# Chapitre 2

# Automates finis et langages reconnus

## 2.1 Définitions et propriétés

Un *automate fini* $Aut = < A, Q, D, F, \delta >$ est caractérisé par la donnée de cinq ensembles :

- un alphabet $A$ dit *alphabet d'entrée* ;
- un ensemble fini $Q$ dont les éléments sont appelés les *états* de l'automate ;
- l'ensemble $D \subseteq Q$ des *états de départ* (ou *initiaux*) de l'automate ;
- l'ensemble $F \subseteq Q$ des *états d'acceptation* (ou *états finals* ou *terminaux* ou *acceptants*) de l'automate ;
- l'ensemble $\delta \subseteq Q \times A \times Q$ des *transitions* de l'automate.

*Remarque.* Dans certains manuels, *Aut* est appelé automate fini sur l'alphabet $A$ et noté $< Q, D, F, \delta >$, voire $< Q, D, F >$.

Dans la pratique, en plus de la représentation ensembliste, deux représentations sont fréquemment utilisées.

### 2.1.1 Graphe d'états de l'automate

On utilise un graphe dont les sommets sont les états de l'automate et les arêtes (étiquetées par les lettres de $A$) correspondent à $\delta$.

Les états de départ sont désignés par une flèche entrante : 

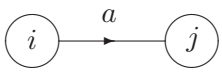
(on rencontre parfois  )

et ceux d'acceptation par une flèche sortante : 

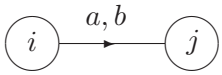
(on rencontre parfois  ou  ou  ).

Si un même état est à la fois état de départ et état d'acceptation, il est repéré par une double flèche entrante et sortante : 

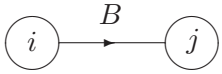
Il existe une arête étiquetée par $a$ entre l'état $i$ et l'état $j$ si et seulement si $(i, a, j) \in \delta$ (l'automate *passe* de l'état $i$ à l'état $j$ en *lisant* la lettre $a$) :




Par convention, si les états $i$ et $j$ sont liés par deux arêtes étiquetées par deux lettres différentes $a$ et $b$, on ne représente qu'une seule arête étiquetée $a, b$ :



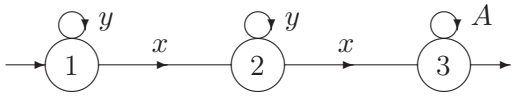

De même si, pour un sous-ensemble $B$ de $A$, $(i, x, j) \in \delta$ pour tout $x \in B$ alors on ne représente qu'une seule arête étiquetée $B$ :




*Exemple.* Le graphe qui suit représente l'automate $Aut = < A, Q, D, F, \delta >$ où

$$A = \{x, y\}, Q = \{1, 2, 3\}, D = \{1\}, F = \{3\}$$

$$\text{et } \delta = \{(1, y, 1), (1, x, 2), (2, y, 2), (2, x, 3), (3, x, 3), (3, y, 3)\}.$$




*Remarque.* Dans le cas où il y a plusieurs transitions par une même lettre depuis un même état on note, dans l'ensemble des transitions, une seule transition avec, comme état d'arrivée, l'ensemble des états atteints par toutes ces transitions. Ainsi, si l'automate contient les deux transitions $(2, y, 2)$ et $(2, y, 3)$, on notera $(2, y, \{2, 3\})$.
Par contre, si toutes les transitions partant d'un état donné aboutissent dans le même état, on peut noter, comme pour l'arête correspondante du graphe, une seule transition utilisant la lettre $A$. Dans l'exemple ci-dessus, les deux transitions $(3, x, 3)$ et $(3, y, 3)$ peuvent ainsi être notées $(3, A, 3)$.

### 2.1.2 Table de transitions

Pour décrire $\delta$, on utilise un tableau $T$ indicé par $Q$ et $A$ tel que $q' \in T[q, a]$ si et seulement si $(q, a, q') \in \delta$.

*Exemple.* La table de transitions correspondant à l'automate de l'exemple précédent vaut :

| $\delta$ | 1 | 2 | 3 |
|---|---|---|---|
| $x$ | 2 | 3 | 3 |
| $y$ | 1 | 2 | 3 |

### 2.1.3 Langages reconnaissables

L'ensemble des transitions s'étend aux mots par passages successifs d'un état à un autre en lisant chacune des lettres (de gauche à droite). Formellement, $\delta^*$ est un sous-ensemble de $Q \times A^* \times Q$ défini comme suit :

- pour tout $q \in Q, (q, \varepsilon, q) \in \delta^*$ (une telle boucle par $\varepsilon$ sur chaque état existe déjà implicitement dans $\delta$ et est utilisée chaque fois que nécessaire) ;
- si $u = a_1 a_2 \dots a_p$ $(a_i \in A, p \geq 1)$ et s'il existe $p+1$ états $q_0, q_1, \dots, q_p$ tels que, pour tout $i \in \mathbb{N}, 1 \leq i \leq p, (q_{i-1}, a_i, q_i) \in \delta$ alors $(q_0, u, q_p) \in \delta^*$ et $u$ est l'*étiquette* d'un *chemin* dans l'automate menant de l'état $q_0$ à l'état $q_p$.

*Remarque.* Un automate comporte des *$\varepsilon$-transitions* si deux états différents de cet automate peuvent être liés par une arête étiquetée par $\varepsilon$ (c'est-à-dire que l'on passe « artificiellement » d'un état à un autre sans lire aucune lettre). Dans certains manuels, les automates sont définis de telle sorte qu'ils peuvent comporter des $\varepsilon$-transitions ; les automates décrits ici n'offrent pas cette possibilité (d'après ce qui précède, les seules transitions par $\varepsilon$ sont celles, implicites, de chaque état vers lui-même). Ceci n'est pas une restriction car nous verrons par la suite (chapitre 7, section 7.4) qu'il est toujours possible de transformer un automate comportant des $\varepsilon$-transitions en un automate qui n'en comporte pas.

Si $(q_0, u, q_p) \in \delta^*$ avec $q_0 \in D$ et $q_p \in F$ alors le mot $u$ est *reconnu* (ou *accepté*) par l'automate.

Le *langage reconnu*, noté $L_{Aut}$, par un automate fini $Aut$ est l'ensemble de tous les mots reconnus par $Aut$, c'est-à-dire :

$$L_{Aut} = \{u \in A^* / \text{il existe } q \in D \text{ et } q' \in F \text{ tels que } (q, u, q') \in \delta^*\}.$$

*Exemple.* Dans l'exemple précédent,

$$L_{Aut} = y^* x y^* x A^* = \{u \in A^* / |u|_x \geq 2\} \ (= A^* x A^* x A^*).$$

*Remarque.* Le langage reconnu par l'automate $< A, Q, D, F, \delta >$ contient le mot vide si et seulement si $D \cap F \neq \emptyset$.

Un langage $L$ sur $A$ est *reconnaissable* (*rationnel*, *régulier*) s'il existe au moins un automate fini $Aut$ ayant $A$ comme alphabet d'entrée et tel que $L = L_{Aut}$.

On note $Rec(A^*)$ la famille des langages reconnaissables sur l'alphabet $A$.

## 2.2 Langage reconnu par un automate fini

*Remarque.* Nous décrivons ci-dessous plusieurs méthodes pour obtenir une expression du langage reconnu par un automate fini donné. Les expressions calculées par ces différentes méthodes seront, en général, dissemblables mais, puisqu'il y a évidemment unicité du langage reconnu par un automate fini, elles doivent être équivalentes, c'est-à-dire représenter le même ensemble de mots (bien qu'il soit parfois difficile de vérifier cette équivalence ; nous verrons au chapitre 7, section 7.3, un moyen de le faire).

### 2.2.1 Méthode intuitive

Il faut trouver un moyen de décomposer le langage recherché en une succession de langages obtenus au cours du cheminement dans l'automate.

Ainsi, dans l'exemple précédent, on va boucler un certain temps sur l'état **1** avec $y$ puis, par une transition avec $x$, on va passer dans l'état **2** et l'on ne reviendra plus dans l'état **1**. Ensuite, on fait la même chose pour le passage de l'état **2** à l'état **3**. Le langage recherché est donc le produit du langage reconnu par la boucle sur l'état **1**, d'une lettre $x$ permettant d'aller dans l'état **2**, du langage reconnu par la boucle sur l'état **2**, d'une lettre $x$ permettant d'aller dans l'état **3** et du langage reconnu par la boucle sur l'état **3**, soit $y^*xy^*xA^*$. (On remarquera que ce langage peut également s'écrire $A^*xA^*xA^*$ comme nous l'avons vu plus haut.)

*Exercices d'application.* Tous les exercices de ce chapitre sont explicitement résolus avec une méthode intuitive [1]. Nous avons en fait divisé la section d'exercices en quatre parties distinctes permettant d'illustrer différentes approches suivant la forme générale de l'automate (cette division est bien sûr arbitraire et le placement d'un automate dans telle ou telle partie n'a rien d'absolu).

Dans la première sous-section, nous considérons des automates composés de plusieurs parties indépendantes (une fois que l'on a quitté une partie, on ne peut plus y revenir).

1. Cette « méthode » est également utilisée, entre autres, pour une dizaine des automates de l'exercice 4.2 et autant de ceux de l'exercice 5.1.

Dans la deuxième, il s'agit d'automates où, pour l'essentiel, on boucle sur un état.

La troisième sous-section illustre le travail de « simplification » nécessaire avant de chercher le langage reconnu par un automate.

Dans la dernière, nous abordons des automates d'une forme plus générale pour montrer qu'une méthode de décomposition peut toujours être appliquée, même avec des automates relativement complexes.

### 2.2.2 L'algorithme de Mac Naughton et Yamada

La notion de décomposition d'un chemin menant d'un état à un autre, que nous venons de voir avec cette méthode « intuitive », se retrouve dans la méthode algorithmique présentée ci-dessous.

En effet, celle-ci repose sur le fait que l'ensemble des transitions menant d'un état $s$ à un état $t$ est, pour un état $q$ distinct de $s$ et $t$, l'union de deux ensembles de chemins :

- d'une part les chemins menant de $s$ à $t$ sans passer par $q$,
- d'autre part les chemins menant de $s$ à $t$ en passant par $q$, c'est-à-dire les chemins consistant, en partant de $s$, à aller en $q$ pour la première fois, puis boucler sur $q$ et, enfin, partir de $q$ vers $t$ sans repasser par $q$.

#### Algorithme de Mac Naughton et Yamada

Soient $Aut = < A, Q, D, F, \delta >$ un automate fini, $s$ et $t$ deux états de $Q$ et $P \subseteq Q$.

On note $W_{s,P,t}$ l'ensemble de tous les mots $u \in A^*$ tels que $u$ est étiquette d'un chemin qui mène de $s$ à $t$ et qui n'utilise pas d'autre état que ceux de $P$ comme états intermédiaires (éventuellement, il n'en utilise aucun) : $L_{Aut} = \bigcup_{s \in D, t \in F} W_{s,Q,t}$.

En partant de $W_{s,\emptyset,t}$ qui est l'ensemble des transitions directes de $s$ vers $t$, chaque ensemble $W_{s,Q,t}$ se calcule récursivement en remarquant que, si $P \subseteq Q$ contient au moins un élément $q$, $W_{s,P,t} = W_{s,P',t} \cup W_{s,P',q}.(W_{q,P',q})^*.W_{q,P',t}$ où $P' = P \setminus \{q\}$.

NB : si $W_{s,P',q}$ ou $W_{q,P',t}$ est vide alors le produit $W_{s,P',q}.(W_{q,P',q})^*.W_{q,P',t}$ est lui-même vide.
D'autre part, vu la définition que nous avons donnée de la notion de chemin dans un automate on a, pour tous $P \subseteq Q$ et $q \in Q$, $\varepsilon \in W_{q,P,q}$.

*Preuve de la validité de cet algorithme*

+ Pour tous $s, t \in Q$, $W_{s,\emptyset,t} = \{x \in A \cup \{\varepsilon\} / (s, x, t) \in \delta\}$ (notons que $\varepsilon \in W_{s,\emptyset,t}$ si et seulement si $s = t$).
D'autre part, à chaque étape de l'algorithme, le calcul d'un ensemble $W_{s,P,t}$ ($P \neq \emptyset$) se décompose en des calculs d'ensembles de la forme $W_{s',P',t'}$ où $\text{card}(P') = \text{card}(P) - 1$.

La terminaison de l'algorithme est donc bien assurée puisque, à chaque étape, le cardinal de l'ensemble des états servant au calcul décroît strictement (de 1).

+ Pour $P \neq \emptyset$, la décomposition en $W_{s,P',t} \cup W_{s,P',q}.(W_{q,P',q})^*.W_{q,P',t}$, avec $P' = P \setminus \{q\}$, assure le calcul de $W_{s,P,t}$.

En effet, soit $u$ un mot de $W_{s,P,t}$ et $u = a_1 a_2 \dots a_n$ ($a_i \in A$, $1 \leq i \leq n$) sa décomposition sur $A$.

Soit $I$ la suite d'états par lesquels passe le chemin d'étiquette $u$ menant de $s$ à $t$. Si $u = \varepsilon$ alors $I = (q_0 = s = t)$ ; si $u \neq \varepsilon$ alors $I = (q_0 = s, q_1, \dots, q_{n-1}, q_n = t)$ avec $(q_{i-1}, a_i, q_i) \in \delta$, $1 \leq i \leq n$.

- Si aucun des états de $I$ ne vaut $q$ alors $u \in W_{s,P',t}$ puisque le chemin n'utilise aucun état hormis (éventuellement) ceux de $P \setminus \{q\}$.
- Sinon, notons $I_q$ l'ensemble numéros d'ordre des états de $I$ qui valent $q$ et $inf = \min(I_q)$, $sup = \max(I_q)$ : $I_q = \{i \in \mathbb{N} / 0 \leq i \leq n, q_i = q\}$ et $q_{inf}$ (resp. $q_{sup}$) est le premier (resp. le dernier) état valant $q$ dans la suite $I$. Mais alors tous les états de $I$ précédant $q_{inf}$, tous ceux suivant $q_{sup}$ et tous ceux strictement compris entre $q_j$ et $q_{j+1}$, où $j$ et $j+1$ sont deux éléments consécutifs de $I_q$, appartiennent à $P'$.

  En posant $u = u_1 u_2 u_3$ où

  - $u_1$ est l'étiquette de la partie du chemin menant de $q_0$ à $q_{inf}$ : $u_1 \in W_{s,P',q}$
  - $u_3$ est l'étiquette de la partie du chemin menant de $q_{sup}$ à $q_n$ : $u_3 \in W_{q,P',t}$
  - $u_2$ est l'étiquette de la partie du chemin menant de $q_{inf}$ à $q_{sup}$, c'est-à-dire $u_2 = u_{21} u_{22} \cdots u_{2n}$ où chaque $u_{2i}$ est l'étiquette d'un chemin réalisant une boucle directe (donc ne passant pas plusieurs fois par $q$) sur l'état $q$, donc appartenant à $W_{q,P',q}$ : $u_2 \in (W_{q,P',q})^*$,

  on a $u \in W_{s,P',q}.(W_{q,P',q})^*.W_{q,P',t}$.

  Notons que dans tous les cas, comme $\text{card}(I_q) \geq 1$, $q_{inf}$ et $q_{sup}$ existent bien ; ils sont confondus si $\text{card}(I_q) = 1$ et alors $u_2 = \varepsilon$.

Puisque $W_{s,P',t}$ et $W_{s,P',q}.(W_{q,P',q})^*.W_{q,P',t}$ sont deux ensembles (éventuellement vides) de mots qui sont tous étiquettes de chemins menant de $s$ à $t$, on a bien prouvé que $W_{s,P,t} = W_{s,P',t} \cup W_{s,P',q}.(W_{q,P',q})^*.W_{q,P',t}$. ■

*Exemple.* L'application détaillée de l'algorithme de Mac Naughton et Yamada dans le cas de l'exemple précédent (page 24) donne le calcul suivant :

$Q = \{1, 2, 3\}$ et $L_{Aut} = W_{1,Q,3}$.

$$\begin{array}{lll} W_{1,\emptyset,1} = \{y, \varepsilon\} & W_{2,\emptyset,1} = \emptyset & W_{3,\emptyset,1} = \emptyset \\ W_{1,\emptyset,2} = \{x\} & W_{2,\emptyset,2} = \{y, \varepsilon\} & W_{3,\emptyset,2} = \emptyset \\ W_{1,\emptyset,3} = \emptyset & W_{2,\emptyset,3} = \{x\} & W_{3,\emptyset,3} = A \cup \{\varepsilon\} \end{array}$$

Nous partons de $P = Q$ et $q = 1$ : alors $P' = \{2, 3\}$, $P = P' \cup \{q\}$ (ici, on a choisi d'isoler l'état 1 mais on aurait aussi bien pu prendre les états 2 ou 3).

$$\begin{array}{rcl} W_{1,Q,3} & = & W_{1,\{2,3\},3} \cup W_{1,\{2,3\},1}.(W_{1,\{2,3\},1})^*.W_{1,\{2,3\},3} \\ & = & (W_{1,\{2,3\},1})^*.W_{1,\{2,3\},3} \end{array}$$

Maintenant, on part de $P' = \{2, 3\}$.

$q' = 2 : P'' = \{3\}, P' = P'' \cup \{q'\}$.

$$W_{1,\{2,3\},1} = W_{1,\{3\},1} \cup W_{1,\{3\},2}.(W_{2,\{3\},2})^*.W_{2,\{3\},1}$$

$q'' = 3 : P'' = \emptyset \cup \{q''\}$.

$$\begin{array}{rcll} W_{1,\{3\},1} & = & W_{1,\emptyset,1} \cup W_{1,\emptyset,3}.(W_{3,\emptyset,3})^*.W_{3,\emptyset,1} & = \{y, \varepsilon\} \\ W_{1,\{3\},2} & = & W_{1,\emptyset,2} \cup W_{1,\emptyset,3}.(W_{3,\emptyset,3})^*.W_{3,\emptyset,2} & = \{x\} \\ W_{2,\{3\},2} & = & W_{2,\emptyset,2} \cup W_{2,\emptyset,3}.(W_{3,\emptyset,3})^*.W_{3,\emptyset,2} & = \{y, \varepsilon\} \\ W_{2,\{3\},1} & = & W_{2,\emptyset,1} \cup W_{2,\emptyset,3}.(W_{3,\emptyset,3})^*.W_{3,\emptyset,1} & = \emptyset \end{array}$$

Donc $W_{1,\{2,3\},1} = W_{1,\{3\},1} = \{y, \varepsilon\}$.

$$\begin{array}{rcll} W_{1,\{2,3\},3} & = & W_{1,\{3\},3} \cup W_{1,\{3\},2}.(W_{2,\{3\},2})^*.W_{2,\{3\},3} & \\ W_{1,\{3\},3} & = & W_{1,\emptyset,3} \cup W_{1,\emptyset,3}.(W_{3,\emptyset,3})^*.W_{3,\emptyset,3} & = \emptyset \\ W_{2,\{3\},3} & = & W_{2,\emptyset,3} \cup W_{2,\emptyset,3}.(W_{3,\emptyset,3})^*.W_{3,\emptyset,3} & = \{x\} \cup xA^* = xA^* \end{array}$$

Donc $W_{1,\{2,3\},3} = W_{1,\{3\},2}.(W_{2,\{3\},2})^*.W_{2,\{3\},3} = xy^*xA^*$.

Par conséquent, $L_{Aut} = W_{1,Q,3} = y^*xy^*xA^*$.

*Exercices d'application.* Cet algorithme est utilisé page 95, ainsi que dans les solutions des exercices 2.3, 2.4, 2.5, 2.8, 2.12, 2.13, 2.14, 2.16, 2.17, 2.20, 2.21, 2.29, 2.30, 2.31, 2.34, 2.36, 4.2(2.7, 2.29, 3.11, 3.19), 5.1(3.7) et dans celle du problème 8.6.

### Technique d'élimination d'états

La réalisation directe de l'algorithme de Mac Naughton et Yamada permet de calculer le langage reconnu par un automate sans modifier celui-ci.

Une autre application du principe développé dans cet algorithme est connue sous le nom de *technique d'élimination d'états*. Il s'agit, cette fois-ci, de trans-

former l'automate initial en un automate à deux états (un de départ et un d'acceptation) reliés par une transition qui n'est plus étiquetée par une lettre ou un mot, mais par un langage reconnaissable (éventuellement réduit au mot vide, voire à l'ensemble vide). C'est donc une généralisation « naturelle » de la notion d'automate (en ce qui concerne l'ensemble des transitions) et, pour l'application de cette technique, nous appellerons sans distinction *automates*, aussi bien ces nouveaux objets que ceux précédemment décrits. Nous noterons également, pour tout couple d'états $(p, q)$ d'un tel automate, $L_{pq}$ le langage étiquette de la transition (directe) entre $p$ et $q$ (au départ, $\cup_{p,q\in Q} L_{pq} = \delta$).

Le principe est le suivant : étant donnés trois états distincts $p, q$ et $r$, et $L_{pqr}$ le langage formé de tous les mots étiquettes de tous les chemins menant de $p$ à $r$ et passant par $q$, on élimine l'état $q$ et l'on place entre $p$ et $r$ une transition étiquetée par $L_{pqr}$. On obtient ainsi un nouvel automate dont une transition est étiquetée par un langage au lieu d'une lettre, reconnaissant le même langage que le précédent et contenant un état de moins. En réitérant ce procédé, on aboutit à un automate ne contenant plus que deux états avec une transition étiquetée par un langage qui est le langage reconnu par l'automate de départ.

Formellement, la procédure se déroule en trois étapes.

Soit $Aut = < A, Q, D, F, \delta >$ un automate fini.

- On commence par ajouter deux nouveaux états $d$ et $f$.

  On relie $d$ à tous les états de départ par une transition étiquetée par le langage $\{\varepsilon\}$ (langage réduit au mot vide) et on relie tous les états d'acceptation à $f$ également par une transition étiquetée par le langage $\{\varepsilon\}$.

  Le nouvel état de départ unique est $d$ et le nouvel état d'acceptation unique est $f$.

  On a ainsi transformé $Aut$ en un nouvel automate $Aut'$ :

$$Aut' = < A, Q', \{d\}, \{f\}, \delta' >$$

$$Q' = Q \cup \{d\} \cup \{f\}$$

$$\delta' = \delta \cup \{(d, \{\varepsilon\}, p)/p \in D\} \cup \{(q, \{\varepsilon\}, f)/q \in F\}.$$

- Tant que $Q$ est non vide, on applique l'opération d'élimination d'état suivante.

  Soit $q \in Q$ ; pour tous les couples $(p, r)$ d'états de $Q' \setminus \{q\}$ tels qu'il existe une transition (directe) $(p, L_{pq}, q)$ entre $p$ et $q$ d'une part et $(q, L_{qr}, r)$ entre $q$ et $r$ d'autre part, on ajoute une transition $(p, L_{pqr}, r)$ entre $p$ et $r$ avec $L_{pqr} = L_{pr} \cup L_{pq}.L_{qq}^*.L_{qr}$ (où $L_{qq}$ est la transition de $q$ vers lui-même, directe donc n'utilisant pas d'autre état).