

# THEORIE DES LANGAGES ET AUTOMATES

# INTRODUCTION

- Les automates ont été définis dans les années 40-50.
- Ils sont utilisés dans les beaucoup de domaines comme:
  - Compilation des langages
  - Reconnaissance de texte
  - Conception de protocoles
  - Synthèse de programmes
  - Vérification de programmes
- Fin des années 50, Chomsky un linguiste a commencé l'étude des grammaires formelles.
- Les liens entre les grammaires et les automates ont été étudiés par la suite

- Alan Turing a étudié en 1930, avant que n'existe le premier ordinateur, une question fondamentale en Informatique:

Existe-t-il une limite à ce qu'on peut calculer?

Sa réponse:

oui, il y a des problèmes qu'on ne peut pas résoudre à l'aide d'un ordinateur même si on suppose une mémoire non bornée

- Bibliographie
  - P. Dehornoy, "Mathématiques d l'Informatique", Dunod, 2000.
  - J. Hopcroft et J. Ullmann, "Introduction to automata theory, Languages and Computations", Addison-Wesley, 1979.
- Webographie
  - <http://njussien.e-constraints.net/lla/>
  - <http://brassens.upmf-grenoble.fr/~alecomte/>

# Plan

- Mots et langages
- Automates à états finis
- Langages réguliers
- Grammaires
- Langages hors contextes et automates à pile
- Machine de Turing

# Mots et langages

# Mots

- Un **alphabet**  $\Sigma$  est un ensemble fini non vide de lettres (ou symboles)
- Un **mot** est une suite finie de symboles de l'alphabet  $\Sigma$   
Exemple: 001110 est un mot sur  $\Sigma = \{0, 1\}$
- Un mot  $w$  constitué de  $m$  symboles est dit de longueur  $m$ . On note  $|w| = m$ .

Exemple:  $|001110| = 6$   
 $|\varepsilon| = 0$

- On note  $\Sigma^*$  l'ensemble des mots sur  $\Sigma$
- $\Sigma^i$  est l'ensemble des mots de longueur  $i$  .  
On a :  $\Sigma^* = \bigcup_n \Sigma^n$
- NB : On note  $\Sigma^+$  l'ensemble  $\Sigma^* \setminus \{\varepsilon\}$



# Opérations

- **Concaténation**

Soit  $x \in \Sigma^*$  et  $y \in \Sigma^*$  tels que  $|x| = m$  et  $|y| = n$ .

La concaténation de  $x$  et  $y$  notée  $xy$  est le mot de longueur  $m + n$  dont les  $m$  premiers symboles sont ceux de  $x$  et les  $n$  derniers ceux de  $y$

- Propriétés de la concaténation

- associativité
- non commutativité
- $\varepsilon$  neutre pour la concaténation

- Portions de mots:

Soient  $w, x, y, z$  quatre mots tels que :  $w = xyz$

- $x$  est un **préfixe** de  $w$
- $y$  est une **sous-chaîne** de  $w$
- $z$  est un **suffixe** de  $w$

- Exemple

Si le mot considéré est  $w = 00110$

- préfixes :  $\varepsilon, 0, 00, 001, 0011$  et  $w$
- suffixes :  $\varepsilon, 0, 10, 110, 0110$  et  $w$
- sous-chaînes :  $\varepsilon, 0, 1, 00, 01, 10, 11, 001, 011, 110, 0011, 0110$  et  $w$

- L'opération **miroir** est définie de la manière suivante :
  - si  $|w| = 0$ , alors  $w = \varepsilon$  et  $w^R = \varepsilon$
  - Sinon  $|w| > 0$ ,  $w = au$  ( $a \in \Sigma$  et  $u \in \Sigma^*$ ),  $w^R = u^R a$ .

Si  $w$  est tel que  $w^R = w$  alors  $w$  est un **palindrome**.

# Langages

- **Langage**

Un langage (formel)  $L$  sur un alphabet  $\Sigma$  est un sous-ensemble quelconque de  $\Sigma^*$  tel que  $L \subseteq \Sigma^*$

- Exemple

Le langage des nombres est défini sur un alphabet  $\Sigma = \{0, 1, \dots, 9\}$ . 02, 00310, 3200 sont alors des mots sur  $\Sigma$ . On définira le langage des nombres comme les mots sur  $\Sigma$  qui ne commencent pas par 0. Ainsi, 1233 et 3200 seront des mots du langage mais pas 00310.

- **Concaténation de langages**

Soient  $L1$  et  $L2$  deux langages, leur concaténation est le langage

$$L1L2 = \{xy | x \in L1, y \in L2\}$$

- Propriétés de la concaténation
  - associative
  - non commutative
  - $\{\varepsilon\}$  neutre
  - $\emptyset$  absorbant

- **Clôture de Kleene**

- $L^2 = LL$  la concaténation de  $L$  avec lui-même
- $L^0 = \{\varepsilon\}$
- clôture de Kleene

$$L^* = \bigcup_{i=0 \rightarrow \infty} L^i$$

- Autres opérations sur les langages:
  - $A + B$  (ou  $A \cup B$ ) =  $\{w \in X^* \mid w \in A \text{ ou } w \in B\}$
  - $A \cap B = \{w \in X^* \mid w \in A \text{ et } w \in B\}$
  - $A - B$  (ou  $A \setminus B$ ) =  $\{w \in X^* \mid w \in A \text{ et } w \notin B\}$
  - $L^R = \{u^R \mid u \in L\}$  (image miroir d'un langage)

- Propriétés:

- $A \emptyset = \emptyset A = \emptyset$

- $A \{\varepsilon\} = \{\varepsilon\} A = A$

- $A \subseteq B \Rightarrow AC \subseteq BC$

- $A^* = \bigcup_{i=0 \rightarrow \infty} A^i$

- $A^+ = \bigcup_{i=1 \rightarrow \infty} A^i$

- Remarque:

On n'a pas la distributivité de la concaténation par rapport à l'intersection



## **Théorème d'Arden**

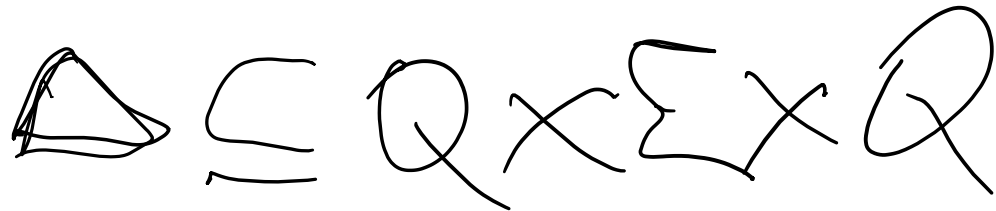
A, B langages sur  $\Sigma$ .

Si  $\varepsilon \notin A$  alors l'équation  $L = AL + B$  admet une unique solution  $L = A^*B$

# **Automates à Etats finis (AEF)**

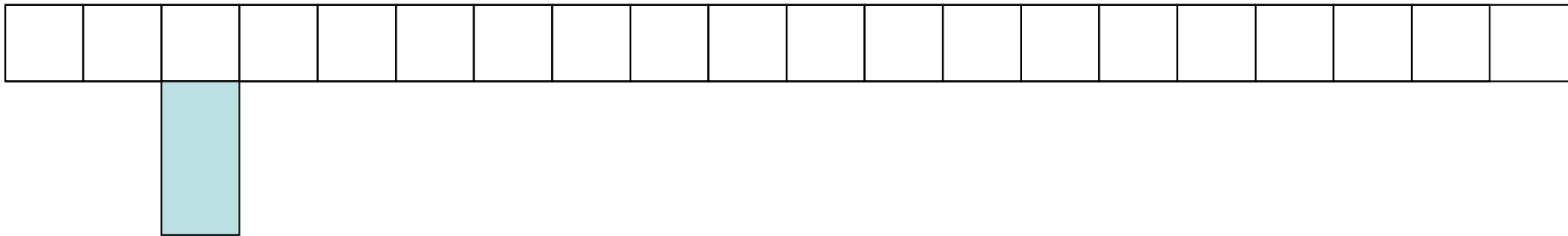
# Automates à états finis

- Un Automate à Etats Finis (AEF) est défini par la donnée de:
  - un ensemble fini non vide de **symboles**  $\Sigma$
  - un ensemble fini d'**états**  $Q$
  - un élément distingué de  $Q$ , appelé **état initial**
  - un sous-ensemble  $F$  de  $Q$ , appelé ensemble des **états finaux**
  - une fonction partielle  $\delta: Q \times \Sigma \rightarrow Q$ , appelée **fonction de transition**.

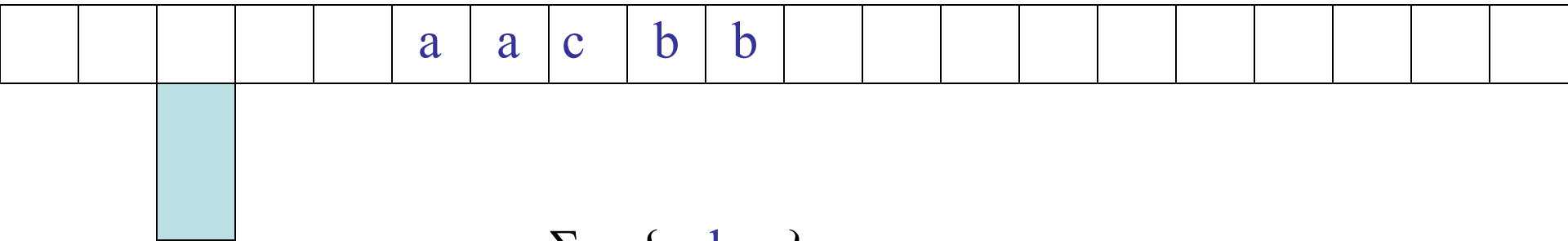

$$\triangle \subseteq Q \times \Sigma \times Q$$

- représentation concrète:

un ruban, illimité à droite, divisé en cases  
une tête de lecture positionnée sur une  
seule case à la fois

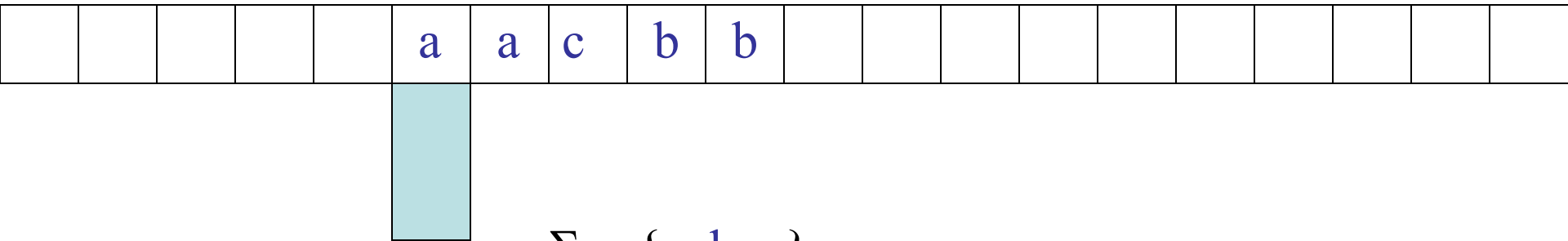


chaque case du ruban contient au plus un symbole, c'est alors  
nécessairement un symbole de  $\Sigma$   
celles qui ne contiennent pas de symbole: cases vides, ou occupées  
par des blancs (« \_ »)



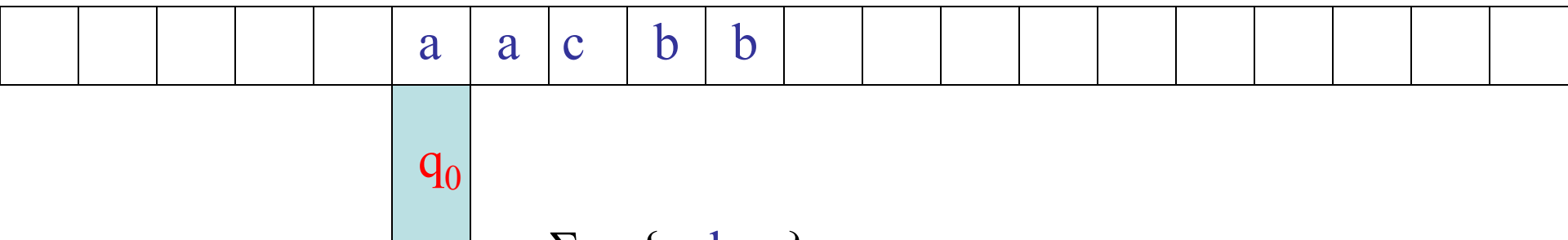
$$\Sigma = \{a, b, c\}$$

Au départ: tête de lecture sur première case non vide à partir  
de la gauche  
si ruban vide: sur n'importe quelle case



$$\Sigma = \{a, b, c\}$$

Au départ: tête de lecture sur première case non vide à partir de la gauche dans l'état initial  $q_0$   
si ruban vide: sur n'importe quelle case

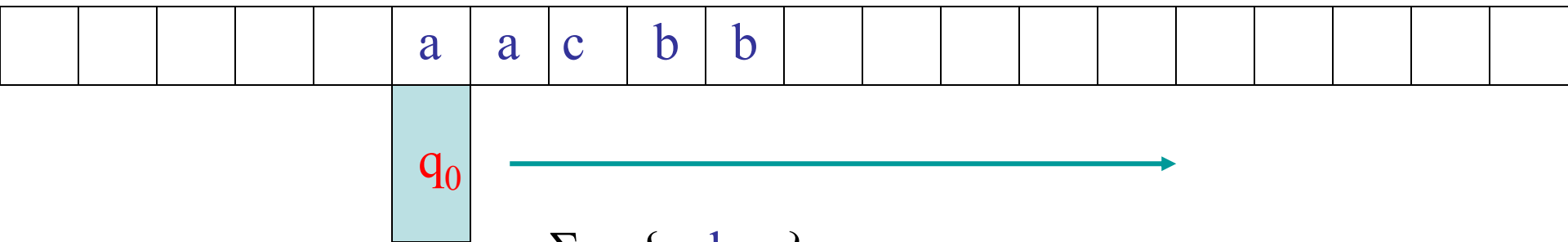


$$\Sigma = \{a, b, c\}$$

$$Q = \{q_0, q_1, q_2\}$$

$q_0$  initial

La tête de lecture se déplace uniquement vers la droite,  
d'une seule case à chaque instant du calcul  
son mouvement est déterminé par la fonction  $\delta$



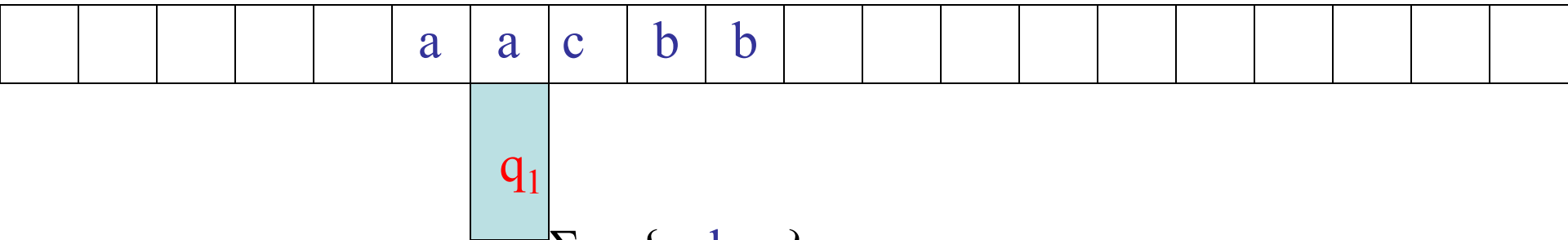
$$\Sigma = \{a, b, c\}$$

$$Q = \{q_0, q_1, q_2\}$$

$q_0$  initial



Si  $\delta(q_0, a) = q_1$  alors la tête de lecture passe à l'état  $q_1$  puis se déplace d'une case vers la droite



$$\Sigma = \{a, b, c\}$$

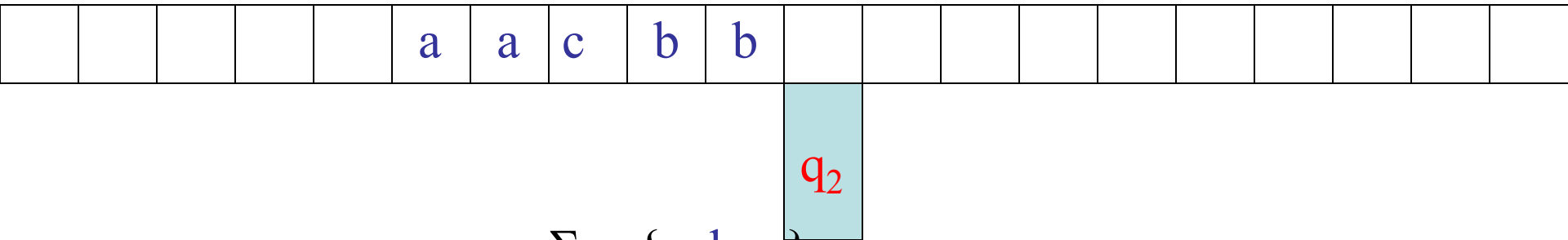
$$Q = \{q_0, q_1, q_2\}$$

$q_0$  initial

But: mot accepté si et seulement si

1- mot entièrement « lu »

2- arrêt dans un état final



$$\Sigma = \{a, b, c\}$$

$$Q = \{q_0, q_1, q_2\}$$

$q_0$  initial       $q_2$  final

# Exemple d'automate

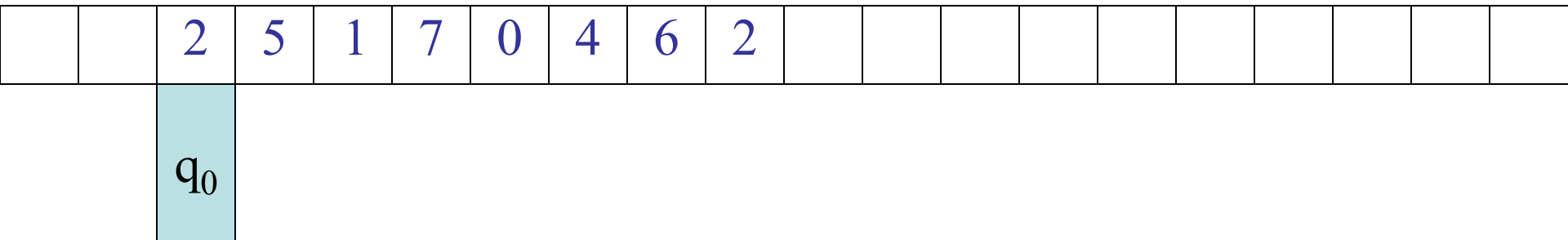
$\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

$Q = \{q_0, q_1, q_2\}$

$Q_f = \{q_0\}$

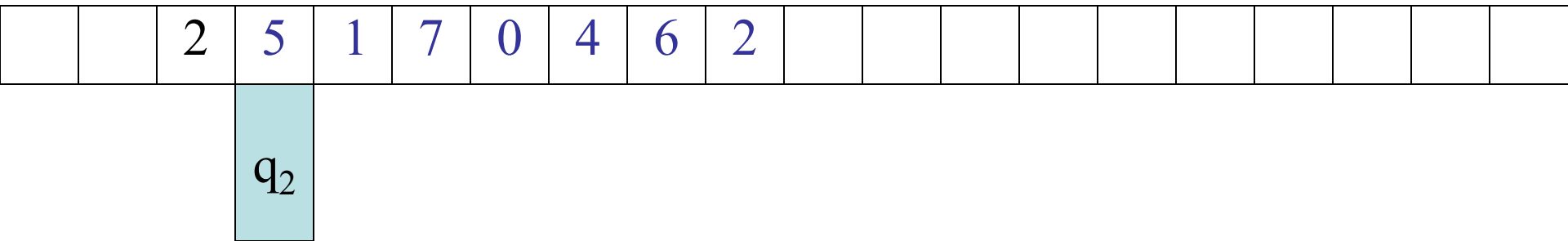
$\tau :$	0, 3, 6, 9	1, 4, 7	2, 5, 8
$q_0$	$q_0$	$q_1$	$q_2$
$q_1$	$q_1$	$q_2$	$q_0$
$q_2$	$q_2$	$q_0$	$q_1$

soit à reconnaître le mot 25170462



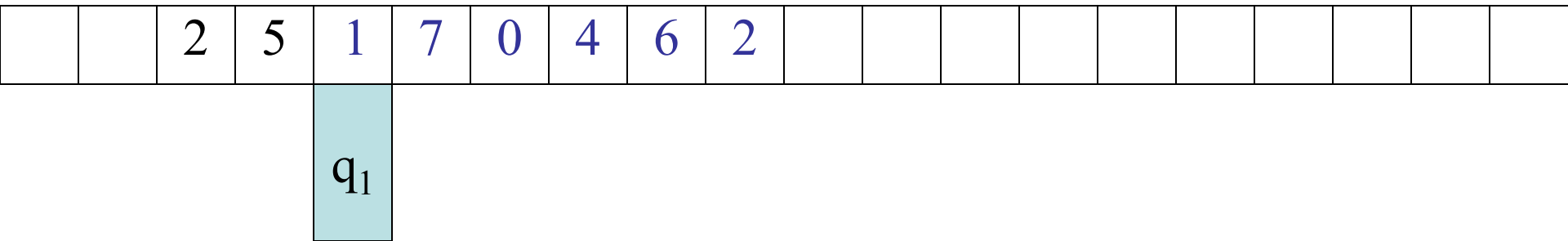
$\tau :$	0, 3, 6, 9	1, 4, 7	2, 5, 8
$q_0$	$q_0$	$q_1$	$q_2$
$q_1$	$q_1$	$q_2$	$q_0$
$q_2$	$q_2$	$q_0$	$q_1$

soit à reconnaître le mot 25170462



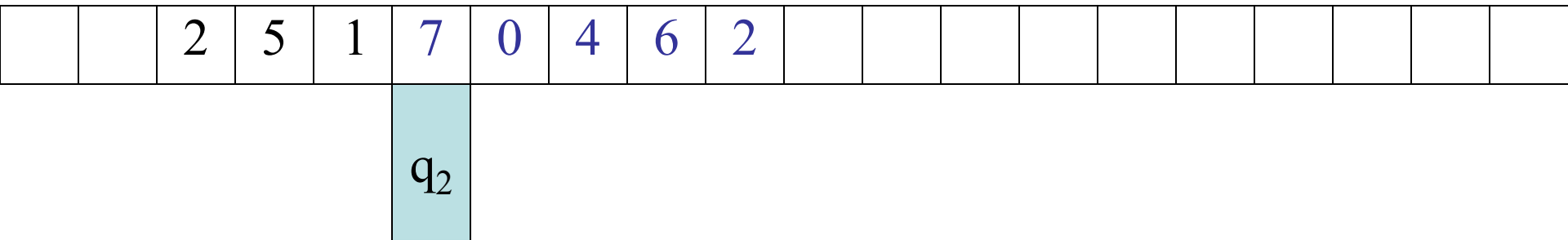
$\tau :$	0, 3, 6, 9	1, 4, 7	2, 5, 8
$q_0$	$q_0$	$q_1$	$q_2$
$q_1$	$q_1$	$q_2$	$q_0$
$q_2$	$q_2$	$q_0$	$q_1$

soit à reconnaître le mot 25170462



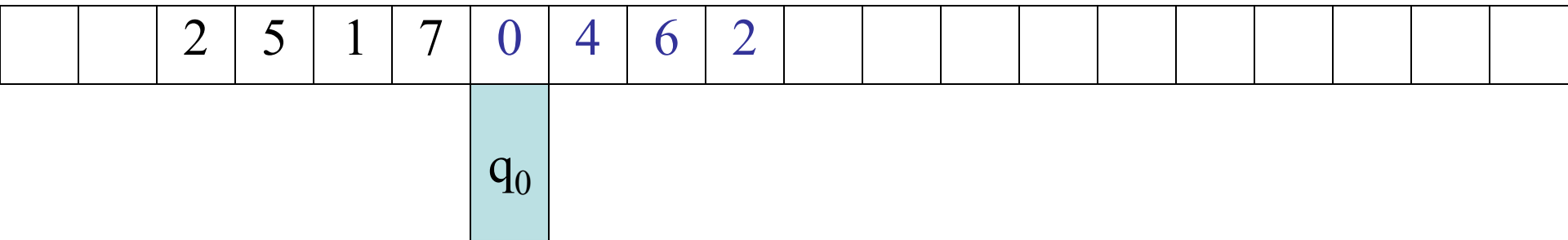
$\tau :$	0, 3, 6, 9	1, 4, 7	2, 5, 8
q0	q0	q1	q2
q1	q1	q2	q0
q2	q2	q0	q1

soit à reconnaître le mot 25170462



$\tau :$	0, 3, 6, 9	1, 4, 7	2, 5, 8
q0	q0	q1	q2
q1	q1	q2	q0
q2	q2	q0	q1

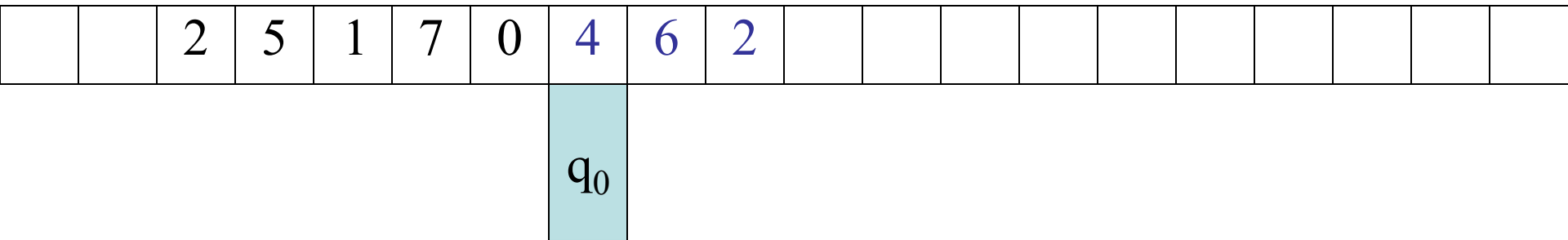
soit à reconnaître le mot 25170462



$\tau :$	0, 3, 6, 9	1, 4, 7	2, 5, 8
$q_0$	$q_0$	$q_1$	$q_2$
$q_1$	$q_1$	$q_2$	$q_0$
$q_2$	$q_2$	$q_0$	$q_1$

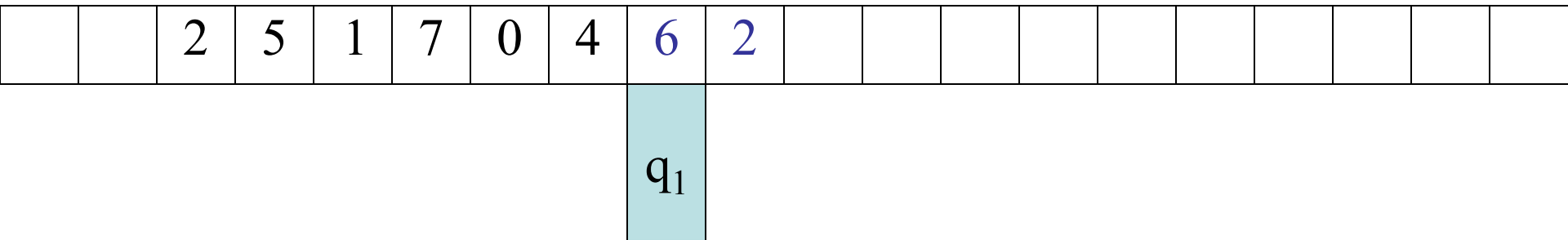


soit à reconnaître le mot 25170462



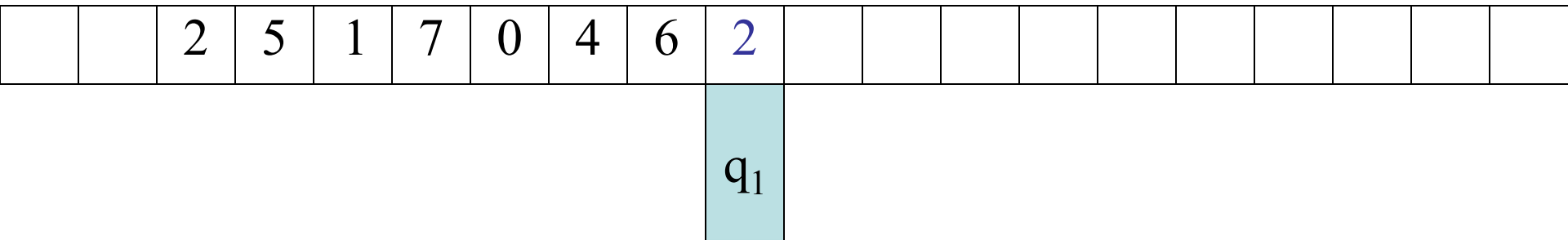
$\tau :$	0, 3, 6, 9	1, 4, 7	2, 5, 8
$q_0$	$q_0$	$q_1$	$q_2$
$q_1$	$q_1$	$q_2$	$q_0$
$q_2$	$q_2$	$q_0$	$q_1$

soit à reconnaître le mot 25170462



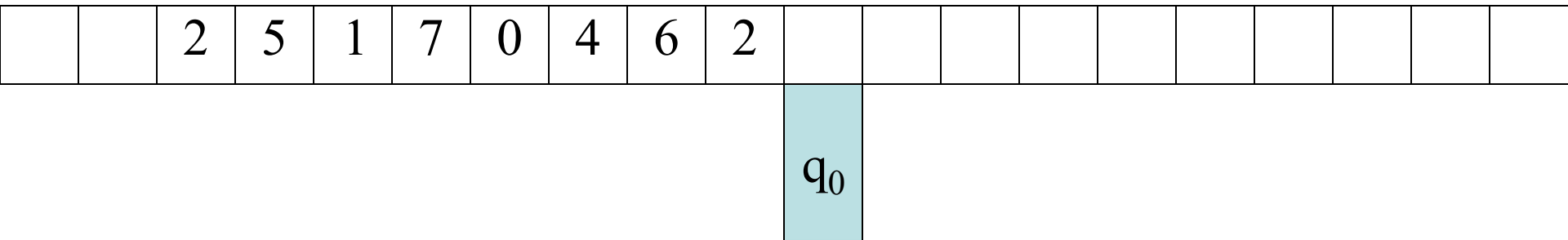
$\tau :$	0, 3, 6, 9	1, 4, 7	2, 5, 8
$q_0$	$q_0$	$q_1$	$q_2$
$q_1$	$q_1$	$q_2$	$q_0$
$q_2$	$q_2$	$q_0$	$q_1$

soit à reconnaître le mot 25170462



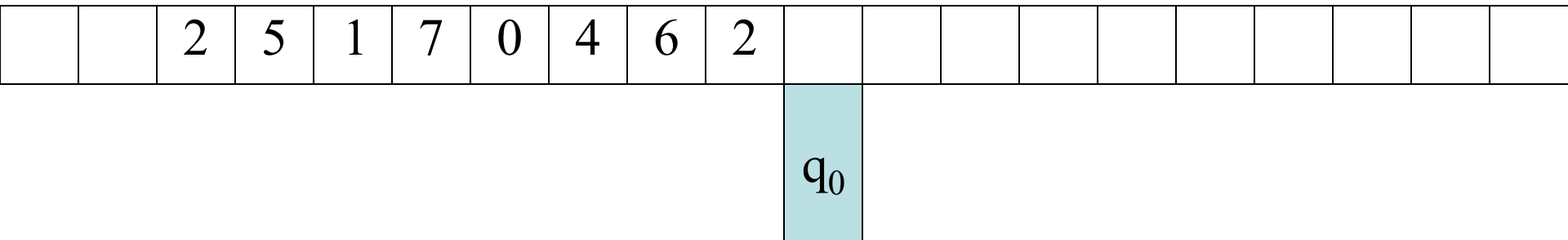
$\tau :$	0, 3, 6, 9	1, 4, 7	2, 5, 8
q0	q0	q1	q2
q1	q1	q2	q0
q2	q2	q0	q1

soit à reconnaître le mot 25170462



$\tau :$	0, 3, 6, 9	1, 4, 7	2, 5, 8
q0	q0	q1	q2
q1	q1	q2	q0
q2	q2	q0	q1

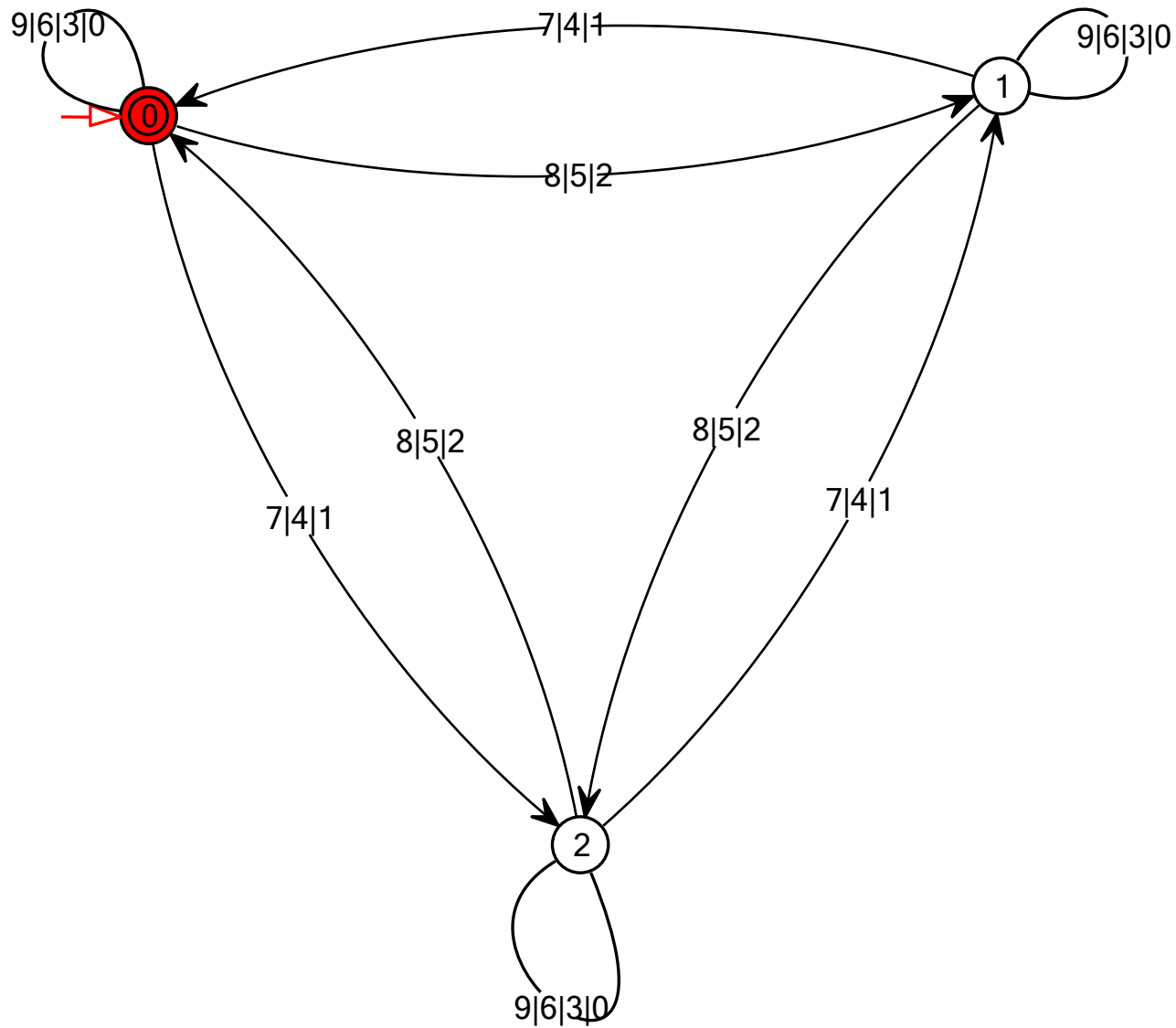
soit à reconnaître le mot 25170462



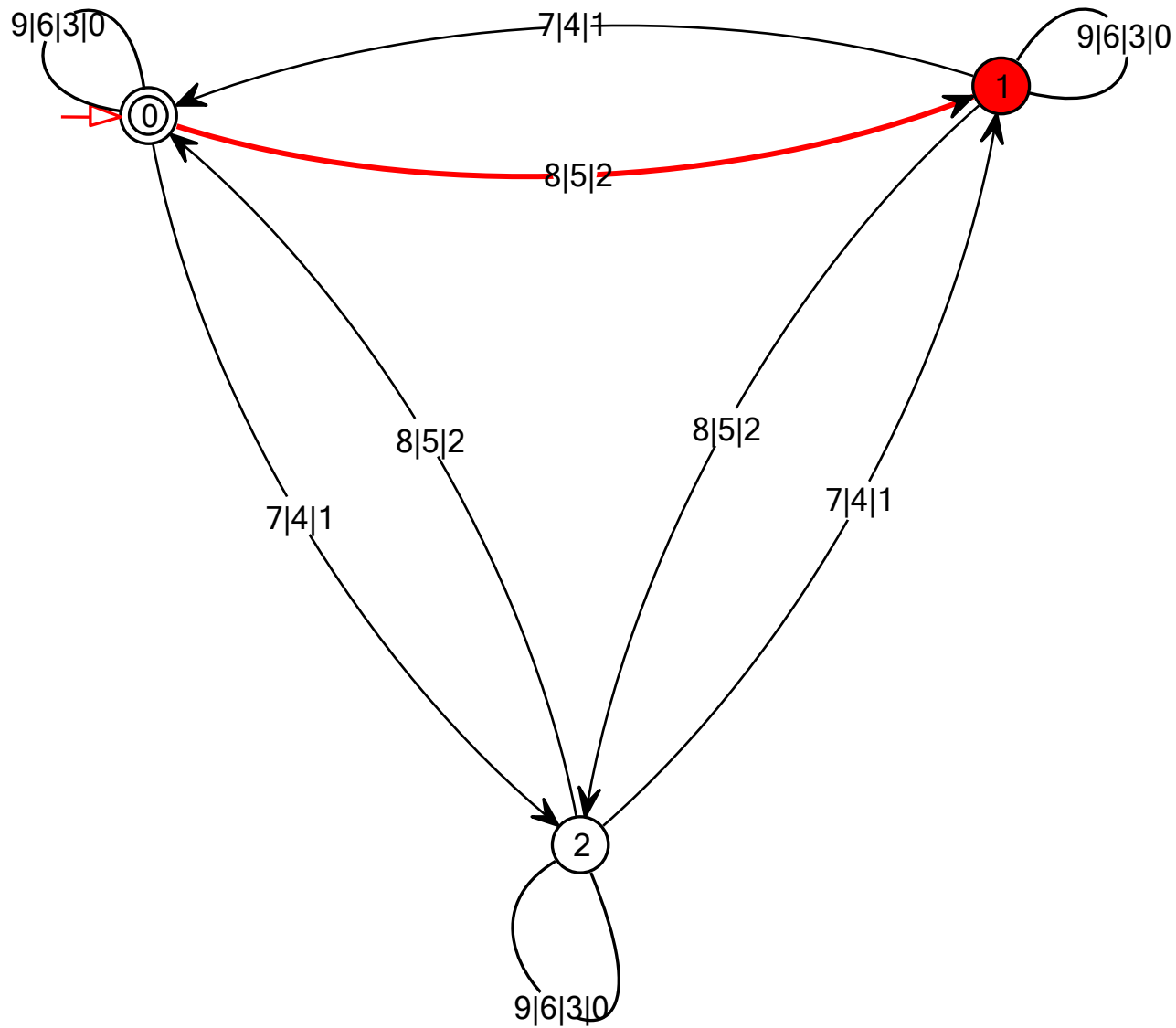
- L'automate s'arrête dans un état final,
- il ne reste plus rien à lire
- le mot est donc accepté

## Représentation par diagramme

- états = sommets
- transitions = arcs étiquetés (par les lettres)
- état initial = sommet de départ
- état final = sommet d'arrivée
- mot = chemin
- mot accepté = chemin allant de l'état initial à un état final

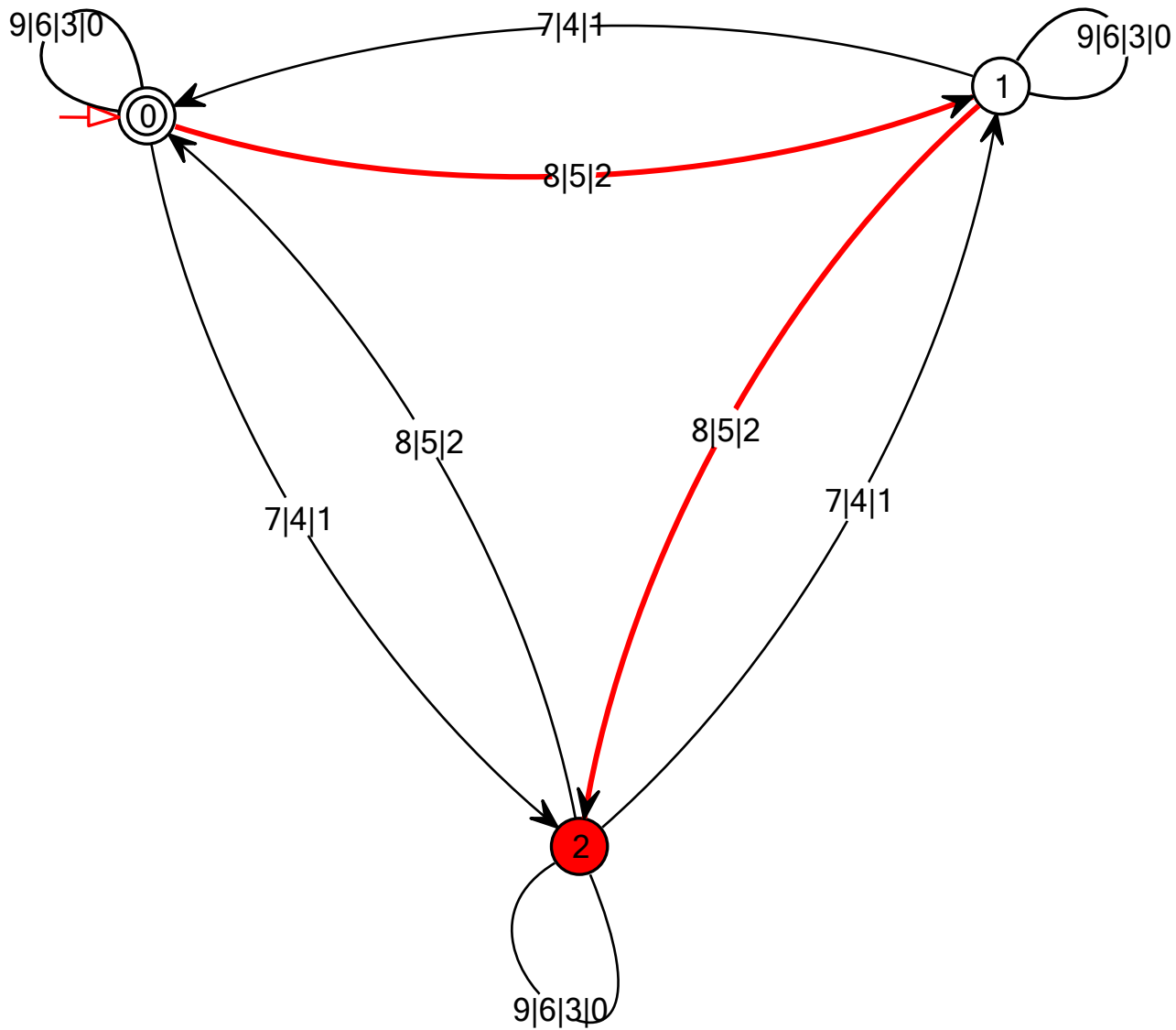


reconnaître 25170462

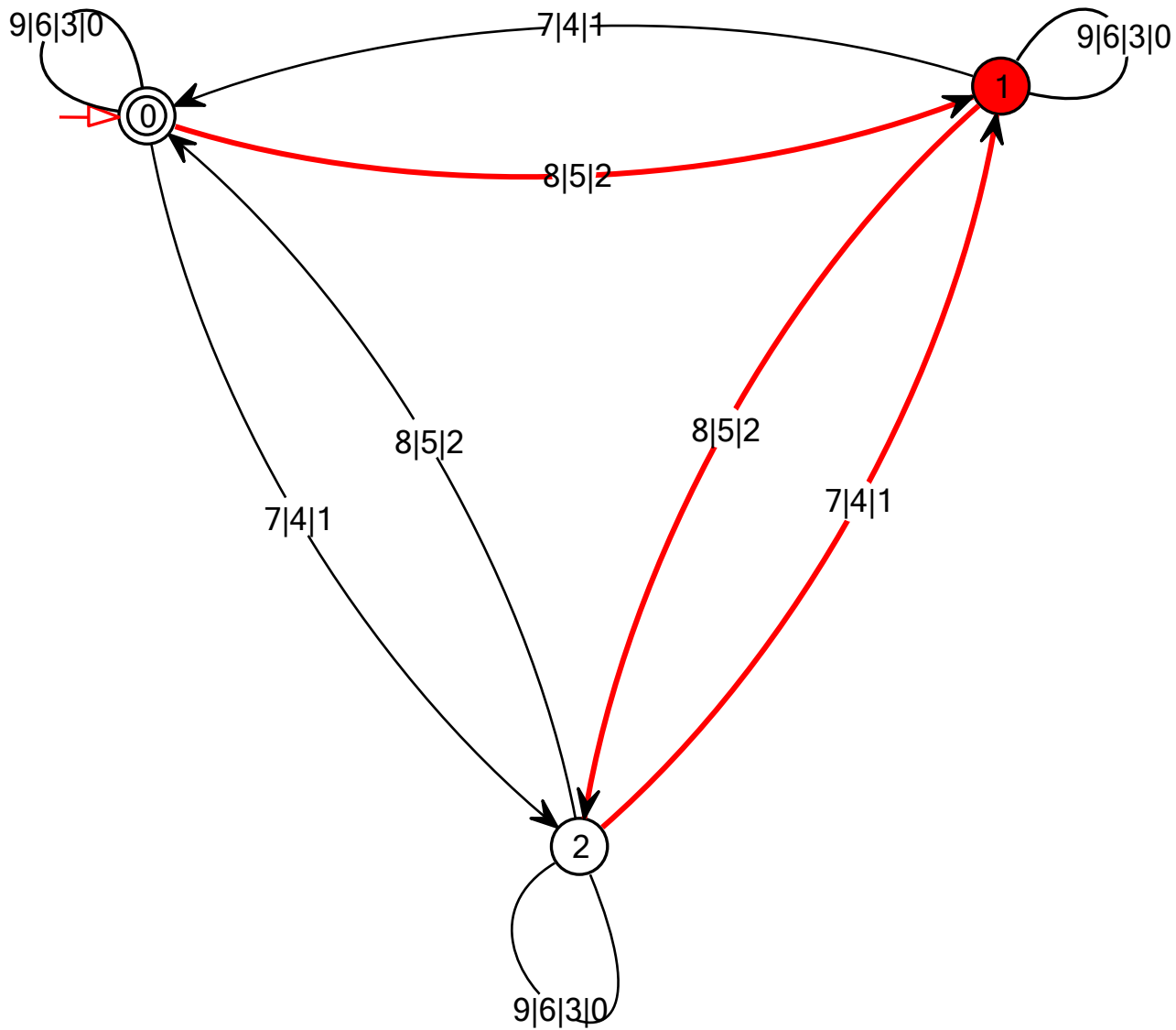


reconnaître **2**5170462

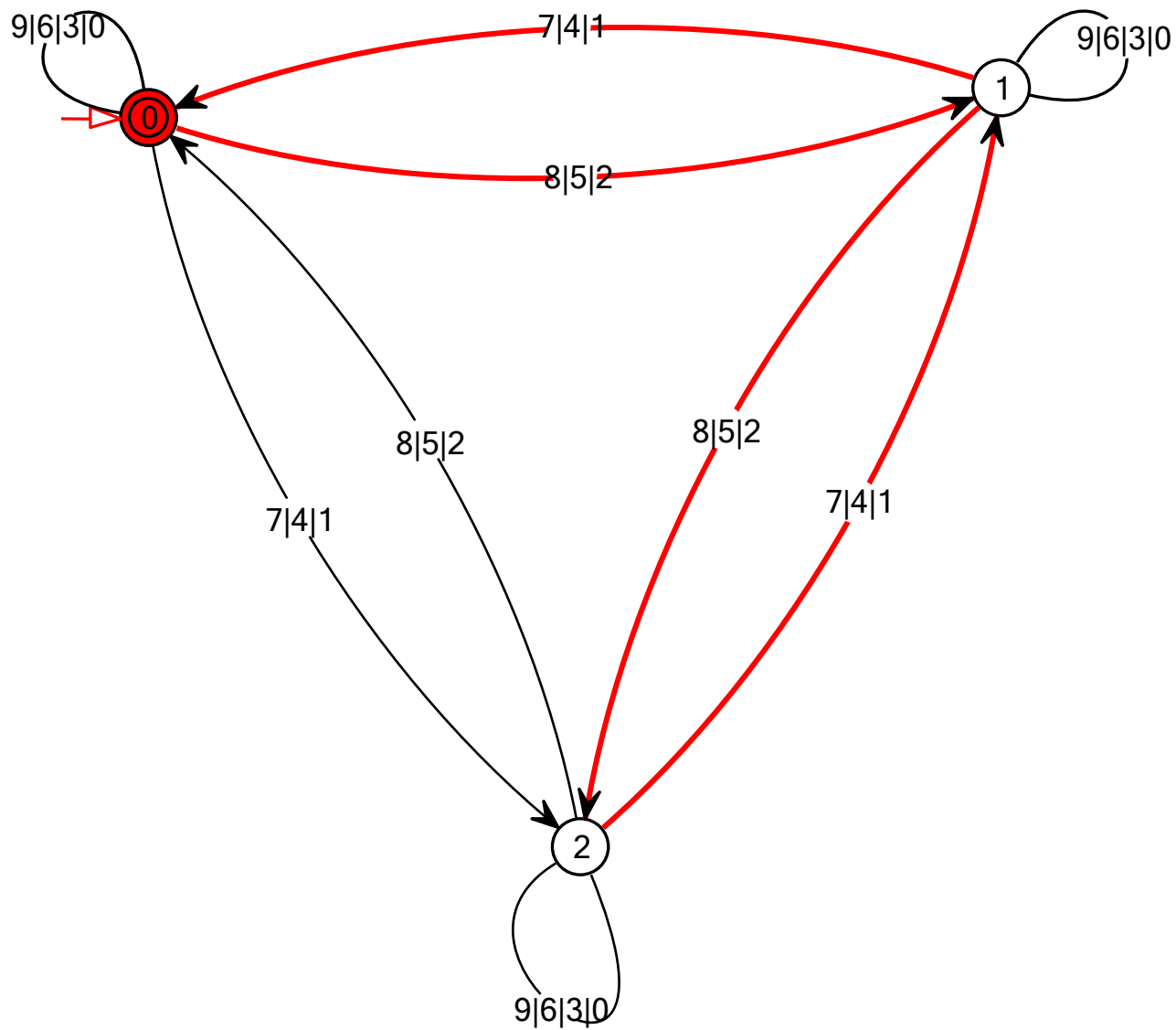




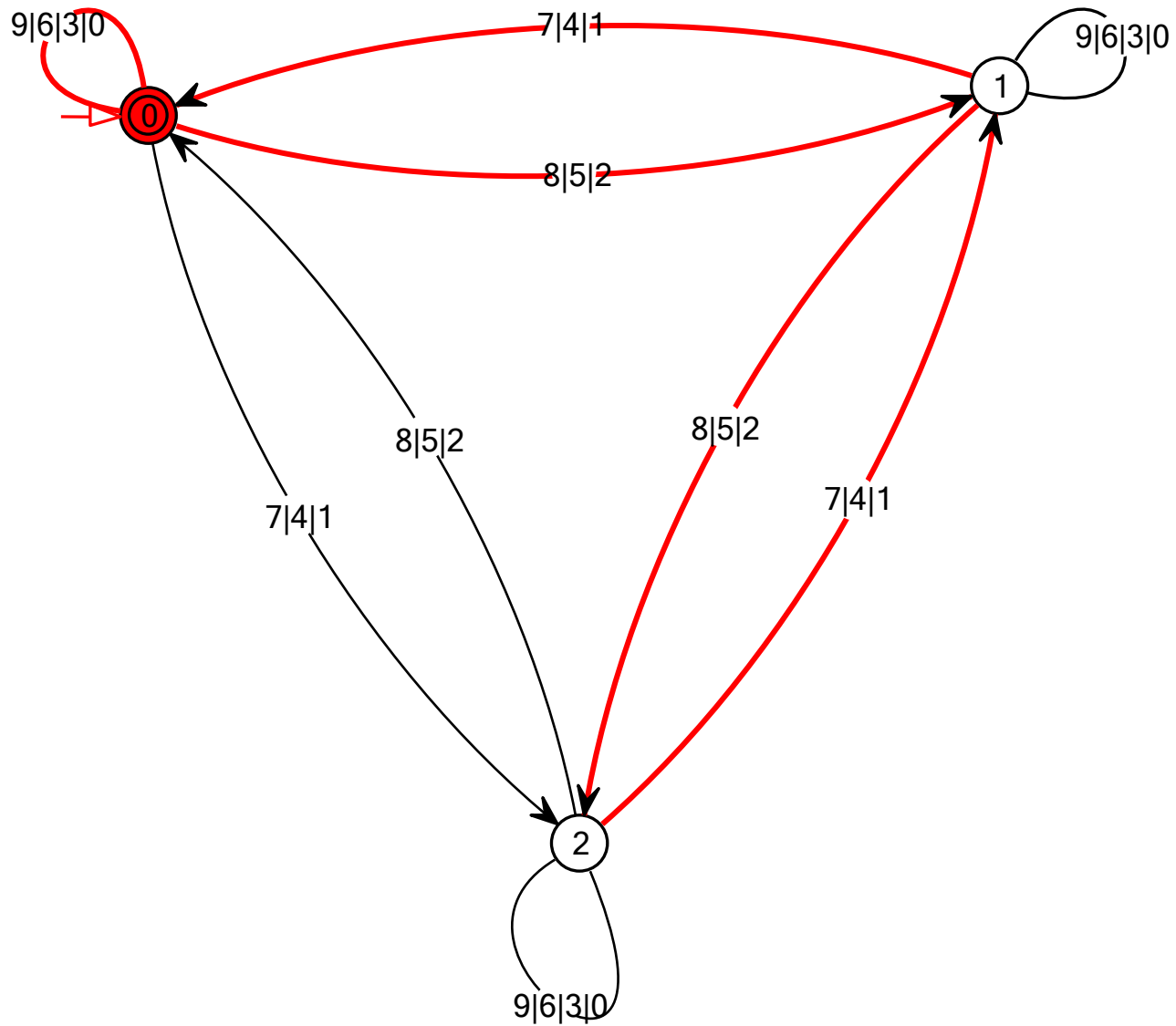
reconnaître 25170462



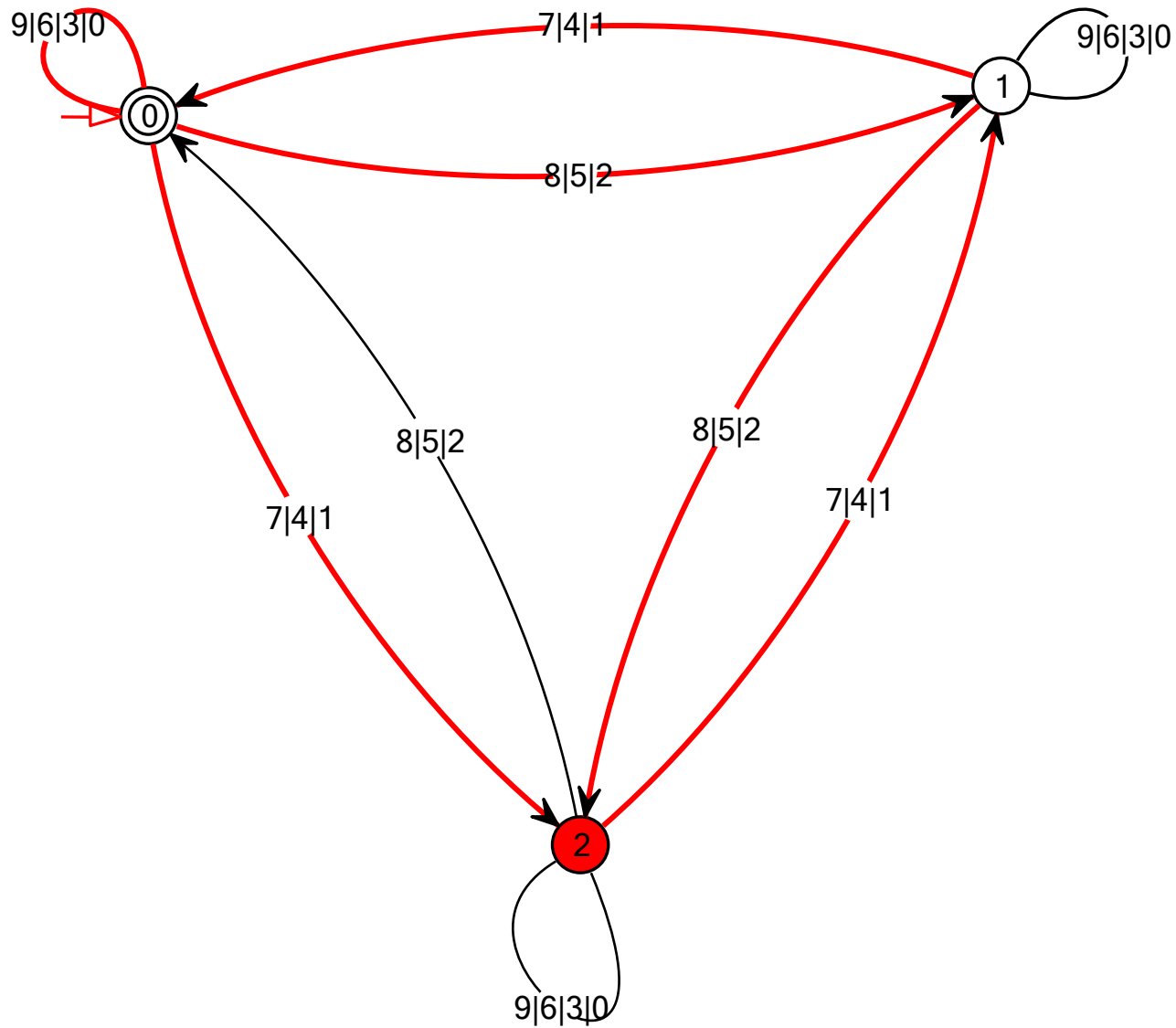
reconnaître 25170462



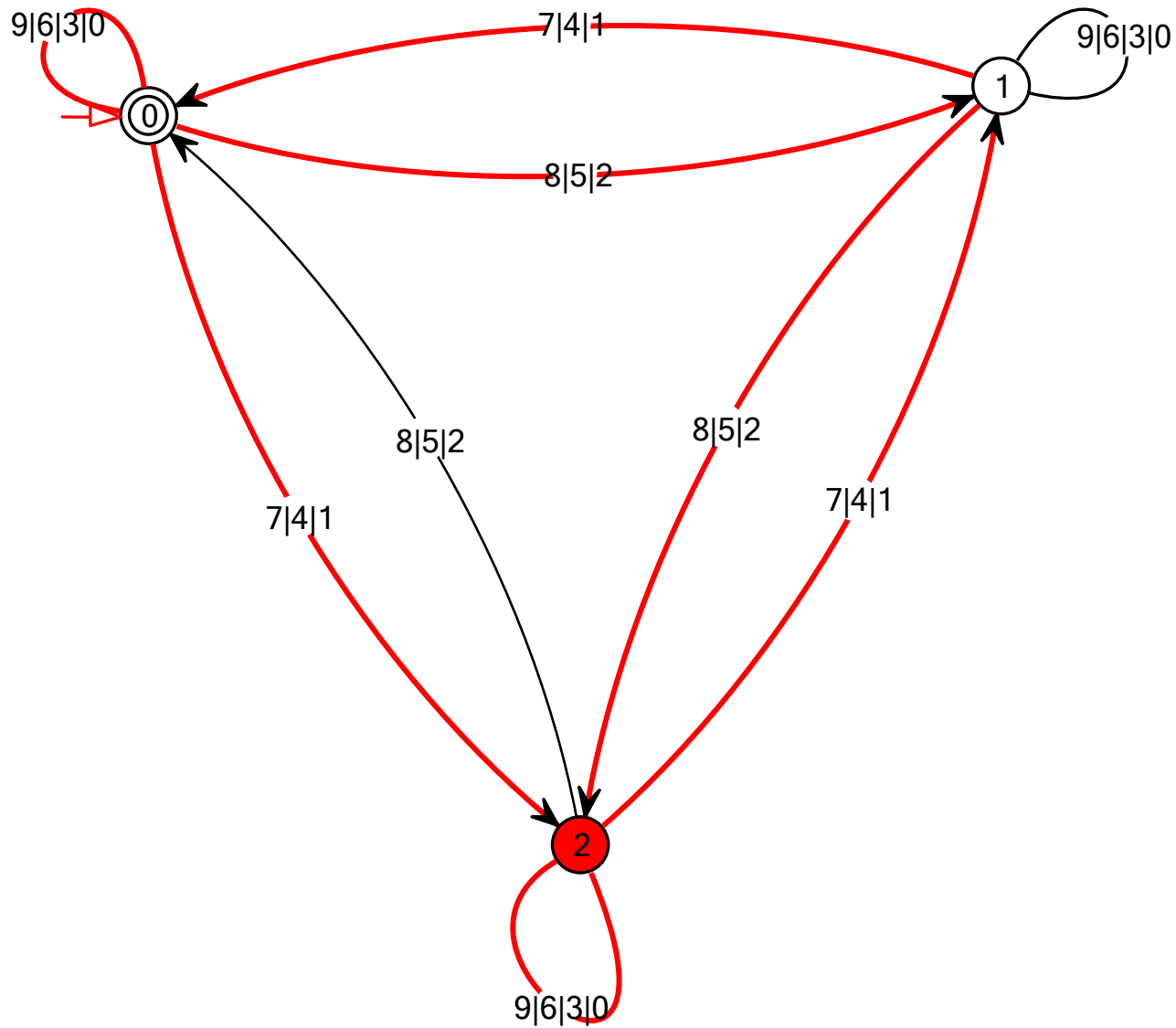
reconnaître 25170462



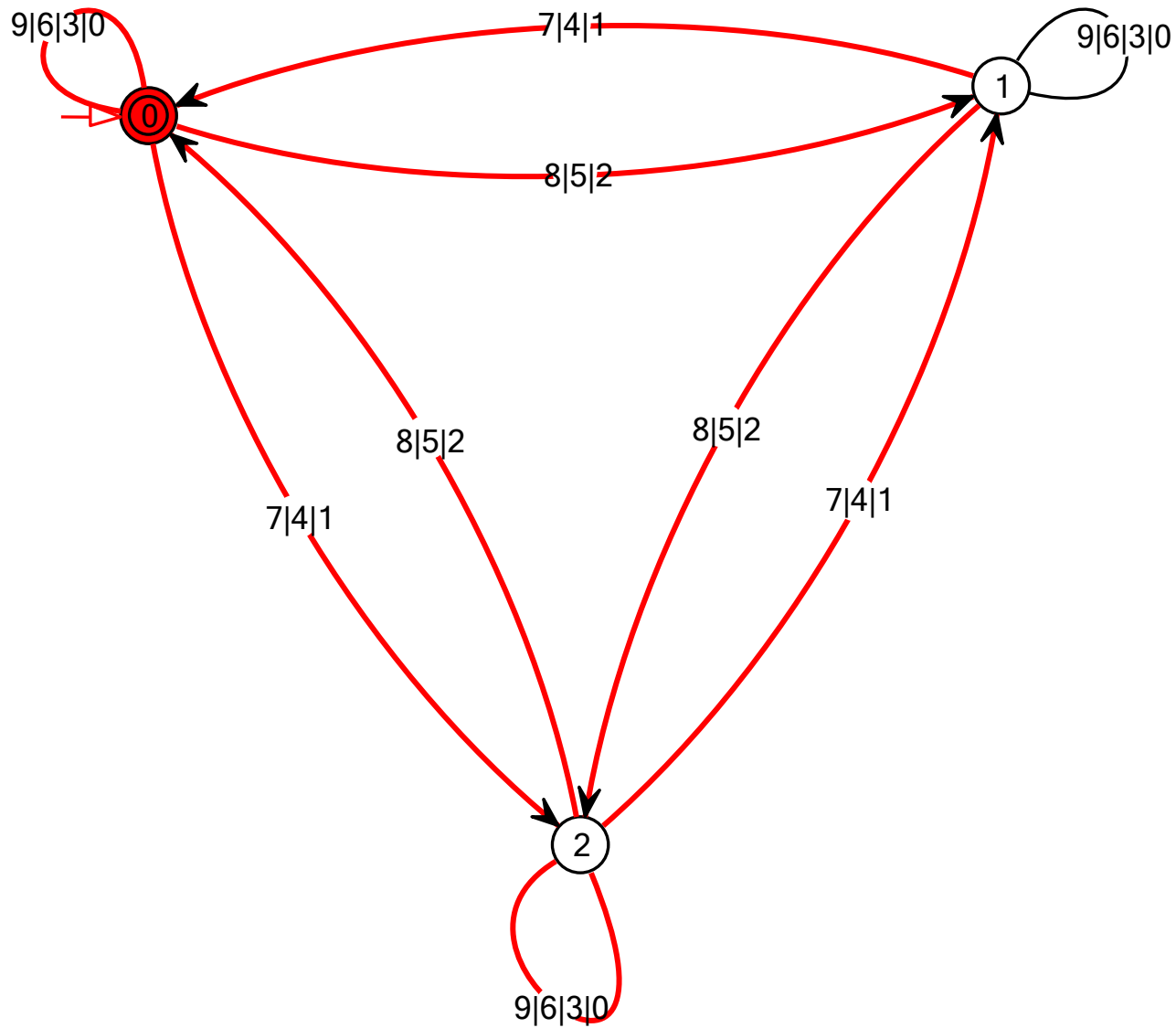
reconnaître 25170462



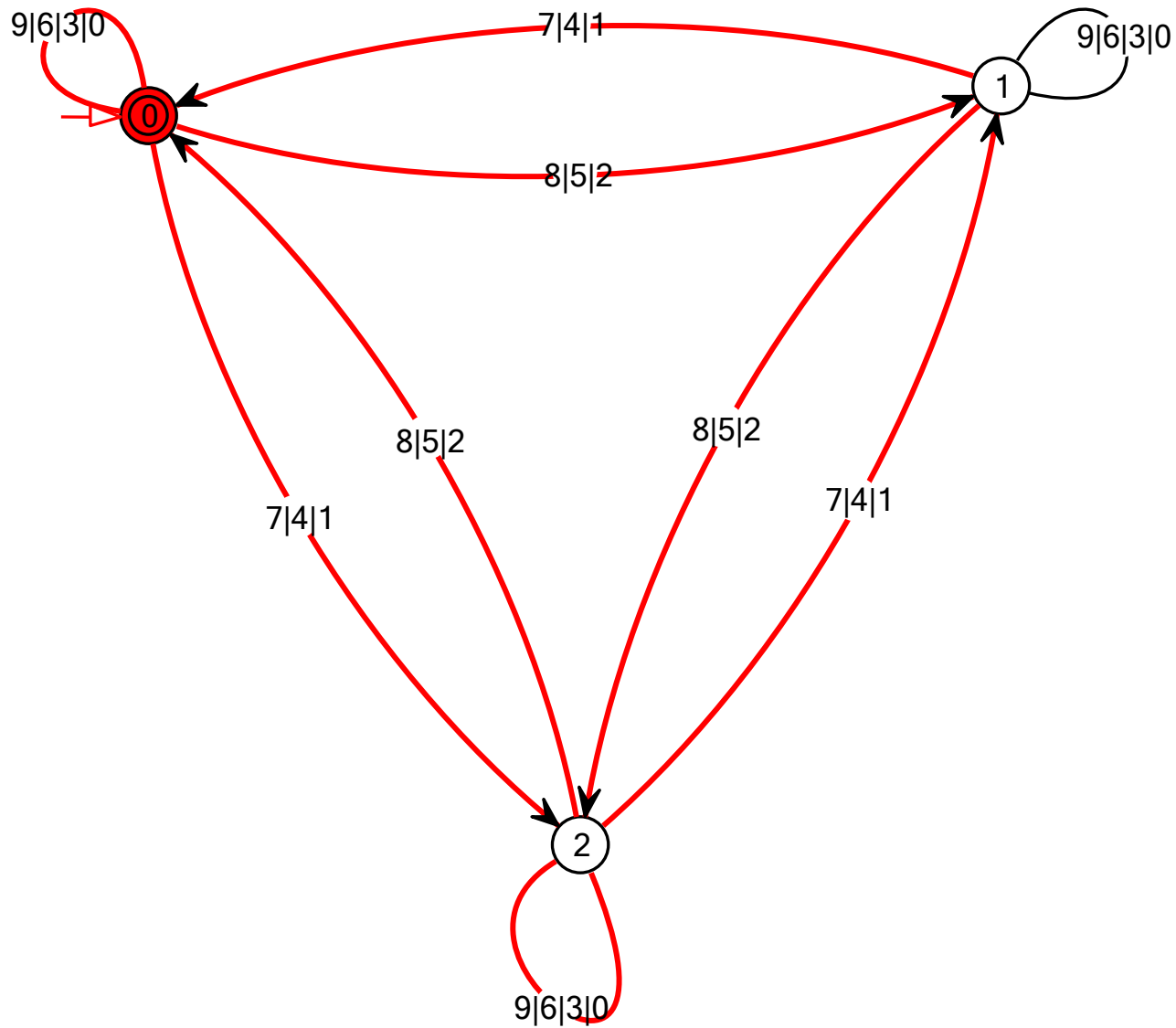
reconnaître 25170462



reconnaître 25170462



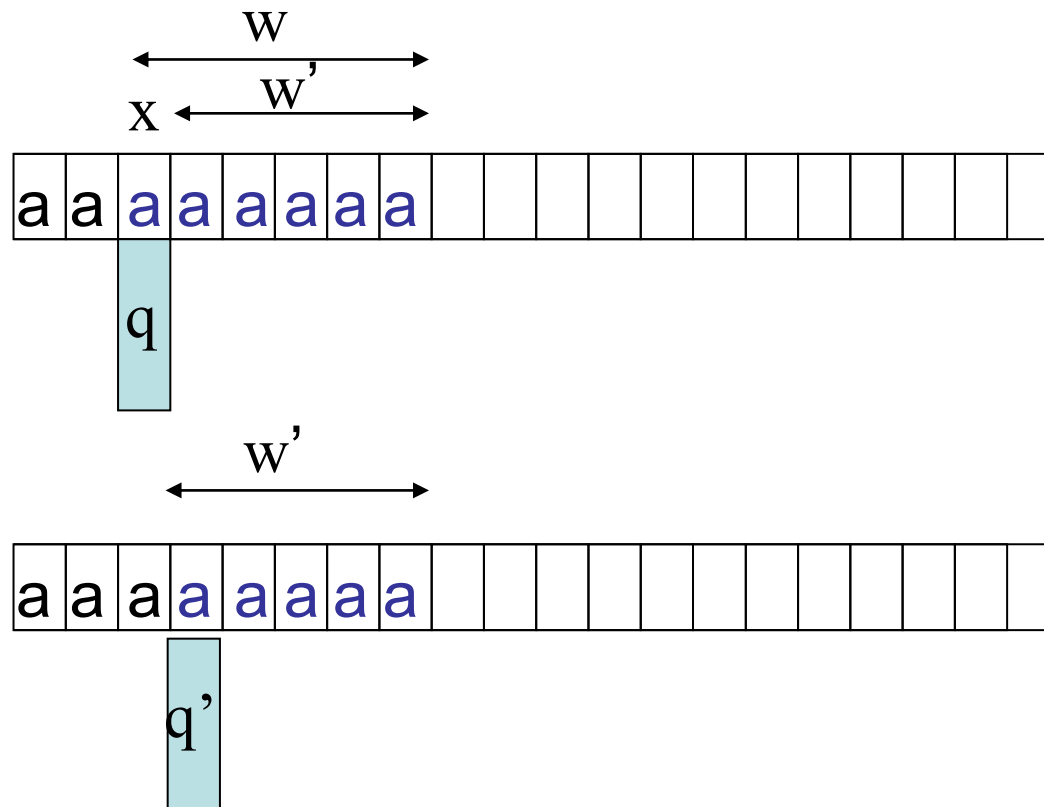
reconnaître 25170462



reconnaître 25170462



- **Configuration** : couple  $(q, w)$  avec  $q \in Q$  et  $w \in \Sigma^*$   
( $q$  représente l'état courant et  $w$  le mot qui reste à lire sur le ruban)
- **Succession immédiate**  
 $(q', w')$  suit immédiatement  $(q, w)$ :  
$$(q, w) \rightarrow (q', w')$$
  
si et seulement si
  - $w = xw'$  avec  $x \in \Sigma$
  - $\delta(q, x) = q'$



- **Succession :**

$$(q, w) \rightarrow^* (q', w')$$

si et seulement si

il existe un entier  $n \geq 0$  et une suite  $(q_0, w_0), (q_1, w_1), \dots, (q_n, w_n)$

telle que:

- $(q_0, w_0) = (q, w)$
- $(q_n, w_n) = (q', w')$
- $(q_0, w_0) \rightarrow (q_1, w_1) \rightarrow \dots \rightarrow (q_n, w_n)$

- **le cas particulier  $n = 0$** , la suite se réduit alors à  $(q_0, w_0)$  (réflexivité)

- **Mot accepté par A** =  $\langle \Sigma, Q, q_0, F, \delta \rangle$   
 $w \in \Sigma^*$  est accepté par A  
 si et seulement si  
 $(q_0, w) \rightarrow^* (q, \varepsilon)$  avec  $q \in F$
- **Langage accepté par A** :  $L(A)$  défini par  
 $L(A) = \{w \in \Sigma^*; w \text{ accepté par A}\}$
- $\rightarrow^*$  : fermeture réflexive transitive
- $\rightarrow^+$  fermeture transitive non réflexive

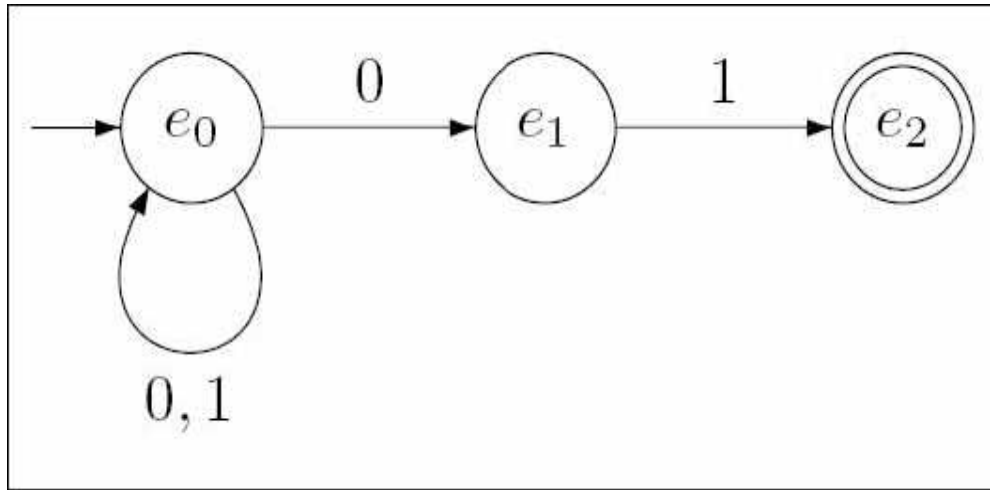
- Un automate fini sur un alphabet  $\Sigma$  est **complet** si pour chaque symbole  $x \in \Sigma$ , et chaque état  $q$ , il existe au moins une transition étiquetée  $x$  qui quitte  $q$ .
- Un automate fini sur un alphabet  $\Sigma$  est **non ambiguë** si pour chaque symbole  $x \in \Sigma$ , et chaque état  $q$ , il existe au plus une transition étiquetée  $x$  qui quitte  $q$ .
- Un automate est dit **déterministe** ssi il est complet et non ambiguë.

# Automates finis non déterministes

- Un **AFN** (Automate Fini Non déterministe) est un quintuplet  $A = (Q, \Sigma, \delta, q_0, F)$  tel que
  - $Q$  est l'ensemble fini d'états
  - $\Sigma$  est un vocabulaire (d'entrée) fini
  - $\delta : Q \times \Sigma \rightarrow \mathcal{P}(Q)$  est une fonction dite de transition
  - $q_0 \in Q$  est l'état initial
  - $F \subseteq Q$  est l'ensemble des états terminaux (ou d'acceptation)

Exemple:

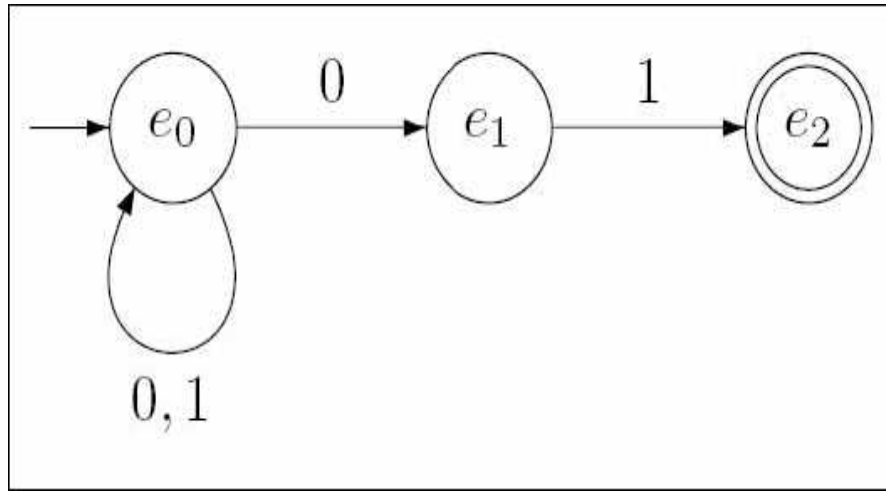
$A = (\{e_0, e_1, e_2\}, \{0, 1\}, \rightarrow, e_0, \{e_2\})$



- Le **prolongement** de  $\delta$  noté  $\delta^*$  à  $Q \times \Sigma^*$  est défini de la façon suivante :
  - $\forall q \in Q, \delta^*(q, \varepsilon) = \{q\}$
  - $\forall q \in Q, \forall x \in \Sigma, \forall w \in \Sigma^*,$   

$$\delta^*(q, wx) = \bigcup_{e \in \delta^*(q, w)} \delta(e, x)$$
- Un mot est **accepté** par l'automate lorsqu'au moins un des états de sortie est terminal



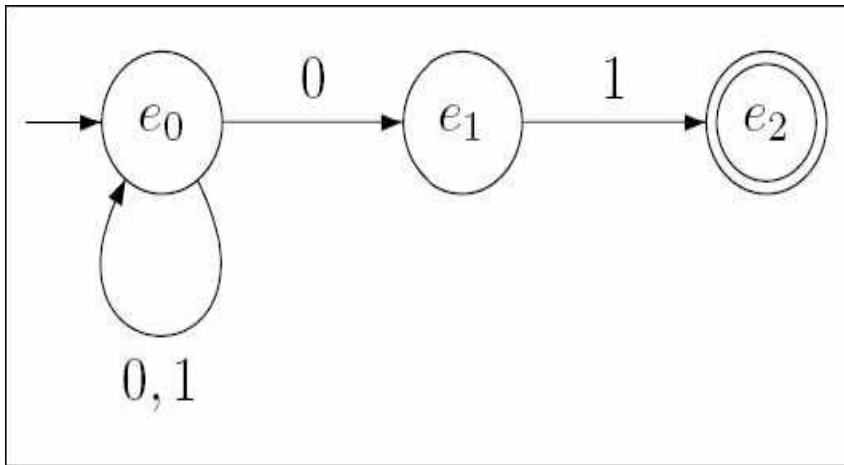


- Exemple – chaîne d'entrée 00101

- $\delta^*(e_0, \varepsilon) = \{e_0\}$
- $\delta^*(e_0, 0) = \delta(e_0, 0) = \{e_0, e_1\}$
- $\delta^*(e_0, 00) = \delta(e_0, 0) \cup \delta(e_1, 0) = \{e_0, e_1\} \cup \emptyset = \{e_0, e_1\}$
- $\delta^*(e_0, 001) = \delta(e_0, 1) \cup \delta(e_1, 1) = \{e_0\} \cup \{e_2\} = \{e_0, e_2\}$
- $\delta^*(e_0, 0010) = \delta(e_0, 0) \cup \delta(e_2, 0) = \{e_0, e_1\} \cup \emptyset = \{e_0, e_1\}$
- $\delta^*(e_0, 00101) = \delta(e_0, 1) \cup \delta(e_1, 1) = \{e_0\} \cup \{e_2\} = \{e_0, e_2\}$

- Langage reconnu par un automate fini non déterministe:

$$L(A) = \{x \in V^* \mid \delta^*(q_0, x) \cap F \neq \emptyset\}$$



Exemple

$$L(A) = \{w \mid \exists y \in V^*, w = y01\}$$

- **Déterminisation d'un AFN:**

Soit un AFN  $N = (Q_N, \Sigma, \delta_N, e_0, F_N)$ . On construit un AFD  $D = (Q_D, \Sigma, \delta_D, e_0, F_D)$  comme suit:

- $Q_D = \mathcal{P}(Q_N)$
- $F_D = \{G \subseteq Q_N \mid G \cap F_N \neq \emptyset\}$
- $\forall G \subseteq Q_N (G \in \mathcal{P}(Q_N)), \forall x \in V, \delta_D(G, x) = \cup_{e \in G} \delta_N(e, x)$

- **Théorème (Equivalence AFN – AFD)**

Soit  $N = (Q_N, \Sigma, \delta_N, e_0, F_N)$  et  $D = (Q_D, \Sigma, \delta_D, e_0, F_D)$  construit comme précédemment à partir de  $N$ . On a alors  $L(D) = L(N)$

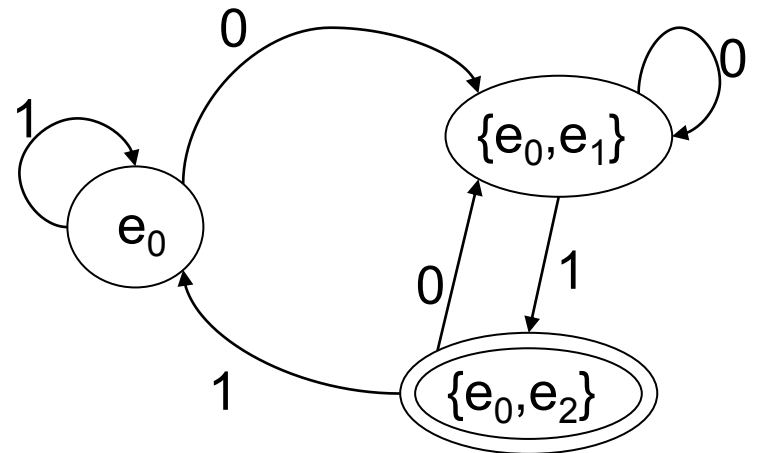
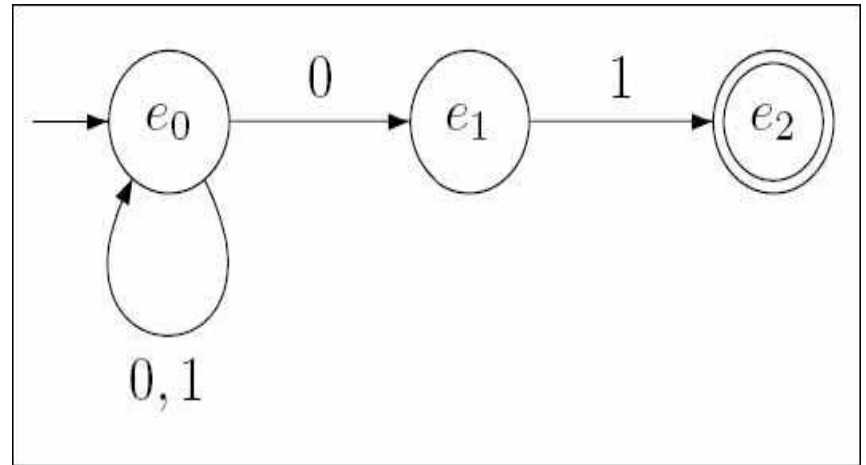
## **Principe de déterminisation:**

considérer des ensembles d'états plutôt que des états

1. Partir de l'état initial
2. Rajouter dans la table de transition tous les nouveaux états produits avec leurs transitions
3. Recommencer 2 jusqu'à ce qu'il n'y ait plus de nouvel état
4. Tous les états contenant au moins un terminal deviennent terminaux

# Example

	0	1
$\emptyset$	$\emptyset$	$\emptyset$
$\{e_0\}$	$\{e_0, e_1\}$	$\{e_0\}$
$\{e_1\}$	$\emptyset$	$\{e_2\}$
$\{e_2\}$	$\emptyset$	$\emptyset$
$\{e_0, e_1\}$	$\{e_0, e_1\}$	$\{e_0, e_2\}$
$\{e_0, e_2\}$	$\{e_0, e_1\}$	$\{e_0\}$
$\{e_1, e_2\}$	$\emptyset$	$\{e_2\}$
$\{e_0, e_1, e_2\}$	$\{e_0, e_1\}$	$\{e_0, e_2\}$



# Minimisation d'AEF

- **Principe:**

On définit des classes d'équivalence d'états par raffinements successifs. Chaque classe obtenue forme un seul même état du nouvel automate

- **Hypothèse de l'algorithme:**

AFD complet dont tous les états sont accessibles.

# Minimisation d'AEF

1. Faire deux classes: l'une contenant les états terminaux et l'autre les états non terminaux
2. S'il existe un symbole  $a$  et deux états  $e_1$  et  $e_2$  d'une même classe tels que  $\delta(e_1, a)$  et  $\delta(e_2, a)$  n'appartiennent pas à la même classe, alors créer une nouvelle classe et séparer  $e_1$  et  $e_2$
3. Recommencer 2 jusqu'à ce qu'il n'y ait plus de classes à séparer
4. Chaque classe restante forme un état du nouvel automate