

## Chapitre 4. Automates.

### 1 Automates finis déterministes

#### 1.1 Introduction et exemples

Nous avons déjà considéré dans le chapitre précédents des automates sans en donner une définition précise.

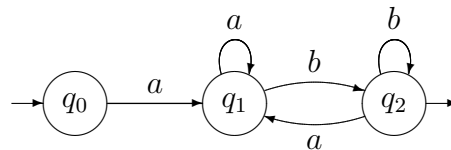
De manière informelle, un automate est une machine abstraite qui peut prendre un nombre fini d'états, qui reçoit en entrée un mot écrit sur un alphabet  $\Sigma$  et qui change d'état à la lecture des lettres de ce mot.

On sélectionne au préalable un état particulier appelé état *initial* et un certain nombre d'états qui sont qualifiés d'*acceptants* ou de *finaux*.

La lecture des mots se fait en partant de l'état initial. À la fin de la lecture du mot, si l'état dans lequel se trouve l'automate est un état acceptants, on dira que le mot est *accepté* ou *reconnu* et sinon, on dira qu'il est *rejeté*.

On représentera les automates sous la forme d'un graphe valué dont les sommets sont les différents états possibles de l'automate, et les arêtes représentent les transitions entre les états et sont étiquetées par des lettres de  $\Sigma$ .

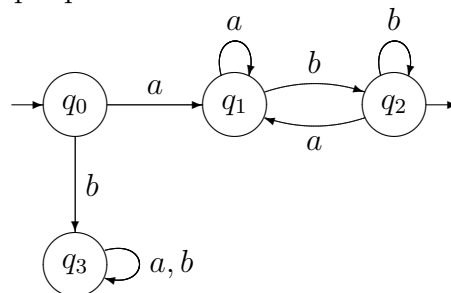
Considérons par exemple l'automate représenté ci-dessous, sur l'alphabet  $\Sigma = \{a, b\}$ .



L'état initial (ici  $q_0$ ) est désigné par une flèche entrante, les états acceptants (ici  $q_2$ ) sont représentés par une flèche sortante.<sup>1</sup>

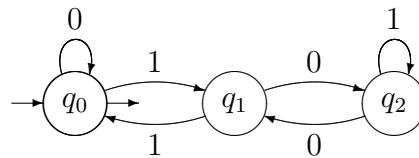
On se convaincra aisément que pour qu'un mot puisse être lu par cet automate, il faut qu'il commence par un  $a$ , et qu'alors la lecture du mot se termine à l'état  $q_1$  si la dernière lettre du mot est un  $a$  et à l'état  $q_2$  si la dernière lettre du mot est un  $b$  : on en déduit que les mots acceptés par cet automate sont les mots commençant par  $a$  et dont la dernière lettre est un  $b$ . On dit que le *langage reconnu* par cet automate est  $a\Sigma^*b$ .

Faute de transition partant de  $q_0$  et étiquetée par  $b$ , un mot débutant par un  $b$  ne peut être lu par l'automate précédent. On dit que  $(q_0, b)$  est un *blocage* de l'automate. un automate sans blocage est dit *complet*. Il est toujours possible de rendre complet un automate présentant des blocages en rajoutant un état vers lequel aboutissent tous les blocages, état qu'on ne peut ensuite plus quitter. Cet état est appelé un *puits*. Le nouvel automate est alors complet, il permet de lire tous les mots de  $\Sigma^*$  et les mots reconnus sont les mêmes que pour l'automate initial. Sur notre exemple, l'automate obtenu est alors :



1. certains auteurs représentent les états acceptants en les entourant d'un double cercle

**Exercice :** On considère l'automate sur l'alphabet  $\Sigma = \{0, 1\}$  représenté par :



Les mots sur l'alphabet  $\Sigma = \{0, 1\}$  seront identifiés aux entiers naturels via leur écriture en base 2.

- Pour les entiers de  $\llbracket 0, 8 \rrbracket$ , indiquer l'état auquel on aboutit à la fin de la lecture du mot associé.
- Déterminer, en justifiant, le langage reconnu par l'automate précédent.

## 1.2 Définitions

**Définition 1.** Un automate fini déterministe (AFD ou DFA pour deterministic finite automaton) est un quintuplet  $A = (\Sigma, Q, q_0, F, \delta)$  où :

- $\Sigma$  est un alphabet (fini) ;
- $Q$  est un ensemble fini dont les éléments sont appelés les états de  $A$  ;
- $q_0$  est un élément de  $Q$  appelé état initial ;
- $F$  est une partie de  $Q$  dont les éléments sont appelés les états acceptants ou finaux ;
- $\delta$  est une application d'une partie de  $Q \times \Sigma$  dans  $Q$ , appelée fonction de transition

Lorsque  $\delta$  est définie sur  $Q \times \Sigma$  tout entier, l'automate  $A$  est dit complet.

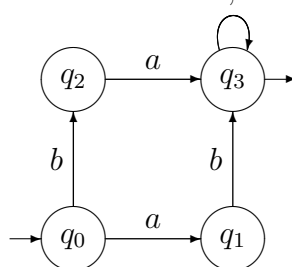
Si  $\delta$  n'est pas définie en  $(q, a) \in Q \times \Sigma$ , on dit que  $(q, a)$  est un blocage de  $A$ .

**Exemple :** Considérons l'automate  $A_0$  défini sur  $\Sigma = \{a, b\}$  à 3 états nommés  $q_0, q_1, q_2$  avec comme état initial  $q_0$  et comme états acceptants  $F = \{q_3\}$  et la fonction de

transition définie par le tableau

$\delta$	$a$	$b$
$q_0$	$q_1$	$q_2$
$q_1$	—	$q_3$
$q_2$	$q_3$	—
$q_3$	$q_3$	$q_3$

$a, b$



**Notations :** Si  $\delta(q_i, a) = q_j$ , on note encore  $q_i.a = q_j$  et cette transition est représentée par  $q_i \xrightarrow{a} q_j$ .

**Définition 2.** Si  $A = (\Sigma, Q, q_0, F, \delta)$  est un AFD, on appelle fonction de transition étendue aux mots la fonction  $\delta^*$  définie récursivement sur  $Q \times \Sigma^*$  par :

- $\forall q \in Q, \delta^*(q, \varepsilon) = q$
- $\forall (q, m, a) \in Q \times \Sigma^* \times \Sigma, \delta^*(q, ma) = \delta(\delta^*(q, m), a)$

Si  $\delta^*(q, m) = q'$  on note encore  $q.m = q'$ .

**Remarque :** Si l'automate  $A$  présente des blocages, la fonction  $\delta^*$  n'est définie que sur une partie de  $Q \times \Sigma^*$ . Plus précisément, si  $m = a_1 a_2 \dots a_n$  avec les  $a_i$  dans  $\Sigma$  et  $q \in Q$ ,  $\delta^*(q, m)$  est défini s'il existe une succession de transitions  $q \xrightarrow{a_1} q_{i_1} \xrightarrow{a_2} \dots \xrightarrow{a_n} q_{i_n}$  débutant en  $q$  et on a alors  $\delta^*(q, m) = q_{i_n}$ . Autrement dit,  $\delta^*(q, m)$  est l'unique état (s'il n'y a pas blocage) auquel on aboutit quand on lit le mot  $m$  à partir de l'état  $q$ .

**Définition 3.**  $\diamond$  Soit un automate  $A = (\Sigma, Q, q_0, F, \delta)$  fini déterministe et  $\delta^*$  la fonction de transition étendue

— Un mot  $m \in \Sigma^*$  est dit reconnu par  $A$  si  $\delta^*(q_0, m) \in F$

— Le langage reconnu par  $A$ , noté  $\mathcal{L}(A)$  est l'ensemble des mots reconnus par  $A$ .

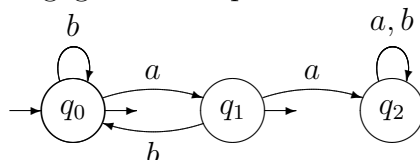
$\diamond$  Deux automates sont dits équivalents s'ils reconnaissent le même langage.

$\diamond$  Un langage  $L$  sur l'alphabet  $\Sigma$  est dit reconnaissable s'il existe un AFD  $A$  sur  $\Sigma$  tel que  $L = \mathcal{L}(A)$ . On note  $\text{Rec}(\Sigma)$  l'ensemble des langages reconnaissables sur  $\Sigma$ .

**Remarques :** – D'après la remarque faite dans le paragraphe précédent, pour tout AFD  $A$ , il existe un automate fini déterministe complet équivalent à  $A$  obtenu par ajout d'un état « puits » vers lequel mènent tous les blocages et qu'on ne quitte plus ensuite.

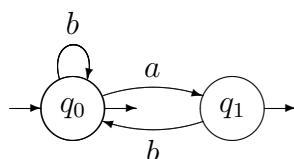
– Une succession de transitions  $q \xrightarrow{a_1} q_{i_1} \xrightarrow{a_2} \dots \xrightarrow{a_n} q_{i_n}$  est appelée un chemin dans l'automate  $A$  et le mot  $m = a_{i_1} a_{i_2} \dots a_{i_n}$  est appelé son étiquette. Ce chemin est dit acceptant si son origine  $q$  est l'état initial  $q_0$  de l'automate et que l'état d'arrivée  $q_{i_n}$  est un état acceptant. Le langage reconnu par un automate  $A$  est donc l'ensemble des étiquettes des chemins acceptants de cet automate.

**Exercice :** Déterminer le langage reconnu par l'automate :



### 1.3 Émondage

La fonction principale d'un automate est de reconnaître des mots en parcourant un par un leurs caractères. Dans le cas d'un automate complet, la complexité de l'algorithme qui en résulte est proportionnelle à la longueur du mot puisqu'il n'y a pas de blocage. Mais ceci peut être pénalisant : par exemple, dans l'automate ci-dessus, si on arrive dans l'état  $q_2$ , il est impossible de le quitter -  $q_2$  étant un « puits » - et puisque  $q_2$  n'est pas un état final, il n'est pas nécessaire de poursuivre la lecture du mot puisque ce dernier ne sera de toute façon pas reconnu. On peut donc supprimer l'état  $q_2$  et toutes les transitions le mettant en jeu sans modifier le langage reconnu. On obtient ainsi l'automate équivalent (non complet) représenté ci-dessous :



Ceci nous conduit aux définitions suivantes :

**Définition 4.** Un état  $q$  d'un AFD  $A = (\Sigma, Q, q_0, F, \delta)$  est dit

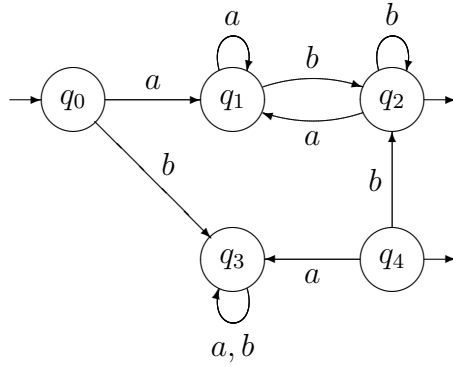
— accessible s'il existe un chemin dans  $A$  menant de l'état initial à  $q$  autrement dit s'il existe  $m \in \Sigma^*$  tel que  $\delta^*(q_0, m) = q$  ;

— co-accessible s'il existe un chemin dans  $A$  menant de  $q$  à un état acceptant c'est-à-dire s'il existe  $m \in \Sigma^*$  tel que  $\delta^*(q, m) \in F$  ;

— utile s'il est à la fois accessible et co-accessible.

**Remarque :** On peut remarquer que la lecture d'un mot reconnu par  $A$  à partir de l'état initial  $q_0$  ne passe que par des états utiles, donc on ne changera pas le langage reconnu en supprimant tous les états qui ne sont pas utiles ainsi que les transitions les impliquant. L'automate obtenu, qui ne possède plus que des états utiles est dit *émondé*.

**Exercice :** Pour l'automate représenté ci-dessous, préciser les états inutiles et déterminer l'automate émondé associé



## 2 Automates non déterministes

Nous allons généraliser la notion d'automate fini en leur permettant de posséder plusieurs états initiaux ou en autorisant deux transitions distinctes de même origine à avoir la même étiquette. Nous constaterons que ces automates non déterministes ne permettent pas de reconnaître plus de langages que les automates déterministes mais qu'ils ont l'avantage de posséder en général moins d'états qu'un automate déterministe équivalent.

**Définition 5.** Un automate fini non déterministe (*AFND* ou *NFA* pour *nondeterministic finite automaton*) est défini par un quintuplet  $A = (\Sigma, Q, I, F, \delta)$  où

- $\Sigma$  est un alphabet (*fini*) ;
- $Q$  est un ensemble fini dont les éléments sont appelés les états de  $A$  ;
- $I \subset Q$  est l'ensemble des états initiaux ;
- $F \subset Q$  est l'ensemble des états acceptants (ou finaux) ;
- $\delta$  est une application d'une partie de  $Q \times \Sigma$  dans  $\mathcal{P}(Q)$  appelée fonction de transition.

**Remarques :** – On notera que pour ces automates, la notion de blocage n'a plus vraiment de sens puisqu'on peut avoir  $\delta(q, a) = \emptyset$ . On peut d'ailleurs convenir que  $\delta$  est définie sur la totalité de  $Q \times \Sigma$  en prolongeant  $\delta$  en les couple  $(q, a)$  en lesquels elle n'est pas définie par  $\delta(q, a) = \emptyset$ .

– Une *transition* est un triplet  $(q, a, q')$  tel que  $q' \in \delta(q, a)$  ; elle sera encore représentée par  $q \xrightarrow{a} q'$ . Un *chemin* est une succession finie de transitions consécutives  $q_{i_0} \xrightarrow{a_1} q_{i_1} \xrightarrow{a_2} \dots \xrightarrow{a_n} q_{i_n}$  :  $a_1 a_2 \dots a_n$  est alors appelée l'*étiquette* de ce chemin.

**Définition 6.** Soit  $A$  un automate fini non déterministe.

- Un mot de  $\Sigma^*$  est dit *reconnu* par  $A$  s'il est l'étiquette d'un chemin dans  $A$  menant d'un état initial à un état acceptant.
- Le langage des mots reconnus par  $A$  est noté  $\mathcal{L}(A)$

**Remarques :** – Pour qu'un mot soit reconnu il suffit donc qu'il existe un chemin d'étiquette ce mot menant d'un état initial à un état final de l'automate. Donc si on veut montrer qu'un mot n'est pas reconnu, il faut tester si tous les chemins partant d'un état initial et d'étiquette ce mot aboutissent à un état non final.

– Comme pour les AFD, on peut pour un AFND  $A = (\Sigma, Q, I, F, \delta)$  définir la fonction de transition  $\delta^*$  étendue aux mots comme la fonction  $\delta^* : Q \times \Sigma^* \mapsto \mathcal{P}(Q)$  définie récursivement par :

- $\forall q \in Q, \delta^*(q, \varepsilon) = \{q\}$
- $\forall (q, m, a) \in Q \times \Sigma^* \times \Sigma, \delta^*(q, ma) = \bigcup_{q' \in \delta^*(q, m)} \delta(q', a).$

On a alors  $m \in \mathcal{L}(A) \iff \exists q_0 \in I, \delta^*(q_0, m) \cap F \neq \emptyset.$

## 2.1 Déterminisation

Nous allons montrer que pour tout automate non déterministe, il existe un automate déterministe équivalent, c'est-à-dire reconnaissant le même langage. La preuve que nous allons établir s'appelle *construction par sous-ensembles*.

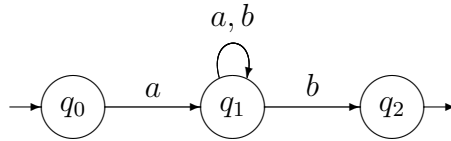
Considérons un AFND  $A = (\Sigma, Q, I, F, \delta)$  et posons :

$$A' = (\Sigma, \mathcal{P}(Q), I, F', \delta') \text{ où } F' = \{P \in \mathcal{P}(Q) \mid P \cap F \neq \emptyset\} \text{ et}$$

$$\forall (P, a) \in \mathcal{P}(Q) \times \Sigma, \delta'(P, a) = \bigcup_{q \in P} \delta(q, a)$$

On peut remarquer que si  $A$  a  $n$  états, l'automate déterministe  $A'$  a  $2^n$  états.

Pour comprendre cette construction mettons la en œuvre sur l'automate suivant - non déterministe - reconnaissant le langage des mots sur  $\Sigma = \{a, b\}$  commençant par  $a$  et terminant par  $b$  :



$A'$  possède alors  $2^3 = 8$  états :  $\emptyset, \{q_0\}, \{q_1\}, \{q_2\}, \{q_0, q_1\}, \{q_0, q_2\}, \{q_1, q_2\}, \{q_0, q_1, q_2\}$ . Son état initial est  $I = \{q_0\}$  et ses états finaux :  $\{q_2\}, \{q_0, q_2\}, \{q_1, q_2\}, \{q_0, q_1, q_2\}$ .

Le tableau des transitions de  $A'$  est :

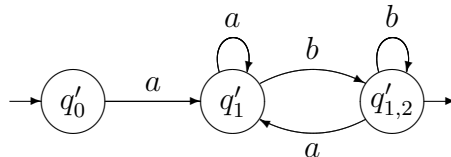
$\delta'$	$a$	$b$
$\emptyset$	$\emptyset$	$\emptyset$
$\{q_0\}$	$\{q_1\}$	$\emptyset$
$\{q_1\}$	$\{q_1\}$	$\{q_1, q_2\}$
$\{q_2\}$	$\emptyset$	$\emptyset$

$\delta'$	$a$	$b$
$\{q_0, q_1\}$	$\{q_1\}$	$\{q_1, q_2\}$
$\{q_1, q_2\}$	$\{q_1\}$	$\{q_1, q_2\}$
$\{q_0, q_2\}$	$\{q_1\}$	$\emptyset$
$\{q_0, q_1, q_2\}$	$\{q_1\}$	$\{q_1, q_2\}$

Pour éviter d'avoir à représenter un automate trop volumineux, on ne représente que les états accessibles : pour ce faire, on commence par remplir le tableau des transitions avec l'état initial et on n'ajoute que les états qui apparaissent comme résultats des transitions précédentes. On ne représente pas non plus l'état  $\emptyset$  qui n'est jamais co-accessible. En revanche, d'autres états non co-accessibles peuvent subsister. Dans notre exemple, cela conduit au tableau simplifié suivant :

$\delta'$	$a$	$b$
$\{q_0\}$	$\{q_1\}$	—
$\{q_1\}$	$\{q_1\}$	$\{q_1, q_2\}$
$\{q_1, q_2\}$	$\{q_1\}$	$\{q_1, q_2\}$

En posant  $q'_0 = \{q_0\}$ ,  $q'_1 = \{q_1\}$  et  $q'_{1,2} = \{q_1, q_2\}$  l'automate déterminisé (et ici émondé) est représenté par :



Revenons à la situation générale et établissons le résultat annoncé plus haut :

**Théorème** Avec les notations précédentes, les automates  $A$  et  $A'$  reconnaissent le même langage.

**Démonstration :** On montre par récurrence sur  $|u|$  que pour tout mot  $u \in \Sigma^*$ , il existe dans  $A$  un chemin étiqueté par  $u$  d'origine  $q_0 \in I$  et menant à un état  $q$  si et seulement s'il existe dans  $A'$  un chemin étiqueté par  $u$  menant de  $I$  à un état  $P$  contenant  $q$ .

- Dans  $A$  tout chemin d'origine  $q_0 \in I$  d'étiquette  $\varepsilon$  mène à  $q_0$  ; dans  $A'$  tout chemin étiqueté par  $\varepsilon$  d'origine  $I$  mène à  $I$  qui contient  $q_0$ . Le résultat énoncé est donc vrai pour un mot de longueur 0.
- Soit  $u \neq \varepsilon$  et supposons le résultat acquis pour tout mot de longueur strictement inférieure et posons  $u = a_1 a_2 \cdots a_n$ .

Soit  $q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} q_n$  un chemin dans  $A$  d'étiquette  $u$  avec  $q_0 \in I$ . D'après l'hypothèse de récurrence, il existe un chemin  $I \xrightarrow{a_1} P_1 \xrightarrow{a_2} \cdots \xrightarrow{a_{n-1}} P_{n-1}$  dans  $A'$  tel que  $q_{n-1} \in P_{n-1}$ . Mais alors  $q_n \in \delta(q_{n-1}, a_n)$  donc  $q_n \in \delta'(P_{n-1}, a_n)$ . En posant  $P_n = \delta'(P_{n-1}, a_n)$ , on met en évidence un chemin dans  $I \xrightarrow{a_1} P_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} P_n$  dans  $A'$  tel que  $q_n \in P_n$ .

Réciproquement soit  $I \xrightarrow{a_1} P_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} P_n$  est un chemin dans  $A'$  d'étiquette  $u$ . Notons  $q_n$  un élément de  $P_n$ . Il existe alors  $q_{n-1} \in P_{n-1}$  tel que  $q_n \in \delta(q_{n-1}, a_n)$ . Or par hypothèse de récurrence, il existe un chemin  $q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots \xrightarrow{a_{n-1}} q_{n-1}$  dans  $A$ , d'où l'existence d'un chemin  $q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} q_n$  dans  $A$  d'étiquette  $u$ .

On peut maintenant conclure : étant donné que  $F' = \{P \in \mathcal{P}(Q) \mid P \cap F \neq \emptyset\}$ , le résultat prouve qu'un mot est reconnu par  $A$  si et seulement si il est reconnu par  $A' \#$

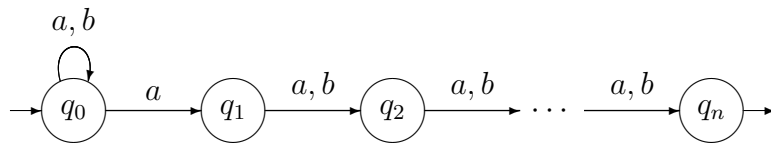
**Remarques :** Le résultat précédent montre que les automates finis non déterministes ne permettent pas de reconnaître plus de langages que les automates déterministes.

De plus, il est moins commode de vérifier qu'un mot n'est pas reconnu par un automate non déterministe puisqu'il est nécessaire de tester tous les chemins d'étiquette ce mot pour pouvoir conclure que le mot n'est pas reconnu, contrairement au cas d'un automate déterministe où on aura au plus un chemin étiqueté par un mot donné donc au plus un chemin à tester.

Toutefois, pour un langage donné, il est parfois plus simple de construire un automate non déterministe le reconnaissant. En outre, les automates non déterministes reconnaissant un langage donné ont souvent un nombre d'états plus petit que les automates déterministes le reconnaissant. Notons que, quand on détermine un automate par l'algorithme des parties décrit ci-dessus, on peut passer de  $n$  états à  $2^n$  états.

En pratique, si on dispose d'un automate non déterministe reconnaissant un langage donné, on le déterminisera, puis on émondera l'automate déterminisé, dans le but d'obtenir un automate déterministe reconnaissant le langage avec un nombre le plus réduit possible d'états. Il existe toutefois des situations où un langage reconnu par un AFND à  $n$  états ne peut être reconnu que par des AFD ayant un nombre d'états de l'ordre de  $2^n$  comme le montre l'exemple suivant.

**Exemple :** Considérons le langage  $\mathcal{L}$  des mots sur l'alphabet  $\Sigma = \{a, b\}$  de longueur supérieure ou égale à  $n$  et dont le suffixe de longueur  $n$  commence par la lettre  $a$  autrement dit  $\mathcal{L}$  est dénoté par l'expression rationnelle  $(a+b)^* a (a+b)^{n-1}$ . On trouve facilement un AFND à  $n+1$  états reconnaissant ce langage :



Montrons que tout automate fini déterministe reconnaissant le langage  $\mathcal{L}$  comporte au moins  $2^n$  états.

**Démonstration :** Soit  $A = (\Sigma, Q, q_0, F, \delta)$  un AFD reconnaissant  $\mathcal{L}$ . Notons pour tout  $u$  mot de longueur  $n$ ,  $q(u)$  l'état auquel aboutit le chemin dans  $A$  d'origine  $q_0$  et d'étiquette  $u$ . On définit ainsi une application  $q : \{a, b\}^n \rightarrow Q$ . Montrons que cette

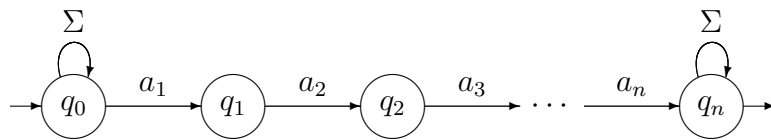
application  $q$  est injective ce qui prouvera que  $Q$  est de cardinal supérieur ou égal à  $2^n$ .

Raisonnons par l'absurde et supposons qu'il existe deux mots distincts  $u$  et  $v$  de longueur  $n$  tels que  $q(u) = q(v)$ . Notons  $s$  le plus grand suffixe commun à  $u$  et  $v$  de sorte que, sans perte de généralité, on peut supposer que  $u = u'as$  et  $v = v'bs$ . Le mot  $s$  est de longueur au plus  $n - 1$  et peut être complété en  $st$  mot de longueur  $n - 1$ . Dans ces conditions le mot  $m = ut = u'ast$  est reconnu par  $A$  tandis que le mot  $m' = vt = v'bst$  ne l'est pas. Pourtant comme  $q(u) = q(v)$  les chemins d'origine  $q_0$  et d'étiquettes  $ut$  et  $vt$  doivent mener au même état (ou être tous les deux bloquants) et puisque le premier est reconnu le second devrait l'être également.  $\#$

## 2.2 Application à la recherche d'un mot dans un texte : algorithme KMP

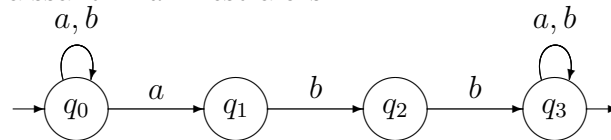
Pour illustrer l'utilité des automates pour résoudre certains problèmes, considérons celui de la recherche d'un mot dans un texte.

Étant donné un mot  $u$ , on souhaite considérer un automate caractérisant la présence de ce mot dans un texte, autrement dit un automate reconnaissant le langage  $\Sigma^*u\Sigma^*$ . Si  $u = a_1a_2 \dots a_n$ , on construit facilement l'automate - non déterministe - ci dessous qui répond à la question.



Cependant, si on veut effectivement utiliser ce formalisme pour traiter un texte, on aura intérêt à avoir un automate déterministe équivalent. On applique donc l'algorithme de déterminisation à l'automate précédent.

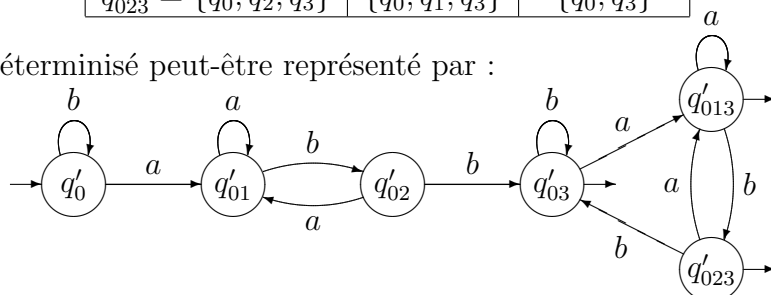
**Exemple :** Considérons le mot  $u = abb$  sur l'alphabet  $\Sigma = \{a, b\}$ . L'automate non déterministe reconnaissant  $\Sigma^*u\Sigma^*$  est alors



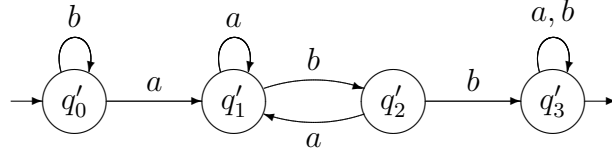
L'algorithme de déterminisation conduit au tableau suivant des transitions utiles :

$\delta'$	$a$	$b$
$q'_0 = \{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$q'_{01} = \{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$q'_{02} = \{q_0, q_2\}$	$\{q_0, q_1\}$	$\{q_0, q_3\}$
$q'_{03} = \{q_0, q_3\}$	$\{q_0, q_1, q_3\}$	$\{q_0, q_3\}$
$q'_{013} = \{q_0, q_1, q_3\}$	$\{q_0, q_1, q_3\}$	$\{q_0, q_2, q_3\}$
$q'_{023} = \{q_0, q_2, q_3\}$	$\{q_0, q_1, q_3\}$	$\{q_0, q_3\}$

L'automate déterminisé peut-être représenté par :



On constate que l'automate obtenu n'est pas minimal, les trois états acceptants pouvant être regroupés en un seul sans modification du langage reconnu, d'où l'automate déterministe plus simple reconnaissant  $\Sigma^*abb\Sigma^*$  :



L'algorithme KMP (du nom de ses inventeurs KNUTH, MORRIS et PRATT) permet d'obtenir mécaniquement un automate déterministe reconnaissant  $\Sigma^*u\Sigma^*$  avec un nombre plus réduit d'états. Pour cela, on note  $P(u)$  l'ensemble des préfixes du mot  $u$  et si  $v$  est un mot, on note  $s(v)$  le plus grand suffixe de  $v$  appartenant à  $P(u)$ .

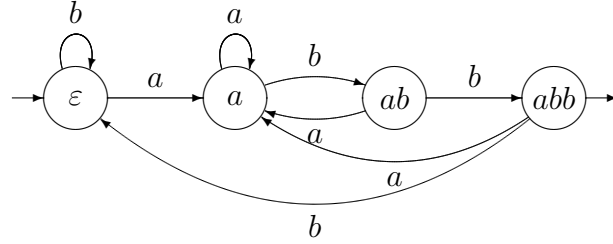
On considère ensuite l'automate déterministe  $A = (\Sigma, P(u), \{\varepsilon\}, \{u\}, \delta)$  où la fonction de transition  $\delta$  est définie par :

$$\forall p \in P(u), \forall x \in \Sigma, \delta(p, x) = s(px)$$

**Exemple :** Lorsque  $u = abb$ , on obtient alors le tableau des transitions

$\delta$	$a$	$b$
$\varepsilon$	$a$	$\varepsilon$
$a$	$a$	$ab$
$ab$	$a$	$abb$
$abb$	$a$	$\varepsilon$

et l'automate



Pour un mot  $u$  quelconque, on a le résultat suivant :

**Proposition :** Le langage reconnu par l'automate  $A$  est  $\Sigma^*u$

**Démonstration :** Notons  $\delta^*$  la fonction de transition étendue aux mots. Montrons par récurrence la propriété :

$$H(n) : \forall p \in P(u), \forall v \in \Sigma^* \text{ de longueur } n, \delta^*(p, v) = s(pv)$$

- $H(0)$  est vraie car si  $p \in P(u)$ ,  $\delta^*(p, \varepsilon) = p$  et  $s(p\varepsilon) = s(p) = p$  puisque  $p \in P(u)$
- soit  $n \in \mathbb{N}$  tel que  $H(n)$  soit vraie. Soit  $p \in P(u)$  et  $v$  un mot de longueur  $n + 1$ ,  $v = v'a$  avec  $a \in \Sigma$ . D'après  $H(n - 1)$ ,  $\delta^*(p, v') = s(pv')$  donc  $\delta^*(p, v) = \delta(s(pv'), a) = s(s(pv')a)$ . Il s'agit donc de montrer que  $s(s(pv')a) = s(pv)$ . Posons  $w' = s(pv')$  et  $w = s(w'a)$ .

$w$  est suffixe de  $w'a$  et  $w'$  est suffixe de  $pv'$  donc  $w$  est suffixe de  $pv'a = pv$ . De plus,  $w \in P(u)$  donc  $w$  est suffixe de  $s(pv)$  (plus long suffixe de  $pv$  qui soit dans  $P(u)$ )

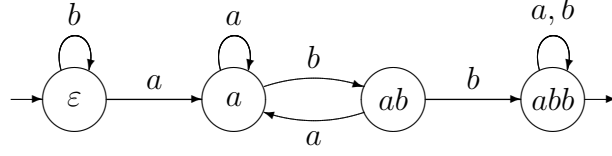
Si  $s(pv) = \varepsilon$  alors  $w = \varepsilon$ . Sinon, il existe  $x \in \Sigma^*$  tel que  $s(pv) = xa$  (car  $v = v'a$ ). Alors  $x \in P(u)$  et  $x$  suffixe de  $pv'$  donc  $x$  suffixe de  $s(pv') = w'$  et  $xa$  suffixe de  $w'a$ . Puisque  $xa \in P(u)$ , alors  $xa$  est suffixe de  $s(w'a) = w$ . Dans les deux cas, on a montré que  $s(pv)$  est suffixe de  $w$  donc finalement  $w = s(pv)$

Ainsi, si  $v \in \Sigma^*$ ,  $v \in \mathcal{L}(A)$  équivaut à  $\delta^*(\varepsilon, v) = u$  ou encore à  $s(v) = u$  soit à  $v \in \Sigma^*u$



**Corollaire :** Pour obtenir un automate qui reconnait  $\Sigma^*u\Sigma^*$ , il suffit de considérer l'automate  $A$  et de transformer l'état  $u$  en un puits.

**Exemple :** Lorsque  $u = abb$ , l'algorithme KMP conduit à l'automate suivant :



On constate, qu'au changement de nom des états près, on retombe sur l'automate qu'on avait déterminé plus haut.

### 3 Automates finis et langages rationnels

Dans cette section, nous allons étudier le théorème fondamental de ce chapitre, qui affirme l'égalité entre l'ensemble des langages rationnels et l'ensemble des langages reconnaissables par un automate fini.

#### Théorème (KLEENE)

Un langage  $L$  sur un alphabet  $\Sigma$  est rationnel si et seulement s'il existe un automate fini  $A$  tel que  $L(A) = L$

Dans le paragraphe suivant, nous montrons que tout langage rationnel est reconnaissable par automate en donnant un algorithme construisant explicitement un automate (l'automate de GLUSHKOV) reconnaissant le langage dénoté par une expression rationnelle. Nous admettrons la réciproque, moins utile en pratique, conformément au programme. Nous donnerons ensuite des propriétés des langages reconnaissables pour les étendre aux langages rationnels.

#### 3.1 Algorithme de BERRY-SETHI

##### • Cas d'un langage local

Nous allons montrer que tout langage local est reconnaissable par un automate fini. Donnons tout d'abord deux définitions concernant les automates.

**Définition 7.** Un automate déterministe  $A = (\Sigma, Q, q_0, F, \delta)$  est dit local si pour tout  $x$  de  $\Sigma$ , toutes les transitions d'étiquette  $x$  aboutissent au même état. Il est dit standard si aucune transition n'aboutit à l'état initial  $q_0$ .

**Proposition :** Tout langage local est reconnaissable par un automate local standard

**Démonstration :** Soit  $L = L_{I,B,F,b}$  un langage local sur l'alphabet  $\Sigma$ .

*Premier cas :* le mot vide  $\varepsilon$  n'appartient pas à  $L$ .

Posons alors  $A = (\Sigma, \Sigma \cup \{\varepsilon\}, \varepsilon, F, \delta)$  l'automate dont la fonction de transition est définie par :

$$\forall x \in I, \delta(\varepsilon, x) = x \text{ et } \forall xy \in B, \delta(x, y) = y$$

Précisons que cet automate ne comporte aucune autre transition que celles que nous venons de décrire.

Alors  $\varepsilon \notin L(A)$  et pour un mot non vide  $m = a_1 \dots a_p$ ,  $m \in L(A)$  si et seulement si  $\varepsilon \xrightarrow{a_1} a_1, a_1 \xrightarrow{a_2} a_2, \dots, a_{p-1} \xrightarrow{a_p} a_p$  sont des transitions de  $A$  et que  $a_p \in F$  c'est-à-dire si  $a_1 \in I, \forall i \in \llbracket 1, p-1 \rrbracket, a_i a_{i+1} \in B$  et  $a_p \in F$  c'est-à-dire si  $m \in L_{I,B,F,b} = L$ .

On a donc  $L = L(A)$  et  $A$  est bien local et standard.

*Deuxième cas :* si  $\varepsilon \in L$ , il suffit de modifier l'automate précédent en ajoutant  $\varepsilon$  aux

états acceptants donc de prendre  $A = (\Sigma, \Sigma \cup \{\varepsilon\}, \varepsilon, F \cup \{\varepsilon\}, \delta)$  où  $\delta$  est la même fonction de transition que précédemment  $\sharp$

On en déduit le corollaire suivant :

**Corollaire :** Tout langage dénoté par une expression rationnelle linéaire est reconnaissable par un automate local standard.

**Démonstration :** En effet, nous avons montré dans le chapitre précédent que tout langage dénoté par une expression rationnelle linéaire (c'est-à-dire où toute lettre de  $\Sigma$  apparaît au plus une fois) est local  $\sharp$

**Exemple :** Représenter un automate local standard reconnaissant le langage dénoté par  $(a + b)^*c$