Introduction à l'Informatique Théorique

Automates finis et langages réguliers

Paolo Tranquilli

28 Septembre 2010

Alphabets et mots

- On va nommer alphabet un ensemble fini et non vide
- L'ensemble des mots est noté

$$\Sigma^* := \{\varepsilon\} \cup \{\, x_1 x_2 \cdots x_k \mid k \geq 1, x_i \in \Sigma \,\}$$

- ε est un symbole utilisé pour le mot vide

Exemples

- $13, 42, 007 \in \{0, 1, \ldots, 9\}^*$
- $\texttt{ciao}, \texttt{mot} \in \{\texttt{a}, \ldots, \texttt{z}\}^*$
- $110, 0101, 1 \in \{0, 1\}^*$
- $\texttt{mot␣?␣oui␣!} \in \{\texttt{a}, \ldots, \texttt{z}, \texttt{␣}, \texttt{?}, \texttt{!}\}^*$
- $babbba, abab, abba \in \{a, b\}^*$
- $\bullet\bullet\bullet\bullet\bullet, \bullet\bullet \in \{\bullet\}^*$

- Notations :
  - $|w|$ est la longueur de w
  - $w_i \in \Sigma$ pour $1 \leq i \leq |w|$ c'est le i-ème caractère de w

Σ^* dénombrable ?

Σ^* dénombrable ? Oui

Pourquoi ?

- Réponse indirecte :

$$\Sigma^* = \bigcup_{k=0}^{\infty} \Sigma^k, \quad \text{où } \Sigma^0 = \{\varepsilon\}$$

  union dénombrable de dénombrables $\Longrightarrow$ dénombrable (vous verrez en TD)

- Réponse directe : Σ fini $\Rightarrow$ il y a $\sigma : \Sigma \leftrightarrow \{1, \ldots, n\}$ où $n = \#\Sigma$. On étend σ :

$$\begin{aligned} \sigma : \Sigma &\longleftrightarrow \mathbb{N} \\ x_1 \cdots x_k &\longmapsto \textstyle\sum_{i=1}^{k} \sigma(x_i) n^{i-1} \end{aligned}$$

Exercice

Prouver que σ est bijective.

Opérations sur Σ^*

Injection de Σ : chaque symbole $a \in \Sigma$ peut être considéré comme $a \in \Sigma^*$ ($|a| = 1$)

Enchaînement : $\cdot : \Sigma^* \times \Sigma^* \to \Sigma^*$

$$(x_1 \cdots x_k) \cdot (y_1 \cdots y_h) = x_1 \cdots x_k y_1 \cdots y_k$$

- opération associative : $(uv)w = u(vw) = uvw$
- mot vide est neutre : $\varepsilon u = u\varepsilon = u$
- notation puissance : $u^n = \underbrace{u \cdots u}_{n \text{ fois}}, \quad u^0 = \varepsilon$

Fonctions et langages

Fonctions (de mots) $f : \Sigma^* \to \Delta^*$

Exemples

- changement de base : $\{0,\ldots,9\}^* \longrightarrow \{0,1\}^*$
  $x_{h-1}\cdots x_0 \longmapsto (\sum_i x_i 10^i)_2$

  p.ex. $1 \mapsto 1$, $6 \mapsto 110$, $21 \mapsto 10101$
- élimination des répétitions : $\Sigma^* \to \Sigma^*$

  p.ex. $abba \mapsto aba$, $baaaabb \mapsto bab$

Langages $L \subseteq \Sigma^*$

Exemples

- $\emptyset$, ou $\{\varepsilon\}$ (différents !)
- $\{0\} \cup \{\, 1w \mid w \in \{0,1\}^* \,\}$ (0 et mots binaires commençant par 1)
- $\{\, w \in \Sigma^* \mid w \text{ sans répétitions} \,\}$
- $\{\, w \in \{0,\ldots,9\}^* \mid w \text{ sont les premières } |w| \text{ chiffres de } \pi \,\}$

Langages comme fonctions et codage de $\mathbb{N}$

- Chaque langage peut être vu comme une fonction de mots

$$\{\text{ langages }\} = \mathcal{P}(\Sigma^*) \leftrightarrow \{0,1\}^{\Sigma^*} \subseteq (\{0,1\}^*)^{\Sigma^*}$$

- Principalement on utilisera deux codages de $\mathbb{N}$ comme mots

unaire $\mathbb{N} \leftrightarrow \{\bullet\}^*$, $n \mapsto \bullet^n$

binaire $\mathbb{N} \to \{0,1\}^*$, $n \mapsto b_0 b_1 \cdots b_{h-1}$ avec $n = \sum_i b_i 2^i$

Est tout calculable ?

On peut déjà répondre !

Est tout calculable ? no !

On peut déjà répondre !

- Comme on avez dit, une procédure/machine est un objet fini
- qu'est-ce que ce soit le modèle de calcul,

$$\#\{\text{procédures}\} = \#\,N$$

- mais

$$\#\{\text{fonctions de mots}\} = \#\{\text{langages}\} = \#\{0,1\}^{\mathbb{N}} > \#\,\mathbb{N}$$

- forcement il existe quelque chose de non-calculable

Reconnaître un langage : une première machine

- Le but : une machine ($=$ procédure automatique) qui sache distinguer les mots en $\{0,1\}^*$ avec un nombre pair de 1

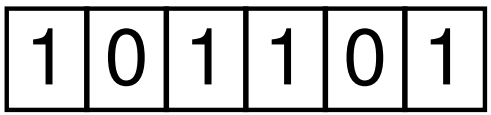
Reconnaître un langage : une première machine

- Le but : une machine (= procédure automatique) qui sache distinguer les mots en $\{0,1\}^*$ avec un nombre pair de 1
- Première idée : on compte le nombre de 1

1 0 1 1 0 1

Reconnaître un langage : une première machine

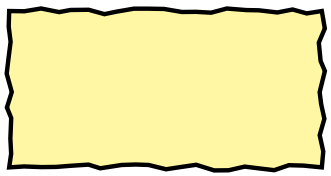
- Le but : une machine (= procédure automatique) qui sache distinguer les mots en $\{0,1\}^*$ avec un nombre pair de 1
- Première idée : on compte le nombre de 1



Reconnaître un langage : une première machine

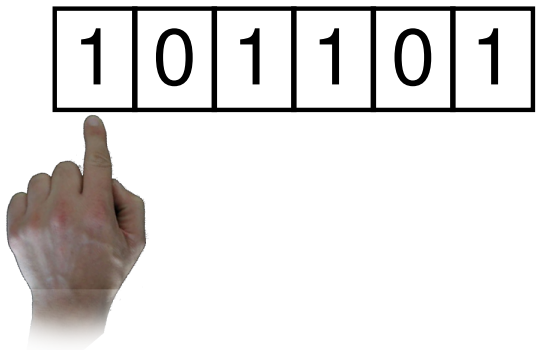
- Le but : une machine (= procédure automatique) qui sache distinguer les mots en $\{0,1\}^*$ avec un nombre pair de 1
- Première idée : on compte le nombre de 1

1	0	1	1	0	1



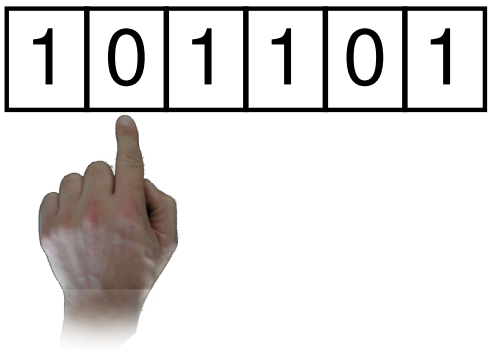
Reconnaître un langage : une première machine

- Le but : une machine (= procédure automatique) qui sache distinguer les mots en $\{0,1\}^*$ avec un nombre pair de 1
- Première idée : on compte le nombre de 1



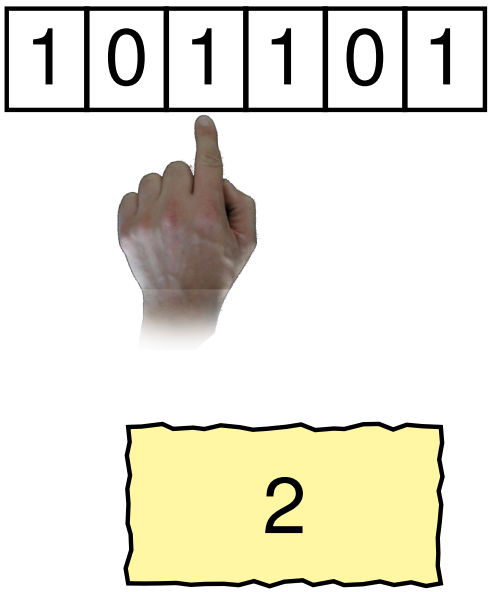
Reconnaître un langage : une première machine

- Le but : une machine (= procédure automatique) qui sache distinguer les mots en $\{0,1\}^*$ avec un nombre pair de 1
- Première idée : on compte le nombre de 1



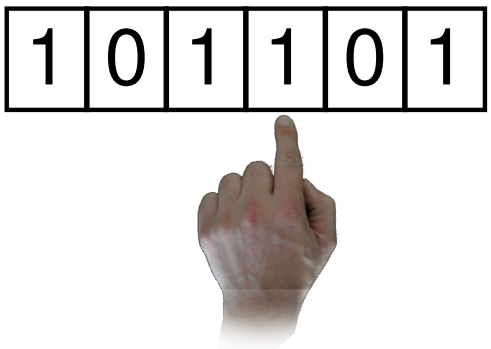
Reconnaître un langage : une première machine

- Le but : une machine (= procédure automatique) qui sache distinguer les mots en $\{0,1\}^*$ avec un nombre pair de 1
- Première idée : on compte le nombre de 1



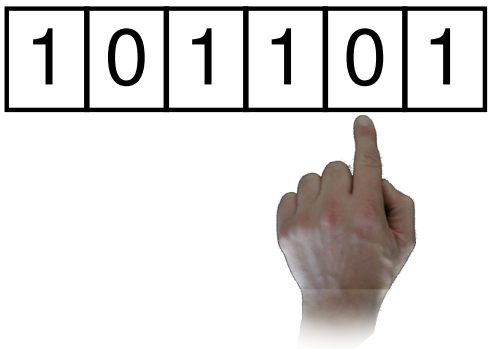
Reconnaître un langage : une première machine

- Le but : une machine ($=$ procédure automatique) qui sache distinguer les mots en $\{0,1\}^*$ avec un nombre pair de 1
- Première idée : on compte le nombre de 1



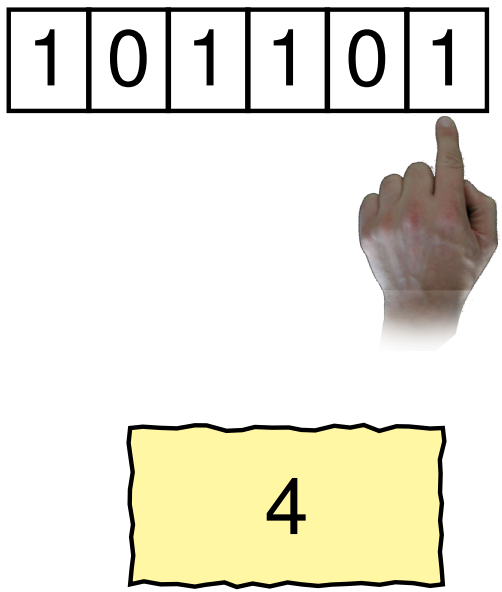
Reconnaître un langage : une première machine

- Le but : une machine (= procédure automatique) qui sache distinguer les mots en $\{0,1\}^*$ avec un nombre pair de 1
- Première idée : on compte le nombre de 1



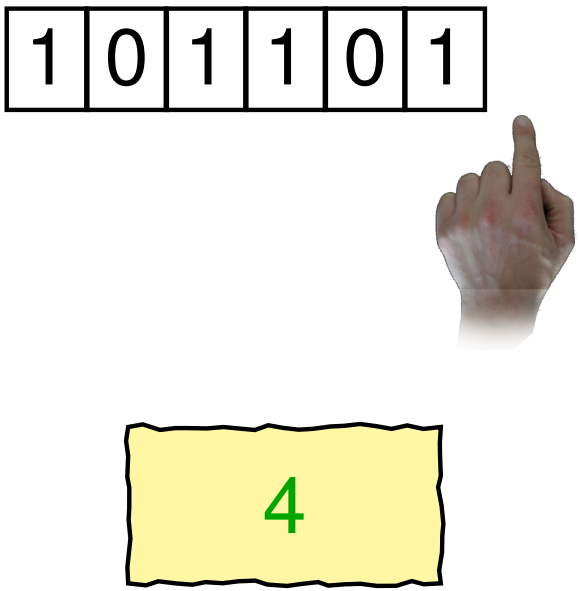
Reconnaître un langage : une première machine

- Le but : une machine (= procédure automatique) qui sache distinguer les mots en $\{0,1\}^*$ avec un nombre pair de 1
- Première idée : on compte le nombre de 1



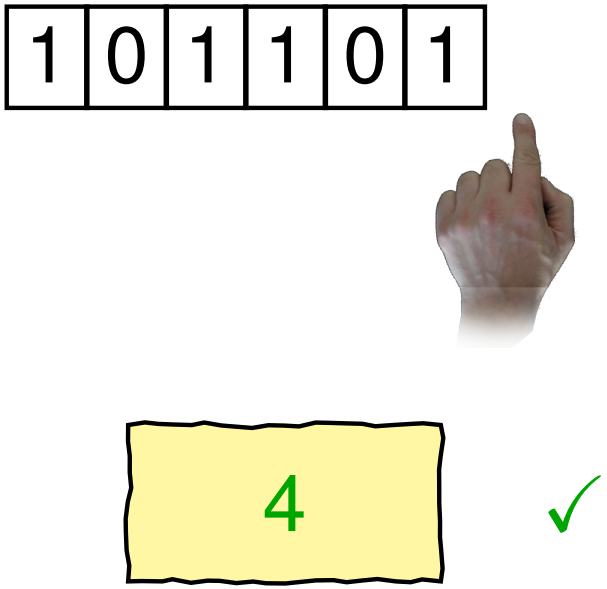
Reconnaître un langage : une première machine

- Le but : une machine (= procédure automatique) qui sache distinguer les mots en $\{0,1\}^*$ avec un nombre pair de 1
- Première idée : on compte le nombre de 1



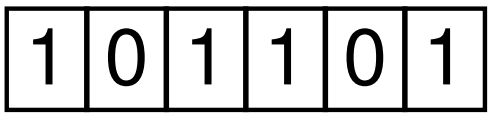
Reconnaître un langage : une première machine

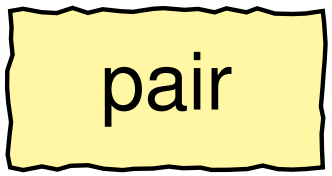
- Le but : une machine (= procédure automatique) qui sache distinguer les mots en $\{0,1\}^*$ avec un nombre pair de 1
- Première idée : on compte le nombre de 1



Reconnaître un langage : une première machine

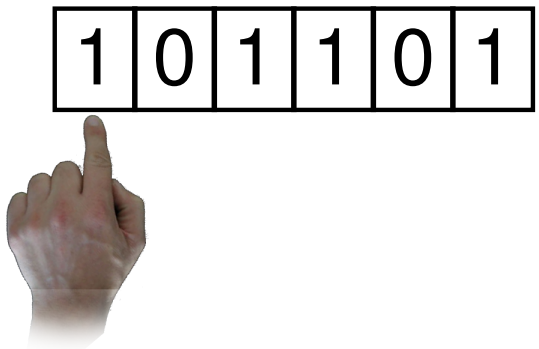
- Le but : une machine (= procédure automatique) qui sache distinguer les mots en $\{0,1\}^*$ avec un nombre pair de 1
- Première idée : on compte le nombre de 1
- Deuxième idée : il suffit de sauvegarder que la parité des 1





Reconnaître un langage : une première machine

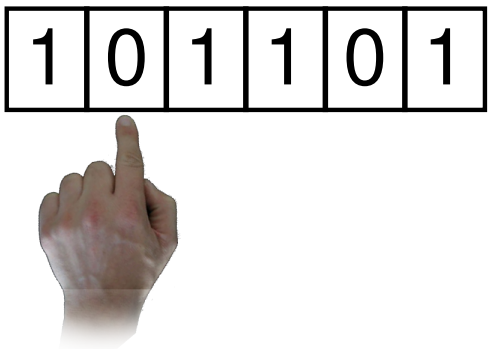
- Le but : une machine (= procédure automatique) qui sache distinguer les mots en $\{0,1\}^*$ avec un nombre pair de 1
- Première idée : on compte le nombre de 1
- Deuxième idée : il suffit de sauvegarder que la parité des 1



impair

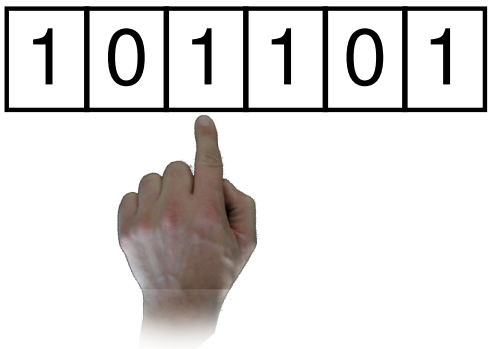
Reconnaître un langage : une première machine

- Le but : une machine (= procédure automatique) qui sache distinguer les mots en $\{0,1\}^*$ avec un nombre pair de 1
- Première idée : on compte le nombre de 1
- Deuxième idée : il suffit de sauvegarder que la parité des 1



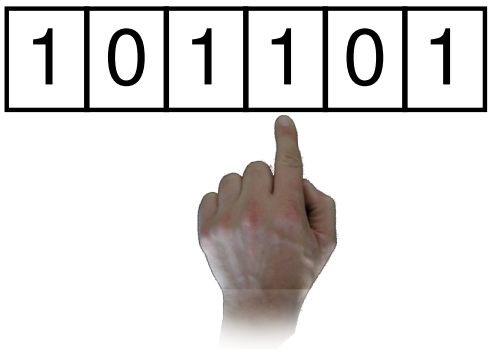
Reconnaître un langage : une première machine

- Le but : une machine (= procédure automatique) qui sache distinguer les mots en $\{0,1\}^*$ avec un nombre pair de 1
- Première idée : on compte le nombre de 1
- Deuxième idée : il suffit de sauvegarder que la parité des 1



Reconnaître un langage : une première machine

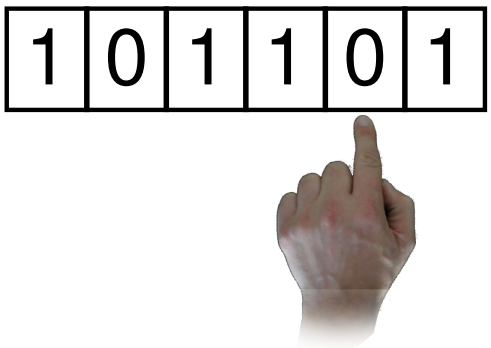
- Le but : une machine ($=$ procédure automatique) qui sache distinguer les mots en $\{0,1\}^*$ avec un nombre pair de 1
- Première idée : on compte le nombre de 1
- Deuxième idée : il suffit de sauvegarder que la parité des 1



impair

Reconnaître un langage : une première machine

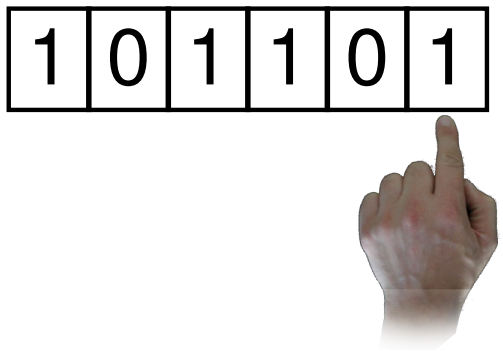
- Le but : une machine ($=$ procédure automatique) qui sache distinguer les mots en $\{0,1\}^*$ avec un nombre pair de 1
- Première idée : on compte le nombre de 1
- Deuxième idée : il suffit de sauvegarder que la parité des 1

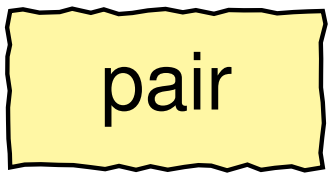


impair

Reconnaître un langage : une première machine

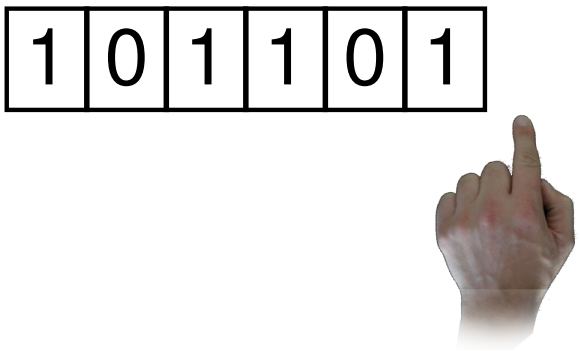
- Le but : une machine ($=$ procédure automatique) qui sache distinguer les mots en $\{0,1\}^*$ avec un nombre pair de 1
- Première idée : on compte le nombre de 1
- Deuxième idée : il suffit de sauvegarder que la parité des 1





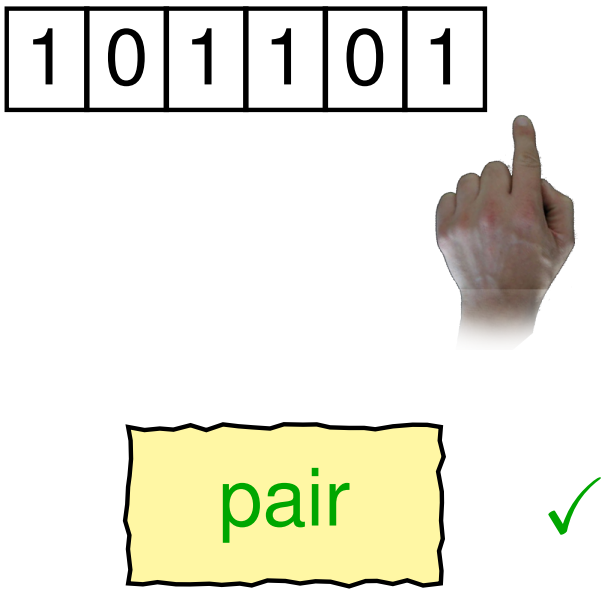
Reconnaître un langage : une première machine

- Le but : une machine ($=$ procédure automatique) qui sache distinguer les mots en $\{0,1\}^*$ avec un nombre pair de 1
- Première idée : on compte le nombre de 1
- Deuxième idée : il suffit de sauvegarder que la parité des 1



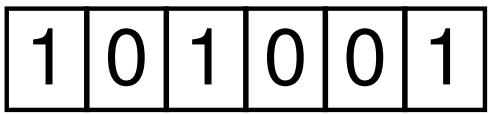
Reconnaître un langage : une première machine

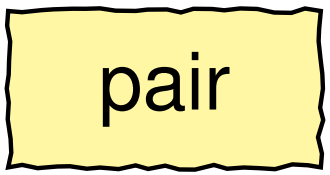
- Le but : une machine (= procédure automatique) qui sache distinguer les mots en $\{0,1\}^*$ avec un nombre pair de 1
- Première idée : on compte le nombre de 1
- Deuxième idée : il suffit de sauvegarder que la parité des 1



Reconnaître un langage : une première machine

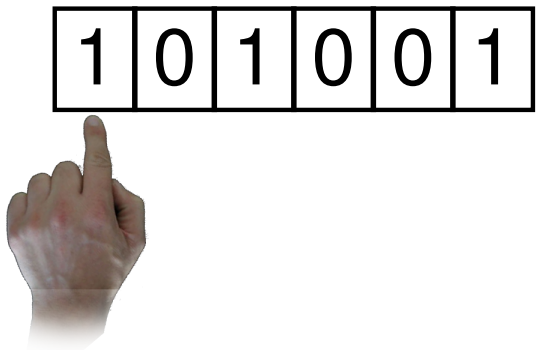
- Le but : une machine (= procédure automatique) qui sache distinguer les mots en $\{0,1\}^*$ avec un nombre pair de 1
- Première idée : on compte le nombre de 1
- Deuxième idée : il suffit de sauvegarder que la parité des 1





Reconnaître un langage : une première machine

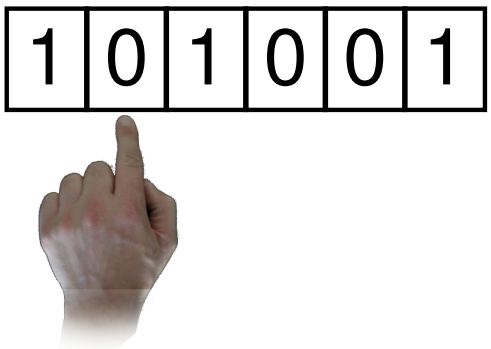
- Le but : une machine ($=$ procédure automatique) qui sache distinguer les mots en $\{0,1\}^*$ avec un nombre pair de 1
- Première idée : on compte le nombre de 1
- Deuxième idée : il suffit de sauvegarder que la parité des 1



impair

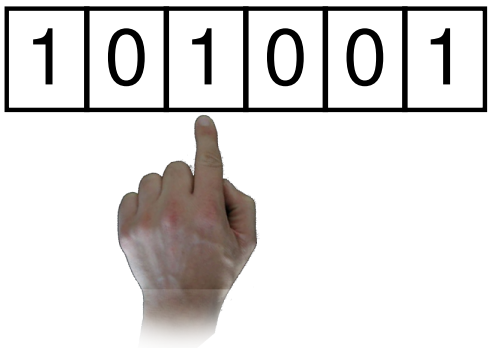
Reconnaître un langage : une première machine

- Le but : une machine (= procédure automatique) qui sache distinguer les mots en $\{0,1\}^*$ avec un nombre pair de 1
- Première idée : on compte le nombre de 1
- Deuxième idée : il suffit de sauvegarder que la parité des 1



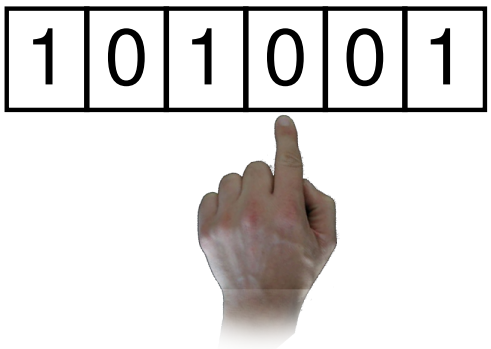
Reconnaître un langage : une première machine

- Le but : une machine (= procédure automatique) qui sache distinguer les mots en $\{0,1\}^*$ avec un nombre pair de 1
- Première idée : on compte le nombre de 1
- Deuxième idée : il suffit de sauvegarder que la parité des 1



Reconnaître un langage : une première machine

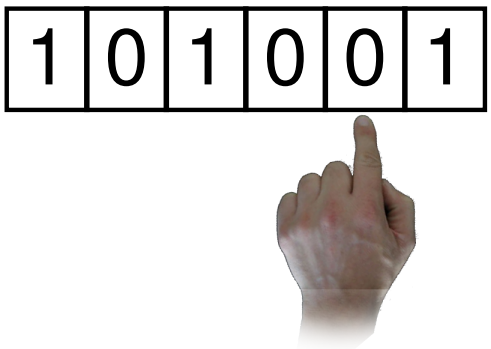
- Le but : une machine ($=$ procédure automatique) qui sache distinguer les mots en $\{0,1\}^*$ avec un nombre pair de 1
- Première idée : on compte le nombre de 1
- Deuxième idée : il suffit de sauvegarder que la parité des 1



pair

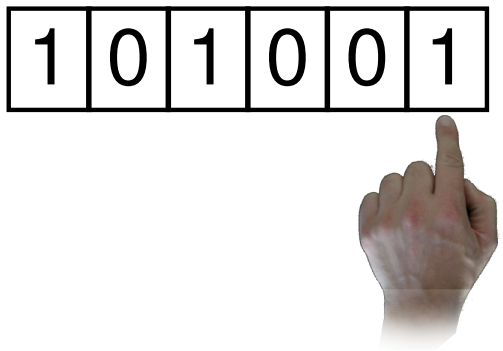
Reconnaître un langage : une première machine

- Le but : une machine ($=$ procédure automatique) qui sache distinguer les mots en $\{0,1\}^*$ avec un nombre pair de 1
- Première idée : on compte le nombre de 1
- Deuxième idée : il suffit de sauvegarder que la parité des 1



Reconnaître un langage : une première machine

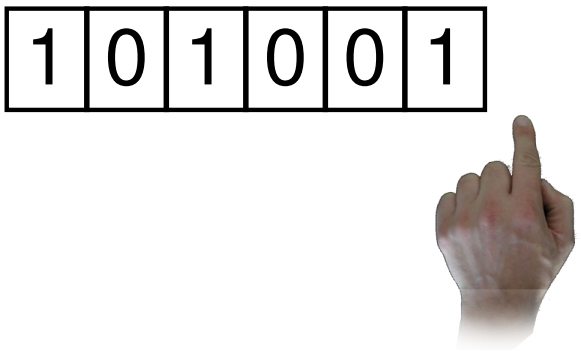
- Le but : une machine ($=$ procédure automatique) qui sache distinguer les mots en $\{0,1\}^*$ avec un nombre pair de 1
- Première idée : on compte le nombre de 1
- Deuxième idée : il suffit de sauvegarder que la parité des 1



impair

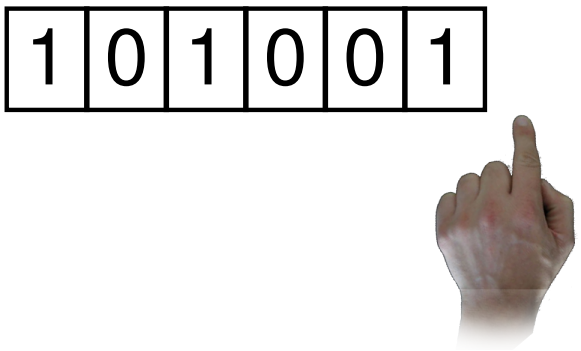
Reconnaître un langage : une première machine

- Le but : une machine (= procédure automatique) qui sache distinguer les mots en $\{0,1\}^*$ avec un nombre pair de 1
- Première idée : on compte le nombre de 1
- Deuxième idée : il suffit de sauvegarder que la parité des 1

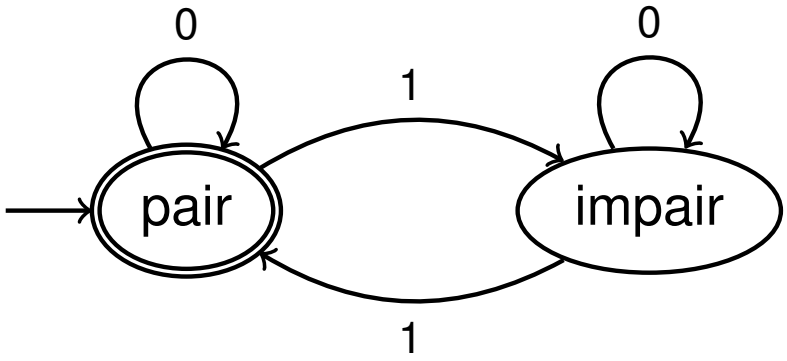


Reconnaître un langage : une première machine

- Le but : une machine (= procédure automatique) qui sache distinguer les mots en $\{0,1\}^*$ avec un nombre pair de 1
- Première idée : on compte le nombre de 1
- Deuxième idée : il suffit de sauvegarder que la parité des 1

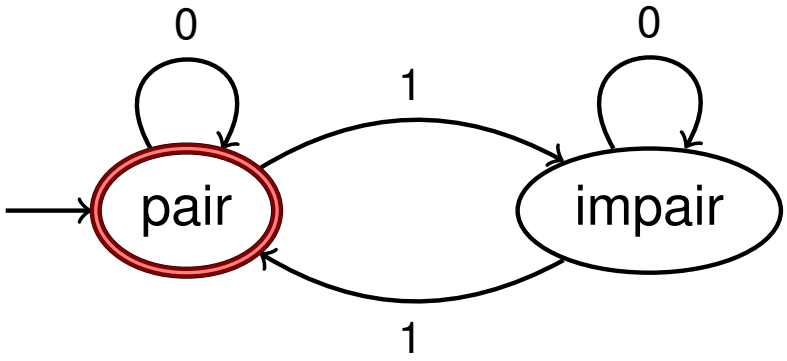


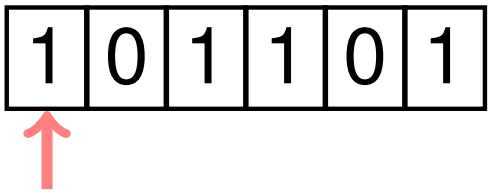
Automate finis : le premier rencontre



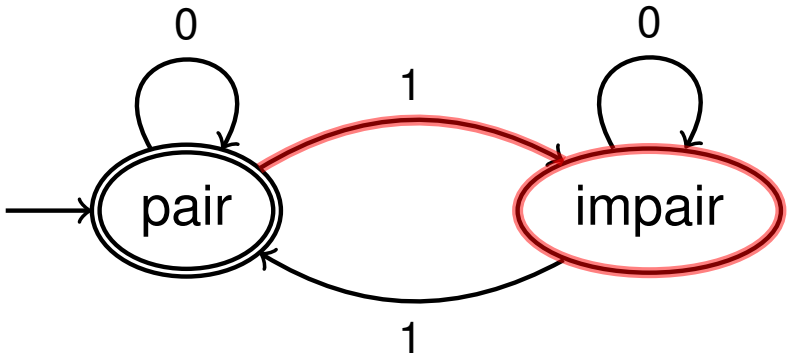
1	0	1	1	0	1

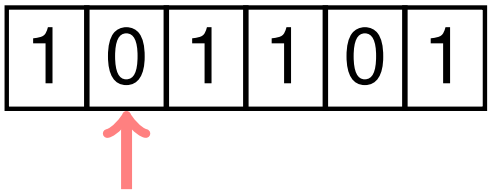
Automate finis : le premier rencontre



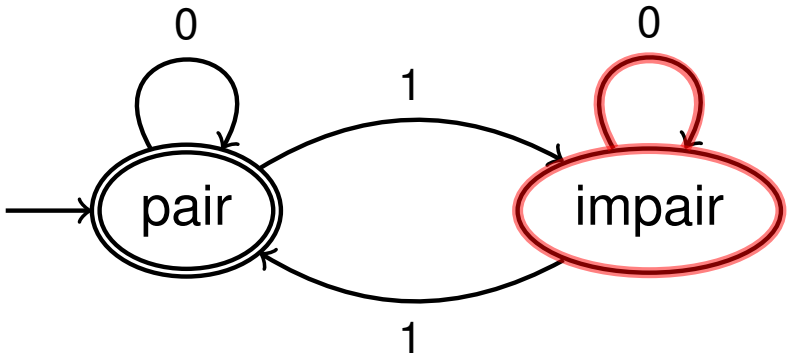


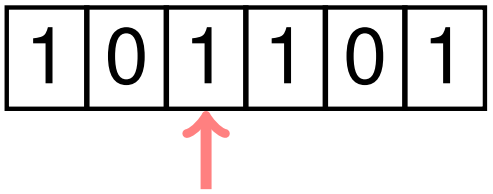
Automate finis : le premier rencontre



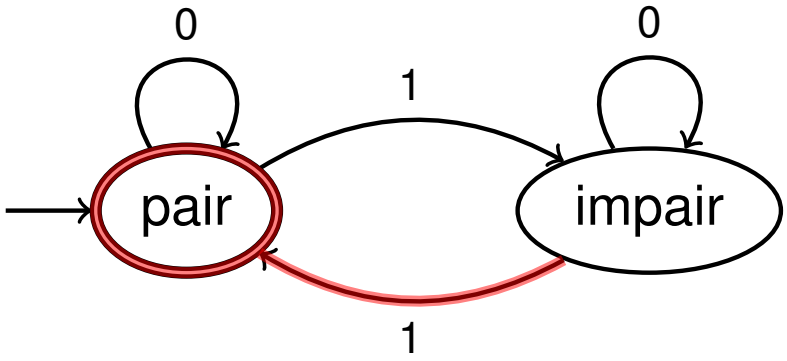


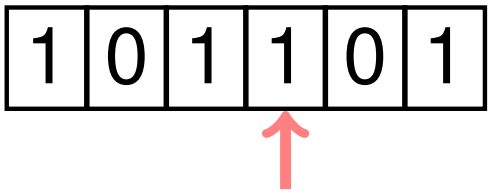
Automate finis : le premier rencontre



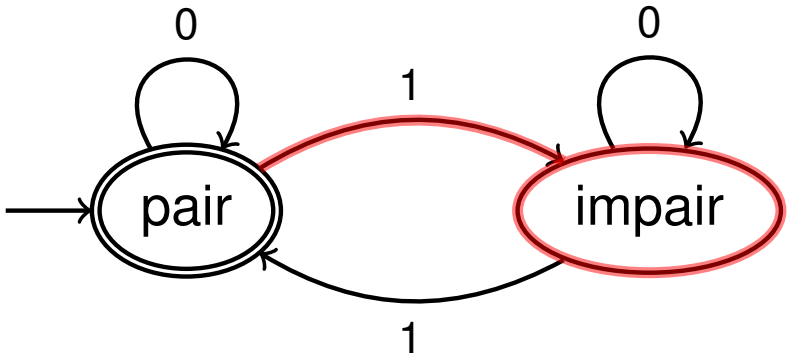


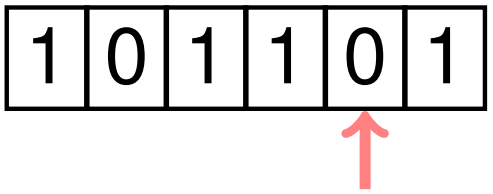
Automate finis : le premier rencontre



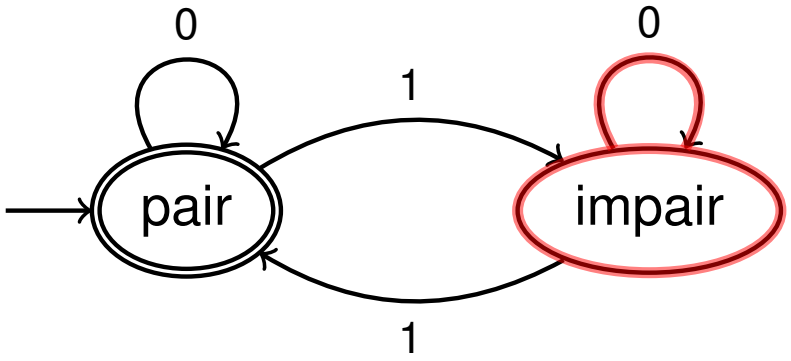


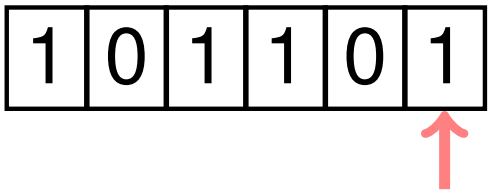
Automate finis : le premier rencontre



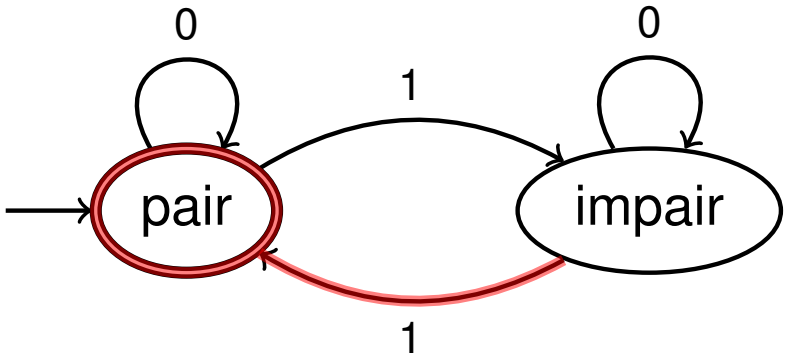


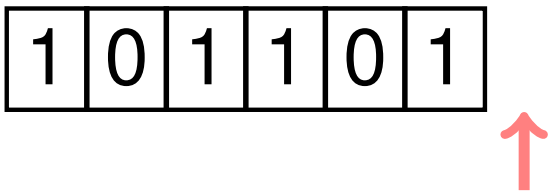
Automate finis : le premier rencontre



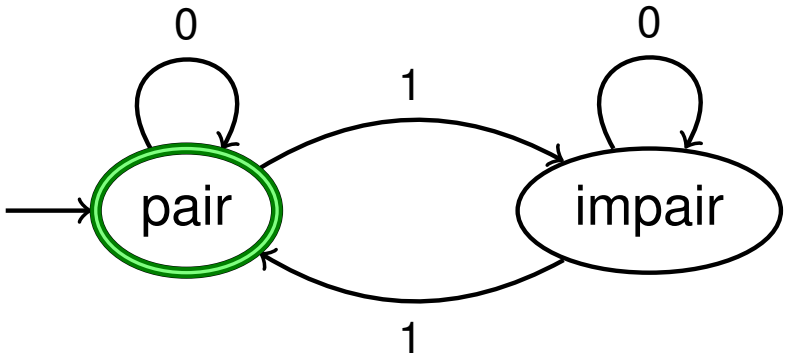


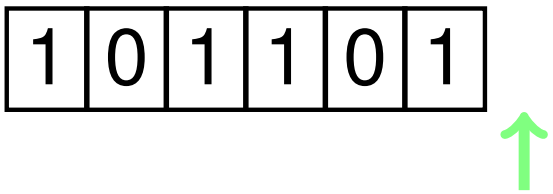
Automate finis : le premier rencontre



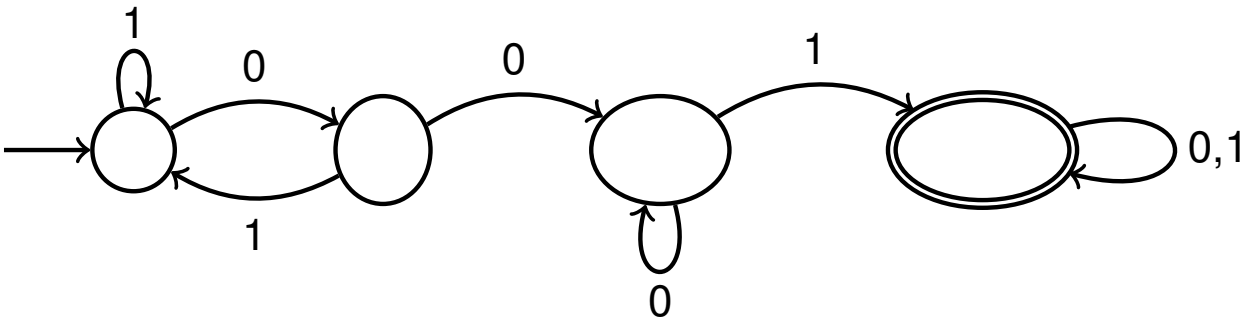


Automate finis : le premier rencontre



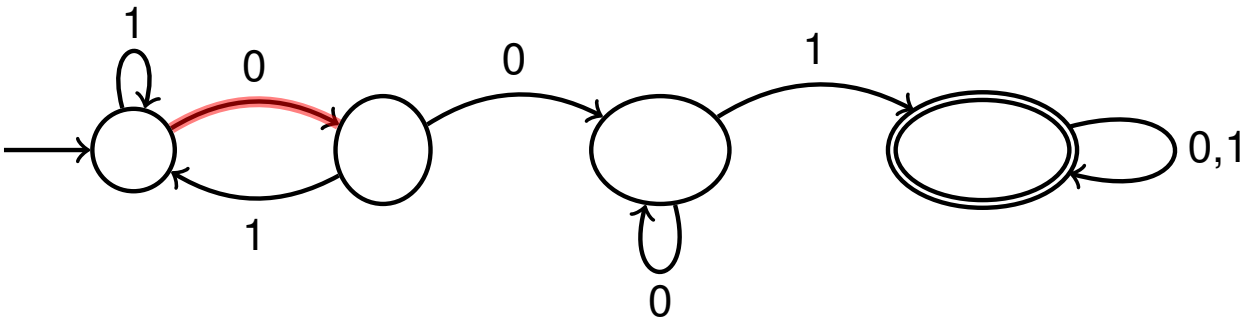


Un autre exemple



Pour créer exemples de mots acceptés, on suive le graphe au hasard

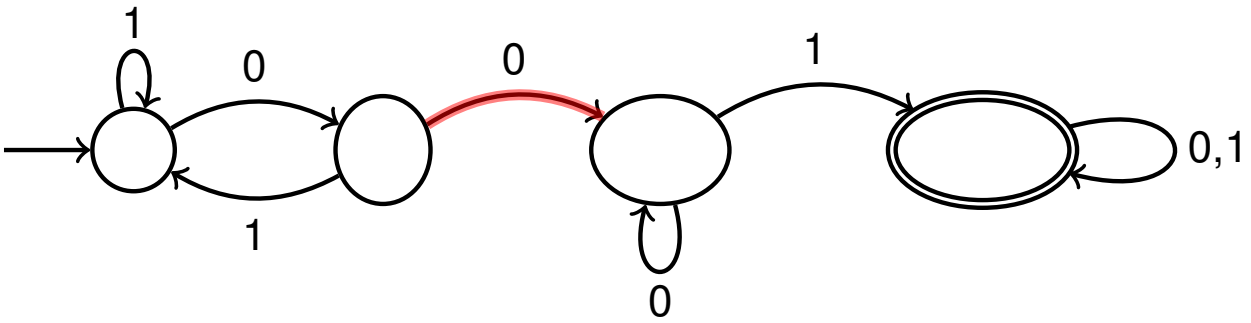
Un autre exemple



Pour créer exemples de mots acceptés, on suive le graphe au hasard

- 0

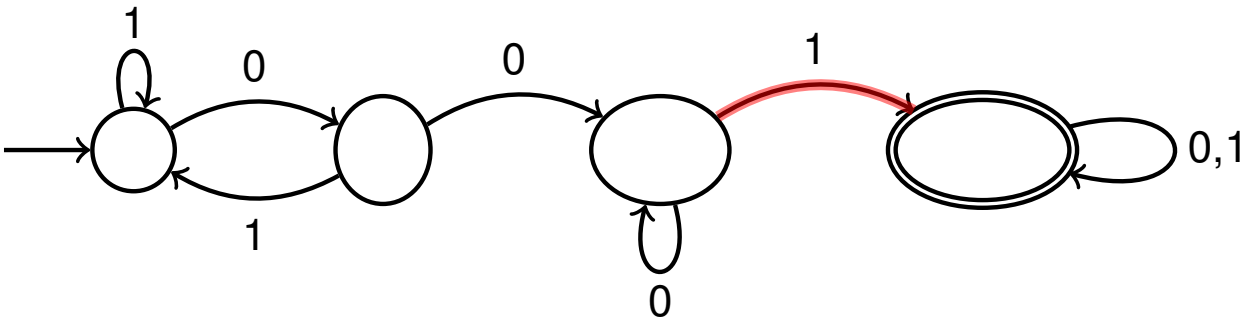
Un autre exemple



Pour créer exemples de mots acceptés, on suive le graphe au hasard

- 00

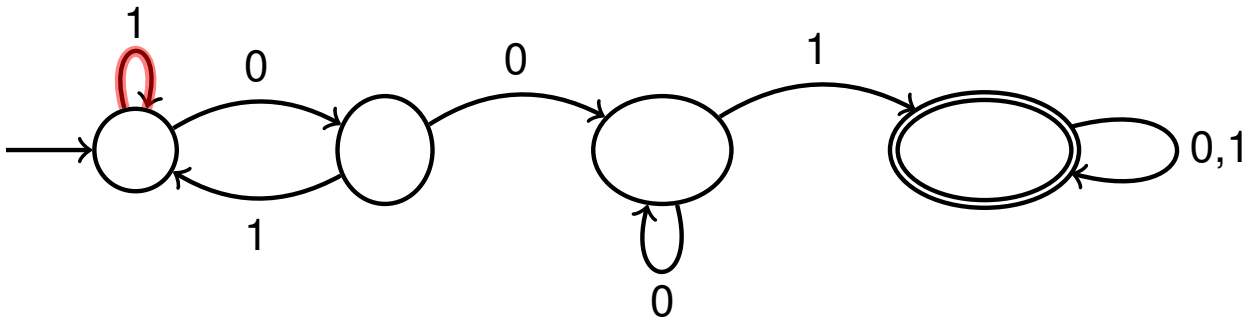
Un autre exemple



Pour créer exemples de mots acceptés, on suive le graphe au hasard

- 001

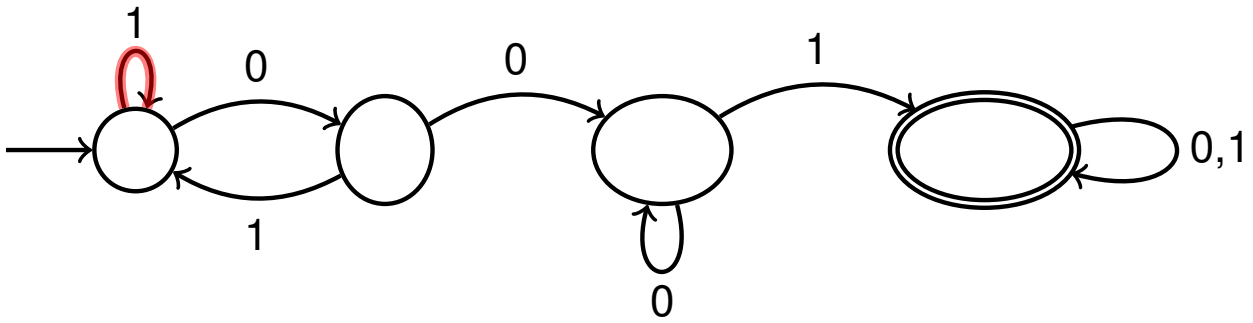
Un autre exemple



Pour créer exemples de mots acceptés, on suive le graphe au hasard

- 001
- 1

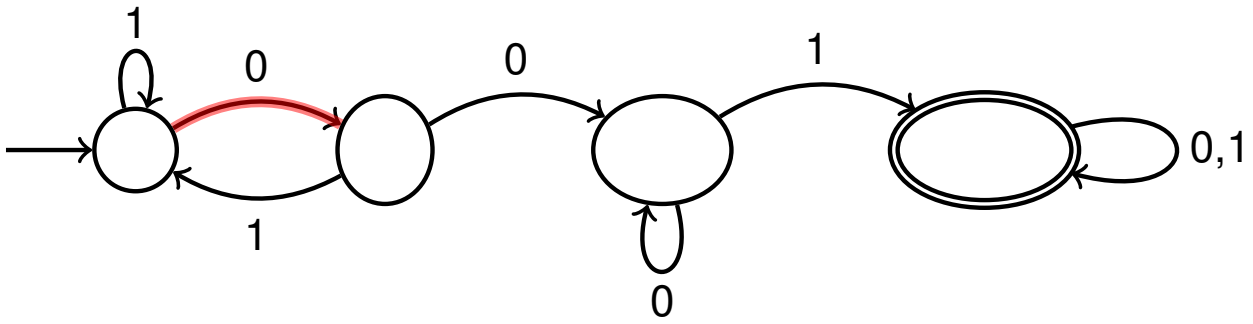
Un autre exemple



Pour créer exemples de mots acceptés, on suive le graphe au hasard

- 001
- 11

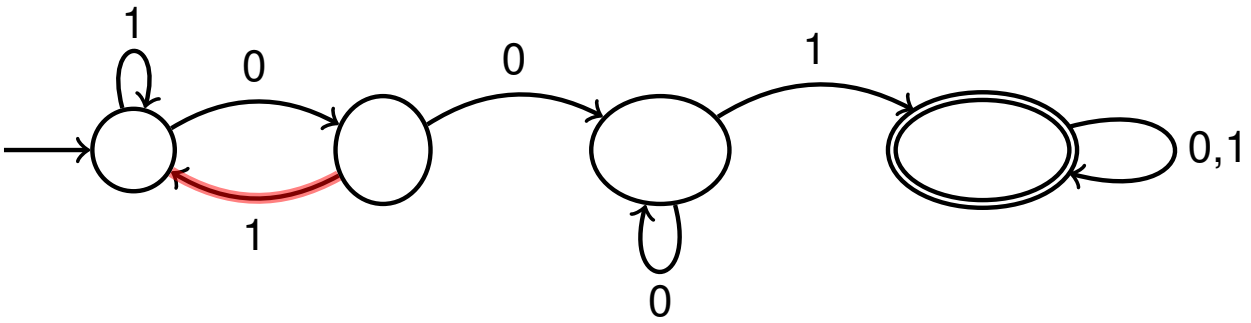
Un autre exemple



Pour créer exemples de mots acceptés, on suive le graphe au hasard

- 001
- 110

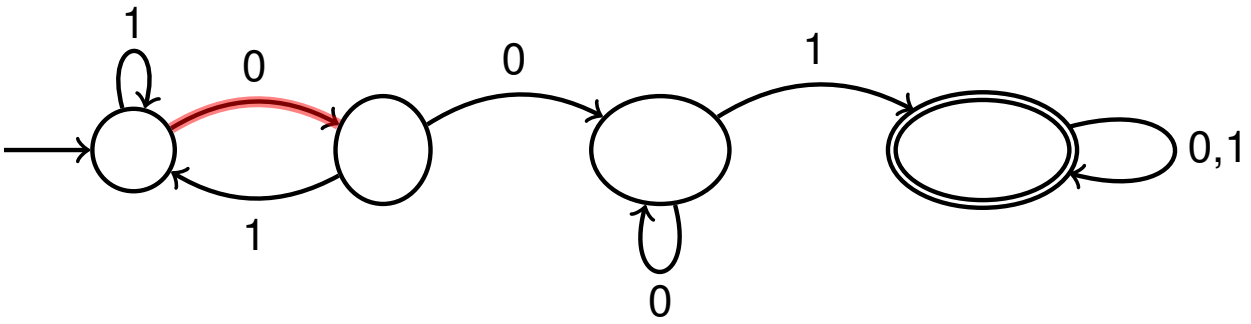
Un autre exemple



Pour créer exemples de mots acceptés, on suive le graphe au hasard

- 001
- 1101

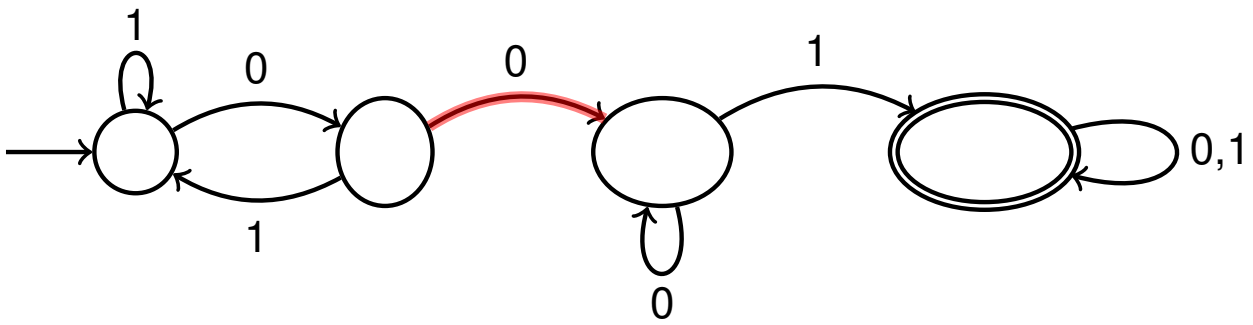
Un autre exemple



Pour créer exemples de mots acceptés, on suive le graphe au hasard

- 001
- 11010

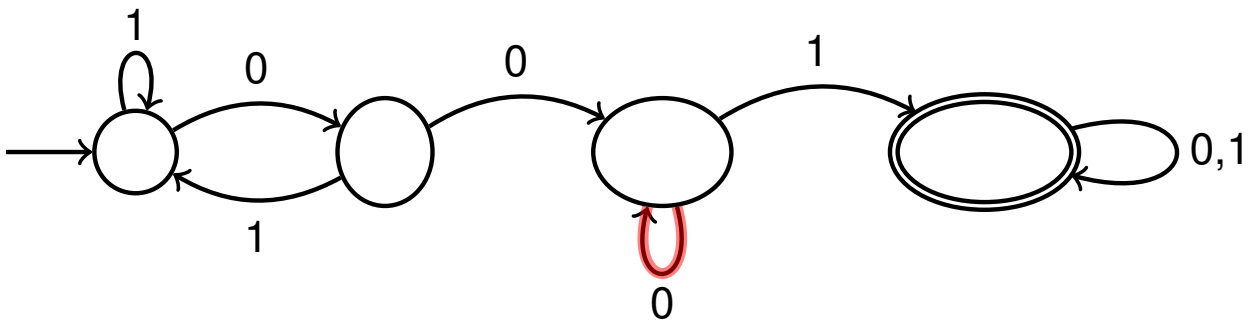
Un autre exemple



Pour créer exemples de mots acceptés, on suive le graphe au hasard

- 001
- 110100

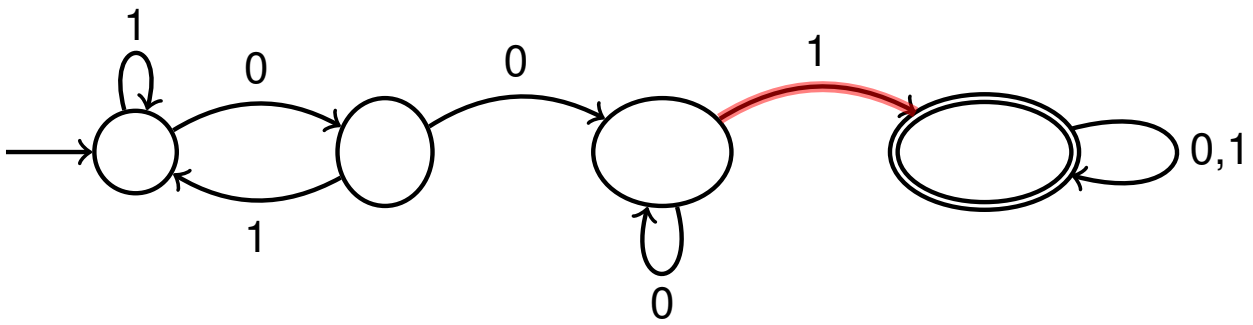
Un autre exemple



Pour créer exemples de mots acceptés, on suive le graphe au hasard

- 001
- 1101000

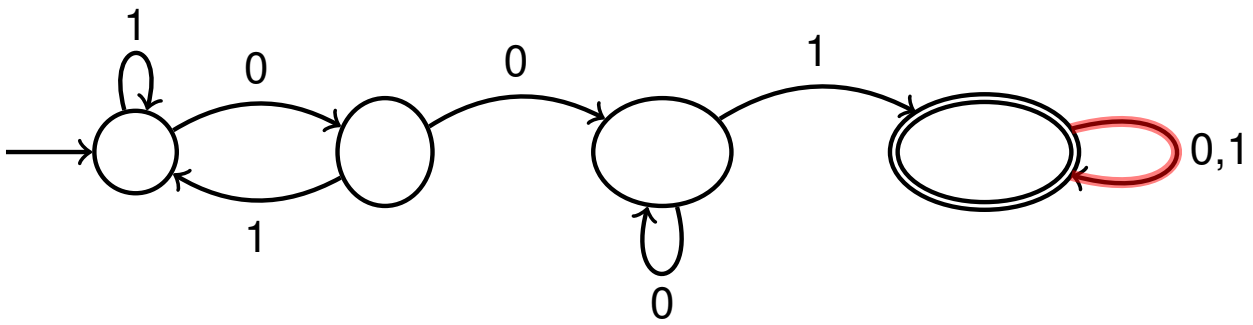
Un autre exemple



Pour créer exemples de mots acceptés, on suive le graphe au hasard

- 001
- 11010001

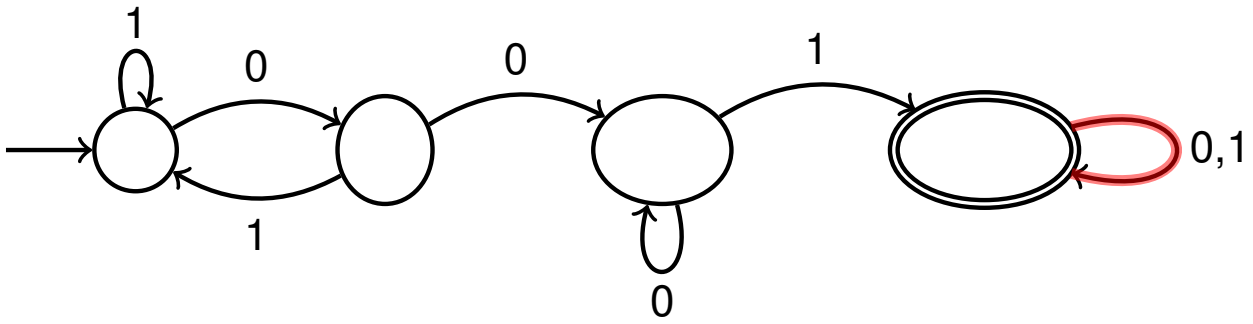
Un autre exemple



Pour créer exemples de mots acceptés, on suive le graphe au hasard

- 001
- 110100010

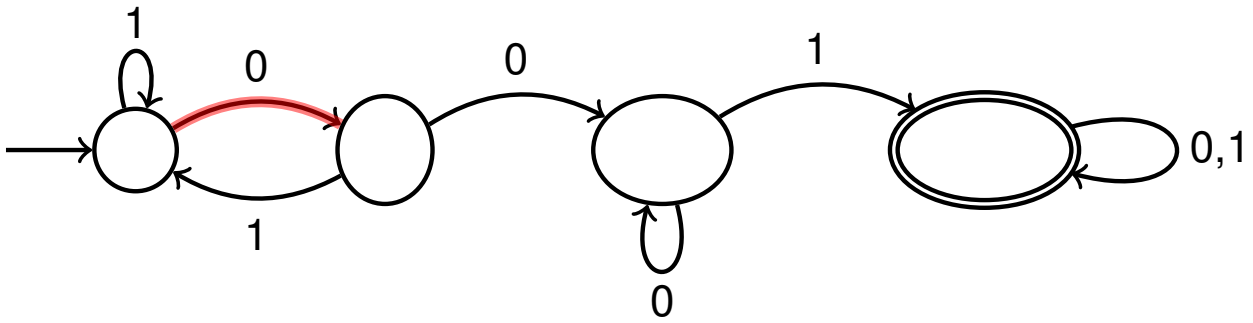
Un autre exemple



Pour créer exemples de mots acceptés, on suive le graphe au hasard

- 001
- 1101000101

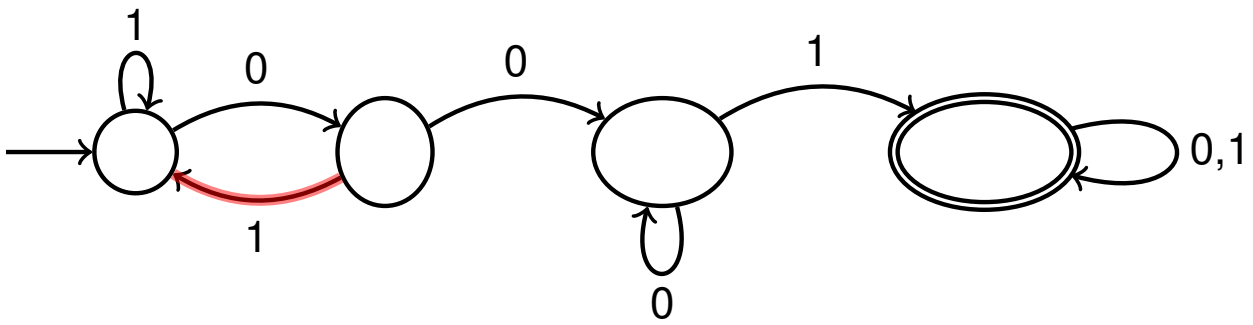
Un autre exemple



Pour créer exemples de mots acceptés, on suive le graphe au hasard

- 001
- 1101000101
- 0

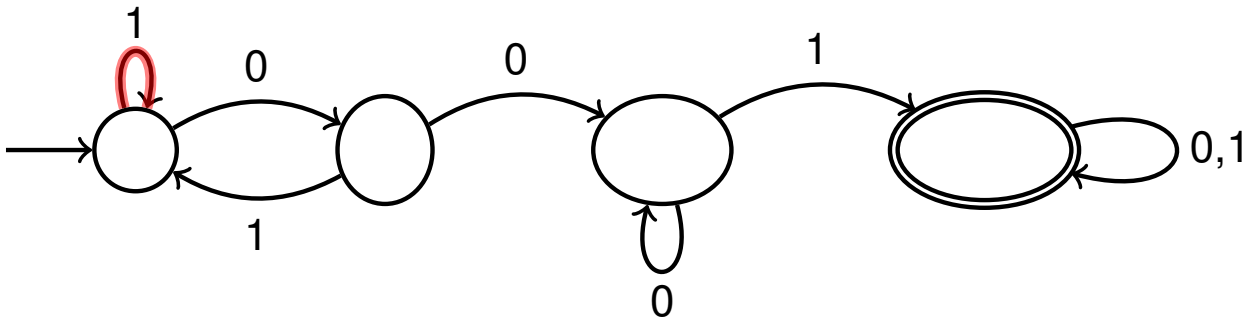
Un autre exemple



Pour créer exemples de mots acceptés, on suive le graphe au hasard

- 001
- 1101000101
- 01

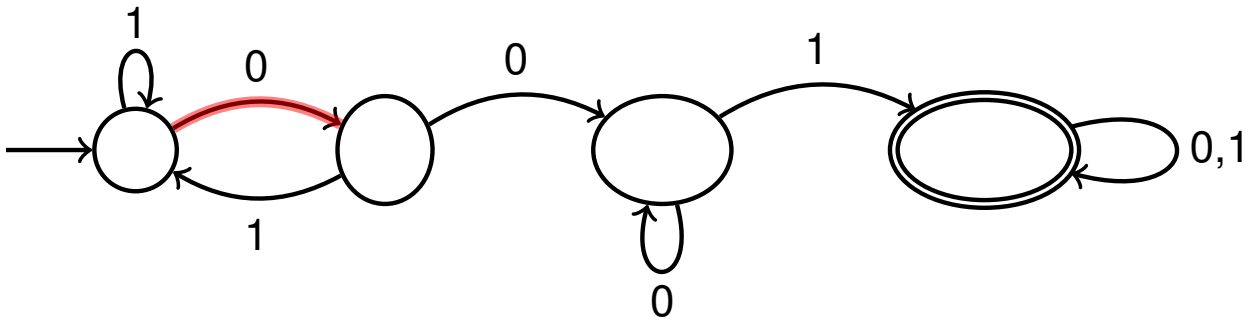
Un autre exemple



Pour créer exemples de mots acceptés, on suive le graphe au hasard

- 001
- 1101000101
- 011

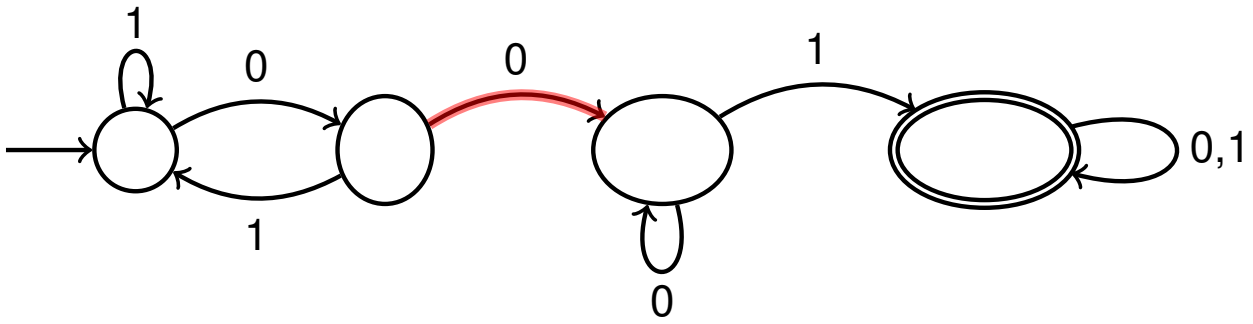
Un autre exemple



Pour créer exemples de mots acceptés, on suive le graphe au hasard

- 001
- 1101000101
- 0110

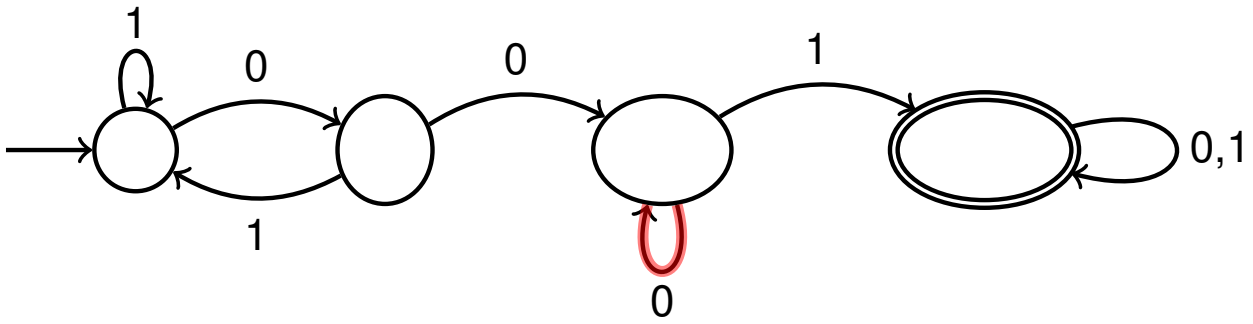
Un autre exemple



Pour créer exemples de mots acceptés, on suive le graphe au hasard

- 001
- 1101000101
- 01100

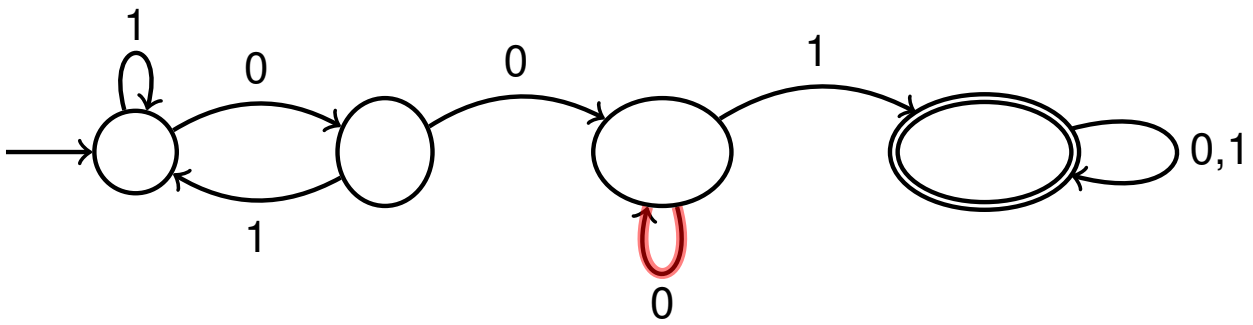
Un autre exemple



Pour créer exemples de mots acceptés, on suive le graphe au hasard

- 001
- 1101000101
- 011000

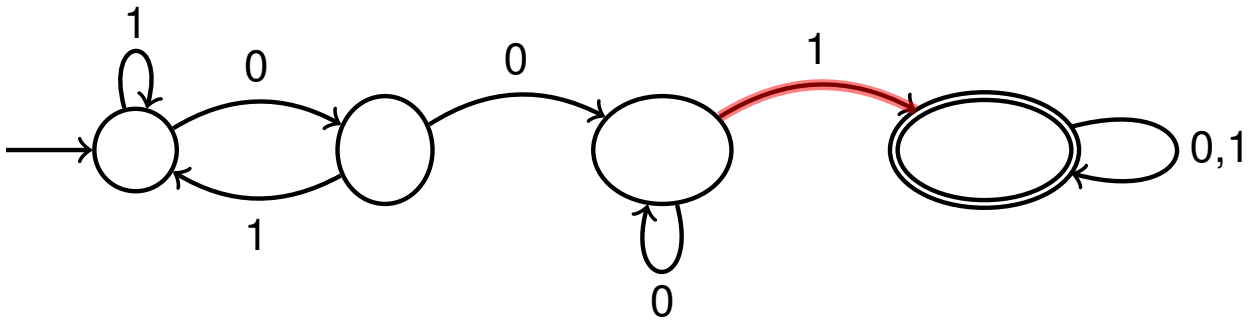
Un autre exemple



Pour créer exemples de mots acceptés, on suive le graphe au hasard

- 001
- 1101000101
- 0110000

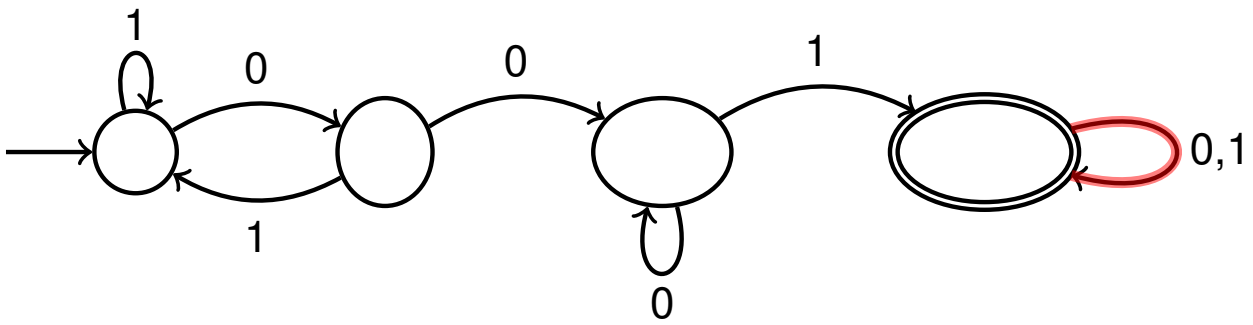
Un autre exemple



Pour créer exemples de mots acceptés, on suive le graphe au hasard

- 001
- 1101000101
- 01100001

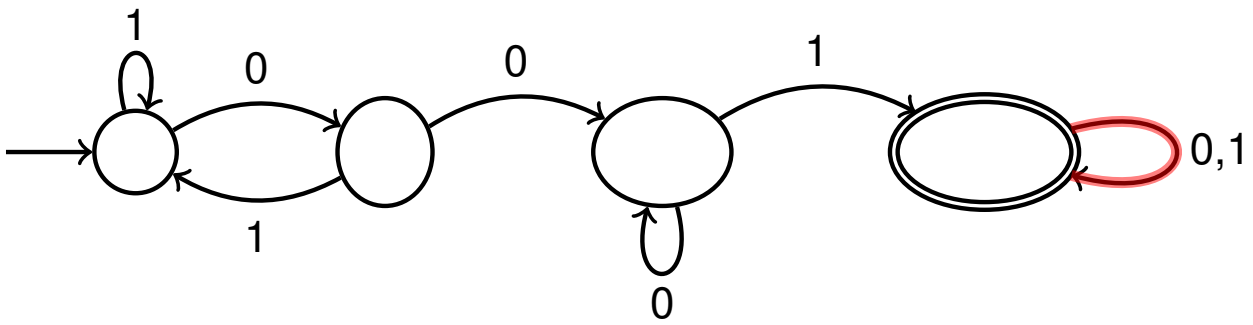
Un autre exemple



Pour créer exemples de mots acceptés, on suive le graphe au hasard

- 001
- 1101000101
- 011000010

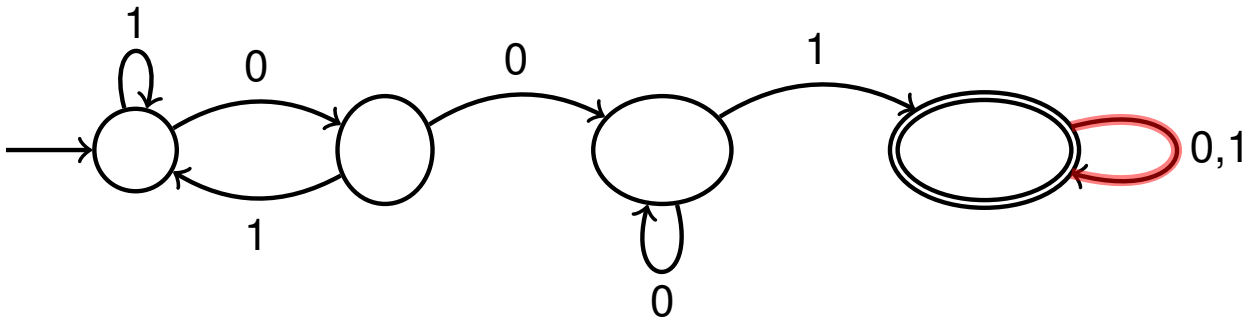
Un autre exemple



Pour créer exemples de mots acceptés, on suive le graphe au hasard

- 001
- 1101000101
- 0110000100

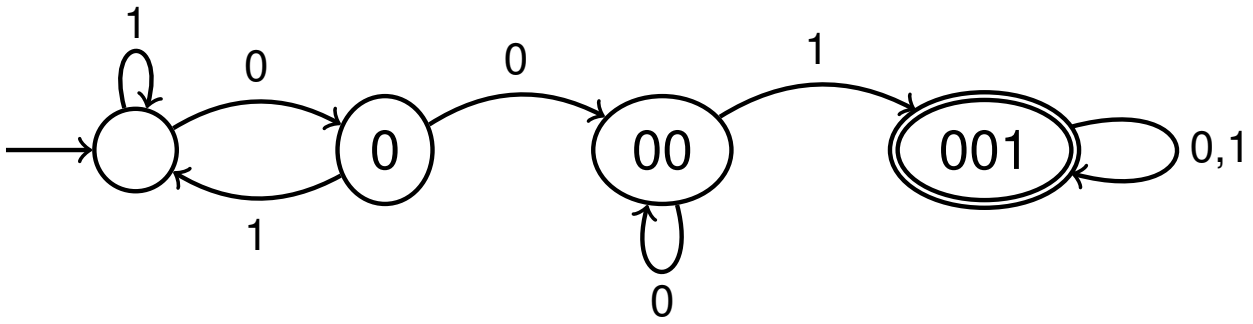
Un autre exemple



Pour créer exemples de mots acceptés, on suive le graphe au hasard

- 001
- 1101000101
- 01100001001

Un autre exemple



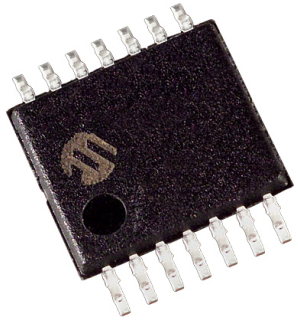
Pour créer exemples de mots acceptés, on suive le graphe au hasard

- 001
- 1101000101
- 01100001001

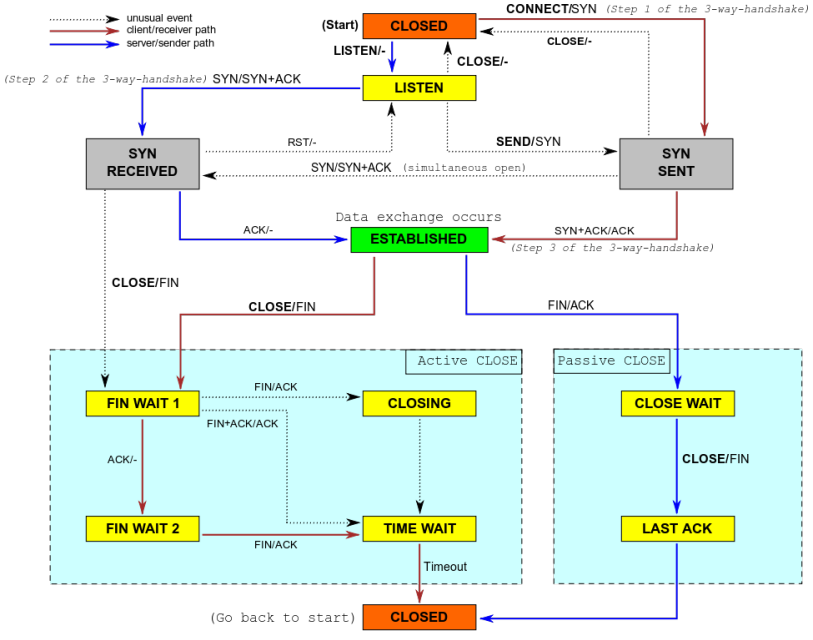
Il reconnait le chaîne contenant 001

. . . mais pourquoi ?

électroménagers



puces électriques



protocoles internet

. . .

```
a +(c|h)at
```

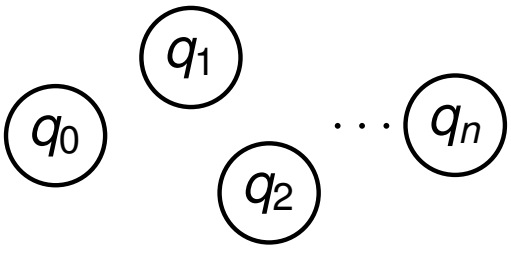
recherches textuelles

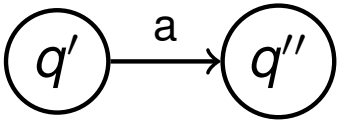
Définition d'automate fini

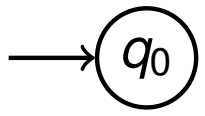
Un automate fini M est donné par

$$
\begin{aligned}
M = (\\
&Q \text{ ensemble fini}, && \text{(les états)}\\
&\Sigma \text{ alphabet}, && \text{(l'alphabet sur le quel } M \text{ travaille)}\\
&\delta : Q \times \Sigma \to Q, && \text{(la fonction de transition)}\\
&q_0 \in Q, && \text{(l'état initial)}\\
&F \subseteq Q, && \text{(les états accepteurs)}\\
)
\end{aligned}
$$

Dessin ou définition ? C'est équivalent !

$Q = \{ q_0, q_1, \ldots, q_n \} \leftrightsquigarrow$ 

$\delta(q', a) = q'' \leftrightsquigarrow$ 

q_0 initial $\leftrightsquigarrow$ 

$q \in F \leftrightsquigarrow$ 

Définition de calcul

- Comment calcule une machine M ?
- Le calcul de $M = (Q, \Sigma, \delta, q_0, F)$ sur un mot $w \in \Sigma^*$ est la séquence $r_0, r_1, \ldots, r_{|w|} \in Q$ définie par
  - $r_0 = q_0$ (le calcul commence sur l'état initial)
  - $r_{k+1} = \delta(r_k, w_k)$ (chaque pas dépende de l'état et du caractère lu)
- Si $r_{|w|} \in F$ on dit que M accepte w
- $L(M) = \{ w \mid M \text{ accepte } w \}$ est le langage reconnu par M

Expressions régulières

On souhaite caractériser les langages reconnus par des automates.
C'est possible !

Définition

Étant fixé un alphabet Σ, les langages réguliers $\mathcal{R}(\Sigma)$ sont le plus petit ensemble dans $\mathcal{P}(\Sigma^*)$ t.q.

- $\{a\} \in \mathcal{R}(\Sigma)$ pour $a \in \Sigma$ (notation $a = \{a\}$)
- $\emptyset \in \mathcal{R}(\Sigma)$
- $R_1 \cup R_2 \in \mathcal{R}(\Sigma)$ pour $R_1, R_2 \in \mathcal{R}(\Sigma)$ (parfois notation $R_1|R_2$)
- $R_1 R_2 = \{ vw \mid v \in R_1, w \in R_2 \} \in \mathcal{R}(\Sigma)$ pour $R_1, R_2 \in \mathcal{R}(\Sigma)$
- $R^* = \{ w_1 w_2 \cdots w_k \mid k \in \mathbb{N}, \forall i : w_i \in R \} \in \mathcal{R}(\Sigma)$ pour $R \in \mathcal{R}(\Sigma)$

Théorème

L régulier $\iff$ $L = L(M)$ pour un automate fini sur Σ