Calculabilité - Décidabilité (ICC)

Cours n°1

Stef Graillat

Sorbonne Université

Format du cours

Responsable du cours : Stef Graillat (bureau 26-00/313)
`stef.graillat@sorbonne-universite.fr`

Évaluation des connaissances

- 3 ou 4 interrogations en cours/TD

Horaire

- Cours : le lundi de 13h45 à 15h45, salle (voir sur Ypareo toutes les semaines)
- TD : le vendredi de 13h45 à 15h45, salle (voir sur Ypareo toutes les semaines) les 20/09, 11/10, 25/10, 15/11, 29/11

Site web :
`http://www-pequan.lip6.fr/~graillat/teach/icc/index.html`

Objectifs du cours

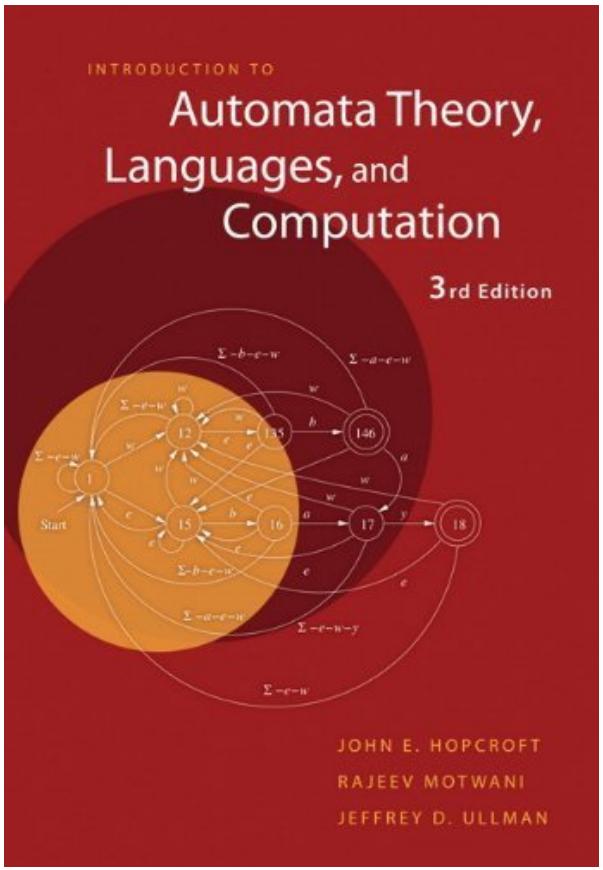
Objectifs :

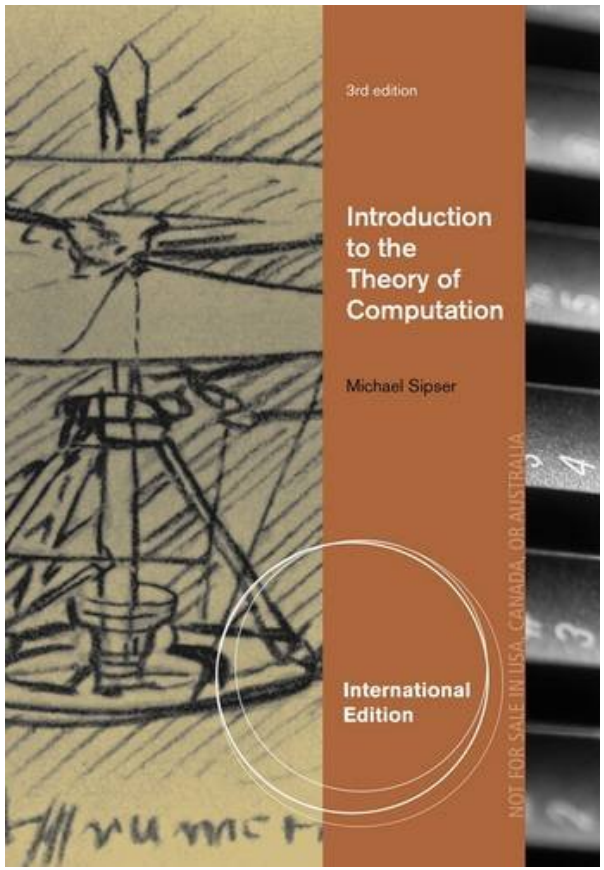
- automate, machine de Turing : quels sont les modèles simples d'ordinateur ?
- calculabilité : quels sont les problèmes que l'on peut résoudre avec un ordinateur ?
- complexité : qu'est-ce qui fait que certains problèmes sont durs à résoudre et d'autres faciles ?

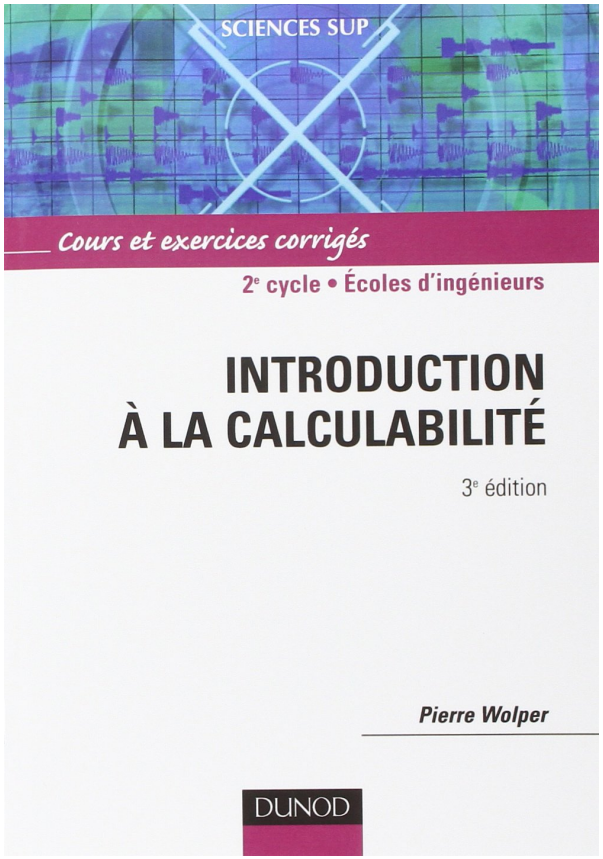
Références principales

- **Introduction to automata theory, languages and computation ; Hopcroft, Motwani, Ullman ; 3e édition ; Addison-Wesley ; 2006**
- Introduction to the theory of computation ; Sipser ; 3e édition ; Cengage Learning ; 2012
- Introduction à la calculabilité ; Wolper ; 3e édition ; Dunod ; 2006
- Langages formels, calculabilité et complexité ; Carton ; Vuibert ; 2014
- An introduction to formal languages and automata ; Linz ; 6e édition ; Jones & Bartlett Learning ; 2016
- Formal language, a practical introduction ; Adam Webber ; Franklin, Beedle & Associates ; 2008
- Automata and Computability, A Programmer's Perspective ; Ganesh Gopalakrishnan ; CRC Press ; 2019

Références principales



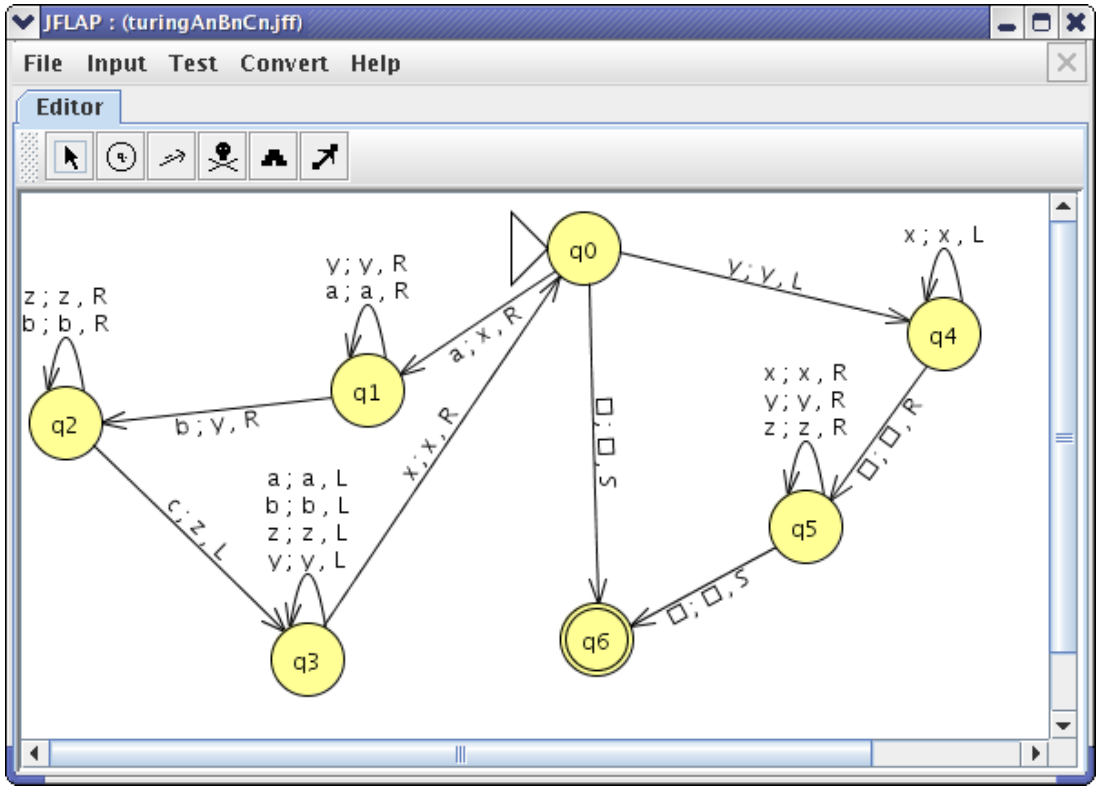




Outil pour la simulation

JFLAP

http://www.jflap.org/



Champs d'application

Les notions vues dans ce cours interviennent dans :

- Les parsers, pattern matching (filtrage par motif)
- La gestion des expressions régulières (voir `emacs`, `grep`)
- La compilation
- La preuve de logiciels
- Les questions de complexité (temps, espace, bornes inférieures)

Plan général du cours

Le cours est divisé en 5 parties :

1. Automates finis et Langages réguliers
2. Automates à piles, Grammaires hors-contexte, Langages hors-contexte
3. Machines de Turing
4. Décidabilité – Indécidabilité
5. Classes de complexité

Partie I : Automates finis et Langages réguliers

Concepts centraux : Alphabets – Mots

Un alphabet est un ensemble *fini* (non vide) de symboles.

En général, il est noté Σ ou $\mathcal{A}$.

- $\Sigma = \{0, 1\}$ l'alphabet binaire
- $\Sigma = \{a, b, \ldots, z\}$ l'ensemble des lettres minuscules
- L'ensemble des caractères ASCII.

Un mot est une suite *finie* de symboles choisis dans un alphabet.

0011 est un mot construit à partir de l'alphabet binaire.
Le mot vide : il sera noté ε.

Longueur d'un mot : nombre de positions dans le mot.

$|w|$ représente la longueur du mot w
$|0011| = 4$, $|\varepsilon| = 0$

Concepts centraux : Langages

Puissance d'un alphabet : si Σ est un alphabet et $k \in \mathbb{N}$, on note Σ^k l'ensemble des mots de longueur k construits à partir de Σ.

Exemple : $\Sigma = \{0, 1\}$
$\Sigma^1 = \{0, 1\}$
$\Sigma^2 = \{00, 01, 10, 11\}$
$\Sigma^0 = \{\varepsilon\}$
Question : combien de mots dans Σ^3 ?

Plus généralement : $\Sigma^0 = \{\varepsilon\}$

Concepts centraux : Langages (suite)

$\Sigma^\star = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \cdots \cup \cdots$ est l'ensemble des mots construits à partir de Σ.

$\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \cdots \cup \cdots$ est l'ensemble des mots de longueur positive.

$\Sigma^* = \Sigma^+ \cup \{\varepsilon\}$

Concaténation de mots : soit $x = x_1 \cdots x_p$ et $y = y_1 \cdots y_q$ deux mots. La concaténation de x et y est notée $xy = x_1 \cdots x_p y_1 \cdots y_q$.

Exemple : si $x = 01101$, $y = 110$ alors $xy = 01101110$
Remarque : pour tout mot x, $x\varepsilon = x\varepsilon = x$

Langages : c'est un sous-ensemble L de $\Sigma^\star$. On dit que L est construit sur Σ.

Concepts centraux : Langages (exemple)

- L'ensemble de tous les mots contenant n 0 suivis de n 1 pour $n \geq 0$: $\{\varepsilon, 01, 0011, 000111, \ldots\}$
- L'ensemble des mots contenant autant de 0 que de 1 :

$$\{\varepsilon, 01, 10, 0011, 0101, 1001, \ldots\}$$

- L'ensemble des codages binaires dont les entiers correspondant sont premiers.

$$\{10, 11, 101, 111, 1011, \ldots\}$$

- Le langage vide $\varnothing$
- le langage $\{\varepsilon\}$ restreint au mot vide.

Remarque : $\varnothing \neq \{\varepsilon\}$

Problématique centrale

Étant donné un mot, décider si celui-ci appartient à un certain langage.

Exemple : test de primalité.

Question : combien ce test de décision va-t-il coûter ? (complexité)

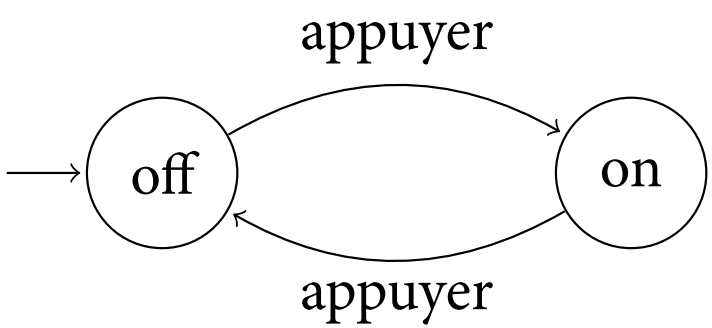
Définition de notre problème un peu restreinte : penser à un compilateur `C` !

Mais très utile pour :

- des problèmes de décidabilité (le test est-il possible ?)
- et des problèmes de calculabilité (bornes de complexité inférieure)

via des techniques de *réduction*.

Automates finis déterministes (AFD)



Un AFD est un 5-uplet (quintuplet) $A = (Q, \Sigma, \delta, q_0, F)$, où

- Q est un ensemble fini d'états
- Σ est un ensemble fini de symboles d'entrées
- $\delta : Q \times \Sigma \to Q$ est une fonction de transition qui prend en argument un état et un symbole d'entrée et retourne un état.
- $q_0 \in Q$ est l'état initiale (ou de départ)
- $F \subset Q$ est l'ensemble d'états *finaux* (ou *acceptants*)

Notations d'un AFD

On peut décrire un AFD en donnant une description détaillée de la fonction de transition δ.

C'est difficile à lire et délicat d'identifier le langage reconnu

- Diagramme de transition : c'est un graphe
- Table de transition : liste tabulée de l'action de δ

Diagramme de transition

Soit $A = (Q, \Sigma, \delta, q_0, F)$ un AFD.

Un diagramme de transition représentant A est défini comme suit :

- Chaque état q de Q est un nœud.
- Pour $q \in Q$ et $a \in \Sigma$, soit $\delta(q, a) = p$. On relie q à p par une flèche. Au-dessus de cette flèche, on mentionne que la transition se fait par le symbole a.
- Une flèche vers l'état de départ.
- Les nœuds correspondant à des états acceptants sont entourés par un double cercle

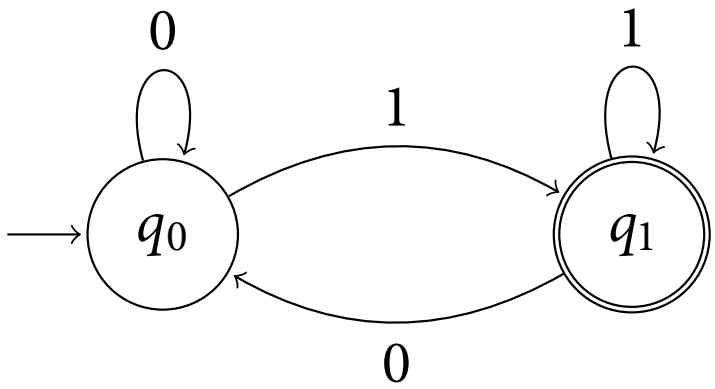
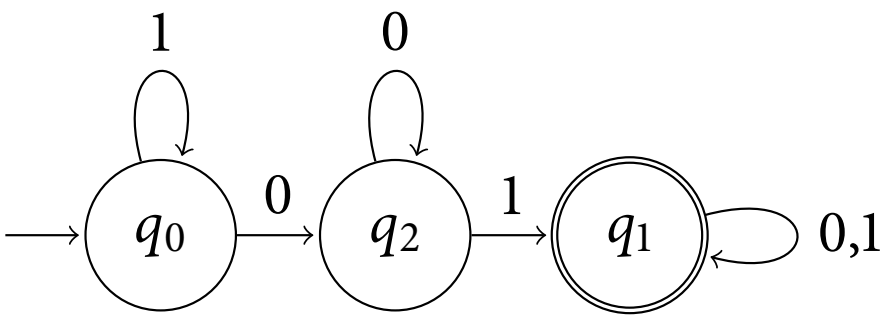


Table de transition

Table de transition : table représentant l'action de δ ; les lignes correspondent aux états, les colonnes aux symboles d'entrée.

L'état de départ est reconnu grâce à une flèche, les états acceptants grâce à une étoile.



	0	1
$\rightarrow q_0$	q_2	q_0
$\star q_1$	q_1	q_1
q_2	q_2	q_1

Fonctionnement d'un AFD

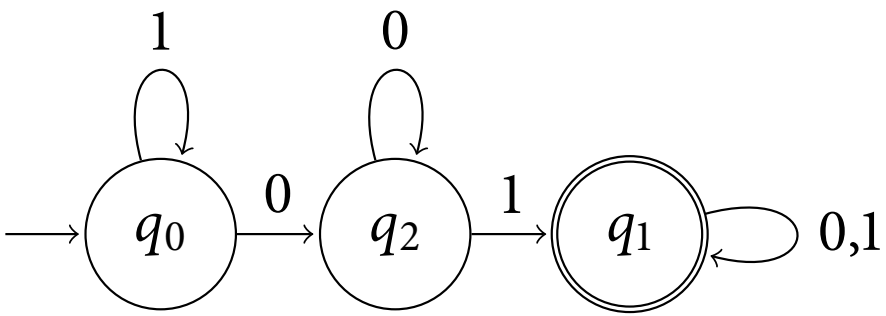
Entrée : un mot $a_1, \ldots, a_n$
État de départ : q_0

$$
\begin{array}{lcl}
\delta(q_0, a_1) & \to & q_1 \\
\delta(q_1, a_2) & \to & q_2 \\
 & \vdots & \\
\delta(q_{i-1}, a_i) & \to & q_i \\
 & \vdots & \\
\delta(q_{n-1}, a_n) & \to & q_n
\end{array}
$$

A-t-on $q_n \in F$?

Extension de la fonction de transition

Informellement : langage reconnu par un AFD = ensemble des mots menant à un état final.



Le langage reconnu est
$L = \{x01y : x, y \in \{0,1\}^*\}$

Description plus rigoureuse : fonction de transition étendue $\widehat{\delta}$

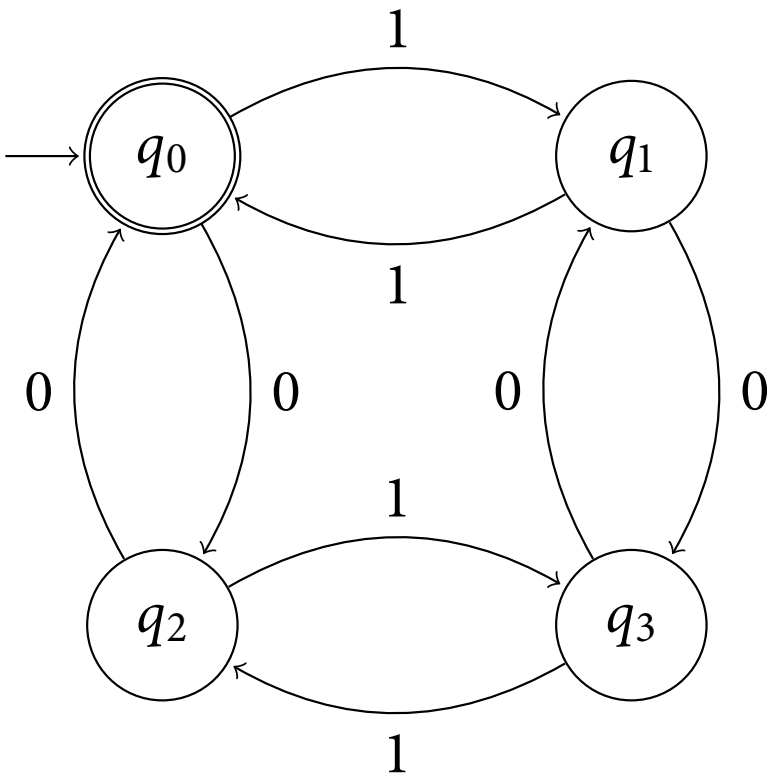
Elle prend en argument un état q et un mot w et retourne un état p (atteint par action de l'automate).

Définition par induction :

- $\widehat{\delta}(q, \varepsilon) = q$
- Si w est un mot de la forme xa $(a \in \Sigma)$: $\widehat{\delta}(q, w) = \delta(\widehat{\delta}(q, x), a)$

Extension de la fonction de transition (suite)

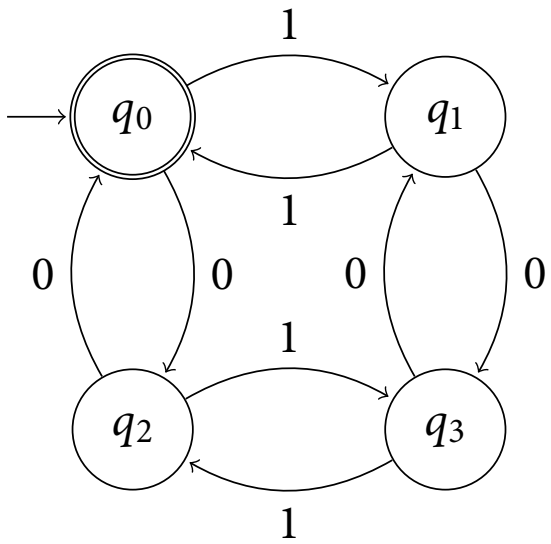
Exemple : Construire un automate reconnaissant l'ensemble des mots dont le nombre de 0 et de 1 est pair. Calcul sur 110101.



		0	1
$\star \rightarrow$	q_0	q_2	q_1
	q_1	q_3	q_0
	q_2	q_0	q_3
	q_3	q_1	q_2

$$q_0 \xrightarrow{1} q_1 \xrightarrow{1} q_0 \xrightarrow{0} q_2 \xrightarrow{1} q_3 \xrightarrow{0} q_1 \xrightarrow{1} q_0$$

Extension de la fonction de transition (suite)



- $\widehat{\delta}(q_0, \varepsilon) = q_0$
- $\widehat{\delta}(q_0, 1) = \delta(\widehat{\delta}(q_0, \varepsilon), 1) = \delta(q_0, 1) = q_1$
- $\widehat{\delta}(q_0, 11) = \delta(\widehat{\delta}(q_0, 1), 1) = \delta(q_1, 1) = q_0$
- $\widehat{\delta}(q_0, 110) = \delta(\widehat{\delta}(q_0, 11), 0) = \delta(q_0, 0) = q_2$
- $\widehat{\delta}(q_0, 1101) = \delta(\widehat{\delta}(q_0, 110), 1) = \delta(q_2, 1) = q_3$
- $\widehat{\delta}(q_0, 11010) = \delta(\widehat{\delta}(q_0, 1101), 0) = \delta(q_3, 0) = q_1$
- $\widehat{\delta}(q_0, 110101) = \delta(\widehat{\delta}(q_0, 11010), 1) = \delta(q_1, 1) = q_0$

Langage associé à un AFD

Soit $A = (Q, \Sigma, \delta, q_0, F)$ un AFD.

Le langage associé à A noté $L(A)$ est défini par

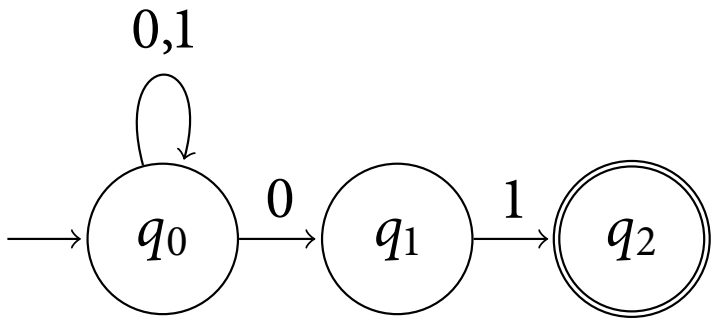
$$L(A) = \{w \mid \widehat{\delta}(q_0, w) \in F\}$$

Si L est égale à $L(A)$ pour un AFD A, on dit que L est un langage régulier

Automates finis non-déterministes (AFN)

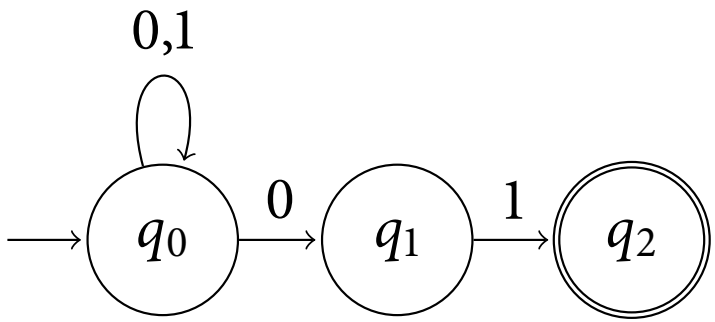
- Capacité d'être à différents états à la fois (pour deviner quelque chose sur l'entrée).
- Permet de d'écrire un automate reconnaissant un langage plus simplement.
- **Calculateur *in fine* pas plus puissant que les AFD !**

Description informelle : reconnaître le langage constitué des mots se terminant par 01.

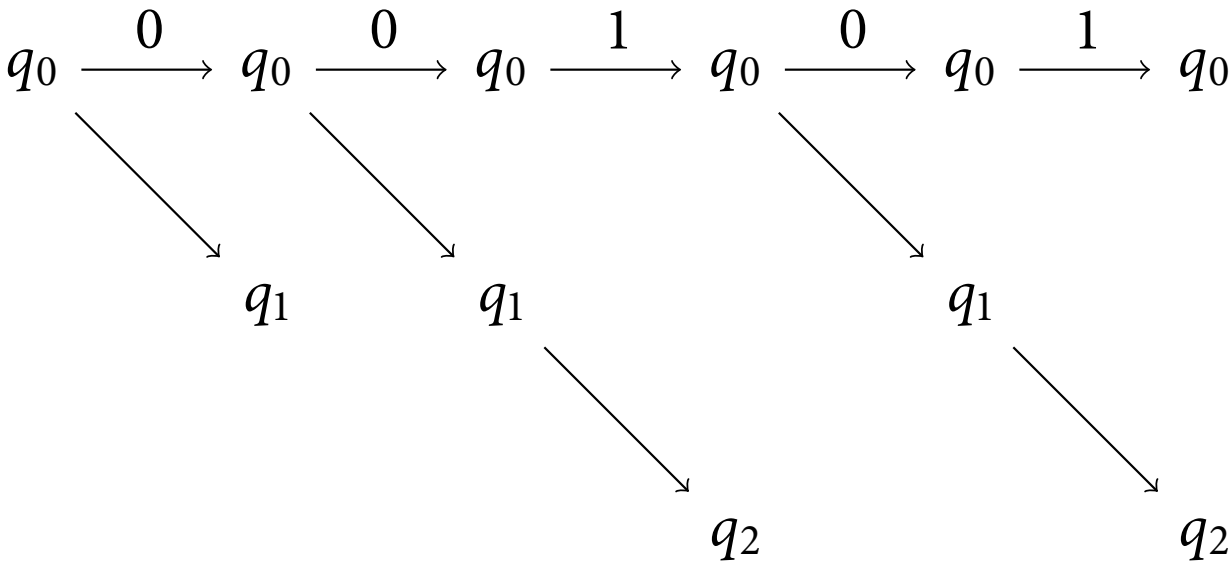


AFN (suite)

Automate reconnaissant le langage constitué des mots se terminant par 01.



Que se passe-t'il si on a 00101 en entrée ?

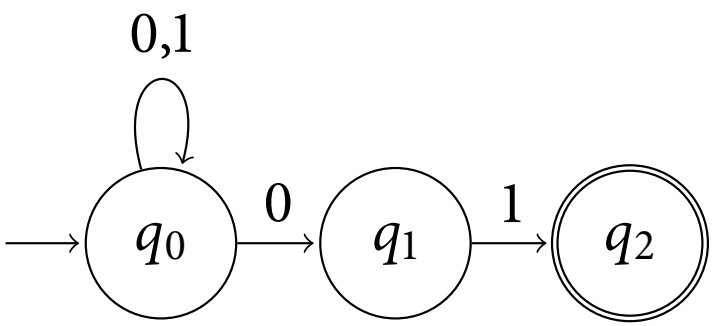


Définition d'un AFN

Un AFN est un 5-uplet $A = (Q, \Sigma, \delta, q_0, F)$ où

- Q est un ensemble fini d'états
- Σ est un alphabet (un ensemble fini de symboles)
- $\delta : Q \times \Sigma \to \mathcal{P}(Q)$ est une fonction de transition qui prend en argument un état de Q et un symbole de Σ et renvoie un **sous-ensemble** de Q.
- $q_0 \in Q$ est l'état initial.
- $F \subset Q$ est l'ensemble des états finaux

Exemple : $(\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_2\})$ et la table de transition :



	0	1
$\to q_0$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\varnothing$	$\{q_2\}$
$\star q_2$	$\varnothing$	$\varnothing$

Fonctions de transition étendues

Construction similaire à celle des AFD sauf que la sortie de δ est un sous-ensemble de Q !

Définition inductive :

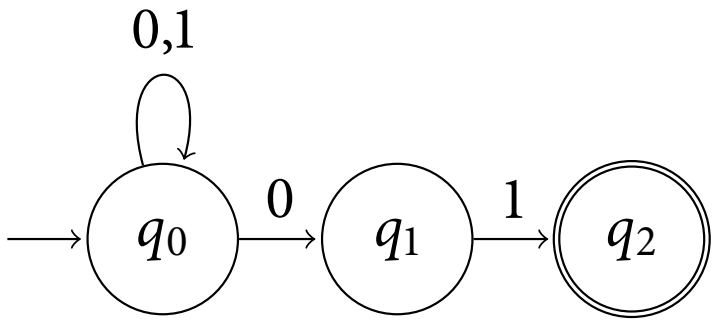
- $\widehat{\delta}(q, \varepsilon) = \{q\}$
- Soit $w = xa$ avec $a \in \Sigma$ et $\widehat{\delta}(q, x) = \{p_1, \ldots, p_k\}$ et

$$\bigcup_{i=1}^{k} \delta(p_i, a) = \{r_1, \ldots, r_m\}$$

Alors, $\widehat{\delta}(q, w) = \{r_1, \ldots, r_m\}$.

Fonctions de transition étendues (suite)

Exemple :



Calcul de $\widehat{\delta}(q_0, 00101)$:

- $\widehat{\delta}(q_0, \varepsilon) = \{q_0\}$
- $\widehat{\delta}(q_0, 0) = \delta(q_0, 0) = \{q_0, q_1\}$
- $\widehat{\delta}(q_0, 00) = \delta(q_0, 0) \cup \delta(q_1, 0) = \{q_0, q_1\} \cup \varnothing = \{q_0, q_1\}$
- $\widehat{\delta}(q_0, 001) = \delta(q_0, 1) \cup \delta(q_1, 1) = \{q_0\} \cup \{q_2\} = \{q_0, q_2\}$
- $\widehat{\delta}(q_0, 0010) = \delta(q_0, 0) \cup \delta(q_2, 0) = \{q_0, q_1\} \cup \varnothing = \{q_0, q_1\}$
- $\widehat{\delta}(q_0, 00101) = \delta(q_0, 1) \cup \delta(q_1, 1) = \{q_0\} \cup \{q_2\} = \{q_0, q_2\}$

Langage associé à un AFN

Soit $A = (Q, \Sigma, \delta, q_0, F)$ un AFN.

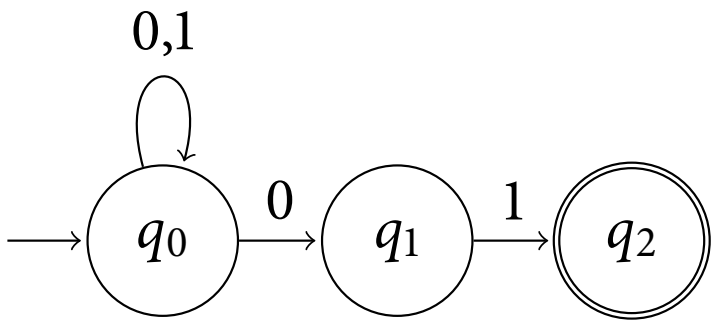
Le langage associé à A noté $L(A)$ est défini par

$$L(A) = \{w \mid \widehat{\delta}(q_0, w) \cap F \neq \varnothing\}$$

$\widehat{\delta}(q_0, w)$ doit contenir au moins un état acceptant.

Langage associé à un AFN (suite)

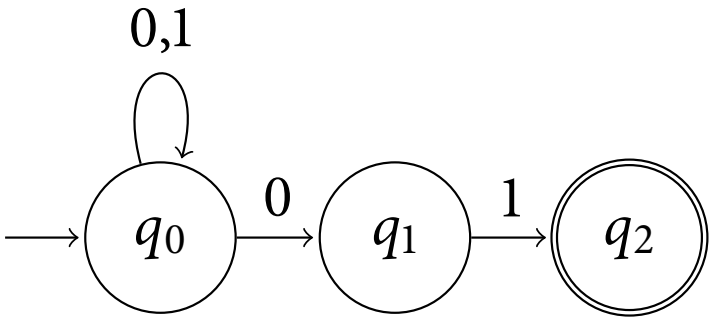
Prouvons que le langage reconnu par l'AFN est $L = \{x01 : x \in \Sigma^*\}$ avec $\Sigma = \{0, 1\}$



Par induction mutuelle sur la longueur du mot, on montre que

1. $w \in \Sigma^* \Rightarrow q_0 \in \widehat{\delta}(q_0, w)$
2. $q_1 \in \widehat{\delta}(q_0, w) \Leftrightarrow w = x0$
3. $q_2 \in \widehat{\delta}(q_0, w) \Leftrightarrow w = x01$

Langage associé à un AFN (suite)

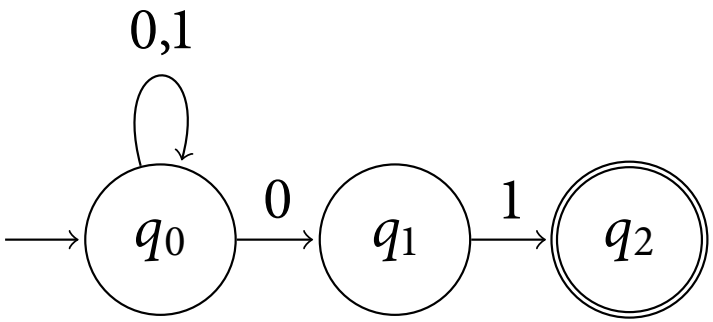


1. $w \in \Sigma^* \Rightarrow q_0 \in \widehat{\delta}(q_0, w)$
2. $q_1 \in \widehat{\delta}(q_0, w) \Leftrightarrow w = x0$
3. $q_2 \in \widehat{\delta}(q_0, w) \Leftrightarrow w = x01$

Base : si $|w| = 0$ alors $w = \varepsilon$. (1) est clair par définition et les deux côtés de (2) et (3) sont faux.

Induction : Supposons $w = xa$ avec $a \in \{0, 1\}$, $|x| = n$ et les énoncés (1) – (3) vrais pour x. Montrons que (1) – (3) sont encore vrais pour $w = xa$

Langage associé à un AFN (suite)

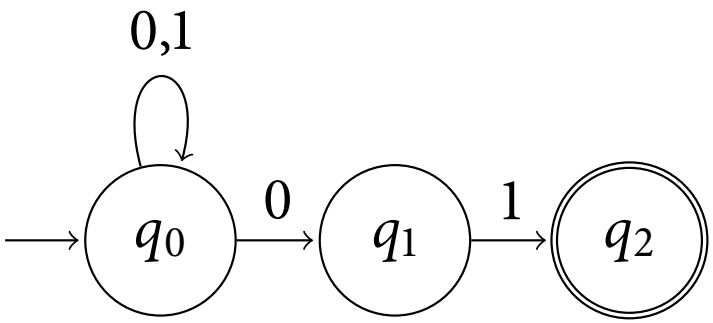


1. $w \in \Sigma^* \Rightarrow q_0 \in \widehat{\delta}(q_0, w)$
2. $q_1 \in \widehat{\delta}(q_0, w) \Leftrightarrow w = x0$
3. $q_2 \in \widehat{\delta}(q_0, w) \Leftrightarrow w = x01$

1. On sait que $\widehat{\delta}(q_0, x)$ contient q_0. Puisqu'il y a des transitions entre q_0 et lui-même en lisant 0 ou 1, on a $q_0 \in \widehat{\delta}(q_0, w)$
2. ($\Rightarrow$) Supposons $q_1 \in \widehat{\delta}(q_0, w)$. On voit en lisant le diagramme que la seule façon d'atteindre l'état q_1 est de lire un mot w de la forme $x0$.

   ($\Leftarrow$) Supposons que w se termine par 0 (c-à-d $a = 0$). D'après (1), $q_0 \in \widehat{\delta}(q_0, x)$. Puisqu'il y a une transition entre q_0 et q_1 en lisant 0, on a $q_1 \in \widehat{\delta}(q_0, w)$

Langage associé à un AFN (suite)



1. $w \in \Sigma^* \Rightarrow q_0 \in \widehat{\delta}(q_0, w)$
2. $q_1 \in \widehat{\delta}(q_0, w) \Leftrightarrow w = x0$
3. $q_2 \in \widehat{\delta}(q_0, w) \Leftrightarrow w = x01$

3. ($\Rightarrow$) Supposons $q_2 \in \widehat{\delta}(q_0, w)$. On voit en lisant le diagramme que la seule façon d'atteindre l'état q_2 est de lire un mot de la forme $x1$ avec $q_1 \in \widehat{\delta}(q_0, x)$. En appliquant (2) à x, on conclut que x se termine par 0. Par conséquent, w se termine par 01.

   ($\Leftarrow$) Supposons que w se termine par 01. Alors si $w = xa$ nous savons que $a = 1$ et que x se termine par 0. En appliquant (2) à x, on sait que $q_1 \in \widehat{\delta}(q_0, x)$. Puisqu'il y a une transition entre q_1 et q_2 en lisant 1, on conclut que $q_2 \in \widehat{\delta}(q_0, w)$.

Équivalence AFD-AFN

- les AFN sont souvent plus faciles à « programmer »
- De manière surprenante, pour tout AFN N, il existe un AFD D tel que $L(D) = L(N)$
- C'est un exemple générique de construction d'un automate B à partir d'un autre automate A
- Étant donné un AFN $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$, on va construire un AFD $D = (Q_D, \Sigma, \delta_D, \{q_0\}, F_D)$ tel que

$$L(D) = L(N)$$

Équivalence AFD-AFN (suite)

Les AFN ne reconnaissent pas plus de langage que les AFD !

Construction des sous-ensembles d'état d'un automate.

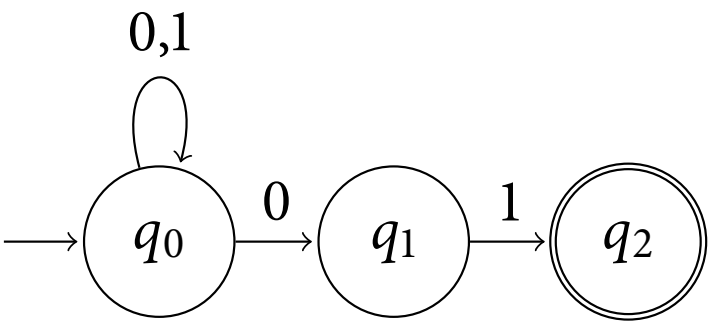
Soit $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$ un AFN.
$\rightarrow$ Construire un AFD $D = (Q_D, \Sigma, \delta_D, \{q_0\}, F_D)$ reconnaissant le même langage que celui reconnu par N.

- Q_D est l'ensemble des sous-ensembles de Q_N
- F_D est l'ensemble des sous-ensembles S de Q_N tels que $S \cap F_N \neq \emptyset$.
- Pour tout sous-ensemble S de Q_N et $a \in \Sigma$

$$\delta_D(S, a) = \bigcup_{p \in S} \delta_N(p, a)$$

Facile à construire à partir de la table de transition.

Équivalence AFD-AFN (suite)



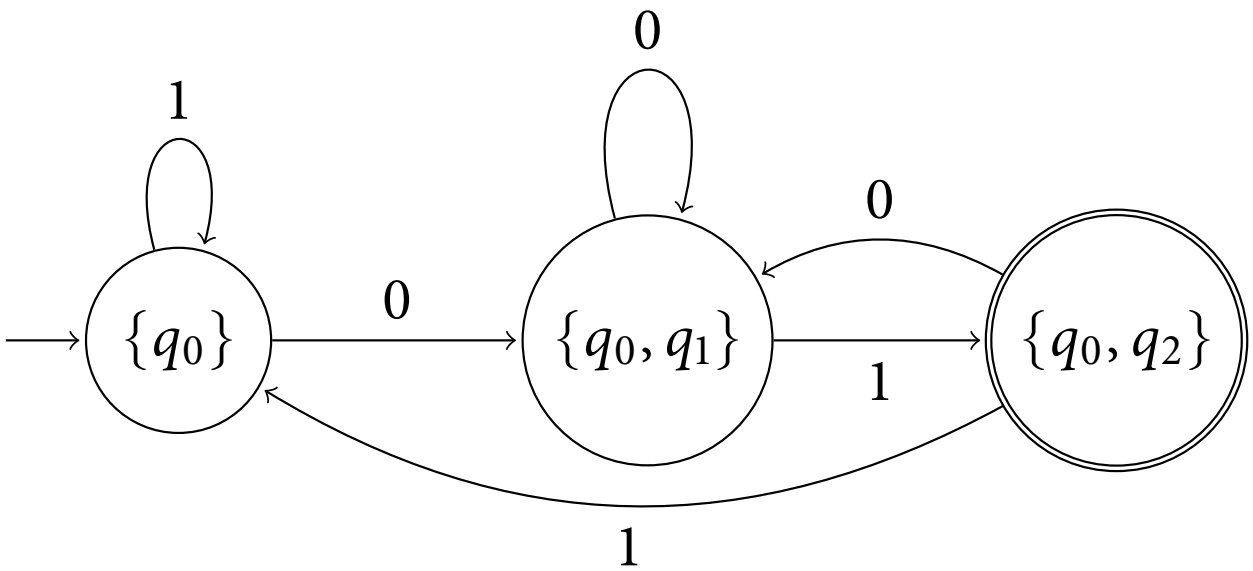
	0	1
$\to q_0$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\varnothing$	$\{q_2\}$
$\star q_2$	$\varnothing$	$\varnothing$

	0	1
$\varnothing$	$\varnothing$	$\varnothing$
$\to \{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_1\}$	$\varnothing$	$\{q_2\}$
$\star \{q_2\}$	$\varnothing$	$\varnothing$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\star \{q_0, q_2\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\star \{q_1, q_2\}$	$\varnothing$	$\{q_2\}$
$\star \{q_0, q_1, q_2\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$

Équivalence AFD-AFN (suite)

Si $|Q_N| = n$, D peut avoir 2^n états ! On peut éviter d'avoir trop d'états en ne construisant la table de transition que pour les états atteignables S définis par

- $S = \{q_0\}$ est atteignable dans D
- si S est atteignable alors pour chaque $a \in \Sigma$, $\delta_D(S, a)$ est aussi atteignable



Équivalence AFD-AFN (suite)

L'automate D ainsi construit reconnaît le même langage que celui reconnu par N.

Théorème 1

Soit D l'AFD construit précédemment à partir de l'AFN N. Alors $L(D) = L(N)$.

Preuve. On montre par induction sur la longueur du mot w que

$$\widehat{\delta}_D(\{q_0\}, w) = \widehat{\delta}_N(q_0, w)$$

Base : pour $w = \varepsilon$, clair d'après la définition.

Induction :

$$\begin{aligned}
\widehat{\delta}_D(\{q_0\}, xa) &= \delta_D(\widehat{\delta}_D(\{q_0\}, x), a) \\
&= \delta_D(\widehat{\delta}_N(q_0, x), a) \\
&= \bigcup_{p \in \widehat{\delta}_N(q_0, x)} \delta_N(p, a) = \widehat{\delta}_N(q_0, xa)
\end{aligned}$$

Équivalence AFD-AFN (suite)

Par conséquent $L(D) = L(N)$

On en déduit le théorème suivant

Théorème 2

Un langage L est accepté par un AFD si et seulement si L est accepté par un AFN.

Automates finis et ε-transitions

On autorise un changement d'état (une transition) à la lecture du mot vide ε.

Les AFN à ε-transitions (ε-AFN) ne reconnaissent pas plus de langage que les AFD.

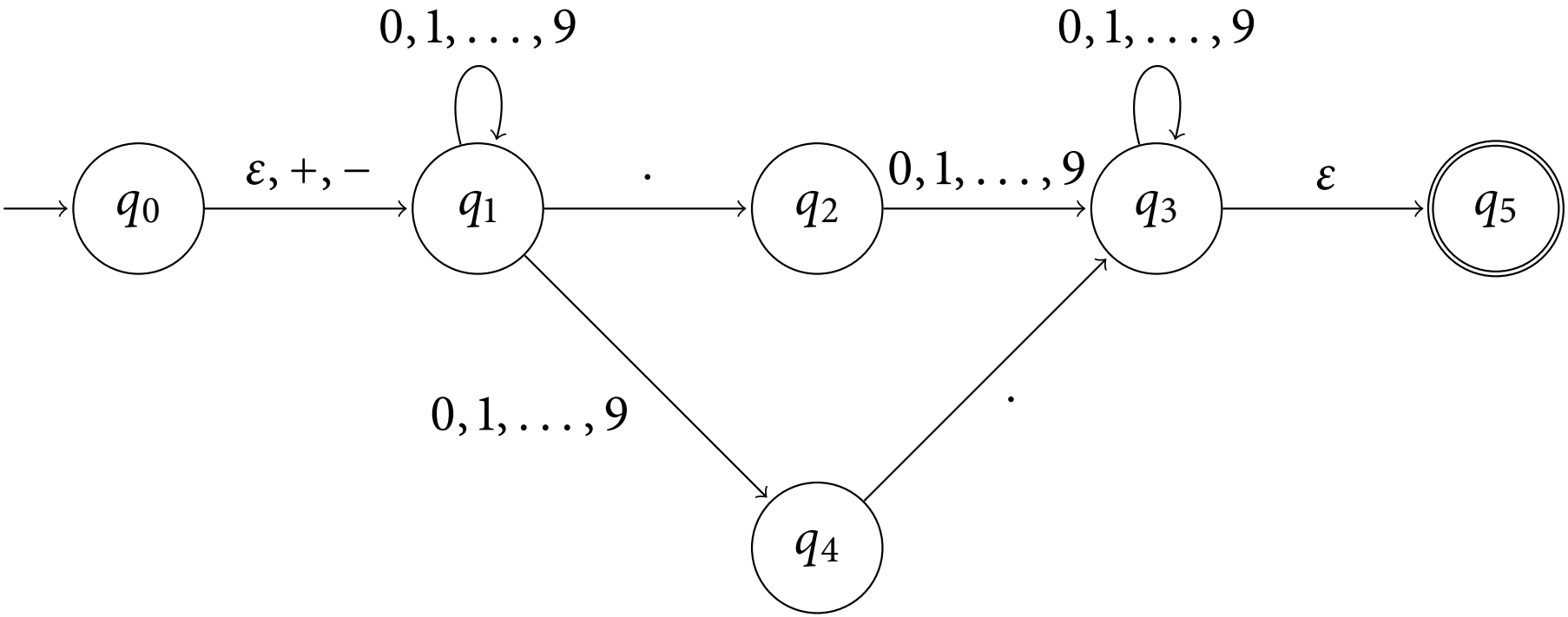
Exemple : reconnaître des nombres décimaux qui consistent en :

1. Un signe optionnel + ou −
2. Une chaîne de chiffres
3. Un point décimal
4. Une autre chaîne de chiffres.

Une des chaînes (2) ou (4) est optionnelle.

Automates finis et ε-transitions (suite)

Comme pour un AFN sauf qu'on travaille sur $\Sigma \cup \{\varepsilon\}$!



Automates finis et ε-transitions (suite)

Exemple : un ε-AFN acceptant les mots-clé `ebay`, `web`