AUTOMATES FINIS

Dans ce chapitre, on décrit mathématiquement des machines très simples, qu'on appelle automates finis (sous-entendu : déterministes). Ce sont des machines qui ont un ensemble fini de comportements et à mémoire finie bornée.

A est un alphabet fini non vide.

1 Généralités

1.1 Définitions et vocabulaire

Définition. On appelle automate fini sur l'alphabet A tout quadruplet $\mathscr{A} = (E, i, T, \delta)$ tel que
- E est un ensemble fini, appelé ensemble des états de l'automate $\mathscr{A}$;
- i est un élément de E, appelé état initial ;
- T est un sous-ensemble de E, dont les éléments sont appelés états terminaux ;
- δ, appelé fonction de transition, est une fonction de $E \times A$ dans E :
  lorsque $\delta(p, a)$ existe, alors en notant $q = \delta(p, a)$, on dit que (p, q) est une transition de l'automate et que a est l'étiquette de la transition (on note (p, a, q) la transition).

Lorsque (p, q) est une transition d'étiquette a, on note souvent $p \xrightarrow{a} q$

La fonction de transition peut être représentée par un tableau à doubles entrées (les états, les lettres) :

par exemple, si $A = \{a, b, c\}$, et $E = \{1, 2, 3, 4\}$, le tableau

	a	b	c
1	2	3	1
2	1	1	
3		3	4
4	4	4	2

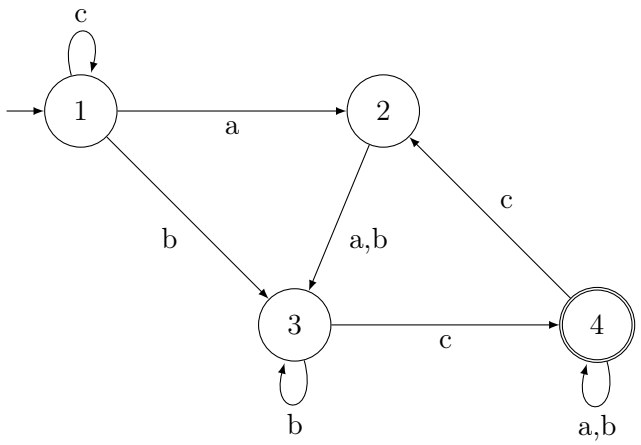
définit une fonction de transition.

Dans toute la suite du chapitre, le mot « automate » signifie de manière sous-entendue « automate fini déterministe ».

1.2 Représentation graphique

Quand le nombre d'états et de lettres n'est pas trop élevé, on peut représenter un automate par un graphe orienté valué : les sommets sont les états et les lettres sont les valeurs des arcs. Ce graphe n'est pas forcément simple (il peut y avoir des boucles et des arcs multiples), les arcs multiples sont représentés une seule fois avec une étiquette constitué de plusieurs lettres séparées par des virgules.

L'exemple ci-dessous représente l'automate $\mathscr{A} = (E, 1, \{4\}, \delta)$ précédent :



On représente souvent l'état initial par une flèche entrante et les états terminaux par un double cercle ou une flèche sortante (ce ne sont que des conventions usuelles, vous adapterez votre style à celui du sujet).

Remarque. Dans ce chapitre, on ne parle que des automates déterministes : il n'existe pas de sommet d'où partent deux flèches avec la même étiquette.

1.3 Langage reconnu par un automate

Définition. Soit $\mathscr{A} = (E, i, T, \delta)$ un automate. On définit la fonction de transition étendue δ^*, définie sur des mots de A^* :

si u est un mot et p un état, on définit inductivement $\delta^*(p, u)$:

— si $u = \varepsilon$, on pose $\delta^*(p, \varepsilon) = p$;

— si $u = va$ (a une lettre et v un mot), on pose $\delta^*(p, u) = \delta((\delta^*(p, v), a)$ quand cela a un sens ;

Dans le cas où $u = a_1 a_2 \dots a_n$ et $q = \delta^*(p, u)$ existe, alors en notant $p_0 = p$, pour tout $k \in [\![0, n-1]\!]$, $p_{k+1} = \delta(p_k, a_k)$ existe et $q = p_n$ est l'état final de la succession de transitions

$$p = p_0 \xrightarrow{a_1} p_1 \xrightarrow{a_2} p_2 \xrightarrow{a_3} \dots \xrightarrow{a_n} p_n = q$$

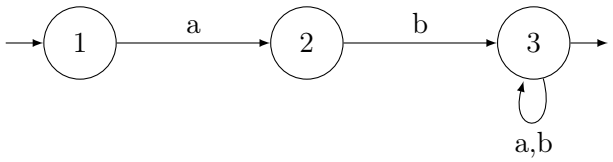
On note alors $p \xrightarrow{u} q$ et on dit que la suite des transitions est un calcul d'étiquette u dans l'automate.

Définition. Soit $\mathscr{A} = (E, i, T, \delta)$ un automate et u un mot.

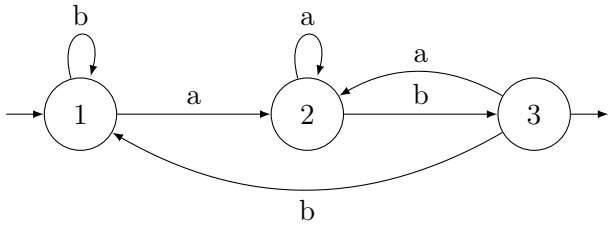
On dit que le mot est reconnu (ou accepté) par l'automate $\mathscr{A}$ si $\delta^*(i, u)$ existe et appartient à T (les états terminaux sont aussi appelés états acceptants). La suite des transitions associée est appelé un calcul réussi dans l'automate.

On appelle langage reconnu par l'automate $\mathscr{A}$ l'ensemble des mots reconnus par $\mathscr{A}$, noté en général $L(\mathscr{A})$.

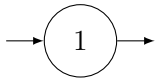
Exemples

a) L'automate 

reconnaît les mots ayant pour préfixe ab.

b) L'automate 

reconnaît les mots ayant pour suffixe ab.

c) L'automate 

reconnaît le langage $\{\varepsilon\}$.

d) Les automates 

reconnaissent le langage $\varnothing$.

En général, il existe plusieurs automates reconnaissant le même langage : on dit qu'ils sont équivalents.

1.4 Langages reconnaissables

Définition. Un langage est dit reconnaissable quand il existe un automate qui le reconnaît.

On a vu ainsi deux types de langages importants : les langages réguliers et les langages reconnaissables. L'objectif de la suite du cours est d'établir (ou presque) le théorème de Kleene.

Théorème 1 *Les langages réguliers sont reconnaissables et réciproquement.*

2 Quelques types d'automates

Il existe une grande faune d'automates. Parmi ceux-ci, quelques-uns ont des propriétés que d'autres n'ont pas et réciproquement.

2.1 Automates émondés

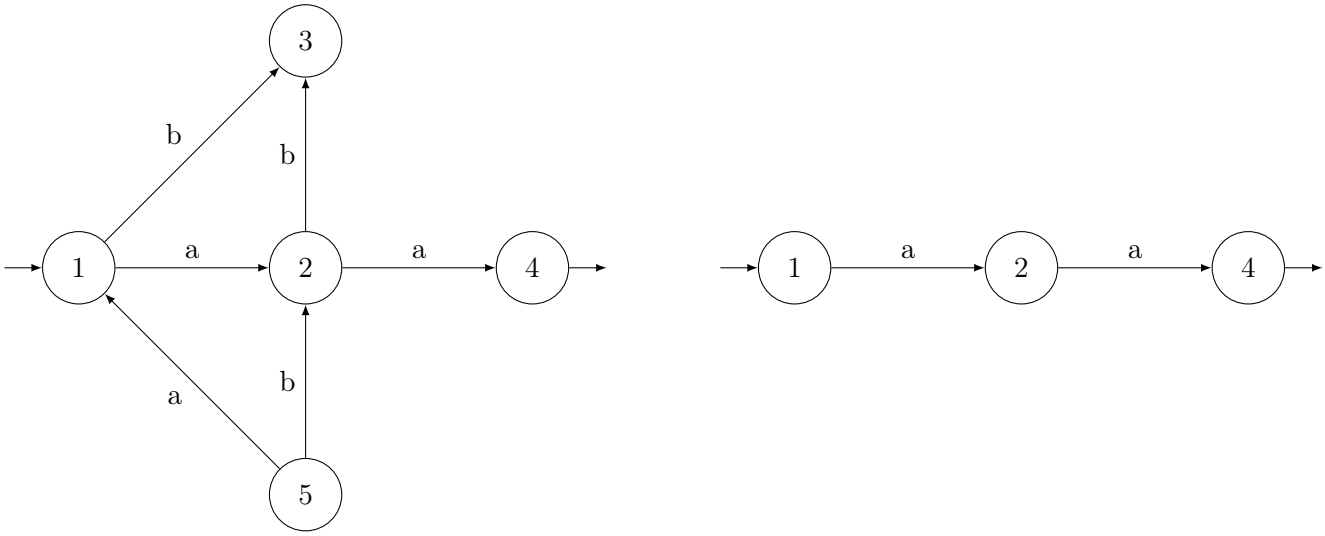
Définition. Soit $\mathscr{A} = (E, i, T, \delta)$ un automate et q un état.
On dit que q est un état accessible s'il existe un mot u tel que $\delta^*(i, u) = q$.
On dit que q est un état co-accessible s'il existe un mot u tel que $\delta^*(q, u) \in T$.
On dit que q est un état utile quand il est à la fois accessible et co-accessible.
On dit que $\mathscr{A}$ est un automate émondé quand tous ses états sont utiles.

Tous les états intermédiaires d'un calcul réussi sont utiles par définition. On montre que seuls les états utiles déterminent le langage reconnu.

Proposition 2 *Tout automate reconnaissant au moins un mot est équivalent à un automate émondé.*

Exemples

a) Les deux automates suivants sont équivalents et on voit bien sur le premier que les états 3 et 5 ne sont jamais actifs lors d'un calcul réussi.



2.2 Automates complets

Définition. Un couple $(q, a) \in E \times A$ d'un automate (E, i, T, δ) est un blocage quand $\delta(q, a)$ n'est pas défini.
Un automate est dit complet quand il ne possède pas de blocages.

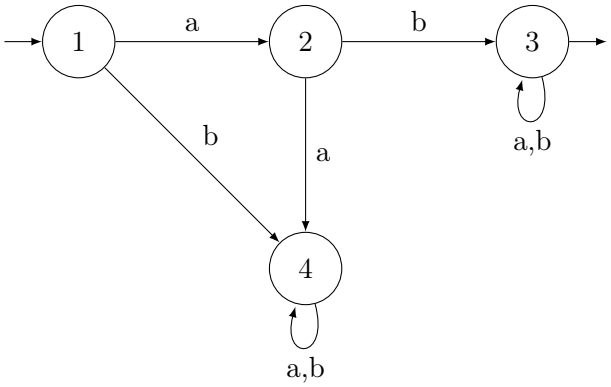
On peut faire le raisonnement inverse du précédent : on peut toujours ajouter des états inutiles jusqu'à rendre un automate complet.

Définition. Tout automate est équivalent à un automate complet.

Exemples

a) L'automate reconnaissant le suffixe *ab* présenté précédemment est complet.

b) L'automate reconnaissant le préfixe *ab* présenté précédemment n'est pas complet. Il est équivalent à l'automate complet suivant



2.3 Automates standard

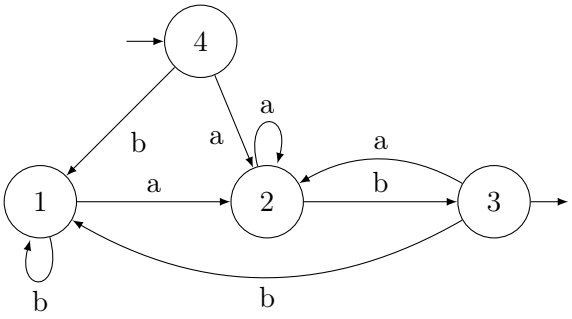
Définition. Un automate est dit standard quand aucune transition ne pointe vers l'état initial.

Encore une fois, cette contrainte peut facilement acceptée.

Proposition 3 *Tout automate est équivalent à un automate standard.*

Exemples

a) L'automate reconnaissant le suffixe ab présenté précédemment n'est pas standard. Il est équivalent à l'automate standard suivant



3 Calcul du langage reconnu par un automate

3.1 Lemme d'Arden

Proposition 4 *Soit U, V deux langages sur un alphabet A tels que $\varepsilon \notin U$.*
Alors l'équation $L = UL \cup V$ d'inconnue L (un langage sur A) a pour unique solution $L = U^*V$.

3.2 Application

Soit $\mathscr{A} = (E, i, T, \delta)$ un automate.
Pour $k \in E$, on pose $L_k = \{u \in A^* \ / \ \delta^*(k, u) \in T\}$: c'est le langage reconnu par l'automate (E, k, T, δ).
Le langage $L(\mathscr{A})$ reconnu par $\mathscr{A}$ est alors L_i.
On peut calculer $L(\mathscr{A})$ en résolvant un système d'équations qui sont souvent du type de celle du lemme d'Arden.
En effet, si k est un état, on considère les transitions qui partent de k : ce sont les $k \xrightarrow{a_j} q_j$.
Alors

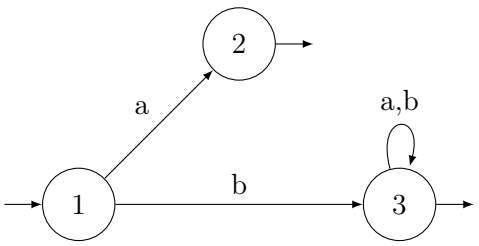
— $L_k = \bigcup_j a_j L_{q_j}$ si k n'est pas un état terminal

— $L_k = \bigcup_j a_j L_{q_j} \cup \{\varepsilon\}$ si k est un état terminal

Malheureusement, cela donne souvent une représentation obscure du langage reconnu.

Exemples

a) Avec l'automate

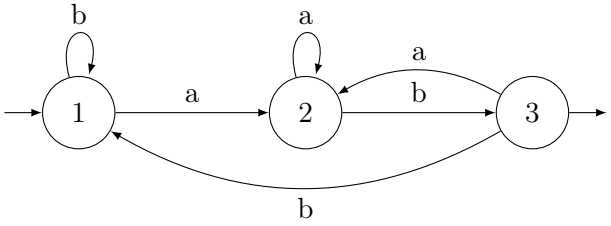


on obtient le système d'équations

$$\begin{aligned} L_3 &= \{a, b\}L_3 \cup \{\varepsilon\} \\ L_2 &= \{\varepsilon\} \\ L_1 &= bL_1 \cup aL_2 \end{aligned}$$

la première équation donne : $L_3 = \{a, b\}^*$
donc $L_1 = \{a\} \cup b\{a, b\}^*$

b) Avec l'automate déjà présenté,



on obtient le système d'équations

$$\begin{aligned} L_3 &= aL_2 \cup bL_1 \cup \{\varepsilon\} \\ L_2 &= aL_2 \cup bL_3 \\ L_1 &= bL_1 \cup aL_2 \end{aligned}$$

les deux dernières équations donnent : $L_1 = b^*aL_2$ et $L_2 = a^*bL_3$, donc $L_1 = b^*aa^*bL_3$, puis en reportant dans la première équation, on a l'égalité $L_3 = aa^*bL_3 \cup bb^*aa^*bL_3 \cup \{\varepsilon\}$, ou encore $L_3 = (a^+b \cup b^+a^+b)L_3 \cup \{\varepsilon\}$,
donc $L_3 = (a^+b \cup b^+a^+b)^* = ((\varepsilon \cup b^+)a^+b)^* = (b^*a^+b)^*$
donc $L_1 = b^*aa^*b(b^*a^+b)^* = b^*a^+b(b^*a^+b)^* = (b^*a^+b)^+$
Un peu de réflexion montre enfin que ce langage est aussi égal à $\{a, b\}^*ab$, c'est-à-dire les mots construits sur les deux lettres a et b et qui se terminent par ab.

4 Automates et algorithmes associés

Si $\mathscr{A}$ est un automate, on peut traduire en algorithme le problème de décision associé : un mot est-il reconnu par cet automate ?

On peut par exemple traduire la fonction de transition par une matrice en numérotant les états et les lettres) ou par branchements (une structure conditionnelle faite d'une succession de **if** ... **then** ... **else** ...).

En CAML, on peut aussi transformer l'automate en un certain nombre de fonctions mutuellement récursives, chacune d'elles lisant une lettre et appelant une autre selon la lettre lue (cela répartit les brachements) : l'analyseur lexical construit par *camllex* est de ce type ou presque.

On peut alors simuler un calcul dans l'automate et vérifier s'il est réussi, ce qui permet de reconnaître ou non un mot.