Expressions régulières

Didier Le Botlan

INSA
contact.lebotlan@insa-toulouse.fr

1er semestre MIC

http://wwwdgeinew.insa-toulouse.fr/~lebotlan/index.html

Le cours

- Micro-enseignement : 1 cours, 2 TDs sur ordi

✍ Implication réelle en TD $\Rightarrow$ 10 (pas besoin de terminer).

- Soyez présents !

✍ Contrôle très rapide vendredi 11/1 matin pour augmenter sa note, ou pas.

- Présence au contrôle facultative. Travaillez en priorité les autre matières.

Plan

Plan

Rechercher dans un fichier :

- Des nombres composés de trois chiffres
- Des adresses IP
- Les balises ouvrantes XML : <foo>
- ...

Exemples :

- Les crochets :

$$[A-F] \qquad [ABC] \qquad [CBA] \qquad [AAABC] \qquad [0-99]$$

- Le point : .[.]..[..]
- Étoile , Plus, ? :

$$[0-9]^* \qquad 1^+ \qquad [1]^+ \qquad [1+2*] \qquad [1][23]^? \qquad A.*$$

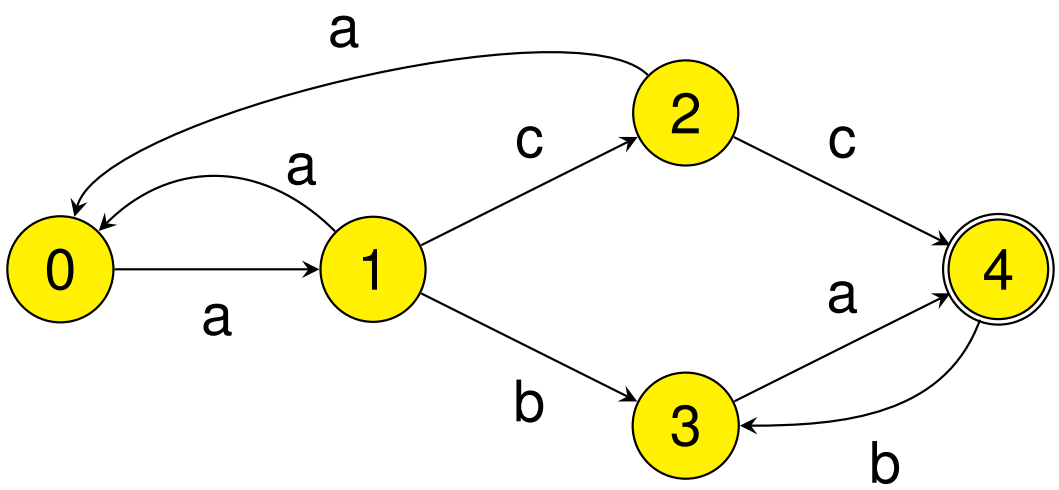
$$A.^+ \qquad A.^?$$

- Union $[0-9]+ \mid [a-z]+$
- Le crochet complémentaire $[\hat{}AB]$

Plan

Définition

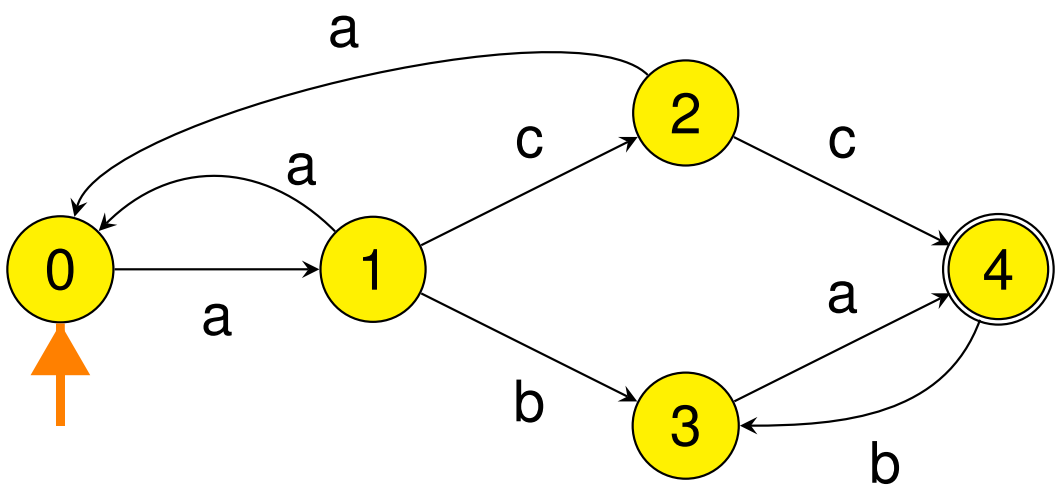
Définition : un automate à états finis, c'est ça :



Il est composé **d'états** (2), **de transitions** ($\xrightarrow{b}$),
et d'un ou plusieurs **états finaux** (4).

En activité

Fonctionnement de l'automate

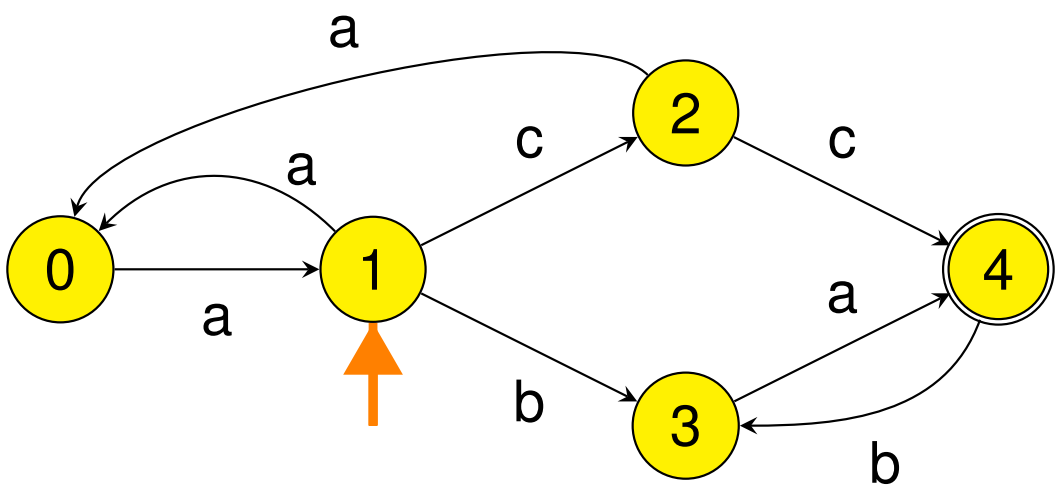


Entrée :

a c a a b a b a

En activité

Fonctionnement de l'automate

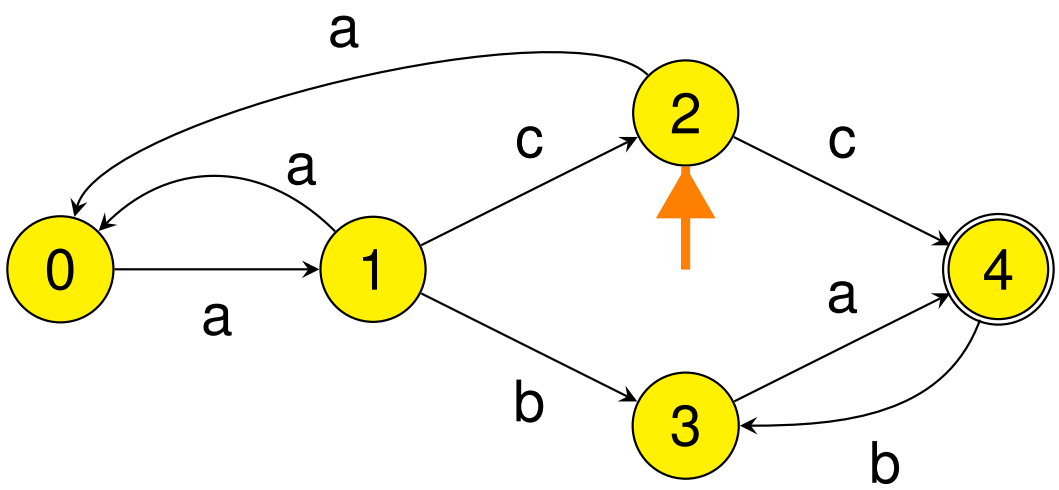


Entrée :

a c a a b a b a

En activité

Fonctionnement de l'automate

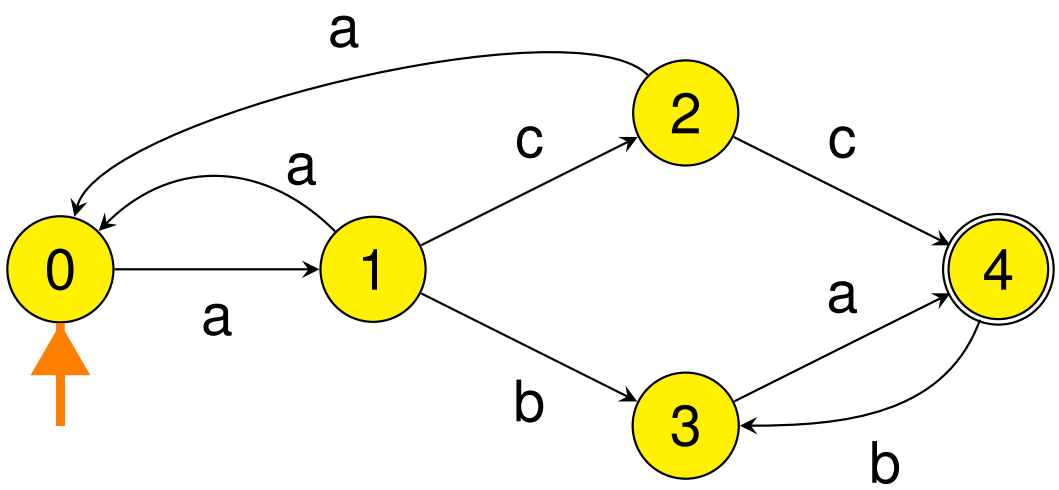


Entrée :

a c a a b a b a

En activité

Fonctionnement de l'automate

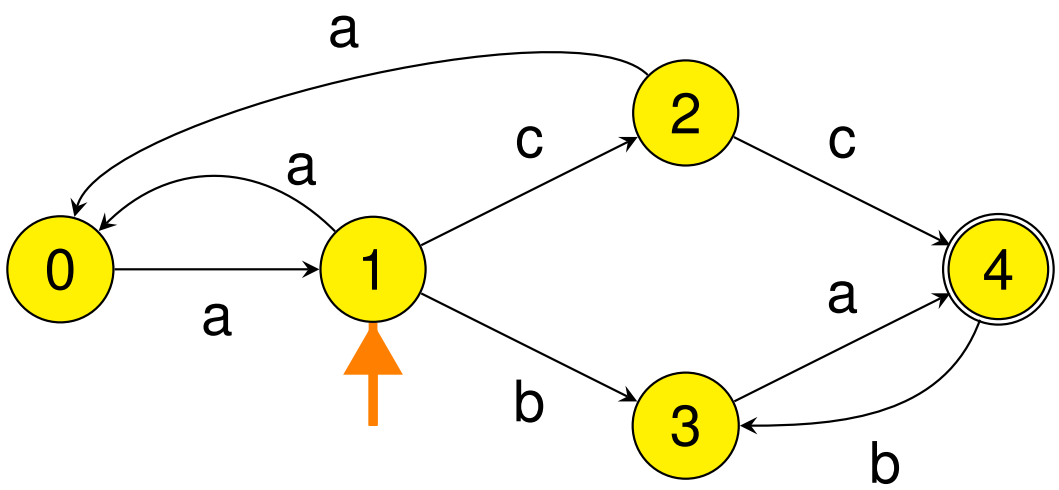


Entrée :

a c a a b a b a

En activité

Fonctionnement de l'automate

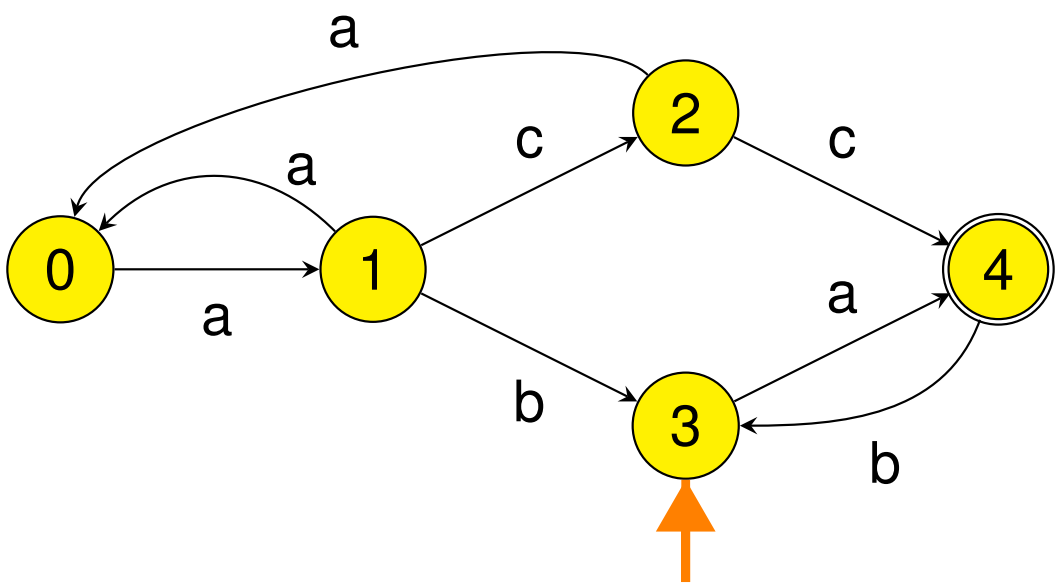


Entrée :

a c a a b a b a

En activité

Fonctionnement de l'automate

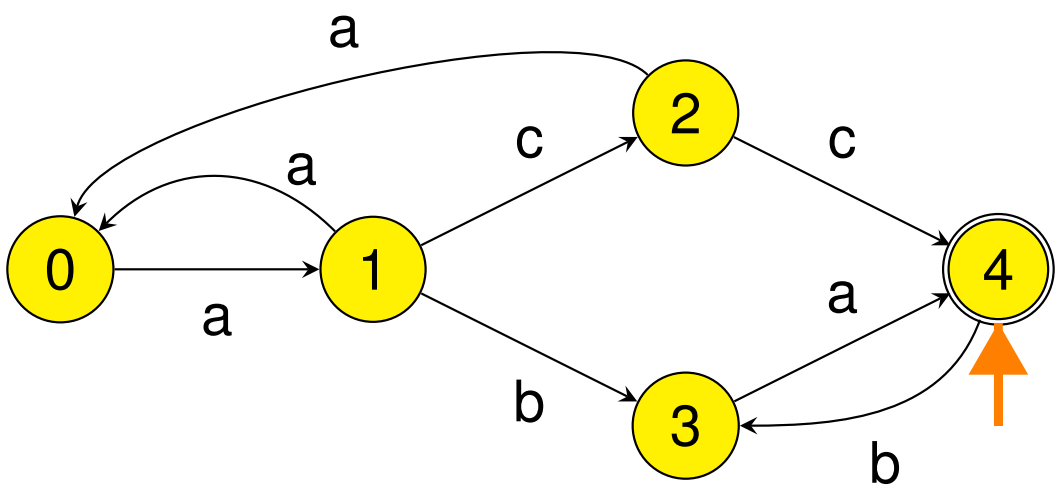


Entrée :

a c a a b a b a

En activité

Fonctionnement de l'automate

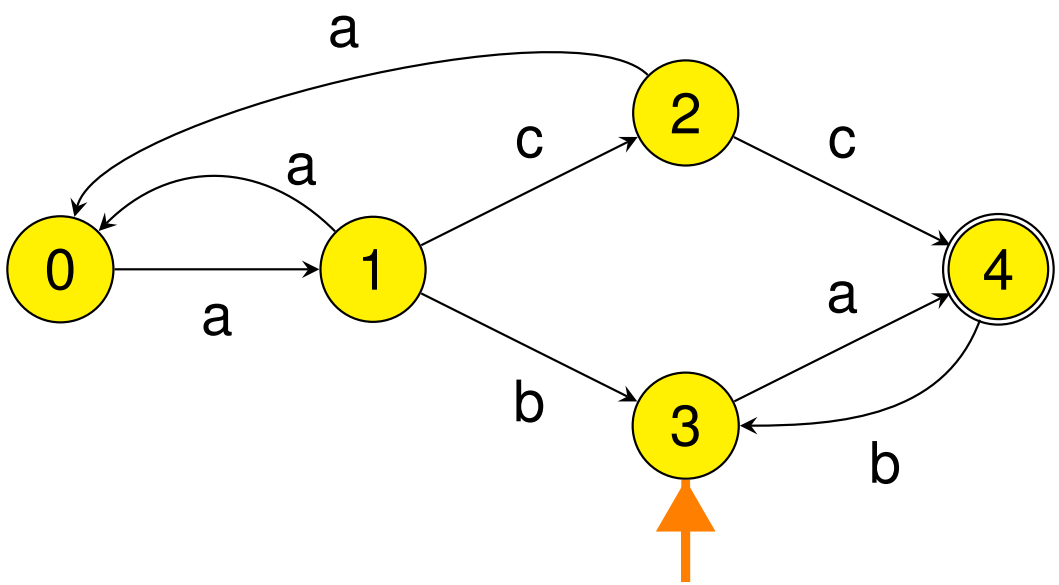


Entrée :

a c a a b a b a

En activité

Fonctionnement de l'automate

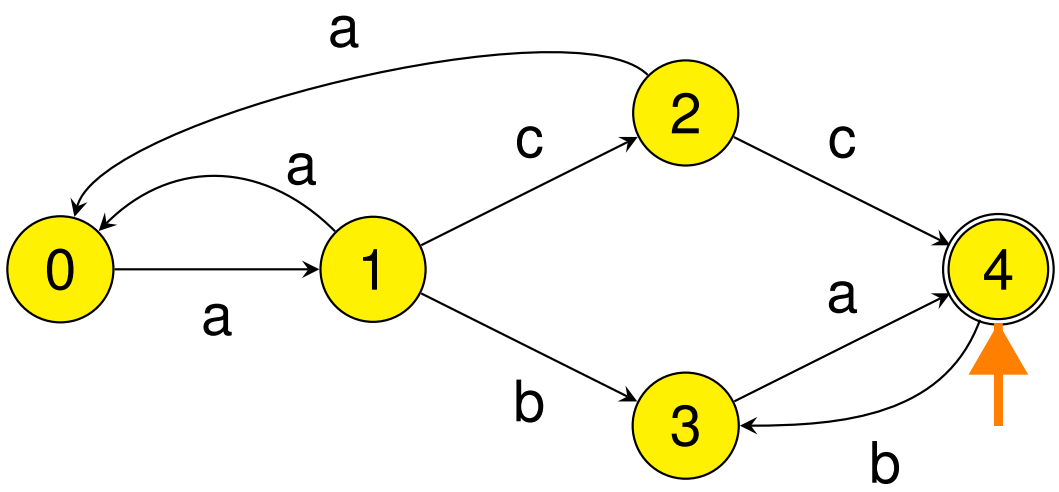


Entrée :

a c a a b a b a

En activité

Fonctionnement de l'automate



Entrée :

a c a a b a b a

Mathématiquement

Définition : **Automate déterministe**

Un **automate à états finis déterministe** est constitué par :

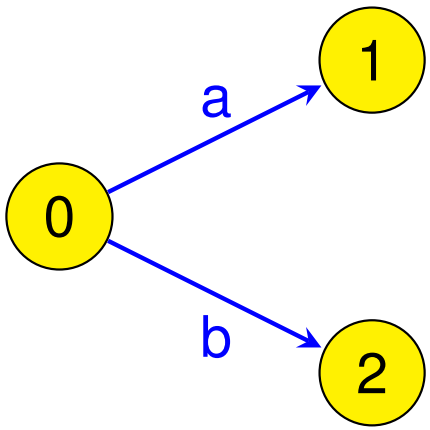
- Un ensemble d'états S
- Un alphabet Σ
- Une fonction de transition $T : S \times \Sigma \to S$
- Un état de départ $s \in S$
- Un ensemble d'états d'arrivée $A \subseteq S$ aussi appelés états finaux ou états acceptants.

Dans le cas d'un **automate non déterministe**,

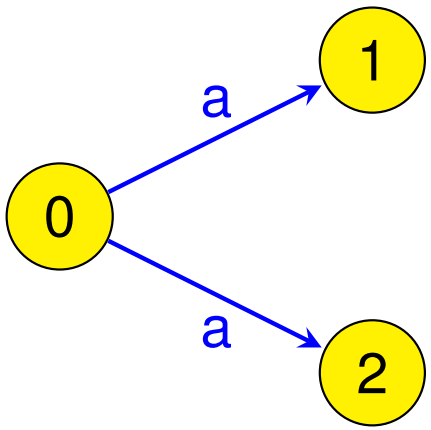
- $T : S \times (\Sigma \cup \{\epsilon\}) \to \mathcal{P}(S)$

En clair

L'automate est déterministe si et seulement si pour chaque état, les transitions qui en partent sont étiquetées différemment (et pas avec ϵ).



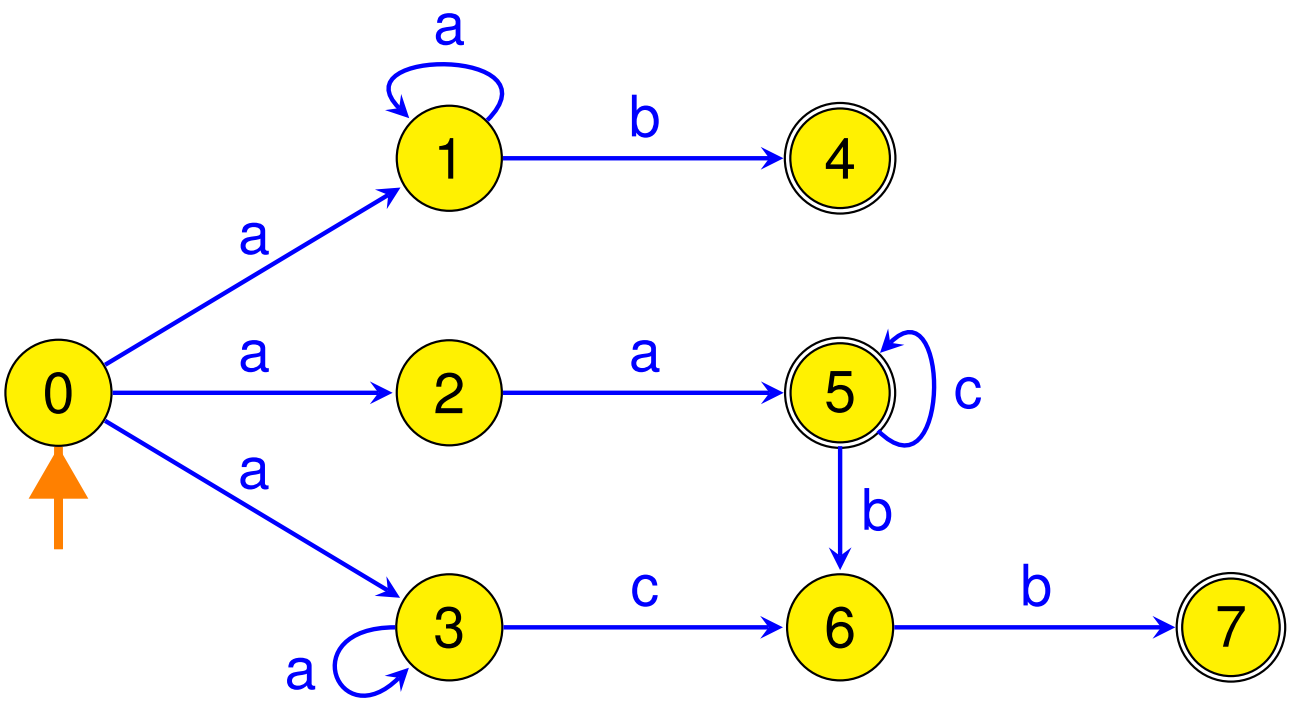
Déterministe



Non déterministe

Automate non déterministe

Exécution d'un automate non-déterministe

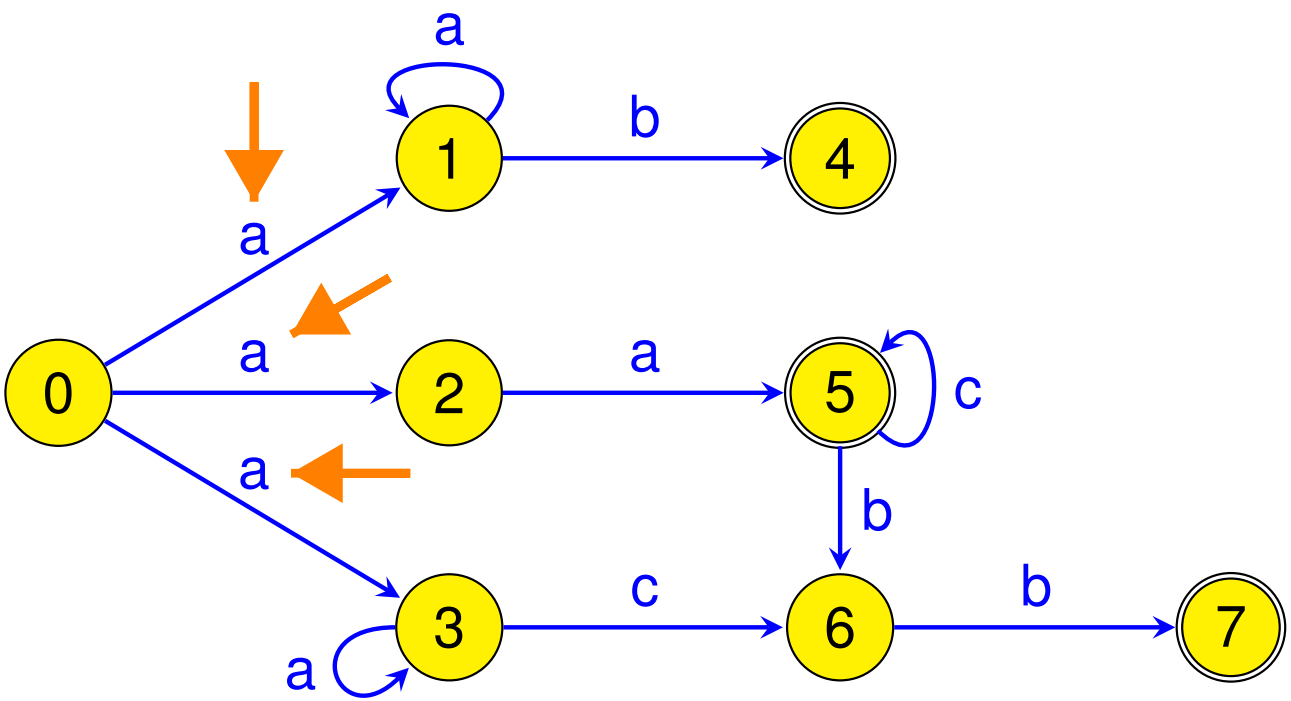


Entrée :

a a c b

Automate non déterministe

Exécution d'un automate non-déterministe

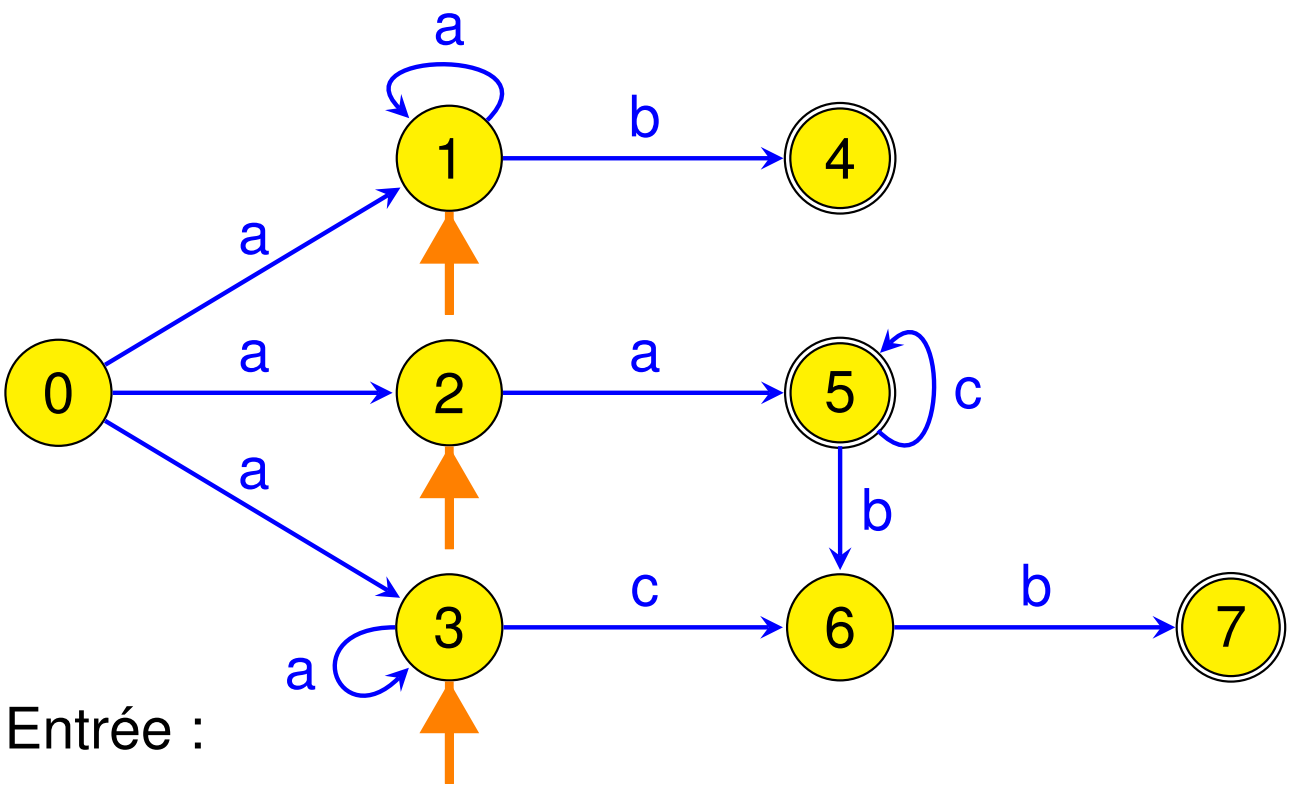


Entrée :

a a c b

Automate non déterministe

Exécution d'un automate non-déterministe

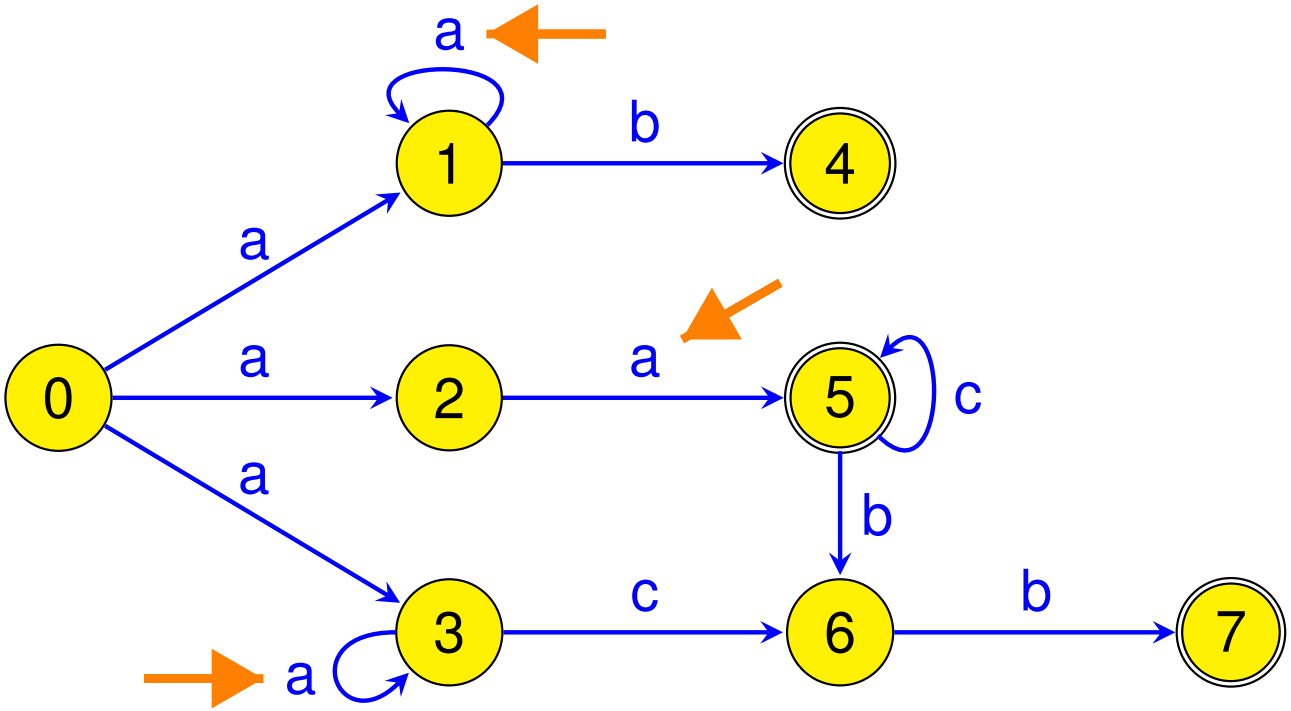


Entrée :

a a c b

Automate non déterministe

Exécution d'un automate non-déterministe

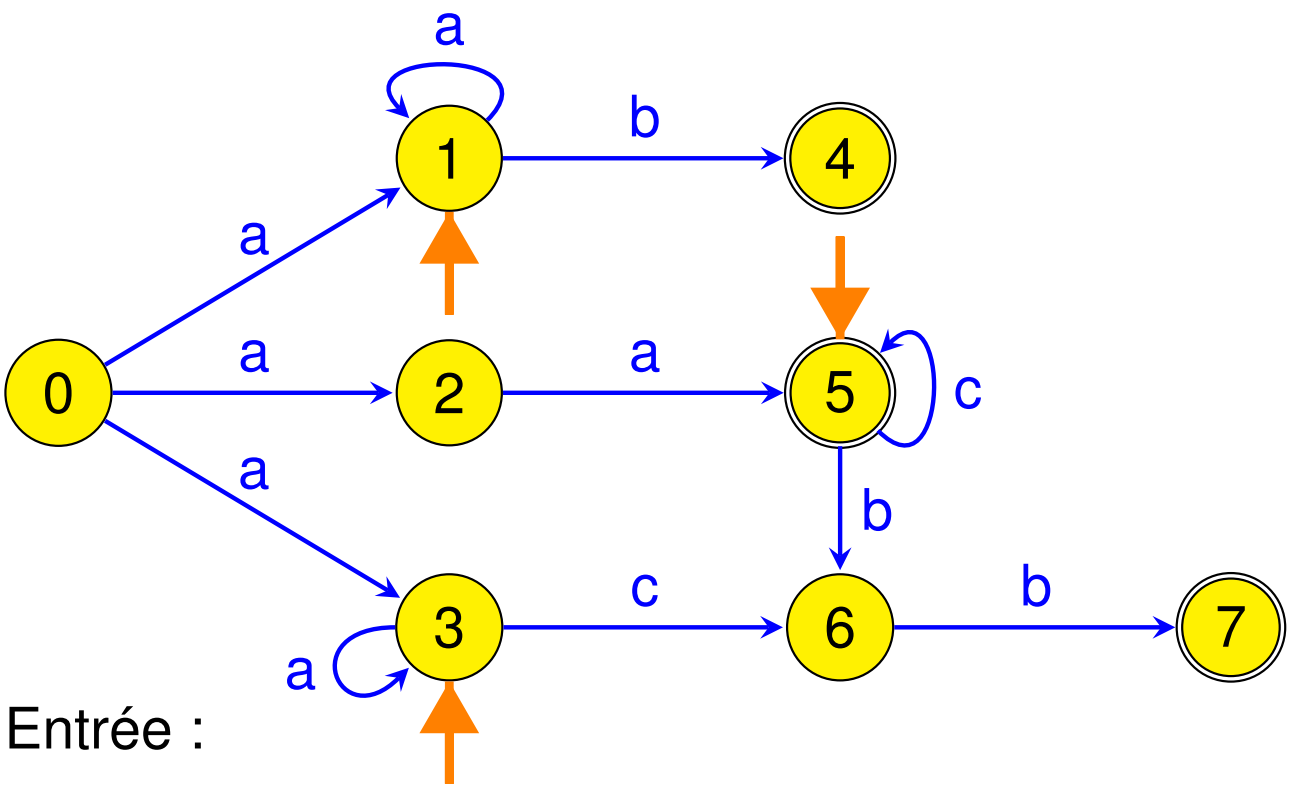


Entrée :

a a c b

Automate non déterministe

Exécution d'un automate non-déterministe

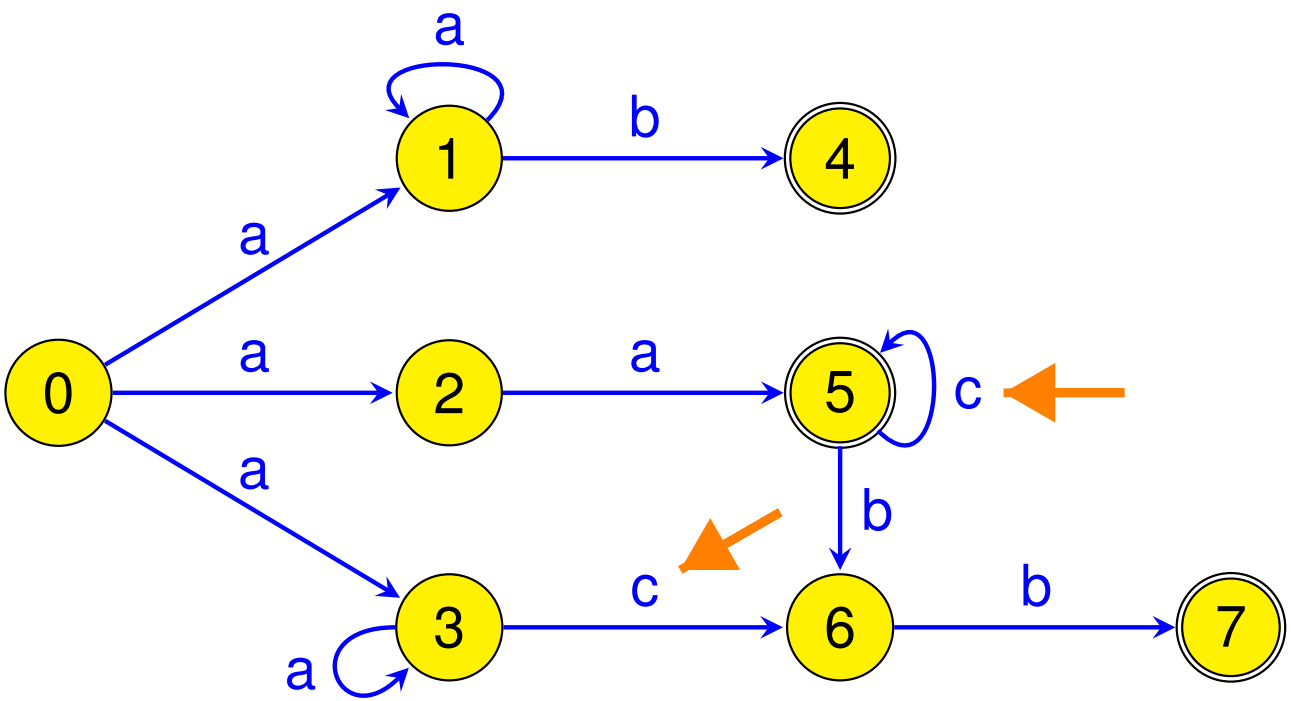


Entrée :

a a c b

Automate non déterministe

Exécution d'un automate non-déterministe

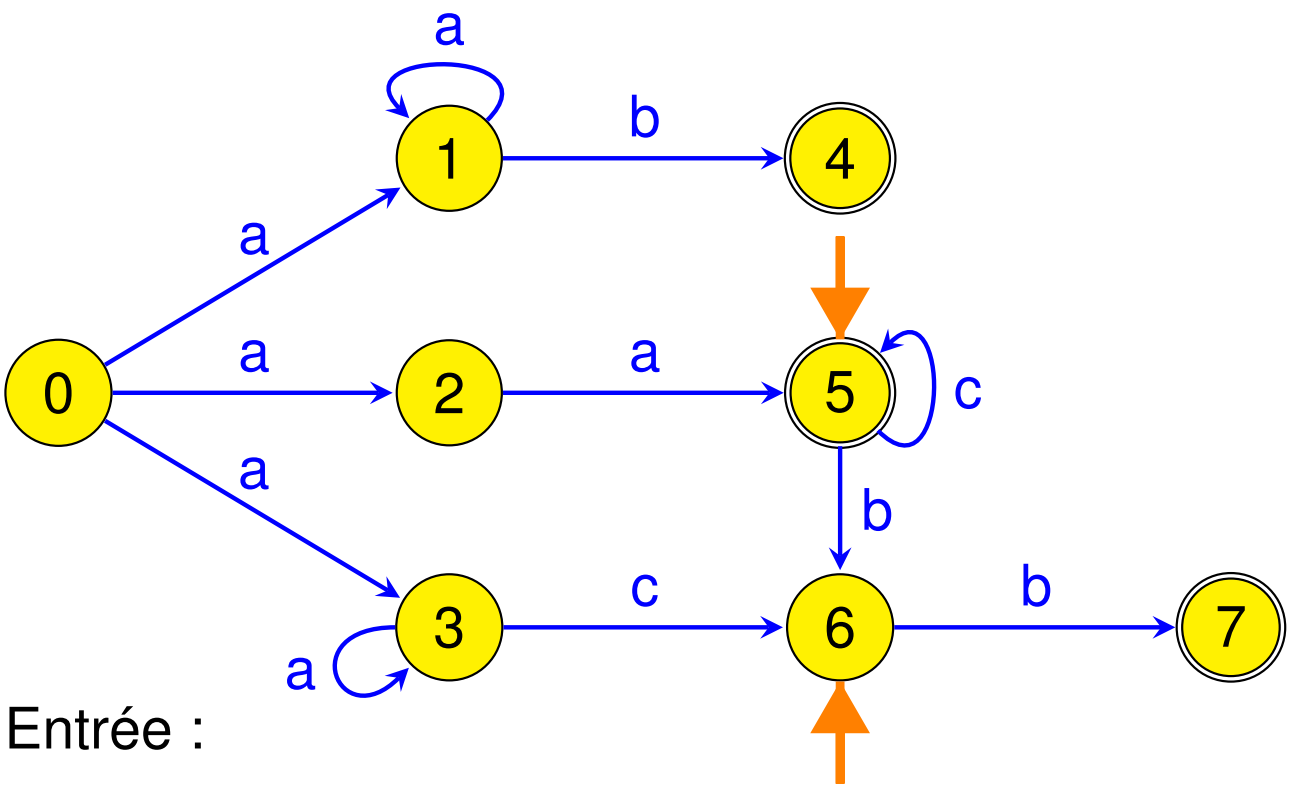


Entrée :

a a c b

Automate non déterministe

Exécution d'un automate non-déterministe

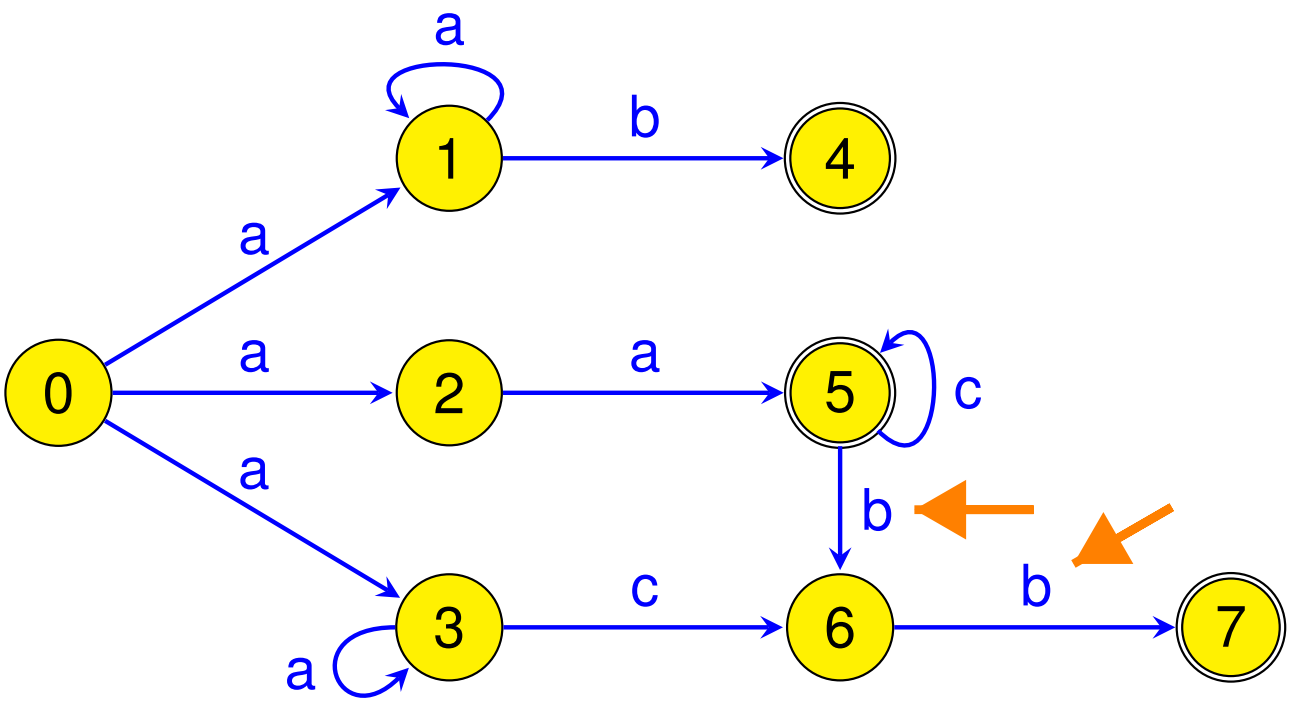


Entrée :

a a c b

Automate non déterministe

Exécution d'un automate non-déterministe

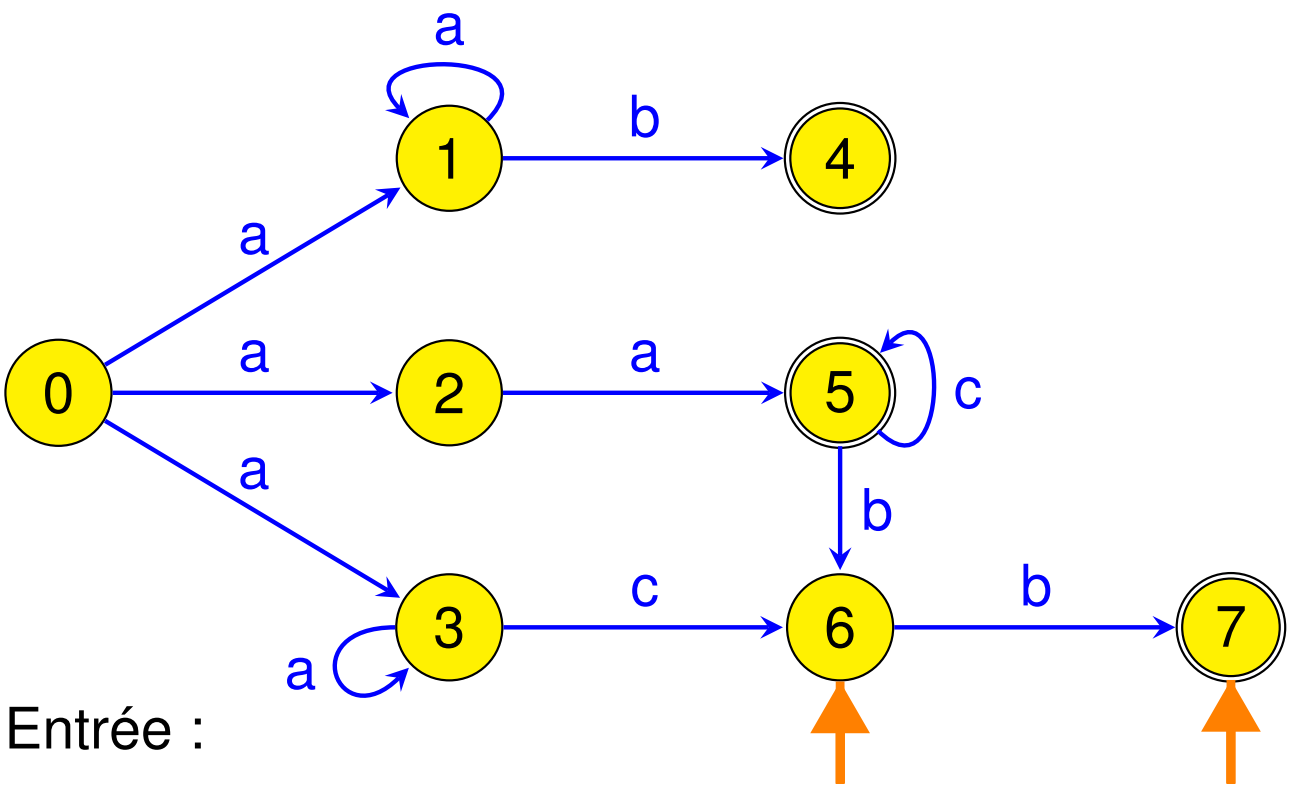


Entrée :

a a c b

Automate non déterministe

Exécution d'un automate non-déterministe

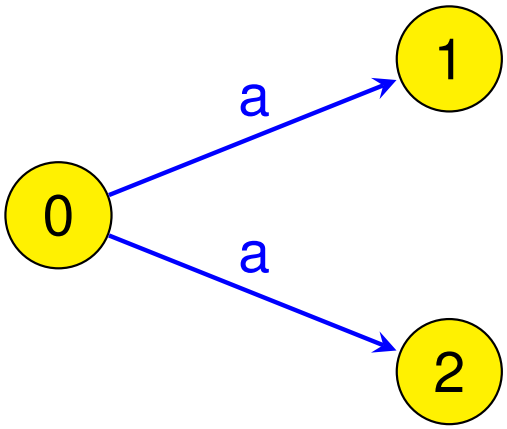


Entrée :

a a c b

Transitions Epsilons

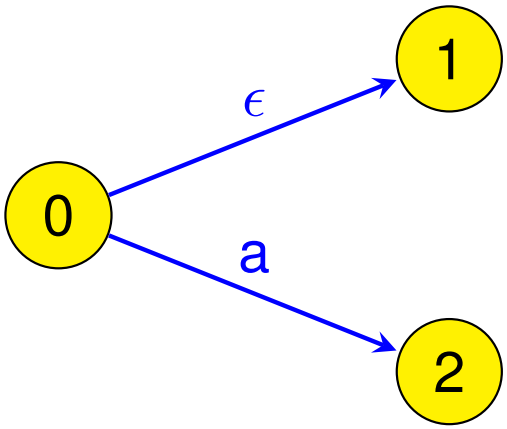
Un automate non-déterministe se complait à faire plusieurs choses en même temps.



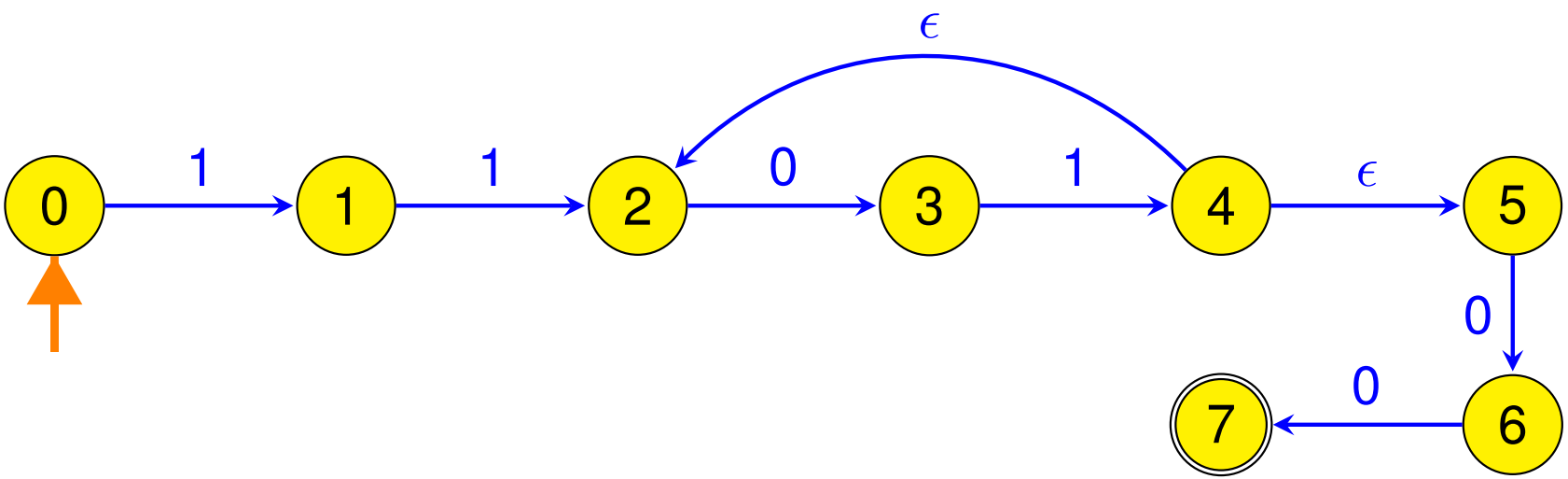
Transitions Epsilons

On peut aussi autoriser un automate non-déterministe à changer spontanément d'état (sans action).

- On dit que l'automate fait une ϵ-transition
- ϵ représente le mot vide.
- L'automate change d'état sans lire aucune entrée

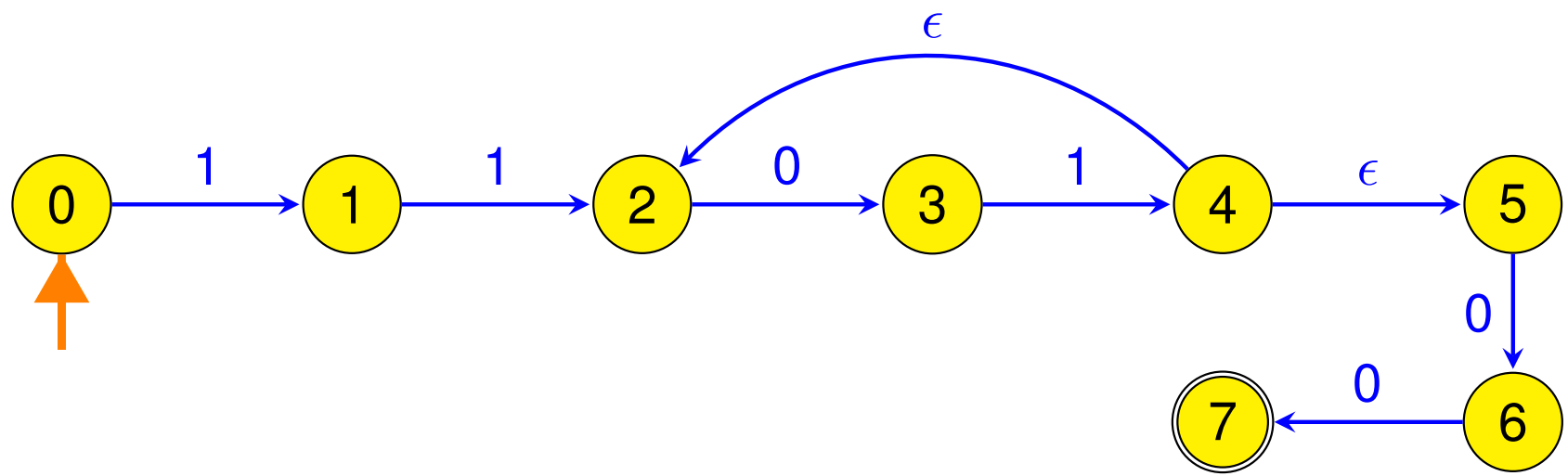


Transitions ϵ - Exemple



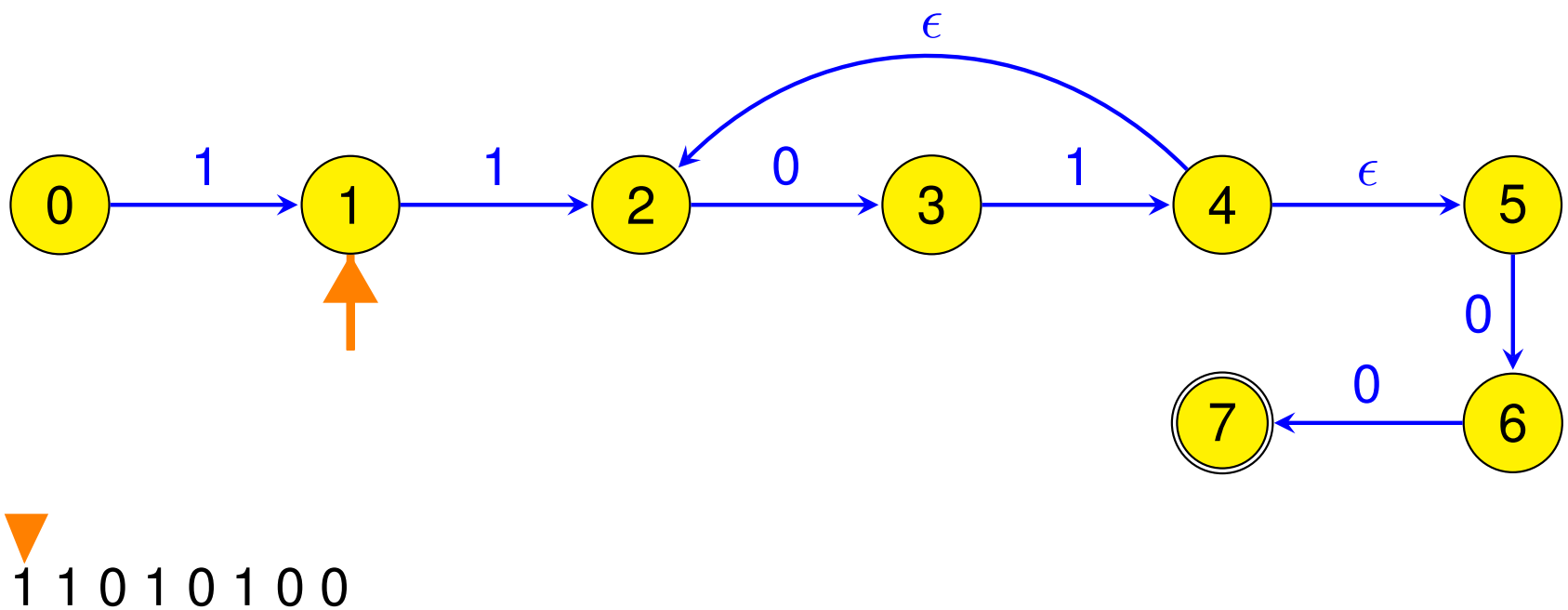
1 1 0 1 0 1 0 0

Transitions ϵ - Exemple



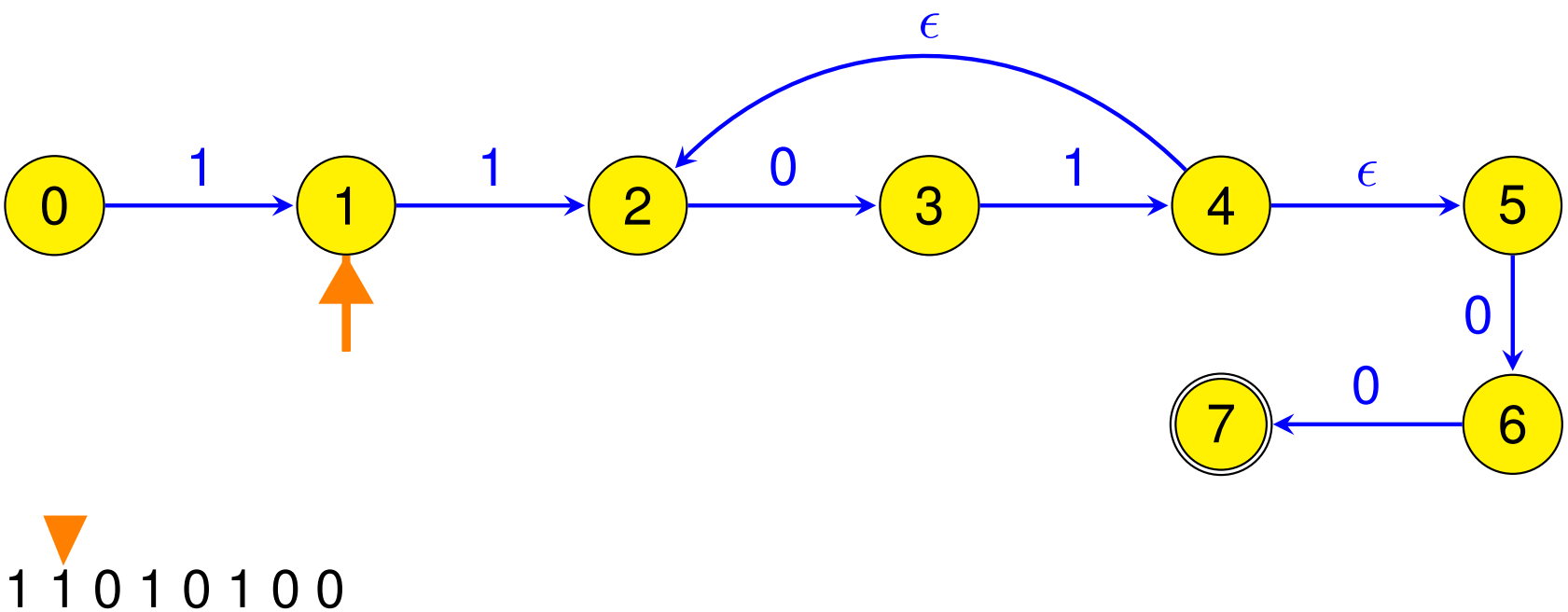
1 1 0 1 0 1 0 0

Transitions ϵ - Exemple

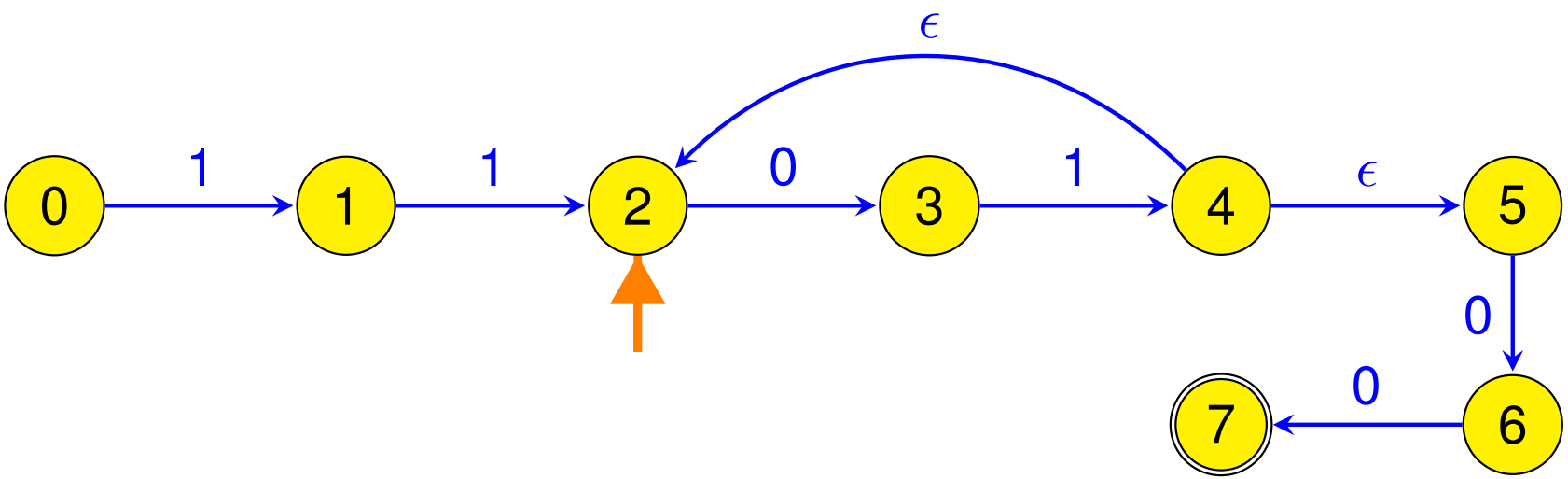


1 1 0 1 0 1 0 0

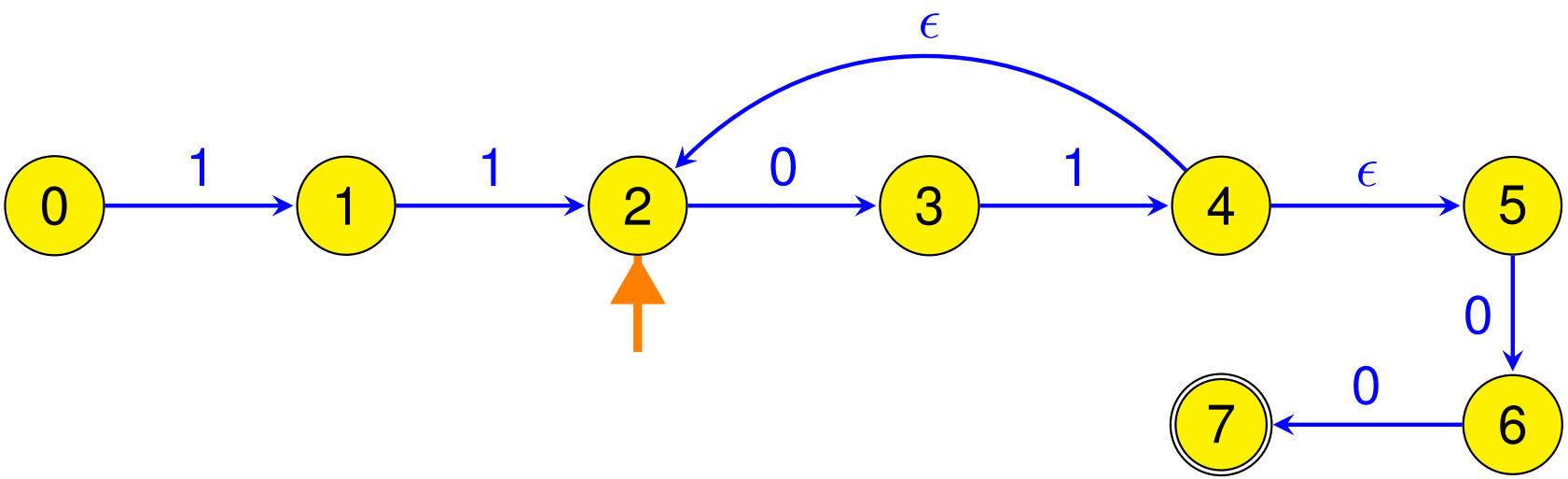
Transitions ϵ - Exemple



Transitions ϵ - Exemple

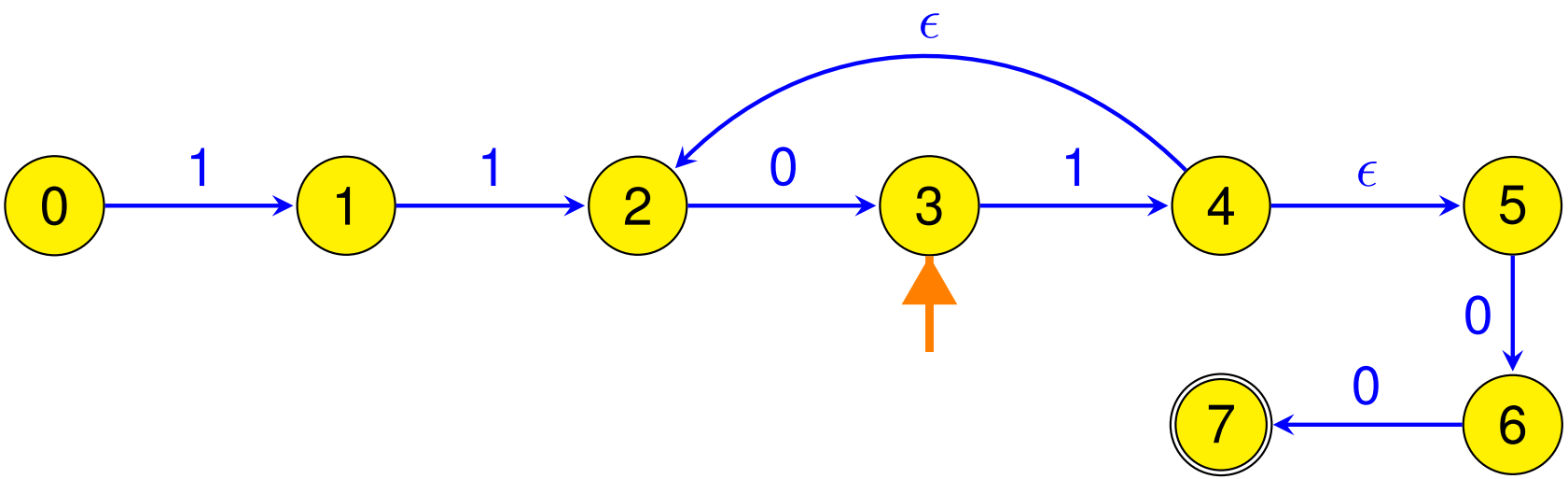


Transitions ϵ - Exemple



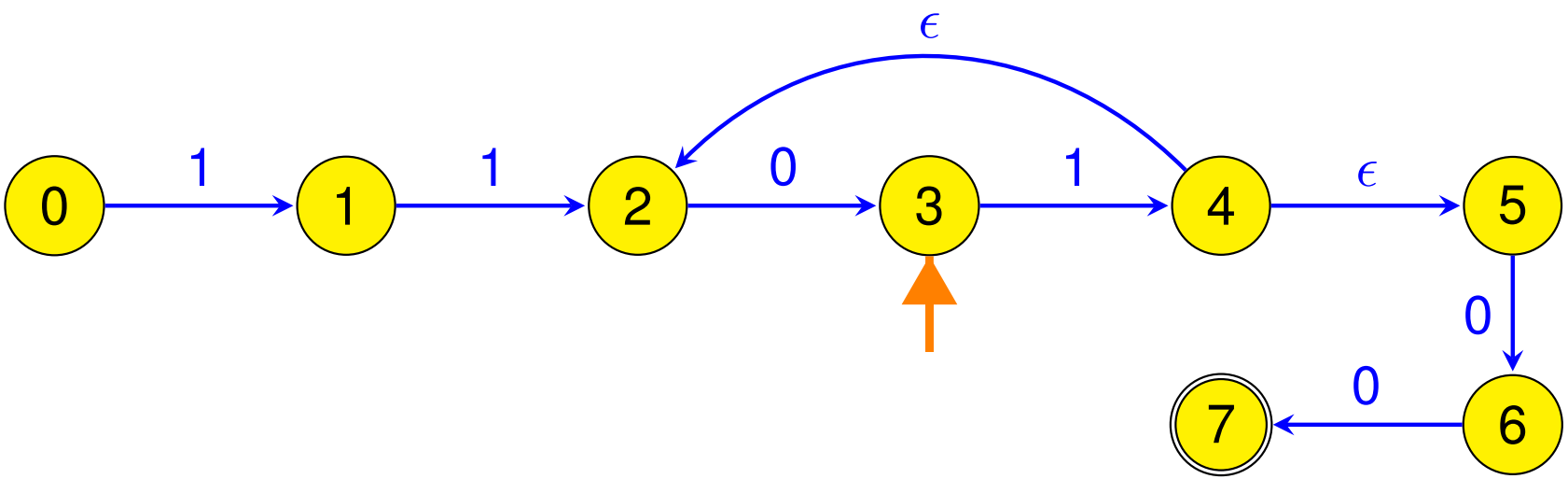
1 1 0 1 0 1 0 0

Transitions ϵ - Exemple



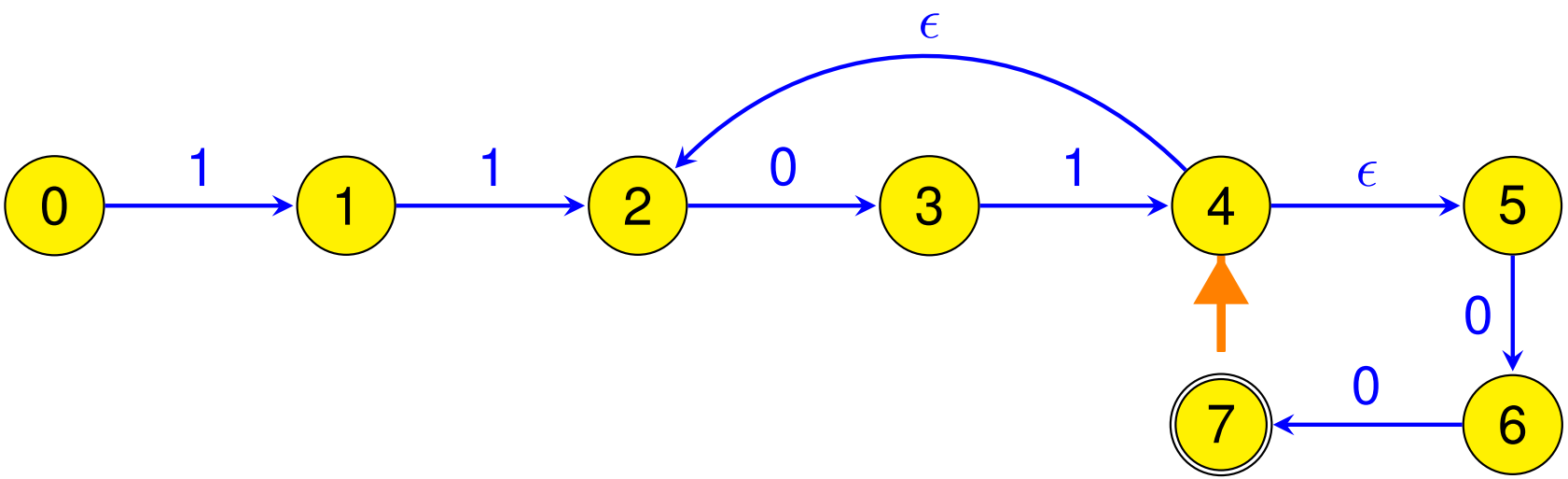
1 1 0 1 0 1 0 0

Transitions ϵ - Exemple



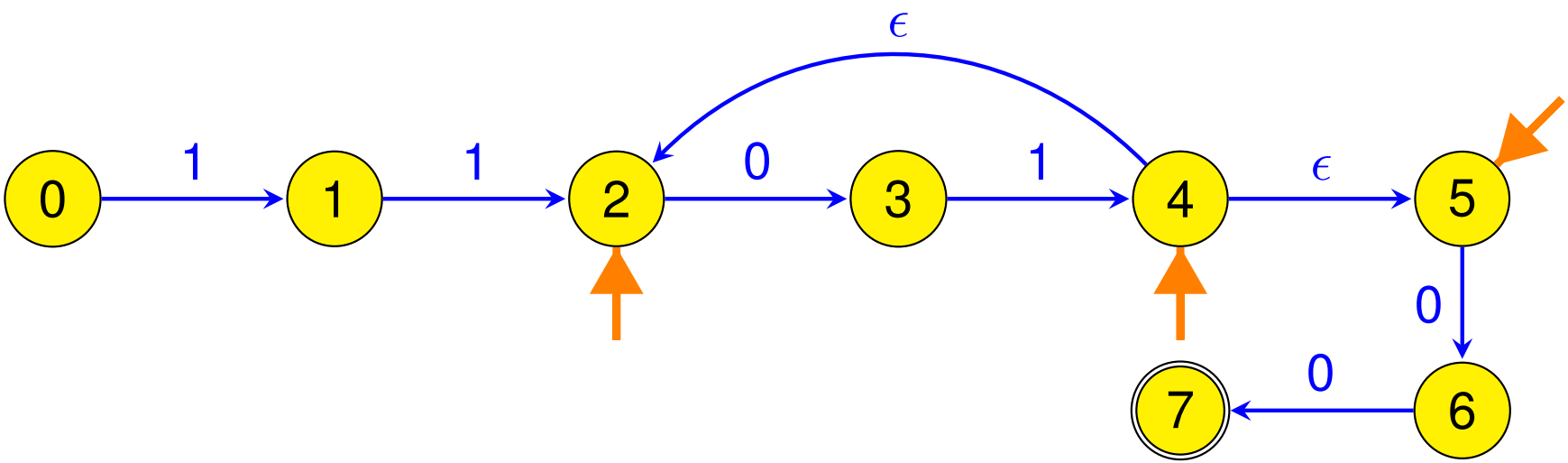
1 1 0 1 0 1 0 0

Transitions ϵ - Exemple



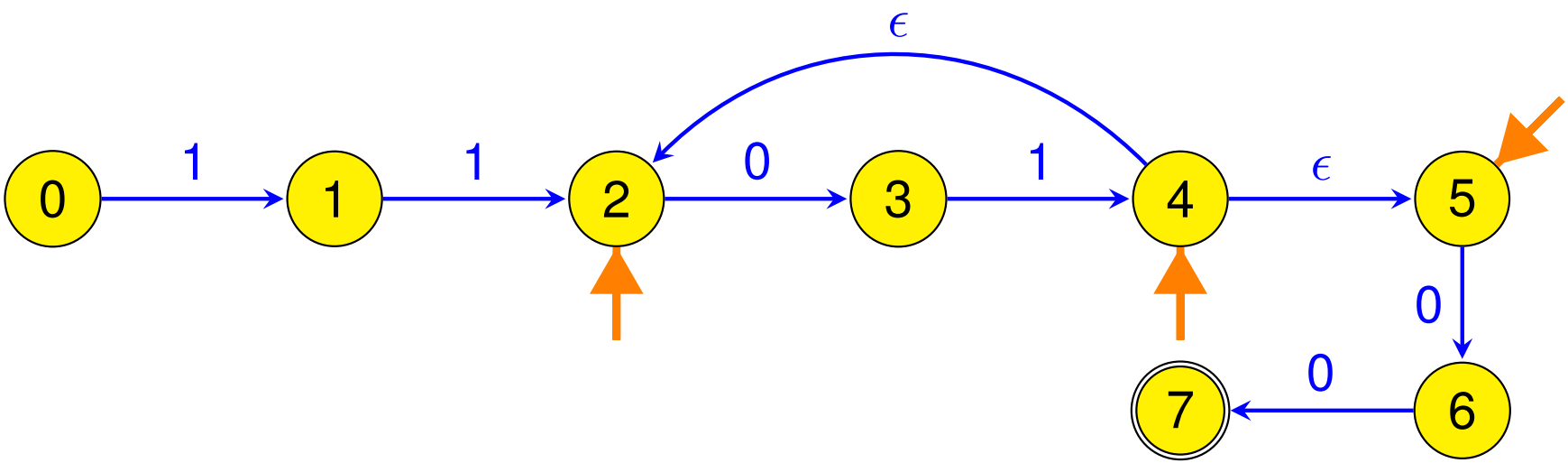
1 1 0 1 0 1 0 0

Transitions ϵ - Exemple



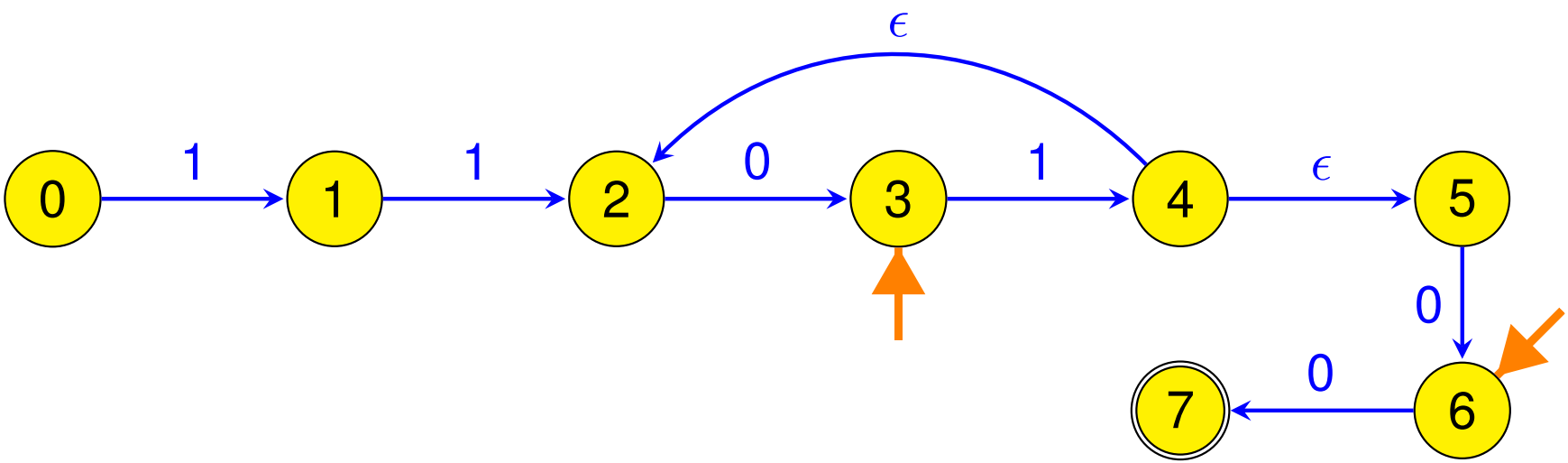
1 1 0 1 0 1 0 0

Transitions ϵ - Exemple



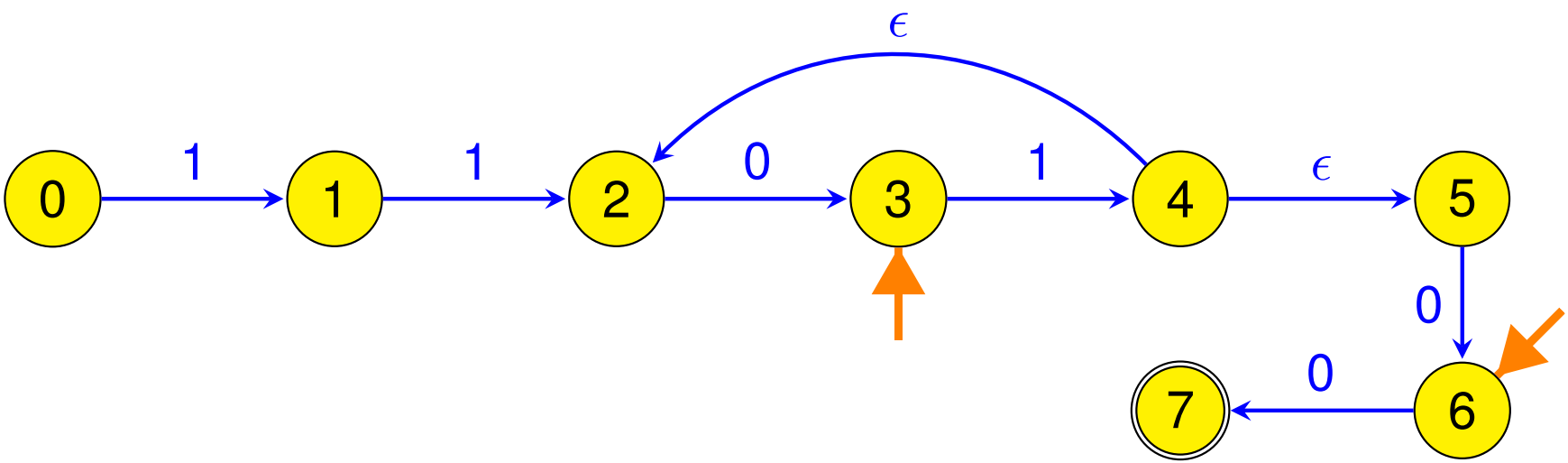
1 1 0 1 0 1 0 0

Transitions ϵ - Exemple



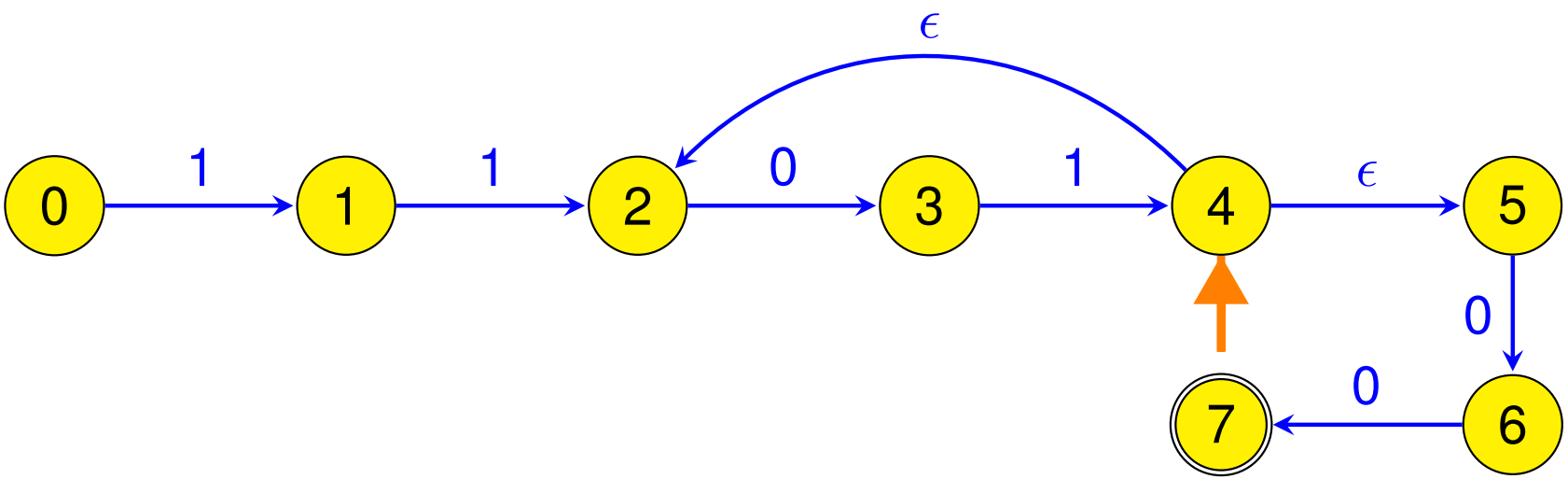
1 1 0 1 0 1 0 0

Transitions ϵ - Exemple



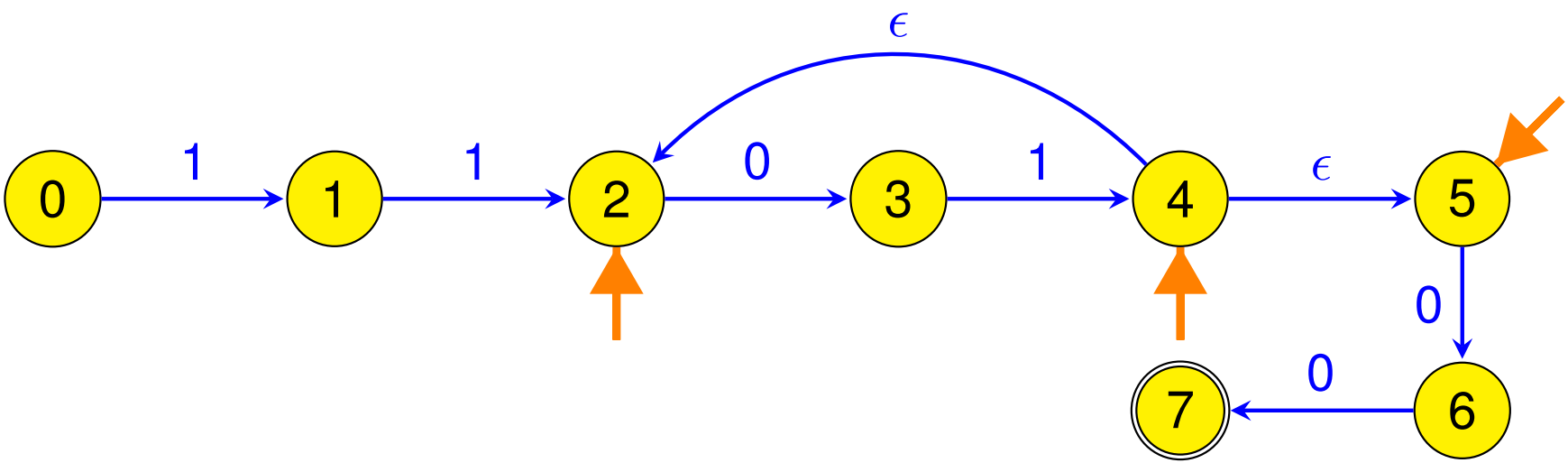
1 1 0 1 0 1 0 0

Transitions ϵ - Exemple



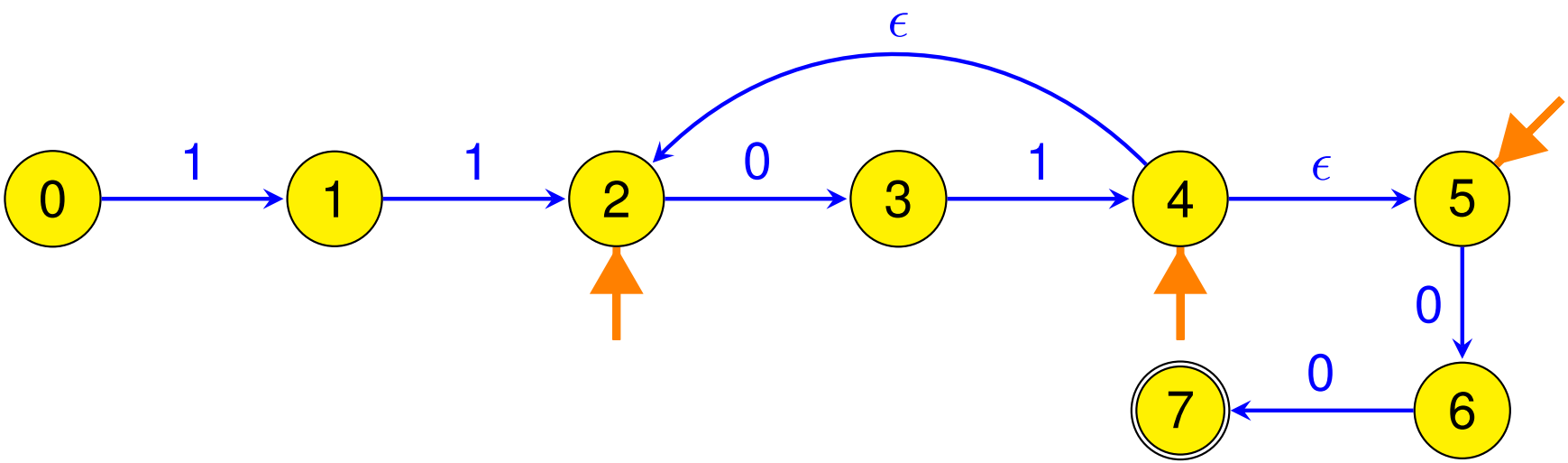
1 1 0 1 0 1 0 0

Transitions ϵ - Exemple



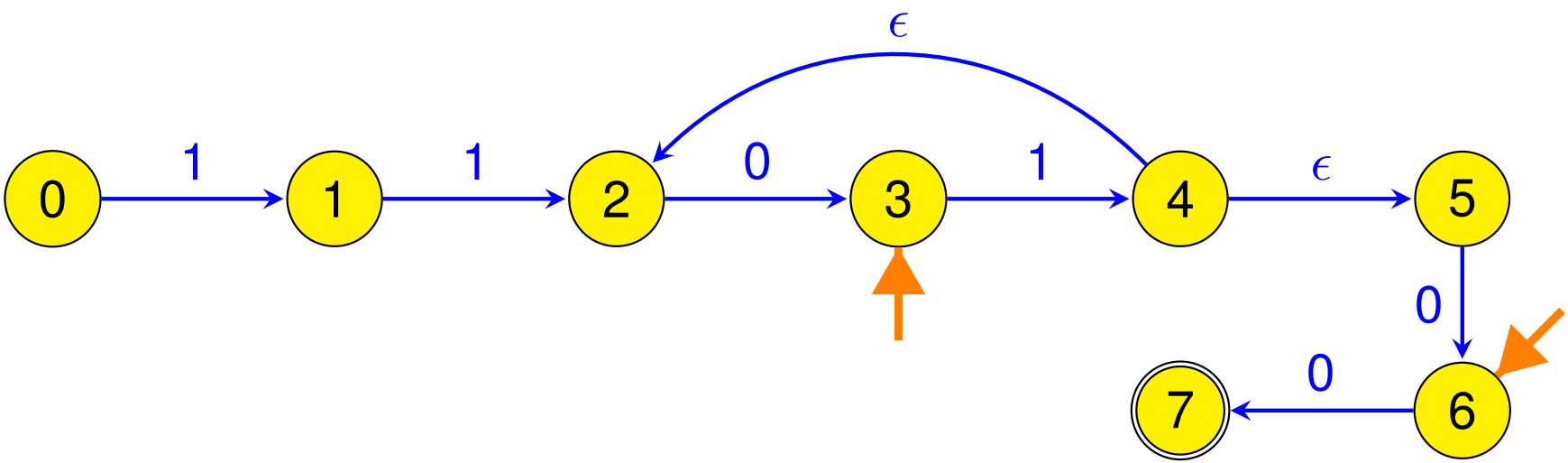
1 1 0 1 0 1 0 0

Transitions ϵ - Exemple



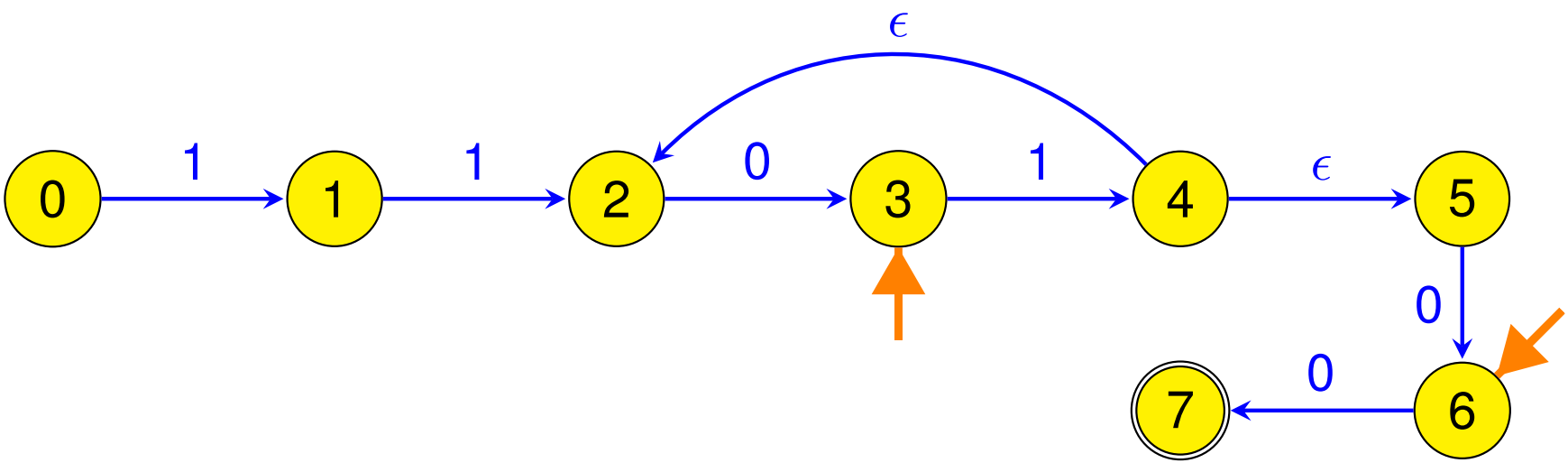
1 1 0 1 0 1 0 0

Transitions ϵ - Exemple



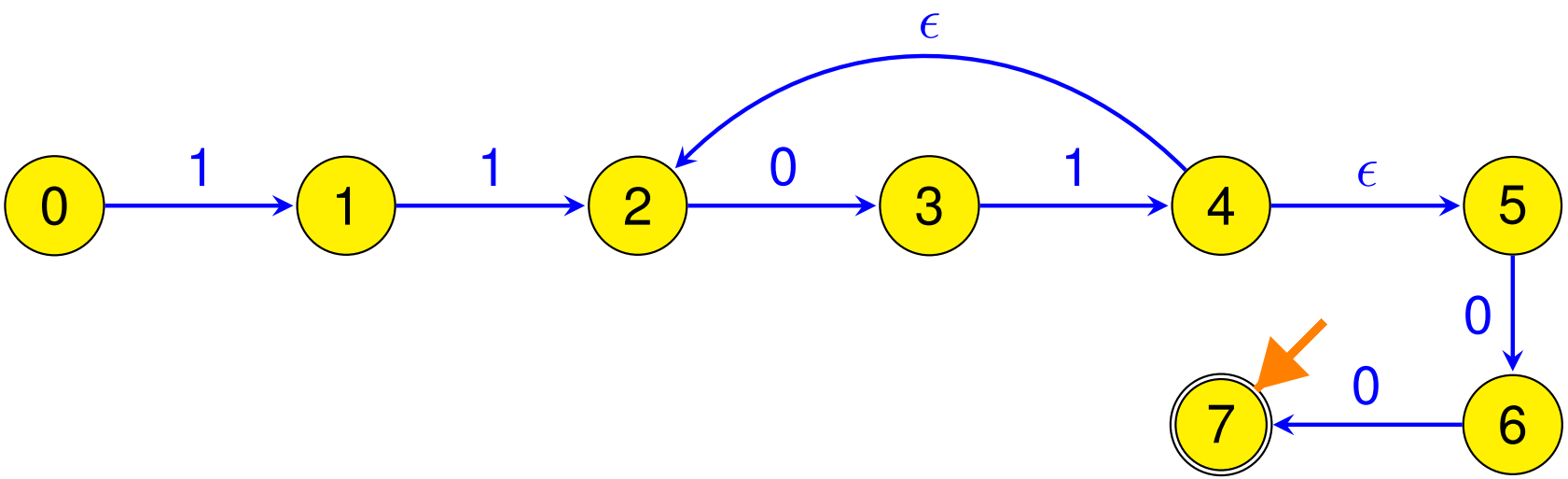
1 1 0 1 0 1 0 0

Transitions ϵ - Exemple



1 1 0 1 0 1 0 0

Transitions ϵ - Exemple



1 1 0 1 0 1 0 0

Transitions Epsilon

- Une transition Epsilon est “invisible” de l’extérieur : elle ne consomme aucune lettre.
- ☞ C’est une transition non-déterministe : l’automate est dans plusieurs états à la fois.

Plan

Expressions régulières

. . . ou encore expression **rationnelle**

Exemples :

- $a^* \, b \, a^*$
- $(ab)^*$
- $(aa)^*$
- $[0-9]^* \, [02468]$
- $([\hat{}3] \, 3 \, [\hat{}3] \, 3)^*$
- $(e \, u^* \, c)^*$

Expressions régulières

. . . ou encore expression **rationnelle**

Exemples :

- $a^* \, b \, a^*$
- $(ab)^*$
- $(aa)^*$
- $[0-9]^* \, [02468]$
- $([\hat{}3] \, 3 \, [\hat{}3] \, 3)^*$
- $(e \, u^* \, c)^*$

- Les mots qui ne finissent pas par c
- Les mots qui ne finissent pas par $k \, o$
- Les URLs

Expressions régulières

Définition : Les regexps – la base

Une expression régulière R est soit :

- Une lettre de l'*alphabet* Σ : x, y, ..
- Une séquence finie de regexps $R_1\ R_2\ ..\ R_n$
- L'opérateur de répétition R^*
- L'opérateur de choix $R_1 \mid R_2 \mid .. \mid R_n$
- Le groupage (R)
- Le choix parmi des lettres $[abcdef]$
- Le choix parmi un intervalle de lettres $[a - f]$
- Le choix hors d'un ensemble de lettres $[\hat{}\, aeiouy]$

Sucre syntaxique

Par convention,

- R^+ désigne $R\,R^*$ (au moins une occurence).
- $R^?$ désigne $(R \mid \epsilon)$ (une occurence ou aucune).

Expressions régulières

Exemple avancé :

- Récupérer toutes les URLs des liens `<a href="url">` d'une page html.

Expressions régulières

Exemple avancé :

- Récupérer toutes les URLs des liens `<a href="url">` d'une page html.

Syntaxe emacs :
$.* < a[\ t\ n]^* href = "([\hat{}"]^*)" >$

- Les parenthèses (groupage) permettent de faire référence à ce qui a été filtré.
- Par exemple, remplacer tous les `<a href="url">` par `<img alt="(url)">`
- Dans emacs, on ferait **replace-regexp**
  $.* < a[\ t\ n]^* href = "([\hat{}"]^*)" >$ par `<img alt="\1">`

Expressions régulières

Dernières remarques :

- Rapidement, une regexp devient horrible.
- Les regexps, on les comprend quand on les écrit, mais pas quand on les lit.

Expressions régulières

Définition : Les regexps – la base

Une expression régulière R est soit :

- Une lettre de l'*alphabet* Σ : x, y, ..
- Une séquence finie de regexps $R_1\ R_2\ ..\ R_n$
- L'opérateur de répétition R^*
- L'opérateur de choix $R_1 \mid R_2 \mid .. \mid R_n$
- Le groupage (R)
- Le choix parmi des lettres $[abcdef]$
- Le choix parmi un intervalle de lettres $[a - f]$
- Le choix hors d'un ensemble de lettres $[\hat{}\, aeiouy]$

Grammaire des regexps

Voici la grammaire formelle des regexps (en forme BNF) :

Définition : **Regexp**

$$
\begin{array}{lcl}
R & ::= & x \mid RR \mid (R \mid R) \mid R^* \mid [E] \mid [\hat{}\, E] \mid (R) \\
x & = & \textit{atome de l'alphabet } \Sigma \\
E & = & \textit{ensemble d'atomes}
\end{array}
$$

Quizz

À votre avis,

- Une regexp peut se traduire en automate équivalent ?

Quizz

À votre avis,

- Une regexp peut se traduire en automate équivalent ? **oui**

Quizz

À votre avis,

- Une regexp peut se traduire en automate équivalent ? **oui**
- N'importe quelle regexp, vraiment ?

Quizz

À votre avis,

- Une regexp peut se traduire en automate équivalent ? **oui**
- N'importe quelle regexp, vraiment ? **OUI !**

Quizz

À votre avis,

- Une regexp peut se traduire en automate équivalent ? **oui**
- N'importe quelle regexp, vraiment ? **OUI !**

- Un automate peut se traduire en regexp équivalente ?

Quizz

À votre avis,

- Une regexp peut se traduire en automate équivalent ? **oui**
- N'importe quelle regexp, vraiment ? **OUI !**

- Un automate peut se traduire en regexp équivalente ? **oui**

Quizz

À votre avis,

- Une regexp peut se traduire en automate équivalent ? **oui**
- N'importe quelle regexp, vraiment ? **OUI !**

- Un automate peut se traduire en regexp équivalente ? **oui**
- N'importe quel automate, vraiment ?

Quizz

À votre avis,

- Une regexp peut se traduire en automate équivalent ? **oui**
- N'importe quelle regexp, vraiment ? **OUI !**

- Un automate peut se traduire en regexp équivalente ? **oui**
- N'importe quel automate, vraiment ? **OUI !**

Regexp ⇒ Automate

«Une regexp peut se traduire en automate»

Exercice

Quel est l'automate équivalent à
$[\mathbf{a}-\mathbf{z}][\mathbf{a}-\mathbf{z}]^* - [\mathbf{1}-\mathbf{9}][\mathbf{0}-\mathbf{9}][\mathbf{0}-\mathbf{9}][\mathbf{0}-\mathbf{9}]$?

Regexp ⇒ Automate

«Une regexp peut se traduire en automate»

Exercice

Quel est l'automate équivalent à
$[\mathbf{a}-\mathbf{z}][\mathbf{a}-\mathbf{z}]^* - [\mathbf{1}-\mathbf{9}][\mathbf{0}-\mathbf{9}][\mathbf{0}-\mathbf{9}][\mathbf{0}-\mathbf{9}]$?

Preuve de l'énoncé

Comment le prouver ?

Regexp ⇒ Automate

La preuve qu'une regexp peut se traduire en automate est **constructive**, i.e. on explique comment tranformer une regexp en automate.

Regexp ⇒ Automate

La preuve qu'une regexp peut se traduire en automate est **constructive**, i.e. on explique comment tranformer une regexp en automate.

La preuve considère tous les cas de regexp possibles, donnés par la définition : $R \quad ::= \quad x \mid R\,R \mid (R \mid R) \mid R^* \mid [E] \mid [\hat{} E] \mid (R)$

Regexp ⇒ Automate

La preuve qu'une regexp peut se traduire en automate est **constructive**, i.e. on explique comment tranformer une regexp en automate.

La preuve considère tous les cas de regexp possibles, donnés par la définition : $R \quad ::= \quad x \mid R\,R \mid (R \mid R) \mid R^* \mid [E] \mid [\hat{}\,E] \mid (R)$

La définition étant récursive, la preuve sera également récursive.

On dit que la preuve est par **induction structurelle sur** R.

Regexp ⇒ Automate

Regexp ⇒ Automate.

- La preuve est par induction structurelle sur R.
- Pour chaque cas, on fournit un automate reconnaissant la regexp donnée.
- Pour faciliter la preuve, on impose des contraintes supplémentaires : l'automate doit avoir un seul état de départ et un seul état d'arrivée ; le départ et l'arrivée doivent être "en sens unique".

Cas x :

Cas $[E]$:

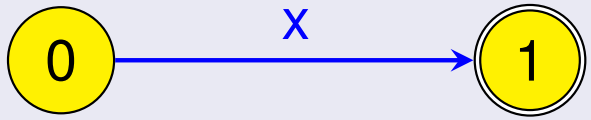
Cas $[\hat{}\, E]$:

Regexp ⇒ Automate

Regexp ⇒ Automate.

- La preuve est par induction structurelle sur R.
- Pour chaque cas, on fournit un automate reconnaissant la regexp donnée.
- Pour faciliter la preuve, on impose des contraintes supplémentaires : l'automate doit avoir un seul état de départ et un seul état d'arrivée ; le départ et l'arrivée doivent être "en sens unique".

Cas x :



Cas $[E]$:

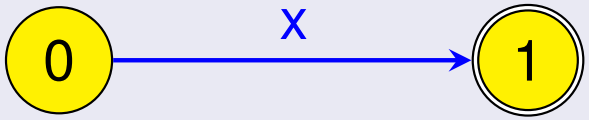
Cas $[\hat{}\, E]$:

Regexp ⇒ Automate

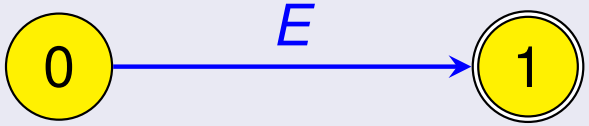
Regexp ⇒ Automate.

- La preuve est par induction structurelle sur R.
- Pour chaque cas, on fournit un automate reconnaissant la regexp donnée.
- Pour faciliter la preuve, on impose des contraintes supplémentaires : l'automate doit avoir un seul état de départ et un seul état d'arrivée ; le départ et l'arrivée doivent être "en sens unique".

Cas x :



Cas $[E]$:



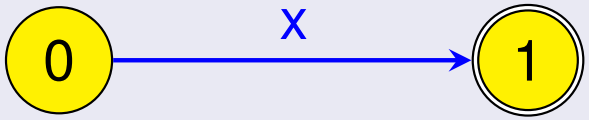
Cas $[\hat{}\, E]$:

Regexp ⇒ Automate

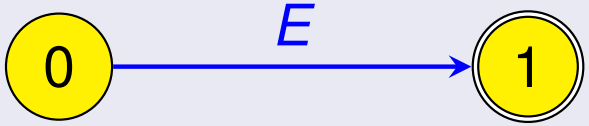
Regexp ⇒ Automate.

- La preuve est par induction structurelle sur R.
- Pour chaque cas, on fournit un automate reconnaissant la regexp donnée.
- Pour faciliter la preuve, on impose des contraintes supplémentaires : l'automate doit avoir un seul état de départ et un seul état d'arrivée ; le départ et l'arrivée doivent être "en sens unique".

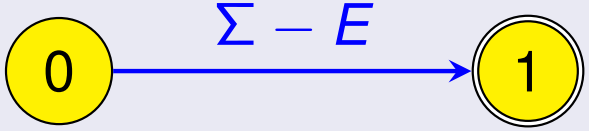
Cas x :



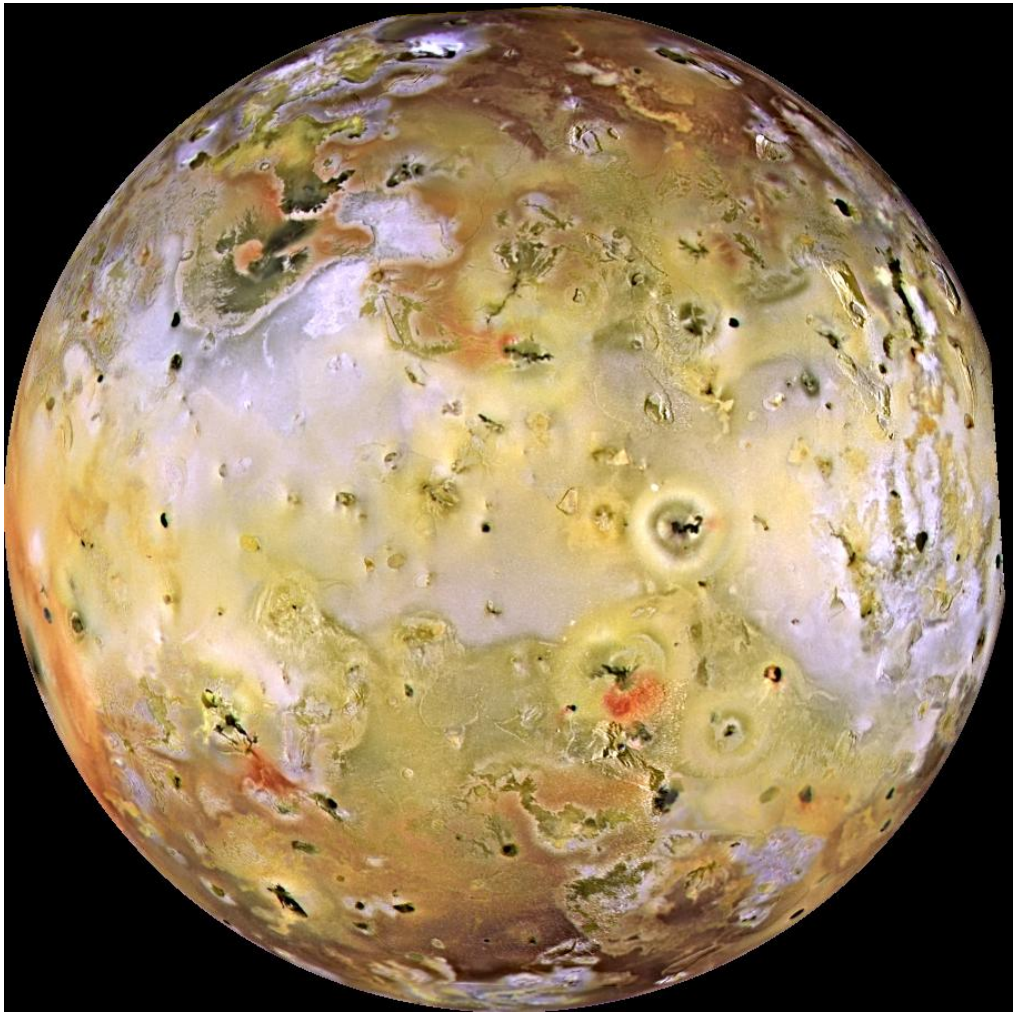
Cas $[E]$:



Cas $[\hat{}\,E]$:



Pause



Regexp ⇒ Automate

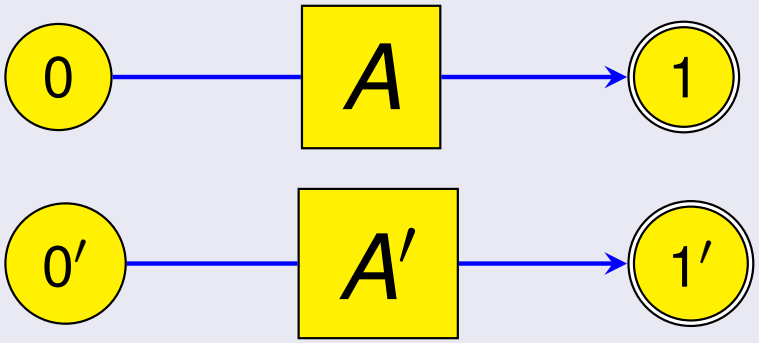
Regexp ⇒ Automate.

Cas $R\,R'$:

Regexp ⇒ Automate

Regexp ⇒ Automate.

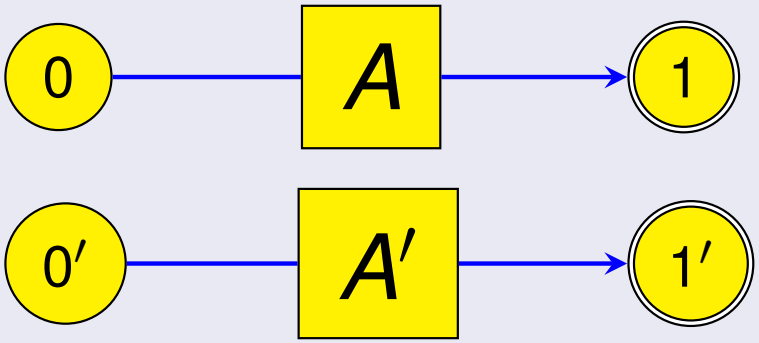
Cas $R\,R'$: Par hypothèse de récurrence (induction structurelle), il existe deux automates reconnaissant R et R' :



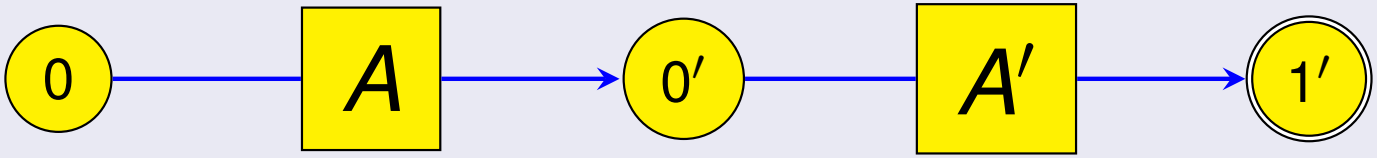
Regexp ⇒ Automate

Regexp ⇒ Automate.

Cas $R\,R'$: Par hypothèse de récurrence (induction structurelle), il existe deux automates reconnaissant R et R' :



L'automate suivant reconnait $R\,R'$:



Regexp ⇒ Automate

Regexp ⇒ Automate.

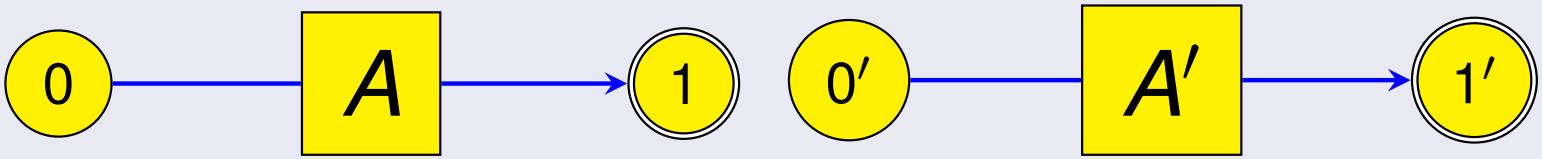
Cas $R \mid R'$:

Regexp ⇒ Automate

Regexp ⇒ Automate.

Cas $R \mid R'$:

Par hypothèse de récurrence, il existe deux automates reconnaissant R et R' :

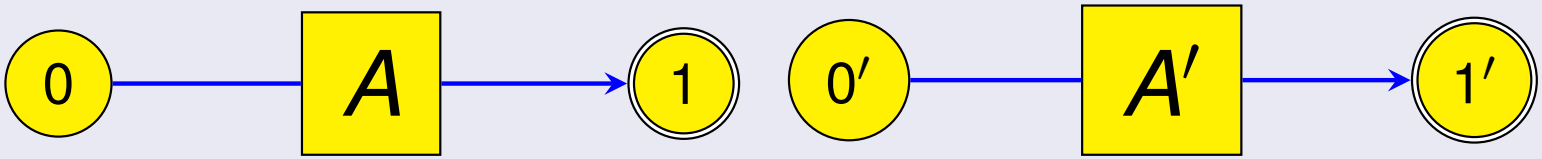


Regexp ⇒ Automate

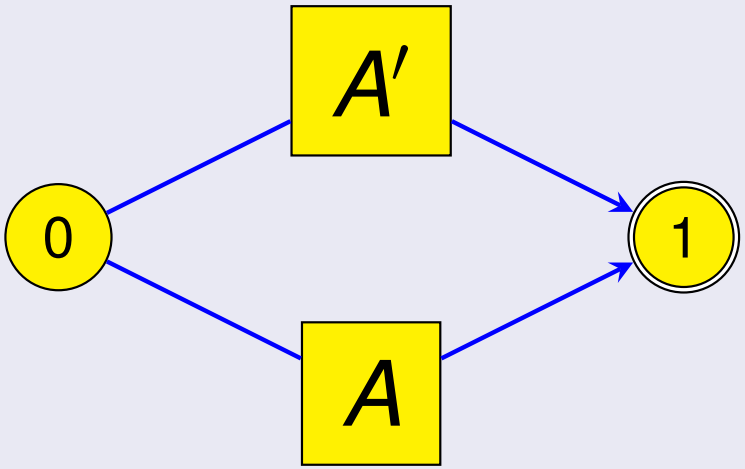
Regexp ⇒ Automate.

Cas $R \mid R'$:

Par hypothèse de récurrence, il existe deux automates reconnaissant R et R' :



L'automate suivant reconnait $R \mid R'$:



Regexp $\Rightarrow$ Automate

Regexp $\Rightarrow$ Automate.

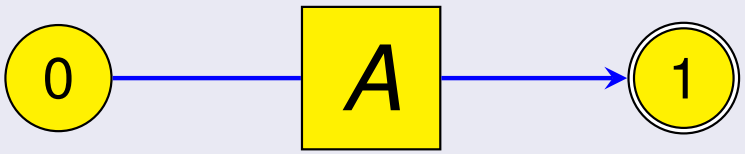
Cas R^* :

Regexp ⇒ Automate

Regexp ⇒ Automate.

Cas R^* :

Par hypothèse de récurrence, il existe un automate reconnaissant R :

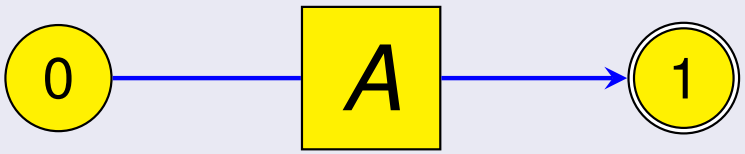


Regexp ⇒ Automate

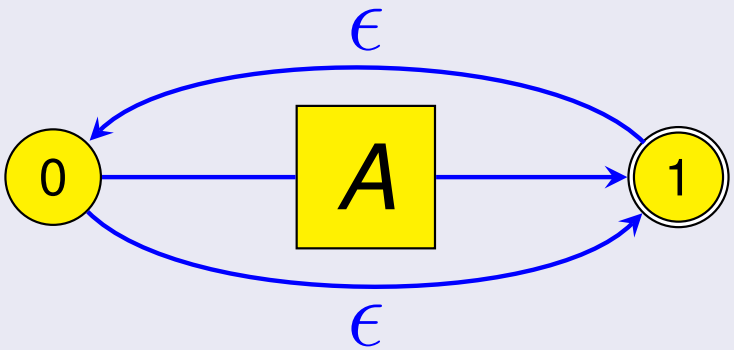
Regexp ⇒ Automate.

Cas R^* :

Par hypothèse de récurrence, il existe un automate reconnaissant R :



L'automate suivant reconnait R^* :

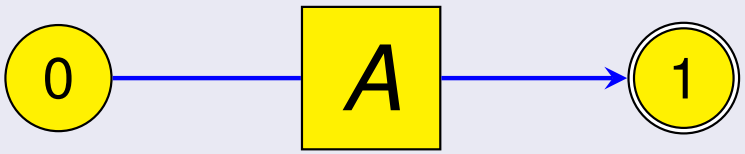


Regexp ⇒ Automate

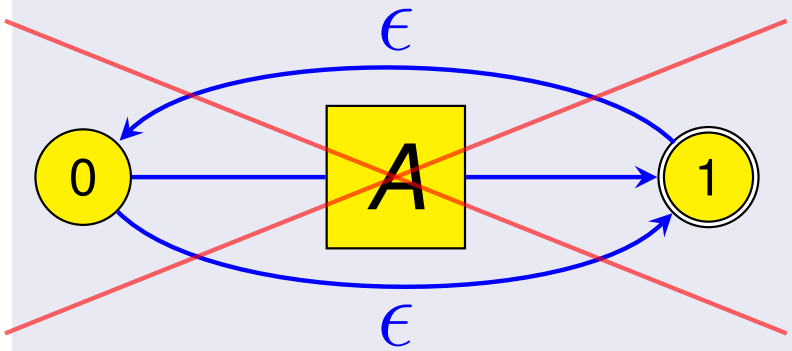
Regexp ⇒ Automate.

Cas R^* :

Par hypothèse de récurrence, il existe un automate reconnaissant R :



L'automate suivant reconnait R^* :

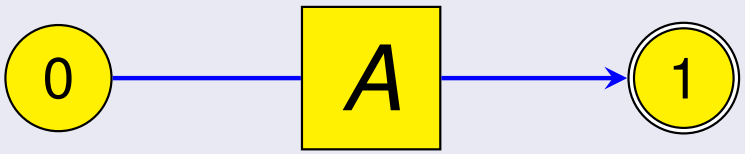


Regexp ⇒ Automate

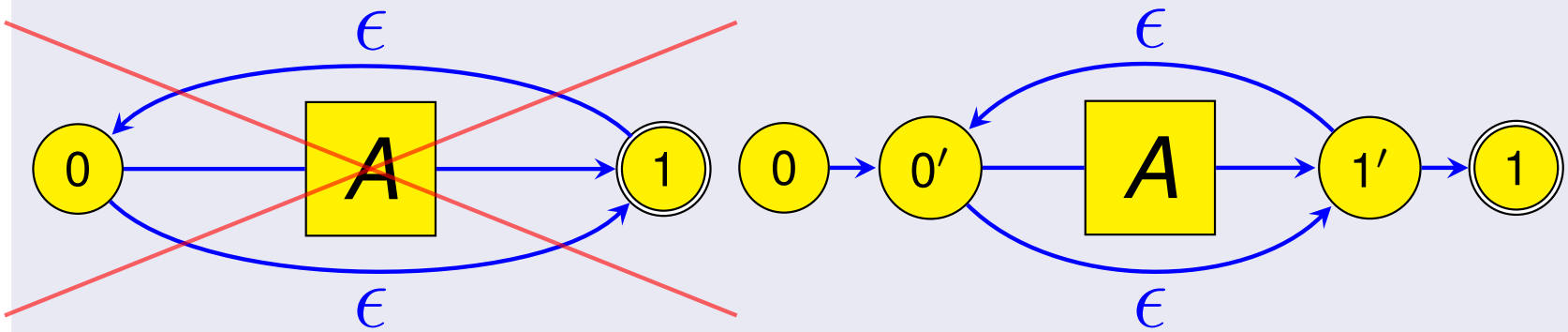
Regexp ⇒ Automate.

Cas R^* :

Par hypothèse de récurrence, il existe un automate reconnaissant R :



L'automate suivant reconnait R^* :



Regexp $\Rightarrow$ Automate

Regexp $\Rightarrow$ Automate.

Cas (R) :

Regexp ⇒ Automate

Regexp ⇒ Automate.

Cas (R) :
C'est le même automate que pour R (hypothèse de récurrence, trivial).

□

Regexp ⇒ Automate

En résumé, pour toute expression régulière, je peux construire un automate équivalent.

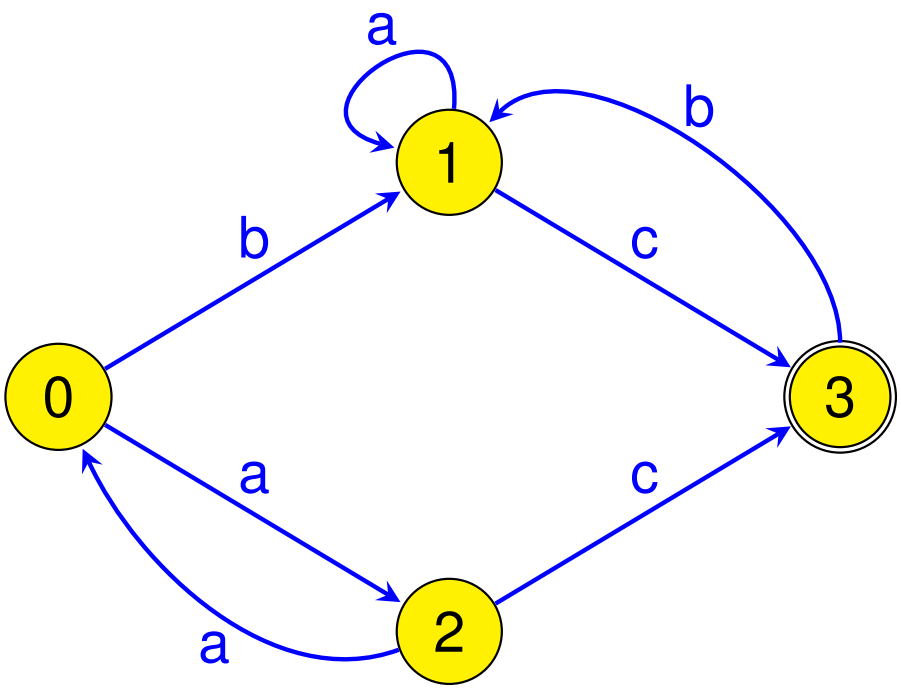
Peut-on trouver un automate déterministe équivalent ? i.e. sans transition epsilon ?

Automate ⇒ Regexp

La preuve dans l'autre sens (Automate ⇒ Regexp) est un peu plus pénible.

- ✎ On se contentera d'exemples sur des cas particuliers.

Automate ⇒ Regexp



Automate ⇒ Regexp

En résumé, à tout automate, je peux associer une regexp qu'il reconnaît.

À retenir

- Notion d'expression régulière
- Définition formelle (grammaire des regexps)
- Équivalence avec les automates (deux sens)