Chapitre 1 : Langages réguliers et automates (1)

1 Mots et langages, rappels

Définition 1.1

*Un **alphabet** est un ensemble fini de symboles, aussi appelés **caractères** ou **lettres**.*

Exemple 1.2

On utilisera souvent les lettres de l'alphabet latin `'a'`, `'b'`, ...

Définition 1.3

Soit Σ un alphabet non vide.
*Un **mot** sur Σ est une suite finie de caractères de Σ.*
*La **longueur** d'un mot x, notée $|x|$, est le nombre de caractères dans x comptés avec multiplicité.*
Si $|x| = n \in \mathbb{N}^*$, on indexe les caractères de x par $[\![1, n]\!]$, le mot x s'écrit alors $x_1\, x_2\, \ldots x_n$.

Définition 1.4

*Peu importe l'alphabet, il existe un seul mot de longueur nulle : **le mot vide**. On le note ε.*

Remarque 1.5

Le mot vide ε **n**'est donc **pas** une lettre.

Définition 1.6

Soit Σ un alphabet non vide.
On note $\Sigma^\star$ l'ensemble des mots sur Σ, et Σ^+ l'ensemble des mots non vides sur Σ.
De plus pour $n \in \mathbb{N}$, on note Σ^n l'ensemble des mots sur Σ de longueur n.

Exercice de cours 1.7

Donner une définition de Σ^+ (resp. de $\Sigma^\star$) comme une union de Σ^n.

Définition 1.8

Soit Σ un alphabet non vide. Soient x et y deux mots sur Σ de longueur respectives m et n.
*La **concaténation** des mots x et y, notée $x \cdot y$ et y est le mot $x_1\, x_2\, \ldots x_m\, y_1\, y_2\, \ldots y_n$.*

Remarque 1.9

Intermède mathématique. On appelle aussi concaténation l'opération qui à x et y associe $x \cdot y$.

- Puisque cette opération envoie deux mots de $\Sigma^\star$ sur un mot de $\Sigma^\star$, on parle d'opération interne, (en faisant cette opération on reste à l'intérieur de l'ensemble). Formellement : $\forall (x, y) \in (\Sigma^\star)^2, (x, y) \in \Sigma^\star$.
- Cette opération admet un élément neutre : le mot vide. Cela signifie qu'en concaténant un mot avec le mot vide, avant ou après, on ne change pas le mot. Formellement : $\forall x \in \Sigma^\star, \varepsilon \cdot x = x \cdot \varepsilon = x$.
- Cette opération est associative, cela signifie que concaténer y à x, puis z au mot obtenu, revient au même que d'abord concaténer z à y, puis le mot obtenu à x. Formellement : $\forall (x, y, z) \in (\Sigma^\star)^3, (x.y).z = x \cdot (y \cdot z)$.

En algèbre, on dit que $(\Sigma^\star, \cdot)$ est un monoïde pour désigner cette structure.

Remarque 1.10

Notez que l'opération de concaténation n'est pas commutative dès que l'alphabet Σ a plus de deux lettres, c'est-à-dire qu'en général $x \cdot y$ et $y \cdot x$ ne sont pas les mêmes mots.

Exercice de cours 1.11

Donner un exemple illustrant la remarque précédente.

Définition 1.12

Soient x et y deux mots sur un alphabet non vide Σ. Notons $n = |x|$ et $m = |y|$.

- *On dit que y est un **facteur** de x dès lors qu'il existe deux mots u et v sur Σ tels que $x = u \cdot y \cdot v$.*
- *On dit que y est un **préfixe** de x dès lors qu'il existe un mot v sur Σ tel que $x = y \cdot v$.*
- *On dit que y est un **suffixe** de x dès lors qu'il existe un mot u sur Σ tel que $x = u \cdot y$.*

Remarque 1.13

- Notez que les mots u et v dans la décomposition de x autour du facteur y ne sont pas uniques, tandis que les mots u et v dans la décomposition de x selon le préfixe ou suffixe y le sont.
- Par définition, les préfixes et suffixes sont des facteurs.
- Le mot vide est préfixe et suffixe de n'importe quel mot.
- N'importe quel mot est toujours son propre préfixe et son propre suffixe.

Exercice de cours 1.14

Illustrer et/ou démontrer chacun des points de la remarque ci-dessus en revenant aux définitions.

Définition 1.15

Soient x et y deux mots sur un alphabet non vide Σ. Notons $n = |x|$ et $m = |y|$.
*On dit que y est un **sous-mot** de x dès lors qu'il existe σ une fonction strictement croissante de $[\![1, m]\!]$ dans $[\![1, n]\!]$ telle que $y = x_{\sigma(1)} x_{\sigma(2)} \dots x_{\sigma(m)}$* ♣.

Remarque 1.16

- Par définition, les facteurs sont des sous-mots, mais le contraire est faux.
- Le mot vide est sous-mot de n'importe quel mot.
- N'importe quel mot est toujours sous-mot de lui-même.

Exercice de cours 1.17

Illustrer et/ou démontrer chacun des points de la remarque ci-dessus en revenant aux définitions.

Exemple 1.18

L'ensemble $\Sigma = \{a, b, c\}$ est un alphabet. $x = ba$, $y = aaa$, $z = babaaa$ sont des mots sur Σ. On a $|x| = 2$, $|y| = 3$ et $|z| = 6$. La concaténation de x et y est $z' = baaaa$, qui est un sous-mot de z, mais pas un facteur de z. Le mot x est un préfixe de z et z', mais pas de y, qui lui est suffixe de z et z'.

Définition 1.19

*Un **langage** sur un alphabet non vide Σ est un ensemble de mots sur Σ, soit une partie de $\Sigma^\star$.*

♣. En mathématiques, on dirait que y est une suite extraite de x, à ceci près qu'il s'agit de suites finies.

Exemple 1.20
Supposons dans cet exemple que $\Sigma = \{a, b\}$. Alors les ensembles suivants sont des langages sur Σ : $\emptyset$, $\Sigma^\star$, $\{abb, aa, baa\}$, $\{\underbrace{aa\ldots a}_{n}\underbrace{bb\ldots b}_{n} \mid n \in \mathbb{N}\}$.

Exercice de cours 1.21

1. Parmi les langages ci-dessus, lesquels sont finis, lequels sont infinis ?
2. Justifier que si Σ n'est pas vide, l'ensemble des langages sur Σ est infini.

Dans toute la suite du chapitre Σ désignera un alphabet non vide.

2 Langages réguliers

2.1 Opérations sur les langages

Les langages étant définis comme des ensembles, on peut appliquer aux langages les opérations ensemblistes usuelles : l'union, l'intersection, le complémentaire (par rapport à $\Sigma^\star$), la différence, la différence symétrique. On définit ci-après quelques autres opérations, plus spécifiques aux langages.

Définition 2.1

Soient L_1 et L_2 deux langages sur Σ.
*La **concaténation** de L_1 et L_2 est $L_1 \cdot L_2 = \{u \cdot v \mid u \in L_1,\ v \in L_2\}$.*

Exercice de cours 2.2

Donner $\{aa, aaa\}\cdot\{b, bb\}$, $\{aa, aaa\}\cdot\{a, aa\}$ et finalement $\{\underbrace{aa\ldots a}_{n} \mid n \in \mathbb{N}\}\cdot\{\underbrace{bb\ldots b}_{n} \mid n \in \mathbb{N}\}$.

Remarque 2.3

L'opération de concaténation des langages, comme celle sur les mots, est associative et non commutative. L'élément neutre de cette opération est le langage réduit au mot vide : $\{\varepsilon\}$. L'ensemble vide est quant à lui absorbant pour la concaténation. L'opération $\cdot$ est distributive à gauche et à droite par rapport à $\cup$, *i.e.* $\forall (K, L, M) \in (\mathcal{P}(\Sigma^\star))^3$, $K\cdot(L\cup M) = (K\cdot L)\cup(K\cdot M)$ et $(K\cup L)\cdot M = (K\cdot M)\cup(L\cdot M)$.

Définition 2.4

On note en exposant l'itéré de la concaténation, ainsi pour L un langage sur Σ, $L^0 = \{\varepsilon\}$ et pour $n \in \mathbb{N}^*$, $L^n = L.L^{n-1}$. De plus on note $L^\star = \bigcup\limits_{n\in\mathbb{N}} L^n$ et $L^+ = \bigcup\limits_{n\in\mathbb{N}^*} L^n$.
*L'opération qui à L associe $L^\star$ est appelée **étoile de Kleene**.*

Exercice de cours 2.5

Dans le cas où $L = \{\varepsilon, aa, b\}$ décrire au moyen d'une phrase en français le langage $L^\star$.

Définition 2.6

*Les **opérations régulières** sur les langages sont l'union, la concaténation et l'étoile de Kleene.*

Définition 2.7

*L'ensemble des **langages réguliers** sur Σ est le plus petit ensemble de $\mathcal{P}\Big(\mathcal{P}(\Sigma^\star)\Big)$ contenant $\emptyset$ et $\{a\}$ pour tout $a \in \Sigma$ et stable par les opérations régulières.*

Remarque 2.8

La définition 2.7 est une définition inductive de l'ensemble des langages réguliers.

Exercice de cours 2.9

Justifier que le langage $\{aa, aab, \varepsilon\}$ est bien un langage régulier.
Justifier que le langage $\{bb, bab, baab, \ldots\}$, *i.e.* $\{b\underbrace{aa\ldots a}_{n}b, \ldots \mid n \in \mathbb{N}\}$, est lui aussi régulier.

Remarque 2.10

Attention, la définition précédente ne dit pas qu'un langage régulier est stable par concaténation, union ♣ et passage à l'étoile, elle dit que l'ensemble des langages réguliers est stable par ….

Exercice de cours 2.11

Donner un langage régulier qui n'est pas stable par concaténation.

2.2 Expressions régulières

Définition 2.12

*L'ensemble des **expressions régulières** sur Σ, noté ici $\mathscr{E}_{reg}(\Sigma)$, est défini par induction à partir des règles suivantes :*

- *le symbole $\emptyset$ est une expression régulière ;*
- *le symbole ε est une expression régulière ;*
- *si $a \in \Sigma$ alors a est une expression régulière ;*
- *si e_1 et e_2 sont deux expressions régulières alors l'expression $e_1 \cdot e_2$ est une expression régulière ;*
- *si e_1 et e_2 sont deux expressions régulières alors l'expression $e_1 | e_2$ est une expression régulière.*
- *si e est une expression régulière alors l'expression $e^\star$ est une expression régulière.*

Exercice de cours 2.13

Donner l'arbre de syntaxe de l'expression régulière $a \cdot (b | (a \cdot \emptyset)^\star)$.

Remarque 2.14

Les expressions régulières sont des objets syntaxiques, leur écriture les définit complètement. Ainsi l'expression $a|b$ n'est pas la même que l'expression $b|a$, l'expression $(a.a).b$ pas la même que $a.(a.b)$ …. (Mais la sémantique définie ci-dessous apporte une notion d'équivalence qui règle tout ça).

Définition 2.15

Le langage associé à une expression régulière sur Σ est donné par la fonction $\mathscr{L}$ définie inductivement comme suit.

$$\mathscr{L} = \begin{pmatrix} \mathscr{E}_{reg}(\Sigma) & \to & \mathscr{P}(\Sigma^\star) \\ \emptyset & \mapsto & \emptyset \\ \varepsilon & \mapsto & \{\varepsilon\} \\ a \in \Sigma & \mapsto & \{a\} \\ E^* & \mapsto & \mathscr{L}(E)^* \\ E_1 | E_2 & \mapsto & \mathscr{L}(E_1) \cup \mathscr{L}(E_2) \\ E_1 \cdot E_2 & \mapsto & \mathscr{L}(E_1) \cdot \mathscr{L}(E_2) \end{pmatrix}$$

Exercice de cours 2.16

Donner le langage de l'expression régulière $(b{\cdot}b{\cdot}b)^\star(a|c|\varepsilon)$.

Proposition 2.17

L'ensemble des langages réguliers est exactement l'image de $\mathcal{L}$.

Remarque 2.18

La structure inductive de $\mathscr{E}_{reg}(\Sigma)$ permet de définir de manière inductive des fonctions sur $\mathscr{E}_{reg}(\Sigma)$. Ainsi, on peut définir comme suit la fonction qui associe à une expression régulière l'ensemble des lettres qui y apparaissent.

$$\mathsf{lettres} = \begin{pmatrix} \mathscr{E}_{reg}(\Sigma) & \to & \mathscr{P}(\Sigma) \\ \emptyset & \mapsto & \emptyset \\ \varepsilon & \mapsto & \emptyset \\ a \in \Sigma & \mapsto & \{a\} \\ E^* & \mapsto & \mathsf{lettres}(E) \\ E_1|E_2 & \mapsto & \mathsf{lettres}(E_1) \cup \mathsf{lettres}(E_2) \\ E_1 \cdot E_2 & \mapsto & \mathsf{lettres}(E_1) \cup \mathsf{lettres}(E_2) \end{pmatrix}$$

Exercice de cours 2.19

Donner $\mathsf{lettres}((b{\cdot}b{\cdot}b)^\star(a|c|\varepsilon))$.

3 Automates finis (sans ε-transition)

3.1 Définitions

Définition 3.1

*Un **automate fini (sans ε-transition)** est un quintuplet $\mathcal{A} = (\Sigma, Q, I, F, \delta)$ où :*
- *Σ est un alphabet ;*
- *Q est un ensemble fini ;*
- *$I \subseteq Q$;*
- *$F \subseteq Q$;*
- *$\delta \subseteq Q \times \Sigma \times Q$.*

*Les éléments de Q (resp. I, F) sont appelés **états** (resp. **état initiaux**, **état finaux**).*
*De plus si $t = (q, a, q') \in \delta$, on dit que t est une **transition** de l'état q à l'état q' d'**étiquette** a, ce que l'on note parfois $q \xrightarrow{a} q'$.*

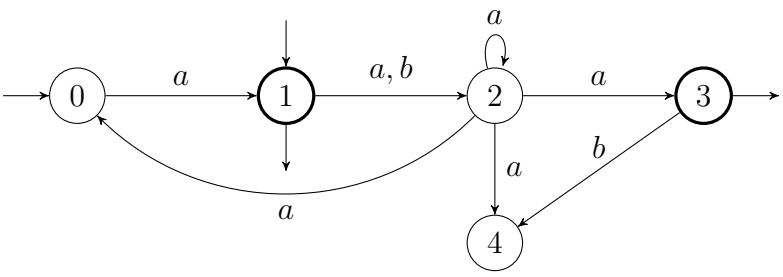


FIGURE 1 – Automate exemple

Remarque 3.2

La représentation graphique utilisée pour représenter un automate dans la figure 1 est une réprésentation assez standard, que nous utiliserons dans tout ce chapitre.
- Les états sont représentés dans des cercles, ici ce sont 0, 1, 2, 3 et 4.
- Les états initiaux sont marqués par des flèches entrantes, ici ce sont 0 et 1.
- Les états finaux sont marqués par des flèches sortantes et du gras, ici ce sont 1 et 3♣.
- Les transitions sont représentées par des flèches entre états. Ici par exemple l'ensemble des transitions est le suivant : $\{(0,a,1),(1,a,2),(1,b,2),(2,a,0),(2,a,2),(2,a,3),(2,a,4),(3,b,4)\}$.
- L'alphabet sous-jacent n'est pas représenté. Par défaut on peut considérer que c'est l'ensemble des lettres apparaissant sur des transitions, mais ce n'est pas toujours le cas♡.

Remarque 3.3

Les automates au programme étant tous des automates finis, on omettra souvent de le préciser. Ainsi tous les "automates" de ce chapitre ont un nombre fini d'états.

Définition 3.4

Soit $\mathcal{A} = (\Sigma, Q, I, F, \delta)$ un automate fini (sans ε-transition).
Soient $(q_i)_{i\in[\![0,n]\!]} \in Q^{n+1}$ et $u \in \Sigma^n$ pour $n \in \mathbb{N}$.
*On dit que $(q_i)_{i\in[\![0,n]\!]}$ est une **exécution** de $\mathcal{A}$ d'**étiquette** u si $q_0 \in I$ et $\forall i \in [\![1,n]\!], (q_{i-1}, u_i, q_i) \in \delta$.*
Ce que l'on note parfois $q_0 \xrightarrow{u_1} q_1 \xrightarrow{u_2} q_2 \dots \xrightarrow{u_n} q_n$.
*Si de plus $q_n \in F$, on parle d'**exécution acceptante** et on dit que le mot u est **reconnu** par $\mathcal{A}$.*

Remarque 3.5

On dira d'une suite d'états $q_0 \xrightarrow{u_1} q_1 \xrightarrow{u_2} q_2 \dots \xrightarrow{u_n} q_n$ (sans les contraintes $q_0 \in I$ et $q_n \in F$) que c'est une **suite de transitions**. On dira par ailleurs qu'elle **part** de q_0 et qu'elle **mène à** q_n.

Exemple 3.6

Dans l'automate exemple (Figure 1) $(0,1,2,0,1)$ est une exécution acceptante étiquetée (entre autres) par $abaa$, on la note $0 \xrightarrow{a} 1 \xrightarrow{b} 2 \xrightarrow{a} 0 \xrightarrow{a} 1$.

Définition 3.7

*Le **langage reconnu** par un automate $\mathcal{A}$, noté $\mathcal{L}(\mathcal{A})$, est l'ensemble des mots reconnus par $\mathcal{A}$.*
*On dit d'un langage sur Σ qu'il est **reconnaissable** s'il existe un automate sur Σ qui le reconnaît.*
*Deux automates sont dits **équivalents** s'ils reconnaissent le même langage.*

Exercice de cours 3.8

Donner deux automates différents qui sont équivalents, donner en français le langage (commun) qu'ils reconnaissent.

Définition 3.9

Soit $\mathcal{A} = (\Sigma, Q, I, F, \delta)$ un automate fini (sans ε-transition).
*On dit que $\mathcal{A}$ est **complet** dès lors que $\mathrm{card}(I) \geqslant 1$ et $\forall q \in Q,\ \forall a \in \Sigma,\ \mathrm{card}\{q' \in Q | (q,a,q') \in \delta\} \geqslant 1$.*
*On dit que $\mathcal{A}$ est **déterministe** dès lors que $\mathrm{card}(I) \leqslant 1$ et $\forall q \in Q,\ \forall a \in \Sigma,\ \mathrm{card}\{q' \in Q | (q,a,q') \in \delta\} \leqslant 1$.*

Exercice de cours 3.10

L'automate exemple (Figure 1) est-il déterministe ? Est-il complet ?

Proposition 3.11

Dans un automate complet, tout mot de $\Sigma^\star$ est l'étiquette d'au moins une exécution.
Dans un automate déterministe, tout mot de $\Sigma^\star$ est l'étiquette d'au plus une exécution.

Exercice de cours 3.12

Démontrer la proposition précédente. Donner un contre-exemple pour la réciproque du premier (resp. du deuxième) point.

3.2 Transformations en automate équivalent

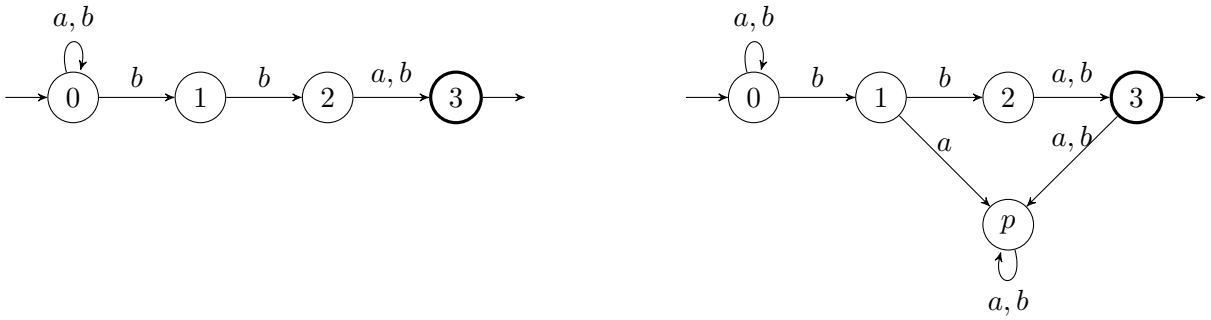
3.2.1 Compléter

Proposition 3.13

Soit $\mathcal{A} = (\Sigma, Q, I, F, \delta)$ un automate fini tel que $I \neq \emptyset$. Soit alors
- *$Q' = Q \sqcup \{p\}$ avec $p \notin Q$,*
- *$\delta' = \delta \sqcup \left\{(q,a,p) \;\middle|\; (q,a) \in Q\times\Sigma \text{ tel que } \{q' \in Q | (q,a,q') \in \delta\} = \emptyset\right\} \sqcup \left\{(p,a,p) \;\middle|\; a \in \Sigma\right\}$,*

l'automate $\mathcal{A}' = (\Sigma, Q', I, F, \delta')$ est complet et équivalent à $\mathcal{A}$.
*On l'appelle parfois **le complété** de $\mathcal{A}$.*

Exemple 3.14
L'automate ci-dessous à gauche a pour complété l'automate ci-dessous à droite.



Démonstration : On vérifie que $\mathcal{A}'$ est complet par construction ($Q' \times \Sigma = (Q\times\Sigma) \sqcup (\{p\}\times\Sigma)$).

Montrons maintenant qu'il est équivalent à $\mathcal{A}$ par double inclusion des langages reconnus.

Puisque $Q \subseteq Q'$ et $\delta \subseteq \delta'$, une exécution de $\mathcal{A}$ est aussi une exécution dans $\mathcal{A}'$, et comme l'ensemble d'états finaux est le même, une exécution acceptante de $\mathcal{A}$ est aussi acceptante dans $\mathcal{A}'$, d'où $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{A}')$.

Réciproquement, on considère $u \in \mathcal{L}(\mathcal{A}')$. Par définition il existe $(q_i)_{i\in[\![0,n]\!]}$ avec $n = |u|$ une exécution acceptante dans $\mathcal{A}'$ d'étiquette u. Nécessairement $q_n \in F \subseteq Q$. Comme $\delta' \cap (Q\times\Sigma\times Q) \subseteq \delta$, pour montrer que cette exécution est aussi une exécution acceptante dans $\mathcal{A}'$, il suffit de s'assurer que $\forall i \in [\![0, n-1]\!]$, $q_i \in Q$, soit $\forall i \in [\![0, n-1]\!]$, $q_i \neq p$. Or toutes les transitions issues de l'état p vont vers l'état p (on parle d'ailleurs d'état puits), donc s'il existait $i \in [\![0, n-1]\!]$ tel que $q_i = p$, on montrerait de proche en proche que $q_n = p \notin Q$. Absurde.

Donc $q_0 \xrightarrow{u_1} q_1 \xrightarrow{u_2} q_2 \dots \xrightarrow{u_n} q_n$ est bien une exécution acceptante dans $\mathcal{A}$, donc $u \in \mathcal{L}(\mathcal{A})$.
D'où $\mathcal{L}(\mathcal{A}') \subseteq \mathcal{L}(\mathcal{A})$, et finalement $\mathcal{L}(\mathcal{A}') = \mathcal{L}(\mathcal{A})$. □

3.2.2 Déterminiser

Proposition 3.15

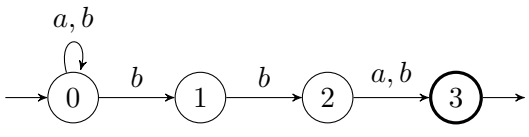
Soit $\mathcal{A} = (\Sigma, Q, I, F, \delta)$ un automate fini. En posant :
- $Q' = \mathcal{P}(Q)$,
- $\delta' = \left\{(X, a, \{q' \in Q | \exists q \in X, (q, a, q') \in \delta\}) \;\middle|\; (X, a) \in Q' \times \Sigma\right\}$,
- $I' = \{I\}$,
- $F' = \{X \in Q' | X \cap F \neq \emptyset\}$,

l'automate $\mathcal{A}' = (\Sigma, Q', I', F', \delta')$ est déterministe et équivalent à $\mathcal{A}$.
*On appelle un tel automate l'**automate des parties** de $\mathcal{A}$ ou le **déterminisé** de $\mathcal{A}$.*

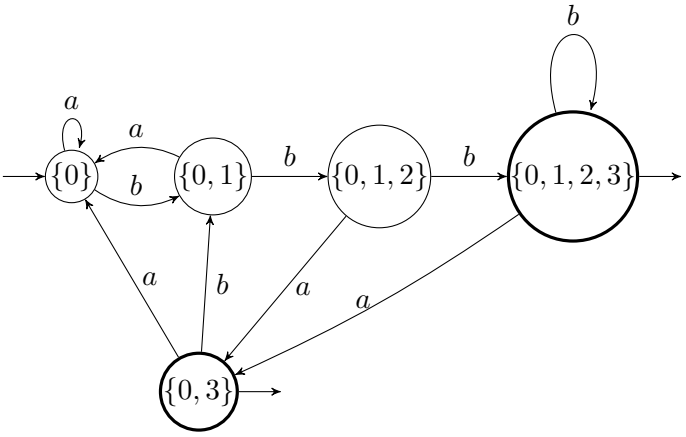
Remarque 3.16

Pour simplifier la définition ci-dessus, l'automate est construit sur les $\mathcal{P}(Q)$ états, cependant il est fréquent qu'une partie non négligeable de ces états soient en fait non accessibles ♣depuis l'état initial. Ne pas construire des états qui seraient inaccessibles ne change pas le langage de l'automate, aussi lors de l'application de l'algorithme de déterminisation, on se contente de construire uniquement les états accessibles. On peut obtenir un tel résultat en construisant les états depuis l'état initial et en visitant de proche en proche les nouveaux états rencontrés par lecture de lettres. On présente une telle construction dans un tableau où chaque ligne correspond à un nouvel état (du déterminisé), *Cf.* exemple 3.2.2

Exemple 3.17
L'automate ci-dessous à gauche a pour déterminisé l'automate ci-dessous à droite. On a construit le déterminisé à l'aide de la table ci-dessous à gauche.



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



Démonstration : On vérifie que $\mathcal{A}'$ est déterministe par construction. En effet $\mathcal{A}$ a bien un seul état initial, à savoir $I \in \mathcal{P}(Q) = Q'$, et pour chaque état de $X \in Q'$ et chaque lettre $a \in \Sigma$ il existe un seul état $Y \in Q'$ tel que $(X, a, Y) \in \delta'$, à savoir l'ensemble des états accessibles depuis un état de X en lisant a.

Montrons maintenant qu'il est équivalent à $\mathcal{A}$ par double inclusion des langages reconnus.

Soit $u \in \mathcal{L}(\mathcal{A})$. On construit de manière récurrente la suite finie des ensembles d'états accessibles dans $\mathcal{A}$ en lisant les lettres de u depuis n'importe quel état initial. Ces ensembles étant des parties de Q, on obtient ainsi une suite d'états de $\mathcal{A}'$ dont on va montrer qu'elle est une exécution acceptante.

On pose donc $Q_0 = I$, ainsi $Q_0 \in I'$, et pour $i \in [\![1, n]\!]$, $Q_i = \{q' \in Q \mid \exists q \in Q_{i-1}, (q, u_i, q') \in \delta\}$, ainsi par construction de δ' on a $\forall i \in [\![1, n]\!]$, $(Q_{i-1}, u_i, Q_i) \in \delta'$. On sait alors que $Q_0 \xrightarrow{u_1} Q_1 \xrightarrow{u_2} Q_2 \dots \xrightarrow{u_n} Q_n$ est une exécution de $\mathcal{A}'$, reste à montrer qu'elle aboutit à un état final, *i.e.* $Q_n \in F'$.

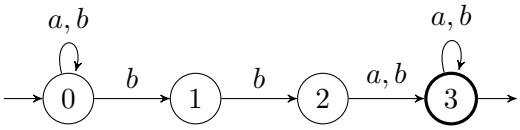
♣. cette notion est formalisée plus loin

Puisque $u \in \mathcal{L}(\mathcal{A})$, il existe par définition $(q_i)_{i\in[\![0,n]\!]} \in Q^{n+1}$ avec $n = |u|$ une exécution acceptante de $\mathcal{A}$ d'étiquette u. On sait que $q_0 \in I$, donc $q_0 \in Q_0$, et puisque $\forall i \in [\![1,n]\!]$, $(q_{i-1}, u_i, q_i) \in \delta$ on peut montrer par récurrence immédiate que $\forall i \in [\![0,n]\!], q_i \in Q_i$. En particulier $q_n \in Q_n$, et comme $q_n \in F$, $Q_n \cap F \neq \emptyset$, donc $Q_n \in F'$. Ainsi $Q_0 \xrightarrow{u_1} Q_1 \xrightarrow{u_2} Q_2 \dots \xrightarrow{u_n} Q_n$ est une exécution acceptante de $\mathcal{A}'$, donc $u \in \mathcal{L}(\mathcal{A}')$.
D'où $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{A}')$.

Réciproquement, on considère $u \in \mathcal{L}(\mathcal{A}')$. Par définition il existe $(Q_i)_{i\in[\![0,n]\!]}$ avec $n = |u|$ une exécution acceptante dans $\mathcal{A}'$ d'étiquette u. L'idée est maintenant d'extraire de chaque état $Q_i \in Q'$ un état $q_i \in Q$ de manière à construire une exécution acceptante dans $\mathcal{A}$. Par définition, Q_n est un état final, soit $Q_n \in F'$, ou encore $Q_n \cap F \neq \emptyset$. Il existe donc au moins un état $q_n \in Q_n \cap F$. De plus la construction de δ' assure le fait suivant : si $(X, a, Y) \in \delta'$ et $y \in Y$, alors il existe $x \in X$ tel que $(x, a, y) \in \delta$. Donc pour tout $i \in [\![1,n]\!]$, si $q_i \in Q_i$, il existe $q_{i-1} \in Q_{i-1}$ tel que $(q_{i-1}, u_i, q_i) \in \delta$. En itérant cet argument à partir de q_n qui appartient à Q_n, on construit $(q_i)_{i\in[\![0,n]\!]}$ telle que $\forall i \in [\![1,n]\!]$, $(q_{i-1}, u_i, q_i) \in \delta$ et $\forall i \in [\![0,n]\!]$, $q_i \in Q_i$. En particulier $q_0 \in Q_0$ et comme $Q_0 = I$, q_0 est un état initial de $\mathcal{A}$. On en déduit que $(q_i)_{i\in[\![0,n]\!]}$ est un exécution de $\mathcal{A}$ étiquetée par u. Enfin, comme on a pris soin de choisir q_n dans F, cette exécution est acceptante, donc $u \in \mathcal{L}(\mathcal{A})$.
D'où $\mathcal{L}(\mathcal{A}') \subseteq \mathcal{L}(\mathcal{A})$, et finalement $\mathcal{L}(\mathcal{A}') = \mathcal{L}(\mathcal{A})$. □

Exercice de cours 3.18

Déterminiser l'automate ci-dessous.



Remarque 3.19

Cette procédure peut être coûteuse, car a priori on construit 2^n états dans $\mathcal{A}'$ pour n états dans A. Les exemples nous ont montré qu'on peut parfois se passer des 2^n états, toutefois on montrera en TD que pour tout $n \in \mathbb{N}$, il existe des automates à n états qu'on ne peut déterminiser qu'avec 2^n états, c'est-à-dire qu'aucun automate déterministe ayant moins d'états ne reconnaît le même langage.

3.2.3 Émonder

Définition 3.20

Soit $\mathcal{A} = (\Sigma, Q, I, F, \delta)$ un automate fini (sans ε-transition). Soit $q \in Q$.

*On dit d'un état q qu'il est **accessible** dès lors qu'il existe une suite de transitions partant d'un état initial et menant à q.*

*On dit d'un état q qu'il est **co-accessible** dès lors qu'il existe une suite de transitions partant de q et menant à un état final.*

Définition 3.21

*Soit $\mathcal{A} = (\Sigma, Q, I, F, \delta)$ un automate fini. On obtient l'automate **émondé** de $\mathcal{A}$ en le réduisant à ses états à la fois accessibles et co-accessibles, autrement dit l'automate émondé est défini comme $\hat{\mathcal{A}} = (\Sigma, \hat{Q}, \hat{I}, \hat{F}, \hat{\delta})$ où :*

- *$\hat{Q}$ est l'ensemble des états à la fois accessibles et co-accessibles de $\mathcal{A}$;*
- *$\hat{\delta} = \delta \cap \left(\hat{Q} \times \Sigma \times \hat{Q}\right)$;*
- *$\hat{I} = I \cap \hat{Q}$;*
- *$\hat{F} = F \cap \hat{Q}$.*

Proposition 3.22

*Un automate est équivalent à son **émondé**.*

Démonstration : On le montre par double inclusion des langages reconnus.

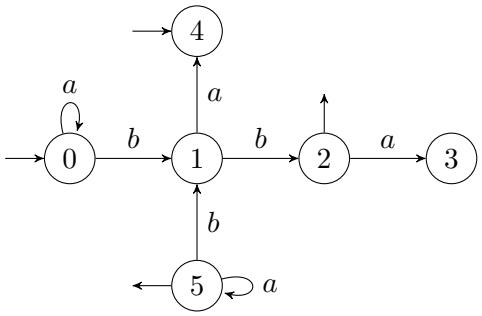
Puisque $\hat{Q} \subseteq Q$, $\hat{\delta} \subseteq \delta$, $\hat{I} \subseteq I$ et $\hat{F} \subseteq F$, une exécution acceptante de $\hat{\mathcal{A}}$ est aussi une exécution acceptante de $\mathcal{A}$, donc $\mathcal{L}(\hat{\mathcal{A}}) \subseteq \mathcal{L}(\mathcal{A})$.

Réciproquement, si $q_0 \xrightarrow{z_1} q_1 \xrightarrow{z_2} q_2 \dots \xrightarrow{z_n} q_n$ est une exécution acceptante de $\mathcal{A}$, alors $\forall i \in [\![0,n]\!],\ q_i$ est accessible ($(q_k)_{k\in[\![0,i]\!]}$ en est la preuve) et co-accessible ($(q_k)_{k\in[\![i,n]\!]}$ en est la preuve), donc $\forall i \in [\![0,n]\!],\ q_i \in \hat{Q}$.

Comme $\delta \cap (\hat{Q}\times\Sigma\times\hat{Q}) = \hat{\delta}$, on en déduit que $\forall i \in [\![1,n]\!],\ (q_{i-1}, z_i, q_i) \in \delta$. Ainsi cette exécution est aussi une exécution acceptante dans $\hat{\mathcal{A}}$, et pour la même étiquette, donc $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\hat{\mathcal{A}})$. □

Exercice de cours 3.23

Donner les états accessibles, co-accessibles de l'automate ci-dessous. Donner l'automate émondé associé.



Proposition 3.24

Le langage reconnu par un automate est non vide si et seulement s'il existe un état initial co-accessible ou de manière équivalente s'il existe un état final accessible.

Démonstration : Soit $\mathcal{A} = (\Sigma, Q, I, F, \delta)$ un automate fini. On a les équivalences suivantes.

$$\begin{aligned}
\mathcal{L}(\mathcal{A}) \neq \emptyset &\Leftrightarrow \exists a \in \mathcal{L}(\mathcal{A}) \\
&\Leftrightarrow \exists n \in \mathbb{N}, \exists a \in \Sigma^n, \exists (q_i)_{i\in[\![0,n]\!]} \in Q^{n+1},\ q_0 \in I, q_n \in F, \forall i \in [\![1,n]\!], (q_{i-1}, a_i, q_i) \in \delta \\
&\Leftrightarrow \exists n \in \mathbb{N}, \exists (q_i)_{i\in[\![0,n]\!]} \in Q^{n+1},\ q_0 \in I, q_n \in F, \forall i \in [\![1,n]\!], \exists a_i \in \Sigma (q_{i-1}, a_i, q_i) \in \delta \\
&\Leftrightarrow \underbrace{\exists q_0 \in I}_{\text{il existe un état initial}}, \underbrace{\exists n \in \mathbb{N}, \exists (q_i)_{i\in[\![1,n]\!]} \in Q^n,\ q_n \in F, \forall i \in [\![1,n]\!], \exists a_i \in \Sigma (q_{i-1}, a_i, q_i) \in \delta}_{\text{qui est co-accessible}}
\end{aligned}$$

□

Corollaire 3.25

Il existe un algorithme permettant de tester la vacuité du langage d'un automate.

Démonstration : La propriété précédente assure que l'on peut tester la vacuité d'un langage en testant si les états initiaux sont co-accessibles, ou en testant si les états finaux sont accessibles. Dans ces tests les étiquettes des transitions ne sont pas importantes, seule l'existence d'une transition d'un état à un autre l'est. On est alors ramené à des questions d'accessibilité dans le graphe orienté G associé à $\mathcal{A}$ qu'on définit comme suit.

$$G = (S, A) \text{ avec } S = Q \text{ et } A = \left\{ (q, q') \in Q^2 \;\middle|\; \exists a \in \Sigma, (q, a, q') \in \delta \right\}$$

Plus précisément, on teste qu'un état initial est co-accessible par un parcours depuis le sommet correspondant, si un état final est atteint il est co-accessible, sinon non. La complexité de cet algorithme naïf est donc $O(n_i(n+m))$ où $n = \text{card}(S) = \text{card}(Q), m = \text{card}(A) \leqslant \text{card}(\delta)$ et $n_i = \text{card}(I)$. □

Exercice de cours 3.26

Donner le graphe construit dans la preuve précédente pour l'automate de l'exercice de cours 3.23.

Exercice de cours 3.27

Proposer un algorithme permettant de tester la vacuité du langage d'un automate ayant une complexité en $\mathcal{O}(n+m)$ en reprenant les notations de la preuve précédente.

4 Automates avec ε-transitions

4.1 Définitions

Définition 4.1

*Un **automate fini avec ε-transitions** est un quintuplet $\mathcal{A} = (\Sigma, Q, I, F, \delta)$ où :*
- *Σ est un alphabet (ne contenant pas le symbole ε)*
- *Q est un ensemble fini ;*
- *$I \subseteq Q$;*
- *$F \subseteq Q$;*
- *$\delta \subseteq Q \times (\Sigma \sqcup \{\varepsilon\}) \times Q$.*

*Les éléments de Q (resp. I, F) sont appelés **états** (resp. **état initiaux**, **état finaux**).*
*De plus si $t = (q, z, q') \in \delta$ avec $z \in \Sigma$, on dit que t est une **transition** de l'état q à l'état q' d'**étiquette** z, tandis que si $z = \varepsilon$ on dit que t est une **ε-transition** de q vers q'.*

Définition 4.2

Soit $\mathcal{A} = (\Sigma, Q, I, F, \delta)$ un automate fini avec ε-transitions. Soient $(q_i)_{i\in[\![0,n]\!]} \in Q^{n+1}$ et $z \in (\Sigma \sqcup \{\varepsilon\})^n$ pour $n \in \mathbb{N}$.

*Si $q_0 \in I$ et $\forall i \in [\![1, n]\!], (q_{i-1}, z_i, q_i) \in \delta$, on dit que $(q_i)_{i\in[\![0,n]\!]}$ est une **exécution** de $\mathcal{A}$, et si de plus $q_n \in F$, on parle d'**exécution acceptante**.*

*L'**étiquette** de cette exécution est alors le mot u obtenu à partir de z en omettant les occurrences de ε, et on dit que u est **reconnu** par $\mathcal{A}$. On note parfois $q_0 \xrightarrow{z_1} q_1 \xrightarrow{z_2} q_2 \dots \xrightarrow{z_n} q_n$.*

Remarque 4.3

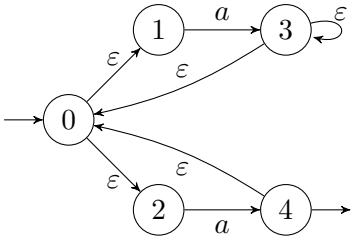
Si $\mathcal{A} = (\Sigma, Q, I, F, \delta)$ un automate fini avec ε-transitions qui reconnaît un mot $u \in \Sigma^\star$ de taille n, on peut bien dire qu'il existe une exécution acceptante de $\mathcal{A}$ d'étiquette u, mais on ne peut pas dire qu'il existe $(q_i)_{i\in[\![0,n]\!]}$ telle que $q_0 \xrightarrow{u_1} q_1 \xrightarrow{u_2} q_2 \dots \xrightarrow{u_n} q_n$. En effet la longueur de l'exécution n'est pas connue a priori, car en plus des transitions étiquetées par des lettres de u, il peut y avoir des ε-transitions.

Définition 4.4

On étend naturellement les définitions de langage reconnu, langage reconnaissable, et d'équivalence d'automates aux automates avec ε-transitions.

Exercice de cours 4.5

Donner, au moyen d'une expression régulière le langage de l'automate ci-dessous. Donner un automate sans ε-transitions reconnaissant le même langage.



Remarque 4.6

On notera parfois π (comme projection) la fonction qui à un mot de $(\Sigma \sqcup \{\varepsilon\})^\star$ associe le mot de $\Sigma^\star$ obtenu en supprimant les caractères ε du mot. On remarque que π est alors un morphisme de monoïdes, *i.e.* $\forall (z,z') \in (\Sigma \sqcup \{\varepsilon\})^\star \times (\Sigma \sqcup \{\varepsilon\})^\star, \pi(z \cdot z') = \pi(z) \cdot \pi(z')$. De plus $\forall z \in (\Sigma \sqcup \{\varepsilon\})^\star$, $\pi(z)$ est un sous-mot de z.

Remarque 4.7

On ne dira pas d'un automate avec ε-transitions qu'il est déterministe, même si $\forall a \in \Sigma, \forall (q,q',q'') \in Q^3, (q,a,q') \in \delta, (q,a,q'') \in \delta \Rightarrow q' = q''$. En effet, même sous cette condition il peut y avoir plusieurs exécutions étiquetées par un même mot du fait des ε-transitions.
En fait le terme "automate fini non déterministe" englobe les automates à ε-transition dans le programme.

4.2 Élimination des ε-transitions

Proposition 4.8

Tout automate avec ε-transitions est équivalent à un automate sans ε-transition.

Démonstration : On donne une preuve constructive de cette propriété.

Soit $\mathcal{A}^0 = (\Sigma, Q, I, F^0, \delta^0)$ un automate avec ε-transitions. On note $n = \mathrm{card}(Q)$ et on identifie Q et $[\![1,n]\!]$. On va construire une suite d'automates équivalents les uns aux autres, en débarrassant à chaque étape un nouvel état de toutes les ε-transitions qui y aboutissaient, et en ajoutant des transitions pour compenser et garantir l'équivalence. Dans le cas particulier où $n = 0$, l'automate $\mathcal{A}^0$ n'a aucun état, et par conséquent aucune transition, et a fortiori aucune ε-transition, donc il n'y a rien à démontrer.

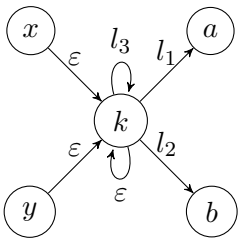
Supposons $n > 0$. Soit $k \in [\![1,n]\!]$. On cherche à construire $\mathcal{A}^k$ équivalent à $\mathcal{A}^{k-1}$ en débarrassant l'état k des ε-transitions qui y aboutissent. Notons A l'ensemble des ε-transitions (de $\mathcal{A}^{k-1}$) aboutissant sur k, hors boucles, B l'ensemble contenant l'ε-transition bouclant sur k si elle existe, et C l'ensemble de toutes les transitions allant de k vers un autre état. On a donc les égalités suivantes.

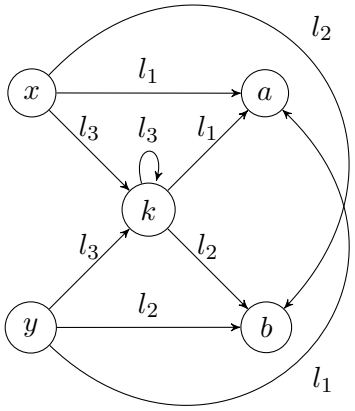
$$A = \left\{(q,\varepsilon,k) \,|\, q \in Q \setminus \{k\},\ (q,\varepsilon,k) \in \delta^{k-1}\right\}$$
$$B = \{(k,\varepsilon,k)\} \cap \delta^{k-1}$$
$$C = \{(k,z,q') \,|\, q' \in Q \setminus \{k\}, z \in \Sigma \sqcup \{\varepsilon\}, (k,z,q') \in \delta^{k-1}\}$$

La suppression de chaque ε-transition $(q,\varepsilon,k) \in A$ doit être compensée par l'ajout d'une transition (q,z,q') pour toute transition $(k,z,q') \in C$, ainsi on définit le nouvel ensemble de transitions δ^k comme suit.

$$\delta^k = \left(\delta^{k-1} \setminus (A \sqcup B)\right) \cup \{(q,z,q') \,|\, (q,\varepsilon,k) \in A, (k,z,q') \in C\}$$

On résume cette transformation ci-dessous. L'état k de l'automate ci-dessous à gauche est débarrassé de ses ε-transitions en le transformant en l'automate ci-dessous à droite. Dans ce schéma A est l'ensemble $\{(x,\varepsilon,k),(y,\varepsilon,k)\}$, $B=\{(k,\varepsilon,k)\}$ et $C=\{(k,l_3,k),(k,l_1,a),(k,l_2,b)\}$.





On note que pour $(q,\varepsilon,k)\in A$ et $(k,z,q')\in C$, la transition (q,z,q') ajoutée à l'ensemble des transitions peut être une ε-transition, mais seulement à condition qu'il existe dans A^{k-1} une ε-transition aboutissant en q' (à savoir (k,z,q')). De plus on note que k ne reçoit aucune ε-transition dans δ^k (c'était le but !). Ainsi l'ensemble des états recevant des ε-transitions de δ^k est celui des états recevant des ε-transitions de δ^{k-1} privé de k.

Par ailleurs, si k était un état final, $k\in F^{k-1}$, en supprimant une ε-transition $(q,\varepsilon,k)\in A$, on a peut-être empêché que la lecture de certains mots n'aboutisse dans un état final, on rend donc q final aussi. On pose donc

$$F^k=\begin{cases} F^{k-1}\cup\{q\in Q\mid(q,\varepsilon,k)\in A\} & \text{si } k\in F\\ F^{k-1} & \text{sinon}\end{cases}$$

On pose finalement $\mathcal{A}^k=(\Sigma,Q,I,F^k,\delta^k)$.

Montrons maintenant que $\mathcal{A}^{k-1}$ et $\mathcal{A}^k$ sont équivalents par double inclusion des langages reconnus.

Soit $u\in\mathcal{L}(\mathcal{A}^{k-1})$. Par définition, il existe une exécution acceptante de $\mathcal{A}^{k-1}$ d'étiquette u. On commence par remarquer que si cette exécution présente des ε-transitions qui bouclent sur un état, ces ε-transitions peuvent être supprimées sans modifier l'étiquette (car ce sont des ε-transitions), ni la validité de l'exécution (car ce sont des boucles). Ainsi on peut supposer, sans perte de généralité, que u est l'étiquette d'une exécution de $\mathcal{A}^{k-1}$ qui ne contient aucune ε-transition qui soit une boucle ($\bigstar$).

- Si cette exécution n'emprunte aucune ε-transition de A, c'est aussi une exécution de $\mathcal{A}^k$ (car on a $\delta^{k-1}\cap\overline{A}\cap\overline{B}\subseteq\delta^k$), et comme $F^{k-1}\subseteq F^k$, elle est aussi acceptante dans $\mathcal{A}^k$, donc $u\in\mathcal{L}(\mathcal{A}^k)$.
- Si elle emprunte une ε-transition $(q,\varepsilon,k)\in A$ suivie d'une transition (k,z,q'), on a nécessairement $q'\neq q$ (d'après ($\bigstar$)), donc $(k,z,q')\in C$ et $(q,z,q')\in\delta^k$. Alors remplacer les transitions (q,ε,k) et (k,z,q') par la transition (q,z,q') ne change pas l'étiquette. En remplaçant ainsi toutes les ε-transition de A de cette exécution, sauf celle éventuellement placée en dernier, on obtient une exécution d'étiquette u qui a au plus une ε-transition de A. Si elle en a aucune on est ramené au cas précédent, si elle en a une, la dernière nécessairement, on est ramené au cas suivant.
- Si elle emprunte une seule ε-transition de A, placée en dernier dans l'exécution, cette transition t est de la forme (q,ε,k) avec $q\in Q$. En supprimant cette ε-transition t, on obtient alors une exécution d'étiquette u sans ε-transition de A, donc valide dans $\mathcal{A}^k$. Elle aboutit en q, or comme $k\in F^{k-1}$ en tant que dernier état d'une exécution acceptante, on a $q\in F^k$ par construction de F^k. Ainsi u est l'étiquette d'une exécution acceptante de $\mathcal{A}^k$, soit $u\in\mathcal{L}(\mathcal{A}^k)$.

Dans tous les cas $u\in\mathcal{L}(\mathcal{A}^k)$, d'où $\mathcal{L}(\mathcal{A}^{k-1})\subseteq\mathcal{L}(\mathcal{A}^k)$.

Réciproquement on considère $u \in \mathcal{L}(\mathcal{A}^k)$. Il existe alors une exécution acceptante de $\mathcal{A}^k$ d'étiquette u. Si elle emprunte une transition de $\delta^k \setminus \delta^{k-1}$, celle-ci s'écrit (q, z, q') avec $(q, \varepsilon, k) \in A$ et $(k, z, q') \in C$, en particulier avec $(q, \varepsilon, k) \in \delta^{k-1}$ et $(k, z, q') \in \delta^{k-1}$. On peut donc remplacer toute transition qui n'est pas dans δ^{k-1} par deux transitions de δ^{k-1}, sans changer l'étiquette de l'exécution. Ainsi u est l'étiquette d'une exécution de $\mathcal{A}^{k-1}$.

- Si elle aboutit en $q \in F^{k-1}$, elle est acceptante, donc $u \in \mathcal{L}(\mathcal{A}^{k-1})$.
- Si elle aboutit en $q \in F^k \setminus F^{k-1}$, par construction de F^k on a nécessairement $(q, \varepsilon, k) \in \delta^{k-1}$ et $k \in F^{k-1}$. Ainsi en ajoutant (q, ε, k) à la fin de l'exécution, on obtient une exécution de $\mathcal{A}^{k-1}$, qui aboutit en $k \in F^{k-1}$, et elle est toujours d'étiquette u puisque la transition ajoutée est une ε-transition. Ainsi u est l'étiquette d'une exécution acceptante de $\mathcal{A}^{k-1}$, soit $u \in \mathcal{L}(\mathcal{A}^{k-1})$.

Dans tous les cas $u \in \mathcal{L}(\mathcal{A}^{k-1})$, d'où $\mathcal{L}(\mathcal{A}^k) \subseteq \mathcal{L}(\mathcal{A}^{k-1})$ □

Exercice de cours 4.9

Appliquer l'algorithme ci-dessus à l'automate de l'exercice de cours 4.5.

Exercice de cours 4.10

On a choisi ici de présenter un algorithme permettant de débarrasser un par un les états des ε-transitions entrantes. Il est en fait aussi possible de définir un algorithme permettant de débarrasser un par un les états des ε-transitions sortantes. Adapter les définitions de A, B, C, δ^l et F^k dans la preuve précédente pour définir cet algorithme. Résumer ces définitions au moyen d'un schéma similaire à celui présenté dans la preuve.

Remarque 4.11

Cette propriété assure que la première définition de langage reconnaissable, comme langage reconnu par automate sans ε-transition, est équivalente à celle qu'on a donné dans cette partie, à savoir un langage reconnu par un automate avec ε-transitions. Autrement dit les ε-transitions ne nous font pas gagner en expressivité, on représente les même langages avec ou sans elles, seulement avec on a parfois des automates plus simples à construire, plus compacts, plus lisibles.

4.3 Opérations sur les automates

Dans cette section on cherche comment, à partir d'automates reconnaissant des langages, construire un automate qui reconnaît le résultat d'une opération sur ces langages.

4.3.1 Concaténation

Proposition 4.12

Soient $\mathcal{A} = (\Sigma, Q, I, F, \delta)$ et $\mathcal{A}' = (\Sigma, Q', I', F', \delta')$ deux automates finis sur le même alphabet. Quitte à renommer les états on suppose que $Q \cap Q' = \emptyset$.
En posant

$$\gamma = \delta \sqcup \delta' \sqcup \left\{ (q_f, \varepsilon, q_i') \;\middle|\; (q_f, q_i') \in F \times I' \right\}$$

l'automate $\mathcal{C} = (\Sigma, Q \sqcup Q', I, F', \gamma)$ reconnaît $\mathcal{L}(\mathcal{A}) \cdot \mathcal{L}(\mathcal{A}')$.

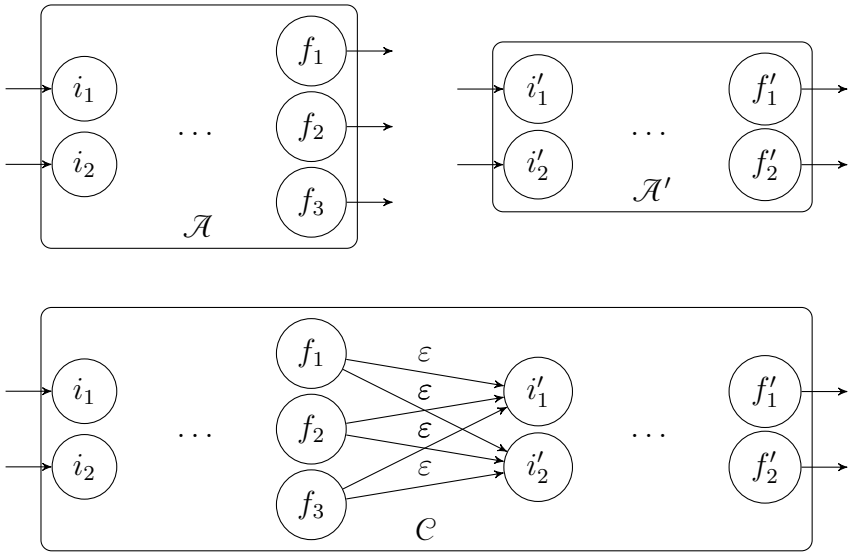


FIGURE 2 – Illustration du principe de la propriété 4.12

Démonstration : Montrons que $\mathcal{L}(\mathcal{C}) = \mathcal{L}(\mathcal{A})\cdot\mathcal{L}(\mathcal{A}')$ par double inclusion.

Soit $u \in \mathcal{L}(\mathcal{C})$. Il existe, par définition, $(q_i)_{i\in[\![0,n]\!]} \in Q^{n+1}$ et $(z_i)_{i\in[\![1,n]\!]} \in (\Sigma \sqcup \{\varepsilon\})^n$ une exécution acceptante de $\mathcal{C}$ d'étiquette u. Puisque q_n est un état final de $\mathcal{C}$, on a $q_n \in F' \subseteq Q'$, et puisque q_0 est un état initial de $\mathcal{C}$, on a $q_0 \in I \subseteq Q$. En particulier $q_0 \neq q_n$ et $n > 0$. De plus cela assure l'existence de $l = \max\{i \in [\![0,n]\!] \mid q_i \in Q\}$, et que $l < n$. Par construction $\forall i \in [\![0,l]\!]$, $q_i \in Q$ et $q_{l+1} \in Q'$, et puisque les seules transitions de Q vers Q' vont de F vers I', on en déduit que $q_l \in F$ et $q_{l+1} \in I'$. De plus, comme il n'existe aucune transition de Q' vers Q, on a par récurrence immédiate que $\forall i \in [\![l+1,n]\!]$, $q_i \in Q'$. Ainsi $(q_i)_{i\in[\![0,l]\!]}$ est une exécution acceptante de $\mathcal{A}$ d'étiquette $\pi(z_{[\![1,l]\!]})$, et $(q_i)_{i\in[\![l+1,n]\!]}$ est une exécution acceptante de $\mathcal{A}'$ d'étiquette $\pi(z_{[\![l+1,n]\!]})$. On a donc $\pi(z_{[\![1,l]\!]}) \in \mathcal{L}(\mathcal{A})$ et $\pi(z_{[\![l+1,n]\!]}) \in \mathcal{L}(\mathcal{A}')$. On en déduit que $\pi(z_{[\![1,l]\!]})\cdot\pi(z_{[\![l+1,n]\!]}) \in \mathcal{L}(\mathcal{A})\cdot\mathcal{L}(\mathcal{A}')$, or π est un morphisme donc $\pi(z_{[\![1,l]\!]})\cdot\pi(z_{[\![l+1,n]\!]}) = \pi(z_{[\![1,l]\!]}\cdot z_{[\![l+1,n]\!]}) = \pi(z) = u$, donc $u \in \mathcal{L}(\mathcal{A})\cdot\mathcal{L}(\mathcal{A}')$.
D'où $\mathcal{L}(\mathcal{C}) \subseteq \mathcal{L}(\mathcal{A})\cdot\mathcal{L}(\mathcal{A}')$.

Réciproquement, si $u \in \mathcal{L}(\mathcal{A})\cdot\mathcal{L}(\mathcal{A}')$, il existe $v \in \mathcal{L}(\mathcal{A})$ et $w \in \mathcal{L}(\mathcal{A}')$ tels que $u = v\cdot w$. Donc il existe $(q_i)_{i\in[\![0,n]\!]} \in Q^{n+1}$ une exécution acceptante de $\mathcal{A}$ d'étiquette v, et $(q'_i)_{i\in[\![0,m]\!]} \in Q'^{n+1}$ une exécution acceptante de $\mathcal{A}'$ d'étiquette w. Par définition d'une exécution $q'_0 \in I'$, et par celle d'une exécution acceptante $q_n \in F$. Donc il existe dans $\mathcal{C}$ une ε-transition de q_n vers q'_0, *i.e.* $(q_n, \varepsilon, q'_0) \in \gamma$. De plus toutes les transitions de $\mathcal{A}$ et $\mathcal{A}'$ sont aussi valides dans $\mathcal{C}$, *i.e.* $\delta \subseteq \gamma$ et $\delta' \subseteq \gamma$. Ainsi la suite $(s_i)_{i\in[\![0,n+m+1]\!]}$ définie par $\forall i \in [\![0,n]\!], s_i = q_i$ et $\forall i \in [\![n+1, n+m+1]\!], s_i = q'_{i-n-1}$ est une suite de transitions de $\mathcal{C}$, et comme $s_0 = q_0 \in I$ et $s_{n+m+1} = q'_m \in F'$, cette exécution est acceptante dans $\mathcal{C}$. On en déduit que le mot $v\cdot w$ est reconnu par $\mathcal{C}$, en effet l'ε-transition ajoutée aux transitions initiales ne modifie pas l'étiquette. Ainsi $u = v\cdot w \in \mathcal{L}(\mathcal{C})$.
D'où $\mathcal{L}(\mathcal{A})\cdot\mathcal{L}(\mathcal{A}') \subseteq \mathcal{L}(\mathcal{C})$ et finalement $\mathcal{L}(\mathcal{A})\cdot\mathcal{L}(\mathcal{A}') = \mathcal{L}(\mathcal{C})$. □

4.3.2 Étoile

Proposition 4.13

Soit $\mathcal{A} = (\Sigma, Q, I, F, \delta)$ un automate fini. On considère un nouvel état $q_\varepsilon \notin Q$. En posant $\rho = \left\{(q_f, \varepsilon, q_i) \;\middle|\; (q_f, q_i) \in F \times I\right\}$, l'automate $\mathcal{E} = (\Sigma, Q \sqcup \{q_\varepsilon\}, I \sqcup \{q_\varepsilon\}, F \sqcup \{q_\varepsilon\}, \delta \cup \rho)$ reconnaît $\mathcal{L}(\mathcal{A})^\star$.

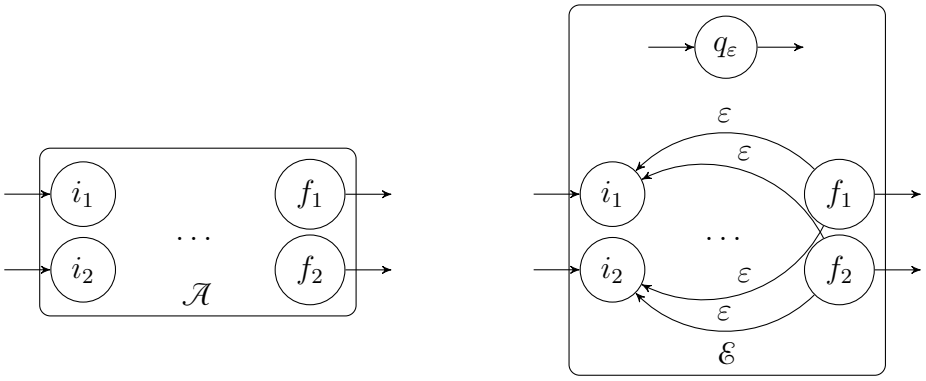


FIGURE 3 – Illustration du principe de la propriété 4.13

Démonstration : Montrons que $\mathcal{L}(\mathcal{E}) = \mathcal{L}(\mathcal{A})^\star$ par double inclusion.

Pour montrer que $\mathcal{L}(\mathcal{E}) \subseteq \mathcal{L}(\mathcal{A})^\star$, on s'intéresse aux exécutions acceptantes de $\mathcal{E}$ d'étiquette non vide et on montre qu'elles se décomposent en exécutions de $\mathcal{L}(\mathcal{A})$ par récurrence sur le nombre de transitions de ρ empruntées. Pour $k \in \mathbb{N}$ on définit donc le prédicat suivant.

P_k : si $(q_i)_{i\in[\![0,m]\!]}$ est une exécution acceptante de $\mathcal{E}$ d'étiquette $u \neq \varepsilon$ empruntant k transitions de ρ alors il existe $(v^j)_{j\in[\![0,k]\!]} \in \mathcal{L}(\mathcal{A})^{k+1}$ tels que $u = v^0 \cdot v^1 \cdot v^2 \cdot \dots v^k$.

On peut déjà remarquer qu'une exécution acceptante qui passe par l'état q_ε ne peut qu'être réduite à cet état — état dont aucune transition ne permet de sortir — et sont donc nécessairement d'étiquette vide. De ce fait, les transitions dont il est question ici ne passent que par des états de Q.

- Si $(q_i)_{i\in[\![0,m]\!]} \in Q^{m+1}$ est une exécution acceptante de $\mathcal{E}$ d'étiquette u n'empruntant aucune transition de ρ, alors toutes les transitions de cette exécution sont dans δ, autrement dit on a une exécution acceptante de $\mathcal{A}$, donc $u \in \mathcal{L}(\mathcal{A})$. Ainsi en posant $v^0 = u$, on a bien la décomposition voulue. D'où P_0 est vraie.
- Soit $k \in \mathbb{N}$. On suppose P_k vraie et on veut montrer que P_{k+1} l'est aussi. Soit $(q_i)_{i\in[\![0,m]\!]} \in Q^{m+1}$ une exécution acceptante de $\mathcal{E}$ d'étiquette u empruntant exactement $k+1$ transitions de ρ. Soit $m' \in [\![0, m-1]\!]$ l'indice tel que la dernière transition de ρ apparaissant dans cette exécution est $(q_{m'}, \varepsilon, q_{m'+1})$.
  Par définition de ρ, on a $q_{m'} \in F$ et $q_{m'+1} \in I$.
  D'une part, on en déduit que $(q_i)_{i\in[\![0,m']\!]}$ est une exécution acceptante de $\mathcal{E}$, et comme elle a exactement k transitions dans ρ, on sait par P_k que son étiquette s'écrit $v^0 \cdot v^1 \dots v^k$ avec $(v^j)_{j\in[\![0,k]\!]} \in \mathcal{L}(\mathcal{A})^k$.
  D'autre part, on en déduit que $(q_i)_{i\in[\![m'+1,m]\!]}$ est une exécution acceptante de $\mathcal{E}$, et comme elle a 0 transition dans ρ, on sait par P_0 que son étiquette s'écrit v^{k+1} avec $v^{k+1} \in \mathcal{L}(\mathcal{A})$.
  Enfin, puisque l'ε-transition $(q_{m'}, \varepsilon, q_{m'+1})$ ne contribue pas à l'étiquette, on en déduit que l'étiquette de l'exécution $(q_i)_{i\in[\![0,m]\!]}$ est $(v^0 \cdot v^1 \dots v^k) \cdot v^{k+1}$, et on a bien la décomposition annoncée, d'où P_{k+1}.

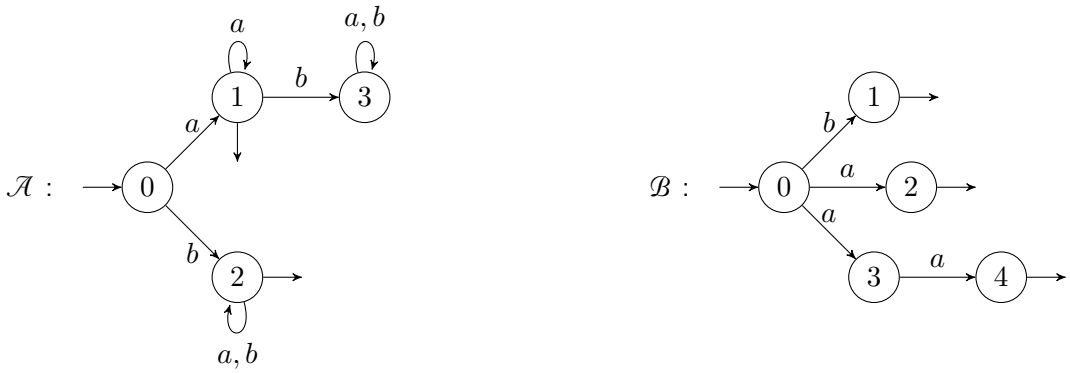
Par récurrence on en déduit que $\forall k \in \mathbb{N}$, P_k est vraie, autrement dit l'étiquette de toute exécution acceptante de $\mathcal{E}$, pourvue que celle-ci soit non vide, se décompose en concaténation de mots de $\mathcal{L}(\mathcal{A})$, et appartient donc à $\mathcal{L}(\mathcal{A})^\star$.

De plus les exécutions acceptantes de $\mathcal{E}$ d'étiquette vide ajoutent éventuellement ε à $\mathcal{L}(\mathcal{E})$, mais par définition de l'étoile de Kleene, $\varepsilon \in \mathcal{L}(\mathcal{A})^\star$, donc on a finalement $\mathcal{L}(\mathcal{E}) \subseteq \mathcal{L}(\mathcal{A})^\star$.

Réciproquement, si $u \in \mathcal{L}(\mathcal{A})^\star$, ou bien $u = \varepsilon$, auquel cas l'état q_ε à lui tout seul constitue une exécution de $\mathcal{E}$ qui accepte u, donc $u \in \mathcal{L}(\mathcal{E})$, ou bien il existe $k \in \mathbb{N}$ et $(v^j)_{j\in[\![0,k]\!]} \in \mathcal{L}(\mathcal{A})^{k+1}$ tels que $u = v^0 \cdot v^1 \cdot v^2 \cdot \dots v^k$. Dans ce cas, pour tout $j \in [\![0, k]\!]$, il existe $(q_i^j)_{i\in[\![0,n^j]\!]} \in Q^{n^j+1}$ une exécution acceptante de $\mathcal{A}$ d'étiquette v^j. Par définition d'une exécution, $\forall j \in [\![0, k]\!]$, $q_0^j \in I$, et par celle d'une exécution acceptante $\forall j \in [\![0, k]\!]$, $q_{n^j}^j \in F$, donc $\forall j \in [\![0, k1]\!]$, $(q_{n^j}^j, \varepsilon, q_0^{j+1}) \in \rho$ *i.e.* il existe dans $\mathcal{E}$ une ε-transition de $q_{n^j}^j$ vers q_0^{j+1}. De plus toutes les transitions de $\mathcal{A}$ sont aussi valides dans $\mathcal{E}$, ainsi la suite $(s_i)_{i\in[\![0,m]\!]}$ définie par $m = -1 + \sum_{j=0}^{k}(n^k + 1)$ et $\forall j \in [\![0, k]\!], \forall i \in [\![0, n^j]\!], s_{j+i} = q_i^j$ est une suite de transitions de $\mathcal{E}$, et comme $s_0 = q_0^0 \in I$ et $s_m = q_{n^k}^k \in F$, cette exécution est acceptante dans $\mathcal{E}$. On en déduit que le mot $v^0 \cdot v^1 \cdot \dots v^k$ est reconnu par $\mathcal{E}$, en effet les ε-transitions ajoutées pour lier les exécutions initiales ne contribuent pas à l'étiquette. Ainsi $u \in \mathcal{L}(\mathcal{E})$.
D'où $\mathcal{L}(\mathcal{A})^\star \subseteq \mathcal{L}(\mathcal{E})$ et finalement $\mathcal{L}(\mathcal{A})^\star = \mathcal{L}(\mathcal{C})$. □

Exercice de cours 4.14

Considérons les deux automates ci-dessous.



En appliquant l'algorithme du cours, donner un automate pour le langage $\mathcal{L}(\mathcal{B})^\star$, puis un automate pour le langage $\mathcal{L}(\mathcal{A})\cdot\mathcal{L}(\mathcal{B})^\star$. Notons $\mathcal{C}$ l'automate ainsi obtenu. **En appliquant l'algorithme du cours**, supprimer les ε-transitions de l'automate $\mathcal{C}$ ainsi obtenu, notons $\mathcal{D}$ l'automate ainsi obtenu.

4.3.3 Union

Proposition 4.15

Soient $\mathcal{A} = (\Sigma, Q, I, F, \delta)$ et $\mathcal{A}' = (\Sigma, Q', I', F', \delta')$ deux automates finis sur le même alphabet. Quitte à renommer les états on suppose que $Q \cap Q' = \emptyset$. L'automate $\mathcal{U} = (\Sigma, Q \sqcup Q', I \sqcup I', F \sqcup F', \delta \sqcup \delta')$ reconnaît $\mathcal{L}(\mathcal{A}) \cup \mathcal{L}(\mathcal{A}')$.

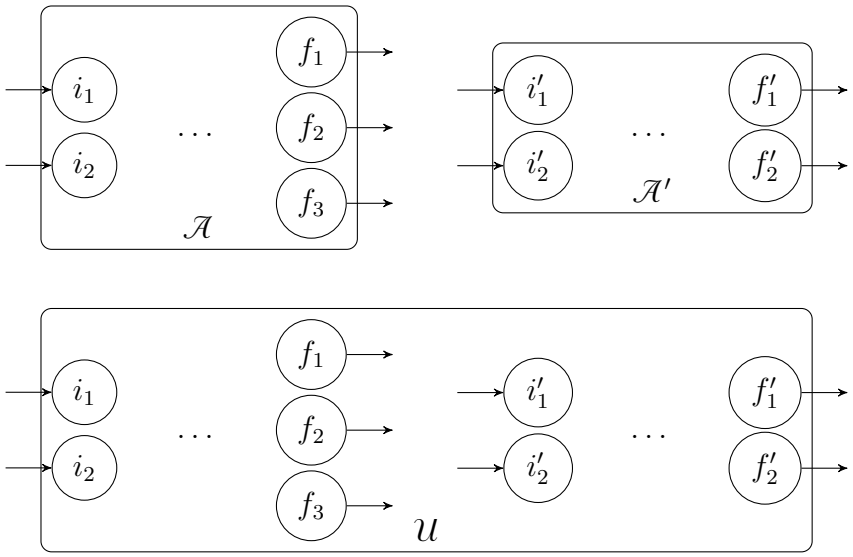


FIGURE 4 – Illustration du principe de la propriété 4.15

Démonstration : Montrons que $\mathcal{L}(\mathcal{U}) = \mathcal{L}(\mathcal{A}) \cup \mathcal{L}(\mathcal{A}')$ par double inclusion.

Soit $u \in \mathcal{L}(\mathcal{U})$. il existe, par définition, $(q_i)_{i\in[\![0,n]\!]}$ une exécution acceptante de $\mathcal{U}$ d'étiquette u. Puisque q_0 est un état initial de $\mathcal{U}$, supposons sans perdre en généralité que $q_0 \in I$, on montre alors de proche en proche que $\forall i \in [\![0,n]\!], q_i \in Q$ et $(q_i)_{i\in[\![0,n]\!]}$ une exécution acceptante de $\mathcal{A}$. Ainsi $u \in \mathcal{L}(\mathcal{A})$.

Réciproquement, si $u \in \mathcal{L}(\mathcal{A})$, il existe $(q_i)_{i\in[\![0,n]\!]} \in Q^{n+1}$ une exécution acceptante de $\mathcal{A}$ d'étiquette u. Cette exécution est aussi une exécution acceptante dans $\mathcal{U}$. Ainsi $u \in \mathcal{L}(\mathcal{U})$. □

5 Théorème de Kleene

Cette section vise à démontrer le résultat principal du chapitre : le théorème de Kleene, qui assure que la classe des langages reconnaissables et celle des langages réguliers sont les mêmes. Nous avons vu dans la section précédente, via l'utilisation d'automates avec ε-transitions, que l'ensemble des langages reconnaissables est stable par les opérations régulières, démontrant ainsi que pour toute expression régulière e sur un alphabet Σ, il existe un automate $\mathcal{A}$ tel que $\mathcal{L}(e) = \mathcal{L}(\mathcal{A})$. Aussi le premier sens de la démonstration a déjà été fourni par la section précédente. On se contente donc ici de démontrer la réciproque, à savoir que pour tout automate $\mathcal{A}$ il existe une expression régulière e telle que $\mathcal{L}(e) = \mathcal{L}(\mathcal{A})$.

5.1 Les langages reconnaissables sont réguliers

Définition 5.1

*Un **automate généralisé** est un quintuplet $\mathcal{A} = (\Sigma, Q, I, F, \delta)$ où :*
- *Σ est un alphabet ;*
- *Q est un ensemble fini ;*
- *$I \subseteq Q$;*
- *$F \subseteq Q$;*
- *$\delta \subseteq Q \times \mathscr{E}_{reg}(\Sigma) \times Q$ tel que $\forall (q, q') \in Q^2$, $\text{card}\{r \in \mathscr{E}_{reg}(\Sigma) \mid (q, r, q') \in \delta\} \leqslant 1$.*

Remarque 5.2

On peut retenir qu'un automate généralisé sur Σ est un automate dont les transitions sont étiquetées par des expressions régulières sur Σ, et ayant au plus une transition entre deux états, à condition de retenir que la définition du langage reconnu (donnée ci-après) est très différente de celle d'un automate classique.

Définition 5.3

Soit $\mathcal{A} = (\Sigma, Q, I, F, \delta)$ un automate généralisé.
*Un mot $w \in \Sigma^\star$ est **accepté** par l'automate généralisé $\mathcal{A}$ s'il existe $(u_i)_{i \in [\![1,n]\!]}$ une suite de mots de $\Sigma^\star$ et $(q_i)_{i \in [\![0,n]\!]}$ une suite d'états de Q telles que $w = u_1 \cdot u_2 \cdot \ldots u_n$, $q_0 \in I$, $q_n \in F$ et*

$$\forall i \in [\![1, n]\!],\ \exists e_i \in \mathscr{E}_{reg}(\Sigma),\ (q_{i-1}, e_i, q_i) \in \delta \text{ et } u_i \in \mathcal{L}(e_i).$$

*On appelle alors **langage reconnu** par $\mathcal{A}$ l'ensemble des mots de $\Sigma^\star$ acceptés par $\mathcal{A}$.*

Exemple 5.4
L'automate ci-dessous est un automate généralisé.

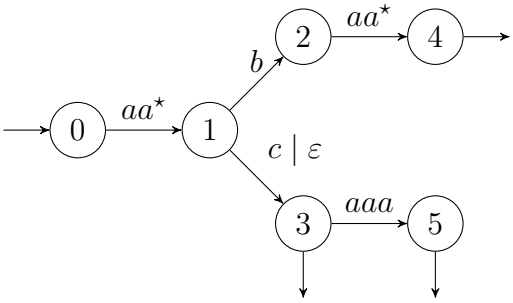


FIGURE 5 – Un automate généralisé

Exercice de cours 5.5

Déterminer, au moyen d'une expression régulière le langage de l'automate de la Figure 5 ci-dessus. Donner un automate non généralisé, sans ε-transitions reconnaissant le même langage.

Remarque 5.6

Puisque l'ajout de transitions étiquetées par l'expression $\emptyset$ n'a aucune conséquence sur le langage reconnu, on peut supposer qu'il y a exactement une transition entre deux états d'un automate généralisé.

Remarque 5.7

Les transitions d'un automate généralisé peuvent être décrites au moyen d'une fonction $T : Q \times Q \to \mathscr{E}_{reg}(\Sigma)$ associant à chaque couple d'états (q, q') l'expression régulière étiquetant l'unique transition de q vers q' si elle existe, et l'expression régulière $\emptyset$ sinon.

Exemple 5.8
Pour l'automate exemple ci-avant la table de transitions est la suivante.

	0	1	2	3	4	5
0	$\emptyset$	$aa^\star$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
1	$\emptyset$	$\emptyset$	b	$c \mid \varepsilon$	$\emptyset$	$\emptyset$
2	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$aa^\star$	$\emptyset$
3	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	aaa
4	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
5	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

Définition 5.9

Un automate généralisé est dit ***bien détouré***[♣] *s'il a un unique état initial i, que celui-ci n'admet aucune transition entrante, et s'il a un unique état final $f \neq i$, et que celui-ci n'admet aucune transition sortante.*

Lemme 5.10

Soit $\mathcal{A} = (\Sigma, Q, I, F, \delta)$ un automate généralisé.
Pour i et f deux nouveaux états (i.e. $i \notin Q$, $f \notin Q$ et $i \neq f$), en posant :
- $Q' = Q \sqcup \{i, f\}$,
- $I' = \{i\}$,
- $F' = \{f\}$,
- *et* $\delta' = \delta \sqcup \{(i, \varepsilon, q) \mid q \in I\} \sqcup \{(q, \varepsilon, f) \mid q \in F\}$,

l'automate $\mathcal{A}' = (\Sigma, Q', I', F', \delta')$ est un automate généralisé bien détouré équivalent à $\mathcal{A}$.

Démonstration : Preuve analogue à celles de la section 3.2. □

Exercice de cours 5.11

En appliquant l'algorithme ci-avant, donner un automate généralisé, bien détouré, équivalent à l'automate de la figure 5.

Lemme 5.12

Si $\mathcal{A}$ est un automate généralisé bien détouré ayant n états avec $n > 2$, alors il existe un automate généralisé bien détouré $\mathcal{A}'$, équivalent à $\mathcal{A}$, et ayant seulement $n-1$ états.

♣. Attention : terminologie locale

Démonstration : Soit $\mathcal{A} = (\Sigma, Q, \{i\}, \{f\}, \delta)$ un automate généralisé bien détouré ayant n états avec $n > 2$. D'après la remarque 5.7, on peut décrire les transitions de $\mathcal{A}$ par une fonction $T : Q \times Q \to \mathscr{E}_{reg}(\Sigma)$. Soit donc un état $q \in Q \setminus \{i, f\}$ (ainsi q n'est ni initial, ni final). On considère alors l'automate généralisé $\mathcal{A}' = (\Sigma, Q', \{i\}, \{f\}, T')$ défini par l'ensemble d'états $Q' = Q \setminus \{q\}$ et la fonction $T' : Q' \times Q' \to \mathscr{E}_{reg}(\Sigma)$ définie comme suit.

$$\forall (q_a, q_b) \in Q',\ T'(q_a, q_b) = T(q_a, q_b) \,|\, T(q_a, q) \cdot (T(q, q))^\star \cdot T(q, q_b)$$

Cela revient à compenser la suppression de l'état q comme le schématise la figure 6. On vérifie alors facilement que cette construction assure que toute exécution de $\mathcal{A}$ passant par q peut se réécrire comme exécution de $\mathcal{A}'$. Ainsi nous avons $\mathcal{L}(\mathcal{A}') = \mathcal{L}(\mathcal{A})$, et $\mathcal{A}'$ a bien un état de moins que $\mathcal{A}$.

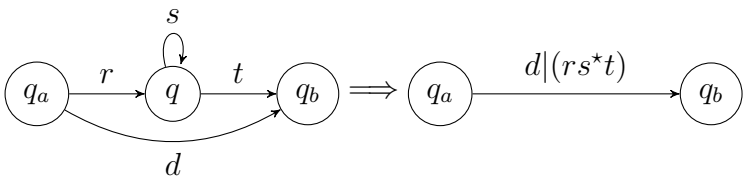


FIGURE 6 – Schéma de la suppression d'un état q ni initial ni final dans un automate généralisé

□

Théorème 5.13

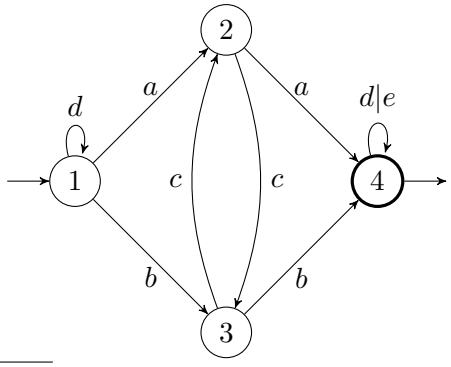
Tout langage reconnaissable est régulier.
De plus on a un algorithme qui, étant donné un automate $\mathcal{A}$, construit une expression régulière e équivalente à $\mathcal{A}$.

Démonstration : Soit un automate $\mathcal{A} = (\Sigma, Q, I, F, \delta)$, On définit $\mathcal{A}_g$ un automate généralisé reconnaissant le même langage que $\mathcal{A}$ par $\mathcal{A}_g = (\mathscr{E}_{reg}(\Sigma), Q, I, F, T)$ où $T(q_a, q_b) = |_{a \in \Sigma | (q_a, a, q_b) \in \delta)} a$ ♣. Ainsi, pour $(q_a, q_b) \in Q^2$, $T(q_a, q_b)$ est l'expression régulière représentant l'union des lettres permettant d'aller de q_a en q_b dans $\mathcal{A}$ s'il en existe, ou l'expression régulière $\emptyset$ s'il n'en existe pas. Dans ce dernier cas, on peut aussi ne pas mettre de transition de q_a à q_b. On construit alors $\mathcal{A}'_g$ un automate généralisé détouré reconnaissant le même langage que $\mathcal{A}_g$. Tant que cet automate a strictement plus de deux états, on supprime un état qui n'est ni initial ni final, tout en maintenant le langage reconnu (comme expliqué dans la propriété 5.12). On obtient finalement un automate généralisé détouré $\mathcal{A}_f = (\mathsf{Reg}(\Sigma), \{i, f\}, \{i\}, \{f\},)$ à seulement deux états et reconnaissant le même langage que $\mathcal{A}$. Or le langage reconnu par cet automate est $\mathcal{L}(T(i, f))$, qui est régulier.

□

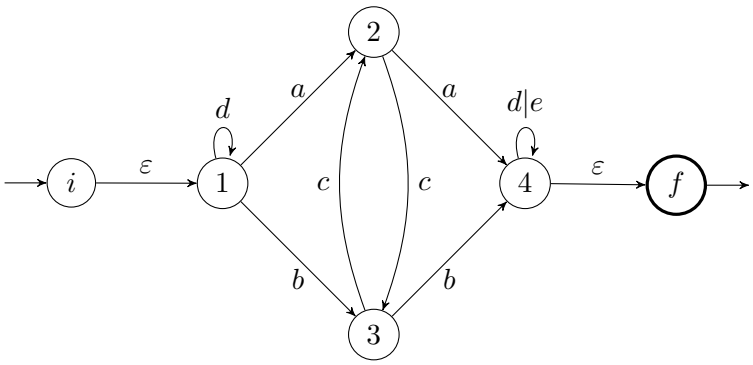
Exemple 5.14

- Automate inital :

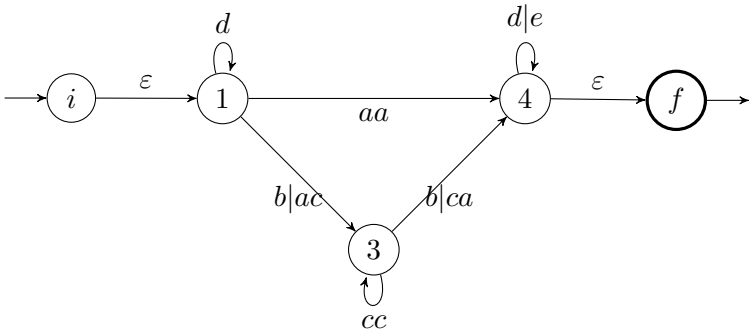


♣. Par exemple si les transitions entre l'état q_a et l'état q_b sont étiquetées par a, b et d, alors $T(q_a, q_b) = a|b|c$

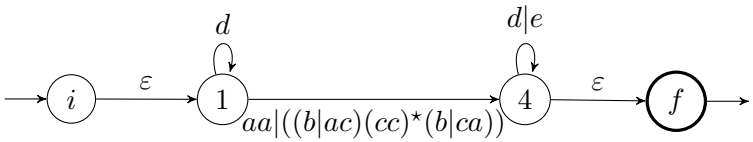
- Transformation en automate généralisé bien détouré :



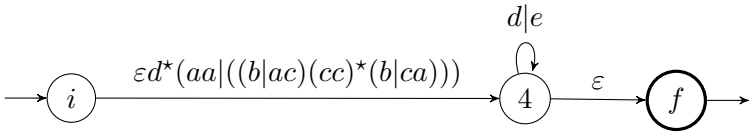
- Suppression de l'état 2 :



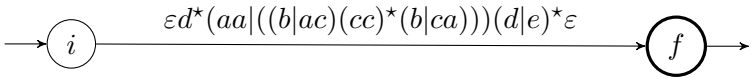
- Suppression de l'état 3 :



- Suppression de l'état 1 :



- Suppression de l'état 4 :



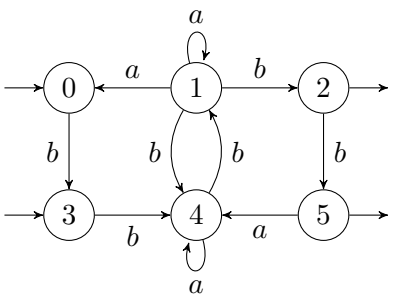
On peut alors lire le résultat sur l'automate : $\varepsilon d^\star(aa|((b|ac)(cc)^\star(b|ca)))(d+e)^\star\varepsilon$

Exercice de cours 5.15

Donner un exemple justifiant que l'ordre de suppression des états influe sur l'expression régulière obtenue.

Exercice de cours 5.16

En appliquant l'algorithme ci-dessus, donner une expression régulière équivalente à l'automate ci-dessous.



5.2 Conclusion théorème

Théorème 5.17

Un langage est reconnaissable si et seulement s'il est régulier, de plus on a des algorithmes permettant la construction d'un automate équivalent à une expression régulière et réciproquement.