AUTOMATES FINIS DÉTERMINISTES

À la fin de ce chapitre, je sais :

- ☞ définir un automate fini déterministe
- ☞ représenter un automate fini déterministe
- ☞ qualifier les états d'un automates en termes d'accessibilité et de co-accessibilité
- ☞ définir un langage reconnu par un automate
- ☞ compléter un automate fini déterministe
- ☞ complémenter un automate fini déterministe
- ☞ faire le produit de deux automates finis déterministes
- ☞ utiliser la stabilité des langages reconnus par complémentation et intersection

Les automates sont utilisés pour réaliser :

- l'automatisation de comportements simples (systèmes embarqués),
- des circuits électroniques, des protocoles de communication, des processus,
- la recherche d'un mot dans un texte en temps linéaire par rapport à la taille du texte,
- la compilation d'un code informatique (analyse lexicale et syntaxique)

Les automates sont des machines simples dont les entrées sont des lettres et la sortie un résultat. Ces machines sont constituées par des états qui sont interconnectés par des liens permettant le passage d'un état à un autre selon un entrée et une direction. Selon les entrées reçues, l'automate réagit et se positionne donc dans un certain état. Les automates sont des éléments essentiels de l'informatique : ils établissent un lien fort entre la théorie des graphes et la théorie des langages.

Les chapitres de ce cours se focalisent sur les automates dont le résultat est l'acceptation ou non d'un mot d'un langage. Plus particulièrement, ce chapitre traite des automates finis déterministes (AFD), le déterminisme étant la capacité de l'automate à se positionner dans un seul état possible après la réception d'un lettre.

A Automate fini déterministe (AFD)

■ **Définition 1 — Automate fini déterministe (AFD).** Un automate fini déterministe est un quintuplet $(Q, \Sigma, q_i, \delta, F)$ tel que :

1. Q est un ensemble non vide et fini dont les éléments sont les états,
2. Σ est l'alphabet,
3. $q_i \in Q$ est l'état initial,
4. $\delta : Q \times \Sigma \longrightarrow Q$ est la **fonction** de transition de l'automate,
5. $F \subseteq Q$ est l'ensemble des états accepteurs ou terminaux.

R Le déterminisme d'un AFD est dû aux faits que :

- l'état initial est un **singleton**,
- δ est une **fonction** : à un couple (état,lettre) (q, a), δ associe au plus un état q'.

■ **Définition 2 — Fonction de transition partielle.** On dit que la fonction de transition δ est partielle s'il existe au moins un couple (état,lettre) pour lequel elle n'est pas définie.

■ **Définition 3 — Automate complet.** Un AFD $\mathcal{A} = (Q, \Sigma, q_i, \delta, F)$ est dit complet si δ est une **application**, c'est-à-dire δ n'est pas partielle, il existe une transition pour chaque lettre de Σ pour tous les états.

■ **Définition 4 — Automate normalisé.** Un automate est normalisé s'il ne possède pas de transition entrante sur son état initial et s'il possède un seul état final sans transition sortante.

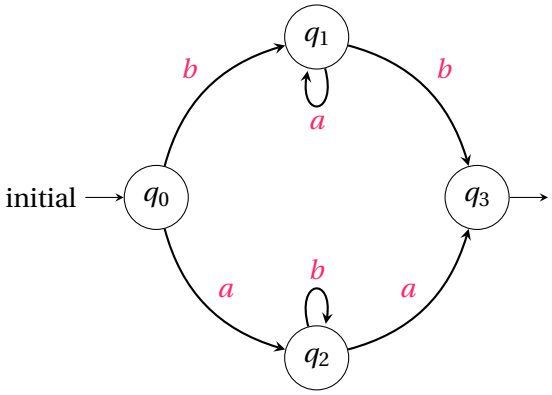


FIGURE 1 – Exemple d'automate fini déterministe normalisé

B Représentations d'un automate

Les automates peuvent être représentés sous la forme de tableaux ou de graphes comme le montre les figures 1 et 1. Les tableaux sont utiles lors de l'exécution manuelle des algorithmes sur les automates. Les algorithmes sur les graphes pourront être d'une aide précieuse pour l'étude des automates.

	$\downarrow q_0$	q_1	q_2	$\uparrow q_3$
a	q_2	q_1	q_3	
b	q_1	q_3	q_2	

	a	b
$\downarrow q_0$	q_2	q_1
q_1	q_1	q_3
q_2	q_3	q_2
$\uparrow q_3$		

TABLE 1 – Exemple d'automate fini déterministe représenté sous la forme de tableaux : on peut choisir de représenter les états en ligne ou en colonne.

C Acceptation d'un mot

■ **Définition 5 — Fonction de transition étendue aux mots.** La fonction de transition peut être étendue aux mots par passages successifs d'un état à un autre en lisant les lettres d'un mot.

On définit inductivement cette fonction étendue noté δ^* :

$$\forall q \in Q, \delta^*(q, \epsilon) = q \tag{1}$$

$$\forall q \in Q, \forall w \in \Sigma^*, \forall a \in \Sigma, \delta^*(q, w.a) = \delta(\delta^*(q, w), a) \tag{2}$$

■ **Définition 6 — Acceptation d'un mot par un automate.** Un mot $w \in \Sigma^*$ est **accepté** par un automate $\mathcal{A}$ si et seulement si $\delta^*(q_i, w) \in F$, c'est-à-dire la lecture du mot w par l'automate conduit à un état accepteur.

■ **Définition 7 — Langage reconnu par un AFD.** Le langage $\mathcal{L}_{\text{rec}}(\mathcal{A})$ **reconnu** par un automate fini déterministe $\mathcal{A}$ est l'ensemble des mots reconnus par $\mathcal{A}$:

$$\mathcal{L}_{\text{rec}}(\mathcal{A}) = \{w \in \Sigma^*, w \text{ est accepté par } \mathcal{A}\} \tag{3}$$

■ **Définition 8 — Langage reconnaissable.** Un langage $\mathcal{L}$ sur un alphabet Σ est reconnaissable s'il existe un automate fini déterministe $\mathcal{A}$ d'alphabet Σ tel que $\mathcal{L} = \mathcal{L}_{\text{rec}}(\mathcal{A})$.

D Accessibilité et co-accessibilité

■ **Définition 9 — Accessibilité d'un état.** Un état q d'un automate est dit accessible s'il existe un mot $w \in \Sigma^*$ tel que $\delta^*(q_i, u) = q$, c'est-à-dire il est possible de l'atteindre depuis l'état initial.

■ **Définition 10 — Co-accessibilité d'un état.** Un état q d'un automate est dit co-accessible s'il existe un mot $w \in \Sigma^*$ tel que $\delta^*(q, u) \in F$, c'est-à-dire à partir de cet état, il est possible d'atteindre un état accepteur.

■ **Définition 11 — Automate émondé.** Un automate est dit émondé tous ses états sont à la fois accessibles et co-acessibles.

Théorème 1 — Automate d'un langage reconnaissable. Si un langage est reconnaissable alors il existe un automate fini déterministe :

- normalisé qui le reconnaît.
- émondé qui le reconnaît.
- complet qui le reconnaît.

E Complétion d'un AFD

Sur la figure 2, on observe que certaines transitions ne sont pas précisées : par exemple, si la lettre a arrive en l'état q_1, aucune transition n'est spécifiée. Compléter un automate, c'est préciser ces transitions en ajoutant un état puits.

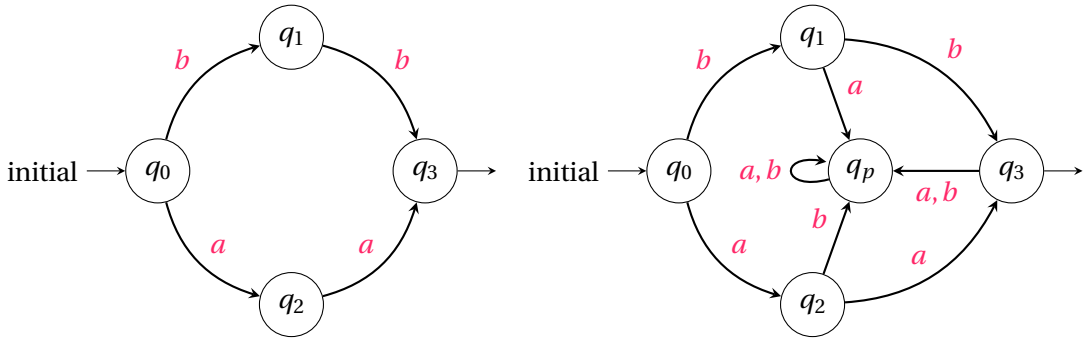


FIGURE 2 – Exemple d'automate fini déterministe non complet (à gauche) et complété (à droite)

M **Méthode 1 — Complété d'un AFD** Le complété d'un automate fini déterministe $\mathcal{A} = (Q, \Sigma, q_i, \delta, F)$ noté $C(\mathcal{A})$ est l'automate

$$C(\mathcal{A}) = (Q \cup \{q_p\}, \Sigma, q_i, C(\delta), F) \tag{4}$$

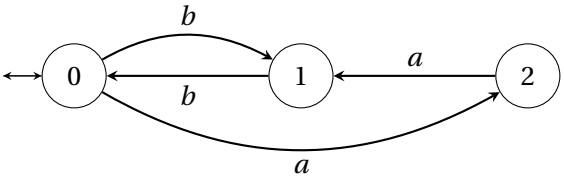
tel que :

- $q_p \notin Q$ est appelé l'état puits,
- $C(\delta)$ est l'application de $Q \cup \{q_p\} \times \Sigma$ dans $Q \cup \{q_p\}$ telle que :

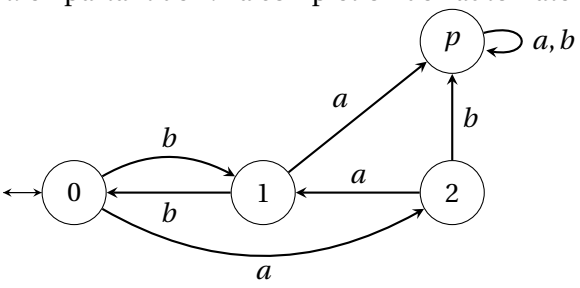
$$C(\delta)(q,a) = \begin{cases} \delta(q,a) \text{ si } \delta \text{ est définie pour } (q,a) \\ q_p \text{ sinon} \end{cases} \tag{5}$$

Cette méthode consiste donc à ajouter un état puits qui n'est pas co-accessible et à y faire converger toutes les transitions manquantes.

■ **Exemple 1 — Complétion d'un automate.** On considère l'automate fini déterministe suivant :



On observe que la fonction de transition n'est pas défini pour b en partant de l'état 2 ni pour a en partant de 1. La complétion de l'automate selon la méthode 1 donne :



Théorème 2 — Langage reconnu par un automate et son complété. Un automate $\mathcal{A}$ et son complété $C(\mathcal{A})$ reconnaissent le même langage :

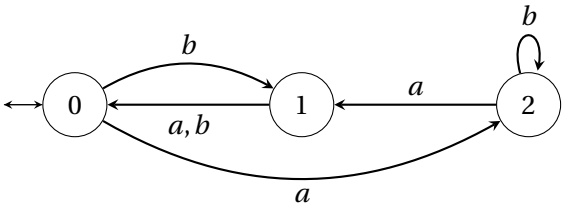
$$\mathcal{L}_{\text{rec}}(\mathcal{A}) = \mathcal{L}_{\text{rec}}(C(\mathcal{A})) \tag{6}$$

Démonstration. La fonction de transition pour un mot reconnu est la même sur les automates $\mathcal{A}$ et $C(\mathcal{A})$. Pour un mot non reconnu, elle diffère mais dans ce cas le mot n'appartient pas au langage. Donc les mots reconnus sont les mêmes. ■

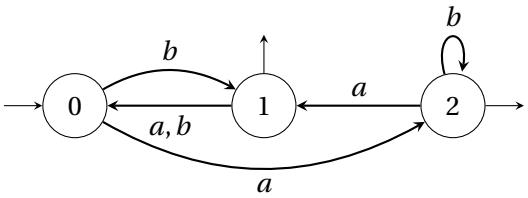
F Complémentaire d'un AFD

M **Méthode 2 — Complémentaire d'un AFD** Le complémentaire d'un automate fini déterministe **complet** $\mathcal{A} = (Q, \Sigma, q_i, \delta, F)$ noté $\mathcal{A}^c$ est l'automate complet $\mathcal{A}^c = (Q, \Sigma, q_i, \delta, Q \setminus F)$.

■ **Exemple 2 — Complémentaire d'AFD.** On considère l'automate fini déterministe **complet** suivant :



On observe que seul l'état 0 est un état accepteur. Le complémentaire de l'automate selon la méthode 2 donne :



Théorème 3 — Langage reconnu par un automate et son complémentaire. Le langage reconnu par l'automate complémentaire d'un automate complet est le complémentaire du langage reconnu par cet automate :

$$\mathcal{L}_{\text{rec}}(\mathcal{A}^c) = \Sigma^* \setminus \mathcal{L}_{\text{rec}}(\mathcal{A}) \tag{7}$$

Démonstration. On procède par double inclusion.

(⊂) Si w est un mot reconnu par l'automate complémentaire, alors l'état accepteur de w n'appartient pas F, $\delta^*(q_i, w) \in Q \setminus F$. Donc w n'appartient pas à $\mathcal{L}_{\text{rec}}(\mathcal{A})$.

(⊃) Soit w un mot de l'ensemble $\Sigma^* \setminus \mathcal{L}_{\text{rec}}(\mathcal{A})$. En utilisant l'automate $\mathcal{A}$, le chemin emprunté en suivant les lettres de w mène donc à un état non accepteur de $\mathcal{A}$. Comme les états non accepteurs de $\mathcal{A}$ sont les états accepteurs de $\mathcal{A}^c$, w est donc un mot de $\mathcal{L}_{\text{rec}}(\mathcal{A}^c)$.

■

R Avant de complémenter un automate, il convient de vérifier que celui-ci est complet afin de ne rater aucun mot.

Théorème 4 — Stabilité des langages reconnaissables par complémentation. Les langages reconnaissables sont stable par complémentation : s'il existe un langage reconnaissable sur Σ par un automate $\mathcal{A}$, alors le complémentaire de ce langage est reconnaissable par $\mathcal{A}^c$.

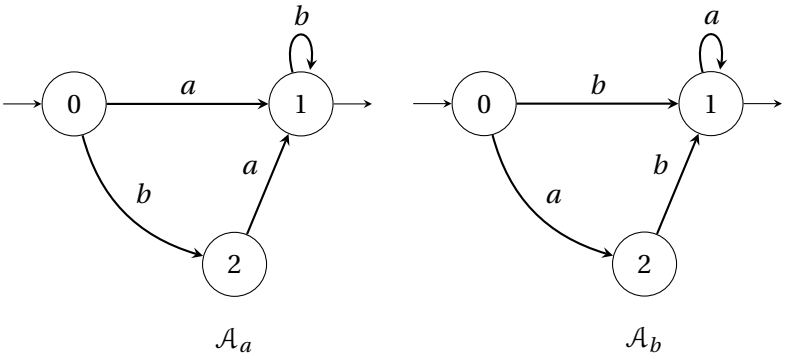
Démonstration. Ce théorème est une conséquence du théorème précédent et de la définition de l'automate complémentaire : si un langage est reconnaissable, alors son complémentaire l'est aussi puisqu'il existe un automate fini qui le reconnaît. ■

G Produit de deux AFD - Automates produit

■ **Définition 12 — Produit de deux automates.** Le produit de deux automates sur un même alphabet Σ, $\mathcal{A}_a = (Q_a, \Sigma, q_{i_a}, \delta_a, F_a)$ et $\mathcal{A}_b = (Q_b, \Sigma, q_{i_b}, \delta_b, F_b)$, noté $\mathcal{A}_a \times \mathcal{A}_b$ est l'automate $\mathcal{A} = (Q, \Sigma, q_i, \delta, F)$ tel que :

- $Q = Q_a \times Q_b$
- $q_i = (q_{i_a}, q_{i_b})$
- $\delta : (Q_a \times Q_b) \times \Sigma \longrightarrow (Q_a \times Q_b)$ est définie par : $\delta((q_a, q_b), s) = (\delta_a(q_a, s), \delta_b(q_b, s))$
- $F = F_a \times F_b$

■ **Exemple 3 — Exemple d'automate produit.** On considère les deux automates suivants sur le même alphabet $\Sigma = \{a, b\}$:



Pour comprendre le fonctionnement de l'automate produit $\mathcal{A}_a \times \mathcal{A}_b$, il faut imaginer que lors du parcours lié à un mot w, on avance simultanément sur les deux automates. Si les deux s'arrêtent simultanément sur un état accepteur à la fin de la lecture de w, alors w est un mot de l'automate produit.

Par exemple, les mots ab et ba sont à la fois reconnus par $\mathcal{A}_a$ et $\mathcal{A}_b$: ce sont des mots de $\mathcal{L}_{\text{rec}}(\mathcal{A}_a \times \mathcal{A}_b)$. Par contre, aba est reconnu par $\mathcal{A}_b$ mais pas par $\mathcal{A}_a$.

Théorème 5 — Langage reconnu par un produit d'automates. Le langage reconnu par un produit d'automates est l'intersection des langages reconnus par ces automates :

$$\mathcal{L}_{\text{rec}}(\mathcal{A}_a \times \mathcal{A}_b) = \mathcal{L}_{\text{rec}}(\mathcal{A}_a) \cap \mathcal{L}_{\text{rec}}(\mathcal{A}_b) \tag{8}$$

Démonstration. On procède par double inclusion et on utilise la fonction de transition étendue aux mots.

($\subset$) Soit w un mot reconnu par l'automate produit. Alors, par définition de l'automate produit :

$\delta^*((q_a, q_b), w) = (\delta_a^*(q_a, w), \delta_b^*(q_b, w))$ et $\delta_a^*(q_a, w) \in F_a$ et $\delta_b^*(q_b, w) \in F_b$.
Cela signifie que w est reconnu à la fois par $\mathcal{A}_a$ et $\mathcal{A}_b$. w appartient donc aux deux langages reconnus par $\mathcal{A}_a$ et $\mathcal{A}_b$.
Donc $w \in \mathcal{L}_{\text{rec}}(\mathcal{A}_a) \cap \mathcal{L}_{\text{rec}}(\mathcal{A}_b)$.

(⊃) soit w un mot reconnu par $\mathcal{A}_a$ et par $\mathcal{A}_b$. Alors un existe un chemin dans $\mathcal{A}_a$ qui, d'après le mot w, mène à un état accepteur de F_a. De même pour F_b. Donc $(\delta_a(q_a, w), \delta_b(q_b, w)) \in F_a \times F_b$. w appartient à $\mathcal{L}_{\text{rec}}(\mathcal{A}_a \times \mathcal{A}_b)$.

■

Théorème 6 — Stabilité des langages reconnaissables par l'intersection. Les langages reconnaissables sont stables par l'intersection : l'intersection de deux langages reconnaissables est un langage reconnaissable.

Démonstration. Ce théorème est un corolaire du théorème précédent. ■

R La stabilité des langages reconnaissables est importante car nous montrerons par la suite que les langages reconnaissables sont les langages réguliers. Or, les langages réguliers ne sont pas stables par définition pour les opérations non régulières, tout comme les langages reconnaissables ne sont pas stables par définition pour les opérations régulières (union, concaténation et fermeture de Kleene). Le théorème de Kleene va nous permettre d'étendre ces résultats d'une représentation à une autre.