INF421-a
Bases de la programmation et de l'algorithmique

(Bloc 9/ 9)

Philippe Baptiste

CNRS LIX, École Polytechnique

20 octobre 2006

Aujourd'hui

Je cherche

- Comment faire pour chercher un mot dans un texte ?
- Comment faire pour chercher une expression complexe ?

```
grep -r 'xy??[0-9][0-9]' fichier.txt
egrep 'abcd|efgh' fichier.txt
```

Symboles, mots, langage

- L'alphabet Σ est un ensemble de caractères (ou symboles).
- Un mot est une suite de caractères.
- L'ensemble des mots sur Σ est noté Σ^*.
- Un *langage* est un sous-ensemble de Σ^*, c'est-à-dire un ensemble particulier de mots.
- Parmi les mots de Σ^* on distingue le mot vide noté ϵ. Le mot vide est l'unique mot de longueur zéro.

Symboles, mots, langage

- La concaténation de deux mots m_1 et m_2 est le mot obtenu en mettant m_1 à la fin de m_2.
- On note $\cdot$ l'opérateur de concaténation (on omet souvent cet opérateur) : $m_1 \cdot m_2$ ou $m_1 m_2$.
- La concaténation est une opération associative qui possède un élément neutre : le mot vide ϵ.
- Dans l'écriture $m = m_1 m_2$, m_1 est le *préfixe* du mot m, tandis que m_2 est le *suffixe* du mot m

Symboles, mots, langage

- La concaténation s'étend aux ensembles de mots, on note $L_1 \cdot L_2$ le langage obtenu en concaténant tous les mots de L_1 avec les mots de L_2.

$$L_1 \cdot L_2 = \{m_1 \cdot m_2 \mid m_1 \in L_1 \wedge m_2 \in L_2\}$$

- On note $L^\star$ le langage obtenu en concaténant les mots de L.
  - $L^0 = \{\epsilon\}$
  - $L^{n+1} = L^n \cdot L$
  - $L^\star = \bigcup_{i \in \mathbb{N}} L^i$

  C'est-à-dire qu'un mot m de L^* est la concaténation de n mots $(n \geq 0)$ $m_1, \ldots m_n$, où les m_i sont tous des mots de L.

Symboles, mots, langage

Plus simple : L'union de deux langages L et M est l'ensemble des mots qui sont dans L ou dans M.

Syntaxe des expressions régulières

Les expressions régulières (ou motifs), notées p, sont définies ainsi :

- Le mot vide ϵ est une expression régulière.
- Un caractère $c \in \Sigma$ est une expression régulière.
- Si p_1 et p_2 sont des expressions régulières, alors l'alternative $p_1 \mid p_2$ est une expression régulière.
- Si p_1 et p_2 sont des expressions régulières, alors la concaténation $p_1 \cdot p_2$ est une expression régulière.
- Si p est une expression régulière, alors la répétition $p*$ est une expression régulière.

!! UN ARBRE !!

Syntaxe des expressions régulières

Il y a deux sortes de feuilles

- mot vide
- caractères

Deux sortes de feuilles nœuds binaires

- concaténation
- alternative

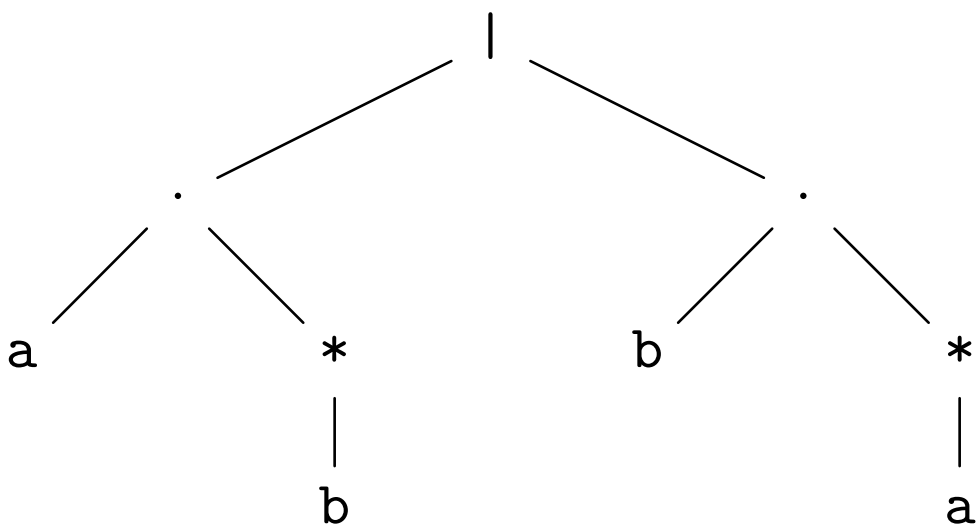
Une sorte de nœud unaire

- la répétition.

Attention, gestion des priorités : la répétition est plus prioritaire que la concaténation, elle-même plus prioritaire que l'alternative (utiliser des parenthèses)

Syntaxe des expressions régulières

Ainsi, pour $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$, le motif `ab*|ba*` est à comprendre comme `((a)(b)*)|((b)(a)*)`, ou



Sémantique des expressions régulières

- Une expression régulière a une valeur (un ensemble de mots)
- Définition inductive, qui à chaque expression régulière p associe un sous-ensemble $[\![p]\!]$ de Σ^*.

$$\begin{aligned} [\![\epsilon]\!] &= \{\epsilon\} \\ [\![c]\!] &= \{c\} \\ [\![p_1 \mid p_2]\!] &= [\![p_1]\!] \cup [\![p_2]\!] \\ [\![p_1 \cdot p_2]\!] &= [\![p_1]\!] \cdot [\![p_2]\!] \\ [\![p*]\!] &= [\![p]\!]^* \end{aligned}$$

- Lorsque m appartient au langage défini par p, on dit aussi que le motif p filtre le mot m.
- Un langage qui peut être défini comme la valeur d'une expression régulière est dit *langage* régulier.

Sémantique des expressions régulières

Les représentations décimales des entiers sont un langage régulier défini par :

$$0|(1|2|3|4|5|6|7|8|9)(0|1|2|3|4|5|6|7|8|9)*$$

(opérateur de concaténation omis)

Sémantique des expressions régulières

Remarque : Un langage qui contient un nombre fini de mots est régulier.

Si $L = \{m_1, m_2, \ldots, m_n\}$ est un langage fini contenant n mots, alors L est exactement décrit par l'alternative de tous les m_i.

Les mots constitués de 0 et de 1 alternés

- $[\![(01)*]\!]$ = Les mots constitués de 0 et de 1 alternés qui commencent par 0 et finissent par 1
- $[\![(10)*]\!]$ = Les mots constitués de 0 et de 1 alternés qui commencent par 1 et finissent par 0
- $[\![1(01)*]\!]$ = Les mots constitués de 0 et de 1 alternés qui commencent par 1 et finissent par 1
- $[\![0(10)*]\!]$ = Les mots constitués de 0 et de 1 alternés qui commencent par 0 et finissent par 0

Le langage recherché est donc $[\![(01)*|(10)*|1(01)*|0(10)*]\!]$
Mieux $[\![(\epsilon|1)(01)*(\epsilon|0)]\!]$

Exercices

- $[\![(\epsilon|1)(00*1)*0*]\!]$? ? ?
- $[\![(a*b*)*aaa(a|b)*]\!]$? ? ?
- $[\![(a|ba)*b*]\!]$? ? ?

Filtrage

La relation de filtrage $m \in [\![p]\!]$, également notée $p \preceq m$, peut être définie directement par les règles d'inférence (+ axiomes)

EMPTY
$$\epsilon \preceq \epsilon$$

CHAR
$$c \preceq c$$

SEQ
$$\frac{p_1 \preceq m_1 \qquad p_2 \preceq m_2}{p_1 p_2 \preceq m_1 m_2}$$

ORLEFT
$$\frac{p_1 \preceq m}{p_1 \mid p_2 \preceq m}$$

ORRIGHT
$$\frac{p_2 \preceq m}{p_1 \mid p_2 \preceq m}$$

STAREMPTY
$$p* \preceq \epsilon$$

STARSEQ
$$\frac{p \preceq m_1 \qquad p* \preceq m_2}{p* \preceq m_1 m_2}$$

Filtrage

En enchaînant les règles on obtient une preuve effective du prédicat, appelée *arbre de dérivation*. Voici par exemple la preuve de $\mathtt{ab*|ba*} \preceq \mathtt{baa}$.

$$\dfrac{\dfrac{\mathtt{b} \preceq \mathtt{b} \qquad \dfrac{\mathtt{a} \preceq \mathtt{a} \qquad \dfrac{\mathtt{a} \preceq \mathtt{a} \qquad \mathtt{a*} \preceq \epsilon}{\mathtt{a*} \preceq \mathtt{a}}}{\mathtt{a*} \prec \mathtt{aa}}}{\mathtt{ba*} \preceq \mathtt{baa}}}{\mathtt{ab*|ba*} \preceq \mathtt{baa}}$$

Notations supplémentaires

En pratique on rencontre diverses notations qui peuvent toutes être exprimées à l'aide des constructions de base des expressions régulières. L'emploi de ces notations n'augmente pas la classe des langages réguliers, mais permet simplement d'écrire des expressions régulières plus compactes.

- Le motif optionnel $p?$ défini comme $p \mid \epsilon$.
- La répétition au moins une fois $p+$ définie comme $pp*$.
- Le joker, noté $.$, qui est l'alternative de tous les caractères de Σ. On a donc $[\![.]\!] = \Sigma$.

Aujourd'hui

Expression régulière et automate

Théorème (Kleene). Tout langage reconnu par un automate fini déterministe peut être représenté par une expression régulière, et réciproquement.

Nous allons démontrer que

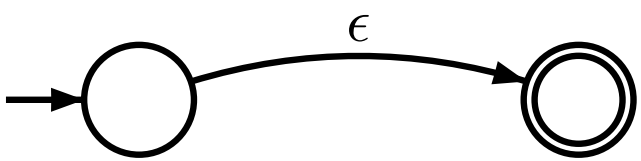
- Tout langage reconnu par une expression régulière est reconnu par un automate fini non-déterministe (et donc par ...)
- Tout langage reconnu par un automate fini déterministe est reconnu par une expression régulière

Tout langage reconnu par une expression régulière est reconnu par un automate fini non-déterministe

- Construction par induction d'un AFN avec ϵ transition à partir des opérations élémentaires de définition des exp. rationnelles.
  - Les automates construits ont exactement un état final, pas d'arc entrant dans q_0 pas d'arc sortant de l'état final
- Base : pour $\emptyset, \epsilon$ et $c \in \Sigma$ (trivial)
- Induction

Tout langage reconnu par une expression régulière est reconnu par un automate fini non-déterministe

Pour ϵ

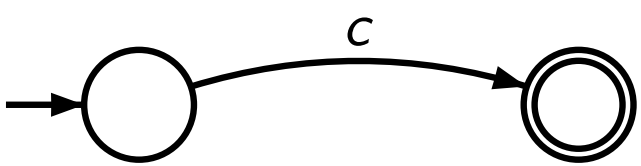


Tout langage reconnu par une expression régulière est reconnu par un automate fini non-déterministe

Pour $\emptyset$

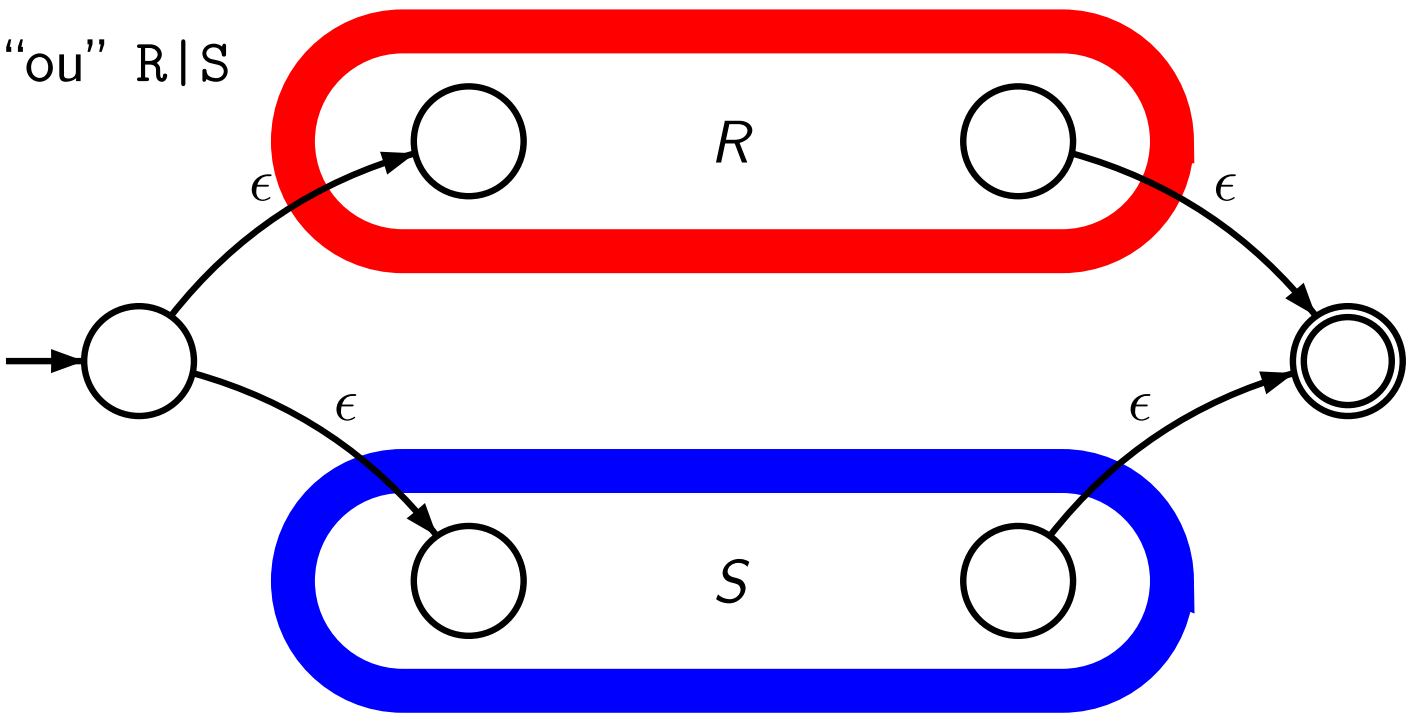
Tout langage reconnu par une expression régulière est reconnu par un automate fini non-déterministe

Pour $c \in \Sigma$



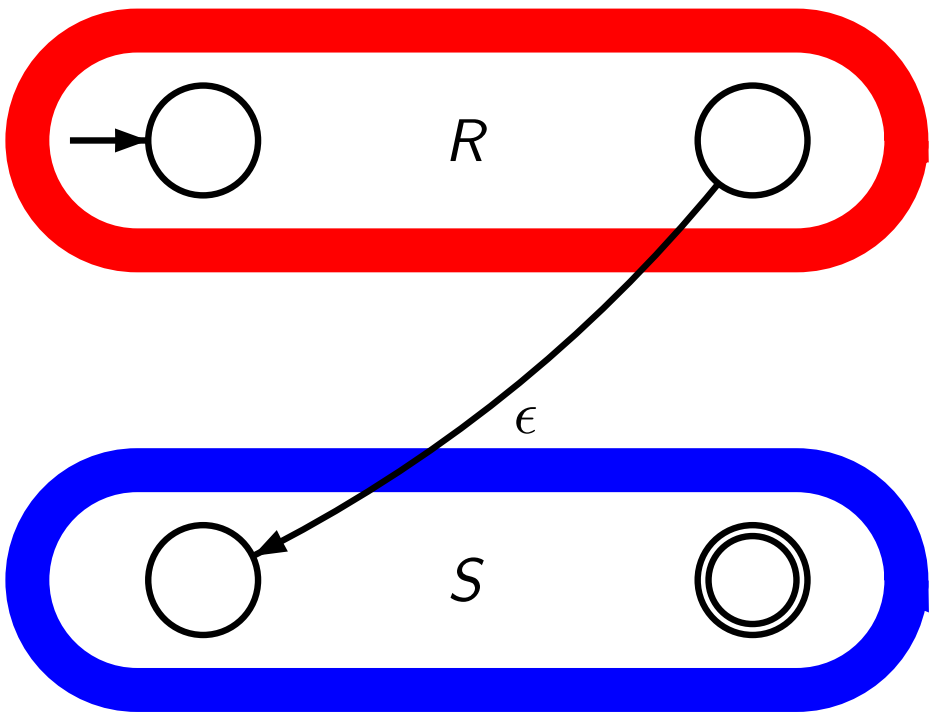
Tout langage reconnu par une expression régulière est reconnu par un automate fini non-déterministe

Induction : Le "ou" `R|S`



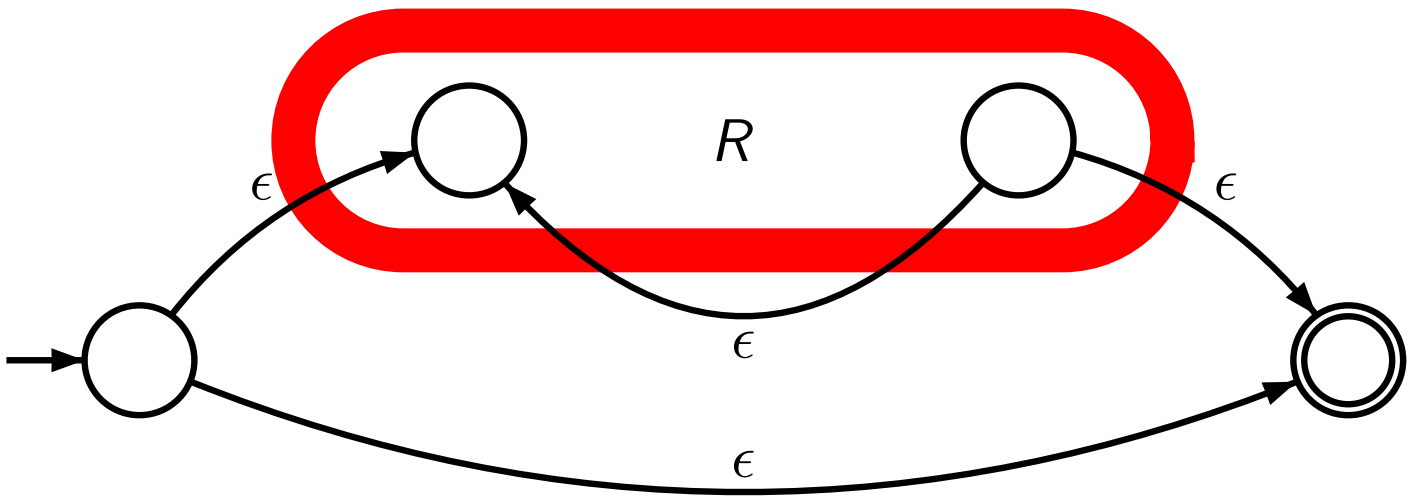
Tout langage reconnu par une expression régulière est reconnu par un automate fini non-déterministe

Induction : La concaténation RS



Tout langage reconnu par une expression régulière est reconnu par un automate fini non-déterministe

Induction : R*



La réciproque : Tout langage reconnu par un automate fini déterministe est reconnu par une expression régulière

C'est un peu plus compliqué... Nous allons utiliser de la programmation dynamique.

- Partons d'un automate fini déterministe A dont les états sont $1, 2, ..., n$. Notons $A(i, j)$ l'ensemble des symboles qui valuent les arcs (i, j).
- Si le mot m permet de passer de l'état i à l'état j dans A, alors m correspond à un chemin de i à j dans le graphe de l'automate.
- Soit R_{ij} l'expression régulière qui reconnaît les mots m qui correspondent à un chemin de i à j qui ne passe que par des états intermédiaires $u \leq k$

Nous allons construire toutes les expressions R_{ij}^k (induction)

Réciproque : $k = 0$

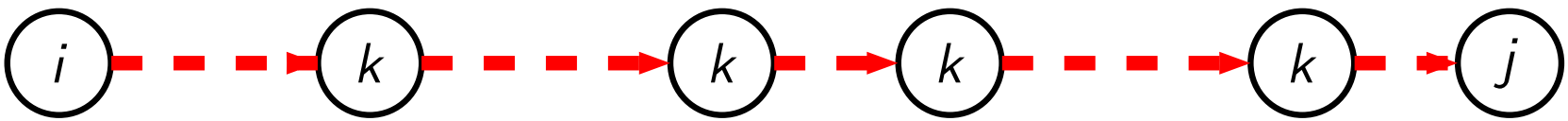
Soit m un mot qui correspond à un chemin de i à j sans sommet intermédiaire.

- Si $i = j$, le chemin est soit de longueur 0 (ϵ) soit c'est une boucle de i à i.
  $\Rightarrow R_{ii}^0 = (\epsilon|a_1|...|a_g)$ pour $A(i,i) = \{a_1, ..., a_g\}$
- Si $i \neq j$ et $A(i,j) = \emptyset$ alors $R_{ij}^0 = \emptyset$.
- Si $i \neq j$ et $A(i,j) \neq \emptyset$ alors $R_{ij}^0 = (a_1|...|a_g)$ pour $A(i,j) = \{a_1, ..., a_g\}$

Réciproque : Induction

Soit m un mot qui correspond à un chemin de i à j avec des sommets intermédiaires $u \leq k$.

- Si les sommets intermédiaires u sont strictement inférieurs à k alors m est reconnu par R_{ij}^{k-1}
- Sinon,



Ce qui correspond à $R_{ij}^{k} = (R_{ij}^{k-1} | R_{ik}^{k-1} (R_{kk}^{k-1}) * R_{kj}^{k-1})$

Réciproque

- Attention : n^3 expressions à construire
- Les expressions générées sont abominables

Nous avons vu

Les mêmes langages sont reconnus

- par les expressions régulières
- par les automates finis déterministes
- par les automates finis non-déterministes
- par les automates finis non-déterministes avec ϵ transition

Aujourd'hui

Un lemme classique ("pompage")

Pour tout langage régulier L, il existe un entier n tel que tout mot m de L de longueur $|m|$ supérieure ou égale à n se décompose en $m = xyz$ avec

- $y \neq \epsilon$
- $|xy| \leq n$
- $\forall k \geq 0, xy^k z \in L$

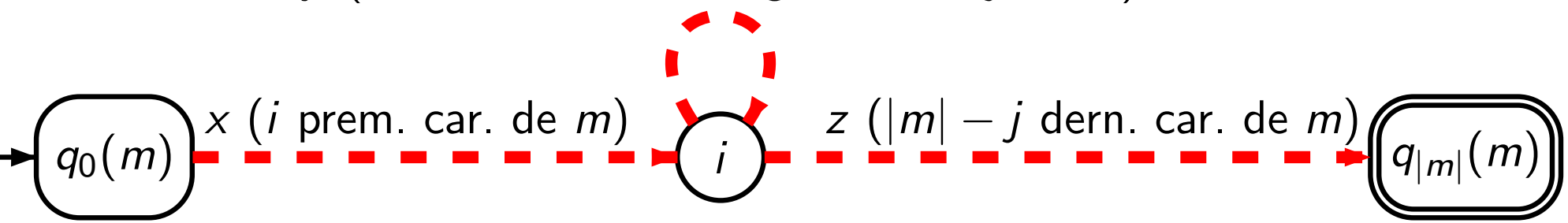
En clair, les gros mots peuvent être "cassés"

Un lemme classique ("pompage")

- Soit L régulier et A un automate fini déterministe qui le reconnaît. Soit n le nombre d'états de A
- Soit m un mot du langage L avec $|m| \geq n$ et notons $q_i(m)$ l'état de l'automate après avoir lu les i premiers symboles de m.
- Comme $|m| \geq n$, les états $q_0(m), q_1(m), ..., q_{|m|}(m)$ ne sont pas tous 2 à 2 distincts

Un lemme classique ("pompage")

- Soient donc i et $j > i$ les 2 premiers entiers tels que $q_i(m) = q_j(m)$



- Les mots xy^kz pour $k \geq 0$ sont donc aussi acceptés par l'automate.
- Par notre choix de i, j, nous avons $|xy| \leq n$.

Exercice

Soit $\Sigma = \{0, 1\}$ et considérons l'ensemble des mots qui contiennent autant de 1 que de 0. Le langage correspondant est-il régulier ?

Aujourd'hui

Les points importants d'INF421

- Des structures de données dynamiques
- Des algorithmes sur ces structures
- Des mesures de complexité un peu plus fines
- Qqes aspects de Java : L'utilisation de méthodes dynamiques, le this, les classes paramétrées
- Automates et langages

$\Rightarrow$ INF431

INF421, la pale

- Tous les documents sont autorisés
- Quand on vous demande d'écrire du code : rarement plus de 15 lignes (sauf XXX)
- Le poly est un sur-ensemble de ce que nous avons vu en amphi (formalisation +++)

INF421, le sondage

C'est IMPORTANT

"L'informatique"

- Domaine très très large
  - Algorithmique
  - Programmation
  - Mais aussi probabilité, analyse numérique, logique, algèbre, etc.
  - Et c'est aussi une discipline expérimentale
- As metionned by S. Ballmer last tuesday : "CS changes the world"
  - SO PLEASE SEND HIM YOUR RESUME