Théorie des Langages et Compilation

Cours 1 : Les Langages

Plan du cours

I. Alphabet

II. Mot

1. Définition

2. Opérations sur les mots

III. Langage

1. Définition

2. Opérations sur les langages

Introduction

De manière générale, les langages sont les supports naturels de communication. Ils permettent aux hommes d'échanger des informations. Ils leur permettent également de communiquer avec les machines.

Les langages utilisés dans la vie de tous les jours entre êtres humains sont dits naturels. Ils sont généralement informels et ambigus et demandent toute la subtilité d'un cerveau humain pour être interprétés correctement.

Alphabet

Définition

Un alphabet ,noté X, est un ensemble fini non vide des éléments appelés symboles.

Remarque

Un symbole est une entité abstraite qui ne peut pas être définie formellement comme les lettres, les chiffre….

Exemple

- X = {0,1}, l'alphabet binaire.
- A= {a,b,c,….,A,B,…é,ç…}, l'alphabet de la langue française
- B={le,bon,lait,0,1,$} est un alphabet de 6 symboles.

Mot

Définition

Un mot sur un alphabet X est une suite finie et totalement ordonnées composées d'éléments de X.

Exemples

- 00, 11, 01001, 10000, 111, 101, ce sont des mots sur l'alphabet X= {0,1}
- bon est un mot sur l'alphabet de la langue française et un symbole sur l'alphabet X={le,bon,lait,0,1,$}. Donc, lebonlait est un mot de 3 symboles sur ce dernier alphabet.

Mot

Définition

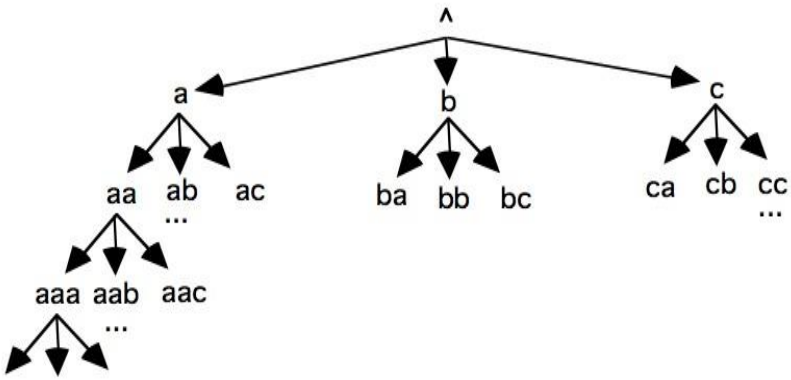
Notations

- Le mot vide, noté par ε, est le mot qui ne contient aucun élément d'un alphabet X.
- L'ensemble des mots construits sur un alphabet X est un ensemble infini noté X*.
- L'ensemble des mots construits sur un alphabet X qui ne contient pas le mot vide est un ensemble infini noté X^+ . On écrit :

$$X^* = X^+ \cup \{\varepsilon\}$$

Mot

Définition



Exemple

Soit l'alphabet X= {a, b, c}, donc X* définit sur X est :

X*={ε, a, b, c, aa, bb, cc, ac, ab, ba, bc, ca, cb, aba, abb,…..}

={ε} U{a, b, c} U{aa, ab, bb,ba,cc,..}U{aaa, bbb, cccc,…}U…U{}

={ε}+{a, b, c} +{aa, ab, bb,ba,cc,..}+{aaa, bbb, cccc,…}+…+{}

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

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

Mot

Longueur d'un mot

Définition:

Soient X un alphabet, $w \in X^*$ et x un des symboles constituant le mot x :

- |w| est la longueur du mot w.
- $|w|_x$ est le nombre de l'occurrence de x dans dans le mot w.

Exemple

X={a, b, c}, |abcbaa|=6, $|abcbaa|_a$=3

Mot

Opérations sur les mots

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

Mot

Opérations sur les mots

Puissance d'un mot

Soit un alphabet X , w ∈X* et n ∈ IN, la puissance de W est donnée comme suit :

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

Exemple

Soit X= {a, b} et w = aba ,

- $w^0=\varepsilon$
- $w^1=w.\ \varepsilon = w=aba$
- $w^2=ww^1=ww=abaaba$
- $w^3=ww^2=www= abaabaaba$

Mot

Opérations sur les mots

Factorisation d'un mot

Soit un alphabet X et w, u ∈ X * :

- u est un facteur gauche (préfixe) de w ⇔ ∃ v ∈ X^* tel que w =uv
- u est un facteur droit (suffixe) de w ⇔ ∃ v ∈ X^* tel que w =vu
- u est un préfixe propre de w ⇔ ∃ v ∈ X^+ tel que w = uv
- u est un suffixe propre de w ⇔ ∃ v ∈ X^+ tel que w = vu

Mot

Opérations sur les mots

Factorisation d'un mot

Exemple:

Soit l'alphabet X= {a, b}, et le mot w = babb :

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

Mot

Opérations sur les mots

Inverse d'un mot ou Miroir

Le miroir d'un mot $w= a_1a_{2.....}a_n$ est le mot noté $w^R= a_n .....a_2a_1$ obtenu en inversant les symboles de w.

Exemple

Soit w=abbc un mot de l'alphabet x={a, b, c}, le mot miroir de w est w^R=cbba.

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 $w^R = w$, ex :$(aba)^R$=aba.

Mot

Opérations sur les mots

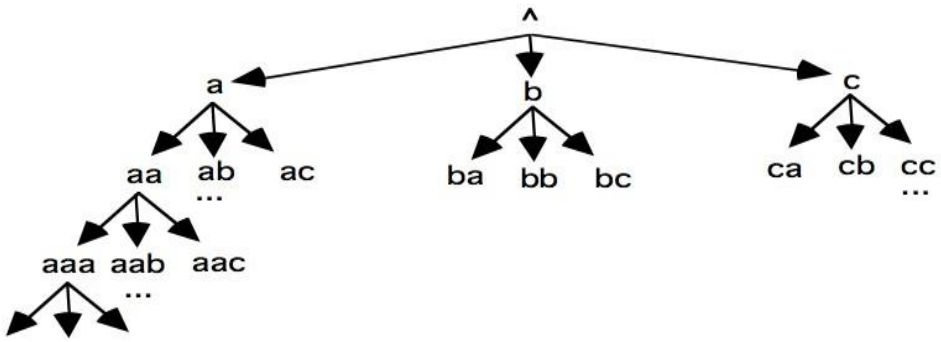
Relation d'ordre

On définit dans X* plusieurs relations d'ordre :

- **Ordre préfixiel** = ordre partiel défini par U < V ssi U est un facteur gauche de V.
- **Ordre lexicographique (ordre du dictionnaire)** = ordre total qui étend l'ordre préfixiel
- **Ordre hiérarchique**= est un ordre total. Les mots sont classés d'abord par longueur, puis pour les mots de même longueur par ordre lexicographique (suppose que l'on a ordonné les lettres de l'alphabet).

-> Sur la représentation graphique

Un mot précède un autre dans un ordre hiérarchique s'il se trouve à un niveau plus élevé ou bien étant sur le même niveau s'il se trouve plus à gauche que celui-ci.



Langage

Définition

Un langage sur un alphabet X est une partie de X*. Donc un langage est un ensemble de mots.

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

❑ **Exemple**

- Soit l'alphabet X={a, b}
- $\varnothing$ est le langage vide, il ne contient aucun mot.
- {ε} est un langage.
- {a, b, aa, bb, aba} est un langage
- {w $\in$ X* / w= a^n tel que n >0}={a,aa,aaa,aaaa,…..} est un langage

❑ **Remarque**

- Un langage sur un alphabet X peut être fini ou infini
- $\varnothing$ est un langage défini sur n'importe quel alphabet.

Langage

Opérations sur les langages

Les langages étant des ensembles, toutes les opérations ensemblistes « classiques » leur sont donc applicables.

Soient $L_1, L_2 \subseteq X^*$: $L_1 \cup L_2 = \{w \in X^* / w \in L_1 \text{ ou } w \in L_2\}$

❑ **Union**

L'union de L_1 et L_2 est l'ensemble des mots de L_1 et L_2 : Elle est :

- Associative.
- Commutative
- Elément neutre , le langage vide $\varnothing$: $L \cup \varnothing = \varnothing \cup L = L$
- Elément absorbant, le vocabulaire X^* : $L \cup X^* = X^* \cup L = X^*$
- Notée + dans la théorie des langages. On écrit aussi : $L_1 + L_2 = \{w \in X^* / w \in L_1 \text{ ou } w \in L_2\}$

Langage

Opérations sur les langages

❑ Intersection

L'intersection de L1 et L2 est l'ensemble des mots qui appartiennent à la fois à L1 et L2 :

$$L_1 \cap L_2 = L_2 \cap L_1 = \{w \in X^* \ / \ w \in L1 \text{ et } w \in L2\}$$

Elle est :

- Associative
- Commutative
- Son élément neutre est X*
- Son élément absorbant est ∅ (puisque " ∀L : ∅ ∩ L=L ∩ ∅ = ∅)

Langage

Opérations sur les langages

❑ Produit ou concaténation

Le produit des langages appelé aussi la concaténation des langages est le résultat de la concaténation de chaque mot de L_1 avec chaque mot de L_2 (Ce n'est pas le produit cartésien) :

Elle est :

$$L_1 . L_2 = \{u . v \,/\, u \in L_1 \text{ et } v \in L_2\}$$

Le produit est :

- Associatif
- Il n'est pas commutatif
- Son élément neutre est {ε}
- Son élément absorbant est ∅

Langage

Opérations sur les langages

- **Théorème**

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

$$\forall L_1, L_2, L_3 \subseteq X^*$$

$$L_1 .(L_2 \cup L_3) = (L_1 .L_2) \cup (L_1.L_3)$$

$$(L_2 \cup L_3) .L_1 = (L_2 .L_1) \cup (L_3.L_1)$$

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

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

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

Attention ! : Le produit de langages n'est pas distributif par rapport à l'intersection.

Langage

Opérations sur les langages

- **Puissance**
  La puissance ou l'itération d'un langage est définit comme suit :

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

Attention ! Ne pas confondre L^n avec le langage contenant les puissances nièmes des mots de L et qui serait défini par $\{u \in X^*, \exists v \in L, u = v^n\}$.

Langage

Opérations sur les langages

- **La fermeture positive**
  La fermeture positive de L noté L+ et on écrit $L^+ = \cup_{i=1}^{\infty} L_i$

- **La fermeture de Kleene**
  La fermeture étoile de L nommée aussi la fermeture de Kleene et notée L* est définit par :

$$L^* = \bigcup_{i=0}^{\infty} L^i$$

$$L^* = L^0 + L^1 + + L^\infty$$

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

- L*=L+∪{ε} et L+=L.L*

Langage

Opérations sur les langages

- **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} ≠ $L_2.L_1$= {0100, 0111, 0101, 1000, 1011, 1001}
- L_2 = {ε} +{L_2} + {L_2. L_2} + {L_2. $L_2.L_2$} +….. +{L_2. $L_2.L_2$……. L_2. $L_2.L_2$}

Application

Soit l'alphabet A = {a, b}

Etant donnés les mots u = aa et v = bab

1. Ecrire les mots uv, $(uv)^2$ et u^3v.
2. Enoncer tous les mots de longueur 2 définis sur A.
3. Définir autrement les ensembles suivants:

$E_1 = \{u.v / u \in A^+, v \in A^+\}$, $E_2 = \{u.v / u \in A^+, v \in A^*\}$, $E_3 = \{u.v / u \in A^*, v \in A^*\}$

Théorie des Langages et Compilation

Cours 2 : Les Grammaires et la Classification de Chomsky

Plan du cours

I. Grammaire

II. Langage engendré par une grammaire

III. Classification - Hiérarchie de Chomsky

Introduction

Au sens littéraire du terme, les grammaires désignent pour les langues naturelles, un ensemble de règles syntaxiques conventionnelles qui déterminent un emploi correct de la langue parlée et écrite. Le même concept s'applique aussi pour la théorie des langages, en suivant les règles de production d'une grammaire on peut engendrer un langage.

Grammaire

Définition

Définition

Une grammaire est un ensemble de règles de production qui sont utilisées pour engendrer un langage.

Exemple

Dans la grammaire de la langue française on peut formuler une phrase en respectant la séquence suivante : '**Sujet' 'Verbe' 'COD'.** On peut formuler un nombre bien défini des phrases en respectant les règles de production suivantes :

PHRASE----------→SUJET VERBE CO

SUJET----------→je l il

VERBE----------→lis l conduit

COD----------→DETERMINANT NOM

DETERMINANT----------→un l la

NOM----------→ livre l voiture

Grammaire

Définition

Exemple: A partir de ces règles syntaxiques on construit $2^4=16$ phrases syntaxiquement correct mais pas forcément sémantiquement correctes.

PHRASE----------→SUJET VERBE COD
SUJET----------→je l il
VERBE----------→lis l conduit
COD----------→DETERMINANT NOM
DETERMINANT----------→un l la
NOM----------→ livre l voiture

Parmi lesquelles, on cite les deux phrases suivantes :

SUJET-----→ Je
VERBE----→lis
DETERMINANT---→ un
NOM-----→ livre

et

SUJET----→ il
VERBE----→ conduit
DETERMINANT----→ la
NOM----→ voiture

PHRASE --→ Je lis un livre
PHRASE ---→ Il conduit la voiture
PHRASE ---→Je conduit la livre

Grammaire

Définition formelle

Définition formelle:

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 ∩ **T=∅**
S : symbole non terminal de départ (axiome).
P : ensemble fini de règles de production de la forme

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

Exemple:

Dans l'exemple précédant, la grammaire est définie comme suit :

- **N**= {PHRASE, SUJET, VERBE, COD, DETERMINANT}
- **T**= {je, il, lire, conduire, un, la, livre, voiture}
- **S**= {PHRASE}
- **P**= {PHRASE →SUJET VERBE COD, COD→DETERMINANT NOM, SUJET→je l il, VERBE→lis l conduit, DETERMINANT→un l la, NOM → livre l voiture }

Grammaire

Définition formelle

Notations:

Dans les règles formelles d'une grammaire :

- Les symboles de l'alphabet sont écrits en minuscule et appelés les symboles des t**erminaux**.
- Les autres symboles (sauf le mot vide) sont écrits en majuscule et appelés les symboles **non terminaux**.
- La génération des mots commence toujours à partir d'un **symbole non terminal** appelé **l'axiome** (dans notre exemple l'axiome = PHRASE).

Grammaire

Dérivation

- **Dérivation directe**

  Un mot w' dérive directement d'un mot w qu'on note w => w' si une règle de G est appliquée **une fois** pour passer de w à w'. C'est-à-dire : il existe une règle α ->β dans P telle que : w=uαv et w'=uβv avec avec u, v ∈ (N∪T)*.

- **Exemple**

Soit la grammaire: G = ({S},{a, b}, {S -> aSb, S->ε }, S) , nous avons :

1. aSb dérive directement de S et on écrit : S => aSb car il existe une règle S->aSb ∈ P telle que : w = S et w' = aSb avec u=v=ε ∈ (N∪T)*

1. ab dérive directement de aSb et on écrit aSb => ab car il existe une règle S->ε telle que : w = aSb et w' = ab avec u=a et v=b

Grammaire

Dérivation

- **Dérivation au sens général**

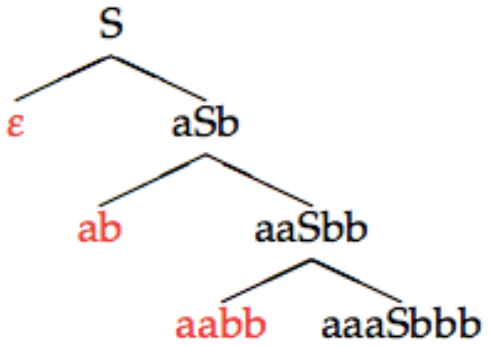
Un mot w' dérive directement d'un mot w qu'on note w -> w', si on applique n fois les règles de G pour passer de w vers w' tel que $n \geq 0$:

C'est-à-dire : 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**

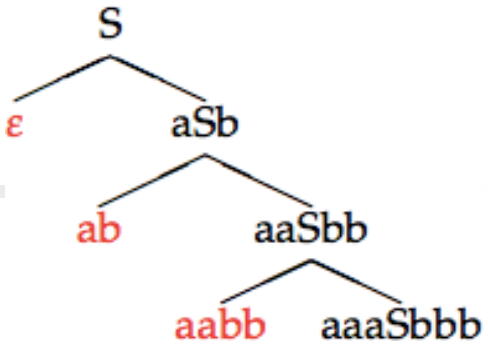
Soit la grammaire: G = ({S},{a, b}, {S -> aSb, S->ε }, S) , nous avons :

S=> aSb => ab, c'est une dérivation de longueur 2



Grammaire

Dérivation



- **Définition**

Un langage engendré par une grammaire G= (N, T, P, S) est l'ensemble des mots obtenus en appliquant des séquences de dérivations à partir de l'axiome S. on note :

$$L(G)=\{ w \in T^* / S \Rightarrow^*_G w \}$$

- **Exemples**

1. Pour la grammaire G = ({A},{a, b}, {A -> aSb, A->ε }, A) , nous avons :

   - Mot minimal: ε

   - La forme générale : L(G)={ $a^n b^n$ / n ≥ 0 }

   S=>aSb=>aaSbb => aaaSbbb…….=> a^n ε b^n => a^n b^n

- **Remarque**

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

Langage engendré par une grammaire

Grammaires équivalentes

- **Définition**

  Deux grammaires sont équivalentes si elles engendrent le même langage.

  $$G \text{ équivalente à } G' \Leftrightarrow L(G) = L(G')$$

- **Exemple**

1. G = ({S, A, B},{a, b},{S ->aS/ABb, A ->Aa/a, B ->b}, S)

S->aS->aABb->aabb ,

S->ABb->AaBb->Aabb->aabb

S->ABb->AaBb->Aabb->Aaabb->Aaaabb>aaaabb

2. G' = ({S, A, B},{a, b},{S ->aS/aA, A ->aA/bB, B->b}, S)

S->aS->aaS->aaaS->aaaaA->aaaabB-> aaaabb

S->aS->aaA->aabB->aabb

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

Langage engendré par une grammaire

Grammaires équivalentes

- **Définition**

  Deux grammaires sont équivalentes si elles engendrent le même langage.

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

- **Exemple**

1. G = ({S, A, B},{a, b},{S ->aS/ABb, A ->Aa/a, B ->b}, S)

S->aS->aABb->aabb ,

S->ABb->AaBb->Aabb->aabb

S->ABb->AaBb->Aabb->Aaabb->Aaaabb>aaaabb

2. G' = ({S, A, B},{a, b},{S ->aS/aA, A ->aA/bB, B->b}, S)

S->aS->aaS->aaaS->aaaaA->aaaabB-> aaaabb

S->aS->aaA->aabB->aabb

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

Classification - Hiérarchie de Chomsky

Classification des grammaires

Hiérarchie de Chomsky

Selon la classification de Chomsky, 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)

- **Grammaire Syntagmatique – Type 0**
- **Grammaire Monotone- Type 1**
- **Grammaire Algébrique (hors contexte)- Type 2**
- **Grammaire Régulière- Type 3**

Classification - Hiérarchie de Chomsky

Grammaires Type 0

❑ **Grammaire Syntagmatique – Type 0**

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

$$\alpha \rightarrow \beta \text{ avec } \alpha \in (N \cup T)^{+} - T^{+} \text{ et } \beta \in (N \cup T)^{*}$$

Exemple :

$$G = (\{S\}, \{a,b,c\}, \{S \rightarrow aS/Sb/c, aSb \rightarrow Sa/bS\}, S)$$

Classification - Hiérarchie de Chomsky

Grammaires Type 1

❑ Grammaire Monotone – Type 1

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

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

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

Exemple :

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

Classification - Hiérarchie de Chomsky

Grammaires Type 2

❑ Grammaire algébrique (hors contexte)– Type 2

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

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

Exemple :

1. $G = (\{S\}, \{a, b\}, \{S \rightarrow aSb / \varepsilon\}, S)$
   Mots : ε, ab, aabb, aaabbb,....., $a^n b^n$
2. $G = (\{S\}, \{0, 1\}, \{S \rightarrow 0S0 / 1S1 / \varepsilon / 0 / 1\}, S)$
   Mots : 00, 11, 010 , 101, 000,111,101101....
   C'est la grammaire des Palindromes

Classification - Hiérarchie de Chomsky

Grammaires Type 2

Arbre de dérivation - Arbre Syntaxique

❑ Définition

Dans le cas d'une grammaire hors-contexte G, on peut représenter la dérivation d'une phrase de L(G) à partir de l'axiome à l'aide d'un arbre syntaxique (ou arbre d'analyse ou arbre de dérivation). Cette représentation fait abstraction de l'ordre d'application des règles de la grammaire et aide à la compréhension de la syntaxe de la phrase considérée. On appelle arbre de dérivation (ou arbre syntaxique), tout arbre tel que :

- La racine est le symbole de départ.
- Les feuilles sont des symboles terminaux ou ε.
- Les nœuds sont des non-terminaux.

Pour déterminer si une chaîne terminale appartient au langage engendré par une grammaire, on établit un arbre de dérivation dont la racine est l'axiome, les feuilles sont des terminaux formant la chaîne donnée et les nœuds sont des variables décrivant les régles utilisées.

Classification - Hiérarchie de Chomsky

Grammaires Type 2

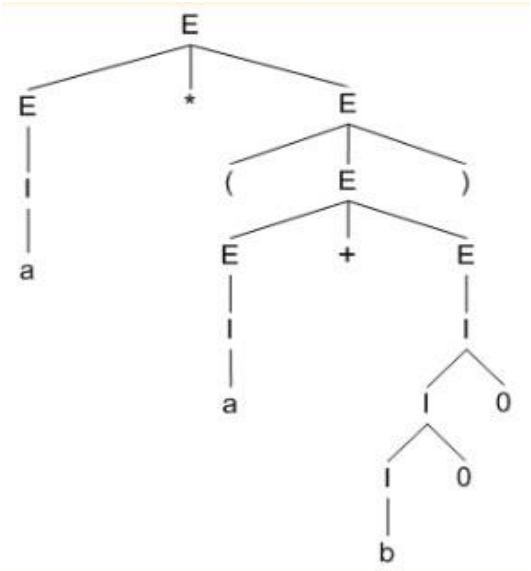
Arbre de dérivation - Arbre Syntaxique

❑ **Exemple**

Soit la grammaire suivante : T={a, b, 0, 1, +, *, (,)}, N={E, I} avec E est l'axiome, et les règles de production suivantes :

$$E \rightarrow I \,/\, E*E \,/\, E+E \,/\, (E)$$

$$I \rightarrow a \,/\, b \,/\, Ia \,/\, Ib \,/\, I0 \,/\, I1$$



La figure ci-coté représente l'arbre de dérivation de l'expression : a ∗ (a + b00). La dérivation la plus à gauche et la plus à droite fournissent le même arbre.

$E \rightarrow E * E$
$\rightarrow I * E \rightarrow a * E$
$\rightarrow a * (E)$
$\rightarrow a * (E+E)$
$\rightarrow a * (I+E)$
$\rightarrow a * (a+E) \rightarrow a * (a+I) \rightarrow a * (a+I0) \rightarrow a * (a+I00) \rightarrow a * (a+b00)$

Classification - Hiérarchie de Chomsky

Grammaires Type 2

Arbre de dérivation - Arbre Syntaxique

❑ **Exemple**

Soit la grammaire suivante : T={a, b, 0, 1, +, *, (,)}, N={E, I} avec E est l'axiome, et les règles de production suivantes :

$$E \rightarrow I \,/\, E*E \,/\, E+E \,/\, (E)$$

$$I \rightarrow a \,/\, b \,/\, Ia \,/\, Ib \,/\, I0 \,/\, I1$$

- **Dérivation la plus à gauche de : a*(b+a0)**

$E \Rightarrow E*E \Rightarrow I*E \Rightarrow a*E \Rightarrow a*(E) \Rightarrow a*(E+E) \Rightarrow a*(I+E) \Rightarrow a*(b+E) \Rightarrow a*(b+I)$
$\Rightarrow a*(b+I0) \Rightarrow a*(b+a0)$

- **Dérivation la plus à droite de : a*(b+a0)**

$E \Rightarrow E*E \Rightarrow E*(E) \Rightarrow E*(E+E) \Rightarrow E*(E+I) \Rightarrow E*(E+I0) \Rightarrow E*(E+a0) \Rightarrow E*(I+a0) \Rightarrow E*(b+a0) \Rightarrow I*(b+a0) \Rightarrow a*(b+a0)$

Classification - Hiérarchie de Chomsky

Grammaires Type 3

❑ Grammaire régulière- Type 3

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 $w \in T^*$

Classification - Hiérarchie de Chomsky

Grammaires Type 3

❑ Grammaire Régulière - Type 3

- 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 :

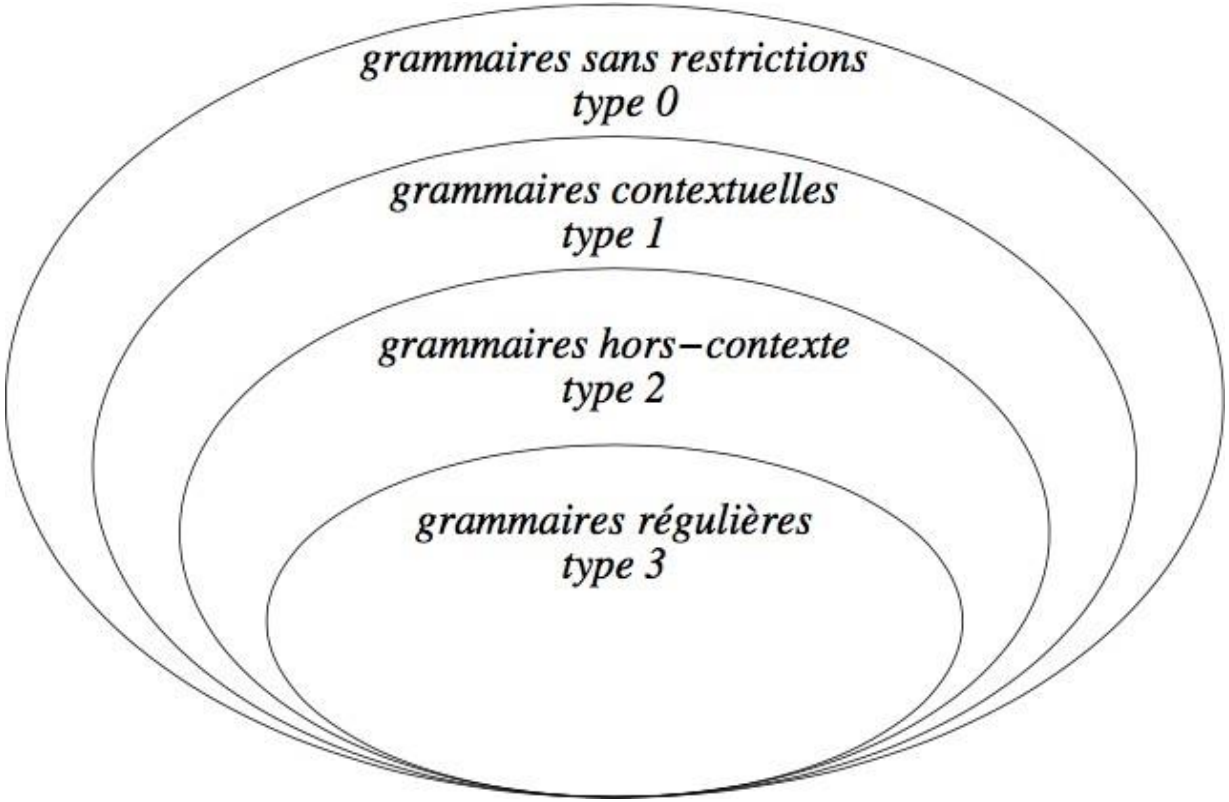
  de la forme $A \longrightarrow wB$ ou $A \longrightarrow w$ avec $A, B \in N$ et $w \in T^*$

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.

Classification - Hiérarchie de Chomsky

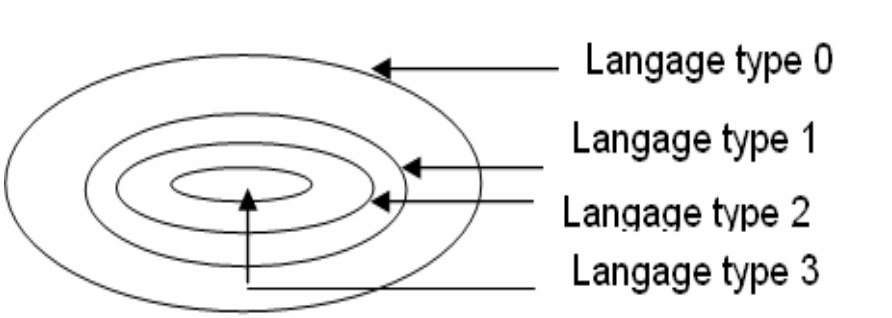
Classification des grammaires



Classification - Hiérarchie de Chomsky

Classification des langages

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

Classification - Hiérarchie de Chomsky

Classification des langages

❑ **Exemples**

- **Langage de Type 0**

  L={ac, acb, ca…}

  G = ⟨{S}, {a, b, c}, {S → aS | Sb | c, aSb → Sa | bS}, S⟩

- **Langages Contextuel - Type 1 :**

  L={$a^n b^n c^n$ | n≥0}

  G = ⟨{S,B,W,X},{a, b, c},{ S → abc, S → aSBc, cB → WB,WB → WX, WX → BX, BX → BC, bB→ bb}, S⟩

Classification - Hiérarchie de Chomsky

Classification des langages

❑ **Exemples**

- **Langages hors Contexte- Type 2:**

  $L = \{a^n b^n \mid n \geq 0\}$

  $G = \langle \{S\}, \{a, b\}, \{S \rightarrow aSb|\varepsilon\}, S \rangle$

- **Langages Rationnel (Régulier)- Type 3**

  $L = \{m \in \{a, b\}*\}$

  $G = \{\{S\}, \{a, b\}, \{S \rightarrow aS \mid bS \mid \varepsilon\}, S\}$

Exercice

Exercice 1:

On considère la grammaire G = (T, N, P, S) où
T = { b, c }
N = { S }
P= { S → bS | cc }
S=S

1. Quel est le type de G.
2. Déterminer L(G).

Exercice 2:

Construire une grammaire pour le langage L = {$ab^n a / n \in N$}.

Théorie des Langages et Compilation

Cours 3 : Langage Régulier et Expressions Régulières

Plan du cours

I. Langage Régulier

1. Définition
2. Exemple
3. Propriétés

II. Expression Régulière

1. Définition
2. Opérateurs
3. Langage décrit par une expression régulière
4. Types d'expressions régulière

Langage Régulier

Introduction

Rappel:

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

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

C'est à dire :

- Si L_1 et L_2 sont des langages réguliers sur X alors: $\mathbf{L_1 \cup L_2}$, $\mathbf{L_1 L_2}$ **et** $\mathbf{L_1}^*$ sont aussi des langages réguliers sur X.

Langage Régulier

Exemple

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 des deux langages réguliers de base {a} et {b}.

   Car on a : {a, b}={a}∪{b} => {a, b}*= {{a}∪{b}}*

2. L = $\{a^n b^n / n \leq 2\}$ 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 {ε}

   C'est à dire: L ={ε, ab, aabb}={ε} ∪ {a}.{b} ∪ {a}.{a}.{b}.{b}

3. L=$\{a^n / n \geq 0\}$ est régulier car il est obtenu en appliquant l'étoile de Kleene sur le langage régulier {a}

   C'est à dire L={ε, a, aa, aaa, aaaa, aaaaa,……}={a}*

Langage Régulier

Exemple

4. 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: $\{a\}^*.\{b\}^* = \{a^n b^m / n, m \geq 0\}$

5. L = $\{a^n b^n / n \geq 0\}$ n'est pas un langage régulier puisqu'il est obtenu en appliquant un nombre infini des opérations d'union et de concaténation des langages réguliers {a} et {b}.

C'est-à-dire $L=\{\varepsilon\} \cup \{a\}.\{b\} \cup \{a\}.\{a\}.\{b\}.\{b\} \cup \{a\}.\{a\}.\{a\}.\{b\}.\{b\}.\{b\} \cup \ldots\ldots\ldots \cup \{a\}.\{a\}\ldots\ldots\{a\}.\{b\}.\{b\}\ldots\ldots\{b\}$

Langage Régulier

Propriétés

Soit un alphabet Σ,

1. Pour tout mot $u \in \Sigma^*$, le langage $\{u\}$ est régulier.

   Si u s'écrit $u=a_1\ a_2 \ldots a_n$ sur Σ, alors le langage $\{u\}$ s'écrit comme la concaténation $\{u\} = \{a_1\}.\{a_2\} \ldots \{a_n\}$.

   $\{u\}$ est régulier car chaque $\{a_i\}$ est un langage régulier.

2. Tout langage fini est régulier.

3. Un langage régulier peut être infini.

Langage Régulier

Propriétés

Soit L et M deux langages réguliers. Alors les langages suivants sont réguliers :

- Union : $L \cup M$
- Intersection : $L \cap M$
- Renversement: $L^R = \{w^R / w \in L\}$
- Fermeture : L^*
- Concaténation : L.M

Expressions Régulières

En pratique

Les expressions régulières sont largement utilisées en informatique. On les retrouve plus particulièrement dans les Shell des systèmes d'exploitation où ils servent à indiquer un ensemble de fichiers sur lesquels est appliqué un certain traitement.

- **Dans une ligne de commande (Shell)** :

Expressions Régulières

En pratique

Les expressions régulières constituent un système très puissant et très rapide pour faire des recherches dans des chaînes de caractères par exemple:

- **Dans une requête SQL**

numetu	nom	prenom	datenaiss	rue	cp	ville
110	Dupont	Albert	1980-06-01	Rue de Crimée	69001	Lyon
222	West	James	1983-09-03	Studio		Hollywood
300	Martin	Marie	1988-06-05	Rue des Acacias	69130	Ecully
421	Durand	Gaston	1980-11-15	Rue de la Meuse	69008	Lyon
575	Titgoutte	Justine	1985-02-28	Chemin du Château	69630	Chaponost
667	Dupond	Noémie	1987-09-18	Rue de Dôle	69007	Lyon
999	Phantom	Marcel	1960-01-30			

```
SELECT * FROM etudiant
WHERE prenom LIKE 'M%';
```

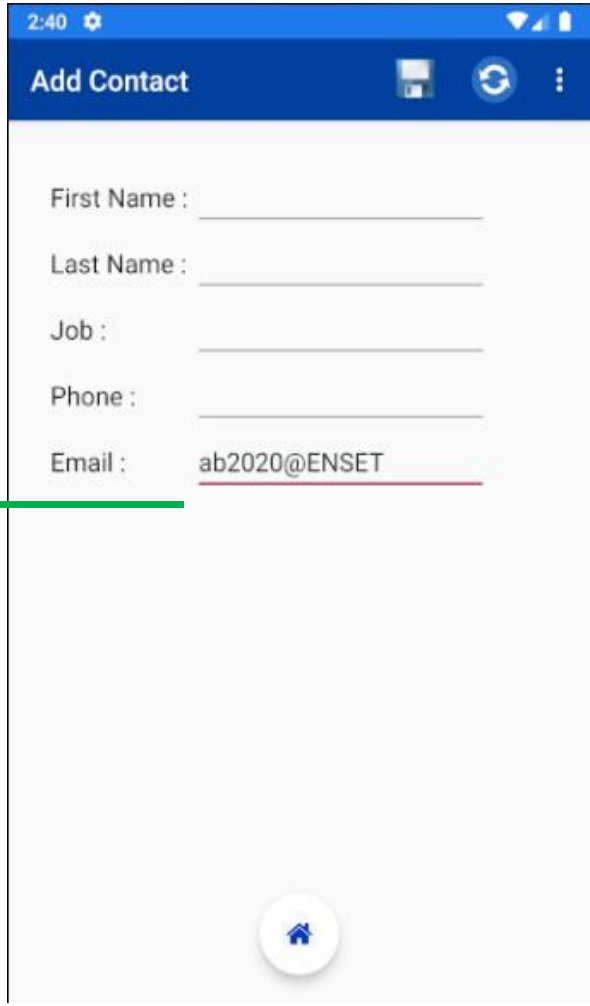
Résultat

numetu	nom	prenom	datenaiss	rue	cp	ville
300	Martin	Marie	1988-06-05	Rue des Acacias	69130	Ecully
999	Phantom	Marcel	1960-01-30			

Expressions Régulières

En pratique

Utilisation des expressions régulières pour le calcul et la manipulation des données



[a-zA-Z0-9]+@[a-zA-Z0-9]+

En Java:

```
email.matches("[a-zA-Z0-9]+@[a-zA-Z0-9]+");
```

Expressions Régulières

Définition

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 = ∅ ou
- E = ε ou
- E=a avec a ∈ X ou
- E = E_1 | E_2 avec E_1 et E_2 sont deux expressions régulières sur X ou
- E = E_1.E_2 avect E_1 et E_2 sont deux expressions régulières sur X ou
- E = E_1^* et E_1 est une expression régulière sur X.
- E=E_{1+} et E_1 est une expression régulière sur X.

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

Expressions Régulières

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

Opérateurs

❑ **Exemples:**

1. a^* : dénote les mots : ε, a, aa, aaa, aaaa, aaaaa,…..a^n.
2. $(a|b)^*$: dénote les mots dans lesquels le symbole a ou b se répètent un nombre quelconque de fois. Elle dénote donc le langage de tous les mots sur {a, b}: ε, a, b, aa, bb, ab,abab,aaaabbb…..etc
3. $(a|b)^*ab\ (a|b)^*$: dénote tous les mots sur {a, b} contenant le facteur ab: ab aabb, aaabbb, abbabaabb
4. b^+ : dénote les mots: b, bb,bbb,bbbbb….

Expressions Régulières

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 $E=a$
- $L(E)=L(E_1) \cup L(E_2)$ si $E=E_1 \mid E_2$
- $L(E) = L(E_1).L(E_2)$ si $E = E_1.E_2$
- $L(E) = L(E_1)^*$ si $E = E_1^*$ où E_1 est une expression régulière sur A.

Expressions Régulières

Langage décrit par une expression régulière

❑ **Exemples**

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

1. E_2 =(ab)* est une expression régulière qui décrit le langage:

$$L(E_2)=\{ab\}^*=\{\varepsilon, ab, abab, ababab,.....\}=\{(ab)^n / n\geq 0\}$$

2. E_3 =a*bbc* est une expression régulière qui décrit le langage

$$L(E_3) =\{a^n bbc^m / n \geq 0, m \geq 0\}$$

3. E_4= (a | b | c)*(bb | cc) a* décrit le langage

$$L(E_4) = \{wbba^n, wcca^n / w \in X^*, n \geq 0\}$$

Expressions Régulières

Types d'Expression Régulière

Il existe plusieurs types d'expression régulières, parmi lesquelles on cite:

- Les expressions régulières basés sur les caractères Jocker.
- Les expressions régulières POSIX de Unix.

Expressions Régulières

Types d'Expression Régulière

❑ **Les expressions régulières basés sur les caractères Jocker (métacaractères)**.

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 utilisés 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 un b

Expressions Régulières

Types d'Expression Régulière

- **Les expressions régulières POSIX**.

Elle 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..etc
- Dans tout les langages de programmation modernes.
- Des éditeurs de textes: Par exemple, rechercher ^http:// et remplacer par HTTP

Expressions Régulières

Types d'Expression Régulière

- **Les expressions régulières POSIX**.

Expression	Signification
[abc]	les symboles a, b ou c
[^abc]	aucun des symboles a, b et c
[a − e]	les symboles de a jusqu'à e {a, b, c, d, 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 deux fois
a{, 5}	a se répétant au plus cinq fois
a{2, 5}	a se répétant entre deux et cinq fois
\x	La valeur réelle de x (un caractère spécial)

Expressions Régulières

Types d'Expression Régulière

- **Les expressions régulières POSIX**.

Exemple:

1. [ab]∗:tous les mots sur {a, b};
2. [^ab]∗ : les mots qui ne comportent ni a ni b ;
3. ([^a] ∗ a[^a] ∗ a[^a]∗)∗ : les mots comportant un nombre pair de a ;

4.(ab{, 4})∗ : en plus de ε, ce sont tous les mots commençant par a où chaque a est suivi de quatre b au plus.

Exercice

Donnez une description de chacune des E.R suivantes, puis donnez leurs ER POSIX

équivalentes:

1. a(a|b)* (b|ε)
2. (aaa)* ;
3. (a|ab|abb)*

Solution

La description de chacune des E.R suivantes et leurs ER POSIX équivalentes:

1. a(a|b)*(b|ε)

- Quelques mots: a, aa, ab, aab, abb, aaab, abbbb, abbba, aaaaa……
- L(E)={ aw / w ∈ {a,b}* }
- ER Unix équivalente: **a[ab]*b?**

2. (aaa)*;

Quelques mots: ε, aaa, aaaaaa, aaaaaaaaa,…….
L(E)={a^{3n} / n ∈ ℕ }
ER Unix équivalente: **(a{3})***

Solution

La description de l'E.R suivante et sa ER POSIX équivalente:

3. (a|ab|abb)*

- Quelques mots: ε, a, ab, abb, aa, aaa,…,abab, aba, abb, abbabb, aababb, aaababbabb
- Cette ER dénote en plus de ε, tous les mots sur l'alphabet {a,b} commençant par a où chaque a est suivi de deux b au plus.
- ER Unix équivalente: **(ab {,2}) * ou (ab?b?) ***