Minimisation d'un AFD

Regroupement des états équivalents:

- Les états β-équivalents:
  2 états qi et qj sont dits β-équivalents s'ils permettent d'atteindre les états finaux à travers les mêmes mots. On écrit alors: qi β qj.

Minimisation d'un AFD

- Regroupement des états équivalents
- Exemple1:
- Soit AEF suivant (initial=0, finaux={2,3})

Etat	a	b
->0	1	0
1	1	2
2	3	2
3	3	2

Etat	a	b
->0	1	0
1	1	2
2	3	2

Minimisation d'un AFD

Regroupement des états équivalents

- Exemple2: soit AEF suivant (initial=1, finaux={1,2})

1. Nous avons éliminé les états inaccessibles (l'état 7)
2. Nous ne trouvons pas des états ayant la même fonction donc pour réduire les états il faut construire des classes d'équivalence en suivant l'algorithme de réduction des états

Etat	a	b
1	2	5
2	2	4
3	3	2
4	5	3
5	4	6
6	6	1

Minimisation d'un AFD

Regroupement des états équivalents:

- Algorithme de réduction des états: fusion des états β - équivalents d'un AEF $A=(X, Q, q0, F, \delta)$:
  1. Créer 2 classes d'états: $A=F$, $B=Q - F$
  2. S'il existe un symbole 'a' et 2 états qi et qj d'une même classe tel que $\delta(qi,a)$ et $\delta(qj,a)$ n'appartiennent pas à la même classe, alors créer une nouvelle classe et séparer qi et qj. On laisse dans la même classe tous les états qui donnent des états d'arrivée dans la même classe
  3. Recommencer l'étape 2 jusqu'à ce qu'il y ai plus d'états à séparer

Minimisation d'un AFD

Regroupement des états équivalents

- Exemple:
  - **Etape1:** créer deux classes d'états:

1. A={1,2} (les états finaux)
2. B={3,4,5,6} (les états non finaux)

  - **Etape2:** vérifier la cohérence des classes: On dit d'une classe A est cohérente par rapport à un symbole 'a' si toute les transitions de A avec 'a' mènent à la même classe. Si une classe n'est pas cohérente il faut la découper au moins en 2 parties jusqu'à trouver des classes cohérentes

Etat	a	b
1	2	5
2	2	4
3	3	2
4	5	3
5	4	6
6	6	1

Minimisation d'un AFD

- Regroupement des états équivalents: nous allons procéder par itération:
  - I ère itération: (A={1,2}, B={3,4,5,6})

Etat	a	b
1	2	5
2	2	4
3	3	2
4	5	3
5	4	6
6	6	1

A	a	b
1	2 ∈A	5∈B
2	2∈A	4∈B

A est cohérente

B	a	b
3	3∈B	2∈A
4	5∈B	3∈B
5	4∈B	6∈B
6	6∈B	1∈A

B n'est pas cohérente. Il faut donc l'éclater en 2 classes B={3,6} et C={4,5}

Minimisation d'un AFD

- Regroupement des états équivalents:
  nous allons procéder par itération:
  - 2ième itération: (A={1,2}, B={3,6}, C={4,5})

A	a	b
1	2 ∈A	5∈C
2	2∈A	4∈C

B	a	b
3	3∈B	2∈A
6	6∈B	1∈A

C	a	b
4	5∈C	3∈B
5	4∈C	6∈B

A est cohérente

B est cohérente

C est cohérente

Donc l'AEF contiendra 3 états A,B et C

Minimisation d'un AFD

- **Regroupement des états équivalents:**
  Résultat de l'algorithme:

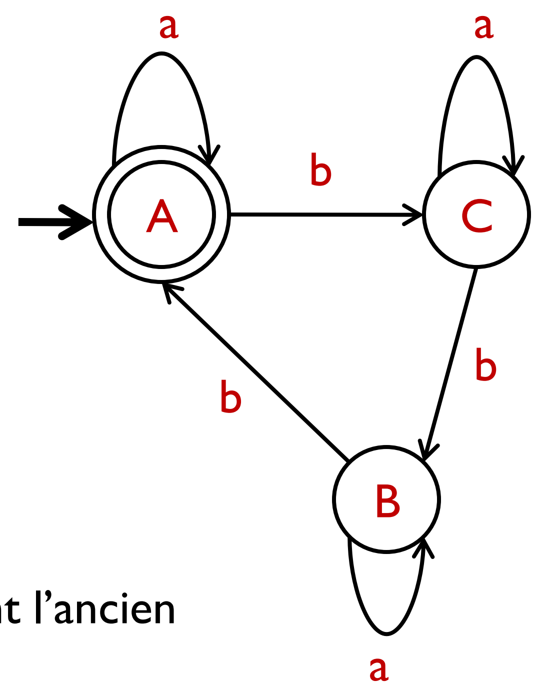
A	a	b
1	2 ∈A	5∈C
2	2∈A	4∈C

B	a	b
3	3∈B	2∈A
6	6∈B	1∈A

C	a	b
4	5∈C	3∈B
5	4∈C	6∈B

AFD minimal

Etat	a	b
A	A	C
B	B	A
C	C	B



Etat initial: la classe qui contient l'ancien état initial: A
Etats finaux: les classes qui contiennent des anciens états finaux: A

Langage régulier et expression régulière

- Les langages réguliers sont les langages générés par des grammaires de type 3 (ou encore grammaires régulières).
- Ils sont reconnus grâce aux automates à états finis.
- Le terme régulier vient du fait que les mots de tels langages possèdent une forme particulière pouvant être décrite par des expressions dites régulières.

Langage régulier

Définition:

Un langage L définit sur l'alphabet X est dit régulier, s'il est obtenu par un nombre d'applications **fini** des opérations (Union, concaténation ou étoile de Kleene) sur les langages réguliers de base suivants:

- Le langage vide ∅
- Le langage qui ne contient que le mot vide { ε }
- Le langage de la forme {a} avec a ∈ X.

⇒Si L1 et L2 sont des langages réguliers sur X alors:

L1∪ L2, L1.L2 et L1* sont aussi des langages réguliers.

Langage régulier

Exemples

Soit $X=\{a,b\}$

1. $L=X^*=\{a,b\}^*$ est un langage régulier car il est obtenu par l'application d'une étoile de kleene sur l'union de 2 langages réguliers de base $\{a\}$ et $\{b\}$

   $\{a,b\}=\{a\} \cup \{b\} \Rightarrow \{a,b\}^*=\{\{a\} \cup\{b\}\}^*$
2. $L=\{a^n b^n \ / \ n<=2\}$ est un langage régulier puisqu'il est obtenu par l'application d'un nombre fini de l'union de la concaténation des langages réguliers $\{a\}$, $\{b\}$ et $\{\varepsilon\}$ $L=\{\varepsilon , ab, aabb\}=\{\varepsilon \} \cup\{a\}.\{b\} \cup \{a\}.\{a\}.\{b\}.\{b\}$
3. $L=\{a^n/ \ n>=0\}$ est un langage régulier puisqu'il est obtenu par l'application de l'étoile de Kleene sur le langage régulier $\{a\}$

Langage régulier

4. $L=\{a^nb^m \ / \ n,m>=0\}$ est un langage régulier puisqu'il est obtenu par l'application d'une concaténation de l'étoile de Kleene du langage régulier {a} avec l'étoile de Kleene du langage régulier {b}: L= {a}*.{b}*
5. $L=\{a^nb^n \ / \ n>=0\}$ n'est pas un langage régulier puisqu'il est obtenu par l'application d'un nombre **infini** de l'union de la concaténation des langages réguliers {a}, {b} et {ε}

   L={ε}U{a}.{b}U{a}.{a}.{b}.{b}….U{a}.{a}….{a}.{b}.{b}...{b}

Langage régulier

- **Propriétés:** Soit un alphabet A:

1. Pour tout mot $u \in A^*$, le langage {u} est régulier.
   Si u s'écrit u=a1a2…an sur A, alors le langage {u} s'écrit comme la concaténation {u}={a1}.{a2}…{an}.
   {u} est régulier car chaque {a1} est un langage régulier.
2. Tout langage fini est régulier
3. Un langage régulier peut être infini

Langage régulier

- **Propriétés:**

4. Soit L et M 2 langages réguliers alors les langages suivants sont réguliers:
   1. Union: $L \cup M$
   2. Intersection: $L \cap M$
   3. Renversement: $L^R=\{w^R / w \in L\}$
   4. Fermeture : L^*
   5. Concaténation: $L.M$

Expressions régulières

Définitions: Soit X un alphabet quelconque ne contenant pas les symboles {*,+,|,.,(,)}.
Une expression régulière est un mot E défini sur l'alphabet X ∪ {*,+,|,.,(,)} si seulement si:

- E= Ø ou
- E= ε ou
- E=a / a ∈ X ou
- E=E1 | E2 avec E1 et E2 sont deux expressions régulières sur X ou
- E=E1.E2 avec E1 et E2 sont deux expressions régulières sur X ou
- E=E1* et E1 est une expression régulière sur X
- E=E1+ et E1 est une expression régulière sur X

Les opérateurs *,. et | ont une priorité décroissante. Si nécessaire, on peut ajouter des parenthèses.

Expressions régulières

Les opérateurs

Opérateurs	Signification	Exemple	Opération
*	Répéter 0 ou plusieurs fois	a*: répéter a 0 ou plusieurs fois	Fermeture transitive de Kleene
|	Le choix	a|b: correspond à a ou b	Union
.	La concaténation	a.B ou ab	Concaténation
( )	Marquer la priorité	(a) Et a signifie la même chose	Ce n'est pas une opération
+	Répéter une ou plusieurs fois	a+: répéter a une ou plusieurs fois	Fermeture positive de Kleene

Expressions régulières

- **Exemples:**
  - **a*:** dénote les mots: ε, a, aa, aaa, ….a^n
  - **(a|b)*:** dénote les mots dans lesquels le symbole a ou b se répètent un nombre quelconque de fois. C'est le langage de tous les mots sur {a,b}: ε, a, b, aa, bb, ab, abab, aaaabbb… etc
  - **(a|b)*ab (a|b)*:** dénote les mots sur {a,b} contenant le facteur ab: ab, aabb, aaabbb, abbabaabb
  - **b+:** dénote les mots; b, bb, bbb, bbbbb…………

Langage décrit par une expression régulière

- **Un langage L(E) décrit par une expression régulière E définie sur un alphabet X est un langage régulier défini par:**
  - $L(E) = \emptyset$ si $E = \emptyset$
  - $L(E) = \{\varepsilon\}$ si $E = \varepsilon$
  - $L(E) = \{a\}$ si $E = a$
  - $L(E) = L(E1) \cup L(E2)$ si $E = E1 \mid E2$
  - $L(E) = L(E1).L(E2)$ si $E = E1.E2$
  - $L(E) = L(E1)^*$ si $E = E1^*$ où E1 est une expression régulière sur X

Langage décrit par une expression régulière

- Exemples
- Sur l'alphabet X={a,b,c}:

1. E2=(ab)* est une expression régulière qui décrit le langage: L(E2)={ab}*={ε, ab, abab, ababab,…} ={$(ab)^n$/n>=0}
2. E4=a*bbc* est une expression régulière qui décrit le langage L={$a^n bbc^m$/ n>=0, m>=0}
3. E5=(a | b | c)*(bb | cc)a* décrit le langage L(E5)={$wbba^n$, $wcca^n$/ w ∈ X*, n>=0}

Les types des expressions régulières

- Ils existent plusieurs types d'expressions, parmi lesquelles on cite:
  - Les expressions régulières basés sur les caractères Jocker
  - Les expressions régulières POSIX de Unix

Les types des expressions régulières

- Les expressions régulières basés sur les caractères Jocker:
- Elles sont utilisées sous Windows en ligne de commande, dans certaines commandes Linux, et en SQL dans la construction like pour rechercher des informations. Les caractères les plus utilisées sont:
  - * : signifie 0 ou plusieurs caractères quelconques
  - ? : Signifie un caractère quelconque
  - % : signifie tous les autres caractères
- Exemple:

La commande sur Linux: ls a*b?: Elle renvoie tous les fichiers qui commencent par a et l'avant dernier symbole est b

Les types des expressions régulières

- Les expressions régulières POSIX:
- Elles offrent un langage plus puissant permettant de:
  - Noter des langages réguliers avec des formes complexes
  - Noter même les langages non réguliers
- Elles sont utilisées dans:
  - Les commandes Linux: find, grep.. Ect
  - Dans tout les langages de programmation modernes
  - Des éditeurs de textes: par exemples, rechercher ^http:// et remplacer par HTTP

Les types des expressions régulières

- Les expressions régulières POSIX:

Expression	Signification
[abc]	Les symboles a, b, c
[^abc]	Aucun des symboles a, b et c
[a-e]	Les symboles de a jusqu'à e
.	N'importe quel symbole sauf le symbole fin de ligne
a*	a se répétant 0 ou plusieurs fois
a+	a se répétant 1 ou plusieurs fois
a?	a se répétant 0 ou une fois
a|bc	Le symbole a ou b suivi de c
a{2,}	a se répétant au moins 2 fois
a{,5}	a se répétant au plus 5 fois
a{2,5}	a se répétant entre 2 et 5 fois
\x	La valeur réelle de x (où x est un caractère spécial)

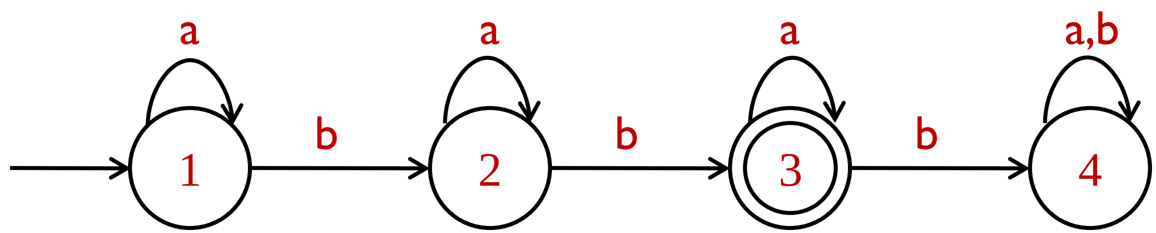
Les types des expressions régulières

- Les expressions régulières POSIX:
- Exemples:
  - [ab]* : tous les mots sur {a,b}
  - [^ab]* : les mots qui ne comportent ni a ni b
  - ([^a]*a[^a]* a[^a]*)* : les mots comportant un nombre paire de a
  - (ab{,4})* : en plus de ε, ce sont tous les mots commençant par 'a' où chaque 'a' est suivi de quatre 'b' au plus.

Automates et expressions régulières

Le passage d'un AEF vers une ER:

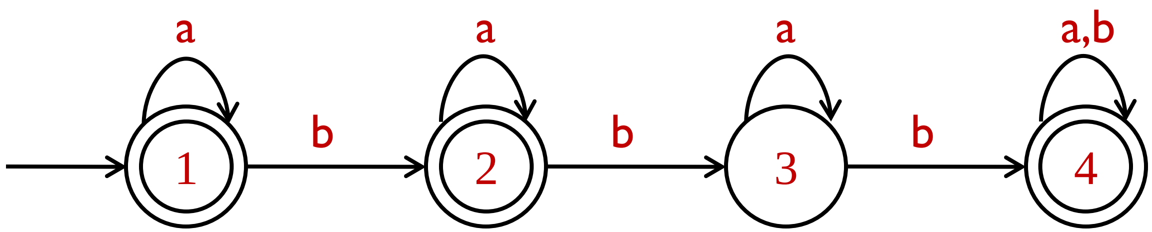
- Principes:
  - Chaque mot accepté correspond à un chemin de l'état initial vers l'état final
  - L'expression régulière obtenue est l'union de tout les chemins possibles
- Exemples:



- ER=a*ba*ba*

Automates et expressions régulières

- Exemples:



- Chemin1 : a*
- Chemin 2: a*ba*
- Chemin 3: a*ba*ba*b(ab)*

ER= a* | a*ba*| a*ba*ba*b(ab)*

Automates et expressions régulières

L'algorithme des équations linéaires selon le lemme d'Arden:

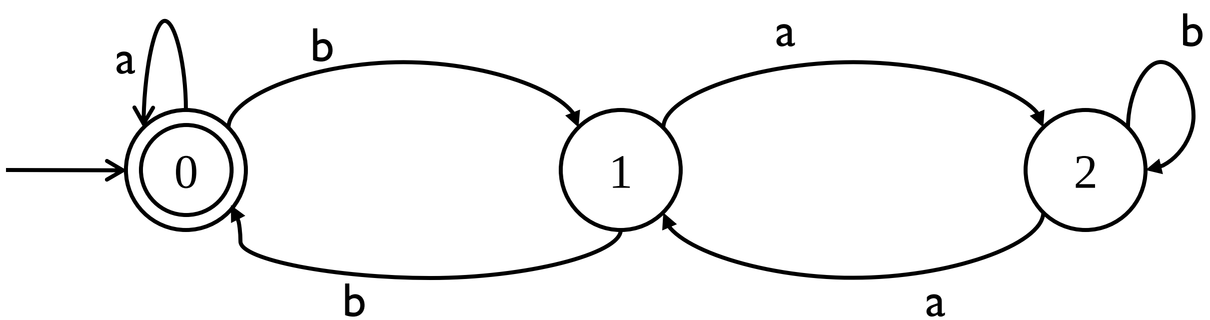
- soit A =(X, Q, q0, F, δ) un automate à états finis quelconque.
- On désigne par Li le langage accepté par l'automate si son état initial était qi.
- =>Trouver le langage accepté par l'automate revient à trouver L0 étant donné que l'analyse commence à partir de l'état initial q0.
- L'automate permet d'établir un système d'équations aux langages de la manière suivante:
  - Si δ(qi,a)=qj alors on écrit Li=a L j;
  - Si qi ϵ F, alors on écrit Li=ε
  - Si Li=a et Li = b alors on écrit: Li= a | b
- Il suffit de résoudre le système en précédant à des substitutions et en utilisant la règle du lemme d'Arden suivante:

  la solution de l'équation L= aL | b est le langage L=a*b

Automates et expressions régulières

- Exemple: Trouvons le langage accepté par cet automate:
- Le système d'équations est le suivant:

1. L0=aL0 | bL1 | ε
2. L1=bL0 | aL2
3. L2= aL1|bL2



Ensuite on applique le lemme d'Arden:

La solution de l'équation L=aL|β est le langage L=a* β

1. L0=aL0 | bL1 | ε
2. L1=bL0 | aL2
3. L2= aL1 |bL2

β

1. L0=aL0 | bL1 | ε
2. L1=bL0 | aL2
3. L2= b*aL1

1. L0=aL0 | bL1 | ε
2. L1=bL0 | ab*aL1
3. L2= b*aL1

1. L0=aL0 | bL1 | ε
2. L1=(ab*a)*bL0
3. L2= b*aL1

1. L0=aL0 | b(ab*a)*bL0 | ε= **(**a| b(ab*a)*b**)**L0| ε
2. L1=(ab*a)*bL0
3. L2=b*a(ab*a)*bL0

Finalement on applique le lemme de Arden sur L0=>**L0=(a| b(ab*a)*b)* ε**

=> **ER=(a|b(ab*a)*b)***

Passage de l'automate vers la grammaire

- Le principe de correspondance entre automates et grammaires régulières est très intuitif : il correspond à l'observation que chaque transition dans un automate produit exactement un symbole, de même que chaque dérivation dans une grammaire régulière *normalisée*.
- Soit A = (X,Q, q0, F, δ) un AEF, la grammaire qui génère le langage reconnu par A est G = (N,T, P, S) :
- On associe à chaque état de Q un non terminal. Ceci permet d'avoir autant de non terminaux qu'il existe d'états dans A;
- L'axiome S est le non-terminal associé à l'état initial q0 ;
- Soit A le non terminal associé à qi et B le non-terminal associé à qj, si δ(qi, a) = qj alors la grammaire possède la règle de production : A → aB ;
- Si qi est final et A est le non-terminal associé à qi alors la grammaire possède la règle de production : A → ε.

Passage de la grammaire vers l'automate

- soit $G = (N, T, P, S)$ une grammaire régulière à droite, si toutes les règles de production sont de la forme : $A \rightarrow aB$ ou $A \rightarrow B$ ($A, B \in N$, $a \in T \cup \{\varepsilon\}$) alors il suffit d'appliquer l'algorithme suivant :

1. Associer un état à chaque non terminal de N;
2. L'état initial est associé à l'axiome ;
3. Pour chaque règle de production de la forme $A \rightarrow \varepsilon$, l'état q_A est final ;
4. Pour chaque règle de production de la forme $A \rightarrow a$ ($a \in T$), alors créer un nouvel état final qf et une transition partant de l'état q_A vers l'état qf avec l'entrée a ;
5. Pour chaque règle $A \rightarrow aB$ alors créer une transition partant de q_A vers l'état q_B en utilisant l'entrée a ;
6. Pour chaque règle $A \rightarrow B$ alors créer une ε -transition partant de qA vers l'état qB ;

Passage de la grammaire vers l'automate

- Pour chaque règle de la forme $A \longrightarrow wB$ ($A, B \in N$ et $w \in T*$) tel que $|w| > 1$, créer un nouveau non-terminal B' et éclater la règle en deux : $A \longrightarrow aB'$ et $B' \longrightarrow uB$ tel que $w = au$ et $a \in T$;
- Pour chaque règle de la forme $A \longrightarrow w$ ($A \in N$ et $w \in T*$) tel que $|w| > 1$, créer un nouveau non-terminal B' et éclater la règle en deux : $A \longrightarrow aB'$ et $B' \longrightarrow u$ tel que $w = au$ et $a \in T$;
- Recommencer 1) et 2) jusqu'à ce qu'il n'y ait plus de nouveaux non terminaux ;

Passage de la grammaire vers l'automate

- Enfin, l'élimination des règles A → B (A, B ∈ N), on applique l'algorithme suivant:

1. Pour chaque règle A → B tel que B → b1b|2|…|bn
   - Rajouter une règle A → b1|b2|…|bn
   - Supprimer la règle A → B
2. Refaire l'étape 1) jusqu'à ce qu'il n'y ait plus de règles de la forme A → B

- **Exemple :**
  Soit la grammaire régulière G = ({S, T}, {a, b}, {S ->aabS|bT, T-> aS|bb}, S).
  Le processus de transformation est:
  - Éclater S → aabS en S → aS1 et S1 → abS ;
  - Éclater S1 → abS en S1 → aS2 et S2 → bS ;
  - Éclater T → bb en T1 → bT1 et T1 → b ;