Université de Batna

SUPPORT DE COURS

MODULE : THEORIE DES LANGAGES

Département : INFORMATIQUE

Niveau : 2ème année LICENCE

Version janvier 2016

Chapitre N° 1 : Rappel mathématique

Introduction : La théorie des langages définit les langages de programmation par contre la compilation transforme les programmes écrits dans ces langages en langage machine.
La structure de base de la théorie des langages est le monoïde, mais avant de définir cette nation on a besoin d'avoir un rappel mathématique.

1. Ensemble et relation :

a) l'ensemble : un ensemble est une collection d'objets appelés éléments.

Exemple : E = {1,2 ,3} est un ensemble de 3 éléments. Ce nombre là est appelé la cardinal de l'ensemble E

Remarques :

- L'ensemble vide noté par { } ou par ϕ, c'est un ensemble dont le cardinal = 0
- Soit E un ensemble. L'ensemble des parties de E noté P(E) contient tous les sous ensembles de E
  Exemple : P({1,2,3})={ϕ ,{1},{2},{3} ,{1,2} ,{1,3} ,{2,3} , {1,2,3} }

Opérations sur les ensembles : soient E et F deux ensembles

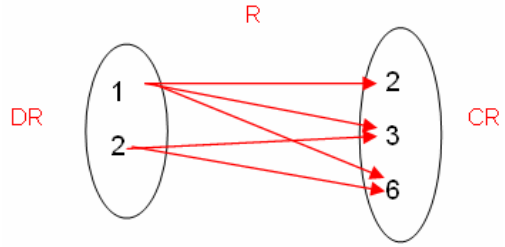
- L'inclusion : $E \subset F \Leftrightarrow \forall x\ (x \in E \Rightarrow x \in F)$
- L'union (U) (+) : $E \cup F = \{x / x \in E \text{ ou } x \in F\}$
- L'intersection : $E \cap F = \{x / x \in E \text{ et } x \in F\}$
- Le complémentaire de F dans E (avec $F \subset E$) = $\{x / x \in E \text{ et } x \notin F\}$
- Le produit cartésien : $EXF = \{(x, y) / x \in E \text{ et } y \in F\}$

b) la relation : Soient E et F deux ensembles

- Une relation R de E dans F est un sous ensemble des couples (x,y) du E x F tel que x R y.
- Une relation R sur un ensemble E est un sous ensemble des couples (x,y) du produit cartésien E x E tel que x R y.

$ExF = \{(1,2),(1,3),(1,6),(2,2),(2,3),(2,6),(0,2),(0,3),(0,6)\}$

Soit R est définie comme suit : x est strictement inférieur à y et x ≠ 0. Le sous ensemble du produit cartésien E x F qui vérifie cette relation est $\{(1,2),(1,3),(1,6),(2,3)(2,6)\}$

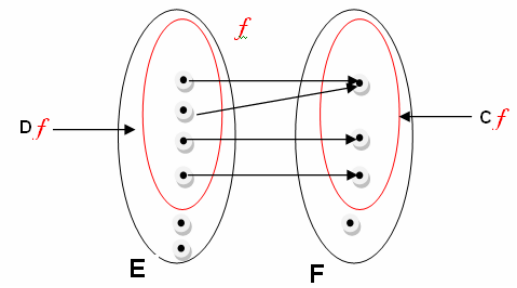


L'ensemble des antécédents de R est appelé le Domaine de R. On écrit : $DR = \{(x /(x, y) \in R\}$

L'ensemble des images de R est appelé le Codomaine de R. On écrit : CR= $\{(y/(x,y)\in R\}$

c) la fonction : Soient E et F deux ensembles

- Une fonction f de E dans F est une relation particulière telle que chaque antécédent de cette relation a exactement une seule image.



On peut dire que chaque élément de E a au plus une image avec la fonction. Et avec la relation, chaque élément peut avoir 0,1 ou plusieurs images.

d) l'application : une application de E dans F est une fonction f telle que le domaine de f = E

Exemple : $f : x \longrightarrow \frac{1}{x}$ est une fonction de IR dans IR mais n'est pas une application puisque l'élément « 0 » n'a pas une image.

2. lois de composition internes

Définition : une loi de composition interne sur un ensemble E est une application qui associe à chaque couple (x,y)$\in$ ExE un élément appartient aussi à E . Si on note cette loi par *

On écrit : * : ExE $\longrightarrow$ E

* est commutative si : $\forall x, y \in E$: x*y=y*x

* est association si : $\forall x, y, z \in E : (x * y) * z = x * (y * z) = x * y * z$

* a un élément neutre «e » si : $\forall x \in E \quad x * e = e * x = x$

3. le monoïde :

Un monoïde E est un ensemble muni d'une loi de composition interne * associative et possédant un élément neutre « e »

On écrit : $< E, *, e >$

Exemple :

$< IN, +, 0 >$ 0 : l'élément neutre pour +

$< IN, \times, 1 >$ 1 : l'élément neutre pour x

4. le morphisme :

Soient E et F deux ensembles munis respectivement des lois de composition internes * et .

- On dit que la fonction f est un morphisme de E dans F si et seulement si

$$f(x * y) = f(x).f(y)$$

Soient $< E, *, e >$ et $< F, ., e` >$ deux Monoïdes ou « e », « e` » sont les éléments neutres respectivement pour * et .

- On dit que f est un morphisme de monoïde de E dans F si et seulement si :
  - f est un morphisme
  - f (e) = e`

Chapitre N° 2 : Les langages

1. Notion de l'alphabet :

Un alphabet noté X est un ensemble fini non vide des éléments appelés les **symboles** ou **lettres.**

Exemples :

L'alphabet binaire : X = {0,1}.
L'alphabet du langage PASCAL : X = {begin, if, then, else,….end}.
L'alphabet de la langue française : X= {a, b,…, A, B, …, é, è,.. ï}.
X= {bonjour, ali, $, 1, w} c'est un alphabet de 5 lettres

2. Notion de mot (chaîne)

Un mot sur un alphabet X est une suite d'éléments de l'alphabet X. Cette suite est finie et ordonnée.

Exemples :

L'ensemble des mots construits sur X= {0,1} est {0,1,01,10,111,00000,……}.
L'ensemble des mots construits sur L'alphabet PASCAL est {tous les programmes (corrects, incorrects) écrits en PASCAL}.
bonjourali : c'est un mot de l'alphabet X= {bonjour, ali, $, 1, w}

Remarques :

☞ L'ensemble des mots construits sur un alphabet X est un ensemble infini noté X^*

☞ Le mot vide, noté par ε , c'est un mot qui ne contient aucun élément. $\forall X : \varepsilon \in X^*$

Exemple : soit X= {a,b}

X^* = { ε , a, b, aa, ab, bb, ba, aaa, aab, aba, baa,….}.
X^* = { ε } + {a, b} + {aa, ab, bb, ba} + {aaa,…} +……

Si on note: X^i l'ensemble des mots de longueur i construits sur l'alphabet X

Donc: $X^* = X^0 + \underbrace{X^1 + X^2 + \ldots\ldots\ldots\ldots\ldots + X^{+\infty}}_{X^+} = \sum_{i=0}^{\infty} X^i$

2.1. Opération sur les mots :

a) La concaténation : soient deux mots w, w'∈ X*, la concaténation de w et w' est définie comme la juxtaposition de w et w'. Elle est notée par w.w' on bien ww'

Exemple X= {0,1} w=10, w'=00
w.w'=1000 ≠ w'w=0010

Remarque : la concaténation, c'est une loi de composition interne sur X^* qui n'est pas commutative mais elle est associative et elle possède un élément neutre qui est ε

☞ Donc X* le monoïde libre engendré par X et on écrit : < X*, **.**, ε >

Exercice : quand est ce qu'on dit qu'un monoïde est libre ?

b) La puissance d'un mot : Soit un alphabet X et w ∈ X*

$$w^n = \begin{cases} \varepsilon & \text{si } n = 0 \\ w & \text{si } n = 1 \\ ww^{n-1} = w^{n-1}w & \text{si } n > 1 \end{cases}$$

Exemple : soit X= {a, b} et w = abb

$w^0 = \varepsilon$

$w^1 = w.\varepsilon = w = abb$ $\qquad$ $w^2 = ww^1 = ww = abbabb$

$w^3 = ww^2 = www = abbabbabb$

c) Factorisation d'un mot : soit un alphabet X et w, u $\in X^*$

- ☞ u est un facteur gauche (préfixe) de w $\Leftrightarrow \exists v \in X^*$ tel que $w = uv$
- ☞ u est un facteur droit (suffixe) de w $\Leftrightarrow \exists v \in X^*$ tel que $w = vu$
- ☞ u est un préfixe propre de w $\Leftrightarrow \exists v \in X^+$ tel que $w = uv$
- ☞ u est un suffixe propre de w $\Leftrightarrow v \in X^+$ tel que w=v.u

Exemple : soit X= {a, b} , w = babb

- Les préfixes de w sont ε ,b , ba, bab, babb
- Les suffixes de w sont ε , b , bb, abb, babb
- Les préfixes propres de w sont ε , b, ba , bab
- Les suffixes propres de w sont ε , b , bb , abb

d) L'inverse (Miroir) : Le miroir d'un mot w est le mot noté w^R obtenu en inversant les symboles (les lettres) de w.

Exemple : w = a_1a_2 ..... a_n
$w^R = a_n$.....a_2a_1

Remarque :

☞ Le miroir de n'importe quel mot composé d'un seul symbole est le mot lui-même.

☞ Le miroir de mot vide est lui-même. $\varepsilon^R = \varepsilon$

☞ Un mot est un palindrome si w^R = w

Exemple : $(aba)^R$= (aba)
$(abccba)^R$=(abccba)

e) La longueur d'un mot : notée par | |, est le nombre de symboles dans ce mot.

Exemple : w=a_1.....a_n $\Rightarrow$|w|=n
w= $\varepsilon \Rightarrow$|w|=0
w=bonjourali $\Rightarrow$|w|=2

Remarque : la langueur d'un mot, est une fonction f .

$$f : X^* \longrightarrow \mathbb{N}$$

Exercice : démontrer que la fonction de longueur est un morphisme de monoïde.

3. Notion de langage

3.1. Définition : un langage L sur un alphabet X est une partie de X*

$L \subseteq X^*$ ou $L \in P(X^*)$

Exemple : soit X= {a,b}

- ϕ est un langage vide ou ensemble qui ne contient aucun mot.
- $\{\varepsilon\}$ est un langage.
- $\{\varepsilon$, ba , a , bba $\}$ est un langage
- {w $\in$ X* / w= a^n tel que n >0} est un langage

Remarque :

☞ Un langage sur X peut être fini ou infini

☞ ϕ est un langage défini sur n'importe quel alphabet.

3.2. Opérations sur les langages : soient L_1, L_2 deux langages construits sur X

a) **L'union** : l'union de L_1 et L_2 est l'ensemble des mots de L_1 et L_2

- ✓ elle est noté aussi par +
- ✓ elle est associative
- ✓ elle est commutative
- ✓ son élément neutre est ϕ
- ✓ son élément absorbant est X* (puisque $\forall$ L : X* $\cup$ L=L $\cup$ X*= X*)

On écrit ainsi : $L_1 + L_2 = L_1 \cup L_2 = \{w \in X^* / w \in L_1 \text{ ou } w \in L_2\}$

b) **L'intersection** : 'intersection de L_1 et L_2 est l'ensemble de mots qui se trouvent en même temps dans L_1 et L_2.
   - ✓ elle est associative
   - ✓ elle est commutative
   - ✓ son élément neutre est X*
   - ✓ son élément absorbant est ϕ (puisque $\forall L : \phi \cap L = L \cap \phi = \phi$)

On écrit ainsi : $L_1 \cap L_2 = L_2 \cap L_1 = \{w \in X^* / w \in L_1 \text{ et } w \in L_2\}$

c) **Le produit** : le produit de L_1 et L_2 est l'ensemble des résultats de la concaténation de chaque mot de L_1avec chaque mot de L_2
   - ✓ il est associatif
   - ✓ il n'est pas commutatif
   - ✓ son élément neutre est $\{\varepsilon\}$
   - ✓ son élément absorbant est ϕ (puisque $\forall L : \phi.L = L.\phi = \phi$)

On écrit ainsi : $L_1.L_2 = \{w_1.w_2 / w_1 \in L_1 \text{ et } w_2 \in L_2\}$

Remarques :

☞Le produit de langages est distributif par rapport à l'union

☞Le produit de langages n'est pas distributif par rapport à l'intersection

$$\forall L_1, L_2, L_3 \subseteq X^*$$
$$L_1.(L_2 \cup L_3) = (L_1.L_2) \cup (L_1.L_3)$$
$$(L_2 \cup L_3).L_1 = (L_2.L_1) \cup (L_3.L_1)$$

De manière générale $\forall L, L_i \subseteq X^*$

$$L.(\bigcup_{i=1}^{\infty} L_i) = \bigcup_{i=1}^{\infty} (L.L_i)$$

$$(\bigcup_{i=1}^{\infty} L_i).L = \bigcup_{i=1}^{\infty} (L_i.L)$$

d) **puissance (itération)** : elle est définie comme suit :

$$L^n = \begin{cases} \{\varepsilon\} & \text{si } n = 0 \\ L & \text{si } n = 1 \\ LL^{n-1} = L^{n-1}L \text{ si } & n > 1 \end{cases}$$

3.3. Notations :

✍ La fermeture positive de L noté L^+ et on écrit $L^+ = \bigcup_{i=1}^{\infty} L^i$

✍ La fermeture étoile de L (dite aussi la fermeture de Kleene de L) note L* et on écrit

$$L^* = \bigcup_{i=0}^{\infty} L^i$$
$$L^* = L^0 + L^1 + ... + L^\infty$$
$$= \{\varepsilon\} + \bigcup_{i=1}^{\infty} L^i = \{\varepsilon\} + L^+$$

Exemple : soit X= {0,1}, L_1= {00, 11, 01} et L_2= {01, 10}

$L_1 + L_2$= {00, 11, 01, 10}

$L_1 \cap L_2$= {01}

$L_1.L_2$ = {0001, 0010, 1101, 1110, 0101, 0110} $\neq L_2.L_1$= {0100, 0111, 0101, 1000, 1011, 1001}

$L_2^* = \{\varepsilon\} \cup L_2 \cup \{L_2.L_2\} \cup .....$

3.4. Quelques résultats sur les langages

☞ $\forall L \subseteq X^* : \varepsilon \in L^*$ ☞ $L.L^* = L^*.L = L^+$ ☞ $(L^*)^* = L^*$ ☞ $L^*.L^* = L^*$ ☞ $\phi^* = \{\varepsilon\}$ ☞ $\{\varepsilon\}^* = \{\varepsilon\}$

3.5. Remarque : soient L_1 , L_2 deux langage

☞ On dit que $L_1 = L_2 \Leftrightarrow L_1 \subseteq L_2$ et $L_2 \subseteq L_1$

☞ On peut dire aussi $L_1 = L_2 \Leftrightarrow L_1$ s'exprime exactement comme L_2

Chapitre N° 3 : Représentation des langages

1. Les grammaires

1.1. définition : une grammaire est un moyen permettant de montrer comment les mots d'un langage sont générés.

Exemple : X= {ali, mohamed, sara, et, , }

L : le langage qui regroupe toutes les listes des noms syntaxiquement correctes.

L= {
ali,mohamed et sara ; ali et mohamed ; ali ; …
ali, sara et mohamed ; mohamed et ali ;
mohamed, ali et sara ; ali et sara ; mohamed ….. ;
mohamed, sara et ali ; sara et ali ;
sara ,mohamed et ali ; mohamed et sara ; sara ….. ;
sara ,ali et mohamed ; sara et mohamed ; ……
}

ali,mohamed,sara $\notin$ L puisque elle est syntaxiquement incorrecte

ali et mohamed,sara $\notin$ L puisque elle est syntaxiquement incorrecte

❖ On construit maintenant la grammaire de ce langage.

1.1.1.Les règles informelles de cette grammaire :

① ali est un nom.
② mohamed est un nom
③ sara est un nom
④ une liste peut être un nom
⑤ une liste peut être suite des noms séparés par « , »
⑥ à la fin de chaque liste qui contient plus d'un nom : remplacer « ,nom » par « et nom »

Remarque :

☞ Dans les règles formelles d'une grammaire :

☑ Les symboles de l'alphabet sont écrits en minuscule et appelés *les symboles des terminaux*

☑ Les autres symboles (sauf le mot vide) sont écrits en majuscule et appelés *les symboles non terminaux.*

☑ La notation $\alpha \longrightarrow \beta / \partial$ signifie que α peut être remplacé par β ou par ∂

☞ Dans notre exemple pour formaliser la règle ⑥ il faut ajouter un symbole spécial (par exemple « FIN ») à la fin de chaque liste qui contient plus d'un nom pour faire le remplacement adéquat. (Puisque le remplacement se fait uniquement à la fin de la liste qui contient plus d'un nom).

1.1.2. Les règles formelles de cette grammaire :

NOM $\longrightarrow$ ali/mohamed/sara

LISTE $\longrightarrow$ NOM/SUITE,NOM FIN

SUITE $\longrightarrow$ NOM/SUITE**,**NOM

,NOM FIN $\longrightarrow$ et NOM

Remarque : la génération des mots (dans notre cas : les listes des noms) commence toujours à partir d'un symbole non terminal appelé l'axiome (dans notre cas l'axiome = LISTE)

1.2. Définition formelle d'une grammaire : Une grammaire G est un quadruplet (N,T,P,S)

tels que

N : ensemble fini de symboles non terminaux.

T : ensemble fini de symboles terminaux.

N $\cap$ **T** $= \phi$

S : symbole non terminal de départ (axiome).

P : ensemble fini de règles de production de la forme

$\alpha \longrightarrow \beta$ tel que $\alpha \in$ $(N \cup T)^+ - T^+$ et $\beta \in (N \cup T)^*$

Exemple : soit la grammaire G qui gérer le langage des listes des noms syntaxiquement correctes.

$G = (\{LISTE, NOM, SUITE, FIN\}, \{ali, mohamed, sara, et, ,\}, P, LISTE)$

Où P est l'ensemble des règles suivantes :

NOM $\longrightarrow$ ali/mohamed/sara

LISTE $\longrightarrow$ NOM/SUITE**,**NOM FIN

SUITE $\longrightarrow$ NOM/SUITE**,**NOM

,NOM FIN $\longrightarrow$ et NOM

1.3. Langage et grammaire : Soient G= (N, T, S, P) et w, w' $\in (N \cup T)^*$

1.3.1. Relation dérive directement : on dit qu'une chaîne w dérive directement sur une chaîne w' (ou w' dérive directement de w) qu'on note $w \Rightarrow w'$ si et seulement si : on applique une fois une règle de G pour passe de w vers w'

C'est-à-dire : il existe une règle $\alpha \to \beta$ dans P telle que $w = u\alpha v$ et $w' = u\beta v$ avec $u, v \in (N \cup T)^*$

Exemple : soit $G = (\{S\}, \{a, b\}, \{S \xrightarrow{(1)} aSb, S \xrightarrow{(2)} \varepsilon\}, S)$

On peut dire que :

S dérive directement sur aSb et on écrit $S \Rightarrow aSb$ puisque $\begin{cases} \exists\ S \to aSb \in P \\ \text{telle que } w = S \text{ et } w' = aSb \\ \text{avec } u = v = \varepsilon \in (N \cup T)^* \end{cases}$

aSb dérive directement sur ab et on écrit $aSb \Rightarrow ab$ puisque $\begin{cases} \exists\ S \to \varepsilon \in P \\ \text{telle que } w = aSb \text{ et } w' = ab \\ \text{avec } u = a \text{ et } v = b \end{cases}$

1.3.2. Relation dérive : on dit qu'une chaîne w dérive sur une chaîne w' (ou w' dérive de w) qu'on note $w \overset{*}{\Rightarrow} w'$ si et seulement si : on applique n fois les règle de G pour passe de w vers w' tel que $n \geq 0$

C'est-à-dire : soit w= w' ou bien il existe une séquence de chaînes $w_1, w_2, \ldots, w_n$ telle que $w = w_1$, $w' = w_n$ et $w_i \Rightarrow w_{i+1}$ pour $\forall\ 1 \leq i < n$

Exemple : soit $G = \left(\{S\}, \{a,b\}, \{S \xrightarrow{(1)} aSb, S \xrightarrow{(2)} \varepsilon\}, S\right)$

On peut dire que :

S dérive sur ab et on écrit $S \overset{*}{\Rightarrow} ab$ puisque pour passer de w vers w' il existe *une séquence de dérivation de longueur 2*

$S \overset{(1)}{\Rightarrow} aSb \overset{(2)}{\Rightarrow} ab$ *c'est la séquence de dérivation de longueur 2*

1.3.3. Langage généré par une grammaire :

définition : le langage engendré par une grammaire G=(N ,T,P,S) est l'ensemble des chaînes qui ne contiennent aucun symbole non terminal. Chaque chaîne est obtenue à travers une séquence de dérivation à partir de l'axiome S.

il est noté par $L(G) = \left\{ w \in T^* / S \overset{*}{\Rightarrow} w \right\}$

Exemple 1 : soit $G = \left(\{S\}, \{a,b\}, \{S \xrightarrow{(1)} aSb, S \xrightarrow{(2)} \varepsilon\}, S\right)$

L(G) = ?

Mot minimal : $S \overset{(1)}{\Rightarrow} \varepsilon$

La forme générale :

$S \overset{n\geq 0(1)}{\Rightarrow} a^nSb^n \overset{(2)}{\Rightarrow} a^nb^n$ Donc : $L(G) = \{ a^nb^n / n \geq 0 \}$

Exemple 2 : soit $G = \left(\{S\}, \{a,b\}, \{S \xrightarrow{(1)} aSb, S \xrightarrow{(2)} ab\}, S\right)$

L(G) = ?

Mot minimal : $S \overset{(1)}{\Rightarrow} ab$

La forme générale :

$S \overset{n\geq 0(1)}{\Rightarrow} a^nSb^n \overset{(2)}{\Rightarrow} a^nabb^n$ Donc : $L(G) = \{ a^nb^n / n \geq 1 \}$

Remarque :

Une grammaire définit un seul langage par contre un même langage peut être engendré par plusieurs grammaires différentes.

1.3.4. Grammaires équivalentes: deux grammaires sont équivalentes si elles engendrent le même langage.

G équivalente à G' $\Leftrightarrow$ L(G) = L(G')

Exemple : $G = \left(\{S,A,B\}, \{a,b\}, \{S \to aS / ABb, A \to Aa / a, B \to b\}, S\right)$

$G' = \left(\{S,A,B\}, \{a,b\}, \{S \to aS / aA, A \to aA / bB, B \to b\}, S\right)$

On trouve que $L(G)=L(G') = \left\{a^k b^2 / k \geq 1\right\}$

1.3.5. Classification des grammaires : Selon la classification de chomoky, les grammaires sont regroupées en quatre types en fonction de la forme de leurs règles de production.

- Soit une grammaire **G** = (N ,T,P,S)

1.3.5.1. G est dite grammaire de type 0 dite aussi grammaire sans restriction (grammaire générale) : si toute ses règles sont de la forme générale suivante

$\alpha \longrightarrow \beta$ avec $\alpha \in (N \cup T)^+ - T^+$ et $\beta \in (N \cup T)^*$

Exemple : $G = \left(\{S\}, \{a,b,c\}, \{S \to aS / Sb / c, aSb \to Sa / bS\}, S\right)$

1.3.5.2. G est dite grammaire de type 1 dite aussi grammaire monotone : si toutes ses règles sont de la forme

$\alpha \longrightarrow \beta$ avec $|\alpha| \leq |\beta|$ tels que $\alpha \in (N \cup T)^+ - T^+$ et $\beta \in (N \cup T)^*$

Exception : $axiome \longrightarrow \varepsilon$ peut appartenir à P.

Exemple :

$G = (\{S, R, T\}, \{a, b, c\}, \{S \to \varepsilon / aRbc / abc, R \to aRTb / aTb, Tb \to bT, Tc \to cc\}, S)$

1.3.5.3. G est dite grammaire de type 2 dite aussi grammaire algébrique (hors contexte) : si toutes ses règles sont de la forme.

$A \longrightarrow \beta$ avec $A \in N$ et $\beta \in (N \cup T)^*$

Exemple: $G = (\{S\}, \{a, b\}, \{S \to aSb / \varepsilon\}, S)$

1.3.5.4. G est dite grammaire de type 3 dite aussi grammaire régulière :si elle est régulière à gauche ou bien à droite.

☑ Une grammaire G est dite régulière à gauche si toutes ses règles sont de la forme $A \longrightarrow Bw$ ou $A \longrightarrow w$ avec A, B $\in$ N et w $\in$ T*

☑ Une grammaire G est dite régulière à droite si toutes ses règles sont de la forme $A \longrightarrow wB$ ou $A \longrightarrow w$ avec A, B $\in$ N et w $\in$ T*

Exemple:

$G_1 = (\{S, A\}, \{a, b\}, \{S \to Sb / Ab, A \to Aa / a\}, S)$ *grammaire régulière à gauche*

$G_2 = (\{S, A\}, \{a, b\}, \{S \to aS / aA, A \to bA / b\}, S)$ *grammaire régulière à droite*

$G_3 = (\{S, A\}, \{a, b\}, \{S \to aS / aA, A \to Ab / b\}, S)$ *grammaire n'est pas régulière; puisque n'est ni régulière à gauche ni régulière à droite*

✍ On peut trouver que G_1et G_2 engendrent le même langage. G_1 génère les mots de ce langage de la droite ver la gauche mais G_2 génère les mots de ce langage de la gauche vers la droite.

Remarques :

☞ Les grammaires de type 0 englobent les grammaires de type 1 qui englobent les grammaires de type 2 qui englobent les grammaires de type 3 .

☞ Il y a une relation d'inclusion stricte entre les 4 types des grammaires c'est à dire :

$$\text{type } 3 \subset \text{type } 2 \subset \text{type } 1 \subset \text{type } 0$$

☞ Une grammaire de type i est aussi de type inférieur à i (avec $1 \le i \le 3$)

☞ Le type retenu pour une grammaire est le type maximum de la grammaire qui vérifie ses règles.

Notation :

✍ Une grammaire est dite de type 2 généralisé, c'est une grammaire qui a la forme de grammaire type 2 et qui contient des règles de la forme $A \longrightarrow \varepsilon$ tel que $A \in N$

✍ Une grammaire est dite de type 3 généralisé, c'est une grammaire qui a la forme de Grammaire type 3 et qui contient des règles de la forme $A \longrightarrow \varepsilon$ tel que $A \in N$

Exemples :

$G_1 = (\{S, R\}, \{a, b\}, \{S \to aS / R / \varepsilon, R \to bR / \varepsilon\}, S)$ *une grammaire de type 3 généralisé*

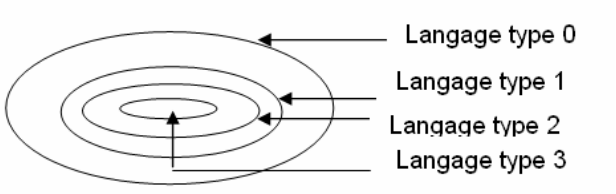
$G_2 = (\{S, A\}, \{a, b, c\}, \{S \to aSc / A, A \to bAc / \varepsilon\}, S)$ *une grammaire de type 2 généralisé*

$G_3 = (\{S, A\}, \{a, b, c\}, \{S \to abcA / Aabc, A \to \varepsilon, Aa \to Sa, cA \to cS\}, S)$ une *grammaire de type 0.*

1.3.6. Classification des langages : La classification des grammaires va permettre de classer les langages selon le type maximum de la grammaire qui l'engendre (puisque un langage peut être engendré par plusieurs grammaires de types différents).

Remarques :

☞ Il y a une relation d'inclusion stricte entre les 4 types des langages.



☞ On dit qu'un langage est de type i s'il est engendré par une grammaire de type i et pas par une grammaire d'un type supérieur.

2. Les automates

2.1. Définition : un automate est un autre moyen de représentation des langages permettant la reconnaissance des mots, c'est-à-dire : un automate d'un langage permet de lire un mot et de dire si ce mot appartient à ce langage ou non.

2.2. Classification des automates : comme les grammaires et les langages, les automates peuvent être classée en 4 types selon la classification de chomsky, la classification de chomsky pour les automates consiste à définir, pour chaque type de langage, l'automate simple permettant de répondre à la question «un mot appartient-il à un langage ? »

- ❖ il à quatre types d'automates :
- ☑ Automate à états finis : il reconnaît la les langages de type 3
- ☑ Automate à pile : il reconnaît les langages de type 2
- ☑ Automate à bornes linéaires : il reconnaît les langages de types 1
- ☑ Machine de Turing : automate qui reconnaît les langages de type 0

Remarque : un langage peut être reconnue par plusieurs automates.

2.3. Automates équivalents : Deux automates sont équivalents s'il reconnaissant le même langage. A équivalente à A' $\Leftrightarrow$ L(A)=L(A')

Chapitre N°4 : Les langages réguliers

☑ **Rappel** : Un langage régulier est un langage engendré par une grammaire régulière (à gauche ou bien à droite).

1. Définition : Soit X un alphabet. Un langage L sur X est 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 ϕ
- ✍ Le langage qui ne contient que le mot vide $\{\varepsilon\}$
- ✍ Le langage de la forme $\{a\}$ avec a $\in$ X

❖ C'est à dire : **Si** L_1 et L_2 sont des langages réguliers sur X
alors $L_1 \bigcup L_2$ est aussi un langage régulier sur X.
alors L_1L_2 est aussi un langage régulier sur X.

Si L est un langage régulier sur X
alors L* est aussi un langage régulier sur X.

Exemples : soit X= $\{a, b\}$

L=X* = $\{a, b\}^*$ est un langage régulier puisqu'il est obtenu par l'application d'une étoile de Kleene sur l'union des deux langages réguliers de base $\{a,\}$ et $\{b\}$
C'est à dire : $\{ \{a\} \cup \{b\} \}^* = \{a, b\}^* = X^*$

$L = \{a^n b^n / n \geq 0\}$ n'est pas un langage régulier puisqu'il est obtenu en appliquant un nombre infini de l'union de la concaténation des langages réguliers de base $\{a,\}, \{b\}$ et $\{\varepsilon\}$

$L = \{a^n b^n / 2 \geq n\}$ 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 $\{\varepsilon\}$
C'est à dire : $\{\varepsilon\} \cup \{\{a\}.\{b\}\} \cup \{\{a\}.\{a\}.\{b\}.\{b\}\}$

L= $\{a^n b^m / n, m \geq 0\}$ est un langage régulier puisque 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 $\{b\}$. C'est à dire: $\{a\}^*.\{b\}^* = \{a^n b^m / n, m \geq 0\}$

Remarque :
- Tout langage fini est régulier.
- Un langage régulier peut être infini.
- Si L_1 et L_2 sont des langages réguliers alors $L_1 \cap L_2$ est aussi régulier.

2. Les expressions régulières :

2.1. Définition : une expression régulière est un autre moyen permettant de représenter un langage régulier de la façon suivante :
- ϕ est une expression régulière qui représente le langage régulière ϕ
- ε est une expression régulière qui représente le langage régulière $\{\varepsilon\}$
- a est une expression régulière qui représente le langage régulière $\{a\}$ avec a ∈ l'alphabet X

❖ C'est à dire :
Si E_1 et E_2 sont des expressions régulières qui représentent respectivement les langages réguliers L_1 et L_2
alors (E_1+ E_2) ou (E_1 / E_2) est une expression régulière qui représente le langage régulier $L_1 \cup L_2$
alors (E_1E_2) est une expression régulière qui représente le langage régulier L_1L_2
alors $(E_1)^*$ est une expression régulière qui représente le langage régulier L^*_1

Remarque :
- Afin d'alléger les parenthèses inutiles des expressions régulières, on introduit l'ordre de priorité suivant :

L'étoile «*» est plus prioritaire que la concaténation «.» et la concaténation «.» est plus prioritaire que l'addition «+»

Exemple : $1+ 01^*$ est donc équivalente à (1+ (0(1)*))

- Comme l'union et la concaténation sont associatives, on peut aussi alléger les parenthèses inutiles des expressions régulières.
  Exemple : (1+ 0 + 011) ⇔ ((1+0) +(01)1))
- La notation [abc] abrège l'expression régulière a/ b/ c
  La notation [a - z] abréger l'expression régulière a/ b/……/z
- Pour tout langage régulier, il existe au moine une expression régulière qui le représente.
- Deux expressions régulières sont équivalentes si elles représentent le même langage.

$$E_1 \text{ équivalente à } E_2 \Leftrightarrow L(E_1)=L(E_2)$$

- Les expressions régulières ont les mêmes propriétés des langages (commutativité, associativité, distribution, élément neutre, élément absorbant)
- $E^+=EE^*=E^*E$

Exemples : soit X= $\{0,1\}$

0^* est une expression régulière qui représente le langage qui contient toutes les chaînes composées uniquement du symbole 0 y compris la chaîne vide ε

110^*1 est une expression régulière qui représente le langage qui contient toutes les chaînes commençant par deux symboles de 1 suivis d'un nombre quelconque de symbole 0, éventuellement nul, et terminés par le symbole 1

$(0/1)(0/1)$ est une expression régulière qui représente le langage $\{00,01,10,11\}$

$(0/1)^*$ une expression régulière qui représente $X^*=\{\varepsilon, \text{toutes les chaînes composées des symboles 0 et 1}\}$

$(0/1)^*01$ est une expression régulière qui représente le langage qui contient toutes les chaînes construites sur X et terminées par le facteur 01.

$0^*/110^*1$ est une expression régulière qui représente le langage qui contient toutes les chaînes générées par l'expression 0^* ou par l'expression 110^*1

3. Les automates d'états finis :

3.1. Définition : un automate d'états finis est une machine abstraite qui permet de lire un mot inscrit sur sa bande d'entrée et de dire à travers sa relation de transition si ce mot là appartient ou non à un langage régulier.

3.2. Schéma d'un automate d'états finis : un automate d'états finis est composé de :

3.2.1. Une bande en entrée : elle est composée d'un nombre fini de cellules, sur une partie desquelles sont inscrits les symboles du mot à lire. Bien sûr un symbole par cellule.

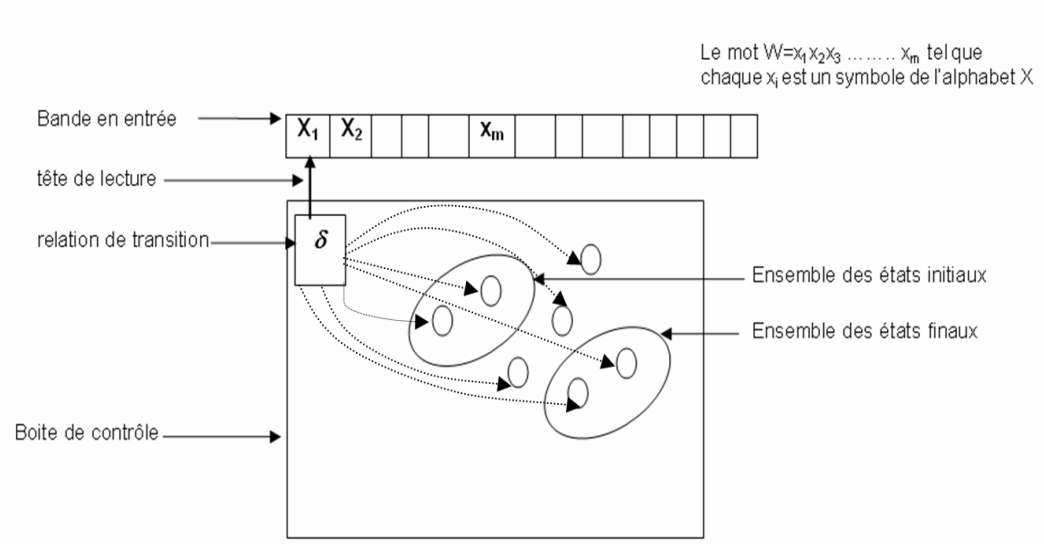
3.2.2. Une tête de lecture : elle ne peut lire qu'un seul symbole à la fois, elle peut être déplacé par la relation de transition pour se positionner sur le symbole juste à droite.

3.2.3. Une boite de contrôle : elle est définie par un ensemble fini d'états et par une relation transition qui décrit le passage entre les états en fonction du symbole lu.

Remarques :

- Parmi les états d'un automate, on distingue :
  - des états initiaux : ce sont les états d'entrée de l'automate.
  - des états finaux : ce sont les états de sortie de l'automate.
- Un état initial peut aussi être final.

- On dit qu'un automate reconnaît un mot inscrit sa bande s'il part d'un état initial en faisant un passage d'état à chaque lecture d'un symbole du mot, l'automate doit se trouve dans un état final à la fin de la lecture du mot.



3.3. <u>Définition formelle d'un automate d'états finis</u> : un automate d'états finis est défini par un 5-uplets A=(X, Q , I , δ , F)

Avec :

X : L'alphabet.

Q : L'ensemble fini des états.

I : L'ensemble des états initiaux ($I \subseteq Q$)

F : L'ensemble des états finaux ($F \subseteq Q$)

δ : La relation de transition définie par l'ensemble fini de transitions de la forme (i,a,j) où i et j sont des états et a est un symbole $\in$ X.

On la note δ (i,a)=j qui signifie la transition de l'état i vers l'état j en lisant le symbole a .

3.4. <u>Représentation d'un automate d'états fini</u> : il existe deux manières de représenter un AEF

3.4.1. <u>Représentation matricielle</u> : elle consiste à représenter un automate d'états finis par une matrice dont :

- Les indices de lignes correspondent aux états de Q.
- Les indices de colonnes correspondent aux symboles de X.
- Chaque case de la matrice de ligne q_i et de colonne x_i correspond à la relation de transition $\delta(q_i, x_i)$

<u>Remarques</u> :

- Les indices de la matrice des états initiaux sont précédés d'une flèche.
- Les indices de la matrice des états finaux sont entourés d'un cercle.
- Avec la relation δ chaque couple (q_i, x_i) peut avoir 0 ,1 ou plusieurs états.

C'est à dire : $\delta(q_i, x_i) = \begin{cases} \phi \text{ représnté par "-" dans la matrice ou} \\ \text{un état par exemple } q_j \text{ ou} \\ \text{plusieurs états regroupés entre } \{\ \} \end{cases}$

3.4.2. <u>Représentation graphique</u> : elle consiste à représenter un automate d'états finis par un graphe orienté comme suit :

- Chaque état est schématisé par un rond (un nœud).
- Chaque état initial est précédé d'une flèche.
- Chaque état final est entouré d'un cercle.
- Pour chaque transition $\delta(q_i, a)=q_j$ on raccorde le noeud q_i au nœud q_j par un arc étiqueté par le symbole a

3.4.2.1. **Exemple** : soit l'automate d'états finis **A**=(X , Q , I , δ ,F) avec

X = {a,b,c}

Q = {q_1, q_2, q_3, q_4}

I = {q_1, q_2}

F = {q_4}

δ (q_1,a)= {q_1,q_3}

δ (q_2,b)= {q_2,q_3}

δ (q_3,c)= {q_3,q_4}

δ (q_4,b)= q_4

La représentation matricielle

	a	b	c
→q_1	{q_1, q_3}	-	-
→q_2	-	{q_2, q_3}	-
q_3	-	-	{q_3, q_4}
(q_4)	-	q_4	-

La représentation graphique