Automates finis

- Automate fini déterministe (AFD)
- Automates finis non déterministes AFN
- Construction d'automates
- Langage d'un AFD
- Théorème de Kleene
- Équivalence d'automates – Minimisation

Construction d'automates

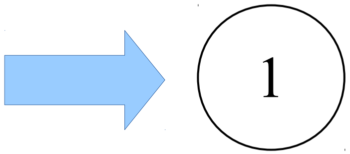
- Nous allons voir dans ce paragraphe comment les opérations sur les langages peuvent être traduites en termes d'automates.

  i.e. Si T est une opération entre langages et que l'on sait construire des automates reconnaissant L_1 et L_2. Comment en déduire un automate reconnaissant L_1 T L_2?

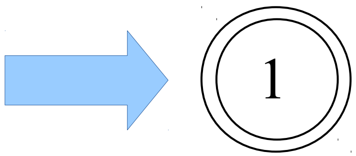
- Nous traiterons successivement le complémentaire, la somme, l'intersection, la différence, le produit et l'étoile d'un ou de deux langages.

  (Voir l'animation OpérationsEntreAFD)

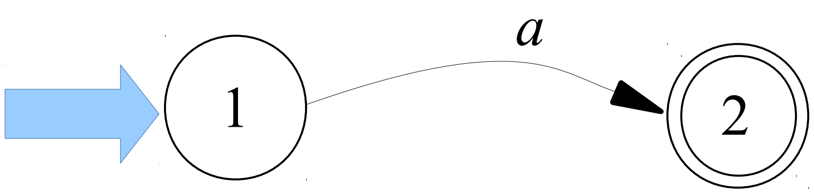
Langages élémentaires



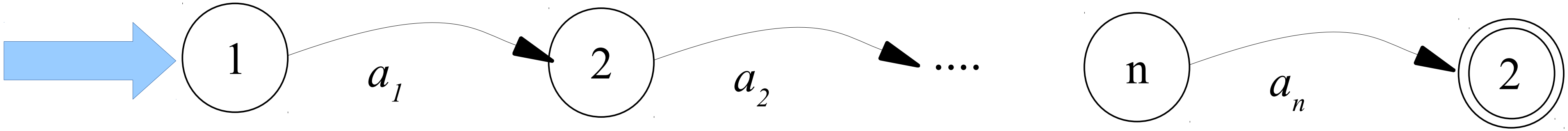
Reconnaît le langage vide



Reconnaît le langage $L=\{\varepsilon\}$



Reconnaît le langage $L=\{a\}$



Reconnaît $L=\{w=a_1..a_n\}$

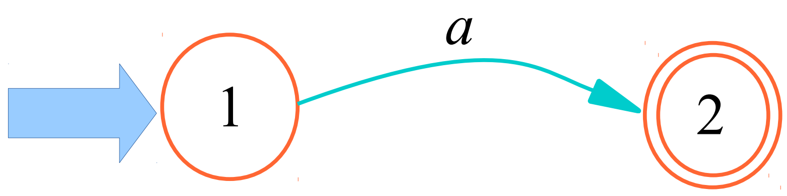
Complémentaire

C'est le cas le plus simple puisqu'il suffit d'inverser les états acceptants et refusant, à condition de bien prendre garde de partir d'un automate complet.

Proposition

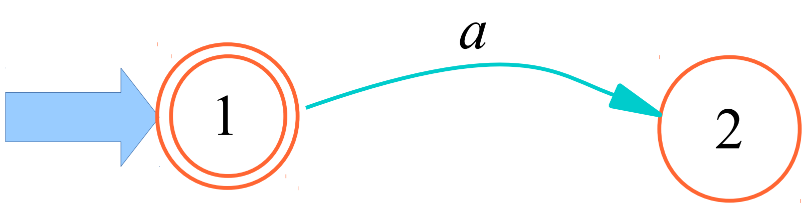
Si $\mathcal{A} = (Q, \Sigma, T, q_0, A)$ est un automate déterministe complet reconnaissant le langage L, alors $\mathcal{A}' = (Q, \Sigma, T, q_0, Q \backslash A)$ reconnaît le langage Σ^* complémentaire de L.

Attention au cas des automates non complets.



reconnaît L={a}

alors que



reconnaît L'={ε} qui n'est pas le complémentaire de L.

Somme d'automates

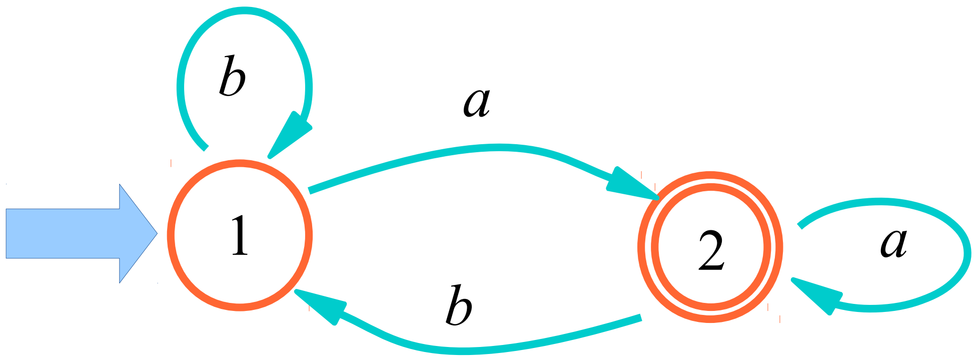
Proposition

Si $\mathcal{A}1$ reconnaît le langage L1 et que $\mathcal{A}2$ reconnaît le langage L2, alors l'union disjointe des deux automates obtenue en juxtaposant les deux automates après avoir renommé si nécessaire les états est un automate non déterministe reconnaissant le langage L=L1+L2.
On l'appelle automate somme des automates $\mathcal{A}1$ et $\mathcal{A}2$.

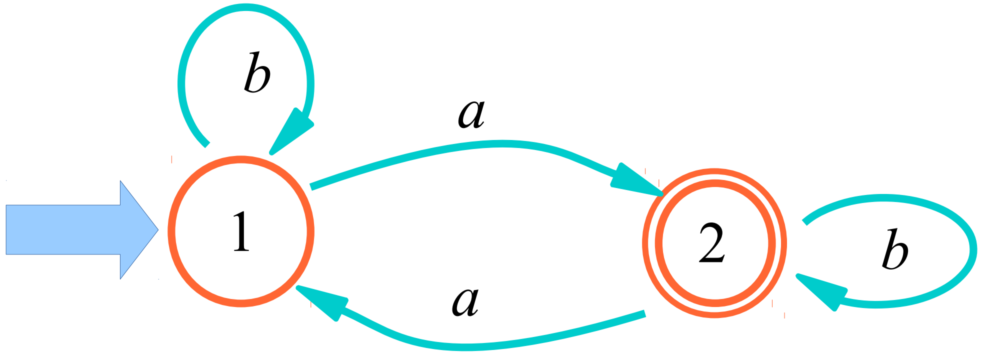
Cet automate est non déterministe puisqu'il possède deux états initiaux mais peut bien sûr être déterminisé...
Exemple :$\Sigma=\{a,b\}$.

L1={mots sur Σ* terminant par a}
L2={mots sur Σ* contenant un nombre pair de a}

L1 est reconnu par



et L2 par



Construire un automate reconnaissant L1+L2.

Proposition

On a vu que tout langage réduit à un mot est automatique.
Par somme, on en déduit que tout langage fini est automatique.

Intersection

Proposition

Si $\mathcal{A}1$ reconnaît le langage L1 et que $\mathcal{A}2$ reconnaît le langage L2, alors d'après les lois de de Morgan

$$\overline{L1 \cap L2} = \overline{L1} + \overline{L2}$$

d'où

$$L1 \cap L2 = \overline{\overline{L1} + \overline{L2}}$$

L'intersection des langages est donc reconnue par un automate construit par somme et complémentaire d'automates.

En pratique, on préfère souvent utiliser l'automate des couples :

Proposition

Si $\mathcal{A}1$= (Q1, Σ ,T1,q_0,A1) reconnaît le langage L1 et que
$\mathcal{A}2$ = (Q2, Σ ,T2,q'_0,A2) reconnaît le langage L2, alors
l'automate des couples :

$$\mathcal{A} = (Q1xQ2, \Sigma, T, (q_0, q'_0), A1xA2)$$

reconnaît le langage $L1 \cap L2$.

- Un état Q de cet automate est un couple $(q,q') \in Q1xQ2$
- L'état initial est le couple des états initiaux (q_0, q'_0)
- La fonction de transition T est définie par

$$\text{Si } q1=T1(q,l) \text{ et } q2=T2(q',l) \text{ alors } T((q,q'),l)=(q1,q2)$$

- Les états acceptants de sont les couples formés de deux états acceptants.

Si *A1* possède n états et *A2* possède p états, il y a n*p couples dans le produit cartésien. L'inconvénient de cette construction est donc d'engendrer un grand nombre d'états.

Les avantages sont qu'elle est très simple à mettre en œuvre et que l'automate obtenu est déterministe.

<u>Exercice</u> : Construire un automate reconnaissant le langage des mots finissant par a et possédant un nombre pair de a.

Différence

Proposition

Si $\mathcal{A}1$ reconnaît le langage L1 et que $\mathcal{A}2$ reconnaît le langage L2, alors on peut construire un automate reconnaissant la différence ensembliste des deux langages par intersection et complémentaire en utilisant :

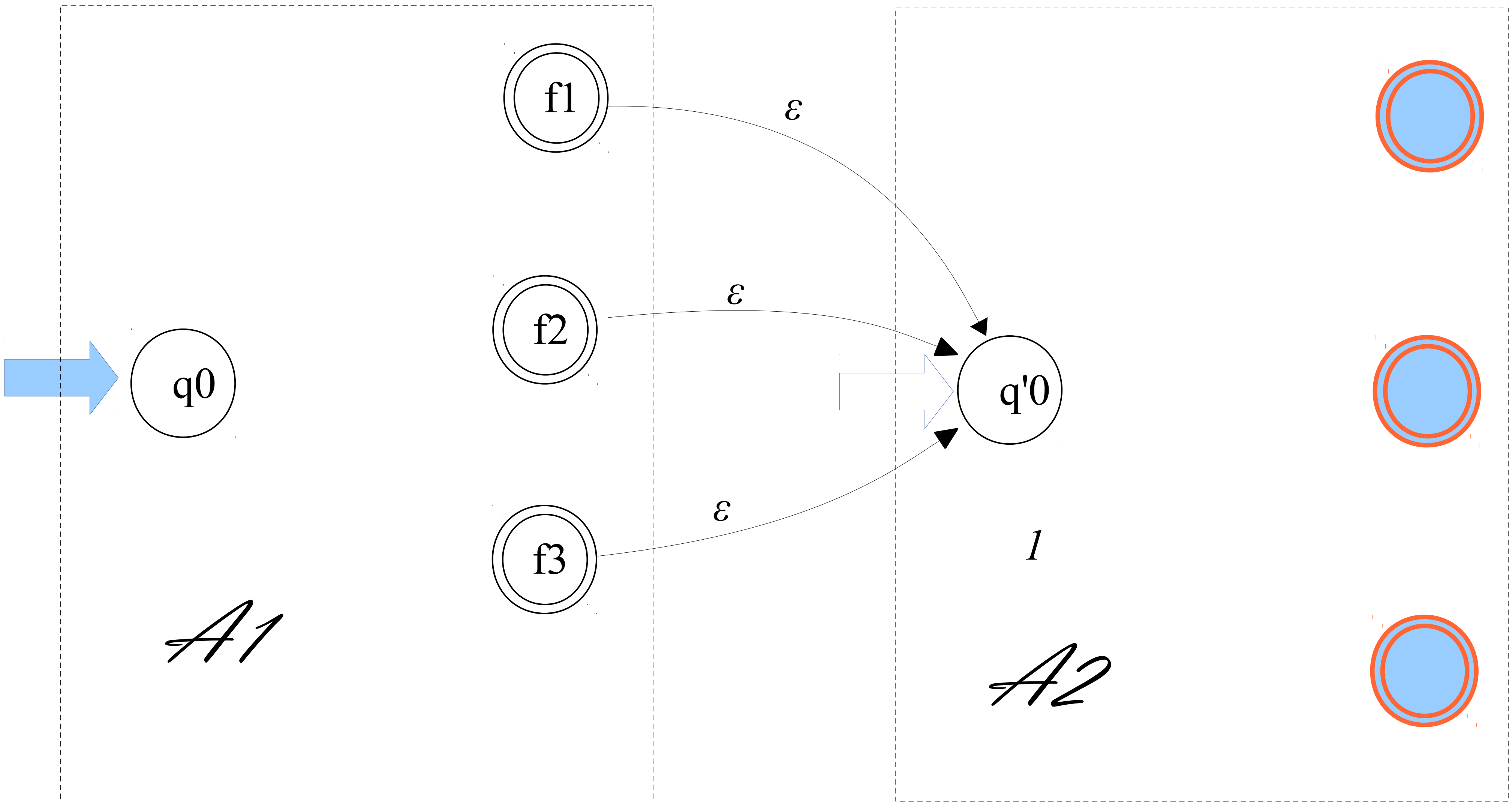
$$L1 \backslash L2 = L1 \cap \overline{L2}$$

Produit

Proposition

Si $\mathcal{A}1$= (Q1, Σ ,T1,q_0,A1) reconnaît le langage L1 et que $\mathcal{A}2$ = (Q2, Σ ,T2,q'_0,A2) reconnaît le langage L2, alors on construit un automate reconnaissant le produit L1.L2 de la manière suivante :

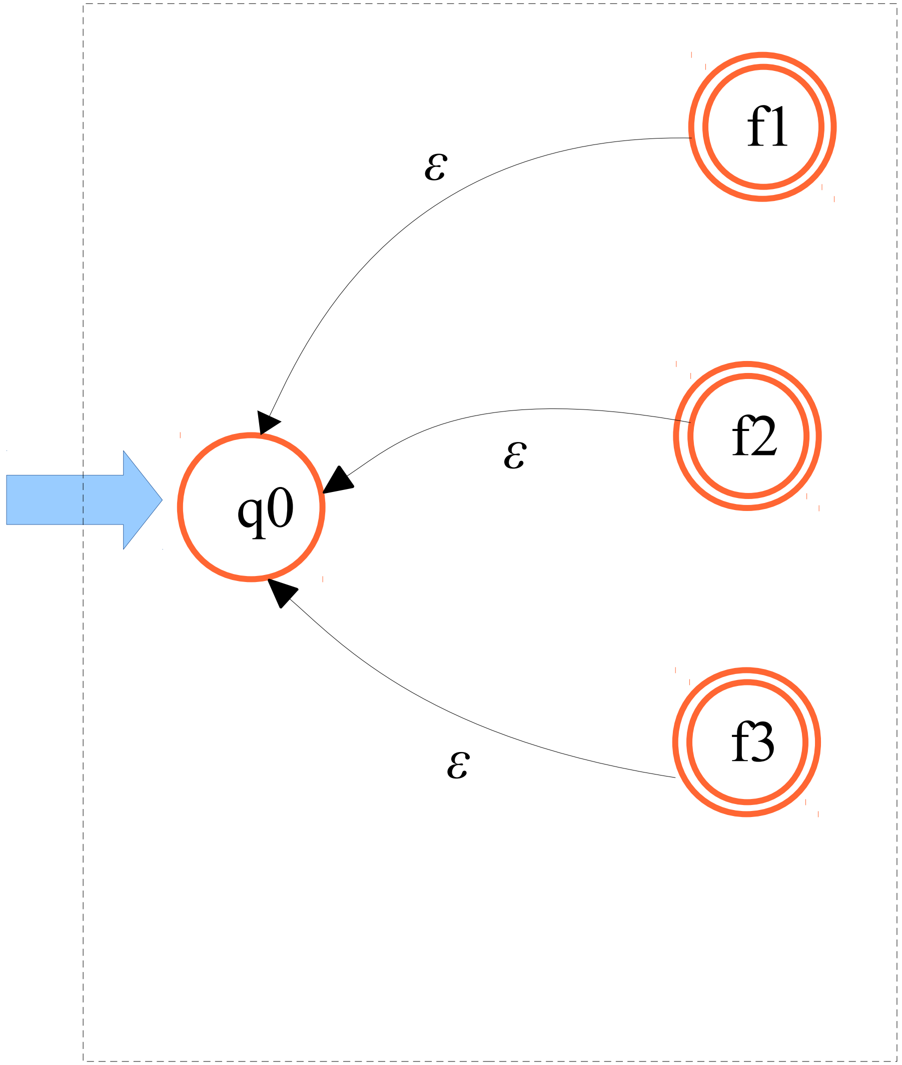
- $Q=Q1\cup Q2$. On effectue une union disjointe des états quitte à renommer les états de Q2.
- $I=\{q_0\}$, l'état initial de $\mathcal{A}1$
- A=A2, l'ensemble des états acceptants de $\mathcal{A}2$
- $T= T1\cup T2$, l'union de toutes les transition auxquelles on ajoute des ε–transitions de tous les états acceptants de $\mathcal{A}1$ vers l'état initial de $\mathcal{A}2$.


f1
f2
f3
q0
A1
ε
ε
ε
q'0
1
A2

Langages L^+

Proposition

Si $\mathcal{A}$= (Q1, Σ ,T1,q_0,A1) reconnaît le langage L alors on construit un automate reconnaissant le langage $L1^+$
en ajoutant à T des ε–transitions de tous les états acceptants de $\mathcal{A}$ vers l'état initial de $\mathcal{A}$.

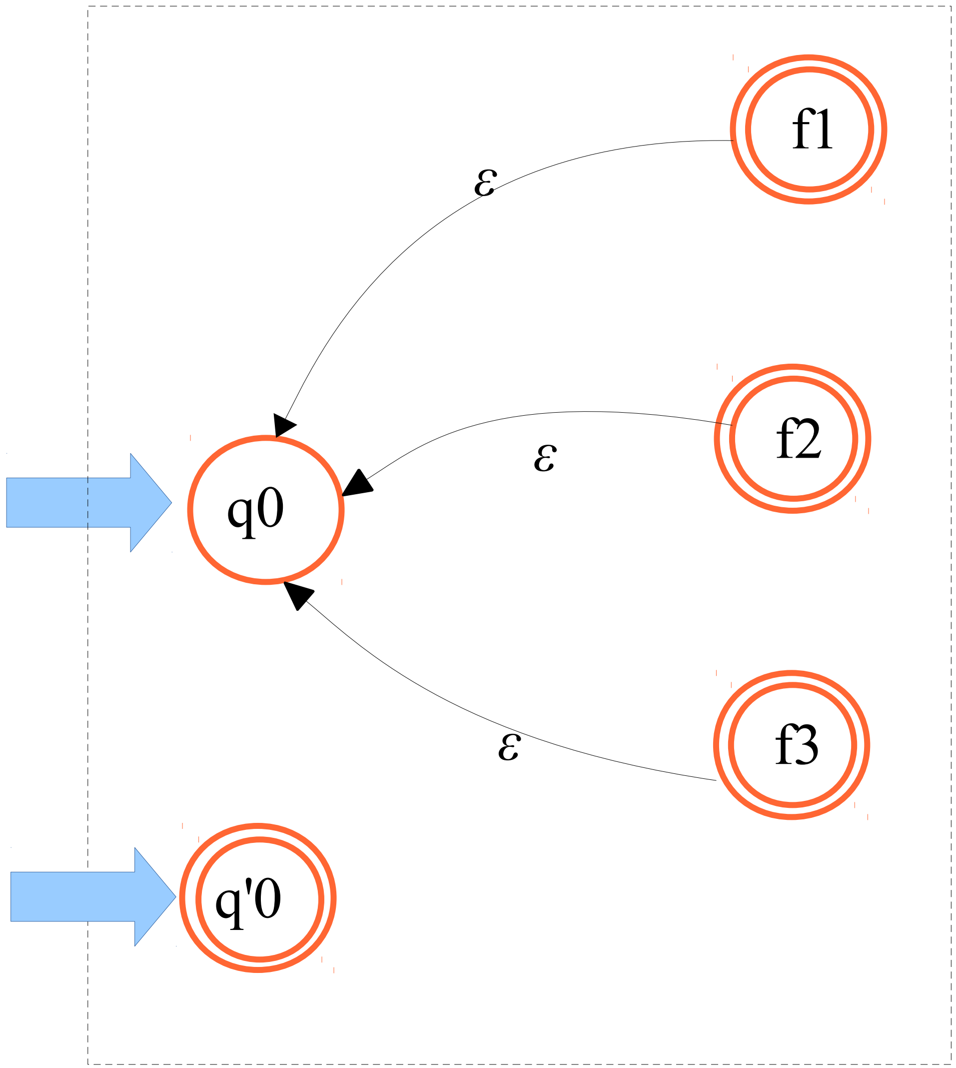

f1
ε
f2
ε
q0
ε
f3

Langage L*

Proposition

Si $\mathcal{A}$= (Q1, Σ ,T1,q_0,A1) reconnaît le langage L alors on construit un automate reconnaissant le langage L1*

- en ajoutant à un état initial acceptant (pour reconnaître ε)
- en ajoutant à T des ε–transitions de tous les états acceptants de $\mathcal{A}$ vers l'état initial de $\mathcal{A}$.


f1
ε
f2
ε
q0
ε
f3
q'0

Attention à la tentation de rendre acceptant l'état initial plutôt que d'ajouter un état initial acceptant...

<u>Exemple</u> : Construire un automate reconnaissant

L=(a*b)*

On peut alors en déduire une première implication

Théorème : de Kleene (1)

Les langages réguliers sont tous automatiques.

En effet On sait décrire à l'aide d'automates les langages réduits à une lettre et on sait effectuer des sommes, des produits et des étoiles d'automates.
Par définition on en déduit que tout langage régulier est automatique.

Langages automatiques
Conditions Nécessaires

Nous allons montrer ici que tout langage automatique vérifie nécessairement des propriétés de gonflement.

Par contraposée, tout langage ne vérifiant pas ces propriétés ne peut pas être automatique.

- Les conditions nécessaires que nous allons étudier dans ce paragraphes sont surtout utiles en pratique pour montrer qu'un langage n'est PAS automatique en montrant qu'il n'est pas gonflable.

Théorème de gonflage (faible)

Proposition

Si L est un langage automatique, alors au delà d'une certaine longueur N, tout mot w admet une décomposition en trois mots :

$$w=w_1.w_2.w_3 \text{ avec } w_2\neq\varepsilon$$

telle que pour tout entier k :

$$w_1.w_2^k.w_3 \in L$$

Exercice 1 :

- Montrer que $L1=\{a^p b^p , p\geq 1\}$ n'est pas un langage automatique.

Attention : Cette condition nécessaire n'est pas suffisante.

Exercice 2 :

- Montrer que $L2=\{w, |w|_a=|w|_b\}$ l'ensemble des mots qui ont autant de a que de b vérifie le théorème de gonflage faible mais n'est pas automatique.
  On pourra remarquer que $L1 = L2\cap(a^*b^*)$

Théorème de gonflage (fort)

Proposition

Si L est un langage automatique, alors il existe un tentier N, tel que tout mot w de longueur supérieure à N dont on a marqué N lettres admet une décomposition en trois mots :

$$w=w_1.w_2.w_3$$

telle que

- w_1 , w_2 et w_3 contiennent tous les trois au moins une lettre marquée
- et pour tout entier k :

$$w_1.w_2^k.w_3 \in L$$

Exercice 3 :

- Montrer que L2={w, $|w|_a=|w|_b$} l'ensemble des mots qui ont autant de a que de b ne vérifie pas le théorème de gonflage fort. On retrouve ainsi le fait qu'il n'est pas automatique.