AUTOMATES FINIS NON DÉTERMINISTES

À la fin de ce chapitre, je sais :

- ☞ reconnaître un automate fini non déterministe (AFND)
- ☞ déterminiser un AFND
- ☞ construire une AFND reconnaissant un langage rationnel simple

A Automate fini non déterministe (AFND)

■ **Définition 1 — Automate fini non déterministe (AFND).** Un automate fini non 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. $Q_i \subseteq Q$ sont **les** états initiaux,
4. $\Delta \subseteq Q \times \Sigma \times Q$ est la **relation** de transition de l'automate,
5. $F \subseteq Q$ est l'ensemble des états accepteurs ou terminaux.

R Un AFND est par essence asynchrone. Son exécution nécessite l'exécution de toutes les transitions possibles au départ de chaque état traversé.

R Le non déterminisme d'un AFND est dû au fait que Δ n'est pas une fonction mais une relation : depuis un état q, une même lettre peut faire transiter l'AFND vers des états différents. Par exemple, (q, a, q') et (q, a, q''). Quelle transition choisir ? Là est le non déterminisme.

R Un AFND peut posséder plusieurs états initiaux, ce qui n'est pas le cas d'un AFD (q_i devient Q_i). On peut facilement se ramener à un seul état initial en utilisant des transitions spontanées. C'est pourquoi on considèrera souvent ce cas.

R Qui peut le plus peut le moins : un AFD est un cas particulier d'AFND pour lequel $\Delta = \{(q, a, q'), \delta(q, a) = q'\}$.

B Représentation d'un AFND

Les AFND peuvent être représentés de la même manière que les AFD sous la forme de tableaux ou de graphes comme le montre les figures 1 et 1.

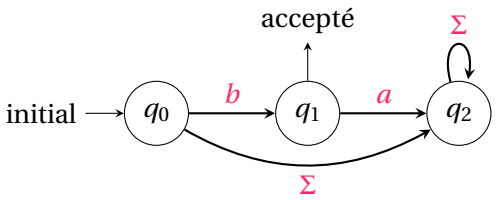


FIGURE 1 – Exemple d'automate fini non déterministe représenté sous la forme d'un graphe. On suppose que $\Sigma = \{a, b\}$ est l'alphabet

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

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

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

R Pour un même mot, il peut donc exister plusieurs exécutions possibles sur un AFND. La programmation des AFND n'est donc pas aussi simple que celles de AFD. Il faut être en mesure de tester tous les chemins possibles!

C Acceptation d'un mot

■ **Définition 2 — Relation de transition étendue aux mots.** La relation de transition peut être étendue aux mots par passages successifs d'un état à un autre en lisant les lettres d'un mot. On la note Δ^* .

■ **Définition 3 — Langage reconnu par un AFND.** Le langage $\mathcal{L}_{rec}(\mathcal{A})$ reconnu par un

automate fini non déterministe $\mathcal{A}$ est l'ensemble des mots reconnus par $\mathcal{A}$:

$$\mathcal{L}_{rec}(\mathcal{A}) = \{w \in \Sigma^*, w \text{ est accepté par } \mathcal{A}\} \quad (1)$$

■ **Définition 4 — Langage reconnaissable par un AFND.** Un langage $\mathcal{L}$ sur un alphabet Σ est reconnaissable s'il existe un automate fini non déterministe $\mathcal{A}$ d'alphabet Σ tel que $\mathcal{L} = \mathcal{L}_{rec}(\mathcal{A})$.

D Déterminisé d'un AFND

M **Méthode 1 — Déterminisé d'un AFND** Le déterminisé d'un automate fini non déterministe $\mathcal{A} = (Q, \Sigma, Q_i, \Delta, F)$ est l'automate $\mathcal{A}_d = (\mathcal{P}(Q), \Sigma, q_i, \delta, \mathcal{F})$ défini par :

- $\mathcal{P}(Q)$ est l'ensemble des parties de Q,
- q_i est **l'ensemble des états initiaux**,
- $\forall \pi \in \mathcal{P}(Q), \forall a \in \Sigma, \delta(\pi, a) = \bigcup_{q \in \pi} \{q' \in Q, (q, a, q') \in \Delta\}$,
- $\mathcal{F} = \{\pi \in \mathcal{P}(Q), \pi \cap F \neq \emptyset\}$.

Ce qui signifie que :

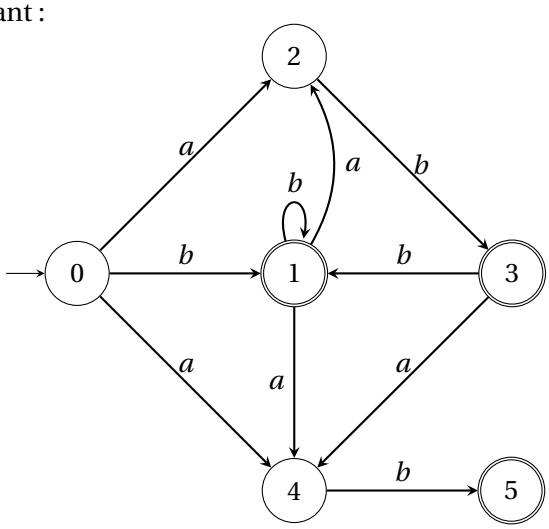
- l'état initial du déterminisé est l'état initial de l'AFND, ou bien, s'il possède plusieurs états initiaux, l'état initial constitué par la partie de tous les états initiaux de l'AFND.
- toute partie de Q est susceptible d'être un état. En pratique dans les exercices, vous construirez les états au fur et à mesure à partir de l'état initial comme dans l'exemple 1. L'ensemble des parties de Q n'a pas souvent besoin d'être explicité.
- les états accepteurs sont **les parties qui contiennent un état accepteur** de l'AFND.

L'algorithme 1 décrit cette méthode d'un point de vue opérationnel. Il s'agit de balayer un graphe en largeur et de mettre à jour les états et les transitions trouvées.

Algorithme 1 Algorithme de déterminisation d'un AFND

```
1 : Fonction DÉTERMINISER(A = (Q, Σ, Q_i, Δ, F))
2 :     transitions ← ∅
3 :     q_i ← la partition des états initiaux                    ▷ l'état initial
4 :     états ← q_i
5 :     file ← q_i
6 :     tant que file n'est pas vide répéter
7 :         q ← DÉFILER(file)
8 :         pour chaque lettre λ de l'alphabet Σ répéter
9 :             q' ← la partition des états possibles depuis (q, λ) d'après Δ
10 :            si q' n'est pas encore dans états alors
11 :                AJOUTER(états, q')
12 :                ENFILER(file,q')
13 :            AJOUTER(transitions, (q, λ, q'))
14 :    accepteurs ← ∅
15 :    pour chaque état e de états répéter
16 :        si e ∈F alors
17 :            AJOUTER(accepteurs, e)
18 :    renvoyer (états, Σ, q_i, transitions, accepteurs)
```

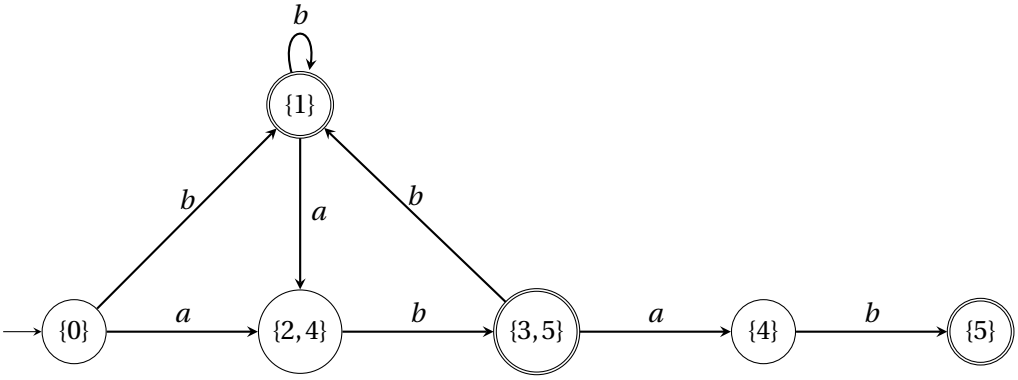
■ **Exemple 1 — Déterminiser un AFND.** On considère l'automate fini non déterministe suivant :



En construisant les parties $\mathcal{P}(Q)$ au fur et mesure à partir de l'état initial, on obtient la fonction de transition :

	$\downarrow\{0\}$	$\uparrow\{1\}$	$\{2,4\}$	$\uparrow\{3,5\}$	$\{4\}$	$\uparrow\{5\}$
a	$\{2,4\}$	$\{2,4\}$	$\emptyset$	$\{4\}$	$\emptyset$	$\emptyset$
b	$\{1\}$	$\{1\}$	$\{3,5\}$	$\{1\}$	$\{5\}$	$\emptyset$

On en déduit l'AFD suivant :



Il est possible de renommer les états arbitrairement.

Théorème 1 Un AFND $\mathcal{A}$ et son déterminisé $\mathcal{A}_d$ reconnaissent le même langage.

$$\mathcal{L}_{rec}(\mathcal{A}) = \mathcal{L}_{rec}(\mathcal{A}_d) \tag{2}$$

Démonstration. $\mathcal{A}_d$ est bien un automate fini déterministe car :

- il possède au plus $2^{|Q|}$ états et $|Q|$ est fini puisque $\mathcal{A}$ est fini.
- son état de départ est unique,

- d'après la définition de δ, s'il existait deux états tels que $\delta(\pi, a) = \pi_1$ et $\delta(\pi, a) = \pi_2$, alors on aurait : $\pi_1 = \pi_2 = \{q', (q, a, q') \in \Delta \text{ et } q \in \pi\}$, c'est-à-dire que ces deux états seraient égaux. Donc δ est bien une fonction de transition et $\mathcal{A}_d$ est déterministe.
- comme $\mathcal{A}$ possède au moins un état final, $\mathcal{F}$ n'est pas vide.

Ensuite, il nous faut montrer que $\mathcal{L}_{rec}(\mathcal{A}_d) = \mathcal{L}_{rec}(\mathcal{A})$.

Tout d'abord, dire que le mot vide ϵ est dans le langage $\mathcal{L}_{rec}(\mathcal{A})$ est équivalent à dire que $q_i \cap F \neq \emptyset$. Sur le déterminé, cela se traduit par $q_i \in \mathcal{F}$. Donc, si le mot vide appartient à l'un, il appartient à l'autre.

($\Longrightarrow$) Soit w un mot non vide sur Σ. Si $w = a_1 \dots a_n$ est accepté par $\mathcal{A}$, alors cela signifie qu'il existe une succession d'états $(q_i, q_1 \dots, q_n)$ de $\mathcal{A}$ telle que $q_n \in F$. Mais alors, comme les transitions sont traduites dans l'automate déterminisé, pour cette succession d'état de $\mathcal{A}$, on peut trouver une succession d'états $(q_i, \pi_1, \dots, \pi_m)$ de $\mathcal{A}_d$ telle que $\pi_m \in \mathcal{F}$ et que chaque q_j de la succession d'états de $\mathcal{A}$ fasse partie d'une partition π_k de la succession d'état de $\mathcal{A}_d$. Donc w est accepté par l'automate déterminisé.

($\Longleftarrow$) On procède de même dans l'autre sens. ■

R Le déterminisé d'un AFND $\mathcal{A}$ comporte donc plus d'états que $\mathcal{A}$. Dans le pire des cas, l'algorithme de détermination a une complexité exponentielle.

E ϵ-transitions

■ **Définition 5 — ϵ-transition.** Une ϵ-transition est une transition dans un automate non déterministe dont l'étiquette est le mot vide ϵ. C'est une transition spontanée d'un état à un autre.

R Une transition spontanée fait qu'un automate peut être considéré dans deux états simultanément, celui qui précède la transition et le suivant. C'est pourquoi un automate déterministe ne comporte pas de transitions spontanées.

R Les automates à transition spontanée ou automates asynchrones sont utilisés notamment par l'algorithme de Thompson pour passer d'une expression régulière à un automate. Ces transitions peuvent aussi servir à normaliser un automate. La plupart du temps on les insère dans un automate pour les éliminer par la suite.

Un exemple d'un tel automate est donné sur la figure 2. Cet automate reconnaît les mots commençant par un nombre de b quelconque suivi d'un a ou bien les mots commençant par un nombre quelconque de a suivi par b. En fait, on va voir qu'on peut très bien exprimer un tel automate de manière non déterministe sans transitions spontanées et même de manière déterministe. Les $\epsilon-$transitions n'apportent donc pas d'expressivité en plus en terme de langage.

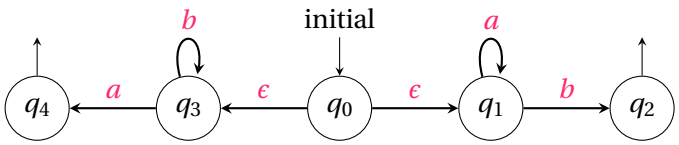


FIGURE 2 – Exemple d'automate fini non déterministe avec transition spontanée représenté sous la forme d'un graphe. On peut facilement les éliminer dans ce cas en créant plusieurs états initiaux.