# 11. Automates finis



## Langages réguliers

- Les langages réguliers sont les langages les plus simples. Ils sont néanmoins très utilisés en informatique.
- Ils sont obtenus à partir des langages finis en effectuant la fermeture par les opérations d'union, de concaténation et étoile.
- Ils coïncident avec l'ensemble des langages décrits par les expressions régulières.
- Il existe un moyen effectif pour reconnaître les éléments d'un langage régulier donné.



## Automates finis

- Les automates finis sont des «machines abstraites» qui savent décider de l'appartenance d'un mot à un langage régulier donné.
- Ces machines abstraites constituent un modèle théorique de référence.
  Dans la pratique, nombreuses sont les applications qui implémentent la notion d'automates finis ou ses variantes (cela va du compilateur ... à la machine à café).
- Un automate «lit» un mot écrit sur son ruban d'entrée.
  Il part d'un état initial et à chaque lettre lue, il change d'état. Si, à la fin du mot, il est dans un état final, on dit qu'il reconnaît le mot lu.



## Automate fini

Un automate fini $A$ est la donnée d'un quintuplet $(\Sigma, Q, \delta, q_0, F)$ tel que :

- $\Sigma$ est un alphabet
- $Q$ est un ensemble *fini* d'états
- $\delta$ est un ensemble de règles de transition
  $\delta \subseteq (Q \times (\Sigma \cup \{\varepsilon\}) \times Q)$
- $q_0$ est l'état initial
- $F$ est un sous-ensemble de $Q$ appelé l'ensemble des états finals

## Représentation

- L'étiquetage d'un graphe est une application de l'ensemble de ses arêtes/arcs dans un ensemble d'étiquettes quelconque.
- On associe à un automate fini (□, Q, □, $q_0$, F) un graphe* orienté et étiqueté G = (S, A) ainsi défini :
  - les sommets de S sont exactement les états de Q
  - à chaque triplet (q, a, q') de □ on associe un arc (q, q') étiqueté par a.
- Enfin, sur le schéma, on apposera un signe distinctif sur l'état initial et sur les états finals.

* Précision : c'est en fait un multigraphe, car entre deux sommets q et q', il peut y avoir plusieurs arcs (q,q').



## Reconnaissance

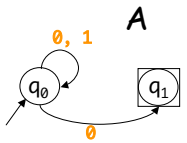
- Un mot m est reconnu ou accepté par un automate fini s'il existe un chemin étiqueté par le mot m menant de l'état initial $q_0$ à un état final de F.
- Le chemin est alors une suite d'arcs étiquetés $((k_i, a_i, k_{i+1}))$ pour i de 1 à $|m|$.
  Le mot m est égal à la concaténation des étiquettes: $a_1 \ldots a_{|m|}$.
  $k_1$ est égal à l'état initial $q_0$, $k_{|m|+1}$ est un état final.
- L'ensemble des mots acceptés par un automate fini *A* forme le langage reconnu par cet automate.
  On le note : L(*A*)



## Exemple

Automate fini

*A* = (□, Q, □, $q_0$, F)




□ = {0,1}
Q = {$q_0$,$q_1$}
□ = {($q_0$, 0, $q_0$ ), ( $q_0$, 0, $q_1$ ), ( $q_0$, 1, $q_0$ ) }
$q_0$ est l'état initial
F = {$q_1$}

*A* accepte un mot m s'il existe un chemin de $q_0$ à un état de F étiqueté par les lettres de m.

Ici, *A* reconnaît le langage L(*A*) décrit par l'expression régulière : (0+1)*0



## Automates finis déterministes

- Un automate fini est déterministe si et seulement si la relation □ est une fonction de transition telle que :

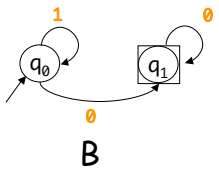
  □ : Q □ □ □ Q

- On ne peut plus effectuer de transition sur □.
- D'un état donné, il part au plus un seul arc étiqueté par une lettre donnée.
- En informatique, le déterminisme est le fait de ne jamais avoir le choix entre plusieurs exécutions.

## Exemple

Automate fini déterministe

$B = (\Sigma, Q, \delta, q_0, F)$



B

$\Sigma = \{0,1\}$
$Q = \{q_0, q_1\}$
$\delta = \{(q_0,0,q_1), (q_0,1,q_0), (q_1,0,q_1)\}$
$q_0$ est l'état initial
$F = \{q_1\}$

B accepte les mots du langage L(B) décrit par l'expression régulière :

$1^*00^*$



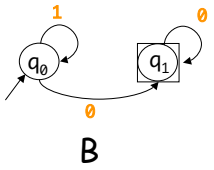
## Table de transition

Pour un automate fini déterministe, on peut écrire la fonction de transition sous la forme suivante :

$\delta$ : $(q_0,0) \to q_1$
$(q_0,1) \to q_0$
$(q_1,0) \to q_1$

ou encore, sous la forme d'une table de transition :

| | 0 | 1 |
|---|---|---|
| $q_0$ | $q_1$ | $q_0$ |
| $q_1$ | $q_1$ | - |



B



## Automates finis complets

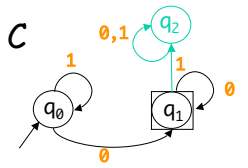
- Un automate fini déterministe est complet si et seulement si $\delta$ est une fonction totale sur $Q \times \Sigma$.
- De chaque état, il part alors exactement un arc étiqueté par chacune des lettres de l'alphabet $\Sigma$.
- Quand la fonction n'est pas totale, l'automate fini peut se trouver bloqué lors de la lecture d'un mot.
- Un automate fini complet ne sera jamais bloqué, quitte à contenir des états surperflus : les états-poubelles.



## Exemple

Automate fini complet

$C = (\Sigma, Q, \delta, q_0, F)$




$\Sigma = \{0,1\}$
$Q = \{q_0, q_1, q_2\}$
$\delta = \{(q_0,0,q_1), (q_0,1,q_0), (q_1,0,q_1), (q_1,1,q_2), (q_2,0,q_2), (q_2,1,q_2)\}$
$q_0$ est l'état initial
$F = \{q_1\}$

C accepte les mots du langage L(C) = L(B) (décrit par $1^*00^*$).
L'état $q_2$ complète l'automate B en l'automate C.

## Non-déterminisme

- Dans un automate fini non-déterministe, il peut y avoir le choix entre plusieurs chemins lors de la lecture d'un mot.
- Pour qu'un mot soit accepté, il suffit que ses lettres étiquettent un chemin allant d'un état initial à un état final.
- Attention !
  la lecture d'un mot dans un automate non-déterministe n'est pas forcément unique.
- Ainsi, pour un mot accepté par un automate, il peut exister d'autres chemins issus de $q_0$ mais ne menant pas à un état final.
- Notez que les transitions peuvent être étiquetées par le mot vide $\square$, ce qui serait exclu dans un automate fini déterministe.



## Déterminisme / non-déterminisme

Théorème : si un langage est reconnu par un automate fini, alors il est également reconnu par un automate fini déterministe.

- si l'automate fini du départ **A** est déterministe, c'est évident
- si l'automate de départ n'est pas déterministe, on se propose de construire un automate fini déterministe B qui intègre tous les choix existant dans l'automate de départ par l'algorithme de déterminisation.
- il resterait à prouver formellement que le nouvel automate B accepte exactement les mots acceptés par **A**.



## Algorithme de déterminisation
(version sans $\square$-transition)

Soit $A = (\square, Q, \square, q_0, F)$ un automate fini non-déterministe sans $\square$-transition, on construit l'automate fini déterministe $B = (\square, Q', \square', q_0', F')$
Q' sera alors un sous-ensemble de P(Q) :

- $\square' \leftarrow \square \varnothing$
- $q_0' \leftarrow \square \{q_0\}$
- $Q' \leftarrow \square \{q_0'\}$

- **pour tout** état q' de Q' non encore considéré **faire**
  **pour toute** lettre $\square$ de $\square$ **faire**
  $q'' \leftarrow \square \{ y$ / il existe $x \square q'$ et $y \square Q$ tels que $(x,\square,y) \square \square \}$
  $\square' \leftarrow \square' \square \{ (q', \square, q'') \}$
  $Q' \leftarrow \square Q' \square \{q''\}$

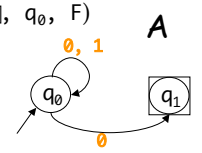
- $F' \leftarrow \square \{q' \square Q'$ tels que $q' \square F \square \varnothing \}$



## Exemple

- Automate fini non déterministe : $A = (\square, Q, \square, q_0, F)$

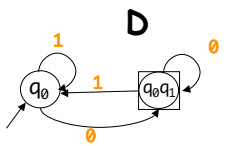
$\square = \{0,1\}$
$Q = \{q_0,q_1\}$
$\square = \{(q_0, 0, q_0), (q_0, 0, q_1), (q_0, 1, q_0)\}$
$q_0$ est l'état initial
$F = \{ q_1 \}$




- On construit **D** $= (\square, Q', \square', q_0', F')$

$q_0' = \{ q_0 \}$ est l'état initial
$Q' = \{\{q_0\}, \{q_0,q_1\}\}$
$\square' = \{(\{ q_0 \},1, \{q_0\}), (\{ q_0 \},0, \{q_0,q_1\}), (\{ q_0,q_1 \},1, \{q_0\}), (\{ q_0,q_1 \},0, \{q_0,q_1\})\}$
$F = \{\{ q_0,q_1\} \}$

| | 0 | 1 |
|---|---|---|
| $q_0$ | $q_0,q_1$ | $q_0$ |
| $q_0,q_1$ | $q_0,q_1$ | $q_0$ |



## Algorithme de déterminisation

$A = (\Sigma, Q, \delta, q_0, F)$ automate fini non-déterministe,
on construit l'automate fini déterministe $B = (\Sigma, Q', \delta', q_0', F')$
*(en vert le traitement des $\varepsilon$-transitions)* :

- $\delta'$ $\Sigma$ $\varnothing$
- $q_0'$ $\Sigma$ $\{q_0\}$ $\Sigma$ { les états q tels que $(q_0, \varepsilon, q) \in \delta$ }
- $Q'$ $\Sigma$ $\{q_0'\}$

- **pour tout** état q' de Q' non encore considéré **faire**
  **pour toute** lettre $\sigma$ de $\Sigma$ **faire**
    q" $\Sigma$ { y / il existe x $\in$ q', $(x, \sigma, y) \in \delta$ }
    **si** q" $\neq \varnothing$ **alors**
      q" $\Sigma$ q" $\cup$ {z / il existe y $\in$ q" et $(y, \varepsilon, z) \in \delta$}
    $\delta'$ $\Sigma$ $\delta'$ $\cup$ { (q', $\sigma$, q") }
    Q' $\Sigma$ Q' $\cup$ {q"}

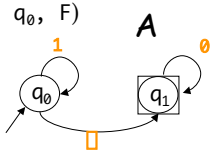
- F' $\Sigma$ {q' $\in$ Q' tels que q' $\cap$ F $\neq \varnothing$ }



## Exemple

- Automate fini non déterministe : $A = (\Sigma, Q, \delta, q_0, F)$

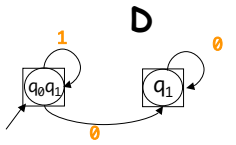
$\Sigma = \{0,1\}$
$Q = \{q_0, q_1\}$
$\delta = \{(q_0, 1, q_0), (q_0, \varepsilon, q_1), (q_1, 0, q_1)\}$
$q_0$ est l'état initial
$F = \{ q_1 \}$




- On construit **D** $= (\Sigma, Q', \delta', q_0', F')$

$q_0' = \{ q_0q_1 \}$ est l'état initial
$Q' = \{ \{q_0q_1\}, \{q_1\} \}$
$\delta' = \{(\{ q_0,q_1 \},0, \{q_1 \}), (\{ q_0,q_1 \},1, \{q_0,q_1 \}), (\{ q_1 \},0, \{q_1 \}) \}$
$F = \{ \{q_0,q_1\}, \{q_1\} \}$

| | 0 | 1 |
|---|---|---|
| $q_0q_1$ | $q_1$ | $q_0q_1$ |
| $q_1$ | $q_1$ | - |






## Théorème de Kleene*

Théorème — un langage sur un alphabet $\Sigma$ est régulier si et seulement si il est reconnu par un automate fini.

Esquisse de la démonstration :

- On sait construire un automate pour chaque élément de la base. Pour chacune des 3 opérations, on sait construire un automate lui correspondant à partir des automates reconnaissant les langages de départ.
- Réciproquement, on peut passer d'un automate fini à un langage régulier (admis pour cette année).

(B)
$\varnothing \in R$
$\{\varepsilon\} \in R$
$\{a\} \in R$, pour tout $a \in \Sigma$
(I)
si L, M $\in$ R alors
L $\cup$ M $\in$ R
L.M $\in$ R
$L^* \in R$

**rappel définition inductive des langages réguliers**

* Stephen Kleene, 1909-1994, logicien américain



## Preuve 1er sens

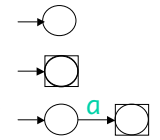
Un langage régulier est reconnu par un automate fini (non-déterministe)

Preuve par induction
*on part de la définition inductive des E.R. plutôt que de celle des L.R.*

➔ L est régulier donc il existe une expression régulière E qui le décrit.
➔ Pour les éléments de la base :
quels sont les automates finis correspondant ?

(B)
$\varnothing \in ER$
$\varepsilon \in ER$
$a \in ER$, pour tout $a \in \Sigma$
(I) ...

Définition inductive des expressions régulières



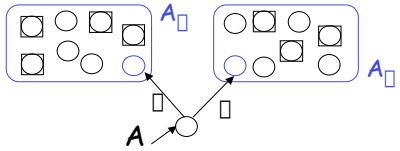
## Suite preuve (opération +)

- Hypothèse d'induction :
  il existe deux automates finis $A_\alpha$ et $A_\beta$ tels que $L(A_\alpha)$ est le langage décrit par $\alpha$ et $L(A_\beta)$ celui décrit par $\beta$.
- Peut-on construire un automate $A$ reconnaissant le langage décrit par $(\alpha + \beta)$?

(B) ...

(I) si $\alpha, \beta \in$ ER alors
$(\alpha + \beta) \in$ ER
...

Définition inductive
des expressions régulières




- $A$ reconnaît le langage décrit par $(\alpha + \beta)$



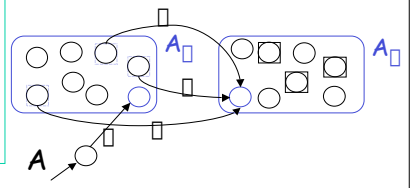
## Suite preuve (opération .)

- peut-on construire un automate $A$ pour reconnaître le langage décrit par $(\alpha.\beta)$ ?

(B) ...

(I) ...
si $\alpha, \beta \in$ ER alors
$(\alpha . \beta) \in$ ER
...

Définition inductive
des expressions régulières




- $A$ reconnaît le langage décrit par $(\alpha.\beta)$



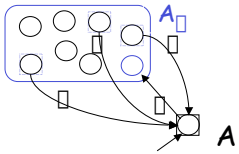
## Suite preuve (opération *)

- Peut-on reconnaître le langage décrit par $\alpha^*$ ?

(B) ...

(I) ...
...
si $\alpha \in$ ER alors
$(\alpha)^* \in$ ER

Définition inductive
des expressions régulières




- $A$ reconnaît le langage décrit par $(\alpha)^*$