Chapitre 3 : Les langages réguliers et les automates d'états finis

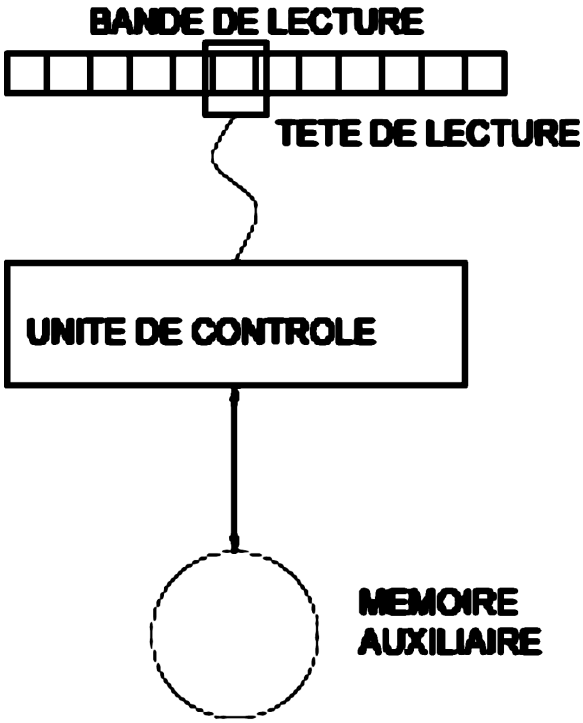
1. Reconnaisseurs

1.1 Définition

Les reconnaisseurs constituent une autre manière de décrire des langages. Un reconnaisseur est en fait la description d'une machine abstraite qui prend en entrée un mot et dit si ce mot appartient ou pas au langage décrit par la machine.

1.2 Composantes d'un reconnaisseur

Les différents éléments d'un reconnaisseur sont représentés graphiquement dans la figure 1.



Un reconnaisseur est composée de quatre parties :

1- Une bande de lecture : est une succession de cases, chaque case peut contenir un seul symbole d'un alphabet d'entrée. C'est dans les cases de cette bande de lecture qu'est écrit le mot à reconnaître.

2- Une tête de lecture : peut lire une case à un instant donné. La case sur laquelle se trouve la tête de lecture à un moment donnée s'appelle la case courante. La tête peut être déplacée par le reconnaisseur pour se positionner sur la case immédiatement à gauche ou à droite de la case courante.

3- Une mémoire : peut prendre des formes différentes. La mémoire permet de stocker des éléments d'un alphabet de mémoire.

4- Une unité de contrôle : constitue le cœur d'un reconnaisseur. Elle peut être vue comme un programme qui dicte au reconnaisseur son comportement.
   - Elle est représentée par un ensemble fini d'états ainsi que par une fonction de transition qui décrit le passage d'un état à un autre en fonction du contenu de la case courante de la bande de lecture et du contenu de la mémoire.
   - L'unité de contrôle décide aussi de la direction dans laquelle déplacer la tête de lecture et choisit quels symboles stocker dans la mémoire.
   - Parmi les états d'un reconnaisseur, on distingue des états initiaux, qui sont les états dans lesquels doit se trouver le reconnaisseur avant de commencer à reconnaître un mot et des états d'acceptation (finaux) qui sont les états dans lequel doit se trouver le reconnaisseur après avoir reconnu un mot.

L'état d'un reconnaisseur à un moment donné est décrit par sa configuration qui se compose de trois informations :

1. L'état de l'unité de contrôle
2. Le contenu de la bande de lecture et la position de la tête de lecture
3. Le contenu de la mémoire

1.3 Fonctionnement d'un reconnaisseur

- Un reconnaisseur fonctionne en effectuant une séquence de mouvements.
- Un mouvement est le passage d'une configuration du reconnaisseur à une autre. Le passage de la configuration C1 à la configuration C2 est représenté de la façon suivante : $C_1 \vdash C_2$. Une suite de k mouvements est notée $\overset{k}{\vdash}$. On définit les deux notations $\overset{*}{\vdash}$ et $\overset{+}{\vdash}$ pour k >= 0 et k > 0.
- Lors d'un mouvement, le reconnaisseur lit le symbole se trouvant dans la case courante, stocke de l'information dans sa mémoire et change d'état.
- Un reconnaisseur est dit déterministe si pour chaque configuration, il existe au plus un mouvement possible sinon, il est dit non déterministe.
- La configuration initiale d'un reconnaisseur est une configuration pour laquelle l'unité de contrôle est dans un état initial, la tête de lecture se trouve sur la case se trouvant la plus à gauche de la bande de lecture et la mémoire contient un symbole initial donnée.
- Une configuration d'acceptation est une configuration pour laquelle l'unité de contrôle se trouve dans un état d'acceptation, la tête de lecture se trouve sur la case la plus à droite et la mémoire se trouve dans un état d'acceptation.
- On dit qu'un reconnaisseur accepte un mot *m* si, ayant *m* sur sa bande de lecture, le reconnaisseur peut effectuer une

séquence de mouvements l'amenant de la configuration initiale à une configuration finale. Si le reconnaisseur est non déterministe, il peut exister plusieurs séquences de mouvements vérifiant ces conditions.

- Le langage reconnu par un reconnaisseur est l'ensemble des mots qu'il accepte.

1.3 Théorème

Les automates d'états finis sont des reconnaisseurs pour les langages réguliers.

2. Automates d'états finis

2.1 Définition

On appelle automate d'états finis tout cinq-uple $A(T, Q, q_0, \delta, F)$ est défini par :

- un alphabet T des symboles d'entrée,
- un ensemble fini Q d'états,
- un état $q_0 \in Q$ distingué comme étant **l'état initial**,
- un ensemble fini F, $F \in Q$ d'états distingués comme étant états finaux (ou états terminaux),
- $P(Q)$: l'ensemble des parties de Q.
- une **fonction de transitions** : $\delta : Q \times T \cup \{\varepsilon\} \rightarrow P(Q)$, qui associe à tout couple formé d'un état et d'un symbole de T∪ $\{\varepsilon\}$ fait correspondre un ensemble (éventuellement vide) d'états : $\delta(q_i, a) = \{q_{i1}, \ldots, q_{in}\}$.

Ex :

Soit l'automate défini par :

$T = \{a, b\}$, $Q = \{q_0, q_1, q_2, q_3\}$, $q_0 = q_0, F = \{q_3\}$,

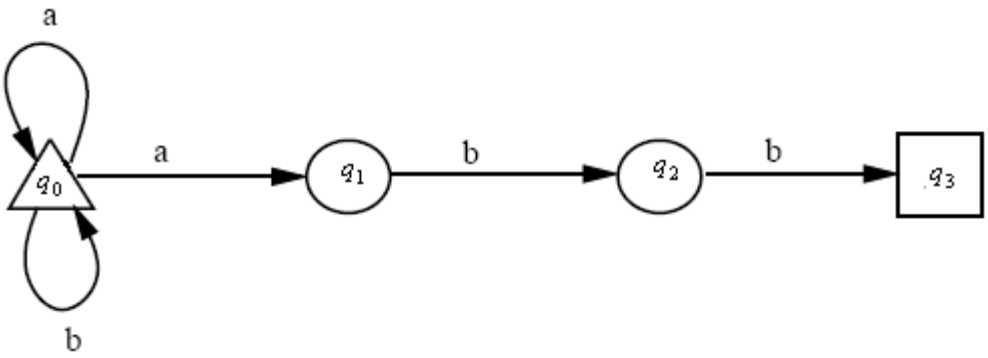
$\delta(q_0, a) = \{q_0, q_1\}, \delta(q_0, b) = q_0, \delta(q_1, b) = q_2, \delta(q_2, b) = q_3$.

2.2 Représentation graphique

Les automates d'états finis sont souvent représentés sous la forme d'un graphe dont les sommets sont les états de l'automate et les arcs (étiquetées par les symboles de *T*) correspondent à la fonction de transition δ : il existe un arc étiqueté par x entre les sommets q_i et q_j si et seulement si: $\delta(q_i, x) = q_j$. On dessinera un triangle pour l'état initial et un carré pour les états finaux.

Ex : le même exemple précédant

La représentation graphique est :



2.3 Table de transition ou représentation tabulaire d'un AEF

On peut aussi représenter l'automate *A(T, Q,*q_0*,*δ*, F)* sous forme d'un tableau Tab indicé par Q et T tel que $q_j \in Tab[q_i, a]$ si et seulement si : $\delta(q_i, x) = q_j$. On distingue l'état initial et les états finaux par des flèches dirigées respectivement vers la droite et vers la gauche,

Ex : le même ex :

	état	*a*	*b*
→	0	0,1	0
	1	-	2
	2	-	3
←	3	-	-

2.4 Langage reconnu par un AEF

Un mot m est reconnu par l'automate s'il existe une suite de mouvements menant de la configuration $\delta(q_0, m)$ à $\delta(q, \varepsilon)$ avec: $q \in F$.
Le langage reconnu par un automate A, noté L(A), est l'ensemble des mots reconnus par ce dernier.

$$L(A) = \{m \in \Sigma^* | (q_0, m) \overset{*}{\vdash} (q, \varepsilon) \text{ avec } q \in F\}$$

3. AEF déterministe et AEF non déterministe

3.1 AEF déterministe

Définition1 : on appelle ε -transition, une transition par le symbole ε entre deux états.

Définition2 : un AEF est dit déterministe lorsqu'il ne possède pas de ε -transition et lorsque pour chaque état q et pour chaque symbole a il y au plus un arc étiqueté a qui quitte q.

Ex : soit le langage régulier L=$\{0^n\ 1^m \quad / \quad n \geq 0,\ m \succ 0\}$. La représentation tabulaire de l'AEFD A est la suivante :

Etat	0	1
$\rightarrow q_0$	q_0	q_1
$\leftarrow q_1$	-	q_1

Tel que L=L(A).

3.2 AEF non déterministe

a) Définition : un A est dit non déterministe si et seulement s'il existe au moins une paire (q, a)$\in Q\times T\cup\{\varepsilon\}$ pour laquelle la fonction δ associe au moins deux états.
Un AEFD est un cas particulier d'un AEFND.

Ex :
Soit le langage régulier L=$\{w0 \ / \ w \in \{0,1\}^*\}$. La représentation tabulaire de l'AEFND A est la suivante :

Etat	0	1
$\rightarrow q_0$	q_0,q_1	q_0
$\leftarrow q_1$	-	-

Tel que L=L(A).

b)Remarque :
Deux AEF A1 et A2 sont équivalents si et seulement s'ils acceptent le même langage : L(A1)=L(A2).

c)Théorème : A tout AEFND correspond un AEFD.

3.2.1 Détermination d'un AEF (ou le passage d'un AEFND vers un AEFD)

But : La complexité de la reconnaissance d'un mot (ex : analyse lexicale) par un AEFND est exponentielle. De ce fait cela influe directement sur le temps de reconnaissance d'un mot. C'est pour ça il est préférable de transformer l'AEFND à un AEFD.

1- Elimination des ε -transitions

Etant donné un AEFND $A(T, Q, q_0, \delta, F)$, on lui associe un AEFND $A'(T', Q', q_0', \delta', F')$ sans ε -transitions en procédant comme suit :

- Tant qu'il existe au moins une ε -transition, on applique la procédure suivante :
- On choisit un état q_k dans lequel aboutit au moins une ε -transition.
  - pour tout $q_i \in Q - \{q_k\}$ tel que : $\delta(q_i, \varepsilon) = q_k$ et pour tous $q_j \in Q$ et $x \in T$ tels que $\delta(q_k, x) = q_j$, on ajoute une transition $\delta(q_i, x) = q_j$.
  - on supprime les transitions $\delta(q_i, \varepsilon) = q_k$
  - si $q_k \in F$, on ajoute $q_i \in F$,
  - si q_k n'est plus accessible, on le supprime.

Ex : soit l'AEFND suivant :

Etat/T	a	b	c	ε
$\rightarrow q_0$	q_0	-	-	q_1
q_1	-	q_1	-	q_2
$\leftarrow q_2$	-	-	q_2	-

En appliquant l'algorithme, on obtient un AEF sans ε -transitions

Etat/T	a	b	C
$\leftarrow$ $\rightarrow q_0$	q_0	q_1	q_2
$\leftarrow q_1$	-	q_1	q_2
$\leftarrow q_2$	-	-	q_2

2- Détermination d'un AEFND qui ne contient pas d'ε - transitions

Principe : considérer des ensembles d'états plutôt que des états.

1- Partir de l'état initial : $Q = \{q_0\}$,
2- Construire $Q^{(1)}$ l'ensemble des états obtenus à partir de Q par la transition x : $Q^{(1)} = \delta(Q, x)$.
3- Répéter 2 pour toutes les transitions possibles et pour chaque nouvel ensemble d'état $Q^{(i)}$
4- Tous les ensembles d'états $Q^{(i)}$ contenant au moins un état terminal deviennent terminaux (finaux),
5- Renommer alors les ensembles d'états entant que simples états.

Ou plus formellement :
Soit $A(T, Q, q_0, \delta, F)$ *un AEFND* $\Leftrightarrow$ $A'(T', Q', q_0', \delta', F')$ *un AEFD. Sa définition est :*

$T' = T$; $Q' = P(Q)$; $q_0' = q_0$;

$$\delta' : P(Q) \times T \to P(Q)$$
$$(\lambda, x) \mapsto \{\delta(q, x) \ / \ \delta(q, x) \in Q, \ \forall q \in \lambda\}$$
$$F' = \{q \in P(Q) \ / \ q \cap F \neq \varphi\}$$

Ex :
Soit l'AEFND suivante : on note q_i par : i.

état	a	b
→ 0	0,2	1
1	3	0,2
← 2	3,4	2
← 3	2	1
4	-	3

En appliquant l'algorithme de détermination :

"état"	a	b
0	0,2	1
0,2	0,2,3,4	1,2
1	3	0,2
0,2,3,4	0,2,3,4	1,2,3
1,2	3,4	0,2
3	2	1
1,2,3	2,3,4	0,1,2
3,4	2	1,3
2	3,4	2
2,3,4	2,3,4	1,2,3
0,1,2	0,2,3,4	0,1,2
1,3	2,3	0,1,2
2,3	2,3,4	1,2

=

état	a	b
→ 0	1	2
← 1	3	4
2	5	1
← 3	3	6
← 4	7	1
← 5	8	2
← 6	9	10
← 7	8	11
← 8	7	8
← 9	9	6
← 10	3	10
← 11	12	10
← 12	9	4

4. Grammaire et automate

a)Théorème : Tout langage régulier est engendré par une grammaire régulière et réciproquement. Autrement dit, un langage reconnu par un AEF est engendré par une grammaire de type 3 (régulière).

$$L\ est\ regulier\ ssi\ \exists\ A\ /\ L = L(A) \Leftrightarrow \exists\ G\ /\ L = L(G)$$

4.1 Passage d'un AEF à une grammaire régulière

Soit L un langage régulier, donc il existe un automate $A(T, Q, q_0, \delta, F)$ tel que L=L(A). Il s'agit de déterminer une grammaire régulière à droite G(T, N, S, P) telle que L=L(G).

- T=T ; - N=Q ; - S= q_0,
- Nous définissons P de la façon suivante, avec : $q_i, q_j \in Q$ et $x \in T$.
  - Si $\delta(q_i, x) = q_j$ Alors : $(q_i \rightarrow xq_j) \in P$,
  - Si $\delta(q_i, x) \in F$ Alors : $(q_i \rightarrow x) \in P$,
  - Si $\varepsilon \in L(A)$ alors :$\varepsilon \in L(G)$, ajouter $S \rightarrow \varepsilon$ à P

Ex : Soit l'AEF A suivant :

Etat	0	1
$\rightarrow q_0$	q_0,q_1	-
q_1	-	q_1,q_2
$\leftarrow q_2$	q_2	-

En appliquant l'algo précédant, on obtient :
G({0, 1},{S, A, B},S,{ S→0S/ 0A ; A→1A/1B/1 ; B→0B/0}), tel que L(G)=L(A) .

4.2 Passage d'une grammaire régulière à un AEF

Soit L un langage engendré par une grammaire régulière stricte à droite G(T, N, S, P). Il s'agit de déterminer un AEF *A(T, Q,q_0,δ, F)* tel que L=L(A). Il suffit de définir les 5 paramètres T, Q, q_0, δ, *F :*

- T=T ; $Q = N \cup F$; q_0=S
- Nous définissons δ de la façon suivante :
  - Si $(A \rightarrow xB) \in P$ Alors : $\delta(A, x) = B$
  - Si $(A \rightarrow x) \in P$ Alors : $\delta(A, x) = R$ avec: $R \in F$
  - $F = \{R \ / \ \delta(A, x) = R\}$
  - Si $\varepsilon \in L(G)$ alors $q_0 \in F$

Ex : Soit la grammaire G({0, 1},{S, A, B},S,P) tel que :
S→0S/ 0A ; A→1A/1B/1 ; B→0B/0
En appliquant l'algorithme du passage pour déterminer *A(T, Q,q_0,δ, F)* tel que L(G)=L(A).

T={0,1} ; $Q = \{S, A, B\} \cup F$; q_0=*S,* la fonction de transition δest :$\delta(S, 0) = S$; $\delta(S, 0) = A$; $\delta(A, 1) = A$; $\delta(A, 1) = B, \delta(A, 1) = R1$; $\delta(B, 0) = B$; $\delta(B, 0) = R2$;
F={R1, R2}.
Par conséquent l'AEF est la suivant :

Etat	0	1
$\rightarrow q_0$	q_0,q_1	-
q_1	-	q_1,q_2, q_{R1}
q_2	q_2, q_{R2}	-
$\leftarrow q_{R1}$	-	-
$\leftarrow q_{R2}$	-	-

Tel que : q_0=*S,* q_1=*A,* q_2=*B*

6. Minimisation d'un AEF

But: obtenir un AEF ayant le minimum d'états possible. En effet, certains états peuvent être équivalents. L'automate minimal qui reconnaît un langage donné est unique, à la numérotation des états près. L'unicité de l'automate minimal est une propriété importante car elle permet de vérifier que deux automates reconnaissent le même langage. En effet, si l'on dispose de deux automates déterministes, il suffit de construire les automates minimaux leur correspondant. Si ces derniers sont égaux (au nom des états près) alors les deux automates initiaux reconnaissent bien le même langage.

Principe: on définit des classes d'équivalence d'états par raffinement successifs. Chaque classe d'équivalence obtenue forme un seul même état du nouvel automate.

Algorithme de construction des $\{\beta_k\}$.

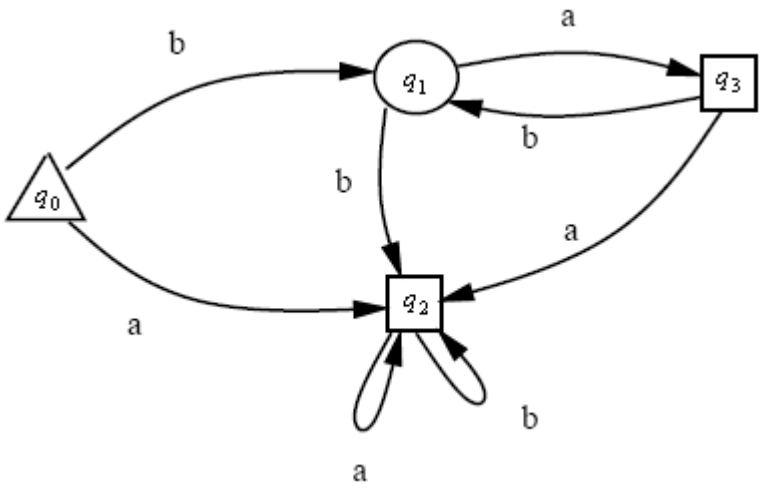
Entrée : $A(T, Q, q_0, \delta, F)$, AEFD sans états inaccessibles
Sortie : $A'(T', Q', q_0', \delta', F')$ minimal.
1) $p\beta_0 q \Leftrightarrow \{(p \in F \text{ et } q \in F) \text{ ou } (p \notin F \text{ et } q \notin F)\}$,
2) Si $\{p\beta_k q \text{ et } (\forall x \in T,\ \delta(p,x)\ \beta_k\ \delta(q,x))\}$ Alors $p\ \beta_{k+1} q$
3) Arrêt quand $\beta_k = \beta_{k+1}$

Les paramètres de l'automate minimal sont, alors, les suivants :
– Chaque classe est un état de l'automate minimal ;
– La classe qui contient l'ancien état initial devient l'état initial de l'automate minimal ;
– Toute classe contenant un état final devient un état final ;
– La fonction de transition est définie comme suit : soient A une classe obtenue, a un symbole de l'alphabet et *qi* un état *qi* ∈ A tel que *δ(qi, a)* est définie. La transition *δ(A, a)* est égale à la classe B qui contient l'état *qj* tel que *δ(qi, a) = qj* .

état	a	b
→ 0	1	2
← 1	3	4
2	5	1
← 3	3	6
← 4	7	1
← 5	8	2
← 6	9	10
← 7	8	11
← 8	7	8
← 9	9	6
← 10	3	10
← 11	12	10
← 12	9	4

Après l'application de l'algo de minimisation, on obtient l'AEFD minimal suivant :

Etat	A	b
$\rightarrow q_0$	q_2	q_1
q_1	q_3	q_2
$\leftarrow q_2$	q_2	q_2
$\leftarrow q_3$	q_2	q_1



7. AEF complet

Définition : un AEF $A(T, Q, q_0, \delta, F)$ est complet si et seulement si (A est déterministe et $\forall q_i \in Q, \forall x \in T \Rightarrow \exists q_j \in Q$ tel que: $\delta(q_i, x) = q_j$).

Ex : l'AEF suivant est complet.

Etat	0	1
$\rightarrow q_0$	q_0	q_1
q_1	q_1	q_2
$\leftarrow q_2$	q_2	q_2

8. Standardisation d'un AEF

Définition : Si un AEF n'est pas standard (contient plusieurs états initiaux) alors il existe AEF standard équivalent contenant un seul état initial.

Ex : soit l'AEF non standard A suivante :

Etat	0	1
$\rightarrow q_0$	q_0	q_1
$\rightarrow q_1$	q_1	q_2
$\leftarrow q_2$	q_2	q_2

L'AEF standard A' correspond à A est le suivant :

Etat	0	1	ε
$\rightarrow q_0'$	-	-	q_0,q_1
q_0	q_0	q_1	-
q_1	q_1	q_2	-
$\leftarrow q_2$	q_2	q_2	-

Ensuite on applique l'algo qui permet d'éliminer les ε -transitions.

9. Les expressions régulières

a) Définition

Les expressions régulières sur un alphabet T et les langages qu'elles décrivent sont définis récursivement de la manière suivante :

- ε est une expression régulière (ER) qui décrit le langage $\{ \varepsilon \}$,
- Si $a \in T$ alors a est une ER qui décrit le langage $\{ a \}$,
- Si r est une ER qui décrit le langage R, alors r^* est une ER qui décrit le langage R^* ,
- Si r est une ER qui décrit le langage R, alors r^+ est une ER qui décrit le langage R^+ ,
- Si r et s sont des ER qui décrivent respectivement les langages R et S alors : $r+s$ et rs sont des ER qui décrivent respectivement les langages $R \cup S$ et RS.
- il n'y a pas d'autres ER

Remarque : on conviendra des priorités décroissantes suivantes : *, concaténation, +

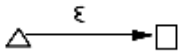
Ex :
$a+b^*c$: est soit le mot *a* ou soit les mots formés de 0 ou +ieurs *b* suivi d'un *c,*
$(a+b)^*abb(a+b)^*$: dénote l'ensemble des mots sur {a,b} ayant le facteur abb.

b) Théorème de Kleene : Un langage est reconnaissable (reconnu par un automate d'états finis) si et seulement s'il est régulier (dénoté par une expression régulière). En d'autres termes, il est possible de construire un automate à partir de n'importe quelle expression régulière et, réciproquement, d'associer une expression régulière à tout automate.

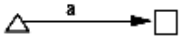
c) Construction d'AEF à partir d'une ER
Méthode : basée sur la récursivité des expressions régulières

Pour une expression régulière s, on note $A(s)$ un automate reconnaissant cette expression.

– automate acceptant la chaîne vide

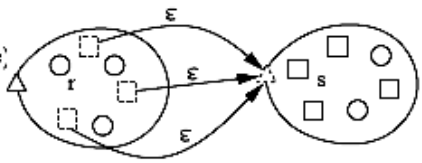


– automate acceptant la lettre a



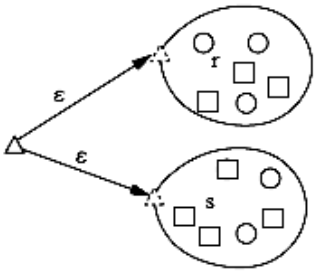
– automate acceptant $(r)(s)$

1. mettre une ε-transition de chaque état terminal de $A(r)$ vers l'état initial de $A(s)$
2. les états terminaux de $A(r)$ ne sont plus terminaux
3. le nouvel état initial est celui de $A(r)$
4. (l'ancien état initial de $A(s)$ n'est plus état initial)



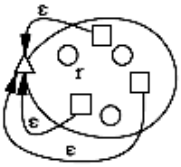
– automate reconnaissant $r + s$

1. créer un nouvel état initial q
2. mettre une ε-transition de q vers les états initiaux de $A(r)$ et $A(s)$
3. (les états initiaux de $A(r)$ et $A(s)$ ne sont plus états initiaux)



– automate reconnaissant r^+

mettre des ε-transition de chaque état terminal de $A(r)$ vers son état initial



Ex :…

10. Quelques propriétés des langages reconnaissables

- Pour montrer qu'un langage est régulier, on peut au choix trouver un AEF pour ce langage ou bien encore une expression régulière.

- Pour montrer qu'un langage n'est pas régulier, il est utile de disposer de la propriété qui est connue sous le nom de lemme de pompage. Intuitivement, cette propriété pose des limitations intrinsèques concernant la diversité des mots appartenant à un langage régulier infini : au-delà d'une

certaine longueur, les mots d'un langage régulier sont en fait construits par itération de motifs apparaissant dans des mots plus courts.

10.1 Lemme de pompage

a) Théorème (Lemme de pompage) Soit L un langage régulier infini. Il existe un entier k tel que pour tout mot *x* de L plus long que k, *x* se factorise en $x = uvw$, avec :

1- |v | >= 1 ,
2- |uv| <= k et
3- pour tout i >= 0, uv^iw est également un mot de L.

Si un langage est infini, alors il contient des mots de longueur arbitrairement grande. Ce que dit ce lemme, c'est essentiellement que dès qu'un mot de L est assez long, il contient un facteur différent de "qui peut être itéré à volonté tout en restant dans L. En d'autres termes, les mots « longs» de L sont construits par répétition d'un facteur s'insérant à l'intérieur de mots plus courts.

b) Preuve : Soit A un AEF de k états reconnaissant L et soit *x* un mot de L, de longueur supérieure ou égale à k. La reconnaissance de *x* dans A correspond à un calcul $q_0, \ldots, q_n$ impliquant | x | + 1 états. A n'ayant que k états, le préfixe de longueur $k + 1$ de cette séquence contient nécessairement deux fois le même état q, aux indices i et j, avec 0 <= i < j<= k. Si l'on note *u* le préfixe de *x* tel que $\delta^*(q_0, u) = q$ et *v* le facteur tel que $\delta^*(q, v) = q$, alors on a bien

(1) car au moins un symbole est consommé le long du cycle $q \ldots q$;
(2) car j<= k ;
(3) en court-circuitant ou en itérant les parcours le long du circuit $q \ldots q$.

c) Remarque importante : le lemme est aussi vérifié pour de nombreux langages non-régulier : il n'exprime donc qu'une

condition nécessaire (mais pas suffisante) pour qu'un langage soit régulier.

d) Application du lemme de pompage : Montrer que certains langages ne sont pas réguliers, Le lemme de pompage est insuffisant pour montrer que d'autres langages ne sont pas réguliers (par exemple les mots ayant un nombre différent de a et de b).

Références et sources :

1- Pierre Berlioux, Mnacho Echenim et Michel Lévy, polycope , Théorie des langages ,Année 2017-2018
2- LIF15 – Théorie des langages formels Sylvain Brandel 2015-2016 sylvain.brandel@univ-lyon1.fr
3- Théorie des langages Notes de Cours François Yvon et Akim Demaille avec la participation de Pierre Senellart 19 Janvier 2015
4- Myriam Noureddine, théorie des langages, OPU