Context Free Grammars
Algorithme CYK

Grammaire hors-contexte : Exemple

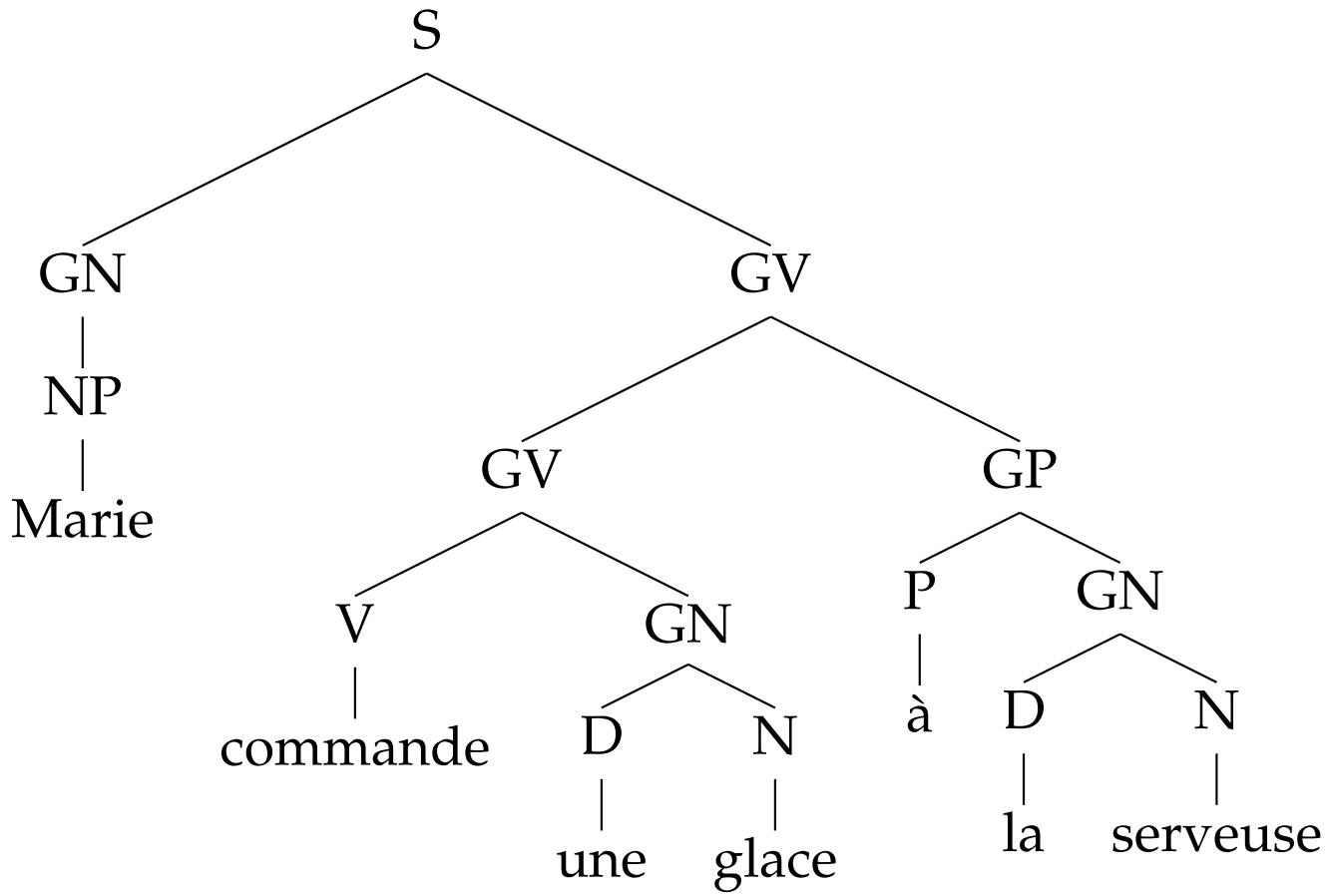
- Alphabet non terminal ($\mathcal{N}$) : $\{S, GN, GV, GP, D, N, NP, V, P\}$
- Alphabet terminal ($\mathcal{T}$) :
  $\{Marie, glace, commande, à, la, serveuse, fraise\}$
- Axiome (N^1) : S
- Règles ($\mathcal{R}$) :

S	$\rightarrow$	$GN\ GV$	GV	$\rightarrow$	$V\ GN$	V	$\rightarrow$	$commande$
GN	$\rightarrow$	$D\ N$	GV	$\rightarrow$	$GV\ GP$	NP	$\rightarrow$	$Marie$
GN	$\rightarrow$	NP	N	$\rightarrow$	$glace$	P	$\rightarrow$	$à$
GN	$\rightarrow$	$GN\ GP$	N	$\rightarrow$	$fraise$	D	$\rightarrow$	la
GP	$\rightarrow$	$P\ GN$	N	$\rightarrow$	$serveuse$			

Dérivation : exemple

	proto mot	*règle utilisée*
	S	$S \rightarrow GN\ GV$
$\Rightarrow$	$GN\ GV$	$GN \rightarrow NP$
$\Rightarrow$	$NP\ GV$	$NP \rightarrow Marie$
$\Rightarrow$	$Marie\ GV$	$GV \rightarrow GV\ GP$
$\Rightarrow$	$Marie\ GV\ GP$	$GV \rightarrow V\ GN$
$\Rightarrow$	$Marie\ V\ GN\ GP$	$V \rightarrow commande$
$\Rightarrow$	$Marie\ commande\ GN\ GP$	$GN \rightarrow DN$
	$\ldots$	
$\Rightarrow$	$Marie\ commande\ une\ glace\ à\ la\ serveuse$	

Arbre de dérivation : exemple



Grammaires hors-contexte

Une grammaire hors-contexte est composée de :

- Un alphabet non terminal $\mathcal{N} = \{N^1 \dots N^n\}$
- Un alphabet terminal $\mathcal{T} = \{t^1 \dots t^m\}$
- Un axiome N^1
- Un ensemble de règles $\mathcal{R}$ de la forme :
  $N^i \rightarrow \alpha$
  avec $\alpha \in (\mathcal{N} \cup \mathcal{T})^*$

Notation

Pour alléger les notations, on note :

$$\alpha \to \beta_1 | \beta_2 | \ldots | \beta_n$$

les n règles :

$$\alpha \to \beta_1 \ , \ \alpha \to \beta_2 \ , \ldots , \ \alpha \to \beta_n$$

Proto-mots d'une grammaire

Les proto-mots d'une grammaire $G = \langle \mathcal{N}, \mathcal{T}, \mathcal{R}, N^1 \rangle$ sont des mots construits sur l'alphabet $\mathcal{N} \cup \mathcal{T}$.

On les définit récursivement de la façon suivante :

- N^1 est un proto-mot de G
- si $\alpha\beta\gamma$ est un proto-mot de G et $\beta \to \delta \in \mathcal{R}$ alors $\alpha\delta\gamma$ est un proto-mot de G.

Un proto-mot de G ne contenant aucun symbole non terminal est appelé un mot généré par G.

Le langage généré par G, noté $L(G)$ est l'ensemble des mots générés par G.

Dérivation

- L'opération qui consiste à générer un proto-mot $\alpha\delta\gamma$ à partir d'un proto-mot $\alpha\beta\gamma$ et d'une règle de production r de la forme $\beta \to \delta$ est appelée l'opération de dérivation.
  Elle se note à l'aide d'une double flèche :

$$\alpha\beta\gamma \Rightarrow \alpha\delta\gamma$$

- On note $\alpha \overset{k}{\Rightarrow} \beta$ pour indiquer que β se dérive de α en k étapes.
- On définit aussi les deux notations $\overset{+}{\Rightarrow}$ et $\overset{*}{\Rightarrow}$ de la façon suivante :
  - $\alpha \overset{+}{\Rightarrow} \beta \equiv \alpha \overset{k}{\Rightarrow} \beta$ avec $k > 0$
  - $\alpha \overset{*}{\Rightarrow} \beta \equiv \alpha \overset{k}{\Rightarrow} \beta$ avec $k \geq 0$

Langage généré par une grammaire

- $L(G)$ est défini de la façon suivante :

$$L(G) = \{m \in \mathcal{T}^* | S \overset{+}{\Rightarrow} m\}$$

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

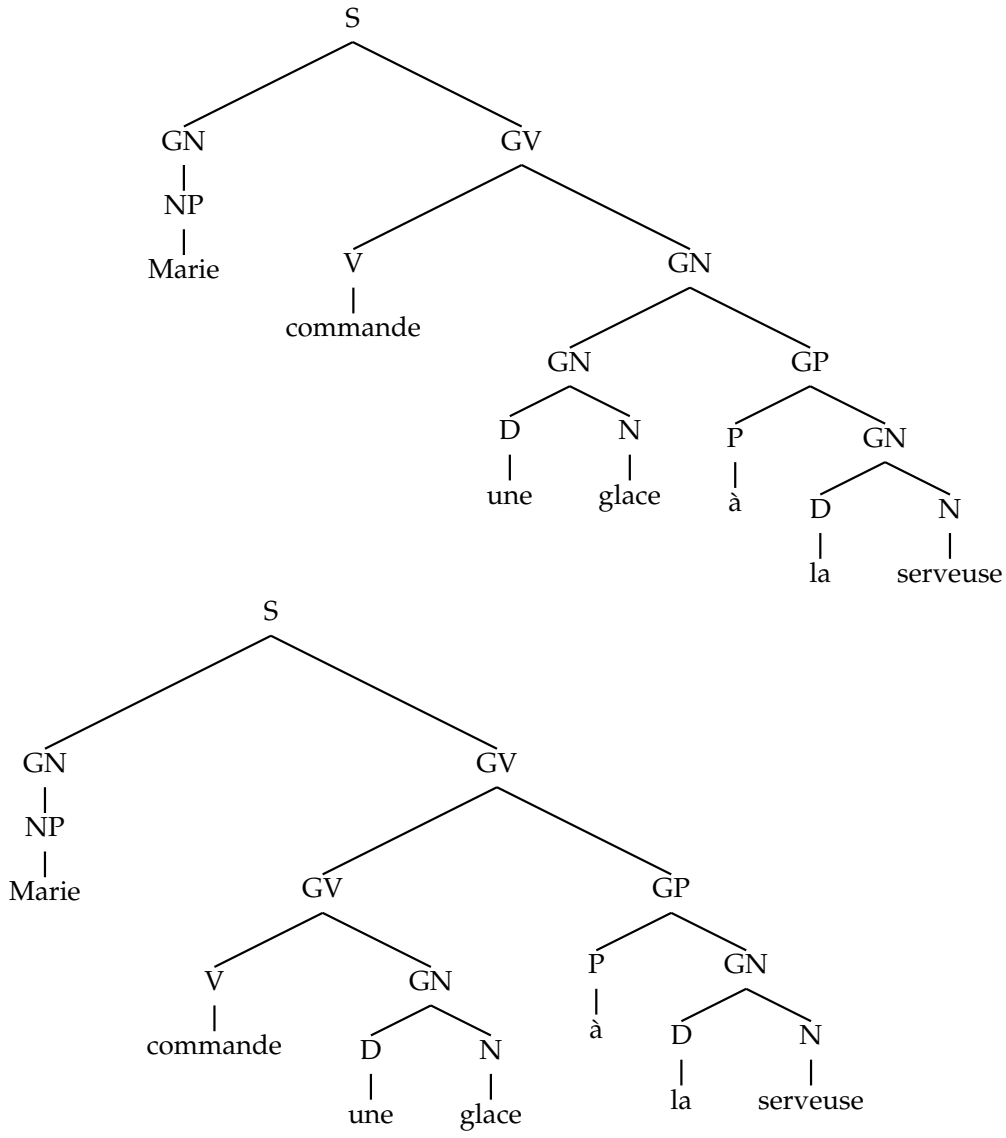
Arbre de dérivation

- Un arbre de dérivation pour G ($G = \langle \mathcal{N}, \mathcal{T}, \mathcal{R}, N^1 \rangle$) est un arbre ordonné et étiqueté dont les étiquettes appartiennent à l'ensemble $\mathcal{N} \cup \mathcal{T} \cup \{\varepsilon\}$.
- Si un nœud de l'arbre est étiqueté par le non terminal A et ses fils sont étiquetés $X_1, X_2, ..., X_n$ alors la règle $A \rightarrow X_1, X_2, ..., X_n$ appartient à $\mathcal{R}$.
- Un arbre de dérivation indique les règles qui ont été utilisées dans une dérivation, mais pas l'ordre dans lequel elles ont été utilisées.

Ambiguïté

Une grammaire G est **ambiguë** s'il existe au moins un mot m dans $L(G)$ auquel correspond plus d'un arbre de dérivation.

G est ambiguë



Analyse syntaxique

- Processus consistant à produire tous les arbres qu'une grammaire G permet d'associer à un mot de $L(G)$.
- Le nombre d'arbres qu'une grammaire peut associer à un mot de longueur n peut être une fonction exponentielle de n (Exemple : $X \to XX$, $X \to a$, $X \to \varepsilon$)
- L'analyse syntaxique ne peut être effectuée par énumération des arbres, elle serait de complexité exponentielle.
- Il existe au moins deux algorithmes d'analyse syntaxique fondés sur la programmation dynamique permettant d'analyser toute grammaire en $\mathcal{O}(n^3)$ où n est la longueur du mot à analyser :
  - l'algorithme de Cocke Younger Kasami
  - l'algorithme de Earley

L'algorithme de Cocke Younger Kasami

Entrée :

- une grammaire hors-contexte sous forme normale de Chomsky sans ϵ-transitions.
- une phrase à analyser $w = w_1 \dots w_n \in \mathcal{T}^*$.

Sortie :

- Une table d'analyse T pour w telle que la case $t_{i,j}$ contient $A \in \mathcal{N}$ si et seulement si $A \stackrel{+}{\Rightarrow} w_i \dots w_j$

Forme normale de Chomsky

- Une grammaire hors-contexte est en forme normale de Chomsky si toutes ses règles sont de la forme :

$$A \to BC \text{ ou } A \to a$$

  avec $A, B \in \mathcal{N}$ et $a \in \mathcal{T}$.
- De plus, on autorise la règle $S \to \varepsilon$ si S est l'axiome de la grammaire et s'il n'apparaît jamais dans la partie droite d'une règle.
- Tout langage hors-contexte peut être généré par une grammaire hors-contexte en forme normale de Chomsky.
- Il existe un algorithme permettant de transformer une grammaire hors-contexte G en une grammaire hors-contexte G' sous forme normale de Chomsky telle que $L(G') = L(G)$

Algorithme CYK

pour $i = 1$ **à** n **faire** { INITIALISATION }
  $t_{i,i} = \{A | A \to w_i\}$

pour $j = 1$ **à** n **faire**
  pour $i = j - 1$ **à** 1 **faire**
    pour $k = i$ **à** $j - 1$ **faire**
      $t_{i,j} = t_{i,j} \cup \{A | A \to BC\}$
      avec $B \in t_{i,k}$ **et** $C \in t_{k+1,j}$

Exemple

- xx