

## Chapitre III : Automates Finis

Goulven GUILLOU

UBO – Département Informatique



1 / 24

La théorie des automates est issue de plusieurs courants scientifiques :

- la logique (Church, Kleene, Turing).
- les systèmes dynamiques discrets (Morse, Conway).
- la théorie de l'information (Shannon) et le codage (Schützenberger, Huffman).
- la linguistique (Chomsky).
- les circuits électroniques (Shannon, Huffman).

Nous verrons les automates comme des modèles de machines. Les automates finis sont des cas particuliers de machines de Turing (1936) qui sont des modèles des ordinateurs d'aujourd'hui. Les automates et les expressions rationnelles interviennent dans de nombreux logiciels.

2 / 24

Les informations sont souvent représentées par des chaînes de caractères :

- en informatique suite de 0 et de 1.
- en génétique suite de A, C, G et de T.
- en français se sont les mots figurant dans le dictionnaire.

Formalisation commune : un *alphabet* est un **ensemble fini** de *symboles* appelés aussi *caractères* ou *lettres*. Ainsi on a :

- l'alphabet binaire  $\{0, 1\}$ .
- l'alphabet du génôme  $\{A, C, G, T\}$ .
- l'alphabet latin usuel  $\{a, b, \dots, z, A, \dots, Z\}$ .

Les alphabets que nous utiliserons seront assez petits tels que  $\{a, b\}$  ou  $\{a, b, c\}$  et seront notés  $A$  ou  $\Sigma$ .

3 / 24

Un *mot* sur un alphabet  $A$  est une suite **finie** de lettres de  $A$ .

On le note par simple juxtaposition, *abracadabra* est un mot sur l'alphabet  $\{a, b, c, d, r\}$ .

La *longueur* d'un mot  $m$  notée  $|m|$  est égale au nombre de lettres figurant dans  $m$  ( $|abracadabra| = 11$ ).

La *concaténation* de deux mots est le mot obtenu en mettant bout à bout ces deux mots.

On note  $u^n$  la concaténation de  $n$  mots égaux à  $u$ .  $(ab)^3 = ababab$ .

Il existe un mot de longueur 0, le *mot vide* noté  $\varepsilon$ .  $\varepsilon$  est l'élément neutre de la concaténation.

Un mot est donc la concaténation d'une suite **finie** de caractères.  $\varepsilon$  est le mot obtenu par la concaténation d'une suite de caractères vide.

4 / 24

Un *langage* est un ensemble (non nécessairement fini) de mots.

$\{aba, babaa, bb\}$ ,  $\{a^n b^n | n \geq 0\}$  sont des langages sur l'alphabet  $\{a, b\}$  mais aussi  $\{a\}$  ou  $\emptyset$  !

On note  $A^*$  l'ensemble des mots sur l'alphabet  $A$ .

$A^*$  est infini dès que  $A$  est non vide.

En effet  $\{a\}^* = \{\varepsilon, a, aa, aaa, aaaa, \dots\}$ .

Un automate déterministe est une machine qui lit une suite de caractères c'est-à-dire un mot.

Avant de lire le premier caractère l'automate est dans un certain *état* : l'*état initial*.

En fonction du caractère lu et de son "programme", l'automate change d'état.

Dans ce nouvel état, il lit le second caractère.

En fonction de ce nouveau caractère et de son "programme", l'automate change d'état ...

Et ainsi de suite jusqu'à la fin du mot.

Un automate est une machine qui *consomme* un mot. A la fin du mot l'automate se trouve dans un certain état.

Un automate peut être vu comme une machine qui *reconnaît* un ensemble de mots.

Pour cela, il existe deux types d'états :

- les états acceptants (accepteurs) ou finaux (finaux).
- les états non-acceptants (non-accepteurs).

Si, quand il a fini de lire le mot, l'automate se trouve dans un état acceptant, le mot est dit *accepté* ou *reconnu* par l'automate, sinon il est dit *non-accepté* ou *refusé*.

Les automates sont souvent représentés à l'aide de *graphes*.

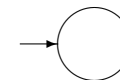
- Les états sont matérialisés par des ronds :



pour les états non-acceptants.

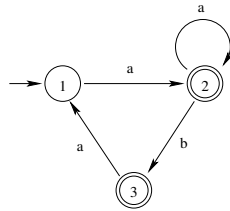


pour les états finaux.



pour l'état initial.

- La *transition* (une étape du programme) entre deux états sur lecture d'un caractère est symbolisé par un *arc orienté étiqueté* par le caractère  $\xrightarrow{a}$  placé entre les deux états concernés.



Trois états 1, 2 et 3. C'est un automate *fini*

1 est l'état initial.

2 et 3 sont les états finaux.

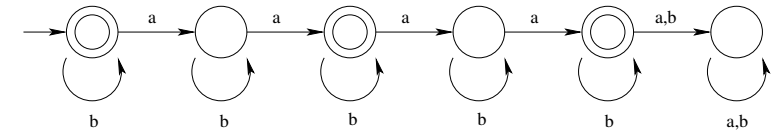
De chaque état sort au plus une flèche d'étiquette donnée : il est *déterministe*.

Il est *incomplet* car aucune flèche partant de l'état 1 n'est étiquetée par *b*.

*aab* est accepté.

*aba* est refusé.

*abb* est refusé car il provoque le *blocage* de l'automate.



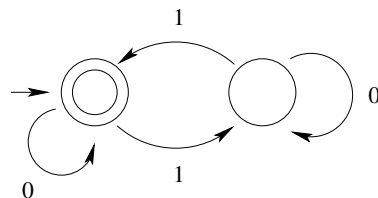
L'ensemble des mots reconnu par un automate est le *langage* de l'automate.

On dit qu'un langage est *reconnaisable* s'il existe un automate fini déterministe qui le reconnaît.

reconnait les mots sur  $\{a, b\}$  qui comportent 0, 2, ou 4 *a*.

Il s'agit d'un automate fini déterministe et *complet*.

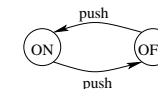
Quels sont les mots sur  $\{0, 1\}$  acceptés par l'automate représenté par le *graphe de transition* suivant ?



Ce sont les mots sur  $\{0, 1\}$  qui contiennent un nombre pair de 1.

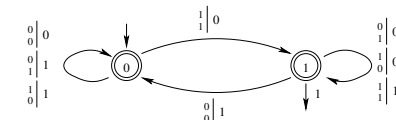
Il s'agit encore d'un automate fini déterministe et complet.

- Système dynamique discret (pas d'état accepteur).



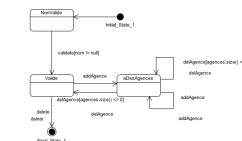
Un bouton poussoir par exemple !

- Transducteur (lecture et écriture).



Automate additionneur.

- Diagramme d'états UML.



Soit un alphabet  $\Sigma$ . On définit les opérations suivantes :

- Union :  $L_1 \cup L_2 = L_1 + L_2 = \{m \mid m \in L_1 \text{ ou } m \in L_2\}$ .
- Intersection :  $L_1 \cap L_2 = \{m \mid m \in L_1 \text{ et } m \in L_2\}$ .
- Complémentaire (dans  $\Sigma^*$ ) :  $\bar{L} = \{m \in \Sigma^* \mid m \notin L\}$ .
- Différence :  $L_1 \setminus L_2 = \{m \in L_1 \mid m \notin L_2\}$ .
- Concaténation (produit) :  
 $L_1.L_2 = L_1L_2 = \{m_1m_2 \mid m_1 \in L_1 \text{ et } m_2 \in L_2\}$ .
- Nième itérée :  $L^n = L.L^{n-1} = L^{n-1}.L$  pour  $n > 0$  et  $L^0 = \{\varepsilon\}$ .
- Etoile de Kleene (itération) :  $L^* = \bigcup_{i=0}^{+\infty} L^i$ .
- Itération stricte :  $L^+ = \bigcup_{i=1}^{+\infty} L^i = L.L^*$ .

Pour simplifier les écritures, on fait les abus suivants pour obtenir ce qu'on appelle des *expressions régulières*.

- $\{a\} = a$
- $\{a, b\} = a + b$
- $\{a\}.\{a, b\}^*.\{c\} + \{cc\} = a(a+b)^*(c+cc) = a(a+b)^*c(\varepsilon+c)$
- $\{a\} = \bar{a}$  (Complémentaire de  $a$  dans l'alphabet)

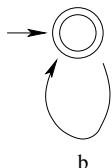
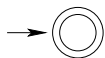
On utilisera indifféremment plusieurs notations

$\{aab, aba, abba\} = aab + aba + abba$ .

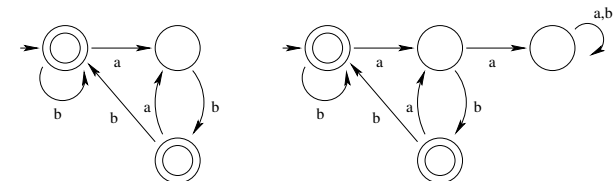
Par ordre de priorité décroissante on a :  $*, ^n$  (*exponentiation*),  $., +$

Si on peut deviner le langage reconnu par un automate, on peut également chercher à construire l'automate qui reconnaît un langage donné.

Construire un automate fini, déterministe et complet qui reconnaît les mots sur  $\{a, b\}$  tels que après un  $a$  il y ait toujours au moins un  $b$ .



Pour un langage donné il est possible de construire plusieurs automates le reconnaissant.

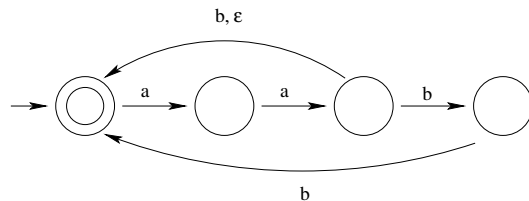


Automates équivalents.

Deux automates reconnaissant le même langage sont dits *équivalents*.

Montrer en exercice que l'équivalence entre automates est bien une relation d'équivalence.

Souvent plus facile à construire qu'un automate déterministe.  
Automate ayant pour langage  $(aab + aa + aabb)^*$  :



Plusieurs *chemins* possibles pour  $aab$ .

Pour que le mot soit reconnu, il suffit d'exhiber au moins un chemin menant à un état accepteur.

Le langage reconnu par l'automate est l'ensemble des mots acceptés par l'automate.

La notion d'équivalence s'étend à l'ensemble des automates déterministes et indéterministes.

Essayer en exercice de construire un automate déterministe reconnaissant  $(aab + aa + aabb)^*$ .

17 / 24

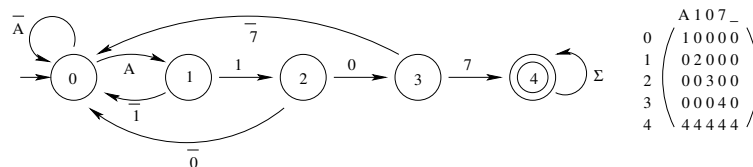
Un automate fini est un quintuplet  $(\Sigma, Q, q_0, F, \delta)$  avec :

- $\Sigma$  l'alphabet d'entrée.
- $Q$  l'ensemble non vide des *états*.
- $q_0 \in Q$  l'*état initial*.
- $F \subseteq Q$  l'ensemble des *états finaux*.
- $\delta : (\Sigma \cup \{\varepsilon\}) \times Q \longrightarrow \mathcal{P}(Q)$  la *fonction de transition*.

Remarque : un automate déterministe (pas d' $\varepsilon$ -transition et au plus une transition pour chaque couple (état,entrée)) est un cas particulier d'automate indéterministe.

18 / 24

La fonction de transition est souvent représentée par une *matrice de transition*. C'est le *programme* de l'automate.



Digicode A107

En cas de blocage, on met  $-1$  par exemple.

Si l'automate est indéterministe, on met des ensembles d'états à la place des états et on ajoute  $\varepsilon$  comme caractère possible.

19 / 24

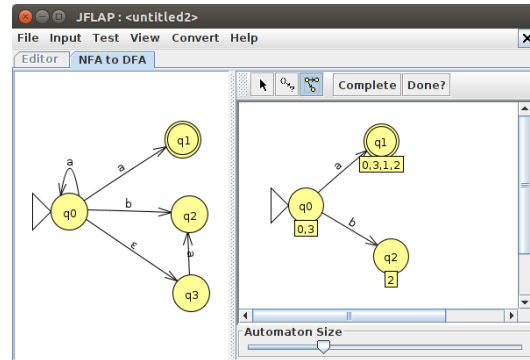
**Tout automate fini est équivalent à un automate fini déterministe.**

Conséquence :

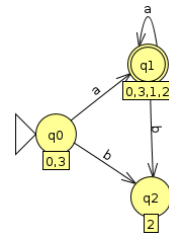
On ne se préoccupe plus d'avoir un automate déterministe ou non.  
La preuve du théorème nous donne un algorithme de détermination.

20 / 24

On regroupe en un ensemble d'états tous les états atteints par la lecture d'un même caractère.



Première étape



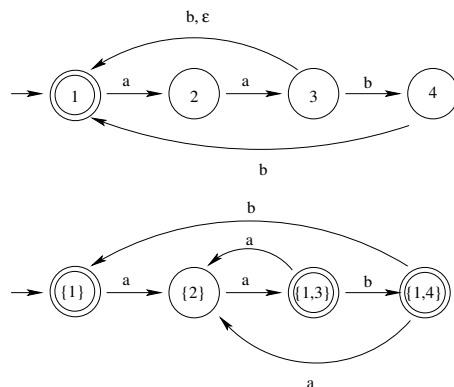
Deuxième étape

### Définition

Pour un état  $q$  d'un automate  $M$ ,  $E(q)$  est l'ensemble des états qui peuvent être atteints à partir de  $q$  par une suite de transitions sur le mot vide.

On définit l'automate  $M' = (\Sigma, Q', q'_0, F', \delta')$  équivalent à  $M = (\Sigma, Q, q_0, F, \delta)$  par :

- $Q' = \mathcal{P}(Q)$ .
- $q'_0 = E(q_0)$
- $\delta'(q, a) = \bigcup \{E(p) \mid \exists q_1 \in q \text{ tq } p \in \delta(q_1, a)\}$
- un état de l'automate  $M'$  sera accepteur s'il contient un état accepteur de  $M$ .



Rappel : un langage  $L$  est reconnaissable s'il existe un automate fini (déterministe ou non) le reconnaissant.

On note  $\text{Rec}(\Sigma^*)$  l'ensemble des langages reconnaissables sur  $\Sigma$ .

- **propriété 1** : les langages finis sont reconnaissables.
- **propriété 2** :  $\text{Rec}(\Sigma^*)$  est fermé par union.
- **propriété 3** :  $\text{Rec}(\Sigma^*)$  est fermé par complément.
- **propriété 4** :  $\text{Rec}(\Sigma^*)$  est fermé par concaténation.
- **propriété 5** :  $\text{Rec}(\Sigma^*)$  est fermé par étoile de Kleene.
- **propriété 6** :  $\text{Rec}(\Sigma^*)$  est fermé par intersection.