

Automates à états et langages

Notion d'automate

Langage reconnu par un automate

Automates non déterministes

Expressions régulières et automates

Limites des automates

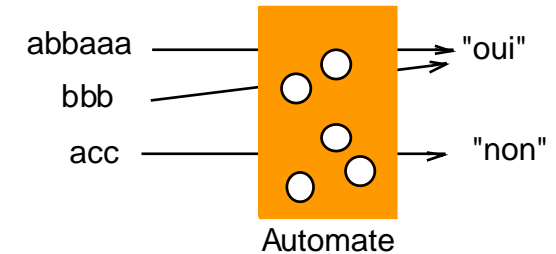
Notion d'automate

Mécanisme de "calcul" abstrait sans mémoire

Quel type de calcul ?

entrée : une séquence de symboles

sortie (réponse) : "oui" ou "non"



Idée du fonctionnement

L'automate possède des états

Il "lit" les symboles un à un (à partir de la gauche)

La lecture d'un symbole

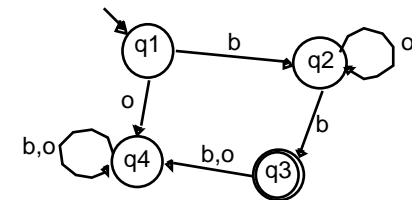
- fait passer l'automate dans un autre état
- en fonction d'une "table"

On regarde l'état atteint après lecture de tous les symboles

Un automate à états fini

- symboles : {b, o}
- états : {q1, q2, q3, q4}
- états finals : {q3}
- état initial : q1
- fonction de transition :

δ :	b	o
q1	q2	q4
q2	q3	q2
q3	q4	q4
q4	q4	q4

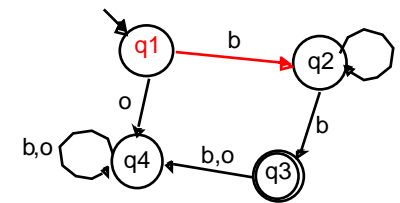


Automate - définition

Un AF est composé de

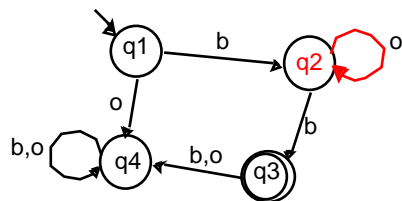
1. un alphabet de symboles reconnus $A = \{s_1, s_2, \dots, s_n\}$
2. un ensemble d'états $Q = \{q_1, q_2, \dots, q_m\}$ (les ronds)
3. un état initial q_1 de Q
4. un ensemble F d'états finals inclus dans Q
5. une fonction de transition δ de $Q \times A$ dans Q (les flèches)

Analyse d'une chaîne



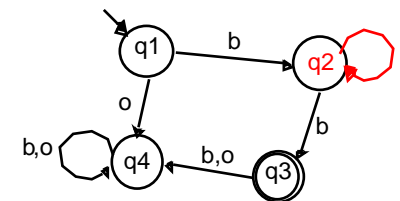
b o o b b b
↑

Analyse d'une chaîne



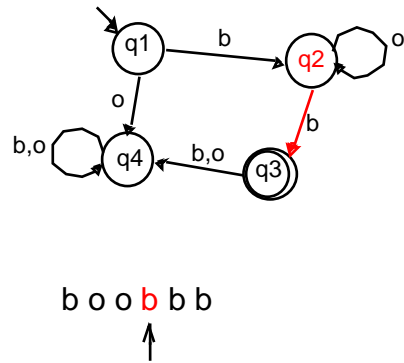
b o o b b b
↑

Analyse d'une chaîne

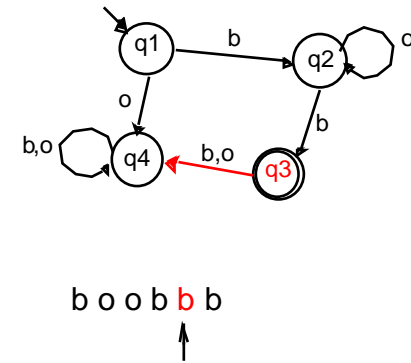


b o o b b b
↑

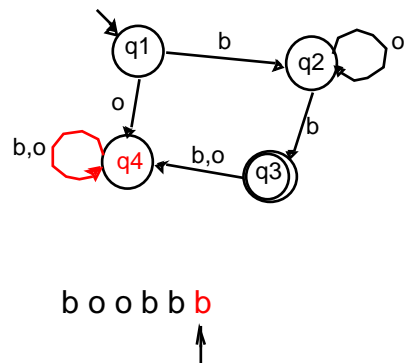
Analyse d'une chaîne



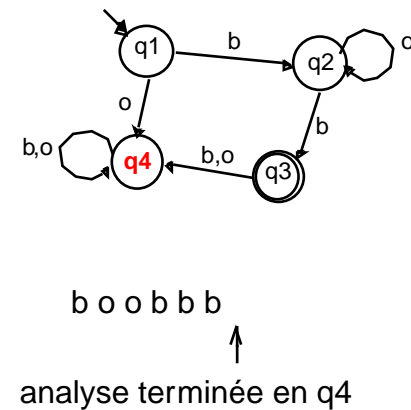
Analyse d'une chaîne



Analyse d'une chaîne



Analyse d'une chaîne



Acceptation d'une chaîne

Le traitement d'une chaîne de symbole $u = [t_1 t_2 \dots t_k]$ par un automate consiste à

1. **Etat** := état initial;
2. répéter pour i allant de 1 à k :
 (Nouvel) Etat := $\delta(\text{Etat}, t_i)$
3. si **Etat** $\in F$ accepter u sinon rejeter u

Définition formelle de l'acceptation

On définit la fonction $\delta^*(e, u)$

si u est la chaîne vide :

$$\delta^*(e, \varepsilon) = e ;$$

si $u = tw$ (commence par le symbole t suivi de la chaîne w) :

$$\delta^*(e, tw) = \delta^*(\delta(e, t), w).$$

Une chaîne u est acceptée par l'automate M dont l'état initial est q_0 si $\delta^*(q_0, u)$ appartient aux états finals de M .

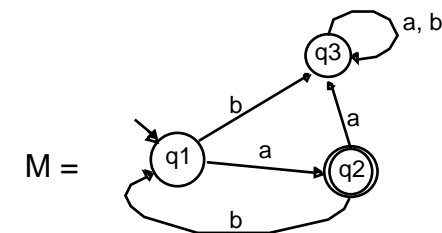
Langage accepté par un automate

Parmi toutes les chaînes de A^*

Celles qui sont acceptées par l'automate M

Forment le **langage** $L(M)$.

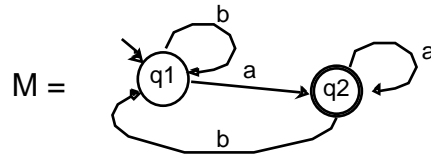
Exemple



Langage accepté

$$L(M) = \{ a, aba, ababa, abababa, \dots \}$$

Exemple - 2



Langage accepté

$L(M) = ?$

Langage régulier

On dit qu'un langage L est régulier s'il existe un automate M tel que $L = L(M)$,

c-à-d un automate qui accepte les chaînes de L et seulement celles-ci.

Exemples / exercices

- les chaînes sur $\{a, b\}$ qui commencent par aa et se terminent par bb ;
- les chaînes sur $\{a, b\}$ qui se terminent par $baba$;
- les chaînes sur $\{a, b\}$ qui se terminent par $bbab$;
- les chaînes sur $\{a\}$ qui ont au plus 5 symboles;
- les chaînes sur $\{a, b, c\}$ où chaque c est précédé de deux a ;
- les chaînes sur $\{0, 1\}$ qui, interprétées comme des nombres en base 2, sont des multiples de 5;
- les chaînes formées de $6k + 1$ fois le symbole a (k entier positif quelconque).

Automates non déterministes

1) La fonction de transition est *multivaluée*,

pour un état e et un symbole s
il peut y avoir *plusieurs états successeurs* :

$$\delta(e, s) = \{e_1, e_2, \dots, e_n\}.$$

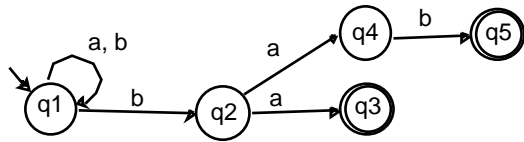
2) L'automate peut avoir des transitions ϵ qui mènent d'un état à un autre sans consommer un seul symbole.

\Rightarrow

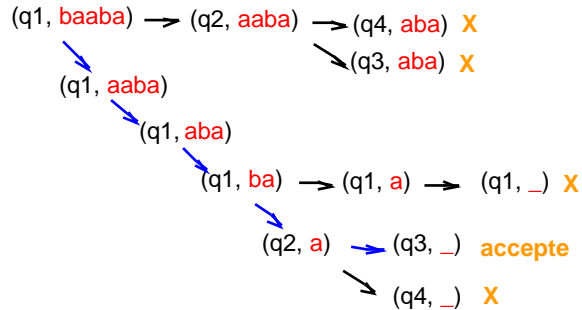
Il y a plusieurs manières d'analyser une chaîne,

Si l'une d'elles conduit à un état final, la chaîne est acceptée.

Exemple



Toutes les séquences de calcul possibles pour **baaba** :



Fonctionnement d'un AFND

Description :

Une description instantannée de l'automate est une paire
 $(\langle \text{état} \rangle, \langle \text{chaîne restant à analyser} \rangle)$

Transition :

$(e, ax) \vdash (f, x)$

- si $f \in \delta(e, a)$

$(e, x) \vdash (f, x)$

- si $f \in \delta(e, \epsilon)$

Acceptation dans un AFND

Une chaîne w est acceptée s'il existe une séquence de descriptions

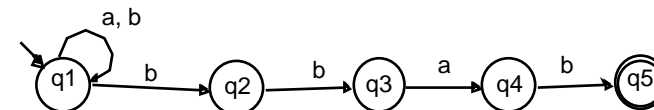
$$s_0 = (q_0, w) \vdash s_1 \vdash s_2 \dots \vdash s_f = (q_f, \langle \text{vide} \rangle)$$

où q_f est un état final.

Autrement dit : s'il y a un moyen d'atteindre un état final en lisant **toute** la chaîne d'entrée.

Exemple

Automate non déterministe pour reconnaître les chaînes de **a** et **b** se terminant par **bbab**

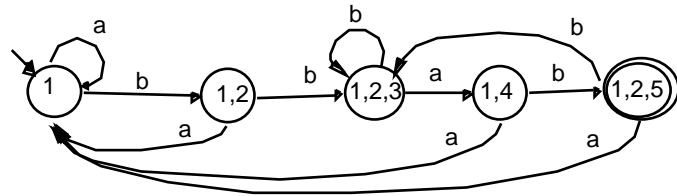
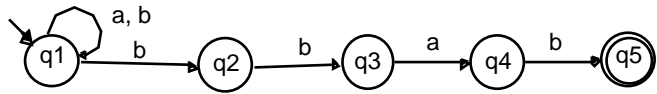


Exemple : acceptation de babbab

$s_0 = (q_1, babbab) \vdash (q_1, abbab) \vdash (q_1, bbab) \vdash (q_2, bab) \vdash (q_3, ab) \vdash (q_4, b) \vdash (q_5, \langle \text{vide} \rangle)$ accepte

Les non déterministes ne sont pas si forts

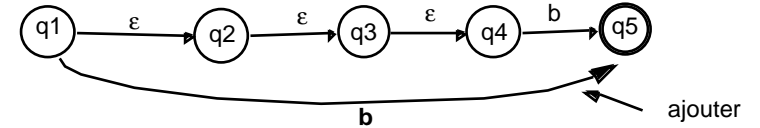
On peut toujours trouver un AFD équivalent à un AFND donné



Construction Non Dét → Dét

1. Construire un automate non déterministe sans transitions ϵ

a) Ajouter des transitions pour remplacer toutes les chaînes de ϵ

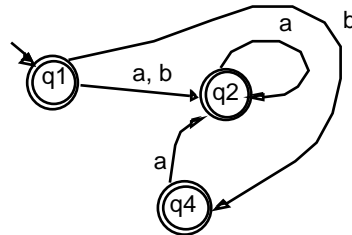
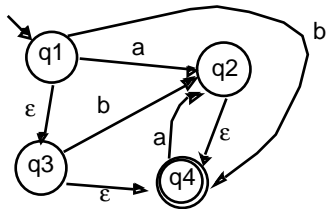


b) Rendre final tout état qui atteint un final par des ϵ

c) Eliminer les transitions ϵ

d) Eliminer les états atteignable uniquement par des transitions ϵ

Exemple



Construction (II)

Etats

Les états de l'automate déterministe sont des ensembles d'états de l'automate non déterministe.

Si l'ensemble contient un état final il est lui-même final.

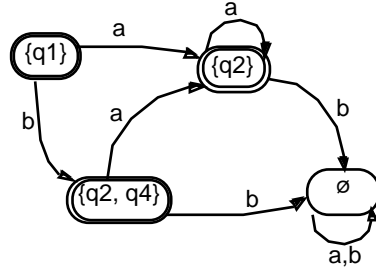
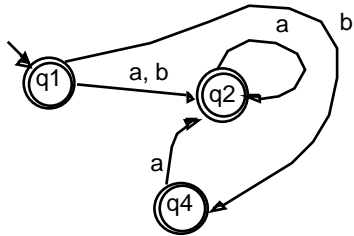
Transitions

$S = \{s_1, \dots, s_k\}$ un état de l'AD

a un symbole

$\delta(S, a) = \{l'ensemble des états de l'AND atteignable avec le symbole a depuis l'un des s_i \}$

Exemple



Création de langages réguliers

On peut démontrer que

si L_1 et L_2 sont des langages réguliers, $L_1 \cup L_2$ est aussi un l.r.

si L_1 est un l.r. alors $A^* - L_1$ aussi

si L_1 et L_2 sont des langages réguliers, $L_1 \cap L_2$ aussi

\emptyset , $\{\epsilon\}$ et $\{u\}$ (pour tout u dans A^*) sont des l.r.

si L_1 et L_2 sont des langages réguliers, $L_1 \cdot L_2$ aussi

$L_1 \cdot L_2 = \{ uv \mid u \in L_1 \text{ et } v \in L_2 \}$ (concaténation de L_1 et L_2)

(suite)

si L est régulier alors L^* aussi

$$L^* = \{u_1 u_2 \dots u_n \mid n \geq 0 \text{ et } u_1, u_2, \dots, u_n \in L\}$$

Les démonstrations consistent à construire les automates correspondant (à partir d'automates sans retour à l'état initial).

On peut aussi démontrer que le langage défini par un automate peut être obtenu par ces opérations, à partir des symboles.

Exemples

Le langage $\{a, cab, bac\}$:

$$\{a\} \cup (\{c\} \cdot \{a\} \cdot \{b\}) \cup (\{b\} \cdot \{a\} \cdot \{c\})$$

Toutes les chaînes contenant des a et des b :

$$(\{a\} \cup \{b\})^*$$

Les chaînes de a , b et c qui commencent par a et finissent par b ou c :

$$\{a\} \cdot (\{a\} \cup \{b\} \cup \{c\})^* \cdot (\{b\} \cup \{c\})$$

Exemple

Nombre entier en décimal :

"Soit '0' soit une séquence de chiffres qui ne commence par pas '0' et qui peut être précédée d'un signe '-'"

(on n'admet pas '007')

Langage régulier:

$\{0\} \cup (\{-\} \cup \{\varepsilon\}) \cdot (\{1\} \cup \dots \cup \{9\}) \cdot (\{0\} \cup \{1\} \cup \dots \cup \{9\})^*$

Expression régulière

ε : le langage qui ne contient que la chaîne vide

a : le langage qui ne contient que la chaîne a

+ : union de deux langages (ou)

* : répétition 0, 1 ou plusieurs fois (fermeture)

<rien> : concaténation

() : sous-expressions

$\{a\} \cdot (\{a\} \cup \{b\} \cup \{c\})^* \cdot (\{b\} \cup \{c\}) \cdot \{c\} \cdot \{c\} \cdot \{a\}$

dans la notation des e.r. :

$a(a+b+c)^*(b+c)cca$

Exemple - 2 - Expression régulière

Les nombres entiers :

0 +

$((- + \varepsilon) (1+2+3+4+5+6+7+8+9) (0+1+2+3+4+5+6+7+8+9)^*)$

Toutes les chaînes de a et de b : $(a+b)^*$

Attention:

$(ab)^* = \{ab, abab, ababab, \dots\}$

$a^* + b^* = \{a, aa, aaa, \dots, b, bb, bbb, \dots\}$

Conception d'un automate pour un langage

Principe

1. Langage à reconnaître
2. Mettre le langage sous forme d'expression régulière
3. Créer un automate non-déterministe d'analyse [auto]
4. Transformer en automate déterministe [auto]

Expr. régulière → automate (ND)

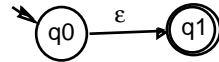
expr. rég.

automate

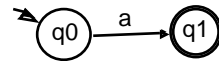
\emptyset (langage vide)



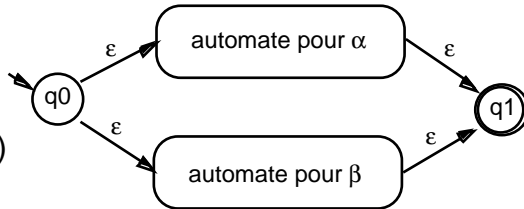
ε (que la chaîne vide)



a (que la chaîne a)

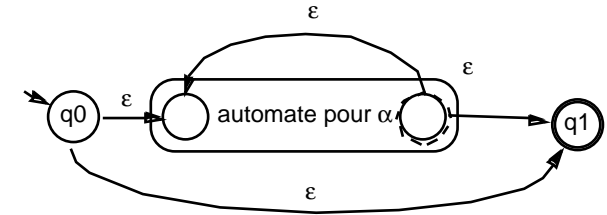


$\alpha + \beta$
(la chaîne α ou β)

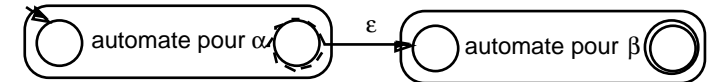


e.r. → automate (suite)

α^*

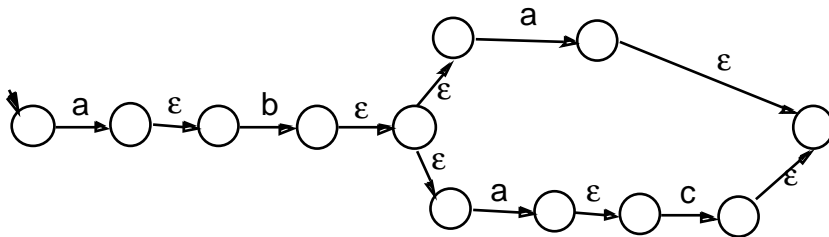


$\alpha \beta$

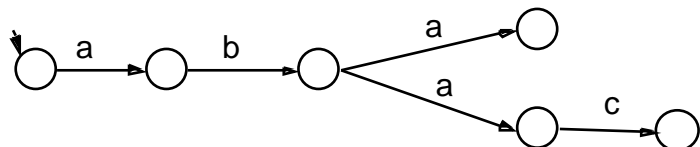


Exemple

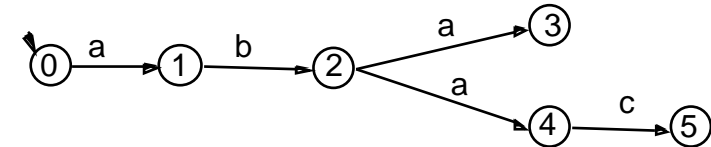
Reconnaître le langage $ab(a+ac)$



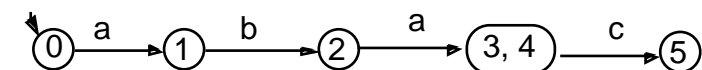
Sans transitions ε :



Exemple (suite)

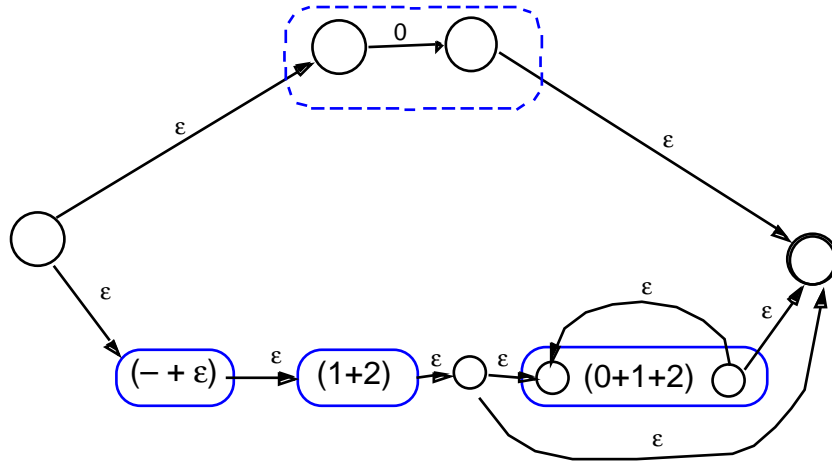


Transformation en déterministe :



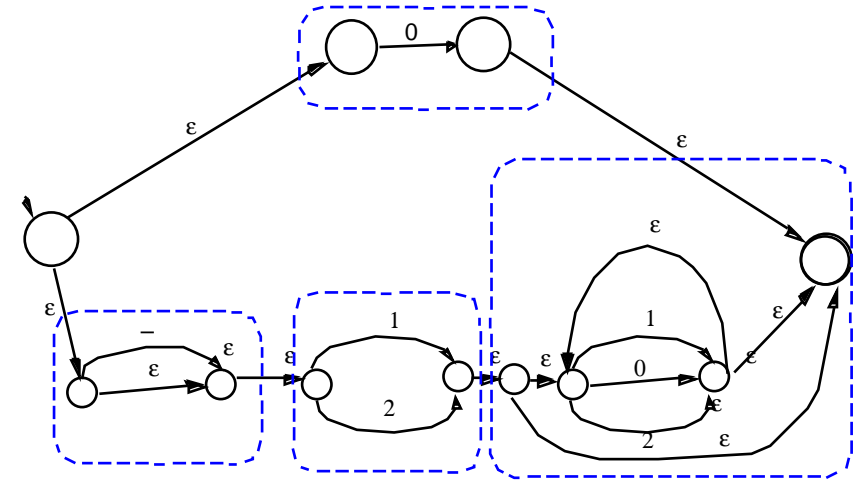
Exemple

$0 + ((- + \varepsilon) (1+2) (0+1+2)^*)$



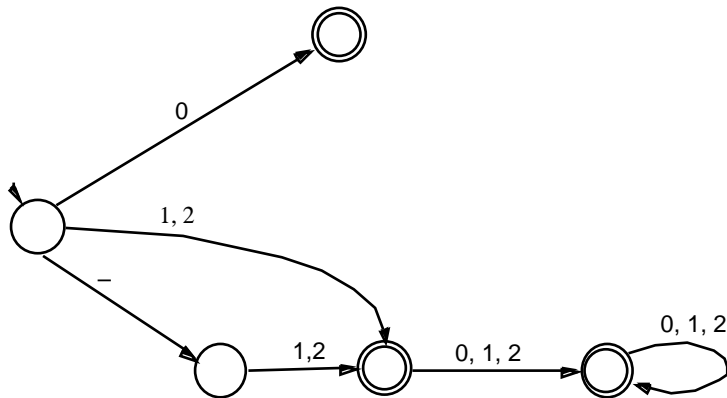
Exemple (suite)

$0 + ((- + \varepsilon) (1+2) (0+1+2)^*)$



Exemple (déterministe)

$0 + ((- + \varepsilon) (1+2) (0+1+2)^*)$



Applications pratique

Génération de programmes d'analyse de langage régulier

p.ex. analyse lexicale pour les compilateurs

Recherche de chaînes appartenant à un l.r.

p.ex. Unix : outils grep, egrep, etc.

recherche "avancée" dans les traitements de texte

Unix grep

Recherche toutes les lignes d'un fichier qui contiennent une chaîne appartenant au langage donné par une e.r.

```
% grep 'for *(' ns.java
    for (int i = 0; i < u.length; i++) {
    for (int i = 0; i < p.length; i++) {
        for (int i = 0; i < bw.length; i++) {
        for (int i = 0; i < an.length; i++){

% grep 'int [a-zA-Z][a-zA-Z0-9]* *= *[a-zA-Z\(\)]' ns.java
int nodeNameStart = ix + 14; // !! length of include markup
int nodeNameEnd = s.indexOf("&u=", ix); // xml jg
int paramEnd = s.indexOf(includeSuffix, nodeNameEnd);
int ifocus = s.indexOf("<focus>expanded</focus>");
int beginAnchor = s.indexOf(">", ix) + 1;
```

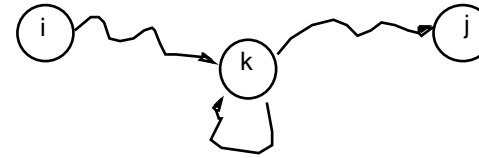
À tout automate correspond une e.r.

Idée de la preuve :

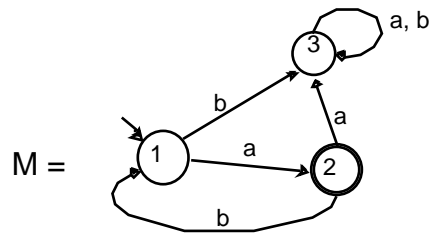
Etudier les e.r. de tous les chemins possibles entre états

C_{ij}^k somme des e.r. des chemins de i à j ne passant que par des états de $\{0, 1, \dots, k\}$ (sauf aux extrémités)

$$C_{ij}^k = C_{ij}^{k-1} + C_{ik}^{k-1} (C_{kk}^{k-1})^* C_{kj}^{k-1}$$



Exemple



$$C_{11}^0 = \varepsilon; C_{22}^0 = \varepsilon; C_{33}^0 = a+b; C_{12}^0 = a; C_{21}^0 = b; \text{etc.}$$

$$C_{12}^1 = \varepsilon + C_{11}^0 (C_{11}^0)^* C_{12}^0 = a$$

$$C_{22}^1 = \varepsilon + C_{21}^0 (C_{11}^0)^* C_{12}^0 = b\varepsilon a = ba$$

$$C_{12}^2 = C_{12}^1 + C_{12}^1 (C_{22}^1)^* C_{12}^1 = a + a(ba)^*ba$$

Exemple (suite)

$$C_{12}^3 = C_{12}^2 + C_{13}^2 (C_{33}^2)^* C_{32}^2$$

$$= C_{12}^2 + C_{13}^2 (C_{33}^2)^* \emptyset$$

$$= C_{12}^2 + \emptyset$$

$$= a + a(ba)^*ba$$

$$\text{Donc } L(M) = a + a(ba)^*ba$$

On peut simplifier :

$$= a (\varepsilon + (ba)^*ba) = a ((ba)^*) = a (ba)^*$$

suite de la preuve

Si A possède

- les états $\{0, 1, \dots, k\}$,
- l'état initial 0
- les états finale $\{f1, \dots, fn\}$

l'expression régulière de A est

$$C_{0f1}^k + C_{0f1}^k + \dots + C_{0fn}^k$$

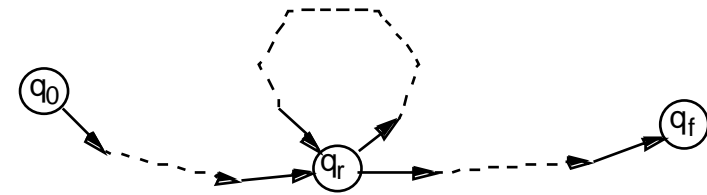
Il y a des langages non réguliers

M un automate avec n états $\{q_0, \dots, q_{n-1}\}$

$w = w_1 \dots w_p$ une chaîne de longueur $p > n$

La succession de transitions qui mène à un état final q_f en acceptant w

passé forcément deux fois (ou plus) par le même état (disons q_r).



-->

Suite des états rencontrés lors de l'acceptation de w :

$w: w_1 w_2 \dots w_r \dots w_s \dots w_p$

etat: $q_0 q_1 \dots q_r \dots q_r \dots q_f$

Il y a donc un circuit de q_r à q_r dans l'automate.

Si M accepte w , M accepte aussi toutes les chaînes de la forme $w_1 \dots w_r (w_{r+1} \dots w_s)^* w_{s+1} \dots w_p$

Lemme de pompage

Si M possède n états

et accepte une chaîne w de longueur supérieure à n

Alors il existe trois sous-chaîne x , y et z de w telles que

- $w = xyz$
- y est non vide
- M accepte toutes les chaînes de la forme $x y^* w$.

Utilisation

Ceci nous permet de montrer qu'il est impossible de construire des automates pour certains langages.

$a^n b^n$ n'est pas régulier

Supposons que M possède m états et accepte les chaînes de la forme $a^n b^n$ (n quelconque) **et uniquement celles-ci**.

$w = a^m b^m$ est de taille supérieure à m,

Donc il doit exister trois sous chaînes x, y, z de w telles que

$w = a^m b^m = xyz$ et M accepte xyyz (entre autres).

- si y ne contient que des a (resp. b), il y aura plus de a (b) que de b (a) dans xyyz
- si y contient des a et de b, disons $a^i b^j$, xyyz sera de la forme $a^k a^i b^j a^i b^j b^r$

Donc M **acceptera des chaînes n'appartenant pas à $a^n b^n$**

Conséquences théoriques

On peut facilement trouver des langages qu'aucun automate à états fini n'accepte.

L'AEF ne peut pas tout "calculer"

Il faut inventer des modèles de machines plus puissants que les aef pour reconnaître ces langages.

- automates à pile
- machines de Turing
- λ -calcul