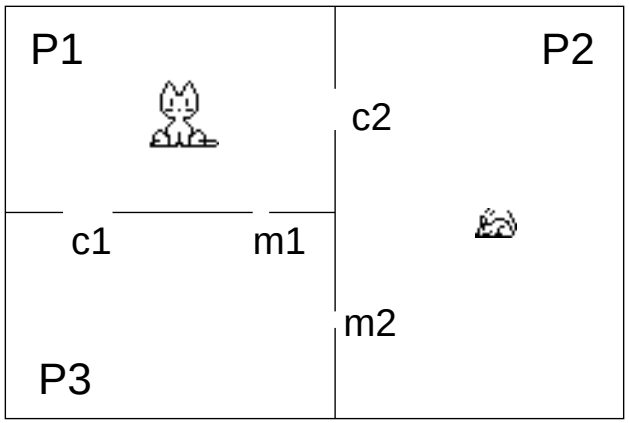
Automates à états finis

Philippe Quéinnec

18 février 2011

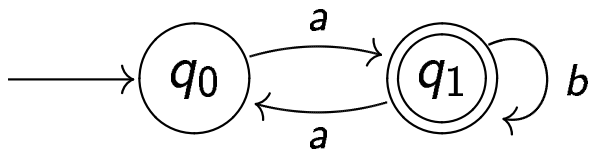
Le chat et la souris



Le chat et la souris se déplacent de manière indépendante et instantanée. Chaque passage est réservé au passage d'un seul animal (c_1, c_2 pour le chat, et m_1, m_2 pour la souris).

(d'après Stéphane Gaubert)

Les points abordés

- Alphabet, mot, langage
- Automate fini à états 

- Modélisation
  - résolution de problèmes
  - description de systèmes matériels
  - analyse lexicale
  - réseau
- Non déterminisme, déterminisation
- Minimisation, égalité d'automates
- Généralisation : automate à compteurs, à variables d'états, temporisé, à pile, sur les mots infinis...
- Expressions régulières

Organisation

- 6 cours, 2 TD
- examen écrit avec documents
- Le support contient
  - les définitions et les énoncés des théorèmes
  - + quelques énoncés d'exemples
- Le support ne contient pas (ou rarement)
  - les démonstrations des théorèmes
  - les solutions des exemples

Références bibliographiques

- Hopcroft & Ullman, *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley 1979.
- Salomaa, *Formal Languages*, ACM Press, 1973.
- Aho, Sethi & Ullman, *Compilateurs : principes, techniques et outils*, InterÉditions, 1989.

Plan

1. Introduction
   - Notion de langage
   - Opérations sur les langages
2. Automate fini à état
   - Définition
   - Langage accepté
   - Codage d'un automate
3. Modélisation
4. Non déterminisme, minimalité
   - Automate non déterministe
   - Déterminisation
   - Propriétés des langages rationnels
   - Minimisation
5. Variantes

Structure algébrique des langages

Informellement :

- Alphabet : ensemble de symboles
- Mot : suite de symboles
- Langage : ensemble de mots

Plus rigoureusement :

- Alphabet : ensemble fini de symboles $X = \{a, b, \ldots\}$
- Opération de concaténation : $\bullet$: Mot $\times$ Mot $\mapsto$ Mot
  - $\bullet$ associative : $(s_1 \bullet s_2) \bullet s_3 = s_1 \bullet (s_2 \bullet s_3) = s_1 \bullet s_2 \bullet s_3$
  - élément neutre (le mot vide) : Λ
  - m mot écrit sur X :

$$\begin{cases} m = a \ (a \in X) \\ m = m_1 \bullet m_2 \ \text{ où } m_1 \text{ et } m_2 \text{ mots écrits sur } X \end{cases}$$

  - Extension aux ensembles de mots : soit A et B deux ensembles de mots, $A \bullet B = \{m_a \bullet m_b \mid m_a \in A \wedge m_b \in B\}$

- longueur d'un mot $|m|$ = nombre de symboles qui le constituent :

$$\begin{aligned} |\Lambda| &= 0 \\ |a| &= 1,\ \forall a \in X \\ |m_1 \bullet m_2| &= |m_1| + |m_2| \end{aligned}$$

- préfixe/suffixe de m : m_1/m_2 tels que $m = m_1 \bullet m_2$

Étoile de Kleene

Objectif : construire l'ensemble de tous les mots possibles

mot vide (0 symbole) :	$X^0 = \{\Lambda\}$
mots de 1 symbole :	$X^1 = X$
mots de 2 symboles :	$X^2 = X \bullet X$
mots de 3 symboles :	$X^3 = X^2 \bullet X$
mots de $n+1$ symboles :	$X^{n+1} = X^n \bullet X$

Étoile de Kleene d'un ensemble

$$X^* = \bigcup_{i=0}^{\infty} X^i$$

Langage sur X = sous-ensemble de X^*

Langage vide : $\varnothing$

(X^* muni de $\bullet$ = monoïde libre engendré par X)

Opérations sur les langages

$$
\begin{array}{lcl}
L_1 \bullet L_2 & = & \{m_1 \bullet m_2 \in X^* \mid m_1 \in L_1 \wedge m_2 \in L_2\} \\
L_1 \cup L_2 & = & \{m \in X^* \mid m \in L_1 \vee m \in L_2\} \\
\overline{L_1} & = & \{m \in X^* \mid m \notin L_1\} \\
L_1 \cap L_2 & = & \{m \in X^* \mid m \in L_1 \wedge m \in L_2\} = \overline{\overline{L_1} \cup \overline{L_2}} \\
L_1 \setminus L_2 & = & \{m \in X^* \mid m \in L_1 \wedge m \notin L_2\} = L_1 \cap \overline{L_2} \\
L^0 & = & \{\Lambda\} \\
L^{n+1} & = & L \bullet L^n \\
L^* & = & \bigcup_{i=0}^{\infty} L^i \\
L^+ & = & L^* \setminus \{\Lambda\} = L \bullet L^* \text{ si } \Lambda \notin L
\end{array}
$$

Plan

1. Introduction
   - Notion de langage
   - Opérations sur les langages
2. Automate fini à état
   - Définition
   - Langage accepté
   - Codage d'un automate
3. Modélisation
4. Non déterminisme, minimalité
   - Automate non déterministe
   - Déterminisation
   - Propriétés des langages rationnels
   - Minimisation
5. Variantes

Automate fini à état

Automate fini à état

quintuplet $A = (Q, X, \delta, q_I, F)$ où :

- Q : ensemble fini d'états
- X : alphabet
- $q_I \in Q$: l'état initial de l'automate
- $F \subseteq Q$: les états finals (ou terminaux)
- $\delta \in Q \times X \mapsto Q$: fonction de transition de l'automate.

Automate complet

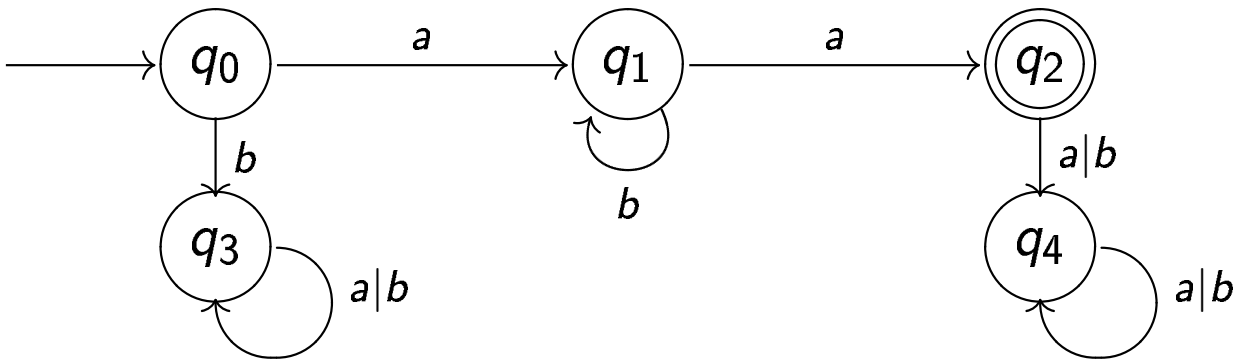
Automate dont la fonction de transition est totale.

Tout automate peut être complété par ajout d'un état piège.

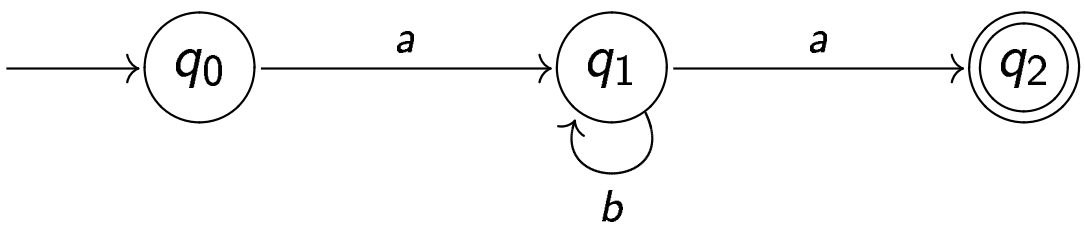
Représentation graphique

Considérons $X = \{a, b\}$, $Q = \{q_0, q_1, q_2, q_3, q_4\}$, $q_I = q_0$, $F = \{q_2\}$ et δ définie par :

δ	a	b
q_0	q_1	q_3
q_1	q_2	q_1
q_2	q_4	q_4
q_3	q_3	q_3
q_4	q_4	q_4



ou (incomplet) :



Extension de δ

$$\begin{cases} \hat{\delta}(q, \Lambda) & = & q \\ \hat{\delta}(q, au) & = & \hat{\delta}(\delta(q, a), u), \ a \in X, \ u \in X^* \end{cases}$$

Configuration

couple (q, m) avec $q \in Q$ et $m \in X^*$

Transitions

Relation $\vdash$ entre configurations : $(q, am) \vdash (q', m)$ si $q' = \delta(q, a)$.
$\vdash^*$ fermeture réflexive transitive de $\vdash$

Langage accepté

$$\begin{aligned} L(A) &= \{m \in X^* \mid \hat{\delta}(q_I, m) \in F\} \\ &= \{m \in X^* \mid \exists q_F \in F : (q_I, m) \vdash^* (q_F, \Lambda)\} \end{aligned}$$

Codage d'un automate

symboles de l'alphabet	$\rightarrow$	entiers $1..M$
états	$\rightarrow$	entiers $1..N$
états terminaux	$\rightarrow$	tableau $1..N$ de booléens
transitions	$\rightarrow$	tableau $1..N$, $1..M$ d'entiers $-1, 1..N$
avancer curseur	$\rightarrow$	fonction `getchar()`
état courant	$\rightarrow$	variable entière `état`

```
état := q_I
dernierLu := getchar()
tantque (état ≠ -1) ∧ (dernierLu ≠ -1) faire
   état := transitions[état][dernierLu]
   dernierLu := getchar()
fintq
si état ≠ -1 ∧ terminal[état] alors succès
                              sinon échec
finsi
```

Plan

1. Introduction
   - Notion de langage
   - Opérations sur les langages
2. Automate fini à état
   - Définition
   - Langage accepté
   - Codage d'un automate
3. Modélisation
4. Non déterminisme, minimalité
   - Automate non déterministe
   - Déterminisation
   - Propriétés des langages rationnels
   - Minimisation
5. Variantes

Modélisation – résolution de problèmes

L'homme, le loup, le mouton et le chou

Sur une rive, se trouvent un Homme, un Loup, un Mouton et un (gros) Chou. Une barque permet de traverser, mais elle ne possède que deux places, et seul l'Homme peut ramer. Seul la présence de l'Homme peut assurer une coexistence pacifique entre le Loup et le Mouton d'une part, et entre le Mouton et le Chou d'autre part. Existe-t-il une solution pour que les quatre se retrouvent sur l'autre rive ?

Modélisation – résolution de problèmes

Les missionnaires et les cannibales

Trois missionnaires et trois cannibales sont sur la rive gauche d'un fleuve. Ils disposent d'une barque qui peut transporter deux personnes (rameur inclus). Si les cannibales se retrouvent en majorité absolue sur une rive, ils mangent les missionnaires. Les six peuvent-ils traverser le fleuve en sécurité (pour les missionnaires) ? Même problème avec quatre missionnaires et quatre cannibales.

Modélisation – système physique

Ampoule électrique

Une ampoule peut être allumée ou éteinte au moyen d'un interrupteur. Si elle subit une surtension, elle grille et ne peut plus s'allumer.

Économiseur d'écran

Un écran peut être dans trois modes : actif, économie, en veille (éteint). Il passe d'actif à économie après un délai fixé d'inactivité ; d'économie à veille après un deuxième délai ; il passe en mode actif sur action de l'utilisateur.

Modélisation – reconnaissance et contrôle

Dictionnaire

Un dictionnaire permet à la fois de reconnaître un ensemble de mots, et de les distinguer les uns des autres.
Automate reconnaissant {`if`, `then`, `else`, `mod`, `module`} ?

Reconnaissance d'un réel

Un nombre réel peut prendre la forme `3.1415`, `276`, `234.`, ou `.1234`.

Saisie d'un code de carte bancaire

Un automate de saisie d'un code secret de carte bancaire doit accepter 4 chiffres suivi de la validation, avec la possibilité de corriger le dernier chiffre saisi et d'annuler toute la saisie.

Modélisation – protocoles de communication

Le protocole du bit alterné

Un émetteur souhaite transmettre des messages à un récepteur, en utilisant un canal non fiable (perte possible du message).
Pour cela, à chaque réception de message, le récepteur renvoie un acquittement (acknowledge), qui peut aussi se perdre.
Quand l'émetteur n'a pas reçu l'acquittement après un délai fixé (timeout), il réémet le message.
Pour que le récepteur puisse distinguer entre un nouveau message et un message réémis dont l'acquittement avait été perdu, il est nécessaire de numéroter les messages (distinction entre le message numéro N et le message $N+1$).
Si le canal est FIFO, l'écart de numérotation (message attendu, message effectivement reçu) ne peut dépasser 1, et il suffit de transmettre, en plus du message, un seul bit, qui est inversé à chaque transmission.

Modélisation – synchronisation

producteur/consommateur

Un meunier dépose, au fur et à mesure de sa production, ses sacs de farine dans un petit local, qui ne peut contenir que 4 sacs. Le boulanger vient prendre un par un les sacs dont il a besoin. Décrire les actions autorisées pour le meunier et le boulanger, en fonction du remplissage du local.

le tournoi de bridge

Un portier de salle de jeu doit assurer que la salle contient toujours un nombre de joueurs multiple de 4 (pour que toutes les tables soient complètes). Pour cela, il retarde l'entrée et la sortie des individus tant qu'il ne peut assurer cette condition. Ainsi, il faut attendre qu'il y ait quatre demandes d'entrées (pour que quatre nouveaux joueurs puissent entrer et former une table), ou quatre demandes de sortie (une table s'en va), ou une demande d'entrée et une demande sortie (échange de deux joueurs).

Plan

1. Introduction
   - Notion de langage
   - Opérations sur les langages
2. Automate fini à état
   - Définition
   - Langage accepté
   - Codage d'un automate
3. Modélisation
4. Non déterminisme, minimalité
   - Automate non déterministe
   - Déterminisation
   - Propriétés des langages rationnels
   - Minimisation
5. Variantes

Automate non déterministe

Automate non déterministe

quintuplet $A = (Q, X, \delta, I, F)$ où :

- Q : ensemble fini d'états
- X : alphabet
- ϵ : symbole $\notin Q \cup X$ (transition arbitraire)
- $I \subseteq Q$: états initiaux de l'automate
- $F \subseteq Q$: les états finals (ou terminaux)
- $\delta \in Q \times (X \cup \{\epsilon\}) \mapsto \mathcal{P}(Q)$: fonction de transition

Trois formes de non déterminisme :

- $card(I) > 1$: plusieurs états initiaux
- $\delta(q, \epsilon) \neq \varnothing$: transition arbitraire sans regarder l'entrée
- $card(\delta(q, a)) > 1$: plusieurs cibles pour une même entrée

Configuration/transitions

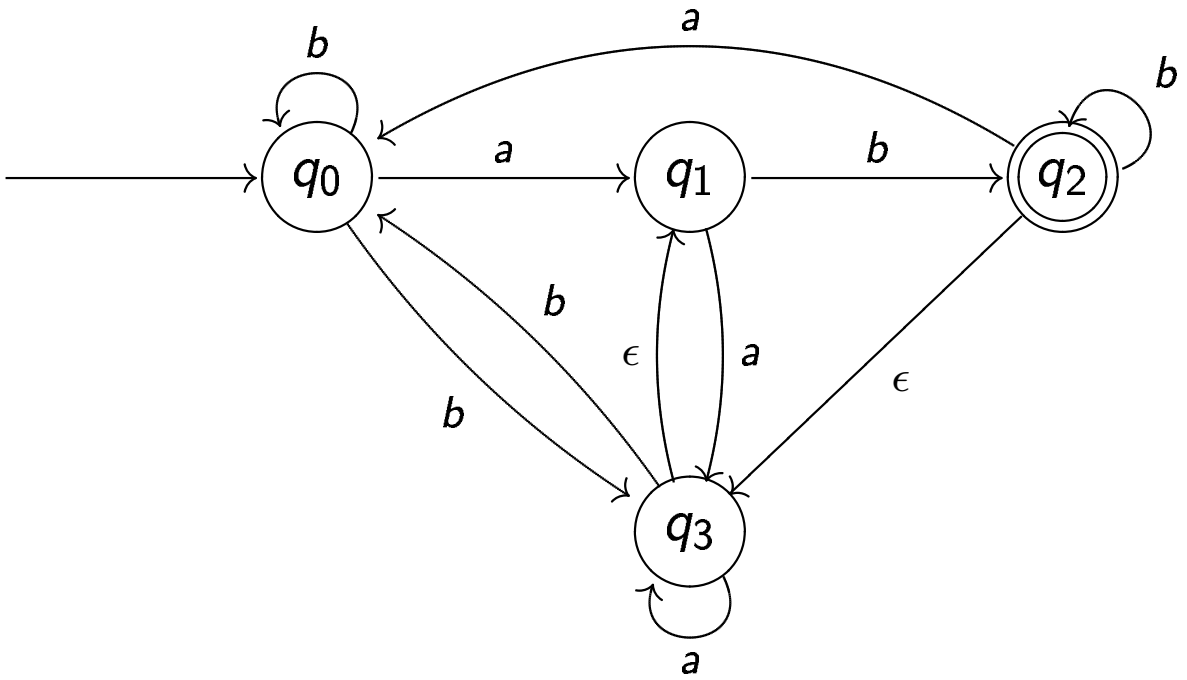
Configuration : couple (q, m) avec $q \in Q$ et $m \in X^*$

Relation $\vdash$ entre configurations :

$$\begin{cases} (q, am) \vdash (q', m) \text{ si } q' \in \delta(q, a) \\ (q, am) \vdash (q', am) \text{ si } q' \in \delta(q, \epsilon) \end{cases}$$

Langage accepté

$$L(A) = \{m \in X^* \mid \exists q_I \in I,\ \exists q_F \in F : (q_I, m) \vdash^* (q_F, \Lambda)\}$$



Indéterminisme :

- $I = \{q_0\}$
- $\delta(q_0, b) = \{q_0, q_3\}$
- $\delta(q_2, \epsilon) = \{q_3\}$

Déterminisation

Équivalence

Pour tout automate indéterministe, il existe un automate déterministe qui accepte le même langage.

Éliminer l'indéterminisme en regroupant :

- les états initiaux ;
- les états reliés par ϵ (ϵ-fermeture) ;
- les états cibles de transition sur une même entrée.

Un état de l'automate déterminisé = ensemble d'états de l'automate indéterministe.
L'automate déterminisé simule en parallèle tous les chemins possibles.

ϵ-fermeture

Soit $P \subseteq Q$,
ϵ-fermeture$(P) = \{q \in Q \mid \exists q' \in P : (q', m) \vdash^* (q, m)\}$
Calcul :

- $F_0 = P$
- $F_{i+1} = F_i \cup \{q \in Q \mid \exists q' \in F_i : q \in \delta(q', \epsilon)\}$
- Comme Q est fini, $\exists n : F_{n+1} = F_n$
- et ϵ-fermeture$(P) = F_n$

Déterminisation à la volée

Soit un AFN $A = (Q, X, \delta, I, F)$ et un mot $m = a_1 a_2 \dots a_n$

$$
\begin{aligned}
q'_0 &= \epsilon\text{-fermeture}(I) \\
q'_i &= \epsilon\text{-fermeture}\Big(\bigcup_{q \in q'_{i-1}} \delta(q, a_i)\Big) \\
&\vdots \\
q'_n &= \epsilon\text{-fermeture}\Big(\bigcup_{q \in q'_{n-1}} \delta(q, a_n)\Big)
\end{aligned}
$$

Mot reconnu ssi $q'_n \cap F \neq \emptyset$.
(Cela revient à simuler tous les chemins possibles en lisant m)

Automate déterminisé

Soit l'AFN $A = (Q, X, \delta, I, F)$.
L'AFD correspondant est $(Q', X, \delta', q'_0, F')$:

- $Q' \subseteq \mathcal{P}(Q)$
- $q'_0 = \epsilon$-fermeture(I)
- δ' définie de $Q' \times X$ dans Q' par :
  $\delta'(P', a) = \epsilon\text{-fermeture}(\bigcup_{q' \in P'} \delta(q', a))$
- $F' = \{q'_t \in Q' \mid q'_t \cap F \neq \emptyset\}$

Calcul :

- $Q'_0 = \{\epsilon\text{-fermeture}(I)\}$
- $Q'_{i+1} = Q'_i \cup \{\epsilon\text{-fermeture}(\bigcup_{q \in P'} \delta(q, a)) \mid P' \in Q'_i, a \in X\}$
- $Q' = Q'_n$ avec $n = \min\{i \geq 0 \mid Q'_i = Q'_{i+1}\}$

Explosion du nombre d'états

Explosion

Pour un automate non déterministe ayant n états, le plus petit (minimal) automate déterministe équivalent peut avoir 2^n états.

Exemple :

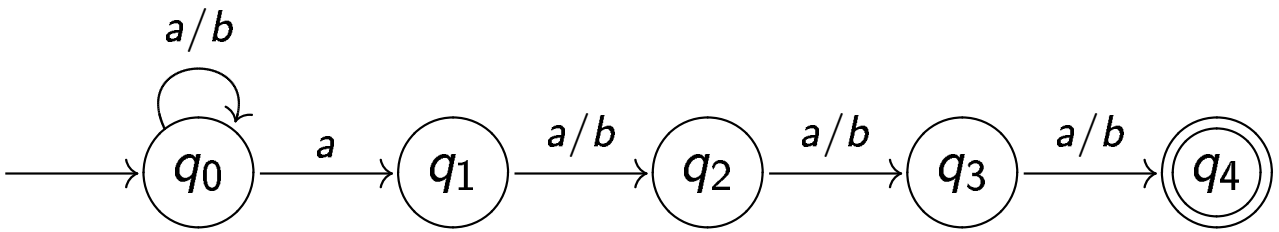
$L_n = L_1 \bullet \{a\} \bullet L_2$

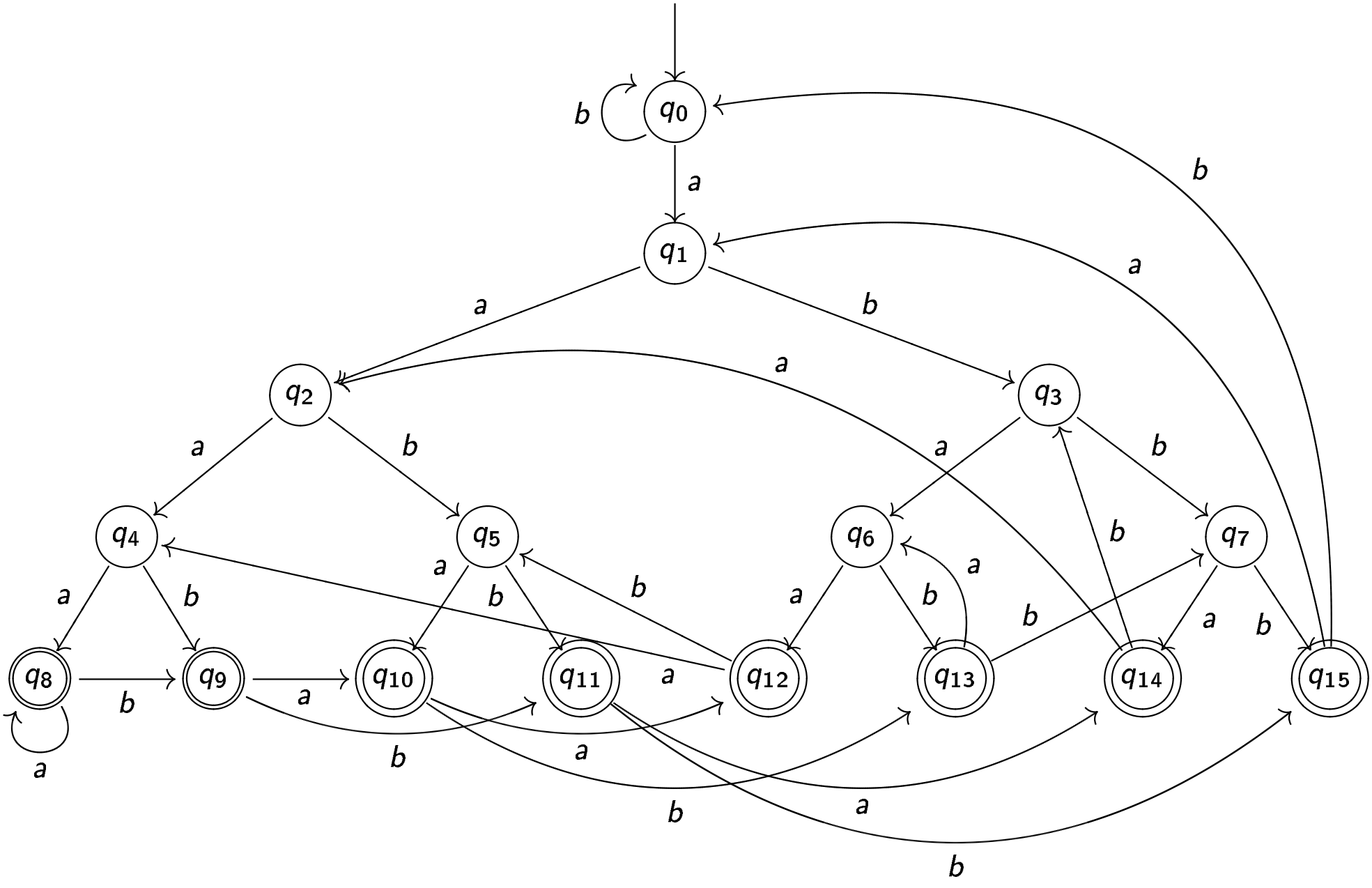
où $L_1 = \{a, b\}^*$ (mots quelconques)

$L_2 = \{m \in \{a, b\}^* \mid |m| = n\}$ (mots de longueur n)

$=$ mots sur $\{a, b\}$ ayant un a à $n + 1$ positions de la fin.

NFA à $n + 2$ états, DFA minimal à $2^{n+1} + 1$ états




q_0
q_1
q_2
q_3
q_4
q_5
q_6
q_7
q_8
q_9
q_{10}
q_{11}
q_{12}
q_{13}
q_{14}
q_{15}
a
b

Propriétés des langages rationnels

Langage rationnel

L est rationnel $\equiv \exists A$ automate : $L = L(A)$
Rat = ensemble des langages rationnels (ou réguliers)

Fermeture

Rat est fermé par union, produit, étoile, intersection, complémentaire, différence et miroir.
Soient $L_1, L_2 \in \text{Rat}$, alors :

$$L_1 \cup L_2 \in \text{Rat} \qquad L_1 \bullet L_2 \in \text{Rat}$$
$$L_1{}^* \in \text{Rat} \qquad L_1 \cap L_2 \in \text{Rat}$$
$$\overline{L_1} \in \text{Rat} \qquad L_1 \setminus L_2 \in \text{Rat}$$
$$\widetilde{L_1} \in \text{Rat}$$

Théorème de Nérode

Soit $L \subseteq X*$. Les propriétés suivantes sont équivalentes :

- $L \in$ Rat.
- Soit la relation d'équivalence :
  $x \equiv_L y \Leftrightarrow \forall z \in X^* \ (xz \in L \Leftrightarrow yz \in L)$.
  Alors $\equiv_L$ a un index fini (index fini = nombre fini de classes d'équivalence).

Les classes d'équivalence de $\equiv_L$ forment l'ensemble des préfixes des mots de L.

Exemple de langage non rationnel

$L_= = \{a^n b^n \mid n \geq 0\} \notin$ Rat :
en effet, $\forall i, a^i b^i \in L_=$, mais $j \neq i, a^j b^i \notin L_=$,
donc $\forall i, j \ i \neq j \Rightarrow a^i \not\equiv_{L_=} a^j$,
donc $\equiv_{L_=}$ est d'index infini.

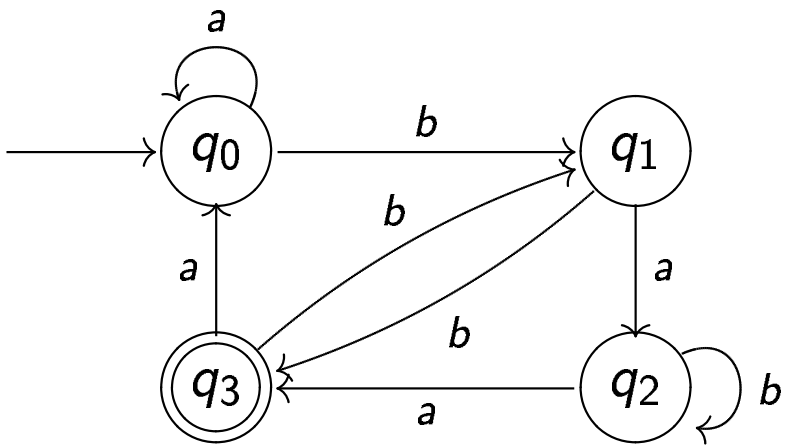
Automate minimal

Automate minimal

$\forall L \in \text{Rat}$, $\exists A_m$ déterministe et complet tel que $L(A_m) = L$ et tel que $\forall A$ déterministe complet, $L(A) = L \wedge A \neq A_m \Rightarrow |Q_A| > |Q_{A_m}|$

- Soit $\equiv$ définie par :
  $q_1 \equiv q_2$ ssi $\forall m \in X^*,\ (\hat{\delta}(q_1, m) \in F \Leftrightarrow \hat{\delta}(q_2, m) \in F)$
- Soit $A/_{\equiv} = (Q/_{\equiv}, X, \delta_{\equiv}, [q_I]_{\equiv}, F/_{\equiv})$
  - états de $A/_{\equiv}$: classes d'équivalence de Q
  - transitions : $\delta_{\equiv}([q]_{\equiv}, a) = [\delta(q, a)]_{\equiv}$
- Alors $L(A) = L(A/_{\equiv})$, et $A/_{\equiv}$ est l'automate minimal qui reconnaît ce langage.

Vérification de la minimalité



- q_3 est le seul état final. Il ne peut donc être équivalent à aucun autre.
- $q_0 \not\equiv q_1$: $\hat{\delta}(q_1, b) \in F$ mais $\hat{\delta}(q_0, b) \notin F$
- $q_0 \not\equiv q_2$: $\hat{\delta}(q_2, bb) \notin F$ mais $\hat{\delta}(q_0, bb) \in F$
- $q_1 \not\equiv q_2$: $\hat{\delta}(q_1, aa) \in F$ mais $\hat{\delta}(q_2, aa) \notin F$

Calcul de l'automate minimal

On partitionne Q en groupe d'états tq q_1 et q_2 sont dans deux groupes différents si $\exists m \in X^*$ tq $(\hat{\delta}(q_1, m) \in F \Leftrightarrow \hat{\delta}(q_2, m) \notin F)$.

$\Pi = \{F, Q \setminus F\}$ (distingués par Λ)
boucle
  $\Pi_{\text{new}} = \emptyset$
  pour chaque groupe G de Π
    Π_G = découpe de G
      tq q_1 et q_2 de G sont dans le même nouveau sous-groupe
      ssi $\forall a \in X$, $\delta(q_1, a)$ et $\delta(q_2, a)$ sont dans le même groupe de Π
    $\Pi_{\text{new}} = \Pi_{\text{new}} \cup \Pi_G$
  finpour
  si $\Pi_{\text{new}} = \Pi$, alors terminé
  $\Pi = \Pi_{\text{new}}$
finboucle

Calcul de l'automate minimal – version 2

Algorithme de Brzozowski

Soit A automate déterministe complet

- Miroir de A pour obtenir $L(\widetilde{A})$
  (inverser le sens des arcs et échanger états terminaux et initiaux)
- Déterminiser
- Miroir
- Déterminiser

Et ça marche...

Plan

1. Introduction
   - Notion de langage
   - Opérations sur les langages
2. Automate fini à état
   - Définition
   - Langage accepté
   - Codage d'un automate
3. Modélisation
4. Non déterminisme, minimalité
   - Automate non déterministe
   - Déterminisation
   - Propriétés des langages rationnels
   - Minimisation
5. Variantes

Variantes

- Machine de Mealy (1955) : automate déterministe, sans état d'acception, avec émission d'une sortie sur transition (dépend de l'état et du symbole lu).
  $\approx$ transducteur déterministe.
- Machine de Moore (1956) : automate déterministe, sans état d'acception, avec émission d'une sortie par état.
  Étonnamment aussi expressif qu'une machine de Mealy, mais nombre d'états plus grand.
- Transition sur présence/absence d'un événement : introduire explicitement dans l'alphabet a et $\bar{a}$.
  Attention : $\bar{0} \neq 1$