Université Mohammed Khider-BISKRA

Faculté des Sciences Exactes et la nature et de la vie

Département d'informatique

Cours du Module

Théorie des langages

Pr. CHERIF Foudil

Chapitre 5: Les langages algébriques

Les langages algébriques (contexte libre)

a. Transformation des grammaires (mot vide, récursivité, ..)
b. Grammaire de Chomsky
c. Grammaire de Greibach
d. Automates à pile

Chapitre 5: Les langages algébriques

5.1 Introduction

Les langages de type 2, appelés aussi les langages algébriques ou encore les langages hors contexte.

Elles sont reconnus par des machines abstraites semblables aux AEF à une différence prêt qui est la mémoire, cette mémoire est une pile.

Le présent chapitre est consacré à ce type de langages ainsi qu'aux automates à piles.

Chapitre 5: Les langages algébriques

5.2 Rappels sur les grammaires et langages algébriques

Définition d'une grammaire de type 2 :

Une grammaire G=(Vt, Vn, S, R) est dite de à contexte libre (Algébrique ou de type 2) si et seulement si toutes ses règles de production sont sous la forme :

$A \rightarrow W$ avec $A \in Vn$ et $W \in (Vt \cup Vn)*$.

Définition des langages de type2 (hors contexte ou algébrique) :

Ce sont les langages engendrés par des grammaires de type 2.

Remarque :

L'ensemble des langages réguliers est inclus dans l'ensemble des langages algébriques.

Chapitre 5: Les langages algébriques

5.2 Rappels sur les grammaires et langages algébriques

Réciproque :

Tout langage régulier est aussi hors-contexte mais l'inverse est fausse

Remarques:

L'ensemble des langages réguliers est inclus dans l'ensemble des langages algébriques.

Un langage hors-contexte:

- n'est pas reconnu par un automate d'états finis
- n'est pas décrit par une expression régulière
- il n'existe pas de grammaire régulière pour l'engendrer.

Chapitre 5: Les langages algébriques

5.2 Rappels sur les grammaires et langages algébriques

Arbre syntaxique:

Vue l'utilisation d'un seul symbole non terminal à gauche des règles de production dans les grammaires à contexte libre, il est toujours possible de construire un arbre de dérivation pour n'importe quel mot généré.

Soit la grammaire G=(Vt, Vn, S, R) et soit $\omega \in L(G)$. Un arbre syntaxique associé à ω est construit comme suit:

- La racine de l'arbre est étiquetée par l'axiome
- Les nœuds intermédiaires (internes) contiennent des non terminaux
- Les feuilles sont des terminaux

La lecture de gauche à droite des feuilles de l'arbre reconstitue le mot auquel l'arbre est associé.

Chapitre 5: Les langages algébriques

Arbre syntaxique:

Soit la grammaire G=(Vt,Vn, ND,R)

Vt={0,1,2,3,4,5,6,7,8,9,+,-,.,10}

Vn={ ND, S, E,P,F,C}

Axiome = ND

R= (ND → S E P E F

E → C E / C

C → 0 / 1 / 2 / 3 / 4 / 5 / 6 / 7 / 8 / 9

P → .

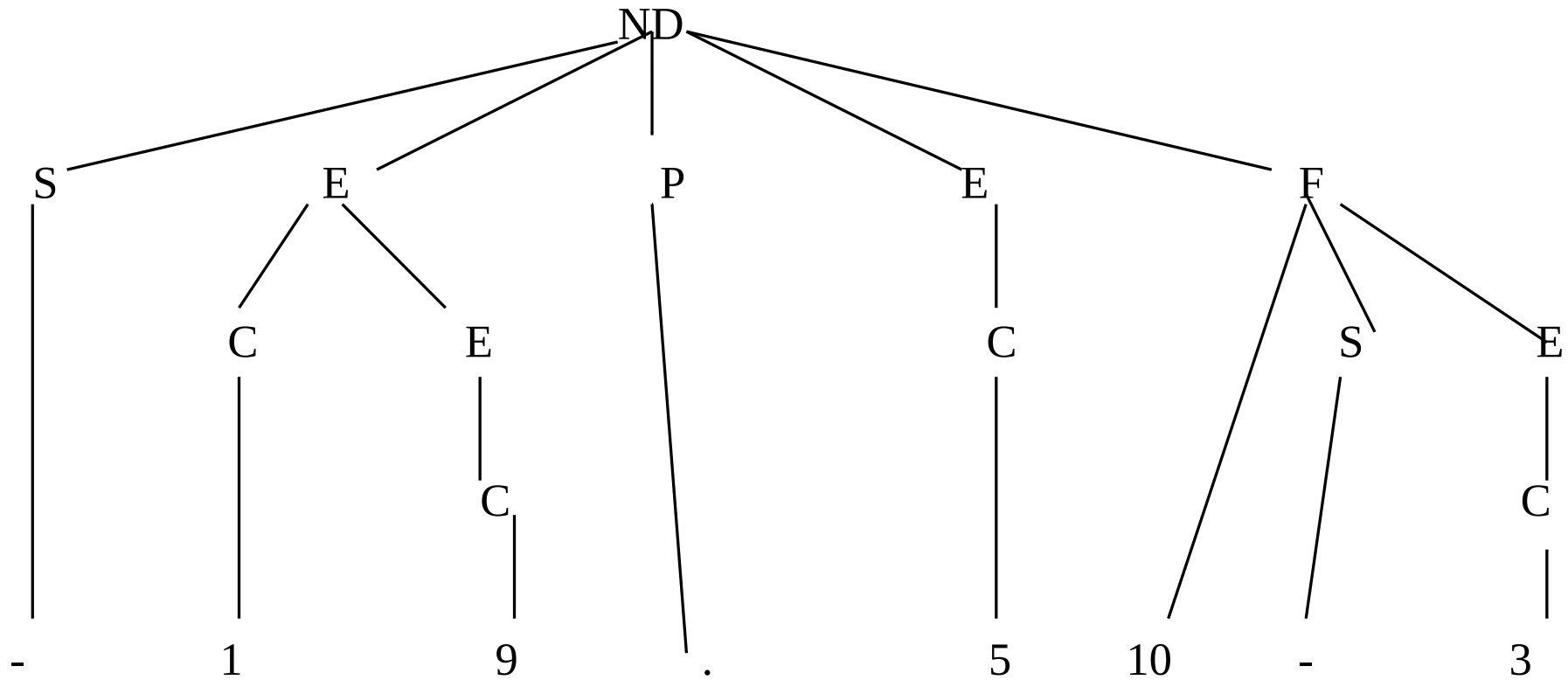
F → 10 S E

S → + / -)

Exemple: Construire l'arbre syntaxique pour le mot : ω = **- 19.5 10^{-3}**

Chapitre 5: Les langages algébriques

Un nombre décimal est représenté par un arbre de dérivation



Chapitre 5: Les langages algébriques

5.2 Rappels sur les grammaires et langages algébriques

Définition d'un mot ambigu :

Un mot ω est dit ambigu si et seulement s'il existe deux arbres de dérivation différents qui lui sont associés, en utilisant la dérivation la plus à gauche.

Définition d'une grammaire ambiguë :

Une grammaire G est dite ambiguë si et seulement s'il existe au moins un mot ambigu appartenant à L(G).

Remarques :

1- Certains langages peuvent être générés, à la fois, par des grammaires ambiguës et des grammaires non ambiguës.

2- Il n'existe aucun algorithme qui permet de trouver une grammaire non ambiguë (si elle existe) qui génère un langage

Chapitre 5: Les langages algébriques

5.2 Rappels sur les grammaires et langages algébriques

Exemple : Soit G la grammaire qui possède les règles de production suivantes :

R=(S→S∧S (1)

S→ S∨S *(2)*

S→ S⇒S (3)

S→ S⇔S (4)

S→ ¬S (5)

S→ q (6)

S→ p (7))

$G=(Vt,Vn,S,R)$

$Vt= \{\wedge, \vee, \Rightarrow, \Leftrightarrow, \neg, q, p\}$

$Vn=\{S\}$

Question : Montrer que G est ambiguë.

Chapitre 5: Les langages algébriques

5.2 Rappels sur les grammaires et langages algébriques

Le mot $\mathbf{p \wedge q \Leftrightarrow p}$ est ambigu, car il existe deux dérivations différentes qui nous permettent de l'engendrer.

$S \rightarrow^{(4)} S \Leftrightarrow S \rightarrow^{(1)} S \wedge S \Leftrightarrow S \rightarrow^{(7)} p \wedge S \Leftrightarrow S \rightarrow^{(6)} p \wedge q \Leftrightarrow S \rightarrow^{(7)} p \wedge q \Leftrightarrow p$

4 → 1 → 7 → 6 → 7

$S \rightarrow^{(1)} S \wedge S \rightarrow^{(6)} p \wedge S \rightarrow^{(4)} p \wedge S \Leftrightarrow S \rightarrow^{(6)} p \wedge q \Leftrightarrow S \rightarrow^{(7)} p \wedge q \Leftrightarrow p$

1 → 6 → 4 → 6 → 7

Nous avons deux chemins différents et par conséquent **la grammaire G est ambiguë**.

Chapitre 5: Les langages algébriques

5.2 Grammaire réduite

Soit G(Vt, Vn, S, R) une grammaire hors contexte.

Symbole productif (utile)et improductif (inutile):

• Un non terminal $A \in Vn$ est dit **utile** si et seulement si $\exists \omega \in Vt^*$ tel que $A \Rightarrow^* \omega$

• Un non terminal $A \in Vn$ est dit **inutile** si et seulement si $\forall \omega \in Vt^*$, il n'existe aucune dérivation indirecte tel que $A \Rightarrow^* \omega$

Symbole accessible et inaccessible

• Un non terminal $A \in Vn$ est dit **accessible** si et seulement si $\exists \alpha \in (Vt \cup Vn)^*$ tel que $S \Rightarrow^* \alpha$ et A apparaît dans α

• Un non terminal $A \in Vn$ est dit **inaccessible** si et seulement si $\forall \alpha \in (Vt \cup Vn)^*$, si $S \Rightarrow^* \alpha$ alors A n'apparaît pas dans α.

Chapitre 5: Les langages algébriques

5.2 Grammaire réduite

Remarque :

Les règles de production(dérivation) qui contiennent des non terminaux inutiles et inaccessibles sont inutiles et peuvent être supprimées sans aucune influence sur le langage généré par la grammaire.

Production unitaire ($A \rightarrow B$ **)**

On appelle **production unitaire** toute règle de production sous la forme $A \rightarrow B$ où A, B $\in Vn$.

Remarque : Pour supprimer la règle unitaire $A \rightarrow B$, il suffit de rajouter aux règles de production de A toutes les productions de B. Cette suppression peut faire apparaître d'autres productions unitaires, pour cela qu'il faut appliquer un algorithme récursif.

Chapitre 5: Les langages algébriques

5.2 Grammaire réduite

Une grammaire est dite **réduite** si et seulement si tous les non terminaux de ses règles de production sont **accessibles** (atteignables) et **utiles** (productifs).

5.2 Grammaire propre

Une grammaire est dite **propre** si et seulement si :

• Elle est réduite

• Elle ne contient pas de productions unitaires

• Il n'y a que l'axiome qui peut générer ε, avec la condition qu'il n'apparaisse dans aucun membre droit des règles.

Chapitre 5: Les langages algébriques

5.2 Grammaire propre

Pour éliminer les ε-productions (mot vide), il faut déterminer d'abord l'ensemble des non terminaux dérivables en ε (directement ou indirectement) ; ensuite, on modifie les productions contenant ces non terminaux, de sorte à remplacer dans toutes les parties gauches des productions les symboles annulables par le mot vide, de toutes les manières possibles.

Exercice 1: Soit G la grammaire hors contexte G(Vt, Vn, S, R)

R= (S→AB | EaE

A→Aa | aB

B→bB | aA

C→AB | aS

E→D

D→dD | ε)

Vt={a,b,d}

Vn={S,A,B,D,E}

S : Axiome

Chapitre 5: Les langages algébriques

1. Trouver le langage engendré par G.

2. Transformer G en une grammaire réduite.

3. Transformer G en une grammaire propre.

4. Vérifier le langage trouvé dans la question 1.

Solution:

1- $L(G)= \{d^n a d^m / n, m \geq 0\}$ on utilise que le symbole E

2- Le non terminal C n'est pas accessible, A et B ne sont pas utiles, donc on supprime les règles contenant A, B ou C et on aura la grammaire suivante :

$S \rightarrow EaE$

$E \rightarrow D$

$D \rightarrow dD \mid \varepsilon$

Chapitre 5: Les langages algébriques

3- La grammaire possède une production unitaire E→D, on l'enlève et on remplace tous les E par des D et on aura la grammaire suivante :

S→DaD

D→dD |ε

La grammaire n'est toujours pas propre à cause de la règle D→ε, donc on l'élimine et pour chaque D qui apparaît à droite d'une règle on crée une autre règle, on obtient la grammaire propre suivante : G'=({a,d}, {S, D } , S, R')

R'= (S→DaD |aD | Da |a

D→dD | d)

4- Le langage trouvé est bien le même, mais plus facile à trouver avec la grammaire propre.

Chapitre 5: Les langages algébriques

Exercice 2:

Réduire la grammaire suivante: G=({a,b}, {S, A,B,C,D } , S, R)

R=(S → aAB | bC | aab A →aA | aB | a C →bb D →b)

Solution: Tous les non terminaux sont utiles, dérivent vers des chaines terminales sauf le symbole B. On supprime toutes les règles contenant B.

G'=({a,b}, {S, A,C,D } , S, R')

R'=(S → bC | aab A →aA | a C →bb D →b)

On remarque que le symbole D est inaccessible a partir de l'axiome S.

G''=({a,b}, {S, A,C } , S, R'') grammaire réduite

R''=(S → bC | aab A →aA | a C →bb)

Chapitre 5: Les langages algébriques

Exercice 3 :

Réduire la grammaire suivante: G=({a,b}, {S, A,B,C } , S, R)

R=(S → aaAb A →bA | a B →bB | b C →a)

Solution:

Tous les non terminaux sont utiles, dérivent vers des chaines terminales sauf le symbole B, par contre les symboles B et C sont inaccessibles à partir de l'axiome S.

On supprime toutes les règles contenant B et C.

G'=({a,b}, {S, A } , S, R') grammaire réduite.

R'=(S → aab A →aA | a)

Remarque: Si l'axiome est inutile le langage engendré par cette grammaire est vide

Chapitre 5: Les langages algébriques

Exercice 4 : Trouver une grammaire propre de la grammaire suivante:

G=({a,b,c}, {S, A,B,C,D } , S, R)

R=($S \rightarrow aSA \mid aS \mid ab \mid aSb \mid BC$ $A \rightarrow SS \mid S \mid c$ $B \rightarrow D$ $C \rightarrow b$

$D \rightarrow a \mid C$)

Solution:

- Tous les non terminaux sont utiles, dérivent vers des chaines terminales

- Tous les symboles sont accessibles à partir de l'axiome S.

- Aucune règle contient le mot vide

- Nous avons 3 règles de type ($A \rightarrow B$, $A, B \in Vn$)

$A \rightarrow S$ $B \rightarrow D$ et $D \rightarrow C$

Chapitre 5: Les langages algébriques

On remplace la règle $A \rightarrow S$ par : $A \rightarrow aSA \mid aS \mid ab \mid aSb \mid BC$

On remplace la règle $D \rightarrow C$ par : $D \rightarrow b$

On remplace la règle $B \rightarrow D$ par : $B \rightarrow a \mid b$

G'=({a,b,c}, {S, A,B,C,D } , S, R')

R'=($S \rightarrow aSA \mid aS \mid ab \mid aSb \mid BC$

$A \rightarrow SS \mid c \mid aSA \mid aS \mid ab \mid aSb \mid BC$

$B \rightarrow a \mid b$

$C \rightarrow b$

$D \rightarrow b$)

On remarque que le symbole **D devient inaccessible**, on supprime la dernière règle

Chapitre 5: Les langages algébriques

G''=({a,b,c}, {S, A,B,C } , S, R'')

R''=($S \rightarrow aSA \mid aS \mid ab \mid aSb \mid BC$

$A \rightarrow SS \mid c \mid aSA \mid aS \mid ab \mid aSb \mid BC$

$B \rightarrow a \mid b$

$C \rightarrow b$)

La grammaire G'' est propre.

Chapitre 5: Les langages algébriques

Exercice 5: Supprimer le mot vide de la grammaire suivante et réduire cette grammaire: G=({a,b,c}, {S, A,B,D,E } , S, R)

R=(S → BE | AaBD | BS A → cB | ε B → bc | AA D → cc

E→ b | ε)

Solution: Les symboles A, E, B,S dérivent directement ou indirectement vers le mot vide (ε), donc on supprime ces règles et on ajoute d'autres règles:

R'=(S → BE | AaBD | BS A → cB B → bc | AA D → cc E→ b

plus ces règles

S → B | E | aBD | AaD | aD | B | S

A → c B → A)

Chapitre 5: Les langages algébriques

R'=($S \rightarrow BE \mid AaBD \mid BS \mid B \mid E \mid aBD \mid AaD \mid aD$

$A \rightarrow cB \mid c \qquad B \rightarrow bc \mid AA \mid A \qquad D \rightarrow cc \qquad E \rightarrow b$)

La grammaire est réduite (sans symboles inutiles et inaccessibles), mais contient des règles de type $(A \rightarrow B)$

R''=($S \rightarrow BE \mid AaBD \mid BS \mid bc \mid AA \mid cB \mid c \mid b \mid aBD \mid AaD \mid aD$

$A \rightarrow cB \mid c$

$B \rightarrow bc \mid AA \mid cB \mid c$

$D \rightarrow cc$

$E \rightarrow b$)

Chapitre 5: Les langages algébriques

5.3 Forme normale de Chomsky FNC (Noam Chomsky)

Une grammaire G=(Vt, Vn, S, R) est dite sous forme normale de Chomsky (FNC) si et seulement si toutes ses règles de dérivation sont sous la forme:

$A \rightarrow BC$ ou $A \rightarrow a$ avec $A, B, C \in Vn$ et $a \in Vt$

- Pour toute grammaire algébrique(contexte libre) , il existe une grammaire équivalente sous forme normale de Chomsky.

- L'intérêt pratique de la FNC est que les arbres de dérivations sont des arbres binaires, ce qui facilite l'application des algorithmes d'exploration des arbres.

Noam Chomsky né en 1928

Chapitre 5: Les langages algébriques

5.3 Transformation d'une grammaire sous FNC

Pour obtenir une grammaire sous forme normale de Chomsky équivalente à une grammaire algébrique G, il faut :

1. Transformer la grammaire en une grammaire propre

2. Pour chaque terminal a, introduire le non terminal C_a, puis rajouter la règle $C_a \rightarrow a$

3. Pour chaque règle $A \rightarrow \alpha$, avec $|\alpha| \geq 2$, on remplace chaque terminal par le non terminal qui lui est associé ;

4. Pour chaque règle $A \rightarrow \beta$, avec $|\beta| \geq 3$, ($\beta = \beta_1\beta_2...\beta_n$), créer les non terminaux D_i, puis remplacer la règle concernée par les règles suivantes : $A \rightarrow \beta_1 D_1$, $D_1 \rightarrow \beta_2 D_2$, ...$D_{n-2} \rightarrow B_{n-1}B_n$. où $D_i = B_{i+1}B_{i+2}...B_n$, avec i variant de 1 jusqu'à $n - 2$.

Chapitre 5: Les langages algébriques

5.3 Exemple de transformation d'une grammaire sous FNC

G=({a,b,c}, {S, A,B,D} , S, R)

R=($S \rightarrow aSB$ (1) $S \rightarrow DcBb$ (2) $A \rightarrow bc$ (3) $B \rightarrow aAb$ (4)

$D \rightarrow b$ (5))

G : est propre

(1) $S \rightarrow aSB$: R'= ($S \rightarrow X_1X_2$ $X_1 \rightarrow a$ $X_2 \rightarrow SB$

(2) $S \rightarrow DcBb$: $S \rightarrow X_3X_4$ $X_3 \rightarrow DX_5$ $X_5 \rightarrow c$ $X_4 \rightarrow BX_6$ $X_6 \rightarrow b$

(3) $A \rightarrow bc$: $A \rightarrow X_6X_5$

(4) $B \rightarrow aAb$: $B \rightarrow X_1X_7$ $X_7 \rightarrow AX_6$

(5) $D \rightarrow b$: $D \rightarrow b$)

G'=({a,b,c},{S,A,B,D, X_1,X_2 ,X_3,X_4 ,$X_5,X_6,$ X_7 }, S, R')

Chapitre 5: Les langages algébriques

5.4 Récursivité à gauche

Une grammaire est récursive à gauche dès qu'elle contient une production de la forme :

$B \rightarrow B\ w$ avec $B \in Vn$ et $w \in (Vn \cup Vt)^*$

- La récursivité à gauche provoque des exécutions infinies ...

En la supprimant, on la transforme alors en récursivité à droite, non gênante

- Prenons un sous-ensemble de productions :

$B \rightarrow B\ \alpha$ $B \rightarrow \beta$ avec $B \in Vn, \alpha, \beta \in (Vn \cup Vt)^*$

et β ne commence pas par B ➔ Il engendre le langage : $\beta\ \alpha^*$

- Ce langage est aussi engendré par :

$B \rightarrow \beta \mid \beta\ Z$

$Z \rightarrow \alpha \mid \alpha\ Z$

avec $Z \notin (Vn \cup Vt)^*$

Chapitre 5: Les langages algébriques

5.4 Récursivité à gauche (exemple)

G=({a,b,c}, {S, A,B,D} , S, R)

R=(S → aS (1) S → BA (2) A → AbD (3) A → b (4) B → bc (5)

D → BA (6) D → DA (7) D → b (8))

Nous avons deux règles récursives à gauche A → AbD (3) et D → DA (7)

Les A-règles seront remplacées par les 4 règles:

A → AbD (3) A → b (4) : **A → b** **A → bZ** **Z → bD** **Z → bDZ**

Les D-règles seront remplacées par les 6 règles:

D → BA (6) D → DA (7) D → b (8) : **D → BA** **D → b** **D → BAZ_1** **D → bZ_1** **Z_1 → a** **Z_1 → aZ_1**

Chapitre 5: Les langages algébriques

5.4 Récursivité à gauche (exemple suite)

La grammaire sans récursivité à gauche G'=({a,b,c}, {S, A,B,D, Z_1, Z_2} , S, R')

R' =($S \rightarrow aS$ $S \rightarrow BA$

$A \rightarrow b$ $A \rightarrow bZ$ $Z \rightarrow bD$ $Z \rightarrow bDZ$

$B \rightarrow bc$

$D \rightarrow BA$ $D \rightarrow b$ $D \rightarrow BAZ1$ $D \rightarrow bZ1$ $Z1 \rightarrow a$ $Z1 \rightarrow aZ1$)

Chapitre 5: Les langages algébriques

5.5 Forme normale de Greibach (FNG)

Une grammaire algébrique est sous forme normale de Greibach (FNG) si et seulement si toutes ses règles de production sont sous la forme :

$A \rightarrow x\alpha$ **ou** $S \rightarrow \varepsilon$, **avec** $x \in Vt, \alpha \in Vn^*$ et S est l'axiome.

Sheila Adele Greibach,

née en 1939 à New York City

Chapitre 5: Les langages algébriques

5.5 Forme normale de Greibach (FNG)

Proposition : Pour toute grammaire algébrique G_1, il existe une grammaire G_2 équivalente sous forme normale de Greibach tel-que $L(G_1) = L(G_2)$.

L'intérêt pratique de la FNG est qu'à chaque dérivation, on détermine un préfixe de plus en plus long formé uniquement de symboles terminaux. Cela permet de construire des automates à piles à partir des grammaires plus facilement, et par conséquent des analyseurs syntaxiques sont facilement implémentables.

Chapitre 5: Les langages algébriques

5.5 Forme normale de Greibach (FNG)

Méthode:

Soit une G une grammaire de type 2

1- Transformer la grammaire sous forme normale de Chomsky(FNC) et éliminer les règles récursives à gauche.

$A \rightarrow BC$ ou $A \rightarrow a$ avec $A, B, C \in Vn$ et $a \in Vt$

Avec un ordre (arbitraire) sur les non terminaux : $Vn = \{ A_1, A_2, A_3, \ldots, A_m \}$ $|Vn| = m$

2- Modifier les règles pour qu'elles vérifient la condition (C) suivante:

$A_i \rightarrow A_j \omega$ avec $j > i$ et $\omega \in Vn^+$ (C)

3- Transformer les règles sous FNG en commençant par le non terminal ayant l'ordre le plus élevé.

Chapitre 5: Les langages algébriques

5.5 Forme normale de Greibach (FNG) (exemple)

Mettre sous FNG la grammaire algébrique suivante: G=({a,b,c,d}, {S, A,B} , S, R)

R=(S → cABdc (1) A → Bb (2) A → aA (3) B → Bd (4)

B→ a (5))

1- Eliminer la récursivité à gauche:

B → Bd (4) B→ a (5) : **B→ a** **B→ a Z** **Z→ d** **Z→ dZ**

Vn'={A,B,C,D,Z}

R'= (S → cABdc A → Bb A → aA B→ a B→ aZ Z→ d

Z→ dZ)

Chapitre 5: Les langages algébriques

5.5 Forme normale de Greibach (FNG) (exemple)

2- Mettre sous forme normale de Chomsky (FNC)

R''=($S \rightarrow X_1X_2$

$X_1 \rightarrow c$

$X_2 \rightarrow AX_3$

$X_3 \rightarrow BX_4$

$X_4 \rightarrow X_5X_1$

$X_5 \rightarrow d$

$A \rightarrow X_6A$

$X_6 \rightarrow a$

$A \rightarrow BX_7$

$X_7 \rightarrow b$

$B \rightarrow a$

$B \rightarrow X_6Z$

$Z \rightarrow d$

$Z \rightarrow X_5Z$)

Ordonner les nouveaux non terminaux:

$Vn'' = \{S,A,B,Z,X1,X2,X3,X4,X5,X6,X7\}$

$Ordre = \{1,2,3,4,5,6,7,8,9,10,11\}$

Chapitre 5: Les langages algébriques

5.5 Forme normale de Greibach (FNG) (exemple)

3- Ordonnancement

R''=($S \rightarrow X_1X_2$ **(C)**

$X_1 \rightarrow c$ **(FNG)**

$X_2 \rightarrow AX_3$ **(non C)**

$X_3 \rightarrow BX_4$ **(non C)**

$X_4 \rightarrow X_5X_1$ **(C)**

$X_5 \rightarrow d$ **(FNG)**

$A \rightarrow X_6A$ **(C)**

$X_6 \rightarrow a$ **(FNG)**

$A \rightarrow BX_7$ **(C)**

$X_7 \rightarrow b$ **(FNG)**

$B \rightarrow a$ **(FNG)**

$B \rightarrow X_6Z$ **(C)**

$Z \rightarrow d$ **(FNG)**

$Z \rightarrow X_5Z$) **(C)**

Chapitre 5: Les langages algébriques

5.5 Forme normale de Greibach (FNG) (exemple)

3- Ordonnancement

$X_2 \rightarrow AX_3$ **(non C)** remplacée par : $X_2 \rightarrow X_6 A\, X_3$ **(C)**

$X_2 \rightarrow BX_7\, X_3$ **(non C)**

$X_2 \rightarrow BX_7\, X_3$ **(non C)** remplacée par : $X_2 \rightarrow aX_7\, X_3$ **(FNG)**

$X_2 \rightarrow X_6 ZX_7\, X_3$ **(C)**

$X_3 \rightarrow BX_4$ **(non C)** remplacée par : $X_3 \rightarrow aX_4$ **(FNG)**

$X_3 \rightarrow X_6 ZX_4$ **(C)**

Chapitre 5: Les langages algébriques

5.5 Forme normale de Greibach (FNG) (exemple)

4- FNG On commence par les non terminaux de priorité supérieure(11,10,….,1)

R’’’= ($X_7 \rightarrow b$ $X_6 \rightarrow a$ $X_5 \rightarrow d$ $X_4 \rightarrow dX_1$ $X_3 \rightarrow aX_4$

$X_3 \rightarrow aZX_4$ $X_2 \rightarrow aA\,X_3$ $X_2 \rightarrow aX_7\,X_3$ $X_2 \rightarrow aZX_7\,X_3$

$X_1 \rightarrow c$ $Z \rightarrow d$ $Z \rightarrow dZ$ $B \rightarrow a$ $B \rightarrow aZ$

$A \rightarrow aA$ $A \rightarrow aX_7$ $A \rightarrow aZX_7$ $S \rightarrow cX_2$)

Chapitre 5: Les langages algébriques

5.5 Forme normale de Greibach (FNG) (exemple)

5-Vérifier la réduction de la nouvelle grammaire:

les symboles B et X_5 sont inaccessibles

R'''= ($X_7 \rightarrow b$ $X_6 \rightarrow a$ $X_4 \rightarrow dX_1$ $X_3 \rightarrow aX_4$

$X_3 \rightarrow aZX_4$ $X_2 \rightarrow aA\,X_3$ $X_2 \rightarrow aX_7\,X_3$ $X_2 \rightarrow aZX_7\,X_3$

$X_1 \rightarrow c$ $Z \rightarrow d$ $Z \rightarrow dZ$ $A \rightarrow aA$

$A \rightarrow aX_7$ $A \rightarrow aZX_7$ $S \rightarrow cX_2$)

Vn'''={ S,A,Z,X_1,X_2,X_3,X_4,X_6,X_7}

Chapitre 5: Les langages algébriques

5.6 Autres Formes normales

Forme normale quadratique:

Un grammaire est sous forme normale quadratique de Greibach si toutes ses règles sont de la forme $A \rightarrow aV$ où V est composé d'au plus deux non terminaux ,

$S \rightarrow aSS \mid b$

Forme normale de Backus :

La forme de Backus-Naur (souvent abrégée en BNF, de l'anglais *Backus-Naur Form*) est une notation permettant de décrire les règles syntaxiques des langages de programmation. Cette syntaxe a été conçue par John Backus et Peter Naur lors de la création de la grammaire du langage Algol 60

Chapitre 5: Les langages algébriques

5.6 Autres Formes normales

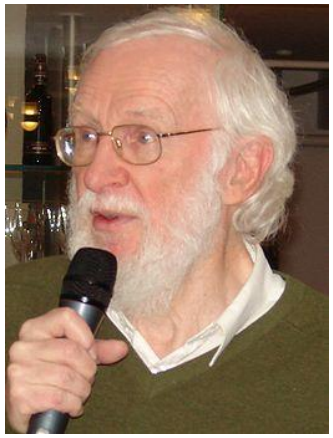
Forme normale de Backus :

Prenons un exemple définissant la structure if du langage C :

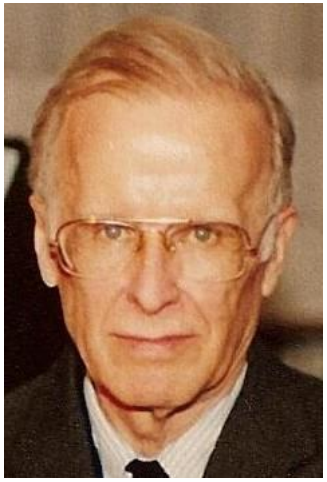
<structure_if> ::= if "(" <condition> ")" "{" <instructions> "}"

<structure_if>, <condition> et <instructions> : sont des non-terminaux.

::= est un méta-symbole signifiant « est défini par ». if, "(", ")", "{" et "}" sont des terminaux.



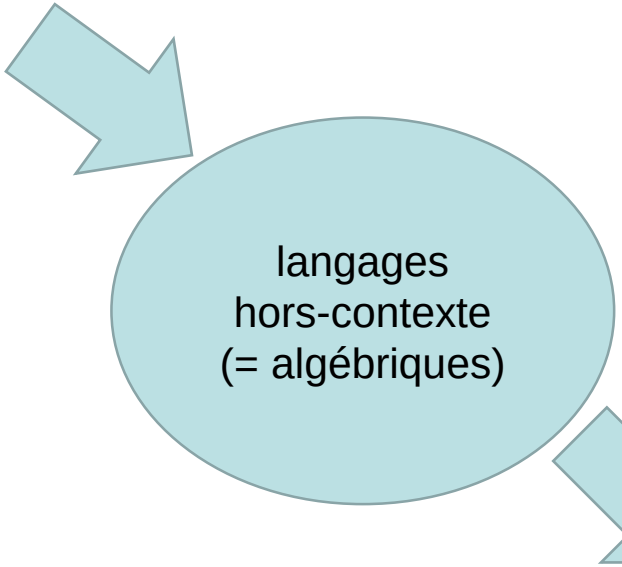
né en 1928 à Frederiksberg (Danemark)



(né en 1924 à Philadelphie)

Chapitre 5: Les langages algébriques

Les grammaires hors-contexte engendrent les…



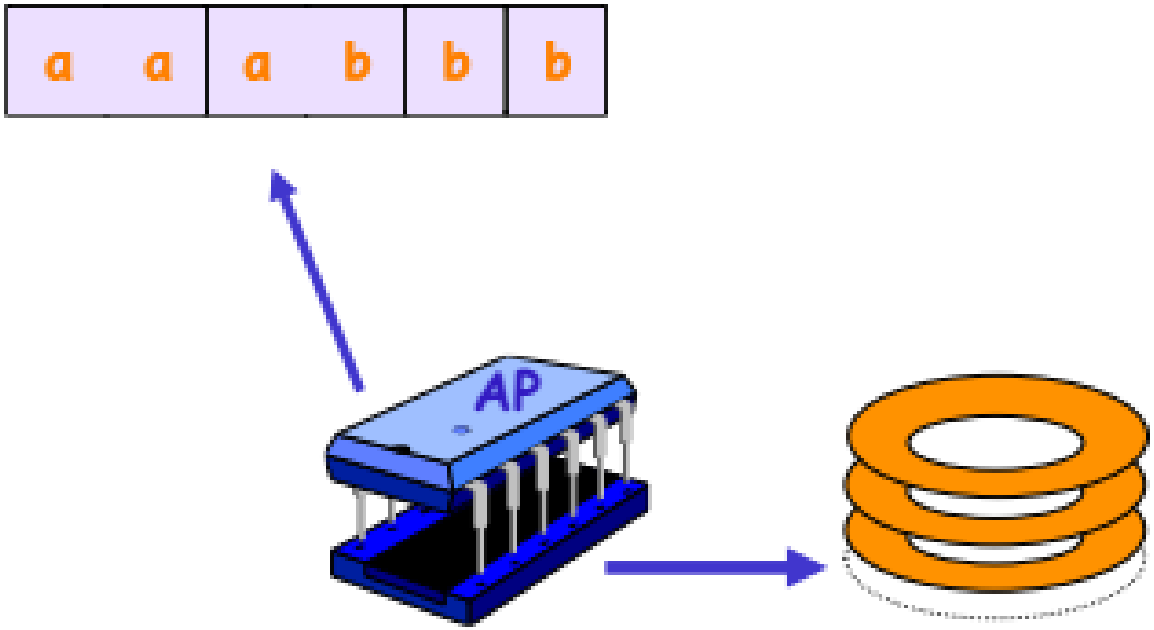
qui sont reconnus par des automates à pile

Comme les AEF, les automates à piles (AP) sont des machines abstraites qui affirment, ou pas, l'appartenance d'un mot à un langage. Les langages reconnus par les AP sont les langages algébriques (Type 2)

Chapitre 5: Les langages algébriques

5.7 Les automates à pile

L'automate à pile AP va tenter de lire le mot **aaabbb :**



Chapitre 5: Les langages algébriques

5.7 Les automates à pile :

Pourquoi une pile ?

- Les automates finis (AEF) n'ont pas d'autre mémoire que leurs états
- Ils ne savent donc pas « compter » au-delà de leur nombre d'états
- Une pile utilise ure une mémoire additionnelle non bornée
- On accède à la pile uniquement par son sommet
- Le nombre de symboles utilisés dans la pile est fini
- La pile vide peut être un critère d'acceptation des mots.

Chapitre 5: Les langages algébriques

5.7 Les automates à pile :

Exemple préliminaire :

Les étapes de reconnaissance du langage $\{a^n b^n, n \geq 1\}$ par un AP pourraient être les suivantes :

1. Lire les a, les stocker dans la pile et ne pas changer d'état

2. A la rencontre du premier b, dépiler un a et changer d'état

3. Dépiler un a pour chaque b rencontré

4. Si les a de la pile se terminent au même moment que les b lus, alors le mot appartient au langage.

Chapitre 5: Les langages algébriques

5.7 Les automates à pile :

Exemple préliminaire :

Les étapes de reconnaissance du langage $\{a^n b^n, n \geq 1\}$ par un AP pourraient être les suivantes :

1. Lire les a, les stocker dans la pile et ne pas changer d'état

2. A la rencontre du premier b, dépiler un a et changer d'état

3. Dépiler un a pour chaque b rencontré

4. Si les a de la pile se terminent au même moment que les b lus, alors le mot appartient au langage.

Chapitre 5: Les langages algébriques

5.7 Les automates à pile :

Définition formelle:

Un AP est défini formellement par un septuplé (Vt, W, Q, q_0, Z_0,f, F) où :

- Vt est l'alphabet d'entrée, fini et non vide
- W est le vocabulaire de la pile, fini et non vide
- Q est l'ensemble d'états, fini et non vide
- q_0 est l'état initial qui appartient à Q
- Z_0 est le symbole initial de la pile(fond de la pile), $Z_0 \in W$
- F est l'ensemble des états finaux (d'acceptation), $F \subset Q$
- f est la fonction de transition

Chapitre 5: Les langages algébriques

5.7 Les automates à pile :

Définition formelle:

f est définie de f: $Q \times (Vt \cup \{\varepsilon\}) \times (W \cup \{\varepsilon\}) \rightarrow Q \times W^*$

Fonctionnement:

Une règle $f(q, a, p) = (q', \chi)$ de transition considère :

- l'état courant q de l'automate
- le caractère lu a sur le ruban (ou peut-être pas : ε)
- le symbole p de sommet de pile (ou peut-être pas : ε)

Une règle indique :

- le prochain état q' de l'automate
- la suite de symboles χ à empiler à la place du sommet de pile

Chapitre 5: Les langages algébriques

5.7 Les automates à pile :

$f(q, a, p) = (q', \chi)$

1- Si $|\chi| = 1$ $\Rightarrow$ $f(q, a, A) = (q', B)$: remplacer A par B

2- Si $|\chi| = 2$ $\Rightarrow$ $f(q, a, A) = (q', BA)$: empiler B

3- Si $|\chi| = 0$ $\Rightarrow$ $f(q, a, A) = (q', \varepsilon)$: dépiler A

4- Si $a = \varepsilon$ $\Rightarrow$ $f(q, \varepsilon, A) = (q', \chi)$: on ne change pas le symbole d'entrée

5- Si $a = \varepsilon$ et $\chi = \varepsilon$ $\Rightarrow$ $f(q, \varepsilon, A) = (q', \varepsilon)$: Vider le contenu de la pile

6- Un mot appartient au langage s'il est entièrement lu et que l'automate dans un état final ou la pile est vide

Chapitre 5: Les langages algébriques

5.7 Les automates à pile : ***Mode de reconnaissance:***

Il existe 2 modes de reconnaissance pour les AP selon que F égale ∅ ou pas :

- Reconnaissance par état final
- Reconnaissance sur pile vide

Le langage accepté par A est :

- **Reconnaissance sur état final**

$L_F(A)$ = { w ∈ Vt*, la pile contenant Z_0 au départ, il existe une lecture de w depuis q_0 à q_f, q_f ∈ F }

- **Reconnaissance par pile vide**

$L_\emptyset(A)$ = { w ∈ Vt*, la pile contenant Z_0 au départ, il existe une lecture de w depuis q_0 qui termine avec la pile vide }

Chapitre 5: Les langages algébriques

5.7 Les automates à pile : ***Configuration initiale:***

Une configuration de l'AP A, à un certain instant, est donnée par le contenu de la pile, l'état courant de l'AP et du mot qui reste à lire:

(contenu de la pile, état courant, mot qui reste à lire).

La configuration initiale est (Z_0, q_0, ω), où q_0 est l'état initial de l'AP et ω le mot soumis à A (à reconnaître).

Chapitre 5: Les langages algébriques

5.7 Les automates à pile : *Equivalence entre les AP*

- Le mode de reconnaissance par état final est équivalent au mode de reconnaissance par pile vide.
- Un AP reconnaît un et un seul langage, mais le même langage peut être reconnu par plusieurs AP
- On dit que deux AP A_1 et A_2 sont équivalents si et seulement s'ils reconnaissent le même langage, $L(A_1) = L(A_2)$.

Chapitre 5: Les langages algébriques

5.7 Les automates à pile :

***Exemple**: Construire l'AP qui reconnait le langage suivant:*

L(A)= { w/ w ∈ Vt* et w= x c x^R tel que x ∈ { 0,1}* } Vt={ 0,1,c}

(x^R signifie x miroir)

Exemples de mots reconnus:

0110c0110 , c, 0c0, 1c1 , 00c00, 11c11

Solution 1: Automate à pile vide F= ϕ

A= (Vt, W, Q, q0, Z_0,f, ϕ)

Vt={ 0,1,c} W={ Z_0, A,B} Q = {q_0,q_1,q_2}

On empile un A pour le symbole 0 et B pour le symbole 1

Chapitre 5: Les langages algébriques

5.7 Les automates à pile :

$f(q_0,c, Z_0) = (q_0, \varepsilon)$: dépiler Z_0 *arret avec une pile vide*

$f(q_0,0, Z_0) = (q_1, AZ_0)$: empiler A *et changer l'état*

$f(q_0,1, Z_0) = (q_1, BZ_0)$: empiler B *et changer l'état*

$f(q_1,0, A) = (q_1, AA)$: empiler A *boucle d'empilement*

$f(q_1,1, A) = (q_1, BA)$: empiler B *boucle d'empilement*

$f(q_1,1, B) = (q_1, BB)$: empiler B *boucle d'empilement*

$f(q_1,0, B) = (q_1, AB)$: empiler A *boucle d'empilement*

Chapitre 5: Les langages algébriques

5.7 Les automates à pile :

$f(q_1,c, A) = (q_2, A)$: remplacer A par A, *passer le c et changer l'état*

$f(q_1,c, B) = (q_2, B)$: remplacer B par B, *passer le c et changer l'état*

$f(q_2,0, A) = (q_2, \varepsilon)$: dépiler A *en cas de correspondance des symboles*

$f(q_2,1, B) = (q_2, \varepsilon)$: déplier B *en cas de correspondance des symboles*

$f(q_2, \varepsilon) , Z_0) = (q_2, \varepsilon)$: déplier Z_0 pile vide *arrêt de l'automate*

Chapitre 5: Les langages algébriques

5.7 Les automates à pile :

***Exemple**: Construire l'AP qui reconnait le langage suivant:*

L(A)= { w/ w ∈ Vt* et w= x c x^R tel que x ∈ { 0,1}* } Vt={ 0,1,c}

(x^R signifie x miroir)

Exemples de mots reconnus:

0110c0110 , c, 0c0, 1c1 , 00c00, 11c11

Solution 2: Automate à pile à états finaux F≠ ϕ

A= (Vt, W, Q, q0, Z_0,f, F)

Vt={ 0,1,c} W={ Z_0, A,B} Q = {q_0,q_1,q_2,q_3}, F={q_3 }

On empile un A pour le symbole 0 et B pour le symbole 1

Chapitre 5: Les langages algébriques

5.7 Les automates à pile :

$f(q_0,c, Z_0) = (q_3, \varepsilon)$: dépiler Z_0 *arret avec* q_3 état final

$f(q_0,0, Z_0) = (q_1, AZ_0)$: empiler A *et changer l'état*

$f(q_0,1, Z_0) = (q_1, BZ_0)$: empiler B *et changer l'état*

$f(q_1,0, A) = (q_1, AA)$: empiler A *boucle d'empilement*

$f(q_1,1, A) = (q_1, BA)$: empiler B *boucle d'empilement*

$f(q_1,1, B) = (q_1, BB)$: empiler B *boucle d'empilement*

$f(q_1,0, B) = (q_1, AB)$: empiler A *boucle d'empilement*

Chapitre 5: Les langages algébriques

5.7 Les automates à pile :

$f(q_1,c, A) = (q_2, A)$: remplacer A par A, *passer le c et changer l'état*

$f(q_1,c, B) = (q_2, B)$: remplacer B par B, *passer le c et changer l'état*

$f(q_2,0, A) = (q_2, \varepsilon)$: dépiler A *en cas de correspondance des symboles*

$f(q_2,1, B) = (q_2, \varepsilon)$: déplier B *en cas de correspondance des symboles*

$f(q_2, \varepsilon) , Z_0) = (q_3, \varepsilon)$: déplier Z_0 , q_3 *état final arrêt de l'automate*

Chapitre 5: Les langages algébriques

5.7 Les automates à pile : ***AP déterministe et non déterministe***

Il existe deux cas de non déterminisme pour les AP :

1. Pour le même sommet de pile, même état et le même symbole d'entrée, il existe au moins deux transitions ;

$f(q_1,a, A) = (q_1, \chi_1)$

$f(q_1,a, A) = (q_2, \chi_2)$

2. Pour le même sommet de pile et même état, on a le choix de lire ou ne pas lire du ruban.

$f(q_1,a, A) = (q_1, \chi_1)$

$f(q_1,\varepsilon, A) = (q_1, \chi_1)$

Chapitre 5: Les langages algébriques

5.7 Les automates à pile : ***Remarques***

- Il existe des langages algébriques pour lesquels il n'existe pas d'AP déterministe les reconnaissant.

- Si un Langage L est reconnu par un automate à pile déterministe, alors il existe une grammaire algébrique non ambiguë générant L.

- Pour chaque langage algébrique L, il existe un AP A tel que L(A)=L

- Les langages reconnus par des AP sont des langages algébriques.

Chapitre 5: Les langages algébriques

5.7 Les automates à pile :

***Exemple**: Construire l'AP qui reconnait le langage suivant:*

L(A)= { w/ w ∈ Vt* et w= x x^R tel que x ∈ { 0,1}* } Vt={ 0,1}

(x^R signifie x miroir)

Exemples de mots reconnus:

01100110 , 00, 11 , 0000, 1111

Solution 1: Automate à pile à états finaux F≠ ϕ

A= (Vt, W, Q, q0, Z_0,f, F)

Vt={ 0,1} W={ Z_0, A,B} Q = {q_0,q_1,q_2,q_3}, F={,q_0, q_2 }

On empile un A pour le symbole 0 et B pour le symbole 1

Chapitre 5: Les langages algébriques

5.7 Les automates à pile :

$f(q_0,0, Z_0) = (q_0, AZ_0)$: empiler A

$f(q_0,1, Z_0) = (q_0, BZ_0)$: empiler B

$f(q_0,0, A) = (q_0, AA)$: empiler B *boucle d'empilement des symboles de x*

$= (q_1, \varepsilon)$: dépiler A *et changer l'état, le premier symbole de* x^R,

$f(q_0,1, A) = (q_0, BA)$: empiler B *boucle d'empilement*

$f(q_0,0, B) = (q_0, AB)$: empiler B *boucle d'empilement*

$f(q_0,1, B) = (q_0, BB)$: empiler A *boucle d'empilement*

$= (q_1, \varepsilon)$: dépiler A *et changer l'état, le premier symbole de* x^R,

---

Chapitre 5: Les langages algébriques

5.7 Les automates à pile :

$f(q_1,0, A) = (q_1, \varepsilon)$: dépiler A *en cas de correspondance des symboles*

$f(q_1,1, B) = (q_1, \varepsilon)$: déplier B *en cas de correspondance des symboles*

$f(q_1, \varepsilon) , Z_0) = (q_2, \varepsilon)$: déplier Z_0 , q_2 *état final arrêt de l'automate*

Remarque: C'est un automate non déterministe, on ne peut pas savoir le premier symbole de $x^{R.}$ Lorsque on a deux symbole consécutifs égaux (00 ou 11), on doit faire deux, soit on empile soit on dépile.

Chapitre 5: Les langages algébriques

5.7 Les automates à pile : Grammaire et AP

- Pour chaque langage algébrique L, il existe un AP A tel que L(A)=L.
- Les langages reconnus par des AP sont des langages algébriques.

Méthode: passage grammaire AP

Pour toute grammaire algébrique G = (Vt, Vn, S, R), il existe un AP

A= (Vt, W, Q, q_0, Z_0,f, F) tel que L(G)=L(A)

Soit la grammaire algébrique G, il s'agit de trouver un AP A reconnaissant le langage L(G) par pile vide. (F= ϕ)

1. En premier lieu, on transforme G sous forme normale de Greibach

2. Ensuite, on définit les paramètres de l'AP comme suit:

Chapitre 5: Les langages algébriques

5.7 Les automates à pile : **Grammaire et AP**

$G = (Vt, Vn, S, R)$ ➔ $A= (Vt', W, Q, q_0, Z_0, f, F)$

$Vt' = Vt$

$W = Vn$

$Q = \{q_0\}$;

$Z_0 = S$;

$B \rightarrow aA_1A_2 \ldots A_n$ ➔ $f(q_0,a,B) = (q_0,A_n..A_2A_1)$

Chapitre 5: Les langages algébriques

5.7 Les automates à pile : Exemple de passage Grammaire et AP

G = (Vt, Vn, S, R) ➔ Vt={ a,b,c} Vn={ S,A,B}

R=(S → aSA S → bSB S → c A → a B → b)

Nous voulons construire l'automate à pile équivalent à pile vide

A= (Vt', W, Q, q_0, Z_0,f, F)

Vt'= Vt W = Vn Q = {q_0} Z_0 = S F= ϕ

S → aSA ➔ f(q_0, a, S) = (q_0, AS)

S → bSB ➔ f(q_0, b, S) = (q_0, BS)

S → c ➔ f(q_0, c, S) = (q_0, ε)

A → a ➔ f(q_0, a, A) = (q_0, ε)

B → b ➔ f(q_0, b, B) = (q_0, ε)

Chapitre 5: Les langages algébriques

5.7 Les automates à pile : Grammaire et AP

Méthode: passage AP grammaire algébrique

Pour tout AP A= (Vt, W, Q, q_0, Z_0,f, F) il existe grammaire algébrique

G = (Vt, Vn, S, R) tel que L(G)=L(A)

Soit l'automate à pile AP A= (Vt, W, Q, q_0, Z_0,f, ϕ) donné:

Il s'agit construire la grammaire G = (Vt', Vn, S, R) de la façon suivante:

1- Vt' = Vt

2- Vn={ [q, A, p] / $\forall$ q $\in$ Q et $\forall$ p $\in$ Q et $\forall$ A $\in$ W $\cup$ {S} }

3- R = 3 types de règles

Chapitre 5: Les langages algébriques

5.7 Les automates à pile : Grammaire et AP

3- R = 3 types de règles

a) S → $[q_0, Z_0, q]$ $\forall q \in Q$ autant de flèches qu'il y a d'états dans A

b) si $f(q, a, A) = (q_1, B_1B_2 \ldots B_m)$ est une transition alors

créer la règle

$[q, A, p] \rightarrow a\,[q_1, B_1, q_2]\,[q_2, B_2, q_3] \ldots [q_m, B_m, q_{m+1}]$

$p = q_{m+1}$, $q_1, q_2, \ldots, q_{m+1} \in Q$

c) si $f(q, a, A) = (p, \varepsilon)$ est une transition alors créer

$[q, A, p] \rightarrow a$

Chapitre 5: Les langages algébriques

5.7 Les automates à pile : Exemple de passage Grammaire et AP

Soit l'automate à pile acceptant le langage suivant:

L={ w/ w ∈ {0, 1}*, w contient autant de 0 que de 1 }

A= (Vt, W, Q, q_0, Z_0,f, ϕ)

Vt={0,1} W= {Z_0 , A, B } Q={ q_0 } ,

Fonctions de transition f:

1) f(q_0, 0, Z_0) = (q_0, AZ_0)

2) f(q_0, 1, Z_0) = (q_0, BZ_0)

3) f(q_0, 0, A) = (q_0, AA)

4) f(q_0, 1, B) = (q_0, BB)

5) f(q_0, 1, A) = (q_0, ε)

6) f(q_0, 0, B) = (q_0, ε)

7) f(q_0, ε , Z_0) = (q_0, ε)

Chapitre 5: Les langages algébriques

5.7 Les automates à pile : **Exemple de passage Grammaire et AP**

La grammaire équivalente est la suivante: G = (Vt, Vn, S, R)

$Vn=\{S\} \cup \{[q_0, Z_0, q_0] , [q_0, A, q_0] , [q_0, B, q_0] \}$

$|Q| = 1$ une seule flèche ➔ $S \rightarrow [q_0, Z_0, q_0]$

1) $f(q_0, 0, Z_0) = (q_0, AZ_0)$ ➔ $[q_0, Z_0, q_0] \rightarrow 0\ [q_0, A, q_0]\ [q_0, Z_0, q_0]$

2) $f(q_0, 1, Z_0) = (q_0, BZ_0)$ ➔ $[q_0, Z_0, q_0] \rightarrow 1\ [q_0, B, q_0]\ [q_0, Z_0, q_0]$

3) $f(q_0, 0, A) = (q_0, AA)$ ➔ $[q_0, A, q_0] \rightarrow 0\ [q_0, A, q_0]\ [q_0, A, q_0]$

4) $f(q_0, 1, B) = (q_0, BB)$ ➔ $[q_0, B, q_0] \rightarrow 1\ [q_0, B, q_0]\ [q_0, B, q_0]$

5) $f(q_0, 1, A) = (q_0, \varepsilon)$ ➔ $[q_0, A, q_0] \rightarrow 1$

6) $f(q_0, 0, B) = (q_0, \varepsilon)$ ➔ $[q_0, B, q_0] \rightarrow 0$

7) $f(q_0, \varepsilon , Z_0) = (q_0, \varepsilon)$ ➔ $[q_0, Z_0, q_0] \rightarrow \varepsilon$

Chapitre 5: Les langages algébriques

5.7 Les automates à pile : **Exemple de passage Grammaire et AP**

La grammaire équivalente est la suivante: G = (Vt, Vn, S, R)

Vn= {S} ∪ {[q_0, Z_0, q_0] , [q_0, A, q_0] , [q_0, B, q_0] } ={S,X, Y, Z}

|Vt={0,1}

R=(S → X

X → ε / 0YX / 1ZX

Y → 1 / 1YY

Z → 0 / 1ZZ)