

Langages et automates

Alphabet : Un alphabet est un ensemble fini et non vide de symboles, appelés aussi lettres ou caractères. Il est généralement noté Σ . Le nombre de symboles qu'il contient et appelé sa cardinalité et est noté $|\Sigma|$.

Exemple : Le langage machine est basé sur l'alphabet $\Sigma = \{0,1\}$, $|\Sigma| = 2$.

Mot (ou chaîne) : Un mot sur un alphabet Σ est une suite finie de symboles appartenant à Σ .

Exemple :

- Le mot 1011 est défini sur l'alphabet $\Sigma = \{0,1\}$
- Le mot 6.1239 est défini sur l'alphabet $\Sigma = \{0,1,2,3,4,5,6,7,8,9,.\}$

Longueur : La longueur d'un mot ω , notée $|\omega|$, est le nombre de symboles qui composent ce mot.

Exemple :

- Le mot $\omega = 010$ est de longueur $|\omega| = 3$.
- Le mot $u = 01110$ est de longueur $|u| = 5$.
- Le **mot vide**, noté ε , est le seul mot de longueur nulle, $|\varepsilon| = 0$.

On note :

- Σ^* désigne l'ensemble de tous les mots définis sur Σ , y compris le mot vide ε . Par exemple, si $\Sigma = \{a, b\}$, alors $\Sigma^* = \{\varepsilon, a, b, aa, bb, ab, ba, aaa, \dots\}$
- Σ^+ désigne l'ensemble des mots sur l'alphabet Σ de longueur supérieure ou égale à 1.
- Σ^n représente l'ensemble des mots sur l'alphabet Σ de longueur n .
- Σ^0 représente l'ensemble des mots sur

Opérations sur les mots

Concaténation : La concaténation de deux mots u et v de Σ^* est un mot noté $u.v$ ou uv défini comme suit : si $u = u_1u_2 \dots u_n$, $v = v_1v_2 \dots v_m$, alors $\omega = uv = u_1u_2 \dots u_nv_1v_2 \dots v_m$, et $|\omega| = |uv| = |u| + |v|$.

Cette opération est associative, non commutative, et le mot vide ε en est l'élément neutre.

Propriété de la concaténation

Soient ω_1, ω_2 et ω_3 trois mots définis sur l'alphabet Σ :

- $|\omega_1\omega_2| = |\omega_1| + |\omega_2|$
- $(\omega_1\omega_2)\omega_3 = \omega_1(\omega_2\omega_3)$ (associativité)
- $\omega_1\varepsilon = \varepsilon\omega_1 = \omega_1$ (ε est l'élément neutre)

Exposant (puissance)

La puissance n -ième d'un mot ω est définie par : $\omega^n = \underbrace{\omega\omega \dots \omega}_{n \text{ fois}}$, $\omega^0 = \varepsilon$, $\omega^{n+1} = \omega^n\omega$.

Mot miroir

Soit $\omega = a_1a_2 \dots a_n$ un mot défini sur l'alphabet Σ . On appelle mot miroir (ou renversé) de ω , noté ω^R , le mot obtenu en écrivant les symboles de ω dans l'ordre inverse, c'est-à-dire : $\omega^R = a_na_{n-1} \dots a_1$. On a alors : $(\omega^R)^R = \omega$.

Mots conjugués

Deux mots ω_1 et ω_2 sont dits conjugués s'il existe deux mots u et v tels que : $\omega_1 = uv$ et $\omega_2 = vu$.

Préfixe et suffixe

Un mot $u \in \Sigma^*$ est préfixe du mot $\omega \in \Sigma^*$ s'il existe $v \in \Sigma^*$ tel que $\omega = uv$. Le mot v est dit suffixe du mot ω .

Projection

Soit $w \in \Sigma^*$ un mot et $\bar{\Sigma} \subseteq \Sigma$ un sous-alphabet. La projection de w sur $\bar{\Sigma}$, notée $w \uparrow \bar{\Sigma}$, est le mot obtenu en supprimant de w tous les symboles n'appartenant pas à $\bar{\Sigma}$.

Exemple : Soit $\Sigma = \{a, b, c\}$ et $\bar{\Sigma} = \{a, b\}$. Pour le mot $w = cacbccca$, la projection sur $\bar{\Sigma}$ consiste à conserver uniquement les symboles a et b . On obtient alors : $w \uparrow \bar{\Sigma} = aba$. Pour le mot $w = c$, aucun symbole n'appartient à $\bar{\Sigma}$, alors : $w \uparrow \bar{\Sigma} = \varepsilon$.

Langage

Un langage L sur un alphabet Σ est un sous-ensemble de Σ^* , c'est-à-dire, $L \subseteq \Sigma^*$ ou $L \in P(\Sigma^*)$. La cardinalité de L , c'est-à-dire le nombre de mots qu'il contient, est notée $|L|$.

Exemple :

- Le langage des nombres binaires définis sur l'alphabet $\Sigma = \{0, 1\}$ est un langage infini.
- Le langage des mots de longueur 2 sur l'alphabet $\Sigma = \{a, b\}$ est $L = \{aa, ab, ba, bb\}$.
- Le langage vide, noté $\emptyset$, ne contient aucun mot ($\varepsilon \notin \emptyset$).

Opérations sur les langages

Soit L, L_1 et L_2 trois langages définis sur un alphabet Σ .

- **Union (ou somme) :** $L_1 \cup L_2 = L_1 + L_2 = \{w \in \Sigma^* \mid w \in L_1 \text{ ou } w \in L_2\}$
- **Intersection :** $L_1 \cap L_2 = \{w \in \Sigma^* \mid w \in L_1 \text{ et } w \in L_2\}$
- **Concaténation (produit) :** $L_1 L_2 = \{w \in \Sigma^* \mid \exists w_1 \in L_1, \exists w_2 \in L_2 : w = w_1 w_2\}$
- **Différence :** $L_1 \setminus L_2 = L_1 - L_2 = \{w \in \Sigma^* \mid w \in L_1 \text{ et } w \notin L_2\}$
- **Complément :** Le complémentaire d'un langage L est défini par $L^c = \Sigma^* \setminus L = \{w \in \Sigma^* \mid w \notin L\}$.
- **Puissance :** La concaténation répétée d'un langage avec lui-même, $L^{n+1} = L^n L$ et $L^0 = \{\varepsilon\}$.
- **Clôture de Kleene (Kleene star) ou étoile de L :** La clôture de Keene d'un langage L est l'ensemble des mots obtenus par concaténation finie de mots de L :

$$L^* = \{w \in \Sigma^* \mid \exists k \in \mathbb{N}, \exists w_1, \dots, w_k \in L : w = w_1 \dots w_k\}$$
$$L^* = \bigcup_{n \geq 0} L^n = L^0 \cup L \cup L^2 \cup L^3 \cup L^4 \cup \dots$$

Par convention, le mot vide ε appartient à L^* . En général, L^* est un langage infini. On a :

$$(L^*)^* = L^*, \emptyset^* = \{\varepsilon\}, \{\varepsilon\}^* = \{\varepsilon\}$$

- **Fermeture plus :** Elle contient toutes les concaténations non vides de mots de L :

$$L^+ = LL^* = \bigcup_{n \geq 1} L^n = L \cup L^2 \cup L^3 \cup L^4 \cup \dots$$

- **Préfixe-clôture :** La préfixe-clôture d'un langage L , notée $\bar{L}$ est le langage contenant tous les préfixes des mots de L : $\bar{L} = \{w_1 \in \Sigma^* \mid \exists w_2 \in \Sigma^* : w_1 w_2 \in L\}$. On a par définition : $L \subseteq \bar{L}$. Un langage L est dit préfixe-clos si $L = \bar{L}$.
- **Projection :** La projection d'un langage L sur $\bar{\Sigma}$, notée $L \uparrow \bar{\Sigma} \subseteq \bar{\Sigma}^*$, est définie comme l'ensemble des projections de tous les mots de L sur $\bar{\Sigma}$: $L \uparrow \bar{\Sigma} = \{w \uparrow \bar{\Sigma} \mid w \in L\}$.

Exemple : Soit $\Sigma = \{a, b, c\}$ et soient les langages $L_1 = \{ab, ba\}$, $L_2 = \{c, cc\}$ et $L_3 = \{ac\}$. Alors :

- $L_1 L_2 = \{abc, abcc, bac, bacc\}$

- $L_1 L_2 L_3 = \{abcac, abccac, bacac, baccac\}$
- $L_1^2 = \{abab, abba, baab, baba\}$
- $\overline{L_1} = \{\varepsilon, a, ab, b, ba\}$

Langages réguliers (rationnels)

En théorie des langages formels, les langages rationnels, également appelés langages réguliers ou langages reconnaissables, peuvent être caractérisés de plusieurs manières équivalentes :

- Ce sont les langages décrits par les expressions régulières ou rationnelles, d'où le nom de langages réguliers ;
- Ce sont les langages obtenus à partir des lettres de l'alphabet et du langage vide, en appliquant un nombre fini d'opérations rationnelles (union, concaténation et étoile de Kleene), d'où le nom de langages rationnels ;
- Ce sont les langages reconnus par des automates finis, ce qui justifie l'appellation de langages reconnaissables.

Les expressions régulières (ou rationnelles) constituent une notation formelle permettant de décrire la manière dont un langage régulier est construit à partir de langages élémentaires à l'aide des opérations rationnelles.

Propriétés :

- Tout langage fini est rationnel.
- Si L est un langage rationnel, alors L^* est également rationnel.
- Si L_1 et L_2 sont des langages rationnels, alors $L_1 + L_2$ et $L_1 L_2$ sont rationnels

Expressions régulières

Les expressions régulières (ER) sur un alphabet Σ sont définies récursivement par les règles suivantes :

- Règle 1 : Le langage vide $\emptyset$, le mot vide ε et chaque lettre de $a \in \Sigma$ sont des expressions régulières.
- Règle 2 : si α et β sont des ER , alors la concaténation $(\alpha\beta)$, l'union $(\alpha + \beta)$ et l'étoile de Kleene α^* sont également des expressions régulières.
- Règle 3 : Toute expression régulière s'obtient par l'application d'un nombre fini des règles 1 et 2.

Langage associé à une expression régulière

À toute expression régulière ER est associé un langage $L(ER)$, défini comme suit :

- $L(\emptyset) = \emptyset, L(\varepsilon) = \{\varepsilon\}$
- $L(a) = \{a\}, \forall a \in \Sigma$
- $L(\alpha\beta) = L(\alpha)L(\beta)$ (concaténation)
- $L(\alpha + \beta) = L(\alpha) + L(\beta)$ (union)
- $L(\alpha^*) = L(\alpha)^*$

Exemple :

Soit l'alphabet $\Sigma = \{a, b, c\}$

- $(a + b + c)^* = \Sigma^*$: L'ensemble de tous les mots définis à partir de l'alphabet Σ
- $(a + b + c)^* a$: L'ensemble de tous les mots finissant par le symbole a .
- $(a + b + c)(a + b + c)^*$: L'ensemble de tous les mots non vides définis sur Σ .
- $(a + b)^* b (a + b)^*$: L'ensemble des mots composés par a et b contenant au moins un b .
- $0 + 10^*$: C'est le langage formé du mot 0 et des mots composés d'un 1 suivi d'un nombre quelconque de 0 : $\{0, 1, 10, 100, \dots\}$.

Remarque — Ordre de priorité des opérations

Dans les expressions régulières, l'ordre de priorité des opérations est le suivant : Etoile de Kleene (*) > Concaténation > Union). Ainsi, l'expression, $0 + 10^*$ est interprétée comme $(0 + (1(0)^*))$.

Automates Finis

Un automate fini (ou automate à états finis) est un modèle mathématique utilisé pour décrire un système à événements discrets dont l'évolution dépend d'une séquence d'entrées symboliques. Il se caractérise par un ensemble fini d'états ainsi que par des transitions reliant ces états.

Les automates finis sont largement employés pour modéliser, analyser et implémenter des comportements réactifs dans de nombreux systèmes matériels et logiciels.

Exemples :

- Un distributeur automatique de boissons qui délivre le produit demandé lorsque la somme introduite est suffisante,
- Les ascenseurs, capables de gérer et d'ordonner les appels successifs afin de s'arrêter aux étages intermédiaires,
- Les machines à laver, dont les cycles suivent une succession d'états (lavage, rinçage, essorage),
- Les feux de circulation, qui s'adaptent à la présence de véhicules en attente,
- Les digicodes, qui analysent la validité d'une suite de chiffres,
- Les protocoles de communication (gestion des états : connexion, attente, transmission, fin),

On distingue principalement deux types d'automates finis :

- AFD : automate fini déterministe,
- AFN : automate fini non déterministe.

Automate fini déterministe

Un automate fini déterministe (AFD) est défini par un quintuplé $A = (\Sigma, Q, \delta, q_0, F)$ où :

- Σ est un ensemble fini de symboles, appelé alphabet.
- Q est un ensemble fini dont les éléments sont appelés états.
- $\delta : Q \times \Sigma \rightarrow Q$ est une fonction de transition. Pour chaque état $q \in Q$ et chaque symbole $a \in \Sigma$, $\delta(q, a)$ donne l'état unique dans lequel l'automate passe.
- $q_0 \in Q$ est l'état initial.
- $F \subseteq Q$ est l'ensemble des états finaux (ou états acceptants, parfois appelés états marqués).

Un automate fini peut être représenté de deux manières équivalentes :

- Par une table de transition décrivant la fonction δ
- Par un graphe orienté, dont les sommets représentent les états et les arcs étiquetés par les symboles de l'alphabet représentent les transitions.

Représentation par table de transition

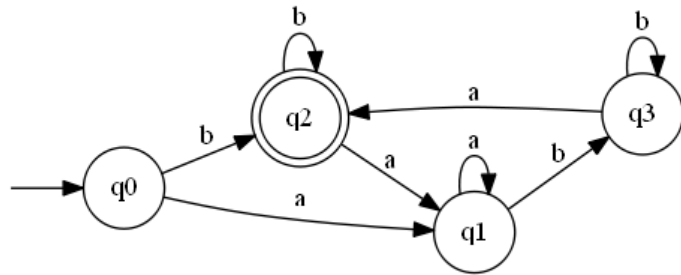
La table de transition comporte une ligne par état et une colonne par symbole de l'alphabet. L'entrée (q_i, a) indique l'état atteint lorsque l'automate, initialement dans l'état q_i , lit le symbole a . L'état initial est signalé par une flèche « $\rightarrow$ » et les états finaux par une étoile « $*$ »

Représentation graphique

Un automate fini se représente par un graphe orienté dont les sommets (cercles) sont les états et les arcs les transitions. Une transition $\delta(q_i, a) = q_j$ est un arc orienté de q_i vers q_j étiqueté par a . L'état initial est indiqué par une flèche entrante et les états finaux par un double cercle.

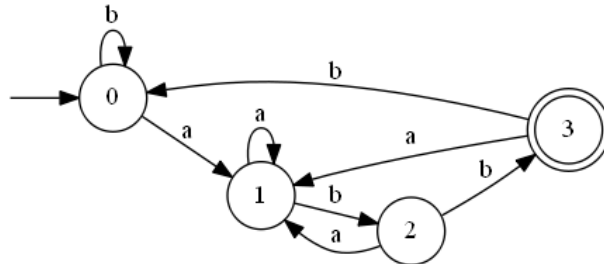
Exemple :

	a	b
$\rightarrow q_0$	q_1	q_2
q_1	q_1	q_3
$*q_2$	q_1	q_2
q_3	q_2	q_3



Exemple :

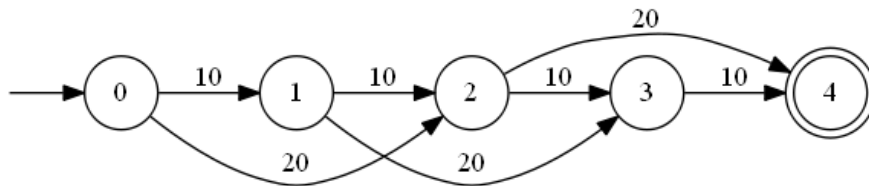
	a	b
$\rightarrow 0$	1	0
1	1	2
2	1	3
$*3$	1	0



Exemple : Distributeur de café

Le café coûte 40 DA et les pièces acceptées sont celles de 10 et 20 DA. Il n'y a pas de retour de monnaie.

L'alphabet $\Sigma = \{10, 20\}$; Ensemble des états : $Q = \{0, 1, 2, 3, 4\}$; Ensemble des états finaux : $F = \{4\}$; Etat initial : 0.



Remarque :

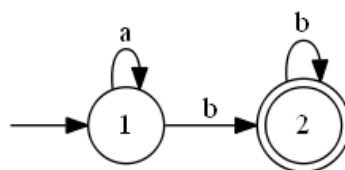
- Dans un AFD, $\forall q_i \in Q, \forall a \in \Sigma$, il existe au plus un état q_j tel que $\delta(q_i, a) = q_j$. Autrement dit, depuis un état donné, une seule transition au maximum peut porter le même symbole.
- Un AFD ne comporte pas des ε -transitions (epsilon-transitions).
- Un AFD possède un unique état initial.
- Un AFD sans états finaux est appelé *accepteur* et se note : $A = (\Sigma, Q, \delta, q_0)$.

Automate fini complet – complétion

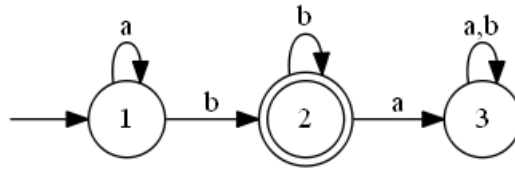
Un automate fini est complet si, pour tout état $q \in Q$ et tout symbole $a \in \Sigma$, la transition $\delta(q, a)$ est définie.

Tout automate incomplet peut être rendu complet en ajoutant un état puits non final, vers lequel sont dirigées toutes les transitions manquantes.

Exemple : Soit l'automate fini incomplet suivant :



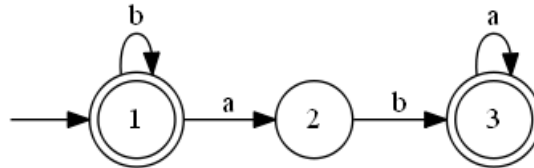
On peut construire l'automate fini complet suivant (l'état 3 est l'état puits) :



Langage reconnu (accepté ou marqué) par un AFD

Le langage $L_m(A)$ reconnu ou accepté par automate fini déterministe A est l'ensemble des mots qui conduisent l'automate, à partir de l'état initial, vers un état final. Formellement, le langage reconnu ou marqué est défini par : $L_m(A) = \{\omega \in \Sigma^* \mid \delta(q_0, \omega) \in F\}$.

Exemple : On considère l'automate fini déterministe suivant :



Calcul sur le mot $baba$: $1 \xrightarrow{b} 1 \xrightarrow{a} 2 \xrightarrow{b} 3 \xrightarrow{a} 3$ (accepté), ce mot est accepté car le calcul se termine sur l'état 3, qui est un état final.

Calcul sur le mot bba : $1 \xrightarrow{b} 1 \xrightarrow{b} 1 \xrightarrow{a} 2$ (rejeté), ce mot n'est pas accepté car le calcul se termine sur l'état 2, qui n'est pas un état final.

Calcul sur le mot $abab$: $1 \xrightarrow{a} 2 \xrightarrow{b} 3 \xrightarrow{a} 3 \xrightarrow{b} ?$ (impasse), ce mot n'est pas accepté car l'automate se bloque. Il n'existe pas de transition définie depuis l'état 3 sur le symbole b .

Remarque : Si l'automate se bloque pendant la lecture d'un mot ou n'atteint pas l'état final, ce mot n'est pas reconnu par l'automate.

Langage généré par un AFD

Le langage généré par l'automate A , noté $L(A)$, est l'ensemble de tous les mots que l'automate peut produire à partir de l'état initial, en suivant ses transitions, sans condition sur l'état d'arrivée : $L(A) = \{\omega \in \Sigma^* \mid \delta(q_0, \omega) \text{ est défini}\}$.

Le langage généré par un automate est toujours préfixe-clos.

Propriété d'un automate fini déterministe

Un automate fini déterministe est :

- Accessible (ou réalisable) si tous ses états sont accessibles. Un état q est accessible s'il existe un chemin partant de l'état initial et conduisant à q .
- Co-accessible (ou co-réalisable) si tous ses états sont co-accessibles. Un état q est co-accessible s'il existe un chemin partant de q et conduisant à un état final.
- Non-bloquant si tous ses états sont non bloquants. Un état q est dit bloquant s'il est accessible mais non co-accessible.
- Emondé (utile) s'il est à la fois accessible et co-accessible.

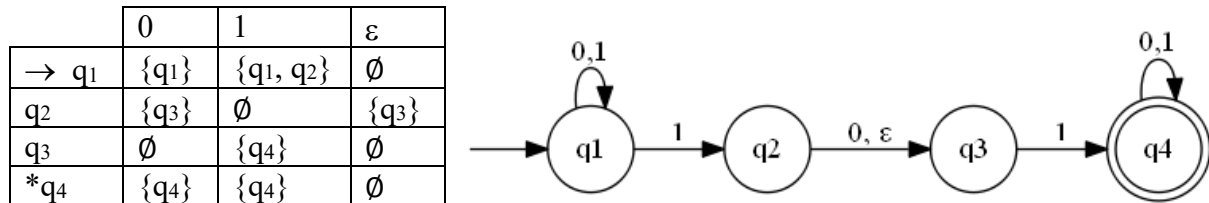
Automates finis non déterministes (AFN)

Un automate fini non déterministe (AFN) est défini par un quintuplé $A = (\Sigma, Q, \delta, q_0, F)$ où :

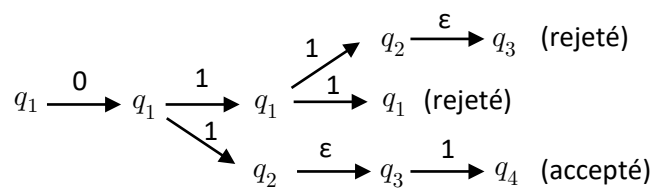
- Σ est un ensemble fini de symboles appelé alphabet.
- Q est un ensemble fini dont les éléments sont appelés états.
- $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \rightarrow P(Q)$ est la fonction de transition, qui associe à un état et à un symbole (ou à ε) un ensemble d'états.
- $q_0 \in Q$ est l'état initial.
- $F \subseteq Q$ est l'ensemble des états finaux.

Un AFN peut comporter des transitions étiquetées par ε ainsi que des états possédant plusieurs transitions sortantes portant la même étiquette. Un mot w est reconnu par un AFN s'il existe au moins un chemin étiqueté par w , partant de l'état initial et menant à un état final, c'est-à-dire si, $\delta(q_0, w) \cap F \neq \emptyset$.

Exemple : On considère l'AFN suivant



Est-ce que cet automate accepte le mot 011 ?

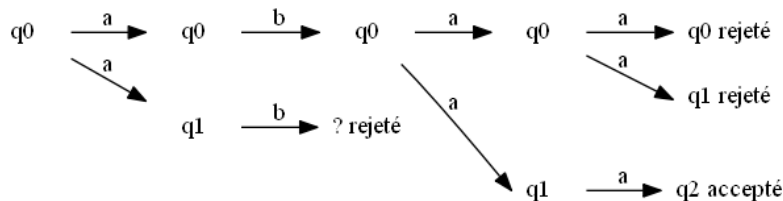


Le mot 011 est accepté car l'un des chemins aboutit à l'état final q_4 .

Exemple : Soit l'AFN suivant



Est ce que cet automate accepte le mot abaa ?



Le mot abaa est accepté car l'un des chemins aboutit à l'état final q_2 .

Langage reconnu (accepté) par un AFN

Le langage $L_m(A)$ reconnu par un AFN A est l'ensemble des mots permettant d'atteindre un état final à partir de l'état initial : $L_m(A) = \{\omega \in \Sigma^* \mid \delta(q_0, \omega) \cap F \neq \emptyset\}$

Remarque :

- Tout AFN possédant plusieurs états initiaux peut être transformé en un AFN standard, muni d'un unique état initial, reconnaissant le même langage. Il suffit d'ajouter un nouvel état initial unique et de relier celui-ci à tous les anciens états initiaux par des ε -transitions. Les autres éléments de l'automate restent inchangés.
- Un AFN explore conceptuellement tous les chemins possibles correspondant au mot lu. Si au moins l'un de ces chemins atteint un état final, le mot est reconnu.
- Pour tout automate fini A , il existe une expression régulière dénotant le langage $L_m(A)$.

- Pour toute expression régulière ER , il existe un automate fini qui reconnaît le langage qu'elle définit.

Passage de l'automate fini vers l'expression régulière

Etant donné un automate fini A , on souhaite construire une expression régulière décrivant le langage $L_m(A)$. Le principe consiste à associer à chaque état q une inconnue L_q représentant le langage reconnu à partir de cet état, puis à établir un système d'équations. Pour chaque état q , on écrit :

- Si l'état q n'est pas un état final : $L_q = \sum_{(q \xrightarrow{a} p) \in \delta} aL_p$
- Si l'état q est un état final, alors ε appartient à L_q : $L_q = \sum_{(q \xrightarrow{a} p) \in \delta} aL_p + \varepsilon$

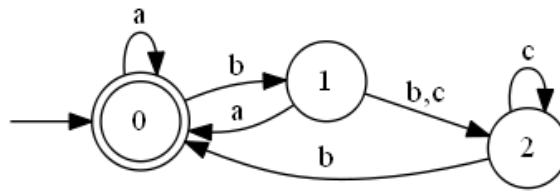
On obtient ainsi un système de N équations à N inconnues (N est le nombre d'états de l'automate). L'objectif est de déterminer une expression régulière pour L_{q_0} , associée à l'état initial.

Lemme d'Arden

L'équation récursive : $L = XL + M$, où L est le langage inconnu et X et M sont des langages connus, admet une solution unique : $L = X^*M$ à condition que $\varepsilon \notin X$.

Exemple :

1) Langage reconnu par cet AFD.



$$\begin{cases} L_0 = aL_0 + bL_1 + \varepsilon \\ L_1 = aL_0 + (b+c)L_2 \\ L_2 = bL_0 + cL_2 \end{cases} \Rightarrow \begin{cases} L_0 = aL_0 + bL_1 + \varepsilon \\ L_1 = aL_0 + (b+c)L_2 \\ L_2 = c^*bL_0 \end{cases} \Rightarrow \begin{cases} L_0 = aL_0 + bL_1 + \varepsilon \\ L_1 = (a + (b+c)c^*b)L_0 \end{cases}$$

Ce qui donne :

$$L_0 = aL_0 + b(a + (b+c)c^*b)L_0 + \varepsilon = (a + b(a + (b+c)c^*b))L_0 + \varepsilon \\ \Rightarrow L_0 = (a + b(a + (b+c)c^*b))^*$$

Alors, le langage reconnu par cet AFD est donné par l'expression régulière :

$$L_m(A) = (a + b(a + (b+c)c^*b))^*$$

2) Langage reconnu par cet AFD.

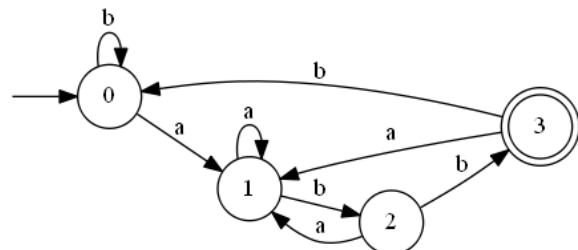
$$\begin{cases} L_0 = bL_0 + aL_1 \\ L_1 = aL_1 + bL_2 \\ L_2 = aL_1 + bL_3 \\ L_3 = aL_1 + bL_0 + \varepsilon \end{cases}$$

Ce qui donne :

$$L_2 = (a + ba)L_1 + bbL_0 + b \\ L_1 = (a + b(a + ba))L_1 + bbbL_0 + bb \Rightarrow L_1 = (a + b(a + ba))^*(bbbL_0 + bb) \\ \Rightarrow L_0 = (b + a(a + b(a + ba))^*bbb)L_0 + a(a + b(a + ba))^*bb \\ \Rightarrow L_0 = (b + a(a + b(a + ba))^*bbb)^*a(a + b(a + ba))^*bb$$

Alors, le langage reconnu par cet AFD est donné par l'expression régulière :

$$L_m(A) = (b + a(a + b(a + ba))^*bbb)^*a(a + b(a + ba))^*bb$$



Remarques :

- Un mot est accepté par un AFN s'il existe une exécution menant à un état final.
- Tout langage accepté par un AFN est aussi accepté par un AFD.
- Un langage est régulier si et seulement s'il est reconnu par un automate fini.
- Le non-déterminisme simplifie la construction des automates sans augmenter leur puissance expressive.
- AFD et AFN reconnaissent exactement la même classe de langages réguliers.
- Ce résultat est à la fois surprenant et utile :
 - Il est surprenant, car les automates non déterministes semblent a priori plus puissants ;
 - Il est utile, car il est souvent plus simple de décrire un langage à l'aide d'un automate non déterministe.

Passage de l'expression régulière vers l'automate fini

Tout langage décrit par une expression régulière est reconnaissable par un automate fini. Ce résultat fondamental établit l'équivalence entre les expressions régulières et les automates finis.

Algorithme de Glushkov

C'est un algorithme qui permet de construire un automate fini (pas nécessairement déterministe et sans ε -transitions) à partir d'une expression régulière.

Entrée : une expression régulière : $(aba + b)^* + a^*$

Étape 1 : Linéariser l'expression régulière.

On remplace toutes les lettres de l'expression par des symboles distincts (x_i pour la lettre en i -ème position) : $(x_1x_2x_3 + x_4)^* + x_5^*$.

Étape 2 : Déterminer

- *Premier* : l'ensemble des symboles pouvant commencer un mot.
- *Dernier* : l'ensemble des symboles pouvant terminer un mot.
- *Suivant* : pour tout symbole x_i , l'ensemble des symboles pouvant suivre x_i .

Étape 3 : Construction de l'AFN

- L'ensemble des états :
 - Un état initial q_0 .
 - Un état q_i par symbole x_i .
- L'ensemble des états finaux :
 - L'état initial q_0 si ε appartient au langage.
 - Un état q_i pour tout x_i appartenant à *Dernier*.
- La fonction de transition :
 - Une transition de l'état initial q_0 vers l'état q_i pour tout x_i appartenant à *Premier* et étiquetée par la lettre correspondant à x_i , i.e. $\delta(q_0, x_i) = q_i$ si $x_i \in \text{Premier}$.
 - Une transition de l'état q_i vers l'état q_j pour tout x_j appartenant à *Suivant*(x_i) et étiquetée par la lettre correspondant à x_j , i.e. $\delta(q_i, x_j) = q_j$ si $x_j \in \text{Suivant}(x_i)$.
 - Puis remplacer les symboles x_i par les lettres d'origine de l'expression régulière.

Remarque : L'automate obtenu est généralement un AFN sans ε -transitions qui reconnaît exactement le langage décrit par l'expression régulière.

Exemple :

Construction d'un AFN qui reconnait le langage : $(aba + b)^* + a^*$

Linéarisation de l'expression régulière : $(x_1x_2x_3 + x_4)^* + x_5^*$

Premier = $\{x_1, x_4, x_5\}$

Dernier = $\{x_3, x_4, x_5\}$

Ensemble des états de l'automate fini

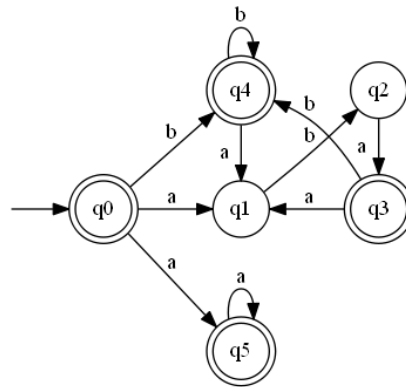
$Q = \{q_0, q_1, q_2, q_3, q_4, q_5\}$

Etat initial : q_0

Etats finals : $\{q_0, q_3, q_4, q_5\}$

(ε appartient au langage $\Rightarrow$ l'état initial est un état final).

	Suivant
x_1	x_2
x_2	x_3
x_3	x_1, x_4
x_4	x_1, x_4
x_5	x_5



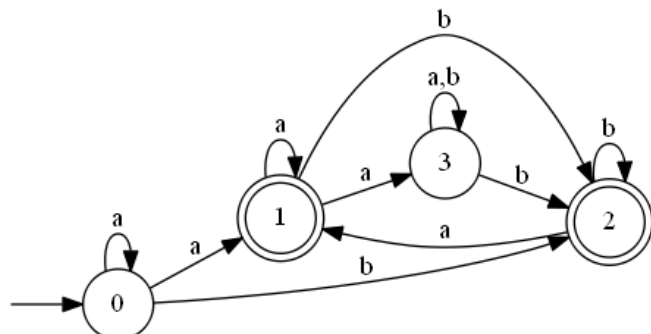
Déterminisation d'un automate

Théorème : Pour tout automate fini non déterministe, il existe un automate fini déterministe équivalent, c'est-à-dire reconnaissant le même langage.

Procédure de déterminisation :

- **Étape 1 :** calculer l' ε -fermeture (clôture, étoile) de chaque état.
L' ε -fermeture d'un état q est l'ensemble des états accessibles à partir de q en ne suivant que des ε -transitions, auquel on ajoute q lui-même.
- **Étape 2 :** Créer la table de transition du nouvel automate comme suit
 1. Commencer par l' ε -fermeture de l'état initial. Cet ensemble constitue l'état initial du nouvel automate.
 2. Pour chaque lettre, ajouter dans la table les états produits, avec leurs ε -fermetures. Chaque ensemble obtenu devient un nouvel état.
 3. Répéter le procédé pour chaque nouvel état jusqu'à ce qu'aucun nouvel ensemble n'apparaisse.
 4. Tout état contenant au moins un état final de l'automate initial est déclaré final.
 5. Pour plus de clarté, renommer éventuellement les états obtenus.

Exemple : Déterminiser l'AFN suivant :



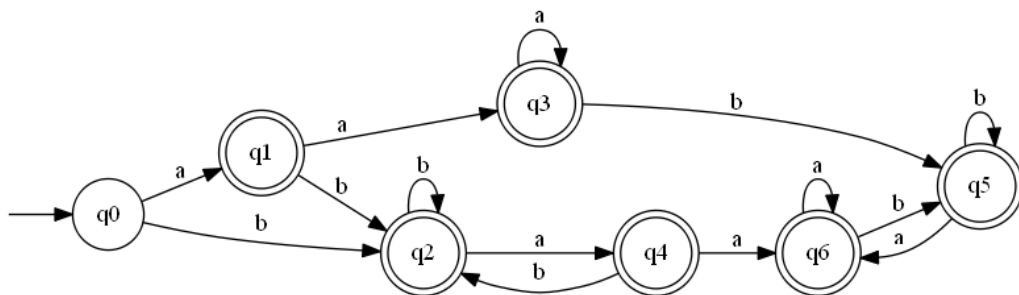
Dans cet automate, on n'a pas de ε -transitions, alors l' ε -fermeture de chaque est l'état lui-même.

La table de transition de l'automate fini déterministe résultant est :

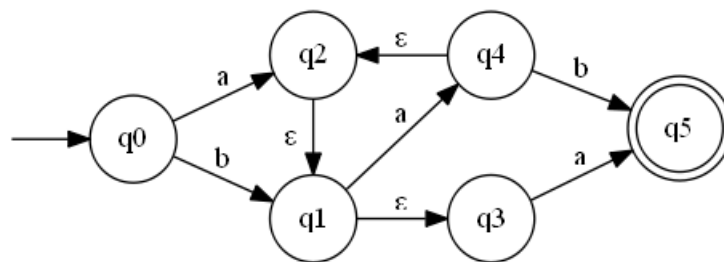
	a	b
$\rightarrow 0$	$\{0, 1\}$	$\{2\}$
$* \{0, 1\}$	$\{0, 1, 3\}$	$\{2\}$
$* \{2\}$	$\{1\}$	$\{2\}$
$* \{0, 1, 3\}$	$\{0, 1, 3\}$	$\{2, 3\}$
$* \{1\}$	$\{1, 3\}$	$\{2\}$
$* \{2, 3\}$	$\{1, 3\}$	$\{2, 3\}$
$* \{1, 3\}$	$\{1, 3\}$	$\{2, 3\}$

Après renommage des états,
on obtient :

	a	b
$\rightarrow q0$	q1	q2
$* q1$	q3	q2
$* q2$	q4	q2
$* q3$	q3	q5
$* q4$	q6	q2
$* q5$	q6	q5
$* q6$	q6	q5



Exemple : Construire un AFD équivalent à l'AFN suivant.



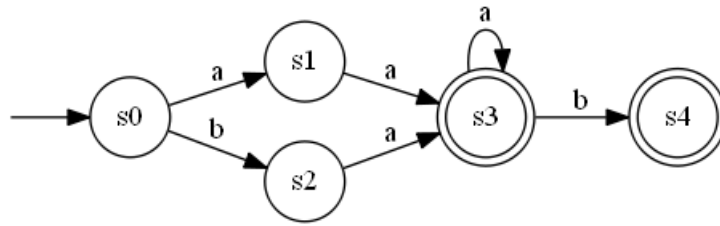
Calcul de l' ε -fermeture (clôture, étoile) de chaque état

état	ε -fermeture
q0	$\{q0\}$
q1	$\{q1, q3\}$
q2	$\{q1, q2, q3\}$
q3	$\{q3\}$
q4	$\{q1, q2, q3, q4\}$
q5	$\{q5\}$

Table de transition de l'automate fini déterministe

état	a	b
$\rightarrow q0$	$\{q1, q2, q3\}$	$\{q1, q3\}$
$\{q1, q2, q3\}$	$\{q1, q2, q3, q4, q5\}$	$\emptyset$
$\{q1, q3\}$	$\{q1, q2, q3, q4, q5\}$	$\emptyset$
$* \{q1, q2, q3, q4, q5\}$	$\{q1, q2, q3, q4, q5\}$	$\{q5\}$
$* q5$	$\emptyset$	$\emptyset$

	a	b
$\rightarrow s0$	s1	s2
s1	s3	$\emptyset$
s2	s3	$\emptyset$
$* s3$	s3	s4
$* s4$	$\emptyset$	$\emptyset$



Minimisation d'un AFD

La minimisation d'un automate fini déterministe (AFD) consiste à réduire le nombre d'états d'un automate tout en conservant le même langage reconnu.

L'idée est de fusionner les états équivalents, c'est-à-dire des états qui se comportent de manière identique pour tous les mots possibles.

Définition. Deux états p et q sont équivalents (indistinguables) si aucun mot ne les sépare. Autrement dit, $p \sim q$ si et seulement si pour tout mot ω , on a : $\delta(p, \omega) \in F \iff \delta(q, \omega) \in F$.

Théorème de Myhill-Nérode : Soit L un langage régulier. Parmi tous les AFD reconnaissant L , il en existe un et un seul qui a un nombre minimal d'états.

Procédure de minimisation – Algorithme de table de distinction (ou méthode de Moore)

Etape 1 : Supprimer les états inaccessibles et compléter l'automate.

- On commence par éliminer tous les états qui ne sont pas accessibles à partir de l'état initial.
- Si l'automate est incomplet, on le complète en ajoutant un état puits pour toutes les transitions manquantes.

Etape 2 : Créer une table des paires d'états de tous les couples (p, q) avec $p \neq q$.

- Commencer avec un tableau de taille $N \times N$ (N est le nombre d'états de l'automate). Chaque case dans ce tableau correspond à un couple d'états (p, q) .
- Pour ne pas avoir deux fois le même couple (p, q) et (q, p) ni les couples inutiles (p, p) , supprimer les cases (q, p) si (p, q) existe déjà et les cases (p, p) . On obtient un tableau (ou matrice) triangulaire inférieur.
- L'objectif est de marquer toutes les paires d'états distinguables.

Etape 3 : Marquer les paires immédiatement distinguables.

- Au départ, les paires (p, q) où $p \in F$ et $q \notin F$ (ou inversement), sont marquées immédiatement par D .
- Ces états sont directement non équivalents.

Etape 4 : Propager les distinctions.

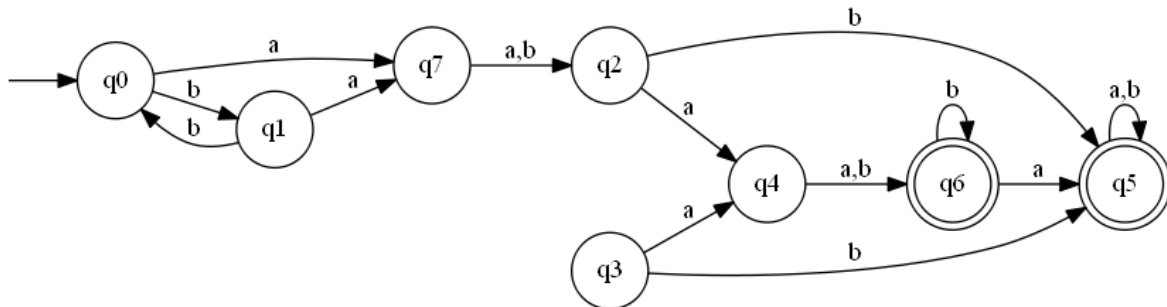
- Pour chaque paire (p, q) non marquée et pour chaque symbole a de l'alphabet, calculer $\delta(p, a) = p'$ et $\delta(q, a) = q'$. Si la case (p', q') est déjà marquée, marquer (p, q) .
- Répéter ce processus jusqu'à ce que le tableau ne change plus (stabilisation).

Etape 5 : Construire l'automate fini minimal (réduit)

- Déterminer les classes d'équivalence : Pour chaque état q , sa classe d'équivalence contient tous les états p pour lesquels (p, q) n'est pas marquée dans le tableau.
- Définir les états de l'automate minimal : Chaque classe d'équivalence devient un état de l'automate minimal. L'état initial est la classe contenant l'état initial original q_0 . Les états finaux de l'automate minimal sont les classes d'équivalence qui se composent des états finaux de l'automate original.

- Définir la fonction de transition : Pour une classe X et un symbole a , choisir n'importe quel état $q \in X$, la transition $\delta'(X, a)$ est la classe d'équivalence qui contient $\delta(q, a)$.
- Supprimer tous les états non accessibles et tous les états non co-accessibles. Toutes les transitions vers ces états depuis d'autres états deviennent indéfinies.

Exemple : Minimiser l'AFD suivant

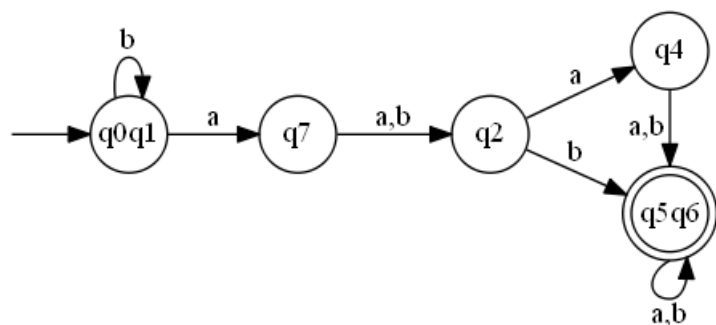


En premier lieu, on supprime l'état q_3 parce que cet état n'est pas accessible à partir de l'état initial. Ensuite, on construit la table d'équivalence (sans q_3)

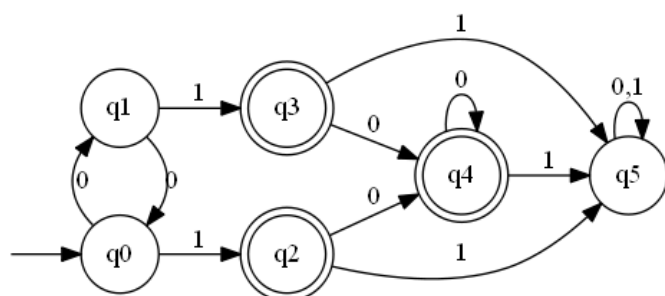
q1							
q2	D	D					
q4	D	D	D				
q5	D	D	D	D			
q6	D	D	D	D			
q7	D	D	D	D	D	D	
	q0	q1	q2	q4	q5	q6	

On regroupe les états dans des classes d'équivalence. A partir du tableau, q_0 et q_1 sont équivalents, et q_5 et q_6 sont équivalents. Alors, les classes d'équivalences sont : $\{q_0, q_1\}$, $\{q_2\}$, $\{q_4\}$, $\{q_5, q_6\}$, et la table de transitions de l'automate minimale est la suivante :

	a	b
$\rightarrow q_0q_1$	q_7	q_0q_1
q_2	q_4	q_5q_6
q_4	q_5q_6	q_5q_6
$* q_5q_6$	q_5q_6	q_5q_6
q_7	q_2	q_2



Exemple : Minimiser l'AFD suivant

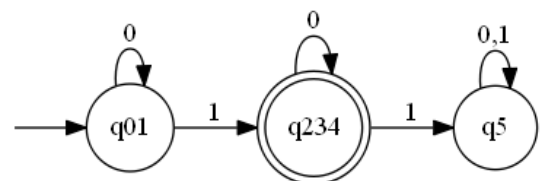


En premier lieu, Il n'y a pas d'état inaccessible. On construit alors la table d'équivalence.

q1					
q2	D	D			
q3	D	D			
q4	D	D			
q5	D	D	D	D	D
	q0	q1	q2	q3	q4

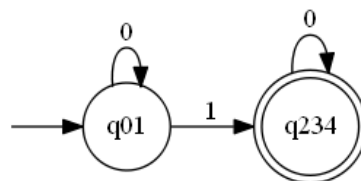
On regroupe les états dans des classes d'équivalence. A partir du tableau, $q0$ et $q1$ sont équivalents, et $q2$, $q3$ et $q4$ sont équivalents. Alors, les classes d'équivalences sont : $q_{01} = \{q0, q1\}$, $q_{234} = \{q2, q3, q4\}$ et $\{q5\}$. La table de transitions de l'automate résultant est la suivante :

	0	1
$\rightarrow q_{01}$	q_{01}	q_{234}
q_{234}	q_{234}	q_5
q_5	q_5	q_5



Le graphe de l'automate obtenu est :

On remarque que l'état $q5$ n'est pas co-accessible, alors on supprime cet état, et l'automate minimal devient :



Opérations sur les automates (les langages)

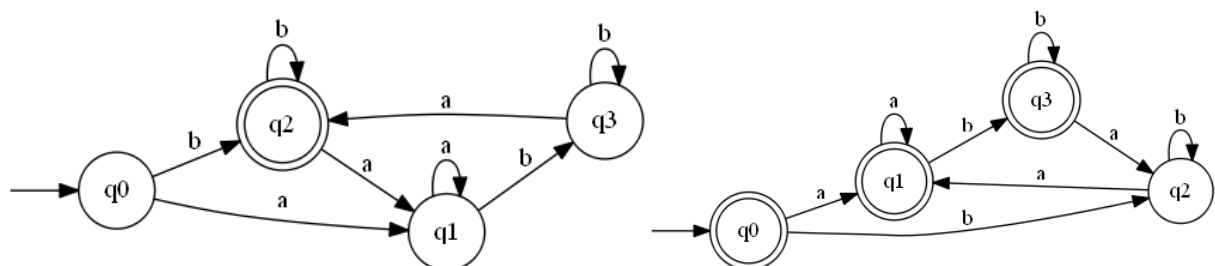
On considère des automates finis déterministes.

Complément

Soit un automate déterministe A reconnaissant le langage L . L'automate complémentaire reconnaissant le langage inverse (c'est-à-dire L^c ou $\Sigma^* - L$) peut être obtenu comme suit :

- Compléter l'automate A s'il n'est pas complet.
- L'automate complémentaire peut être obtenu en transformant les états non finaux à des états finaux et vice-versa.

Exemple :



Concaténation

Soit deux automates déterministes $A_1 = (\Sigma_1, Q_1, \delta_1, q_{10}, F_1)$ et $A_2 = (\Sigma_2, Q_2, \delta_2, q_{20}, F_2)$ reconnaissant respectivement les langages L_1 et L_2 . Pour construire l'automate A qui reconnaît le langage $L = L_1 L_2$, on peut procéder de la manière suivante :

- L'ensemble des états du nouvel automate A est $Q = Q_1 \cup Q_2$.
- L'état initial de l'automate A est l'état initial du premier automate A_1 .
- Les états finaux de l'automate A sont les états finaux du deuxième automate A_2 .
- Chaque transition des deux automates A_1 et A_2 est une transition dans l'automate A .
- On ajoute de nouvelles ε -transitions entre chaque état final de A_1 (qui n'est plus final dans le nouvel automate A) et l'état initial de A_2 (qui n'est plus initial dans le nouvel automate A).

Union (Somme des automates)

Soit deux automates déterministes $A_1 = (\Sigma_1, Q_1, \delta_1, q_{10}, F_1)$ et $A_2 = (\Sigma_2, Q_2, \delta_2, q_{20}, F_2)$ reconnaissant respectivement les langages L_1 et L_2 . L'union de deux langages consiste à construire un automate reconnaissant : $L = L_1 \cup L_2 = L_1 + L_2$, c'est-à-dire l'ensemble des mots appartenant à l'un au moins des deux langages. L'automate fini reconnaissant L peut être construit de la manière suivante :

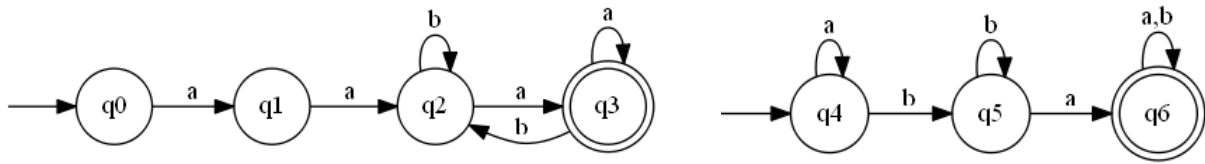
- L'état initial est le couple (q_{10}, q_{20}) ,
- L'alphabet est $\Sigma = \Sigma_1 \cup \Sigma_2$,
- Pour chaque état créé (p, q) et pour chaque lettre a de l'alphabet Σ , la transition est définie par :
$$\delta((p, q), a) = (p', q'), \text{ avec } p' = \begin{cases} \delta_1(p, a), & \text{si } \delta_1(p, a) \text{ est définie} \\ \emptyset, & \text{sinon} \end{cases}, q' = \begin{cases} \delta_2(q, a), & \text{si } \delta_2(q, a) \text{ est définie} \\ \emptyset, & \text{sinon} \end{cases}$$
- Chaque nouvel état obtenu est ajouté à l'ensemble des états.
- Le procédé est répété jusqu'à ce qu'aucun nouvel état ne soit créé.
- Un état (p, q) est final si $p \in F_1$ ou $q \in F_2$.
- Pour plus de clarté, renommer éventuellement les états.

Intersection (Produit des automates)

Soit deux automates déterministes $A_1 = (\Sigma_1, Q_1, \delta_1, q_{10}, F_1)$ et $A_2 = (\Sigma_2, Q_2, \delta_2, q_{20}, F_2)$ reconnaissant respectivement les langages L_1 et L_2 . L'automate fini reconnaissant le langage $L = L_1 \cap L_2$ peut être construit comme suit :

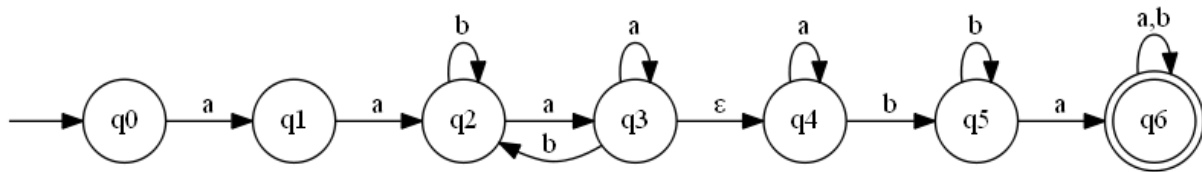
- L'état initial est le couple (q_{10}, q_{20}) ,
- L'alphabet est $\Sigma = \Sigma_1 \cap \Sigma_2$
- Pour chaque état créé (p, q) et pour chaque lettre a de l'alphabet Σ , la transition est définie par :
$$\delta((p, q), a) = (p', q'), \text{ si } \delta_1(p, a) = p' \text{ et } \delta_2(q, a) = q'$$
- Chaque nouvel état obtenu est ajouté à l'ensemble des états.
- Le procédé est répété jusqu'à stabilisation (aucun nouvel état créé).
- Un état (p, q) est final si et seulement si $p \in F_1$ et $q \in F_2$.
- Pour plus de clarté, renommer éventuellement les états.

Exemple : Soit A_1 l'automate de gauche et A_2 l'automate de droite. Soient L_1 et L_2 leurs langages respectifs



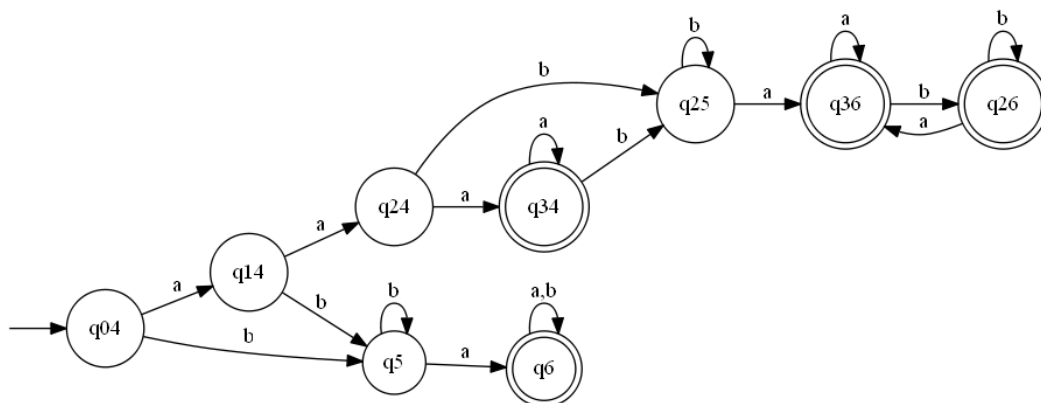
- Construire l'automate reconnaissant le langage $L_1 L_2$.
- Construire l'automate reconnaissant le langage $L = L_1 + L_2$.
- Construire l'automate reconnaissant le langage $L = L_1 \cap L_2$.

○ Automate pour le langage $L_1 L_2$



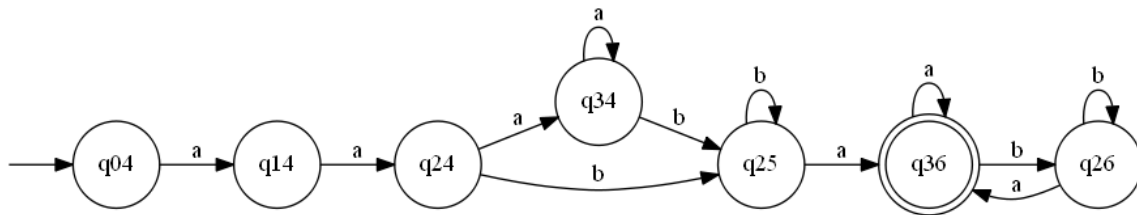
○ Automate reconnaissant le langage $L = L_1 + L_2$

		a	b
$\rightarrow q_{04}$	$\rightarrow (q_0, q_4)$	(q_1, q_4)	q_5
q_{14}	(q_1, q_4)	(q_2, q_4)	q_5
q_5	q_5	q_6	q_5
q_{24}	(q_2, q_4)	(q_3, q_4)	(q_2, q_5)
$*q_6$	$*q_6$	q_6	q_6
$*q_{34}$	$*(q_3, q_4)$	(q_3, q_4)	(q_2, q_5)
q_{25}	(q_2, q_5)	(q_3, q_6)	(q_2, q_5)
$*q_{36}$	$*(q_3, q_6)$	(q_3, q_6)	(q_2, q_6)
$*q_{26}$	$*(q_2, q_6)$	(q_3, q_6)	(q_2, q_6)



- Automate reconnaissant le langage $L = L_1 \cap L_2$

		a	b
→ q04	→ (q0,q4)	(q1,q4)	-
q14	(q1,q4)	(q2,q4)	-
q24	(q2,q4)	(q3,q4)	(q2,q5)
q34	(q3,q4)	(q3,q4)	(q2,q5)
q25	(q2,q5)	(q3,q6)	(q2,q5)
*q36	*(q3,q6)	(q3,q6)	(q2,q6)
q26	(q2,q6)	(q3,q6)	(q2,q6)



Composition d'automates

L'opération de composition d'automates permet de construire le modèle global d'un système à partir des modèles de ses sous-systèmes. On distingue deux types de composition :

- *Composition synchrone* : Elle est utilisée lorsque les alphabets des automates considérés possèdent au moins un événement commun ; ces événements sont alors exécutés simultanément par les automates.
- *Composition asynchrone* : Elle intervient lorsque les alphabets des automates ne partagent aucun événement ; chaque automate évolue alors indépendamment de l'autre.

Soient deux automates $A_1 = (\Sigma_1, Q_1, \delta_1, q_{10}, F_1)$ et $A_2 = (\Sigma_2, Q_2, \delta_2, q_{20}, F_2)$. La composition entre A_1 et A_2 est l'automate $A = A_1 \parallel A_2 = (\Sigma, Q, \delta, q_0, F)$, où :

- Ensemble des états : $Q = Q_1 \times Q_2$,
- Alphabet : $\Sigma = \Sigma_1 \cup \Sigma_2$,
- Etat initial : $q_0 = (q_{10}, q_{20})$,
- Ensemble des états finaux : $F = F_1 \times F_2$,
- Fonction de transition $\delta : Q \times \Sigma \rightarrow Q$, elle associe à chaque état (p, q) et à tout événement $\sigma \in \Sigma$ un état calculé de la façon suivante :
 - Pour $\sigma \notin \Sigma_1 \cap \Sigma_2$ (événements propres à chaque automate)
 - $\delta((p, q), \sigma) = (p', q)$ si $\sigma \in \Sigma_1$ et $\delta_1(p, \sigma) = p'$,
 - $\delta((p, q), \sigma) = (p, q')$ si $\sigma \in \Sigma_2$ et $\delta_2(q, \sigma) = q'$,
 - Ces événements se produisent indépendamment dans l'automate concerné.
 - Pour $\sigma \in \Sigma_1 \cap \Sigma_2$ (événements communs)
 - $\delta((p, q), \sigma) = (p', q')$ si $\delta_1(p, \sigma) = p'$ et $\delta_2(q, \sigma) = q'$
 - Ces événements se produisent simultanément dans les deux automates.

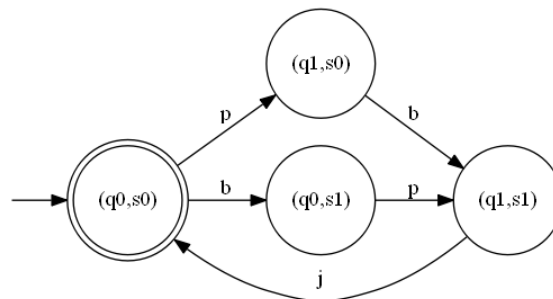
Remarque : Si un événement $\sigma \in \Sigma_1 \cap \Sigma_2$, il doit se produire de manière synchrone dans les deux automates. En revanche, s'il n'appartient qu'à l'un des deux alphabets, il se produit de manière asynchrone.

Exemple : Considérons deux automates pour modéliser le fonctionnement d'un distributeur de jetons. On a $\Sigma_1 = \{p, j\}$ et $\Sigma_2 = \{b, j\}$

Pour A_1 , on suppose que le symbole p modélise l'événement « introduction d'une pièce » dans la machine et que j modélise l'événement « libération d'un jeton ». Pour A_2 , le symbole b modélise l'événement « pression d'un bouton » et le symbole j a la même signification que dans A_1 .



	p	b	j
$\rightarrow * (q_0, s_0)$	(q_1, s_0)	(q_0, s_1)	-
(q_1, s_0)	-	(q_1, s_1)	-
(q_0, s_1)	(q_1, s_1)	-	-
(q_1, s_1)	-	-	(q_0, s_0)



Remarques :

- L'événement commun j se produit simultanément dans les deux automates.
- Les événements propres à un automate (p pour A_1 , b pour A_2) se produisent indépendamment.