Automates et grammaires (1/2)

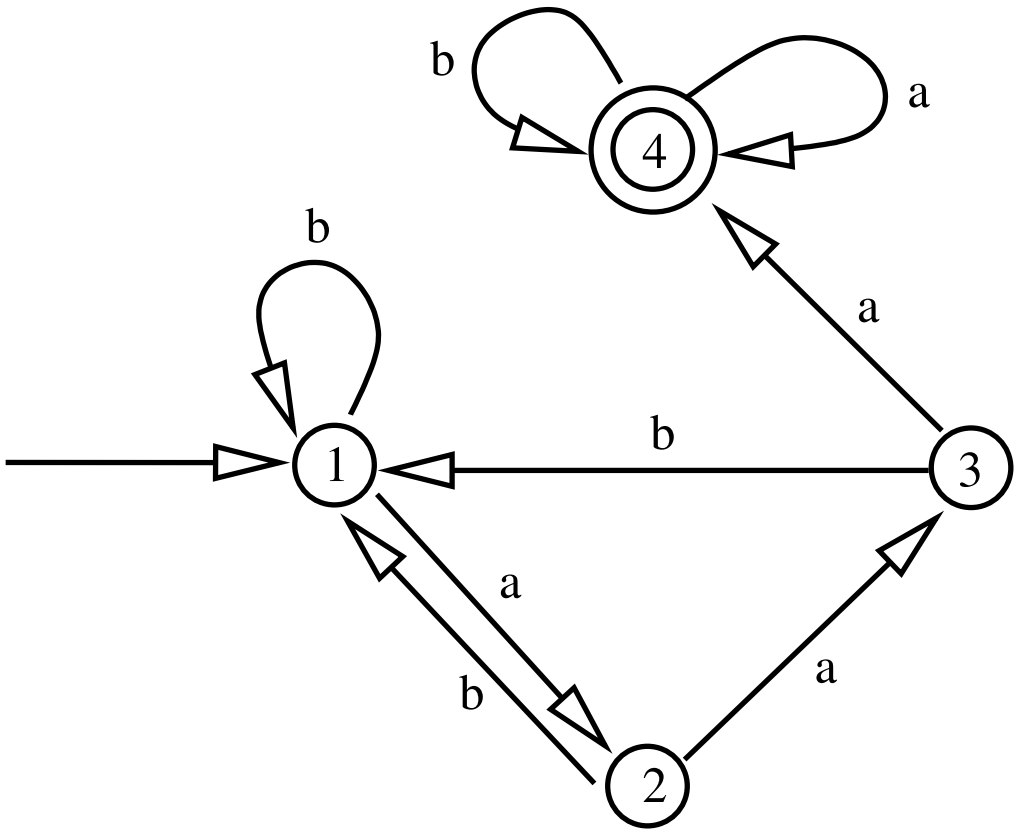
Malika More
(malika.more@u-clermont1.fr)

IREM Clermont-Ferrand

Formation ISN

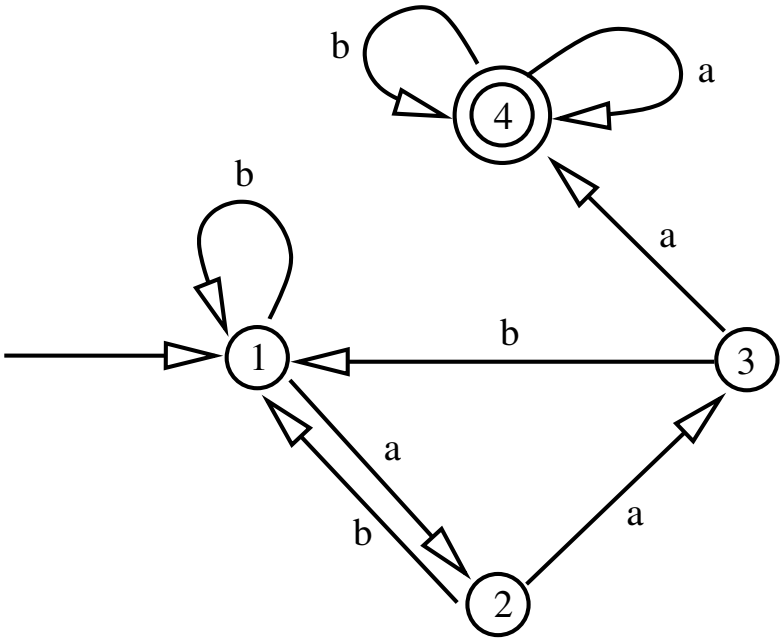
02 février 2012

Un petit labyrinthe

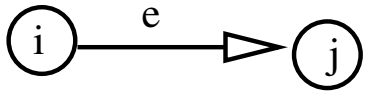


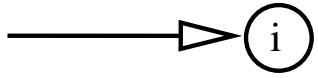
- *abaab*
- *aababaaa*
- *baabaaaaabaab*
- Cas général ?

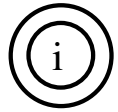
L'automate



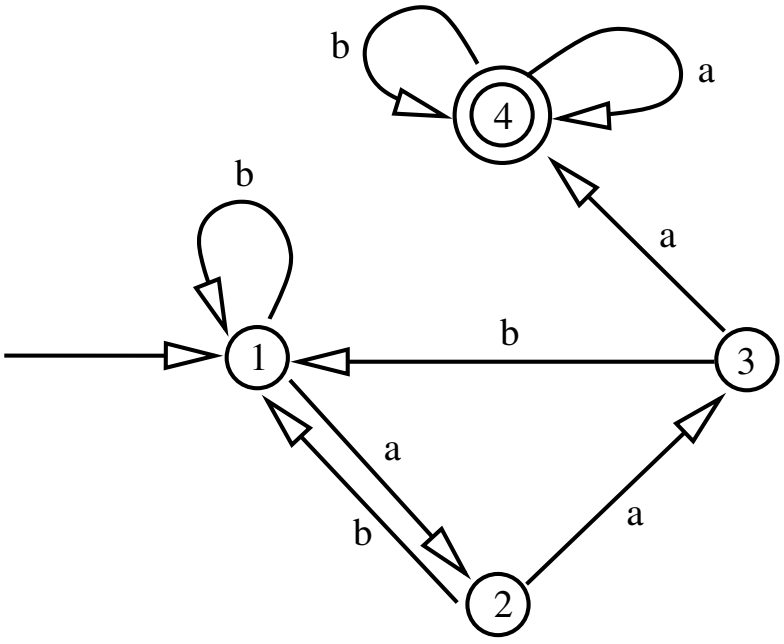
- **État** 

- **Transition** 

- **État initial** 

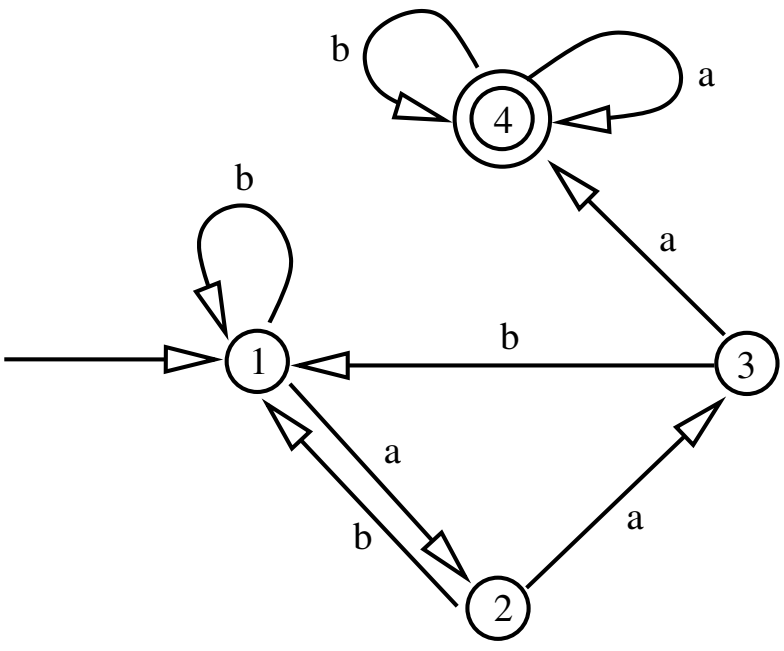
- **État(s) acceptant(s)** 

Les objets



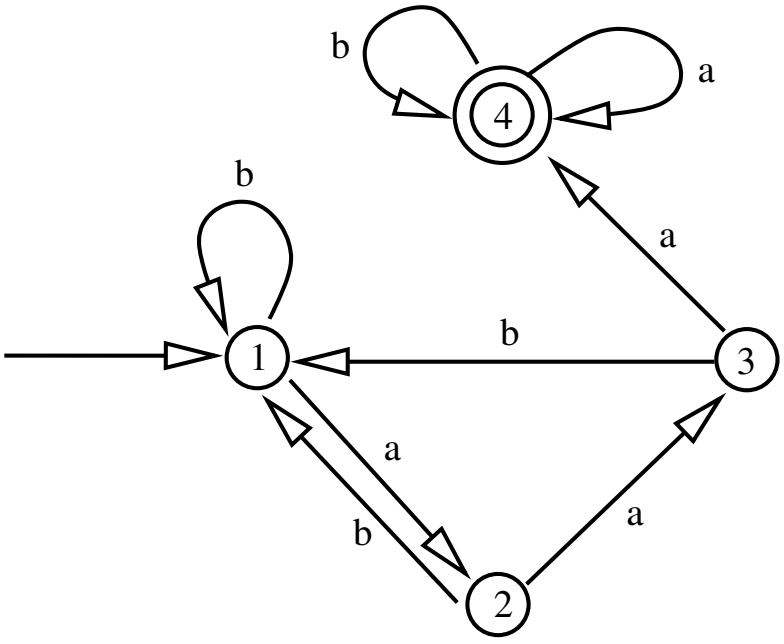
- **Lettres** a, b, etc.
- **Alphabet** $\Sigma = \{a, b\}$ (un ensemble fini de lettres)
- **Mot** $w = abaab$ (une suite finie de lettres de l'alphabet)
- **Mot vide** ε (qui ne contient aucune lettre)

Le calcul



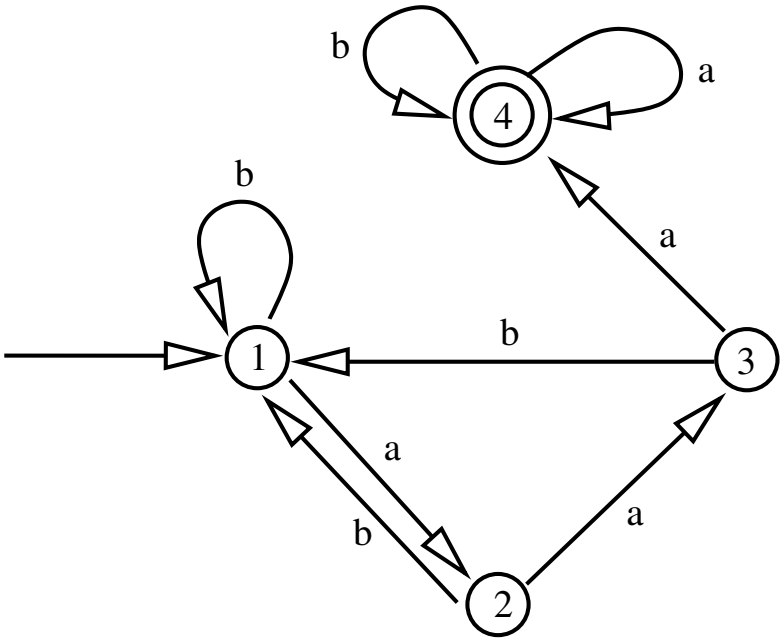
- Donnée : *w*
- Début : état initial
- `Lire` ***w*** `de gauche à droite lettre par lettre en suivant les transitions`
- Fin :
  - Mot terminé
  - Pas de transition possible
- Réponse :
  - `Accepté`
  - `Refusé`

Les langages



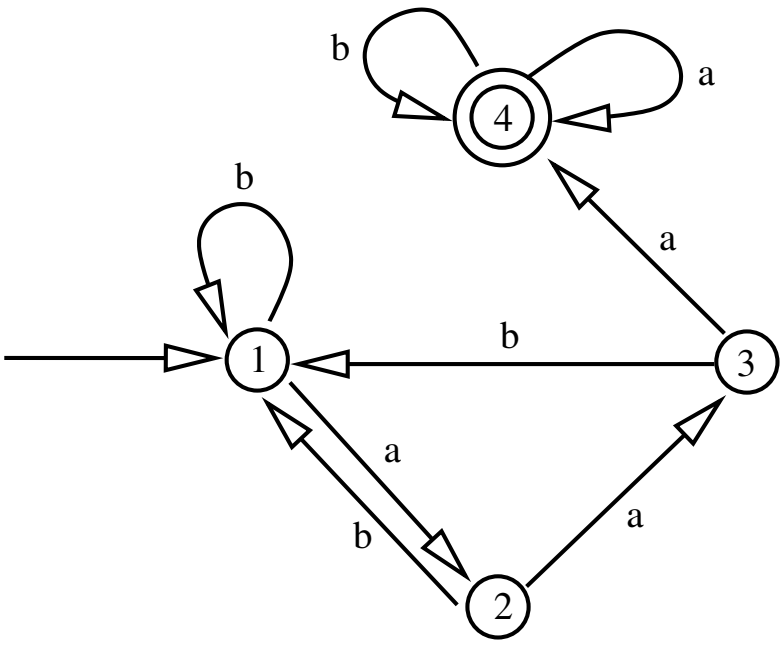
- Σ^* (l'ensemble de tous les mots finis composés de lettres de l'alphabet)
- **Langage** $\mathcal{L}$ (un ensemble de mots)
  - $\mathcal{L}$ fini ou infini
  - $\mathcal{L} \subseteq \Sigma^*$
  - Cas particuliers : Σ^*, $\varnothing$, $\{\varepsilon\}$

Les langages



- **Le langage de l'automate** : Ensemble des mots acceptés par l'automate
- On parle aussi du langage (ou des mots) reconnu(s) par l'automate.

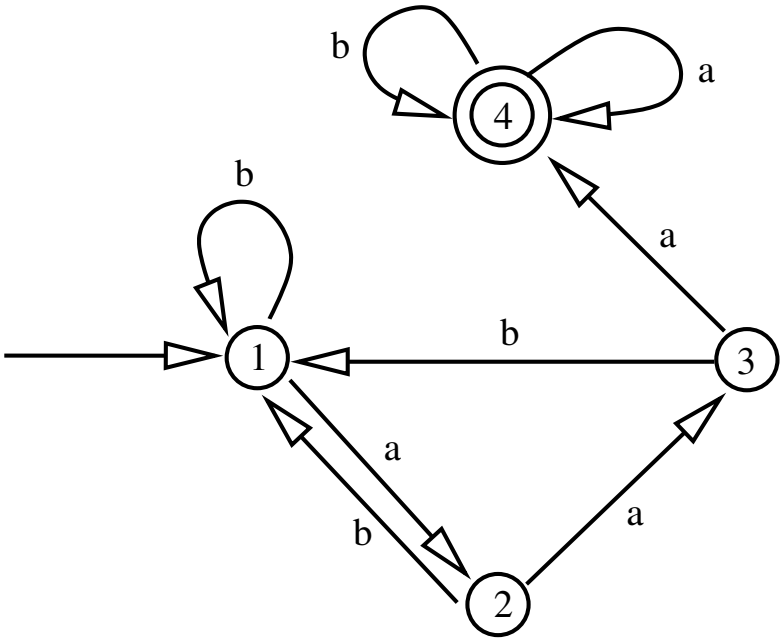
Définition formelle



- Alphabet $\Sigma = \{a, b\}$
- Ensemble (fini) d'états $Q = \{1, 2, 3, 4\}$
  - État initial $q_0 = 1$
  - État(s) acceptant(s) $Q_A = \{4\}$
- Table de transition δ

	a	b
1	2	1
2	3	1
3	4	1
4	4	4

Utilisations



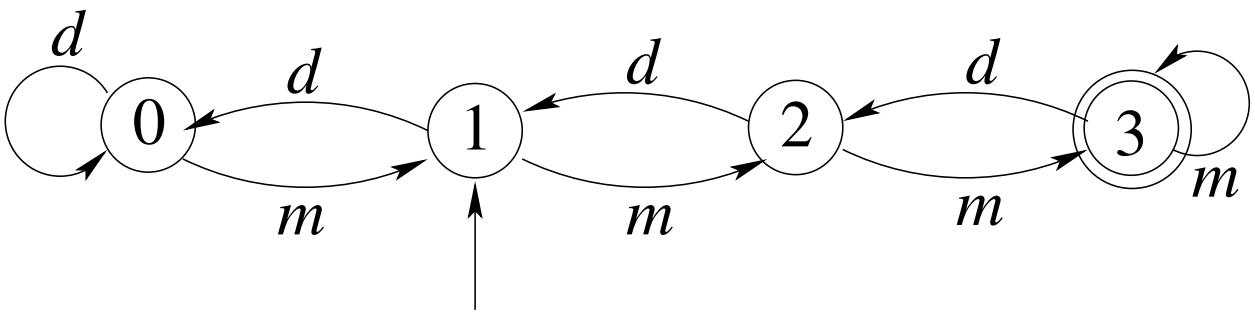
- Modéliser des systèmes discrets possédant un nombre fini d'états (spécification, vérification)
- Algorithmique du texte
- Informatique théorique (« modèle de calcul »)

Monter le son

- Le bouton de la télécommande permettant de régler le son de la TV

Monter le son

- Le bouton de la télécommande permettant de régler le son de la TV

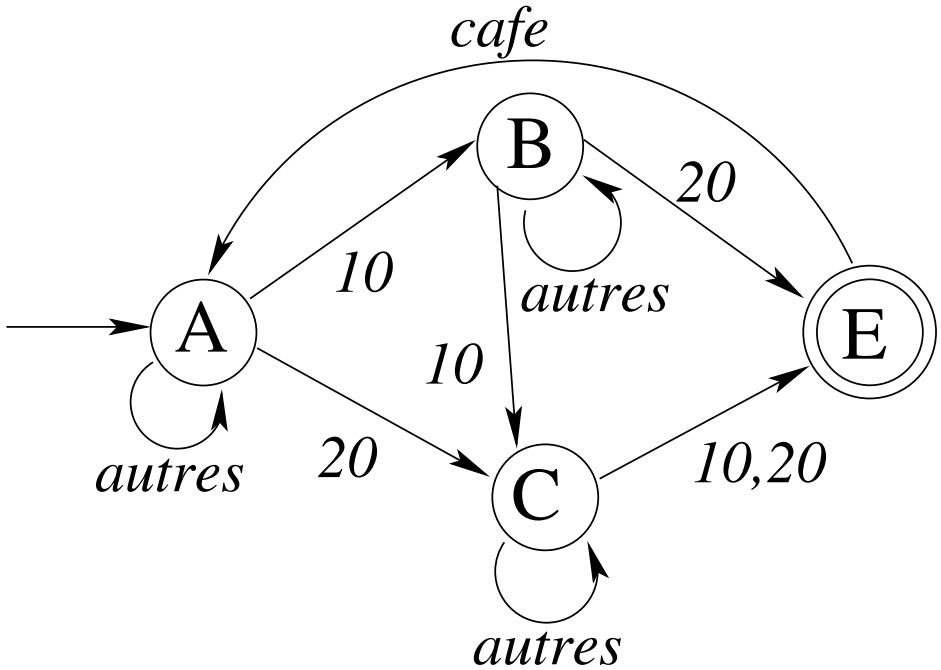


Machine à café

- Une machine à café simple accepte les pièces de 10 centimes et 20 centimes
- Un café coûte 30 centimes
- Quand le montant est suffisant, elle libère un bouton permettant d'obtenir le café
- Elle ne rend pas la monnaie

Machine à café

- Une machine à café simple accepte les pièces de 10 centimes et 20 centimes
- Un café coûte 30 centimes
- Quand le montant est suffisant, elle libère un bouton permettant d'obtenir le café
- Elle ne rend pas la monnaie

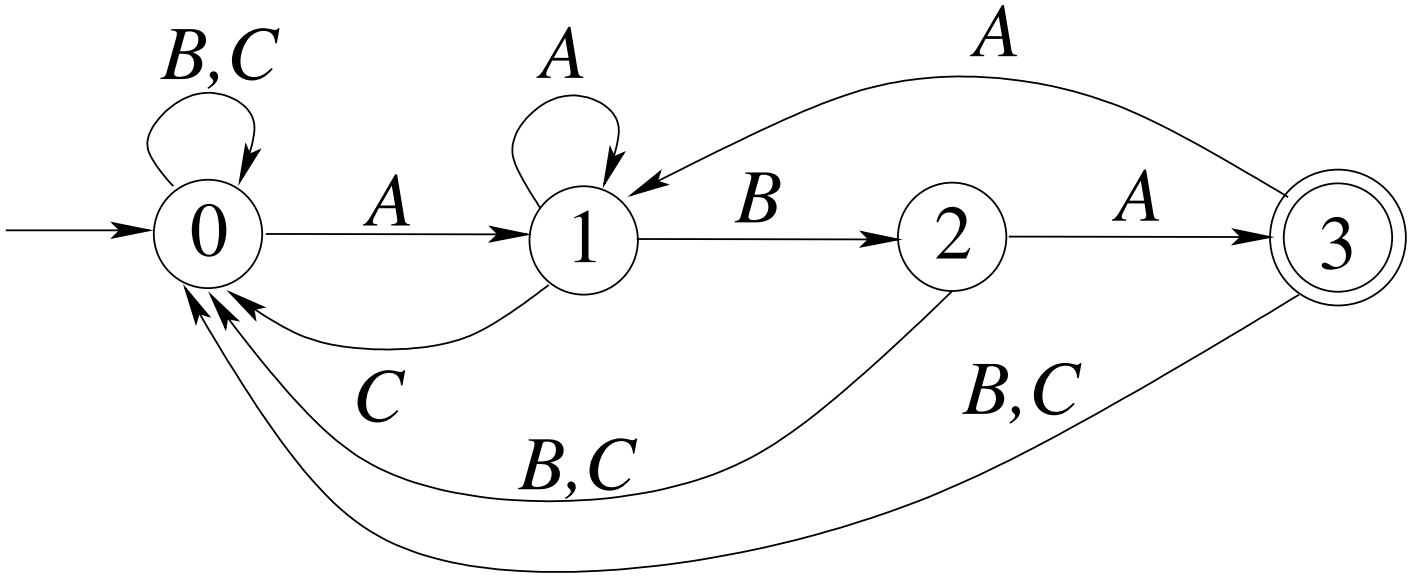


Digicode

- Un digicode avec trois touches A, B, C ouvre la porte quand on tape ABA

Digicode

- Un digicode avec trois touches A, B, C ouvre la porte quand on tape ABA



Caissière

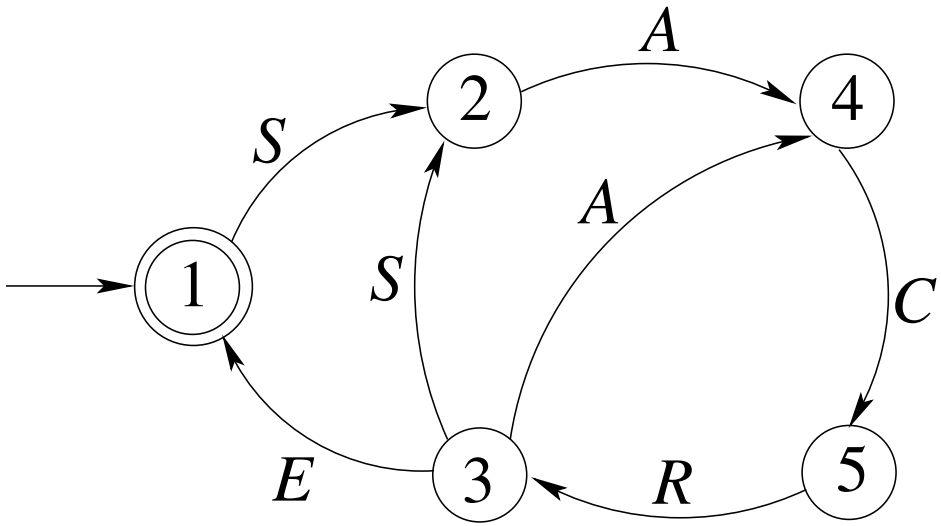
Dans un supermarché, le travail d'une caissière peut être décrit ainsi (l'alphabet est $\sigma = \{S, A, C, R, E\}$) :

- elle prépare un sac (S)
- puis pour chaque article : elle prend l'article (A), lit son code barre (C) et le range dans le sac (R)
- ceci jusqu'à épuisement des articles, ensuite elle encaisse (E)
- puis elle passe au client suivant et recommence.

Caissière

Dans un supermarché, le travail d'une caissière peut être décrit ainsi (l'alphabet est $\sigma = \{S, A, C, R, E\}$) :

- elle prépare un sac (S)
- puis pour chaque article : elle prend l'article (A), lit son code barre (C) et le range dans le sac (R)
- ceci jusqu'à épuisement des articles, ensuite elle encaisse (E)
- puis elle passe au client suivant et recommence.



L'énigme du passeur

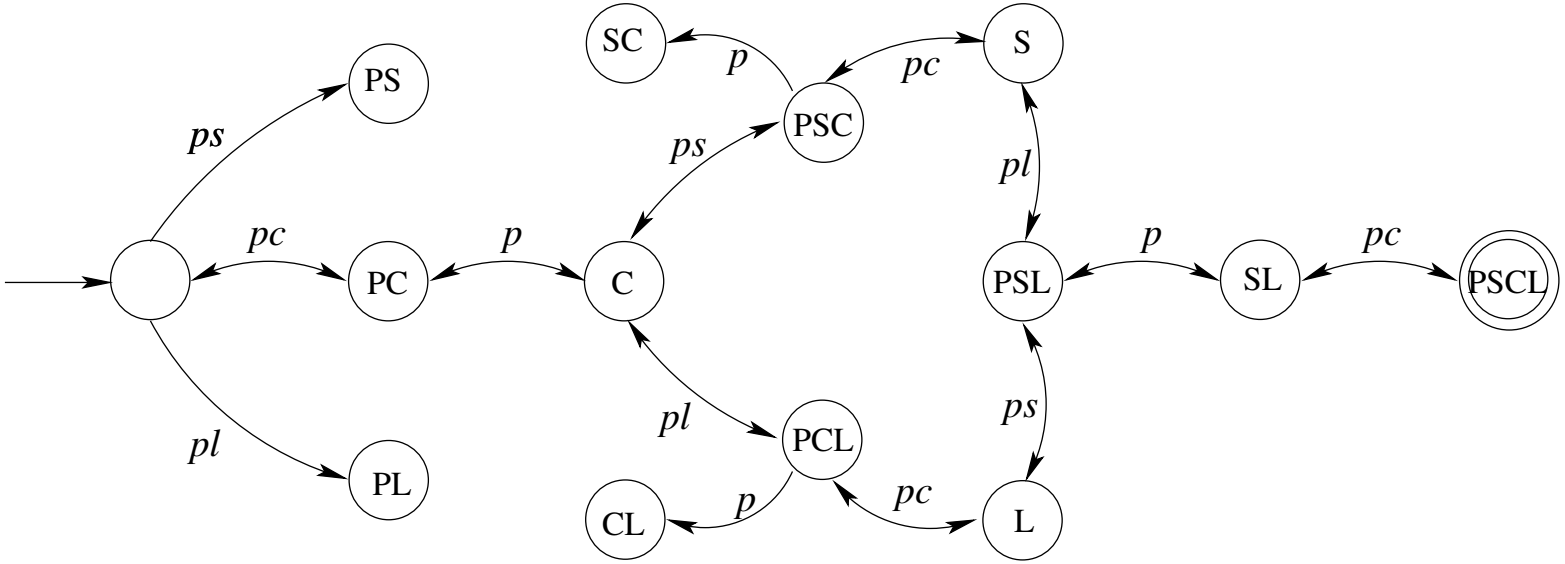
Un passeur doit faire traverser un fleuve à un loup, une chèvre et une salade.

- Il ne peut prendre qu'un seul objet à la fois avec lui dans sa barque
- Le loup va manger la chèvre s'il les laisse seuls sur une rive
- La chèvre va manger la salade s'il les laisse seules sur une rive

L'énigme du passeur

Un passeur doit faire traverser un fleuve à un loup, une chèvre et une salade.

- Il ne peut prendre qu'un seul objet à la fois avec lui dans sa barque
- Le loup va manger la chèvre s'il les laisse seuls sur une rive
- La chèvre va manger la salade s'il les laisse seules sur une rive

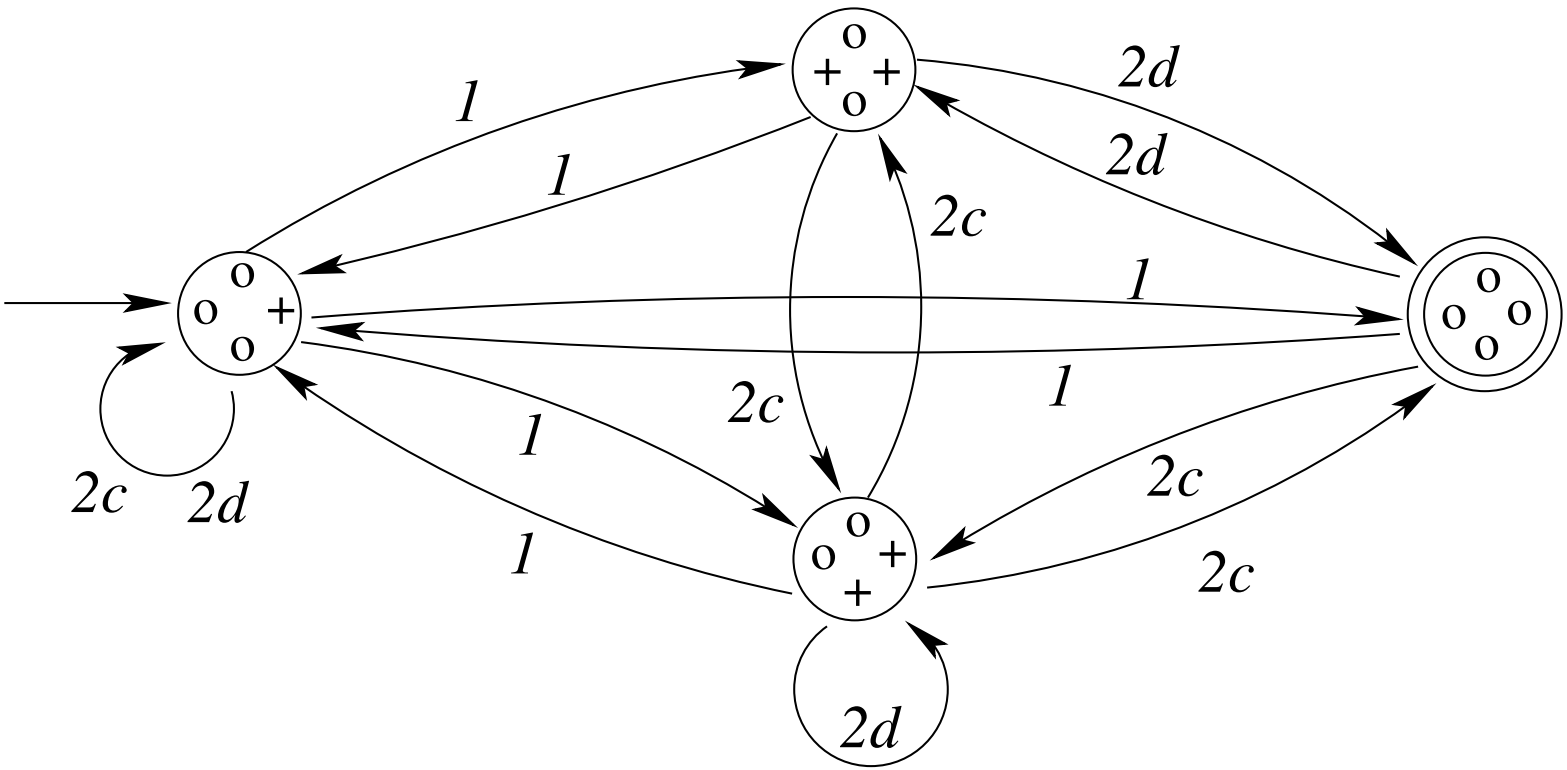


Le barman aveugle avec des gants de boxe

- C'est toute une histoire

Le barman aveugle avec des gants de boxe

- C'est toute une histoire

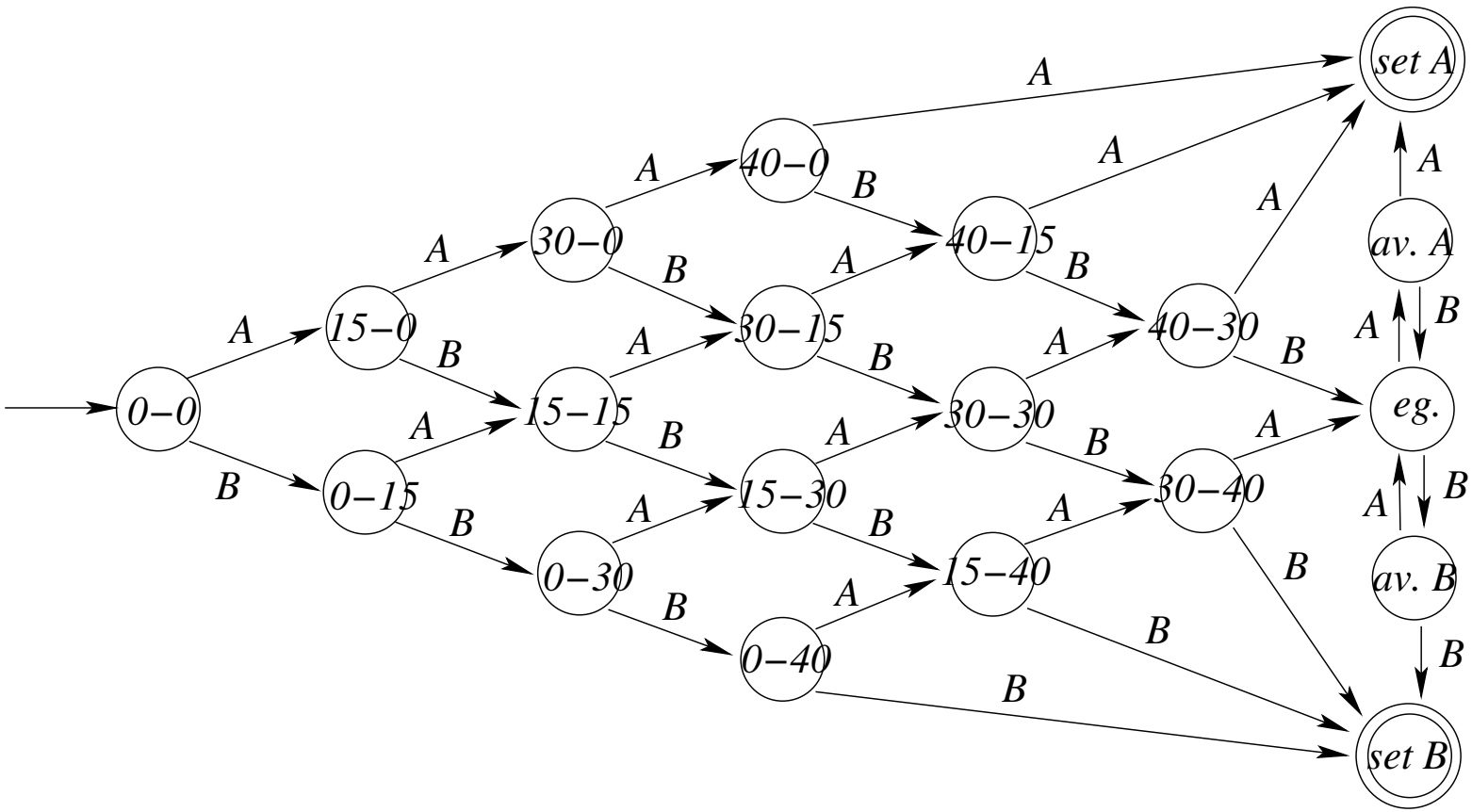


Score au tennis

- Évolution du score pendant un set

Score au tennis

- Évolution du score pendant un set

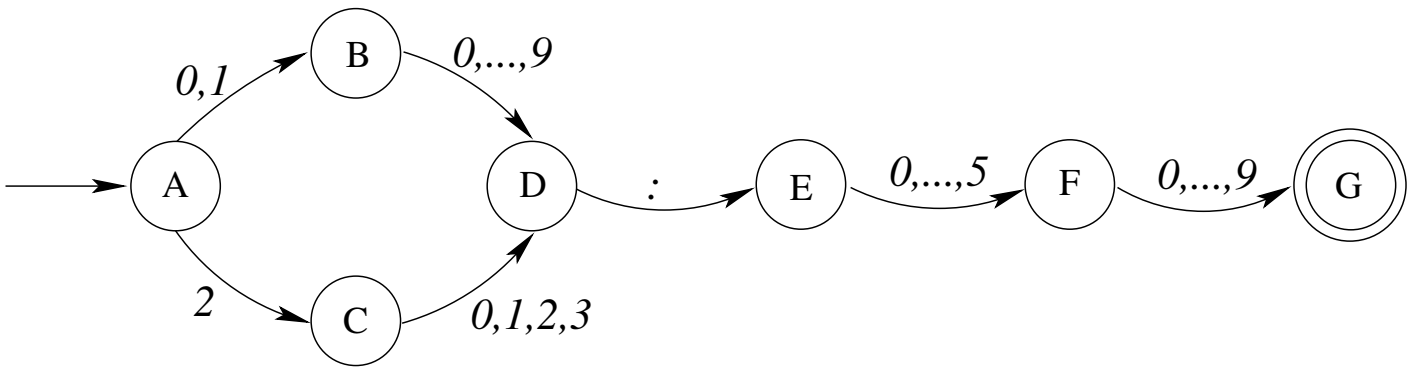


Lire l'heure

- Reconnaître les chaînes représentant des heures valides comme 08 : 46 ou 17 : 15

Lire l'heure

- Reconnaître les chaînes représentant des heures valides comme 08 : 46 ou 17 : 15

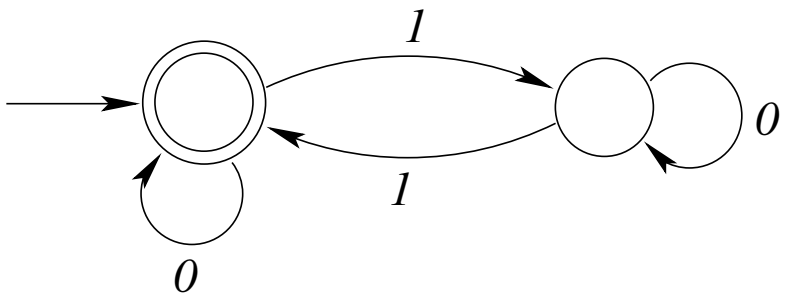


Code de parité

- On désire construire un automate fini permettant de tester la conformité des messages reçus à un format prédéfini.
- Les messages sont des mots sur l'alphabet $\{0, 1\}$, suivis d'un bit supplémentaire appelé bit de parité.
- La longueur des messages n'est pas fixée à l'avance.
- Le bit de parité crée une redondance dans l'information transmise, susceptible de révéler si le message a été altéré. Par définition, le bit de parité est le nombre de 1 que contient la partie de message qui le précède, calculé modulo 2.
- Par exemple : le message 101011 est interprété comme étant le mot 10101 suivi du bit de parité 1 ; le message 0 est le mot vide suivi du bit de parité 0.

Code de parité

- On désire construire un automate fini permettant de tester la conformité des messages reçus à un format prédéfini.
- Les messages sont des mots sur l'alphabet $\{0, 1\}$, suivis d'un bit supplémentaire appelé bit de parité.
- La longueur des messages n'est pas fixée à l'avance.
- Le bit de parité crée une redondance dans l'information transmise, susceptible de révéler si le message a été altéré. Par définition, le bit de parité est le nombre de 1 que contient la partie de message qui le précède, calculé modulo 2.
- Par exemple : le message 101011 est interprété comme étant le mot 10101 suivi du bit de parité 1 ; le message 0 est le mot vide suivi du bit de parité 0.

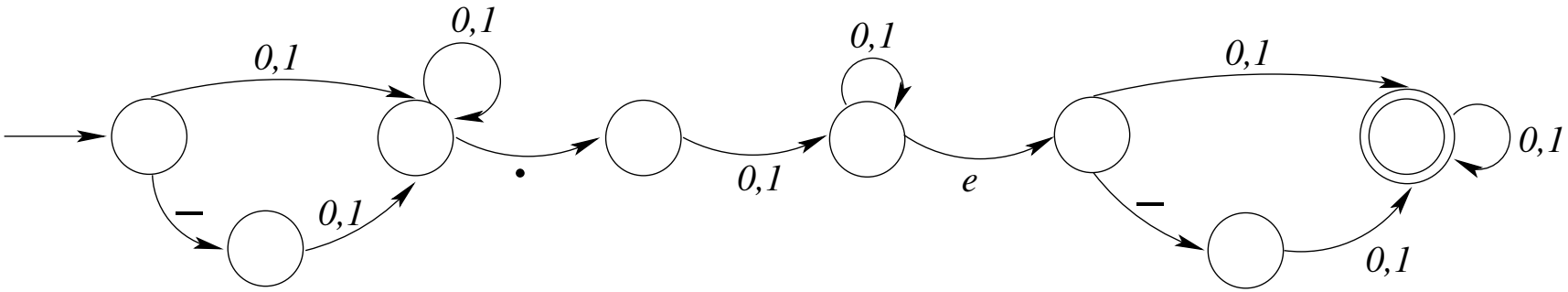


Nombres flottants binaires

- Reconnaître les nombres écrits en binaire en virgule flottante avec un exposant, comme par exemple : $1.001e-101$ ou $-001.0e10$.
  Attention : les mots sont lus de gauche à droite, le point décimal et l'exposant sont obligatoires, et on ne s'intéresse pas au nombre de bits des mots.

Nombres flottants binaires

- Reconnaître les nombres écrits en binaire en virgule flottante avec un exposant, comme par exemple : $1.001e-101$ ou $-001.0e10$.
  Attention : les mots sont lus de gauche à droite, le point décimal et l'exposant sont obligatoires, et on ne s'intéresse pas au nombre de bits des mots.

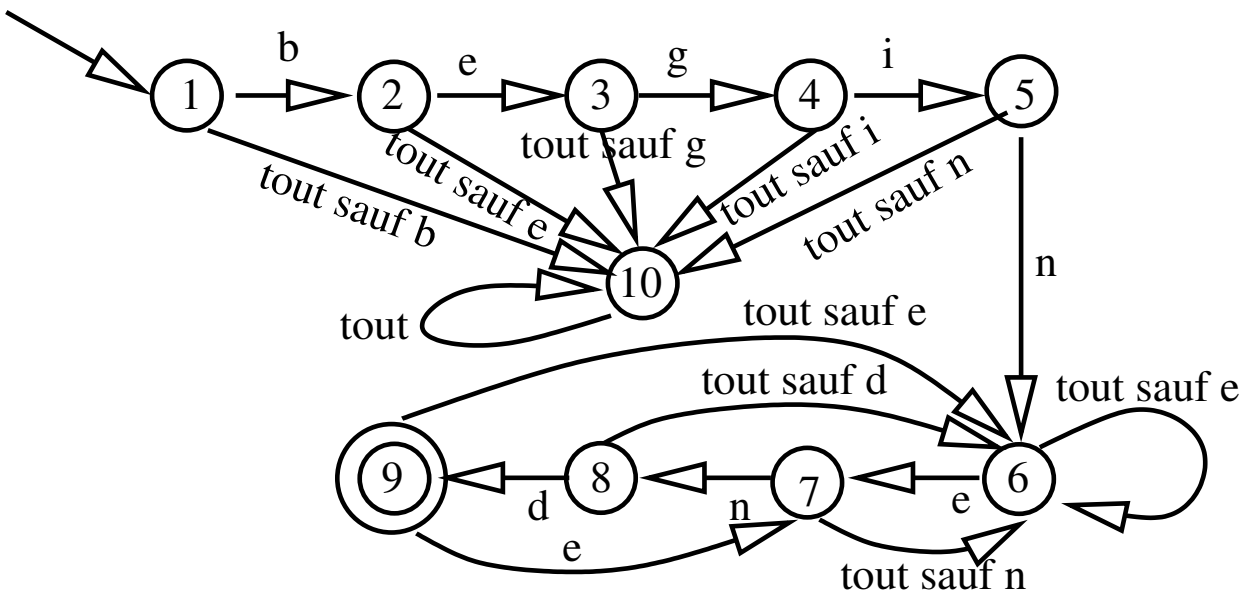


Langages de progammation

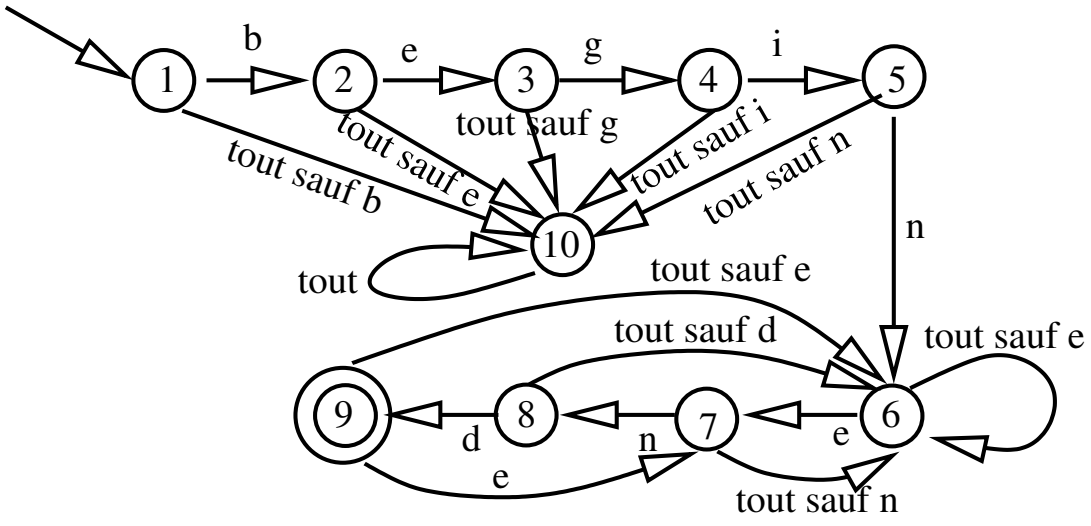
- Contraintes syntaxiques
- Par exemple : Tous les programmes doivent commencer par `begin` et finir par `end`

Langages de progammation

- Contraintes syntaxiques
- Par exemple : Tous les programmes doivent commencer par `begin` et finir par `end`

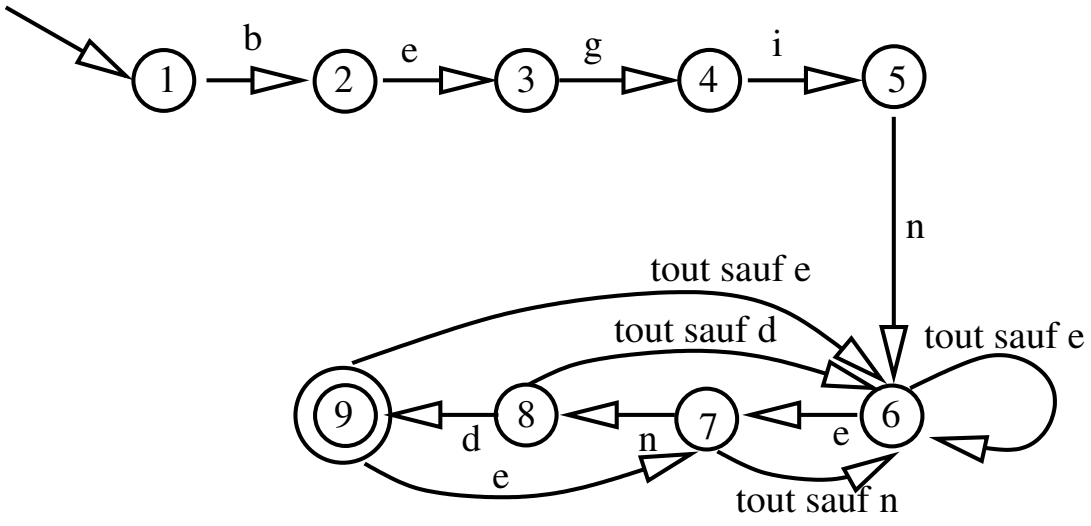


État de rebut



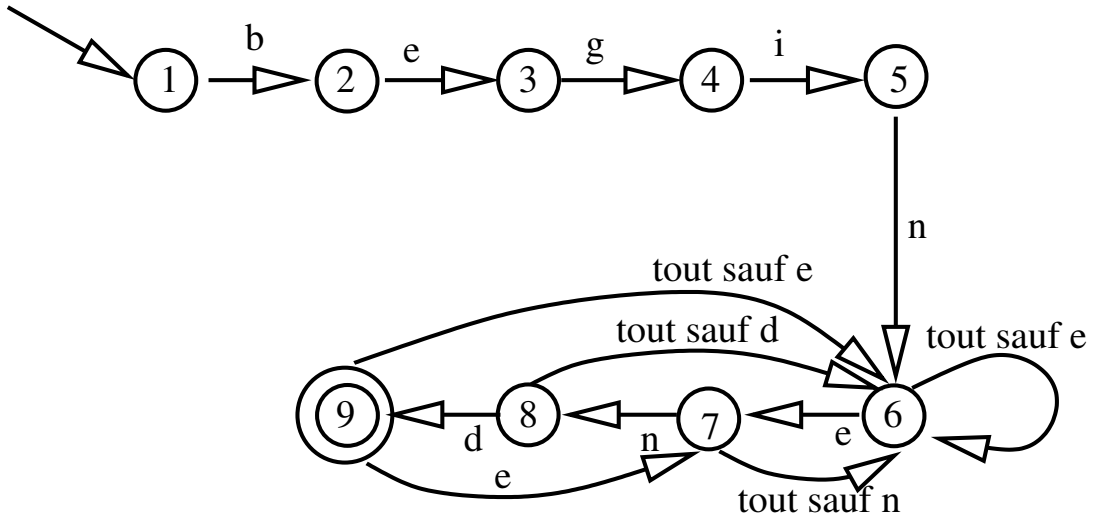
- Rebut pas obligatoire
- Si pas de rebut, le calcul peut se bloquer avant la fin
  - Le mot est alors REFUSÉ
- Si pas de rebut, la table de transition peut contenir des cases vides

État de rebut



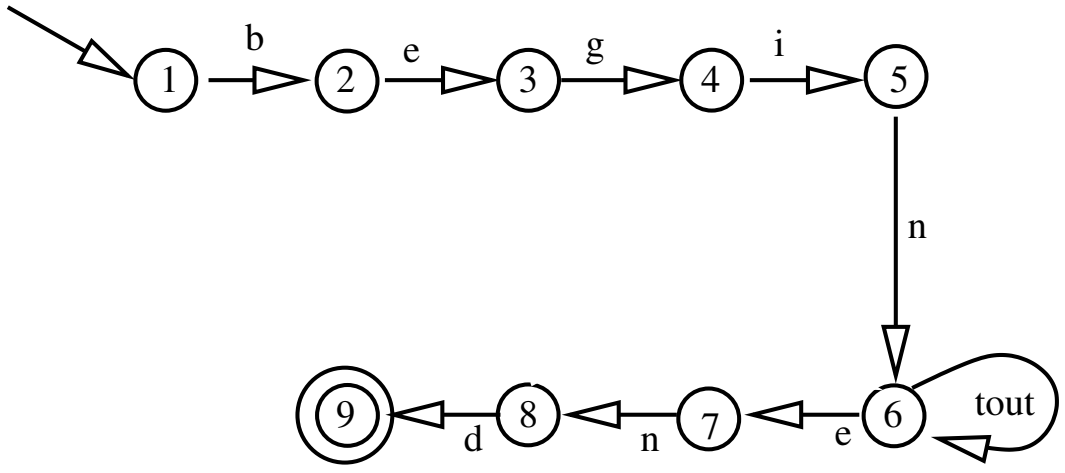
- Rebut pas obligatoire
- Si pas de rebut, le calcul peut se bloquer avant la fin
  - Le mot est alors REFUSÉ
- Si pas de rebut, la table de transition peut contenir des cases vides

Non-déterminisme



- Plusieurs flèches étiquetées par la même lettre partent d'un même état
- Plusieurs calculs possibles pour un même mot
  - Un mot est accepté quand il est accepté par AU MOINS UN CALCUL
- La table de transition peut contenir plusieurs états par case

Non-déterminisme



- Plusieurs flèches étiquetées par la même lettre partent d'un même état
- Plusieurs calculs possibles pour un même mot
  - Un mot est accepté quand il est accepté par AU MOINS UN CALCUL
- La table de transition peut contenir plusieurs états par case

Langages de progammation

- Contraintes syntaxiques
- Alphabet
  $\Sigma_{if} := \{r, w, i, n, a, A, B, C, \cdots, X, Y, Z, <, =, :, ;, \sqcup, \hookleftarrow\}$.

Langages de progammation

- Contraintes syntaxiques
- Alphabet
  $\Sigma_{if} := \{r, w, i, n, a, A, B, C, \cdots, X, Y, Z, <, =, :, ;, \sqcup, \hookleftarrow\}$.
- Compilation

r	$\rightarrow$	read	$<=$	$\rightarrow$	less or equal
w	$\rightarrow$	write	$:=$	$\rightarrow$	variable assignment
i	$\rightarrow$	if	$;$	$\rightarrow$	sequentiation
n	$\rightarrow$	not	$\sqcup$	$\rightarrow$	blank space
a	$\rightarrow$	and	$\hookleftarrow$	$\rightarrow$	carriage return

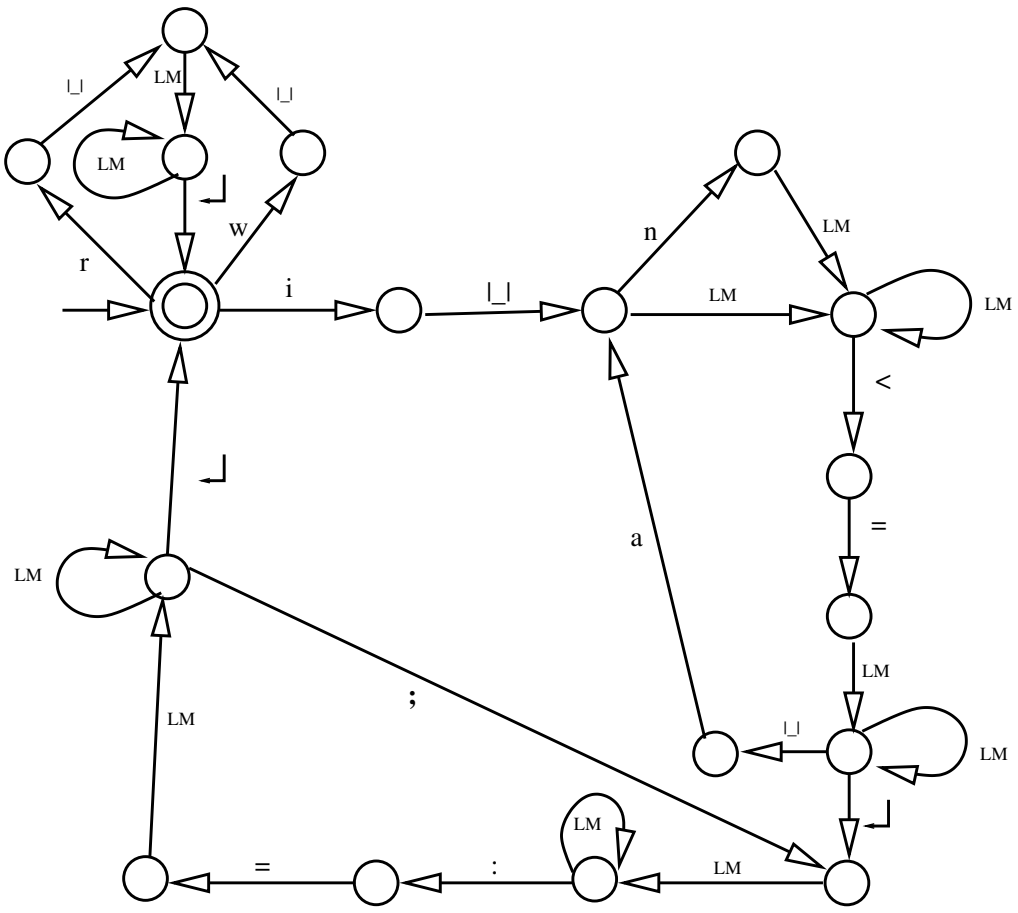
Langages de progammation

- Contraintes syntaxiques
- Alphabet
  $\Sigma_{if} := \{r, w, i, n, a, A, B, C, \cdots, X, Y, Z, <, =, :, ;, \sqcup, \hookleftarrow\}$.

- Exemple de programme

$r \sqcup A \hookleftarrow r \sqcup B \hookleftarrow i \sqcup A <= B \hookleftarrow C := A; D := B \hookleftarrow i \sqcup n \sqcup A <= B \hookleftarrow$
$C := B; D := A \hookleftarrow$

Langages de progammation



Langages de progammation

- Contraintes syntaxiques
- Nombreuses
- Parfois compliquées

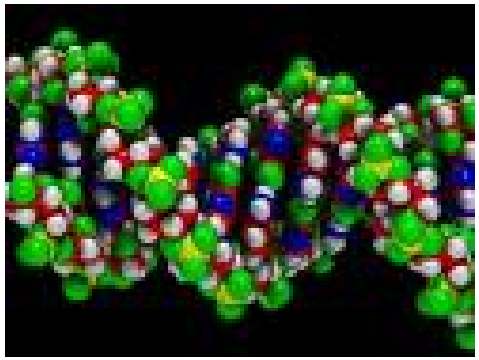
- Compilation
- Analyse syntaxique à l'aide d'automates finis
- Mais pas seulement (parenthésage)

Exercice (Correction syntaxique de programmes)

1. Écrire avec l'aide du langage Σ_{if} un programme qui trie trois nombres entiers.
2. Vérifier la correction syntaxique de ce programme avec l'aide de l'automate de ce langage.

ADN

- Quatre bases azotées
  - Adénine (A)
  - Thymine (T)
  - Cytosine (C)
  - Guanine (G)



ADN

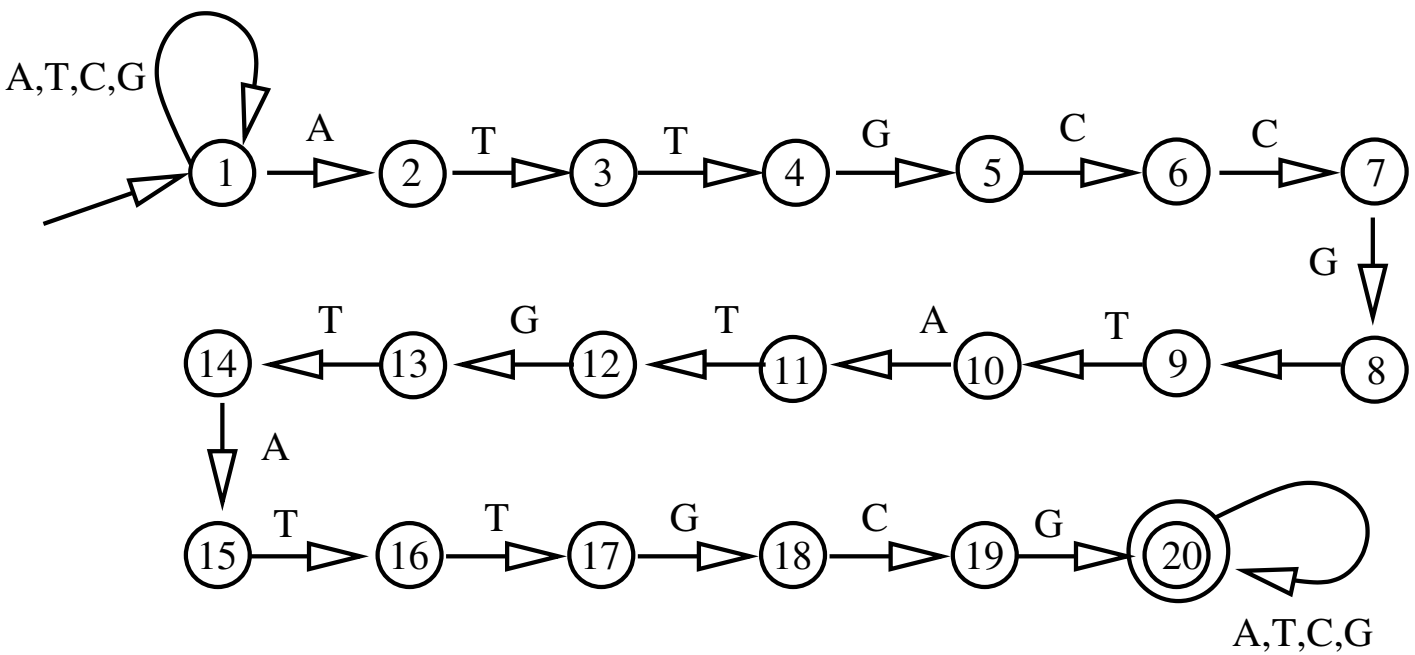
- Rechercher dans un fragment de gène la suite

ATTGCCGTATGTATTGCG

ADN

- Rechercher dans un fragment de gène la suite

ATTGCCGTATGTATTGCG



Problème

- La description d'un langage en Français est souvent ambigüe
- On a besoin d'une façon plus mathématique de décrire les langages des automates, appelée « les expressions régulières »
- On utilise pour cela trois opérations : la concaténation, la réunion et l'étoile

Expressions régulières

- La **concaténation** $L_1 L_2$: un mot du langage L_1 suivi d'un mot du langage L_2

Exemple

$$L_1 = \{aa, bb\} \text{ et } L_2 = \{ab, bba, b\}$$

$$\Downarrow$$

$$L_1 L_2 = \{aaab, aabba, aab, bbab, bbbba, bbb\}$$

Expressions régulières

- La **réunion** $L_1 + L_2$: un mot du langage L_1 ou bien du langage L_2

Exemple

$$L_1 = \{aa, bb\} \text{ et } L_2 = \{ab, bba, b\}$$
$$\Downarrow$$
$$L_1 + L_2 = \{aa, bb, ab, bba, b\}$$

Expressions régulières

- L'**étoile** L^* : concaténation d'un nombre quelconque (peut-être nul) de mots du langage L

Exemple

$$L = \{aa, bb\}$$
$$\Downarrow$$
$$L^* =$$
$$\{\varepsilon, aa, bb, aaaa, aabb, bbaa, bbbb, aaaaaa, aaaabb, aabbaa, \ldots\}$$

Expressions régulières

Remarques

- Langage composé d'un seul mot : $\{abba\}$ est noté $abba$
- Langage fini : $\{ab, bba, b\}$ est noté $ab + bba + b$
- Langage composé de tous les mots : Σ^* est aussi noté $(a + b)^*$

Expression régulière représentant un langage donné

- Sur l'alphabet $\{a, b\}$, le langage constitué des mots qui contiennent aa ou bb

Expression régulière représentant un langage donné

- Sur l'alphabet $\{a, b\}$, le langage constitué des mots qui contiennent aa ou bb

$$(a+b)^*(aa+bb)(a+b)^*$$

Expression régulière représentant un langage donné

- Sur l'alphabet $\{a, b\}$, le langage constitué des mots qui contiennent aa ou bb

$$(a+b)^*(aa+bb)(a+b)^*$$

ou

$$(a+b)^*aa(a+b)^* + (a+b)^*bb(a+b)^*$$

Expression régulière représentant un langage donné

- Sur l'alphabet $\{a, b\}$, le langage constitué des mots qui commencent par aa et finissent par bb

Expression régulière représentant un langage donné

- Sur l'alphabet $\{a, b\}$, le langage constitué des mots qui commencent par aa et finissent par bb

$$aa(a+b)^*bb$$

Expression régulière représentant un langage donné

- Sur l'alphabet $\{a, b, c\}$, le langage constitué des mots qui commencent par aa et finissent par bb

Expression régulière représentant un langage donné

- Sur l'alphabet $\{a, b, c\}$, le langage constitué des mots qui commencent par aa et finissent par bb

$$aa(a + b + c)^* bb$$

Expression régulière représentant un langage donné

- Sur l'alphabet $\{a, b\}$, le langage constitué des mots qui commencent par aa ou finissent par bb

Expression régulière représentant un langage donné

- Sur l'alphabet $\{a, b\}$, le langage constitué des mots qui commencent par aa ou finissent par bb

$$aa(a+b)^* + (a+b)^*bb$$

Expression régulière représentant un langage donné

- Sur l'alphabet $\{a, b\}$, le langage constitué des mots qui ne contiennent pas aa

Expression régulière représentant un langage donné

- Sur l'alphabet $\{a, b\}$, le langage constitué des mots qui ne contiennent pas aa

$$(ab + b)^*(a + \varepsilon)$$

Langage représenté par une expression régulière donnée

$$(a+b+c)^*aa(a+b+c)^*bb(a+b+c)^*$$

Langage représenté par une expression régulière donnée

$$(a+b+c)^*aa(a+b+c)^*bb(a+b+c)^*$$

- Sur l'alphabet $\{a, b, c\}$, cette expression régulière représente le langage constitué des mots qui contiennent aa puis bb

Langage représenté par une expression régulière donnée

$$(b + ab^*a)^*$$

Langage représenté par une expression régulière donnée

$$(b + ab^*a)^*$$

- Sur l'alphabet $\{a, b\}$, cette expression régulière représente le langage constitué des mots qui contiennent un nombre pair de a

Langage représenté par une expression régulière donnée

$$b^*a(b + ab^*a)^*ab^*$$

Langage représenté par une expression régulière donnée

$$b^* a (b + ab^* a)^* ab^*$$

- Sur l'alphabet $\{a, b\}$, cette expression régulière représente le langage constitué des mots qui contiennent un nombre pair de a et au moins deux a

Langage représenté par une expression régulière donnée

$$(a+b)^* \left(aa(a+b)^*bb + bb(a+b)^*aa\right)(a+b)^*$$

Langage représenté par une expression régulière donnée

$$(a+b)^* \left(aa(a+b)^*bb + bb(a+b)^*aa\right)(a+b)^*$$

- Sur l'alphabet $\{a, b\}$, cette expression régulière représente le langage constitué des mots qui contiennent aa et bb dans n'importe quel ordre

Additions

On s'intéresse aux additions en binaire, comme par exemple :

$$\begin{array}{rccccc} & 0 & 1 & 1 & 0 & 1 \\ + & 0 & 0 & 1 & 1 & 1 \\ \hline & 1 & 0 & 1 & 0 & 0 \end{array}$$

- On considère une telle addition (lue de droite à gauche) comme un mot sur l'alphabet constitué par l'ensembles des colonnes de 3 bits, c'est-à-dire

$$\Sigma = \left\{ \begin{array}{ccccccc} 0 & & 0 & & 1 & & 1 \\ 0 & , & 1 & , & 0 & , \ldots , & 1 \\ 0 & & 1 & & 0 & & 1 \end{array} \right\}$$

- Il y a donc 8 « lettres » dans l'alphabet.

Additions

On s'intéresse aux additions en binaire, comme par exemple :

$$\begin{array}{rccccc} & 0 & 1 & 1 & 0 & 1 \\ + & 0 & 0 & 1 & 1 & 1 \\ \hline & 1 & 0 & 1 & 0 & 0 \end{array}$$

- Certains mots sur l'alphabet Σ représentent des additions justes (comme ci-dessus), et d'autres des additions fausses, comme par exemple :

$$\begin{array}{rcccccc} & 0 & 1 & 0 & 1 & 0 & 1 \\ + & 0 & 0 & 0 & 1 & 1 & 1 \\ \hline & 1 & 0 & 0 & 1 & 0 & 0 \end{array}$$

Additions

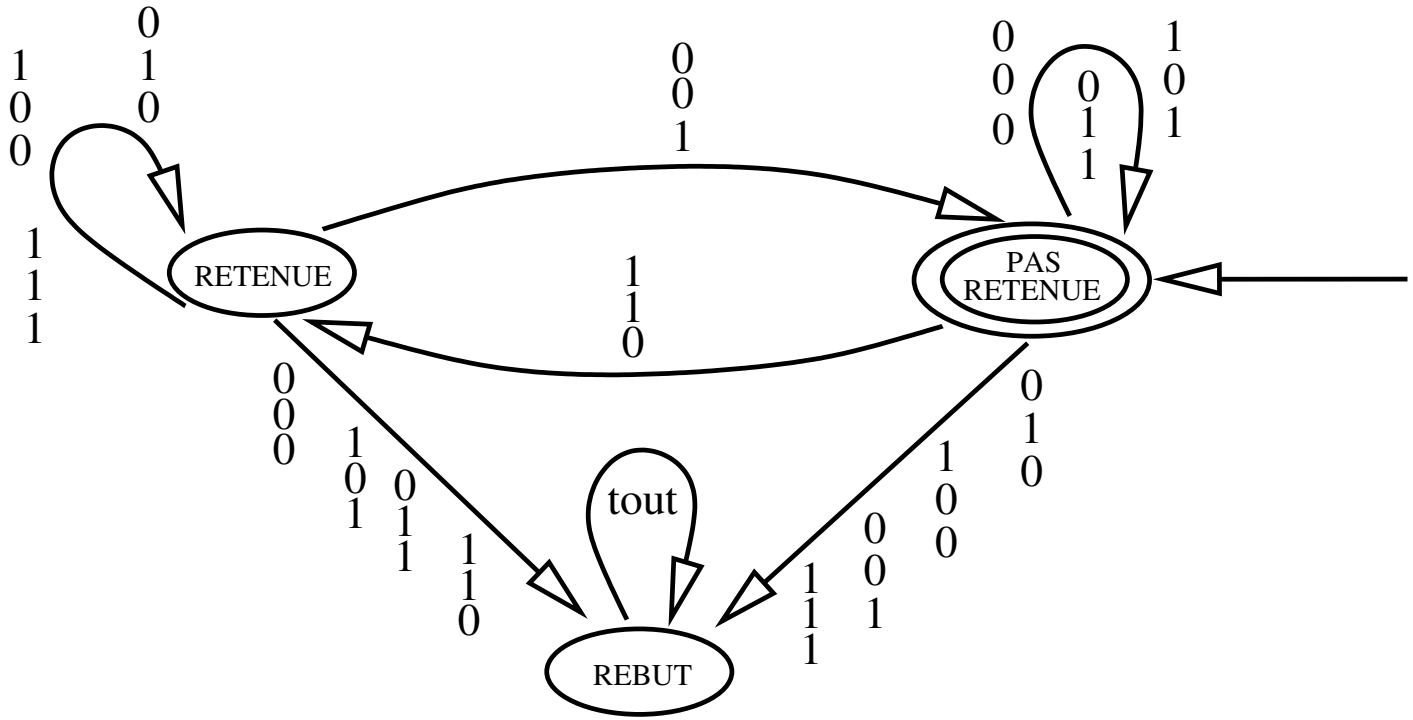
$$\begin{array}{rccccc} & 0 & 1 & 1 & 0 & 1 \\ + & 0 & 0 & 1 & 1 & 1 \\ \hline & 1 & 0 & 1 & 0 & 0 \end{array}$$

$$\begin{array}{rcccccc} & 0 & 1 & 0 & 1 & 0 & 1 \\ + & 0 & 0 & 0 & 1 & 1 & 1 \\ \hline & 1 & 0 & 0 & 1 & 0 & 0 \end{array}$$

Additions

$$\begin{array}{cccccc} & 0 & 1 & 1 & 0 & 1 \\ + & 0 & 0 & 1 & 1 & 1 \\ \hline & 1 & 0 & 1 & 0 & 0 \end{array}$$

$$\begin{array}{cccccc} & 0 & 1 & 0 & 1 & 0 & 1 \\ + & 0 & 0 & 0 & 1 & 1 & 1 \\ \hline & 1 & 0 & 0 & 1 & 0 & 0 \end{array}$$



Objectif

Question

Étant donné un langage L, existe-t-il un automate fini qui reconnaît ce langage ?

Objectif

Question

Étant donné un langage L, existe-t-il un automate fini qui reconnaît ce langage ?

- Si le langage L est donné par une **expression régulière**, on sait que la réponse est **OUI**, et on sait même **construire** un automate fini qui reconnaît ce langage.
- Mais si la réponse est **NON**, autrement dit s'il n'existe **aucun** automate fini qui reconnaît le langage L, comment le **prouver** ?

Objectif

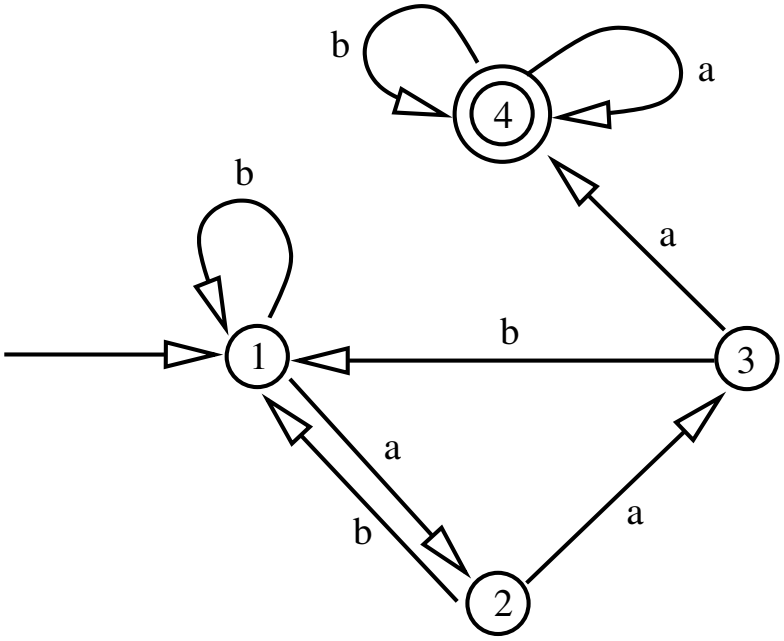
Question

Étant donné un langage L, existe-t-il un automate fini qui reconnaît ce langage ?

Outil pour le NON

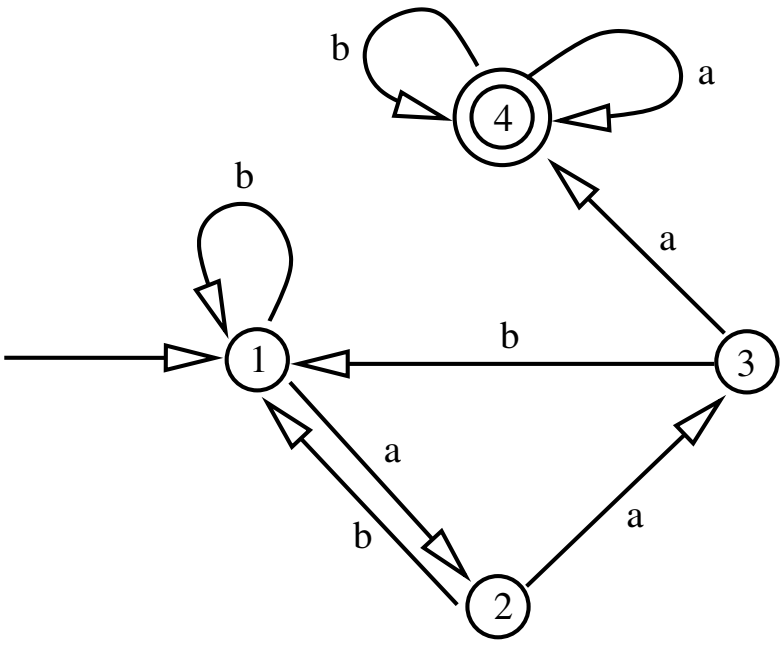
- On va voir une propriété (appelée le **lemme de la pompe**) qui est vraie pour **tous** les langages réguliers
- Par conséquent, si un langage **ne vérifie pas** cette propriété, il est **impossible** qu'il soit un langage régulier

Ce que pomper veut dire



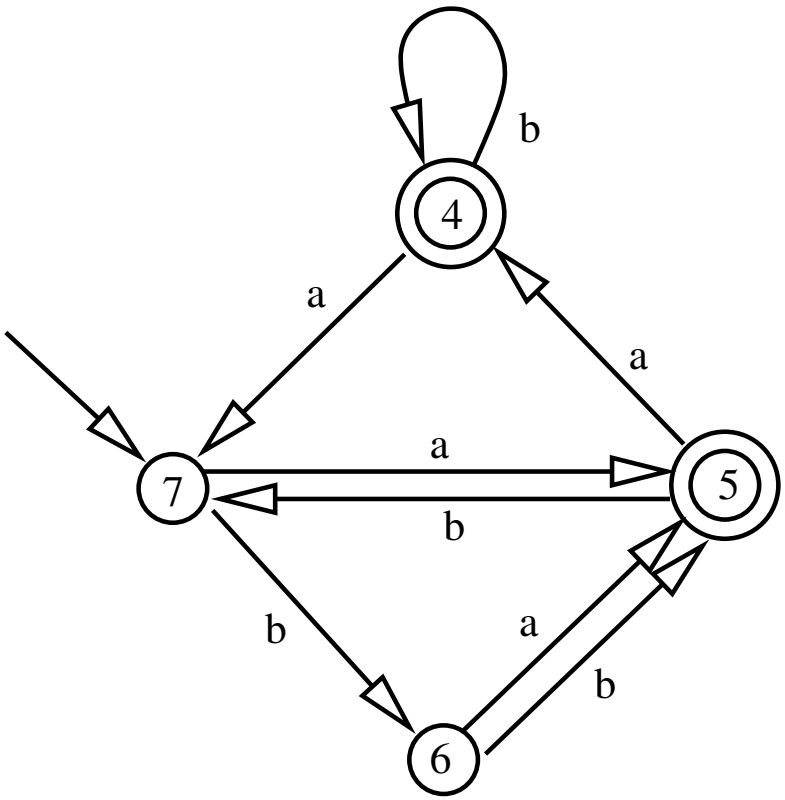
- *aab*baaaba accepté
- *aab*aab*baaaba* accepté
- *aab*aab*aab*aab*baaaba* accepté
- N'importe quel mot de la forme $(aab \ldots aab)baaaba$ accepté
- Car il y a un cycle $1 \xrightarrow{a} 2 \xrightarrow{a} 3 \xrightarrow{b} 1$
- On dit qu'on peut **pomper** *aab* dans le mot *aab*baaaba

Pomper ici ou pomper ailleurs



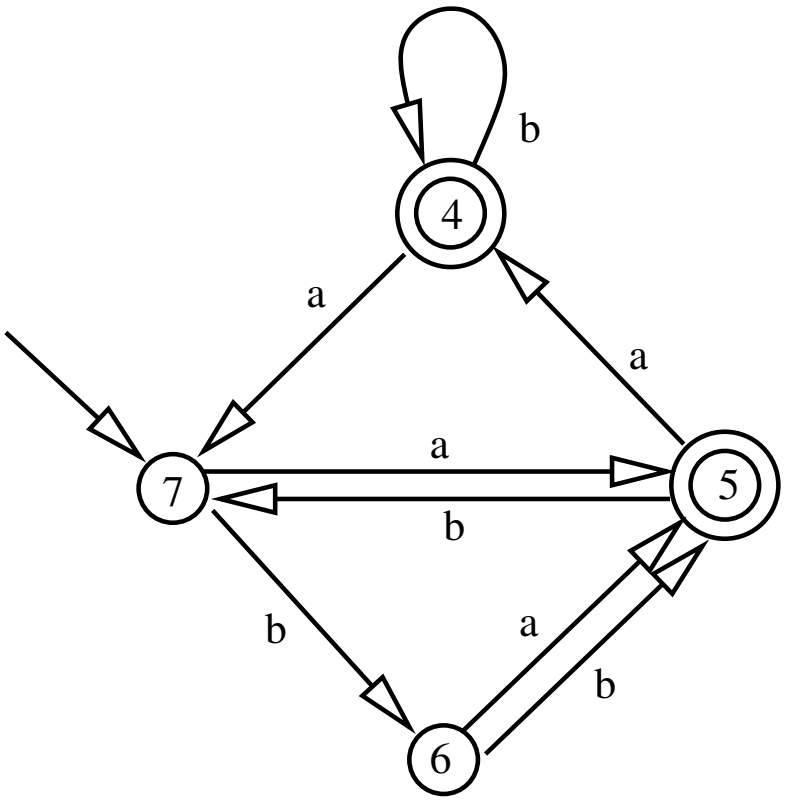
- *aabbaaaba* accepté
- *aabbbaaaba* accepté
- *aabbbbbbaaaba* accepté
- N'importe quel mot de la forme $aab(b \ldots b)aaaba$ accepté
- Car il y a un cycle $1 \xrightarrow{b} 1$
- On peut aussi **pomper** *b* dans le mot *aabbaaaba*

Peut-on toujours pomper ?



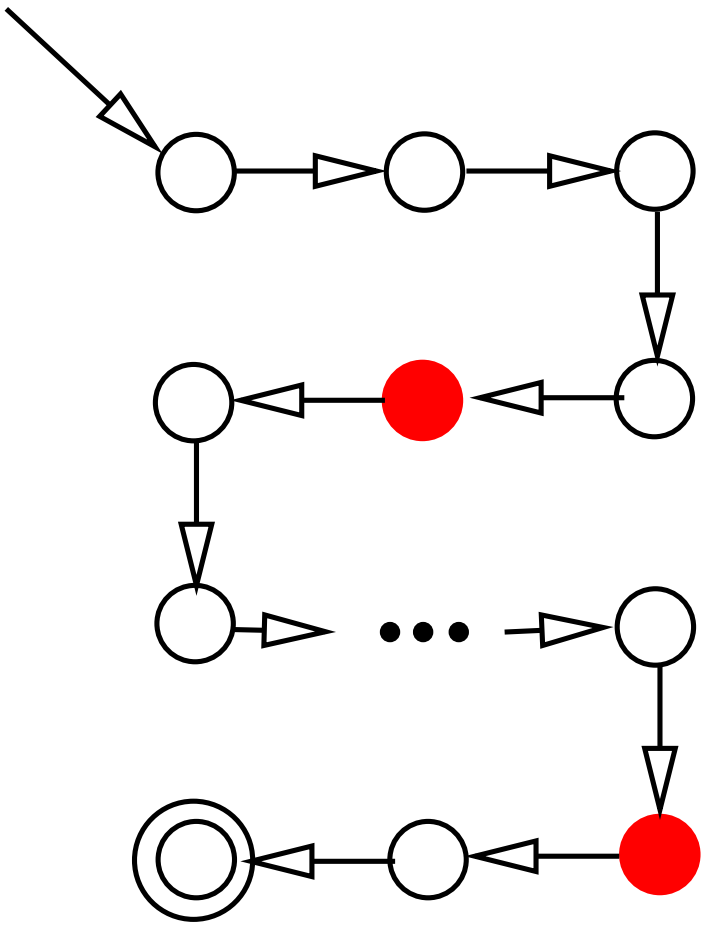
- *bbbaabaa* accepté
- Peut-on pomper quelque chose quelque part ?
- *bbbaabaa*
  (cycle $5 \xrightarrow{b} 7 \xrightarrow{a} 5$)
- ou *bbbaabaa*
  (cycle
  $5 \xrightarrow{a} 4 \xrightarrow{b} 4 \xrightarrow{a} 7 \xrightarrow{a} 5$)

Peut-on toujours pomper ?



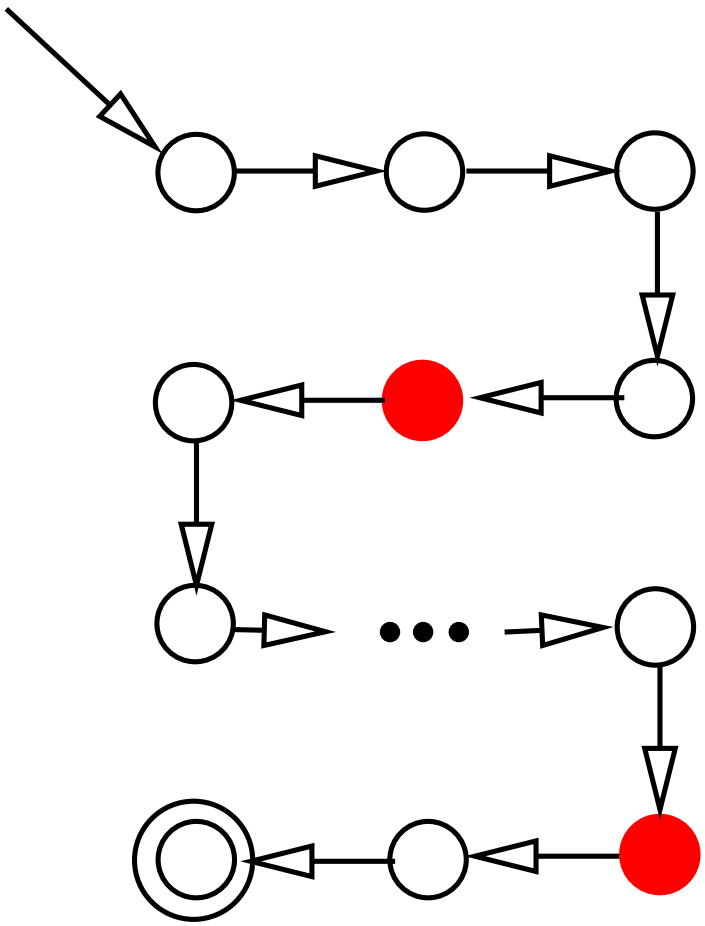
- *bb* accepté
- Peut-on pomper quelque chose quelque part ?
- **Non** : aucune partie de *bb* ne correspond à un cycle sur l'automate

Dans quels mots peut-on pomper ?



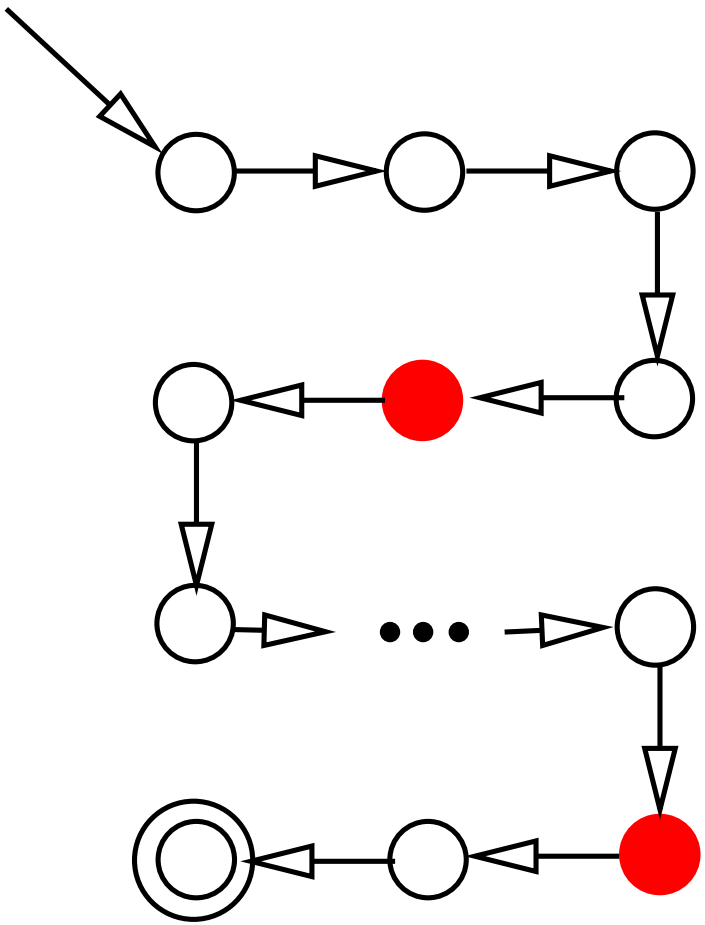
- On peut pomper chaque fois qu'une partie d'un mot accepté correspond à un **cycle** sur l'automate

Dans quels mots peut-on pomper ?



- Dès que la **longueur** d'un mot accepté est **plus grande que le nombre d'états** de l'automate, on doit forcément repasser par (au moins) un état en lisant ce mot
  $\Longrightarrow$ il y a un cycle

Dans quels mots peut-on pomper ?



- Finalement, dès que la longueur d'un mot accepté est plus grande que le nombre d'états de l'automate, on est sûr de pouvoir **pomper** une partie du mot

Lemme de la pompe

Théorème

Soit L un langage reconnu par un automate $\mathcal{A}$ avec n états. Alors on peut pomper quelque chose dans tout mot de L de longueur $\geqslant n$.

Autrement dit :
Tout mot w de L de longueur $\geqslant n$ se décompose sous la forme $w = uxv$ de telle sorte que :

1. $|ux| \leqslant n$
2. $x \neq \varepsilon$
3. Tout mot de la forme $u(x \ldots x)v$ appartient à L

Utilisation du lemme de la pompe

$L = \{a^k b^k, k \geqslant 0\} = \{\varepsilon, ab, aabb, aaabbb, \ldots\}$

On va montrer que L n'est pas régulier par un **raisonnement par l'absurde**.

- **Supposons** que L soit reconnu par un automate $\mathcal{A}$ et appelons n le nombre d'états de $\mathcal{A}$.
- Le mot $w = a^n b^n \in L$ avec $|w| = 2n \geqslant n \Longrightarrow$ **pomper**
- C.-à-d. $w = uxv$ avec $|ux| \leqslant n$, $x \neq \varepsilon$ et $u(x \ldots x)v \in L$
- Donc $x = a^d$ avec $d > 0$
- Par exemple, $w' = uxxv$ **appartient à** L
- Mais $w' = a^{n+d} b^n$, donc w' **n'appartient pas à** L
- **Contradiction**
- Donc L n'est pas reconnu par $\mathcal{A}$

Conclusion : Il n'existe aucun automate fini qui reconnaisse le langage L, c.-à-d. L **n'est pas un langage régulier**

Utilisation du lemme de la pompe

$L = \{a^k b^k, k \geqslant 0\} = \{\varepsilon, ab, aabb, aaabbb, \ldots\}$

On va montrer que L n'est pas régulier par un **raisonnement par l'absurde**.

- **Supposons** que L soit reconnu par un automate $\mathcal{A}$ et appelons n le nombre d'états de $\mathcal{A}$.
- Le mot $w = a^n b^n \in L$ avec $|w| = 2n \geqslant n \Longrightarrow$ **pomper**
- C.-à-d. $w = uxv$ avec $|ux| \leqslant n$, $x \neq \varepsilon$ et $u(x \ldots x)v \in L$
- Donc $x = a^d$ avec $d > 0$
- Par exemple, $w' = uxxv$ **appartient à** L
- Mais $w' = a^{n+d} b^n$, donc w' **n'appartient pas à** L
- **Contradiction**
- Donc L n'est pas reconnu par $\mathcal{A}$

Conclusion : Il n'existe aucun automate fini qui reconnaisse le langage L, c.-à-d. L **n'est pas un langage régulier**

Utilisation du lemme de la pompe

$L = \{a^k b^k, k \geqslant 0\} = \{\varepsilon, ab, aabb, aaabbb, \ldots\}$

On va montrer que L n'est pas régulier par un **raisonnement par l'absurde**.

- **Supposons** que L soit reconnu par un automate $\mathcal{A}$ et appelons n le nombre d'états de $\mathcal{A}$.
- Le mot $w = a^n b^n \in L$ avec $|w| = 2n \geqslant n \Longrightarrow$ **pomper**
- C.-à-d. $w = uxv$ avec $|ux| \leqslant n$, $x \neq \varepsilon$ et $u(x \ldots x)v \in L$
- Donc $x = a^d$ avec $d > 0$
- Par exemple, $w' = uxxv$ **appartient à** L
- Mais $w' = a^{n+d} b^n$, donc w' **n'appartient pas à** L
- **Contradiction**
- Donc L n'est pas reconnu par $\mathcal{A}$

Conclusion : Il n'existe aucun automate fini qui reconnaisse le langage L, c.-à-d. L **n'est pas un langage régulier**

Utilisation du lemme de la pompe

$L = \{a^k b^k, k \geqslant 0\} = \{\varepsilon, ab, aabb, aaabbb, \ldots\}$

On va montrer que L n'est pas régulier par un **raisonnement par l'absurde**.

- **Supposons** que L soit reconnu par un automate $\mathcal{A}$ et appelons n le nombre d'états de $\mathcal{A}$.
- Le mot $w = a^n b^n \in L$ avec $|w| = 2n \geqslant n \Longrightarrow$ **pomper**
- C.-à-d. $w = uxv$ avec $|ux| \leqslant n$, $x \neq \varepsilon$ et $u(x \ldots x)v \in L$
- Donc $x = a^d$ avec $d > 0$
- Par exemple, $w' = uxxv$ **appartient à** L
- Mais $w' = a^{n+d} b^n$, donc w' **n'appartient pas à** L
- **Contradiction**
- Donc L n'est pas reconnu par $\mathcal{A}$

Conclusion : Il n'existe aucun automate fini qui reconnaisse le langage L, c.-à-d. L **n'est pas un langage régulier**

Utilisation du lemme de la pompe

$L = \{a^k b^k, k \geqslant 0\} = \{\varepsilon, ab, aabb, aaabbb, \ldots\}$

On va montrer que L n'est pas régulier par un **raisonnement par l'absurde**.

- **Supposons** que L soit reconnu par un automate $\mathcal{A}$ et appelons n le nombre d'états de $\mathcal{A}$.
- Le mot $w = a^n b^n \in L$ avec $|w| = 2n \geqslant n \Longrightarrow$ **pomper**
- C.-à-d. $w = uxv$ avec $|ux| \leqslant n$, $x \neq \varepsilon$ et $u(x \ldots x)v \in L$
- Donc $x = a^d$ avec $d > 0$
- Par exemple, $w' = uxxv$ **appartient à** L
- Mais $w' = a^{n+d} b^n$, donc w' **n'appartient pas à** L
- **Contradiction**
- Donc L n'est pas reconnu par $\mathcal{A}$

Conclusion : Il n'existe aucun automate fini qui reconnaisse le langage L, c.-à-d. L **n'est pas un langage régulier**

Utilisation du lemme de la pompe

$L = \{a^k b^k, k \geqslant 0\} = \{\varepsilon, ab, aabb, aaabbb, \ldots\}$

On va montrer que L n'est pas régulier par un **raisonnement par l'absurde**.

- **Supposons** que L soit reconnu par un automate $\mathcal{A}$ et appelons n le nombre d'états de $\mathcal{A}$.
- Le mot $w = a^n b^n \in L$ avec $|w| = 2n \geqslant n \Longrightarrow$ **pomper**
- C.-à-d. $w = uxv$ avec $|ux| \leqslant n$, $x \neq \varepsilon$ et $u(x \ldots x)v \in L$
- Donc $x = a^d$ avec $d > 0$
- Par exemple, $w' = uxxv$ **appartient à** L
- Mais $w' = a^{n+d} b^n$, donc w' **n'appartient pas à** L
- **Contradiction**
- Donc L n'est pas reconnu par $\mathcal{A}$

Conclusion : Il n'existe aucun automate fini qui reconnaisse le langage L, c.-à-d. L **n'est pas un langage régulier**

Utilisation du lemme de la pompe

$L = \{a^k b^k, k \geqslant 0\} = \{\varepsilon, ab, aabb, aaabbb, \ldots\}$

On va montrer que L n'est pas régulier par un **raisonnement par l'absurde**.

- **Supposons** que L soit reconnu par un automate $\mathcal{A}$ et appelons n le nombre d'états de $\mathcal{A}$.
- Le mot $w = a^n b^n \in L$ avec $|w| = 2n \geqslant n \Longrightarrow$ **pomper**
- C.-à-d. $w = uxv$ avec $|ux| \leqslant n$, $x \neq \varepsilon$ et $u(x \ldots x)v \in L$
- Donc $x = a^d$ avec $d > 0$
- Par exemple, $w' = uxxv$ **appartient à** L
- Mais $w' = a^{n+d} b^n$, donc w' **n'appartient pas à** L
- **Contradiction**
- Donc L n'est pas reconnu par $\mathcal{A}$

Conclusion : Il n'existe aucun automate fini qui reconnaisse le langage L, c.-à-d. L **n'est pas un langage régulier**

Utilisation du lemme de la pompe

$L = \{a^k b^k, k \geqslant 0\} = \{\varepsilon, ab, aabb, aaabbb, \ldots\}$

On va montrer que L n'est pas régulier par un **raisonnement par l'absurde**.

- **Supposons** que L soit reconnu par un automate $\mathcal{A}$ et appelons n le nombre d'états de $\mathcal{A}$.
- Le mot $w = a^n b^n \in L$ avec $|w| = 2n \geqslant n \Longrightarrow$ **pomper**
- C.-à-d. $w = uxv$ avec $|ux| \leqslant n$, $x \neq \varepsilon$ et $u(x \ldots x)v \in L$
- Donc $x = a^d$ avec $d > 0$
- Par exemple, $w' = uxxv$ **appartient à** L
- Mais $w' = a^{n+d} b^n$, donc w' **n'appartient pas à** L
- **Contradiction**
- Donc L n'est pas reconnu par $\mathcal{A}$

Conclusion : Il n'existe aucun automate fini qui reconnaisse le langage L, c.-à-d. L **n'est pas un langage régulier**

Utilisation du lemme de la pompe

$L = \{a^k b^k, k \geqslant 0\} = \{\varepsilon, ab, aabb, aaabbb, \ldots\}$

On va montrer que L n'est pas régulier par un **raisonnement par l'absurde**.

- **Supposons** que L soit reconnu par un automate $\mathcal{A}$ et appelons n le nombre d'états de $\mathcal{A}$.
- Le mot $w = a^n b^n \in L$ avec $|w| = 2n \geqslant n \Longrightarrow$ **pomper**
- C.-à-d. $w = uxv$ avec $|ux| \leqslant n$, $x \neq \varepsilon$ et $u(x \ldots x)v \in L$
- Donc $x = a^d$ avec $d > 0$
- Par exemple, $w' = uxxv$ **appartient à** L
- Mais $w' = a^{n+d} b^n$, donc w' **n'appartient pas à** L
- **Contradiction**
- Donc L n'est pas reconnu par $\mathcal{A}$

Conclusion : Il n'existe aucun automate fini qui reconnaisse le langage L, c.-à-d. L **n'est pas un langage régulier**

D'autres langages non réguliers

- Le langage des palindromes
  - Mot qui se lit pareil de droite à gauche et de gauche à droite
  - En français : RADAR, KAYAK, RESSASSER, ...
  - On note u^R le mot u renversé : bba devient abb
  - Tout mot w de la forme uu^R est un palindrome
- Le langage des parenthèses bien formées
  - $(())()(())$ ou $((((()))))$
  - mais pas $())($ ni $((()(()$
- Le langage composé des mots qui contiennent le même nombre de a que de b
- etc.

Une arme absolue ?

- Si on montre qu'un langage **ne satisfait pas** le lemme de la pompe, c'est une **preuve** que ce langage n'est pas régulier
- **MAIS** il peut arriver qu'un langage qui **satisfait** le lemme de la pompe ne soit pas régulier **:-(**
- Par exemple c'est le cas pour $L = \{uu^R v, u \neq \varepsilon, v \neq \varepsilon\}$

Guide d'utilisation

Pour décider si L est régulier ou pas :

- On sait construire un automate fini qui reconnaît L
  - **Conclusion** : L est régulier
- On montre que L ne satisfait pas le lemme de la pompe
  - **Conclusion** : L n'est pas régulier
- On n'arrive pas à construire un automate fini qui accepte L, mais on montre que L satisfait le lemme de la pompe
  - **On ne peut pas conclure**

Vérifier les multiplications ?

Exercice

On considère l'alphabet $\Sigma = \left\{ \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \ldots, \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \right\}$ constitué des 8 colonnes de 3 bits. On s'intéresse au langage $\mathcal{M}$ des mots qui représentent une multiplication juste en binaire,

comme par exemple $\begin{array}{rccccc} & 0 & 0 & 1 & 0 & 0 \\ \times & 0 & 0 & 1 & 0 & 0 \\ \hline & 1 & 0 & 0 & 0 & 0 \end{array}$, soit $4 \times 4 = 16$ en décimal.

Montrons que le langage $\mathcal{M}$ n'est pas un langage régulier.

FIN