils sont équivalents, faux sinon