

THÉORIES DES LANGAGES

Mr,HEMIOUD
hemourad@yahoo,fr
Université de Jijel
Département d'informatique

1

Grammaire Système générateur de langage

2

GRAMMAIRE

(SYSTÈME GÉNÉRATEUR DE LANGAGE)

- **Définition** . Une grammaire est un moyen permettant de décrire la construction des mots d'un langage. Elle a plusieurs avantages :
 - Elle permet de raisonner sur le langage ;
 - Elle permet de construire des algorithmes efficaces pour le traitement des langages ;
 - Elle facilite l'apprentissage des langages.
- **NB:** Les expressions régulières ne sont pas suffisantes pour représenter les langages

3

Exemple : Grammaire ?

Pour analyser une classe de phrases simples en français, nous allons supposer qu'une phrase est construite de la manière suivante :

PHRASE → ARTICLE SUJET VERBE ARTICLE COMPLEMENT
 et **COMPLEMENT** → NOM ADJECTIF

4

PHRASE → ARTICLE SUJET VERBE ARTICLE COMPLEMENT;

- COMPLEMENT → NOM ADJECTIF ;
- SUJET → “garçon” ou “fille”;
- VERBE → “voit” ou “mange” ou “porte”;
- ARTICLE → “un” ou “le”;
- NOM → “livre” ou “plat” ou “wagon”;
- ADJECTIF → “bleu” ou “rouge” ou “vert”;

En remplaçant les parties gauches par les parties droites nous arrivons à générer les deux phrases suivantes :

- *Le garçon voit un livre vert*
- *Une fille mange le plat bleu*

5

Génération de phrases

- PHRASE → ARTICLE SUJET VERBE ARTICLE COMPLEMENT
 - → “le” **SUJET** VERBE ARTICLE COMPLEMENT
 - → “le” “garçon” **VERBE** ARTICLE COMPLEMENT
 - → “le” “garçon” “voit” **ARTICLE** COMPLEMENT
 - → “le” “garçon” “voit” “le” **COMPLEMENT**
 - → “le” “garçon” “voit” “le” **NOM ADJECTIF**
 - → “le” “garçon” “voit” “le” “livre” **ADJECTIF**
 - → “le” “garçon” “voit” “le” “livre” “vert”

De même pour la phrase « *Une fille mange le plat bleu* »

« *Une fille mange le livre bleu* » (la syntaxe est correcte)

6

Définition formelle des grammaires

Définition : On appelle grammaire le quadruplet (V,N, X, R)

- **V** : est un ensemble fini de symboles dits **terminaux**, (vocabulaire terminal) ;
- **N** : est un ensemble fini (disjoint de V) de symboles dits **non-terminaux** (concepts) ;
- **S** : un non-terminal particulier appelé **axiome** (point de départ de la dérivation) ;
- **R** : est un ensemble de **règles de productions** de la forme $\alpha \rightarrow \beta$ tel que $\alpha \in (V + N)^+$ et $\beta \in (V + N)^*$.

La notation $\alpha \rightarrow \beta$ est appelée une dérivation et signifie que α peut être remplacé par β .

7

Exemple

PHRASE \rightarrow ARTICLE SUJET VERBE ARTICLE COMPLEMENT;

- P \rightarrow ASVAC
- o COMPLEMENT \rightarrow NOM AdJECTIF ; ----- C \rightarrow NAd
- o A \rightarrow "un" ou "le" ----- A \rightarrow un / le
- o SUJET \rightarrow "garçon" ou "fille"; ----- S \rightarrow garçon / fille
- o VERBE \rightarrow "voit" ou "mange" ou "porte"; ----- V \rightarrow voit / mange / porte
- o NOM \rightarrow "livre" ou "plat" ou "wagon"; ----- N \rightarrow livre / plat / wagon
- o AdJECTIF \rightarrow "bleu" ou "rouge" ou "vert"; ---- Ad \rightarrow bleu / rouge / vert

G=(V,N, X, R)

V={garçon, fille, voit, mange, porte, un, le, livre, plat, wagon, bleu, rouge, vert}

N={P, A, S, V, C, N, Ad}

X= P

R={P \rightarrow ASVAC; C \rightarrow NAd; S \rightarrow garçon / fille; V \rightarrow voit / mange / porte; A \rightarrow un / le; N \rightarrow livre / plat / wagon; Ad \rightarrow bleu / rouge / vert; }

8

Remarques

- On utilisera les lettres majuscules pour les non-terminaux, et les lettres minuscules pour représenter les terminaux.
- Les règles de la forme $\varepsilon \rightarrow \alpha$ sont *interdites*.
- Soit une suite de dérivations :

$w_1 \rightarrow w_2 \rightarrow w_3 \rightarrow \dots \rightarrow w_n$ alors on écrira : $w_1 \rightarrow^* w_n$.

On dit alors qu'il y a une chaîne de **dérivation** qui mène de w_1 vers w_n .

Exemple : Soit la grammaire

$G = (\{a\}, \{S\}, S, \{S \rightarrow aS \mid \varepsilon\})$. On peut construire la chaîne de dérivation suivante :

$S \rightarrow aS \rightarrow aaS \rightarrow aaaS \dots$

9

Les mots générés par une grammaire

Soit une grammaire $G = (V, N, S, R)$. On dit que le mot u appartenant à V^* est **dérivé** (ou bien **généré**) à partir de G s'il existe une suite de dérivation qui, partant de l'axiome S , permet d'obtenir u , noté ' $S \rightarrow^* u$ '

Le langage engendré par une grammaire

Le langage engendré par une grammaire G est l'ensemble de tous les mots générés par la grammaire G est noté $L(G)$.

Deux grammaires G et G' sont équivalentes si $L(G) = L(G')$.

10

Exemple : Soit la grammaire $G = (\{a, b\}, \{S, T\}, S, \{S \rightarrow aS \mid aT, T \rightarrow bT \mid b\})$.

Elle génère les mots abb et aab parce que

$S \rightarrow aT \rightarrow abT \rightarrow abb$

$S \rightarrow aS \rightarrow aaT \rightarrow aab.$

.....

$S \rightarrow aS \rightarrow aaT \rightarrow aaT \rightarrow aaaT \rightarrow \dots \rightarrow aaa\dots aT.$

On peut facilement voir alors que le langage généré par cette grammaire est : tous les mots sur $\{a, b\}$ de la forme $a^m b^n$ avec $m, n > 0$.

11

Les arbres de syntaxe de la grammaire

- Étant donnée une grammaire $G = (V, N, S, R)$, les arbres de syntaxe de G sont des arbres où les nœuds internes sont étiquetés par des symboles de N , et les feuilles étiquetés par des symboles de V , tels que, si le nœud p apparaît dans l'arbre et si la règle $p \rightarrow a_1 \dots a_n$ (a_i terminal ou non terminal) est utilisée dans la dérivation, alors le nœud p possède n fils correspondant aux symboles a_i .
- Si l'arbre syntaxique a comme racine S , alors il est dit arbre de dérivation du mot u tel que u est le mot obtenu en prenant les feuilles de l'arbre dans le sens *gauche* \rightarrow *droite* et *bas* \rightarrow *haut*.

12

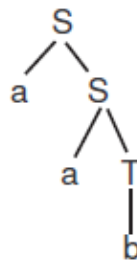
Exemple :

Soit la grammaire $G = (\{a, b\}, \{S, T\}, S, \{S \rightarrow aS \mid aT, T \rightarrow bT \mid b\})$.

Elle génère le mot *aab* selon la chaîne de dérivation

$S \rightarrow aS \rightarrow aaT \rightarrow aab$.

Ce qui donne donc l'arbre syntaxique suivant :



13

Classification de Chomsky

- La classification de Chomsky est un moyen permettant de *maîtriser la complexité* des langages ainsi que de celle des grammaires qui les génèrent.
- En effet, certains langages sont simples et peuvent être décrits par des grammaires facilement compréhensibles. Cependant, il existe des langages d'une telle complexité que les grammaires qui les génèrent sont trop difficile à appréhender (par exemple, $\{a^n \mid n \text{ est premier}\}$).

→ comment mesurer la complexité d'une grammaire ou d'un langage ?

14

- Noam Chomsky remarquer que la complexité d'une grammaire (et celle du langage aussi) **dépend** de la *forme des règles* de production
- Chomsky a ainsi proposé **quatre** classes (hiérarchiques) de grammaires (et de langages) de sorte qu'**une grammaire de type i génère un langage de type j tel que $j \geq i$** .

15

Soit $G = (V, N, S, R)$ une grammaire, les classes de grammaires de Chomsky sont :

- **Type 3** ou **grammaire régulière** (à droite 1) : toutes les règles de production sont de la forme $\alpha \rightarrow \beta$ où $\alpha \in N$ et $\beta = aB$ / tel que $a \in V^*$ et $B \in N \cup \{\epsilon\}$;
- **Type 2** ou **grammaire hors-contexte** : toutes les règles de production sont de la forme $\alpha \rightarrow \beta$ où $\alpha \in N$ et $\beta \in (V + N)^*$;
- **Type 1** ou **grammaire contextuelle** : toutes les règles sont de la forme $\alpha \rightarrow \beta$ tel que $\alpha \in (N + V)^+$, $\beta \in (V + N)^*$ et $|\alpha| \leq |\beta|$. De plus, si ϵ apparaît à droite alors la partie gauche doit seulement contenir S (l'axiome).

On peut aussi trouver la définition : toutes les règles sont de la forme $\alpha B \beta \rightarrow \alpha \omega \beta$ tel que $\alpha, \beta \in (V + N)^*$, $B \in X$ et $\omega \in (V + N)^*$

- **Type 0** : aucune restriction. Toutes les règles sont de la forme : $\alpha \rightarrow \beta$, $\alpha \in (V + N)^+$, $\beta \in (V + N)^*$

16

- Il existe une relation d'inclusion entre les types de grammaires :

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

- Pour trouver la classe d'un langage on procède cependant comme suit :
 - Chercher une grammaire de type 3 qui le génère, si elle existe, le langage est de type 3 (ou **régulier**)
 - Sinon, chercher une grammaire de type 2 qui le génère, si elle existe, le langage est de type 2 (ou **algébrique**)
 - Sinon, chercher une grammaire de type 1 qui le génère, si elle existe, le langage est de type 1 (ou **contextuel**)
 - Sinon, le langage est de type 0.

17

Exemple:

$G_1 = (\{S, A, B\}, \{0, 1\}, P_1, S)$	$G_2 = (\{S, A, B\}, \{0, 1\}, P_2, S)$
$P_1 = \left\{ \begin{array}{l} S \rightarrow \varepsilon 0A B1 \\ A \rightarrow 1 S1 \\ B \rightarrow 0 0S \end{array} \right\}$	$P_2 = \left\{ \begin{array}{l} S \rightarrow 1S 0A \\ A \rightarrow 1S 0B 0 \\ B \rightarrow 0 1 0B 1B \end{array} \right\}$
$G_3 = (\{S\}, \{0, 1\}, P_3, S)$	$G_4 = (\{S, A, B, C\}, \{0, 1\}, P_4, S)$
$P_3 = \{ S \rightarrow \varepsilon 0 01S 1S \}$	$P_4 = \left\{ \begin{array}{l} S \rightarrow AB \\ A \rightarrow \varepsilon 0 \\ B \rightarrow 10B C \\ C \rightarrow \varepsilon 1 \end{array} \right\}$

18

Le tableau suivant résume les différentes classes de grammaires, les langages générés et les types d'automates qui les reconnaissent :

Grammaire	Langage	Automate
Type 0	Récursivement énumérable	Machine de Turing
Type 1 ou contextuelle	Contextuel	Machine de Turing à borne linéaire
Type 2 ou hors-contexte	Algébrique	Automate à pile
Type 3 ou régulière	Régulier ou rationnel	Automate à états fini

Exercice:

Soient les grammaires $G_i = (\{a, b, c\}, \{S, A, B, C\}, S, P_i)$, ($i=1,...,8$) ; où les P_i sont :

- $P_1 : S \rightarrow aA \mid bB ; A \rightarrow a \mid ab ; B \rightarrow b \mid cb$
- $P_2 : S \rightarrow bA ; A \rightarrow aA \mid \epsilon$
- $P_3 : S \rightarrow aAb \mid \epsilon ; A \rightarrow aSb ; Ab \rightarrow \epsilon$
- $P_4 : S \rightarrow AB \mid aS \mid a ; A \rightarrow Ab \mid \epsilon ; B \rightarrow AS$
- $P_5 : S \rightarrow 0S \mid 1B ; B \rightarrow 0C \mid 1S \mid \epsilon, C \rightarrow 0B \mid 1C$
- $P_6 : S \rightarrow 0B ; B \rightarrow S1 ; S \rightarrow \epsilon$
- $P_7 : S \rightarrow \epsilon \mid a \mid abS \mid bS$
- $P_8 : S \rightarrow AB ; A \rightarrow \epsilon \mid 0 ; B \rightarrow 10B \mid C ; C \rightarrow \epsilon \mid 1$

Pour chacune des grammaires G_i ($i=1,...,8$) ; donner le type de celle-ci, puis trouver le langage engendré par chacune d'elles.

Passage de la grammaire vers l'automate

soit $G = (V, N, S, R)$ une grammaire régulière à droite, si toutes les règles de production sont de la forme : $A \rightarrow aB$ ou $A \rightarrow B$ ($A, B \in N, a \in V \cup \{\epsilon\}$) alors il suffit d'appliquer l'algorithme suivant :

1. Associer un état à chaque non terminal de N ;
2. L'état initial est associé à l'axiome ;
3. Pour chaque règle de production de la forme $A \rightarrow \epsilon$, l'état q_A est final ;
4. Pour chaque règle de production de la forme $A \rightarrow a$ ($a \in V$), alors créer un nouvel état final q_f et une transition partant de l'état q_A vers l'état q_f avec l'entrée a ;
5. Pour chaque règle $A \rightarrow aB$ alors créer une transition partant de q_A vers l'état q_B en utilisant l'entrée a ;
6. Pour chaque règle $A \rightarrow B$ alors créer une ϵ -transition partant de q_A vers l'état q_B ;

21

Langages hors-contexte (algébriques) et Automates à pile

22

Certains langages ne peuvent pas être décrits par une grammaire régulière, et ne peuvent donc pas être reconnus par un automate fini (par exemple le langage $\{a^n b^n / n > 0\}$).

On étudie dans ce chapitre une classe de langages plus générale que celle des langages réguliers : la classe des **langages hors-contexte**, décrits par des grammaires hors-contexte et reconnus par des **automates à pile**.

23

- **Grammaire hors-contexte :**

$G = (T, N, S, R)$ est une grammaire hors-contexte si toutes les règles de R sont de la forme $A \rightarrow w$ avec $A \in N$ et $w \in (N \cup T)^*$.

- **Langage hors-contexte :** On appelle langage hors-contexte un langage généré par une grammaire hors contexte.

24

25

Simplification des grammaires hors-contextes

1. Les grammaires propres

Une grammaire hors-contexte (V, N, S, R) est dite **propre** si elle vérifie :

- $\forall A \rightarrow u \in R : u \neq \varepsilon$ ou $A = S$;
- $\forall A \rightarrow u \in R : S$ ne figure pas dans u ;
- $\forall A \rightarrow u \in R : u \notin N$;
- Tous les non terminaux sont utiles, c'est-à-dire qu'ils vérifient :
 - $\forall A \in N : A$ est atteignable depuis $S : \exists \alpha, \beta \in (N + V)^* : S \xrightarrow{*} \alpha A \beta$;
 - $\forall A \in N : A$ est productif : $\exists w \in V^* : A \xrightarrow{*} w$.

- Il est toujours possible de trouver une grammaire propre pour toute grammaire hors contexte. En effet, on procède comme suit :

26

1. **Rajouter** une nouvelle règle $S' \rightarrow S$ tel que S' est le nouvel **axiome** ;
2. **Éliminer** les règles $A \rightarrow \varepsilon$:
 - Calculer l'ensemble $E = \{A \in N \cup \{S'\} \mid A \rightarrow^* \varepsilon\}$;
 - Pour tout $A \in E$, pour toute règle $B \rightarrow \alpha A \beta$ de R
 - Rajouter la règle $B \rightarrow \alpha \beta$
 - Enlever les règles $A \rightarrow \varepsilon$;
3. **Éliminer** les règles $A \rightarrow^* B$, on applique la procédure suivante sur R privée de $S' \rightarrow \varepsilon$:
 - Calculer toutes les paires (A, B) tel que $A \rightarrow^* B$
 - Pour chaque paire (A, B) trouvée
 - Pour chaque règle $B \rightarrow u_1 \mid \dots \mid u_n$ rajouter la règle $A \rightarrow u_1 \mid \dots \mid u_n$
 - Enlever toutes les règles $A \rightarrow B$
4. **Supprimer** tous les non-terminaux **non-productifs**
5. **Supprimer** tous les non-terminaux **non-atteignables**