Les Automates

Chapitre 5

1 Automates finis

1.1 Automates finis déterministes

On a défini des langages mais on aimerait avoir un moyen simple de déterminer si un mot appartient à langage donné (au moins dans le cas où le langage est un langage rationnel). Nous allons voir la notion d'automates finis qui permet de faire cela. Le principe général est le suivant :

— Un automate possède un ou plusieurs états

— Chaque lettre de l'alphabet engendre une transition qui permet d'aller d'un état donné à un autre état (ou des autres états voir plus loin)

— Il y a des états initiaux et des états finaux.

Définition 1.1.1 (Automate fini déterministe)

Un automate (fini) déterministe ou AFD sur l'alphabet $\mathscr{A}$ est un quadruplet $A = (Q, q_0, F, \delta)$ où

— *L'ensemble Q est un ensemble fini. C'est l'ensemble des états de l'automate.*

— *L'élément q_0 est un élément de Q. Il s'appelle l'état initial.*

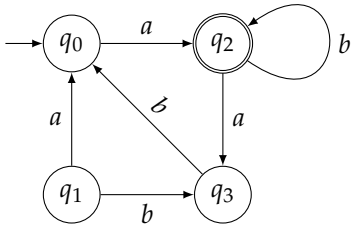
— *L'ensemble F est une partie de Q. Ses éléments s'appellent les états finals (ou acceptants)*

— *L'application δ est définie sur une partie de $Q \times \mathscr{A}$ et à valeurs dans Q. C'est la fonction de transition.*

Exemple : On veut définir un automate sur $\mathscr{A} = \{a, b\}$. On se donne $Q = \{q_0, q_1, q_2, q_3\}$ avec q_0 l'état initial, $F = \{q_2\}$ et δ donnée par

	q_0	q_1	q_2	q_3
a	q_2	q_0	q_3	
b		q_3	q_2	q_0

Afin de mieux visualiser l'automate on le représente ainsi :



— La flèche d'entrée indique l'état initial

— On désigne les états acceptants par un double cercle. On trouve aussi une flèche sortante vers rien.

Exemple : On peut par exemple penser au monnayeur d'une machine :

—

—

—

—

Terminologie

Soit (Q, q_0, F, δ) un automate fini deterministe :

— Un blocage de l'automate est un couple $(q, x) \in Q \times \mathscr{A}$ tel que $\delta(q, x)$ n'est pas défini.

— Un automate sans blocages est dit complet.

— L'automate est dit standard si pour tout $(q, x) \in Q \times \mathscr{A}$, $\delta(q, x) \neq q_0$. C'est-à-dire que l'automate ne « retombe » pas dans l'état initial.

Remarque : Nous verrons plus loin l'intérêt des automates standards.

1.2 Calculs dans un automate déterministe

Nous allons maintenant « calculer » dans l'automate. Cela consiste à lire un mot de $\mathscr{A}^\star$ en parcourant en conséquence les différents états de l'automates.

Pour cela on va étendre la fonction de transition à des mots de $\mathscr{A}^\star$ en faisant évoluer l'état de l'automate à chaque lettre du mot.

Définition 1.2.2

Soit $A = (Q, q_0, F, \delta)$ un automate fini déterministe sur $\mathscr{A}$ on définit une application de transition notée $\delta^\star : Q \times \mathscr{A}^\star \to Q$ par :

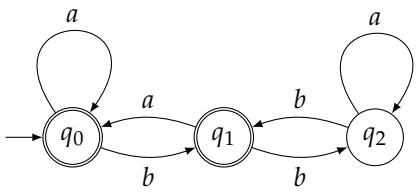
—

—

Remarques :

1. Il arrive que l'on note aussi simplement δ pour $\delta^\star$.
2. Si l'automate est complet, la fonction $\delta^\star$ est définie pour tout $q \in Q$ et tout $w \in \mathscr{A}^\star$.
3. Pour $w \in \mathscr{A}^\star$, on appelle donc calcul dans l'automate sur le mot $w = w_0 w_1 \cdots w_{n-1}$ issu de l'état q un « chemin » dans l'automate :
$$q = \alpha_0 \xrightarrow{w_0} \alpha_1 \xrightarrow{w_1} \cdots \xrightarrow{w_{n-1}} \alpha_n$$
où pour tout $i \in [\![0\,;\, n-1]\!]$, $\delta(\alpha_i, w_i) = \alpha_{i+1}$. On a alors $\delta^\star(q, w) = \alpha_n$.

Exemple : Considérons l'automate complet suivant



On a $\delta^\star(q_0, abba) = q_2$ et $\delta^\star(q_1, aabb) =$

Définition 1.2.3 (Langage reconnu par un automate)

Soit $A = (Q, q_0, F, \delta)$ un automate fini déterministe.

1. *Un mot w sur $\mathscr{A}$ est dit reconnu par l'automate si $\delta^\star(q_0, w)$ est défini et appartient à F.*
2. *L'ensemble des mots reconnus par l'automate A se note $L(A)$. C'est le langage reconnu par l'automate.*

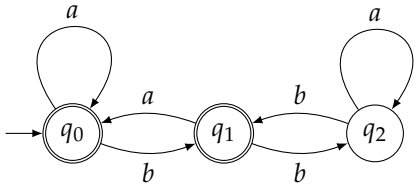
Remarques :

1. Le but d'un automate est de reconnaitre des mots.
2. On dit aussi que les mots reconnus sont les mots acceptés.
3. Un mot $w \in \mathscr{A}^\star$ **n'est pas reconnu** par A si

—

—

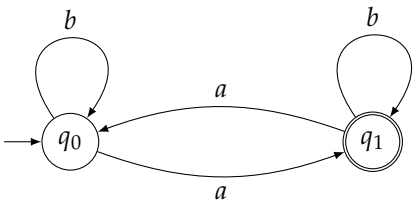
Exemple : Si on reprend l'automate précédent



On voit que a, b, aab sont reconnus, par contre $abba$ n'est pas reconnu.

Exercices :

1. Soit A l'automate



Déterminer des mots reconnus par A et des mots qui ne sont pas reconnus. Quel est le langage reconnu par l'automate ?

2. Déterminer un automate dont le langage reconnu est $a\mathscr{A}^\star$.

Proposition 1.2.4

Tout langage reconnu par un automate fini déterministe est reconnu par un automate fini déterministe complet

Démonstration : Soit L un langage et $A = (Q, q_0, \delta, F)$ un automate qui le reconnait. On peut construire un automate complet A' tel que $L(A') = L(A) = L$. Il suffit d'ajouter à Q un nouvel état p que l'on appelle un état puit. On pose $A' = (Q \cup \{p\}, q_0, \delta', F)$ où δ' est définie par :

— Pour $q \in Q$ et $x \in \mathscr{A}$ tel que (q, x) n'est pas un blocage dans A, $\delta'(q, x) = \delta(q, x)$.

— Pour $q \in Q$ et $x \in \mathscr{A}$ tel que (q, x) est un blocage dans A, $\delta'(q, x) = p$.

— Pour $x \in \mathscr{A}$, on pose $\delta'(p, x) = p$.

Prouvons alors proprement que $L(A) = L(A')$.

— Soit $w = w_0 w_1 \dots w_n \in L(A)$. Il existe un calcul dans A

$$q_0 \xrightarrow{w_0} \cdots \xrightarrow{w_n} q$$

où $q \in F$. Ce calcul est aussi possible dans A' donc $w \in L(A')$. Finalement $\boxed{L(A) \subset L(A')}$.

— Soit $w \notin L(A)$. Il y a deux cas :

— Le calcul dans A est impossible à cause d'un blocage. Dans ce cas, $(\delta')^\star(q_0, w) = p \notin F$.

— Le calcul est possible dans A mais $\delta(q_0, w) \notin F$. Dans ce cas, le calcul est identique dans A' et donc on a encore $(\delta')^\star(q_0, w) \notin F$.

Dans les deux cas $w \notin L(A')$. Finalement, par contraposée, $\boxed{L(A') \subset L(A)}$.

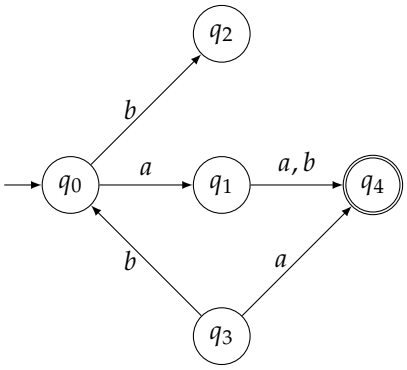
On a bien construit A' complet qui reconnait L. □

Définition 1.2.5 (Langage reconnaissable)

Un langage L est dit reconnaissable s'il existe un automate fini déterministe (que l'on peut supposer complet) A tel que $L = L(A)$.

1.3 Emondage

Quand on considère un automate A qui reconnait un langage L, on ne s'intéresse qu'aux états que l'on va « atteindre » à partir de l'état initial lors d'un calcul. Par exemple dans l'automate



— On ne peut pas atteindre l'état q_3 en partant de q_0

— Si on est à q_2 on ne peut plus rejoindre l'état final q_4.

Définition 1.3.6

Soit $A = (Q, q_0, F, \delta)$ un automate fini deterministe.

— *Un état $q \in Q$ est dit **accessible** s'il existe un mot $w \in \mathscr{A}^\star$ tel que $\delta(q_0, w) = q$. Autrement dit, si on peut atteindre le sommet q en partant du sommet initial.*

— *Un état $q \in Q$ est dit **co-accesible** s'il existe un mot $w \in \mathscr{A}^\star$ tel que $\delta(q, w) \in F$. Autrement dit, si on peut atteindre un sommet acceptant en partant du sommet q.*

L'automate est dit émondé si tous ses états sont accessibles et co-accessibles.

Remarque : 'A tout automate $A = (Q, q_0, F, \delta)$ reconnaissant un langage non vide (c'est-à-dire que q_0 est co-accessible) on lui associe l'automate émondé $A' = (Q', q_0, F \cap Q', \delta')$ où

Proposition 1.3.7

L'automate A et l'automate A' reconnaissent le même langage.

1.4 Automates finis non-déterministes

Dans certains cas, il va être plus facile de construire des automates d'un type différent.
Commençons par un exemple.
On veut construire un automate reconnaissant le langage $L = \mathscr{A}^\star aab$ des mots qui se terminent par aab.
On peut le faire avec un automate déterministe :

Mais on peut le faire plus simplement

Dans les automates que nous avons vus, il y avait une **unique** transition à un état fixé et une lettre fixée. Nous allons regarder maintenant des automates non-déterministes où, d'un état donné et avec une lettre donnée on peut passer à plusieurs états.

Définition 1.4.8 (Automate fini non déterministe)

Un automate (fini) non déterministe ou AFND sur l'alphabet $\mathscr{A}$ est un quadruplet $A = (Q, I, F, \delta)$ où

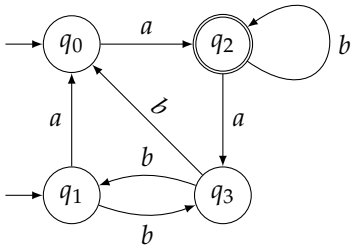
- *L'ensemble Q est un ensemble fini. C'est l'ensemble des états de l'automate.*
- *L'ensemble I est une partie de Q. Ce sont les états initiaux*
- *L'ensemble F est une partie de Q. C'est l'ensemble des états finaux (ou acceptants)*
- *L'application δ est définie sur $Q \times \mathscr{A}$ et à valeurs dans $\mathscr{P}(Q)$. C'est la fonction de transition.*

Remarques :

1. Il y a deux différences (qui ne sont qu'une seule) entre un automate déterministe et un automate non déterministe. La première est que dans ce dernier il peux y avoir plusieurs états initiaux. La deuxième est que, partant d'un état donné, une même lettre nous permet d'atteindre zéro, un ou plusieurs états donnés par une partie de Q.
2. Il n'y a plus de blocage au sens précédent, cependant, $\delta(q, x)$ peut être vide.

ATTENTION

Exemple : On les représente comme précédemment



Définition 1.4.9

Soit $A = (Q, I, F, \delta)$ un automate (fini) non déterministe.

1. *On étend la fonction δ à l'ensemble des mots de $\mathscr{A}^\star$ comme précédemment. Précisément on construit $\delta^\star : Q \times \mathscr{A}^\star \to \mathscr{P}(Q)$ par*

$$\forall q \in Q, \delta^*(q, \varepsilon) = \{q\} \text{ et } \forall a \in \mathscr{A}, \forall w \in \mathscr{A}^\star, \delta^\star(q, aw) = \bigcup_{q' \in \delta(q,a)} \delta^\star(q', w).$$

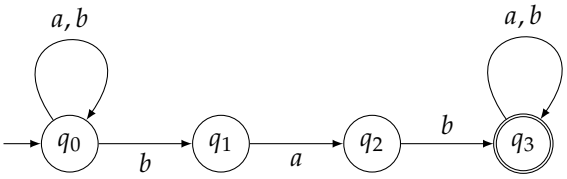
2. *Soit w un mot de $\mathscr{A}^\star$, il est reconnu par l'automate s'il existe $q_0 \in I$ tel que $\delta(q_0, w) \cap F \neq \emptyset$.*
   On note encore $L(A)$ le langage des mots reconnus par l'automate A.

ATTENTION

Exemple : Si on reprend l'automate précédent, les mots a, aa, bba sont reconnus. Par contre ba ne l'est pas.

Exercices :

1. On considère l'automate suivant :



Déterminer des mots acceptés par l'automate et des mots qui ne sont pas acceptés. Quel est le langage des mots reconnus par l'automate ?

2. Déterminer un automate (éventuellement non déterminé) reconnaissant les mots finissant par ab.

1.5 Quelques considérations sur la complexité - Hors programme

ATTENTION

Proposition 1.5.10

Soit A un automate.

1. On suppose que A est déterministe. L'appartenance d'un mot w à $L(A)$ à pour complexité :
   - Complexité temporelle :
   - Complexité en espace :

   Dans le cas général. L'appartenance d'un mot w à $L(A)$ à pour complexité :
   - Complexité temporelle :
   - Complexité en espace :

Démonstration :

1.6 Déterminisation

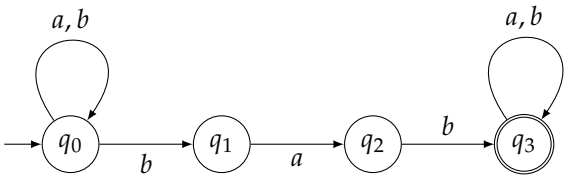
Soit L un langage reconnu par un automate A non déterministe. On vient de voir qu'il est préférable d'avoir un automate déterministe reconnaissant ce langage. On va donc construire cet automate déterministe. L'idée principale est de remplacer l'ensemble Q des états par $\mathscr{P}(Q)$.

Définition 1.6.11 (Automate des parties)

Soit $A = (Q, I, F, \delta)$ un automate non déterministe. On appelle automate des parties l'automate déterministe $A' = (Q', I, F', \delta')$ où :

- $Q' = \mathscr{P}(Q)$.
- $F' = \{X \in \mathscr{P}(Q) \mid X \cap F \neq \emptyset\}$
- *L'application δ' définie par* $\forall X \in \mathscr{P}(Q), \forall x \in \mathscr{A}$,

Remarque : Notons qu'ici I est un **élément** de $\mathscr{P}(Q)$.
Exemple : Reprenons l'automate



Il a 4 états ce qui fait que l'automate des parties aura $2^4 = 16$ états. Cependant, de nombreux états ne seront pas accessibles. En pratique :

—

—

—

état	a	b
$\{q_0\}$	$\{q_0\}$	$\{q_0, q_1\}$

Les états acceptants sont

Ce qui donne l'automate

Proposition 1.6.12

Soit $A = (Q, I, F, \delta)$ un automate non déterministe et A' l'automate des parties : $A' = (\mathscr{P}(Q), I, F', \delta')$.

1. L'automate A' est déterminé
2. Les automates A et A' reconnaissent les mêmes langages

Démonstration :

1. Par construction
2. Soit $w \in \mathscr{A}^\star$, on veut vérifier que $w \in L(A) \iff w \in L(A')$.
   On sait que
   $$w \in L(A) \iff \exists q_0 \in I, \delta^\star(q_0, w) \cap F \neq \emptyset$$
   et
   $$w \in L(A') \iff (\delta')^\star(I, w) \in F'.$$
   Maintenant, $(\delta')^\star(I, w) = \bigcup_{q_0 \in I} \delta^\star(q_0, w)$ donc
   $$\begin{aligned} (\delta')^\star(I, w) \in F' &\iff \left(\bigcup_{q_0 \in I} \delta^\star(q_0, w) \right) \cap F \neq \emptyset \\ &\iff \exists q_0 \in I, \delta^\star(q_0, w) \cap F \neq \emptyset \end{aligned}$$

□

Corollaire 1.6.13

Tout langage L reconnu par un automate non déterministe est reconnu par un automate déterministe.
Cela signifie que l'ensemble des langages reconnaissables est l'ensemble des langages L tels qu'il existe un automate (pas nécessairement déterministe) fini A tel que $L = L(A)$.

Remarque : On peut donc déterminiser un automate afin de pouvoir dire plus aisément / rapidement si un mot appartient (ou non) à un langage donné. Cependant, il faut prendre en compte la complexité de la déterminisation et surtout le fait qu'il peut y avoir une explosion du nombre d'états. A priori, le nombre d'états de l'automate des parties croit exponentiellement en fonction du nombre d'états de l'automate de départ. Nous verrons par exemple en exercice un langage reconnu par un automate non déterministe à $N + 2$ états mais qui n'est pas reconnu par un automate déterministe avec moins de 2^{N+1} états.

2 Théorème de Kleene

2.1 Enoncé et utilisation

Nous allons maintenant faire le lien entre les langages rationnels (ou réguliers) et les automates finis.

Théorème 2.1.14 (Théorème de Kleene)

Tout langage rationnel est reconnaissable.

Remarques :

1. Cela signifie que si L est rationnel, il existe un automate fini (que l'on peut supposer déterministe par déterminisation) A tel que pour tout mot w de $\mathscr{A}^*$, $w \in L$ si et seulement si w est reconnu par A.
2. La réciproque est aussi vraie (tout langage reconnaissable est rationnel) mais elle n'est pas au programme.
3. La méthode de démonstration est constructive : on va construire une automate (automate de Glushkov) qui reconnait un langage rationnel donné.

★ **Méthode :** Avant de se lancer dans la démonstration qui est longue, voyons que ce théorème permettra de montrer que des langages ne sont pas rationnels.

L'argument principal est qu'un automate reconnaissant un langage n'aura qu'un nombre fini d'états.

Prenons par exemple le langage $L = \{a^p b^p \mid p \in \mathbf{N}\}$. Montrons qu'il n'est pas rationnel.

Nous allons donner deux rédactions plus ou moins équivalentes.

—

—

— Note :

Remarque : On reformulera ceci à la fin du chapitre avec le lemme de l'étoile.

Exercice : Montrer que le langage des palindromes sur $\mathscr{A} = \{a, b\}$ n'est pas rationnel

2.2 Langages locaux

2.2.1 Définition

Nous aurons besoin par la suite d'une sous-classe de langages : les langages *locaux*. Ce sont les langages qui sont tels que l'on puisse déterminer si un mot w appartienne au langage juste en regardant, la première lettre, la dernière lettre et les facteurs de longueurs 2 (d'où le caractère local).

Définition 2.2.15

Soit L un langage sur un alphabet $\mathscr{A}$. On définit :

- $P(L) = \{a \in \mathscr{A} \mid a\mathscr{A}^\star \cap L \neq \emptyset\}$ *l'ensembles des lettres qui apparaissent en tête d'un mot de L (les préfixes)*
- $S(L) = \{a \in \mathscr{A} \mid \mathscr{A}^\star a \cap L \neq \emptyset\}$ *l'ensembles des lettres qui apparaissent en fin d'un mot de L (les suffixes)*
- $F(L) = \{u \in \mathscr{A}^2 \mid \mathscr{A}^\star u \mathscr{A}^\star \cap L \neq \emptyset\}$ *l'ensembles des facteurs de longueur 2 d'un mot de L*
- $N(L) = \mathscr{A}^2 \setminus F(L)$ *les mots de longueur 2 qui ne sont pas des facteurs de mots de L*

Proposition 2.2.16

Soit L un langage,

$$L \setminus \{\varepsilon\} \subset (P(L)\mathscr{A}^\star \cap \mathscr{A}^\star S(L)) \setminus \mathscr{A}^\star N(L)\mathscr{A}^\star$$

Démonstration : Par définition □

ATTENTION

Avec les notations précédentes,

Définition 2.2.17

*Soit L un langage, il est dit **local** s'il existe des parties P et S de $\mathscr{A}$ et une partie N de $\mathscr{A}^2$ telles que*

$$L \setminus \{\varepsilon\} = (P\mathscr{A}^\star \cap \mathscr{A}^\star S) \setminus \mathscr{A}^\star N \mathscr{A}^\star$$

Remarques :

1. Si L est local, nécessairement, $P = P(L)$, $S = S(L)$ et $N = N(L)$.
2. On préfère souvent définir le langage par le triplet (P, S, F) plutôt que (P, S, N), c'est-à-dire avec les facteurs que l'on accepte plutôt que ceux que l'on refuse.

Exemples :

1. Le langage $L = \{a^n b^m \mid (n, m) \in \mathbf{N}^2\}$ est un langage local.

2. Le langage $L = (ab)\mathscr{A}^\star$ des mots qui commencent par ab n'est pas local.

On veut savoir si l'ensemble des langages locaux est stable par les opérations ensemblistes sur les langages : intersection, union, concaténation et étoile.

2.2.2 Intersection

Proposition 2.2.18

L'intersection de deux langages locaux est un langage local.

Démonstration : Soit L_1 et L_2 deux langages locaux. On note pour $i \in \{1,2\}$ P_i, S_i et N_i les préfixes, suffixes et facteurs interdits de L_i. Soit $w \in \mathscr{A}^\star$

$$\begin{aligned} w \in L_1 \cap L_2 \setminus \{\varepsilon\} &\iff w \in (L_1 \setminus \{\varepsilon\}) \cap (L_2 \setminus \{\varepsilon\}) \\ &\iff w \in (P_1\mathscr{A}^\star \cap \mathscr{A}^\star S_1) \cap (P_2\mathscr{A}^\star \cap \mathscr{A}^\star S_2) \\ &\qquad \text{et } w \notin \mathscr{A}^\star N_1 \mathscr{A}^\star \text{ et } w \notin \mathscr{A}^\star N_2 \mathscr{A}^\star \\ &\iff w \in (P_1 \cap P_2)\mathscr{A}^\star \cap \mathscr{A}^\star (S_1 \cap S_2) \\ &\qquad \text{et } w \notin \mathscr{A}^\star (N_1 \cup N_2)\mathscr{A}^\star \\ &\iff w \in ((P_1 \cap P_2)\mathscr{A}^\star\mathscr{A}^\star \cap (S_1 \cap S_2)) \setminus \mathscr{A}^\star (N_1 \cup N_2)\mathscr{A}^\star \end{aligned}$$

Finalement,

$$L_1 \cap L_2 \setminus \{\varepsilon\} = (P\mathscr{A}^\star \cap \mathscr{A}^\star S) \setminus \mathscr{A}^\star N \mathscr{A}^\star$$

où $P = P_1 \cap P_2, S = S_1 \cap S_2$ et $N = N_1 \cup N_2$ (et donc $F = \overline{N} = F_1 \cap F_2$). Le langage $L_1 \cap L_2$ est bien local. □

2.2.3 Union et concaténation

Exercice : Montrer que les langages $L_1 = L(a\star)$ et $L_2 = L((ab)\star)$ sont locaux mais que ni L_1L_2 ni $L_1 \cup L_2$ ne le sont

On a cependant un résultat

Proposition 2.2.19

Soit L_1 et L_2 deux langages locaux **sur des alphabets disjoints** alors $L_1 \cup L_2$ et $L_1.L_2$ sont des langages locaux.

Démonstration :

— On note comme d'habitude, pour $i \in \{1,2\}$ P_i, S_i et F_i les préfixes, suffixes et facteurs autorisés de L_i. On note $\mathscr{A}_1$ (resp. $\mathscr{A}_2$) l'alphabet sur lequel est défini L_1 (resp. L_2) ainsi que $\mathscr{A} = \mathscr{A}_1 \cup \mathscr{A}_2$.
On veut montrer que le langage $L_1 \cup L_2$ sur l'alphabet $\mathscr{A}$ est local. On pose $P = P(L_1 \cup L_2) = P_1 \cup P_2$, $S = S(L_1 \cup L_2) = S_1 \cup S_2$ et $F = F(L_1 \cup F_2) = F_1 \cup F_2$. On veut montrer que

$$L_1 \cup L_2 \setminus \{\varepsilon\} = (P\mathscr{A}^\star \cap \mathscr{A}^\star S) \setminus \mathscr{A}^\star N \mathscr{A}^\star$$

où $N = (\mathscr{A}^\star)^2 \setminus F$.
Si on considère un mot $w = w_1 \dots w_n =\in (P\mathscr{A}^\star \cap \mathscr{A}^\star S) \setminus \mathscr{A}^\star N \mathscr{A}^\star$ alors $w_1 \in P = P_1 \cup P_2$. Supposons (par symétrie) que $w_1 \in P_1 \subset \mathscr{A}_1$. Maintenant, $w_1 w_2 \in F = F_1 \cup F_2$ mais comme $w_1 \in \mathscr{A}_1$ et que L_1 et L_2 sont définis sur des alphabets distincts, nécessairement $w_1 w_2 \in F_1$ et donc $w_2 \in \mathscr{A}_1$. On montre ainsi, de proche en proche, que $w \in \mathscr{A}_1^\star$. Il appartient alors à $(P_1\mathscr{A}_1^\star \cap \mathscr{A}_1^\star S_1) \setminus \mathscr{A}_1^\star N_1 \mathscr{A}_1^\star = L_1 \setminus \{\varepsilon\}$ car L_1 est local.

— Gardons les mêmes notations. On pose cette fois

$$P = P(L_1.L_2) = \begin{cases} & \text{si } \varepsilon \notin L_1 \\ & \text{si } \varepsilon \in L_1 \end{cases}$$

De même on pose

$$S = S(L_1.L_2) = \begin{cases} & \text{si } \varepsilon \notin L_2 \\ & \text{si } \varepsilon \in L_2 \end{cases}$$

Pour finir, on pose

$$F =$$

On procède alors de même soit $w = w_1 w_2 \cdots w_n \in (P\mathscr{A}^\star \cap \mathscr{A}^\star S) \setminus \mathscr{A}^\star N \mathscr{A}^\star$.

— Si $w \in \mathscr{A}_2^\star$ alors $w_1 \in P_2$ (c'est donc que $\varepsilon \in L_1$). On a aussi $w_n \in S_2$ et tous les facteurs sont dans F_2 donc $w \in L_2 \subset L_1.L_2$ car $\varepsilon \in L_1$.

— Si $w \in \mathscr{A}_1^\star$ on procède de même (cette fois $\varepsilon \in L_2$).

— Si $w \notin (\mathscr{A}_1^\star \times \mathscr{A}_2^\star)$. On a alors $w_1 \in P_1$ et $w_n \in S_2$ car si une lettre w_k appartient à $\mathscr{A}_2$ toutes celles qui suivent aussi d'après la forme des facteurs autorisés. On note $u = w_1 \cdots w_p$ le plus grand préfixe appartenant à $\mathscr{A}_1^\star$. On a alors $w_p \in \mathscr{A}_1$ et $w_{p+1} \in \mathscr{A}_2$ donc le facteur $w_p w_{p+1} \in S_1 \times P_2$. On montre alors que $u \in L_1$ et $w_{p+1} \cdots w_n \in L_2$.

Dans tous les cas, $w \in L_1.L_2$.

2.2.4 Étoile de Kleene

Proposition 2.2.20

Soit L un langage local, le langage $L^\star$ est encore un langage local.

Démonstration : Si on pose encore $P = P(L), S = S(L)$ et $F = F(L)$.

On voit que $P(L^\star) = P$, $S(L^\star) = S$ et $F(L^\star) = F \cup S \times P$.

Comme précédemment, on peut montrer par récurrence sur la longueur du mot que si $w \in (P(L^\star)\mathscr{A}^\star \cap \mathscr{A}^\star S(L^\star)) \setminus \mathscr{A}^\star N(L^\star)\mathscr{A}^\star$ alors $w \in L^\star$. L'idée est de « couper » ăle mot $w = w_1 \cdots w_n$ dès que l'on voit un facteur $w_p w_{p+1} \in S \times P$. Dans ce cas, $w_1 \cdots w_p \in L$ et on recommence avec le suffixe $w_{p+1} \cdots w_n$. □

2.3 Expressions rationnelles linéaires

Définition 2.3.21 (Expressions rationnelles linéaires)

Une expression rationnelle est dite linéaire si toute lettre de l'alphabet apparait au plus une fois

Exemple : Les expressions $(a|b)\star$, $a\star|b$ sont linéaires. L'expression $a\star|ba$ ne l'est pas.

Proposition 2.3.22

Le langage associé a une expression régulière linéaire est local.

Démonstration : Cela se démontre par récurrence sur la longueur de l'expression e [1]

- si $e = \emptyset$, $L(e) = \emptyset$ qui est local. On prend $N = \mathscr{A}^2$, $P = S = \emptyset$.
- si $e = \varepsilon$, $L(e) = \{\varepsilon\}$ qui est local. On prend $N = \mathscr{A}^2$, $P = S = \emptyset$.
- si $e = a$ avec $a \in \mathscr{A}$, $L(e) = \{a\}$ qui est local défini par $P = \{a\} S = \{a\}$ et $N = \mathscr{A}^2$
- si $e = e_1|e_2$ alors $L(e) = L(e_1) \cup L(e_2)$. Par hypothèse de récurrence, $L(e_1)$ et $L(e_2)$ sont des langages locaux construits sur des alphabets disjoints (car e est linéaire). Alors $L(e_1) \cup L(e_2)$ est local d'après ce qui précède.
- si $e = e_1 e_2$ alors $L(e) = L(e_1)L(e_2)$. Par hypothèse de récurrence, $L(e_1)$ et $L(e_2)$ sont des langages locaux construits sur des alphabets disjoints (car e est linéaire). Là encore, $L(e_1)L(e_2)$ est local.
- Si $e = e_1\star$. Alors $L(e_1)$ est local et donc $L(e)$ aussi.

□

Remarque : La réciproque n'est pas vraie. Par exemple le langage $aa\star$ est un langage local ($P = \{a\}, S = \{\varepsilon\}, F = \{aa\}$) mais il n'est pas associé à une expression linéaire.

★ **Méthode :**

Il faut savoir calculer explicitement les paramètres P, S, F associés à un langage défini par une expression linéaire. Cela se fait par induction. On garde un marqueur λ defini par $\lambda(e) = V$ si $\varepsilon \in L(e)$ et $\lambda(e) = F$ sinon. On a alors les formules suivantes

e	$\lambda(e)$	$P(e)$	$S(e)$	$F(e)$
$\emptyset$				
$\{\varepsilon\}$				
a				
$f\star$				
$e_1\|e_2$				

Et pour $e_1 e_2$

1. ou par induction structurelle.

— $\lambda(e_1e_2) =$

— $P(e_1e_2) = \begin{cases} \\ \\ \end{cases}$

— $S(e_1e_2) = \begin{cases} \\ \\ \end{cases}$

— $F(e_1e_2) =$

Exemple :

Prenons l'expression $e = x_1x_2\star$. On a

e	$\lambda(e)$	$P(e)$	$S(e)$	$F(e)$
x_2				
$x_2\star$				
x_1				
$x_1x_2\star$				

Bien évidemment dans le cas présent on aurait pu donner directement la réponse.

ATTENTION

On peut maintenant expliquer le principe de la démonstration du théorème de Kleene. Si on se donne un langage régulier L on peut lui associer une expression régulière e. Ensuite on va linéariser cette expression pour obtenir une expression linéaire f dont le langage associé sera local. On saura alors définir un automate reconnaissant ce langage. Il ne restera plus qu'à faire marche arrière.

Définition 2.3.23 (Linéarisation)

Soit e une expression rationnelle sur $\mathscr{A}$. On dit que l'on linéarise l'expression en remplaçant chaque lettre par la lettre x_k où k est la position de la lettre dans l'expression. L'expression obtenue est linéaire par construction

Exemple : Soit $e = (ab|b) \star ba$. La linéarisation est $(x_1x_2|x_3) \star x_4x_5$.

ATTENTION

Afin de pouvoir faire marche arrière, il faut garder en memoire les lettres initiales. Cela peut se faire à l'aide d'une tableau en indiquant en position i la « valeur » de la lettre x_i.

Exemple : Dans notre exemple $e = (ab|b) \star ba$ on récupère le tableau

k	1	2	3	4	5
lettre(k)	a	b	b	b	a

2.4 Algorithme de Berry-Sethi

Le but de l'algorithme de Berry-Sethi est d'associer à une expression régulière e un automate fini déterministe A tel que $L(A) = L(e)$.

Définition 2.4.24

Un automate fini déterministe $A = (Q, q_0, F, \delta)$ est dit local si pour toute lettre x de $\mathscr{A}$, il existe un état q' tel que pour tout état q, (q, x) est un blocage ou $\delta(q, x) = q'$.

Remarque : Cela signifie que toutes les transitions indicées par une lettre x fixée arrivent au même état. C'est-à-dire que lors de la lecture d'un mot, en regardant la dernière lettre lue, on sait dans quel état on est arrivé.

Cela peut s'écrire

$$\forall x \in \mathscr{A}, \#\{q' \in Q \mid \exists q \in Q, \delta(q,x) = q'\} \leqslant 1$$

Proposition 2.4.25

Soit L un langage local. Il existe un automate fini déterministe local A tel que $L(A) = L$.

Démonstration : On se donne un langage local L sur $\mathscr{A}$ défini par (P, S, F) où F est l'ensemble des facteurs de L. On définit alors l'automate local suivant $A = (Q, q_0, S, \delta)$ où

— L'ensemble des états $Q = \mathscr{A} \cup \{q_0\}$ est l'alphabet $\mathscr{A}$ sur lequel est défini le langage auquel on ajoute un élément q_0 (on suppose que q_0 n'est pas un élément de $\mathscr{A}$).

— L'état initial est q_0.

— Les états finaux sont les suffixes S si ε n'appartient pas à L et $S \cup \{q_0\}$ si $\varepsilon \in L$.

— On définit δ par $\delta(q_0, x) = x$ si $x \in P$ et $\delta(y, x) = x$ si $yx \in F$. Toutes les autres couples sont des blocages de l'automate.

On voit que cet automate est un automate fini, déterministe, standard (par contre il n'est pas complet à priori). De plus il est local car toutes les transitions définies par une lettre x arrivent sur l'état x.

Il est clair que tous les mots du langage L sont reconnus par l'automate par construction.

Réciproquement soit $w_1 w_2 \cdots w_n$ un mot (non vide) reconnu par A. On peut le schématiser par :

$$q_0 \xrightarrow{w_1} w_1 \xrightarrow{w_2} \cdots w_{n-1} \xrightarrow{w_n} w_n.$$

Alors

— $w_1 \in P$ (car sinon $\delta(q_0, w_1)$ est un blocage)

— $w_n \in S$ car c'est un état final et qu'il ne peut valoir q_0

— pour tout $k \in [\![1\,;\, n-1]\!]$ $w_k w_{k+1} \in F$ car $\delta(w_k, w_{k+1}) = w_{k+1}$.

Donc $w \in (P\mathscr{A}^\star \cap \mathscr{A}^\star S) \setminus (\mathscr{A}^\star N \mathscr{A}^\star) = L \setminus \{\varepsilon\}$.

□

Remarque : L'automate défini ainsi a $\#\mathscr{A} + 1$ états.

Exemple : Si on reprend l'expression linéaire $(x_1x_2|x_3) \star x_4x_5$. On peut trouver les paramètres P, S et F qui définissent son langage local.

e	$\lambda(e)$	$P(e)$	$S(e)$	$F(e)$
x_1x_2				
x_3				
$(x_1x_2\|x_3)$				
$(x_1x_2\|x_3)\star$				
x_4x_5				
e				

On peut alors construire un automate local reconnaissant ce langage.

Il suffit alors de remplacer de les étiquettes des transitions par les lettres initiales de l'alphabet (que l'on avait pris garde de conserver).

Dans notre cas on obtient

ATTENTION

L'automate obtenu n'est à priori pas déterministe

Définition 2.4.26

On a associé ainsi à tout expression rationnelle un automate (à priori non déterministe). Il s'appelle l'automate de ***Glushkov*** *de l'expression.*

Proposition 2.4.27

L'automate de Glushkov associé à une expression rationnelle e reconnait le langage $L(e)$.

Démonstration : Soit $\mathscr{A}$ l'alphabet sur lequel est défini e. On note :

— e' l'expression rationnelle linéaire associée à e obtenue précédemment

— $L(e')$ le langage reconnu par cette expression régulière

— $\mathscr{B}$ l'alphabet sur lequel est défini e'.

On note f l'application de $\mathscr{B}$ dans $\mathscr{A}$ qui associe à une lettre la lettre dont elle est issue (c'est une application de « démarquage »).

On a construit un automate A' reconnaissant $L(e')$.

On note A l'automate de Glushkov obtenu en replaçant les x_i par $f(x_i)$.

Soit $w \in \mathscr{A}^\star$,

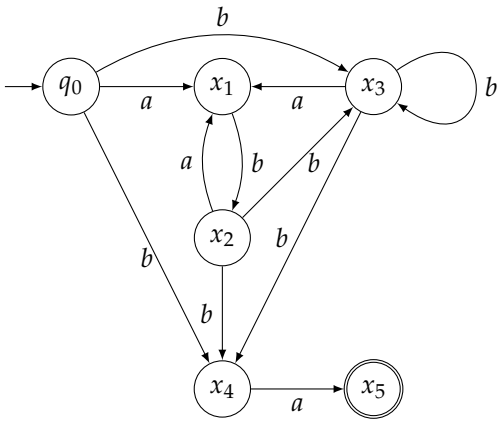
$$A \text{ reconnait } w \iff \text{il existe } w' \in \mathscr{B}^\star \text{ tel que } f(w') = w \text{ et } A' \text{ reconnait } w'.$$

On en déduit que le langage reconnu par A est $f(L(e')) = L(e)$. □

ATTENTION

Si on veut obtenir un automate déterministe il suffit d'appliquer la détermination à l'automate de Glushkov.

Exemple : On reprend notre automate



On fait notre tableau

état	a	b
$\{q_0\}$		

Ce qui nous donne l'automate déterministe

On a bien démontré le sens direct du théorème de Kleene

Théorème 2.4.28 (Théorème de Kleene)

Tout langage rationnel est reconnaissable (par une automate fini déterministe).

3 Propriétés de cloture

3.1 Introduction

On a vu que tout langage rationnel était reconnu par un automate fini (déterministe). La réciproque est aussi vraie (résultat hors programme - voir plus loin). De ce fait, les langages reconnaissables par un automate sont exactement les langages rationnels.

Par définition on sait que $R(\mathscr{A})$ l'ensemble des langages rationnels est stable par union, concaténation et par étoile. On veut essayer de retrouver ces propriétés via les automates, c'est à dire que si on se donne deux automates A et A' reconnaissant les langages $L(A)$ et $L(A')$, on veut construire des automates reconnaissants les langages $L(A) \cup L(A')$, $L(A)L(A')$ et $L(A)\star$.

On en profitera aussi pour voir les propriétés de clôture pour d'autres opérations ensemblistes : intersection, complémentaire, différence et différence symétrique.

3.2 Automate produit et intersection

Définition 3.2.29

On se donne deux automates déterministes $A = (Q, q_0, F, \delta)$ et $A' = (Q', q'_0, F', \delta')$.
On appelle automate produit l'automate $A \times A' = (Q \times Q', (q_0, q'_0), F \times F', \Delta)$ où

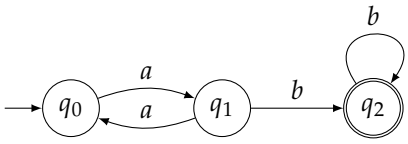
$$\forall (q, q') \in Q \times Q', \forall x \in \mathscr{A}, \Delta((q, q'), x) = (\delta(q, x), \delta'(q', x)) \text{ si les deux existent.}$$

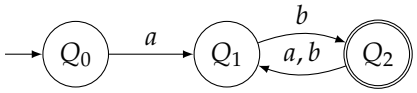
Remarques :

1. La notation pour l'automate produit n'est pas standardisée. On trouve $A \times A'$, $A \oplus A'$, $A \otimes A'$,...
2. Avec les notations de la définition, $((q, q'), x)$ est un blocage dès que (q, x) ou (q', x) en est un

Exemple :

On considère les deux automates suivants





Ils reconnaissent les langages :

On note $q_{i,j}$ l'état (q_i, Q_j). On a alors l'automate

Automate que l'on peut émonder pour obtenir

Le langage reconnu est

Proposition 3.2.30

Avec les notations précédentes, le langage reconnu par l'automate produit est $L(A) \cap L(A')$

Démonstration :

— Soit $w = w_0 w_1 \dots w_n \in L(A) \cap L(A')$. Il existe un calcul dans A

$$q_0 \xrightarrow{w_0} \cdots \xrightarrow{w_n} q$$

où $q \in F$ ainsi qu'un calcul dans A'

$$q'_0 \xrightarrow{w_0} \cdots \xrightarrow{w_n} q'$$

où $q' \in F'$.
Cela nous donne donc un calcul dans l'automate produit :

$$(q_0, q'_0) \xrightarrow{w_0} \cdots \xrightarrow{w_n} (q, q')$$

et $(q, q') \in F \times F'$ est un état terminal de $A \times A'$ donc $w \in L(A \times A')$.

— Réciproquement si $w \in L(A \times A')$ il existe un calcul

$$(q_0, q'_0) \xrightarrow{w_0} \cdots \xrightarrow{w_n} (q, q')$$

dans $A \times A'$ où (q, q') est terminal, c'est-à-dire $q \in F$ et $q' \in F'$. On peut alors « projeter » ce calcul dans A et dans A' pour obtenir que $w \in L(A) \cap L(A')$.

□

Corollaire 3.2.31

L'ensemble des langages reconnaissables est stable par intersection.

3.3 Réunion (HP)

On se donne encore deux automates déterministes $A = (Q, q_0, F, \delta)$ et $A' = (Q', q'_0, F', \delta')$. On veut cette fois construire un automate $A \oplus A'$ tel que $L(A \oplus A') = L(A) \cup L(A')$. On peut penser faire la même chose que ci-dessus mais en prenant pour états finals les états

Proposition 3.3.32

Soit L et L' deux langages et deux automates déterministes **complets** $A = (Q, q_0, F, \delta)$ et $A' = (Q', q'_0, F', \delta')$ qui reconnaissent respectivement L et L'. Le langage $L \cup L'$ est le langage reconnu par l'automate $A \otimes A' = (Q \times Q', (q_0, q'_0), F \times Q' \cup Q \times F', \Delta)$ où

$$\forall (q, q') \in Q \times Q', \forall x \in \mathscr{A}, \Delta((q, q'), x) = (\delta(q, x), \delta'(q', x))$$

Démonstration : Comme ci-dessus. □

Corollaire 3.3.33

L'ensemble des langages reconnaissables est stable par union.

Démonstration : On sait que si L et L' sont des langages reconnus par des automates finis (déterministes). Ils sont reconnus par des automates complets (quitte à ajouter un « puit »). On peut alors considérer l'automate $A \oplus A'$ qui reconnait $L \cup L'$.

□

3.4 Complémentaire

Proposition 3.4.34

Soit L un langage rationnel et $A = (Q, q_0, F, \delta)$ un automate **complet** qui reconnait L. Le langage $\overline{L} = \mathscr{A}^\star \setminus L$ est reconnu par l'automate $A' = (Q, q_0, Q \setminus F, \delta)$.

Démonstration : Il suffit de voir que pour tout $w \in \mathscr{A}^\star$,

$$w \in L \iff \delta(q_0, w) \in F$$

puisqu'il n'y a pas de blocages. □

Corollaire 3.4.35

L'ensemble des langages reconnaissables est stable par complémentaire.

3.5 Différence et différence symétrique (HP)

Proposition 3.5.36

Soit L et L' deux langages et deux automates déterministes **complets** $A = (Q, q_0, F, \delta)$ et $A' = (Q', q_0', F', \delta')$ qui reconnaissent respectivement L et L'.

— L'automate

reconnait $L \setminus L'$

— L'automate

reconnait $L \Delta L' = (L \cup L') \setminus (L \cap L')$

Corollaire 3.5.37

On peut décider si deux automates reconnaissent le même langages car $L = L' \iff L \Delta L' = \emptyset$

3.6 Concaténation (HP)

Soit L et L' deux langages reconnus par les automates $A = (Q, q_0, F, \delta)$ et $A' = (Q', q_0', F', \delta')$. On veut construire un automate reconnaissant LL'.

3.7 Étoile (HP)

Soit L un langage reconnu par l'automate $A = (Q, q_0, F, \delta)$. On peut construire un automate non déterministe reconnaissant le langage $L\star$. Il suffit pour cela :

— De considérer un automate standard reconnaissant le langage L.

— Ajouter l'état initial aux états finaux (pour accepter le mot vide s'il ne l'était pas).

— Pour tout $(p, x) \in Q \times \mathscr{A}$ tel que $p = \delta(q_0, x)$ et tout état final q on ajoute une flèche $q \xrightarrow{x} p$.

4 Compléments

4.1 Lemme de l'étoile et applications

Nous avons déjà vu comment montrer qu'un langage n'était pas rationnel en se basant que s'il l'était alors il serait reconnu par un automate ayant un nombre fini d'états. Il existe une version un peu plus sophistiquée de cet argument.

L'idée est que si un mot de longueur n est reconnu par un automate ayant N états et que $n \geqslant N$ alors, tout calcul réussi dans l'automate sur ce mot passe par $n+1$ états et donc (au moins) deux fois par le même état.

Lemme 4.1.38 (Lemme de l'étoile)

Soit L un langage rationnel, il existe un entier N tel que tout mot w de L tel que $|w| \geqslant N$ peut s'écrire $w = xyz$ tel que

— $0 < |y| \leqslant N$ - en particulier $y \neq \varepsilon$.

— Pour tout $k \in \mathbf{N}$, $xy^kz \in L$

Remarques :

1. Cela s'appelle aussi le lemme de pompage.
2. Cela implique que $x(y\star)z \subset L$.

Démonstration : On sait qu'il existe un automate fini déterministe A reconnaissant L. Si on note N le nombre d'état de A. Tout mot w de L est reconnu par A. De ce fait, il existe un calcul réussi dans A sur w :

$$q_0 \xrightarrow{w_1} q_1 \xrightarrow{w_2} \cdots \xrightarrow{w_p} q_p$$

où $p = |w|$.

Maintenant, si on suppose $p \geqslant N$, on voit que le calcul ci-dessus passe par $p+1 > N$ états. Par le principe des tiroirs, il existe $(k, k') \in [\![0\,;\, p]\!]^2$ tels que $q_k = q_{k'}$.

Cela signifie que le calcul fait une « boucle » dans l'automate.

On pose alors

$$x = w_1 \cdots w_k ; y = w_{k+1} \cdots w_{k'} \text{ et } z = w_{k'+1} \cdots w_p.$$

On a donc

$$q_k \xrightarrow{y} q_{k'} = q_k.$$

De ce fait, pour tout $n \in \mathbf{N}$, xy^nz est reconnu par l'automate et donc est dans L. □

ATTENTION

Le lemme de l'étoile n'est pas explicitement au programme. Si vous voulez l'utiliser vous **devez** le redémontrer.

Exemple : On note L le langage

$$L = \{a, ab, abba, abbabaab, abbabaabbaababba, \cdots\}$$

défini comme étant le plus petit langage contenant a et tel que si $w \in L$ alors $w\overline{w} \in L$ où $\overline{w}$ est obtenu à partir de w en remplaçant les a par des b et réciproquement.

Montrons que L n'est pas rationnel.

4.2 Réciproque du théorème de Kleene

On veut justifier la réciproque du théorème de Kleene, à savoir que tout langage reconnu par un automate est rationnel.

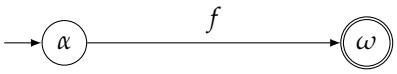
Là encore, la preuve est algorithmique. Nous allons, à partir d'un automate fini déterministe A, construire une expression rationnelle e telle que $L(e) = L(A)$.

Présentons, sur un exemple l'algorithme BMC (Brzozowski-McCluskey). Il existe aussi un autre algorithme (McNaughton Yamada).

On se donne un automate déterministe et on veut construire une expression rationnelle. Pour travailler on étiquette les flèches de l'automate par des expressions rationnelles et plus nécessairement des lettres de $\mathscr{A}$. L'algorithme est le suivant :

1. On ajoute deux nouveaux états α et ω. On relie α à l'état initial q_0 par une transition étiquetée par ε. On relie les états terminaux à ω par une transition ε. *Le but est d'obtenir un automate standard qui n'a qu'un état final. Si l'automate est standard et qu'il existe un unique état final on peut sauter cette étape*
2. On simplifie l'automate en appliquant les deux méthodes ci-dessus autant que possible.
   - S'il existe deux états p, q et deux transition $p \xrightarrow{e_1} q$ et $p \xrightarrow{e_2} q$ on les remplace par $p \xrightarrow{e_1|e_2} q$. Ici p peut-être égal à q.
   - On supprime un état q (différent de α et ω), et pour tous les états (p, r) distincts de q (mais éventuellement égaux) tels qu'il existe des transitions $p \xrightarrow{e_1} q$, $q \xrightarrow{f} q$ et $q \xrightarrow{e_2} r$, on ajoute une transition $p \xrightarrow{e_1 f^\star e_2} r$

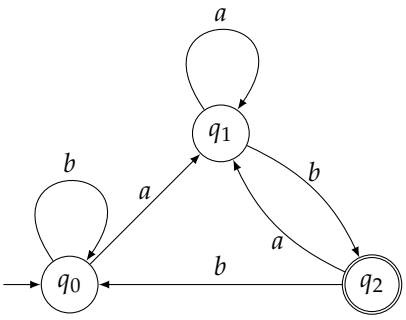
On aboutit à un automate



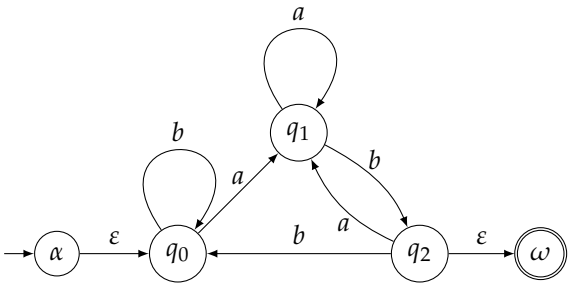
Remarques :

1. Cet algorithme termine car il diminue le nombres d'états et le nombres de transitions (penser à un ordre lexicographique)
2. Par construction, le language reconnu par l'automate est celui qui est définit par l'expression rationnelle trouvée.

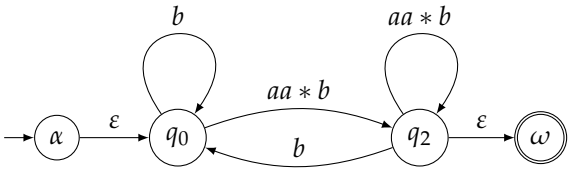
Exemple : Considérons l'automate suivant



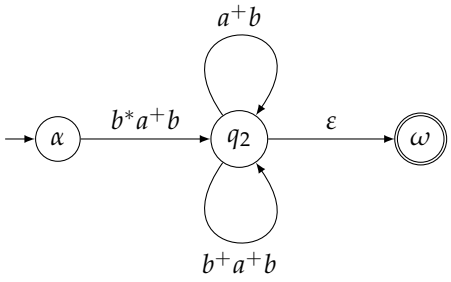
1. On ajoute les états α et ω.



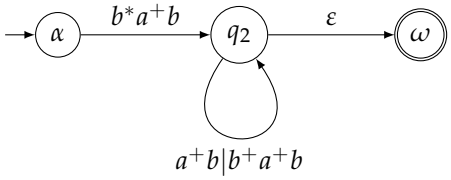
2. On supprime alors l'état q_1



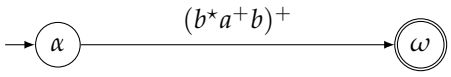
Puis l'état q_0 [2] :



On supprime la double flèche :



On remarque que $a^+b|b^+a^+b = b^*a^+b$, puis on supprime q_2.



L'expression rationnelle est donc $(b^\star a^+ b)^+$.

2. On rappelle que l'expression rationnelle a^+ désigne $aa\star$.