PCFG

Alexis Nasr

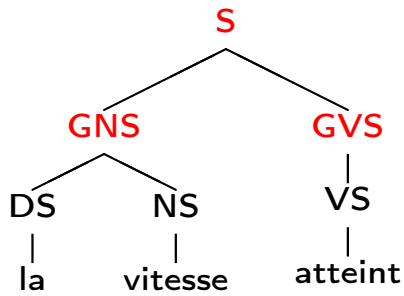
Plan

- Motivation
- Langages et grammaires hors-contextes
- Forme normale de Chomsky
- L'algorithme de Cocke Kasami Younger
- Les grammaires hors contexte probabilistes
- Calcul de la probabilité d'une phrase
- Construction de l'arbre le plus probable
- Estimation des probabilités de la grammaire
- Limites des grammaires hors contexte probabilistes

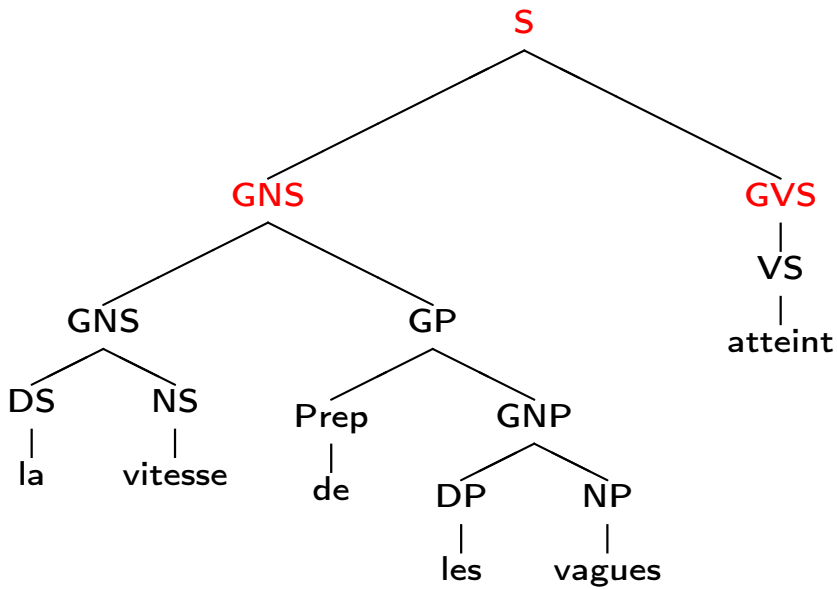
Limites des modèles *n*-gram

- la vitesse atteint . . .
- la vitesse des vagues atteint . . .
- la vitesse des vagues sismiques atteint . . .
- la vitesse des grandes vagues sismiques atteint . . .

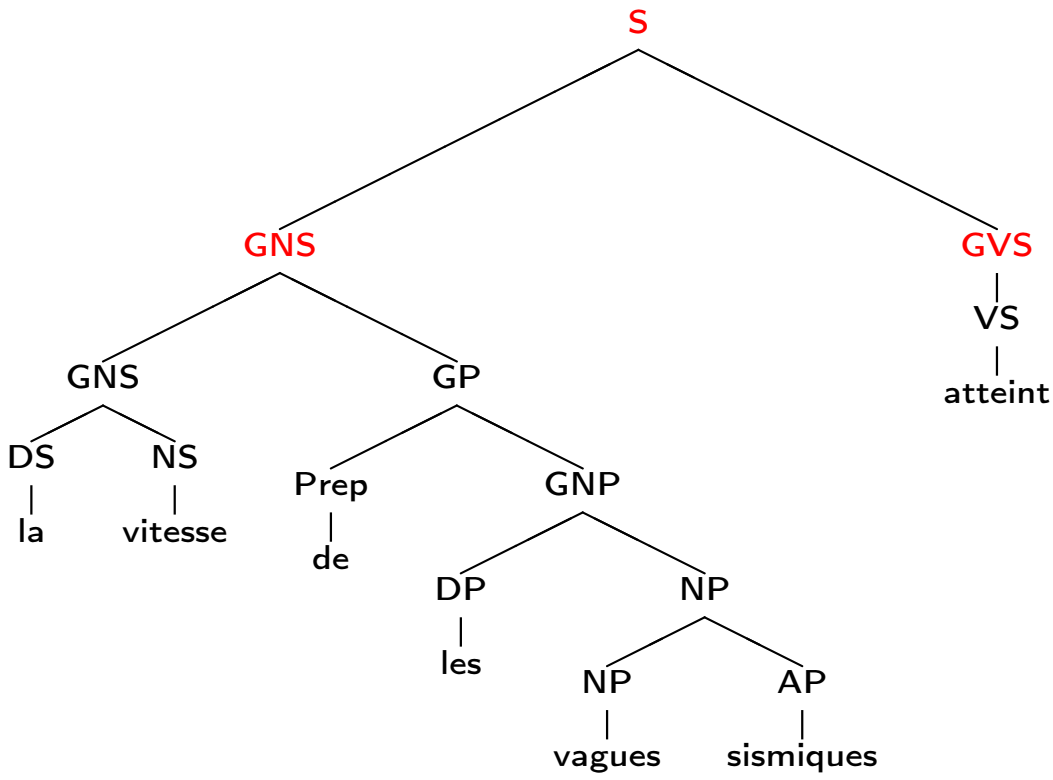
Structure syntaxique de la phrase



Structure syntaxique de la phrase



Structure syntaxique de la phrase



Grammaire probabiliste vue comme un modèle de langage

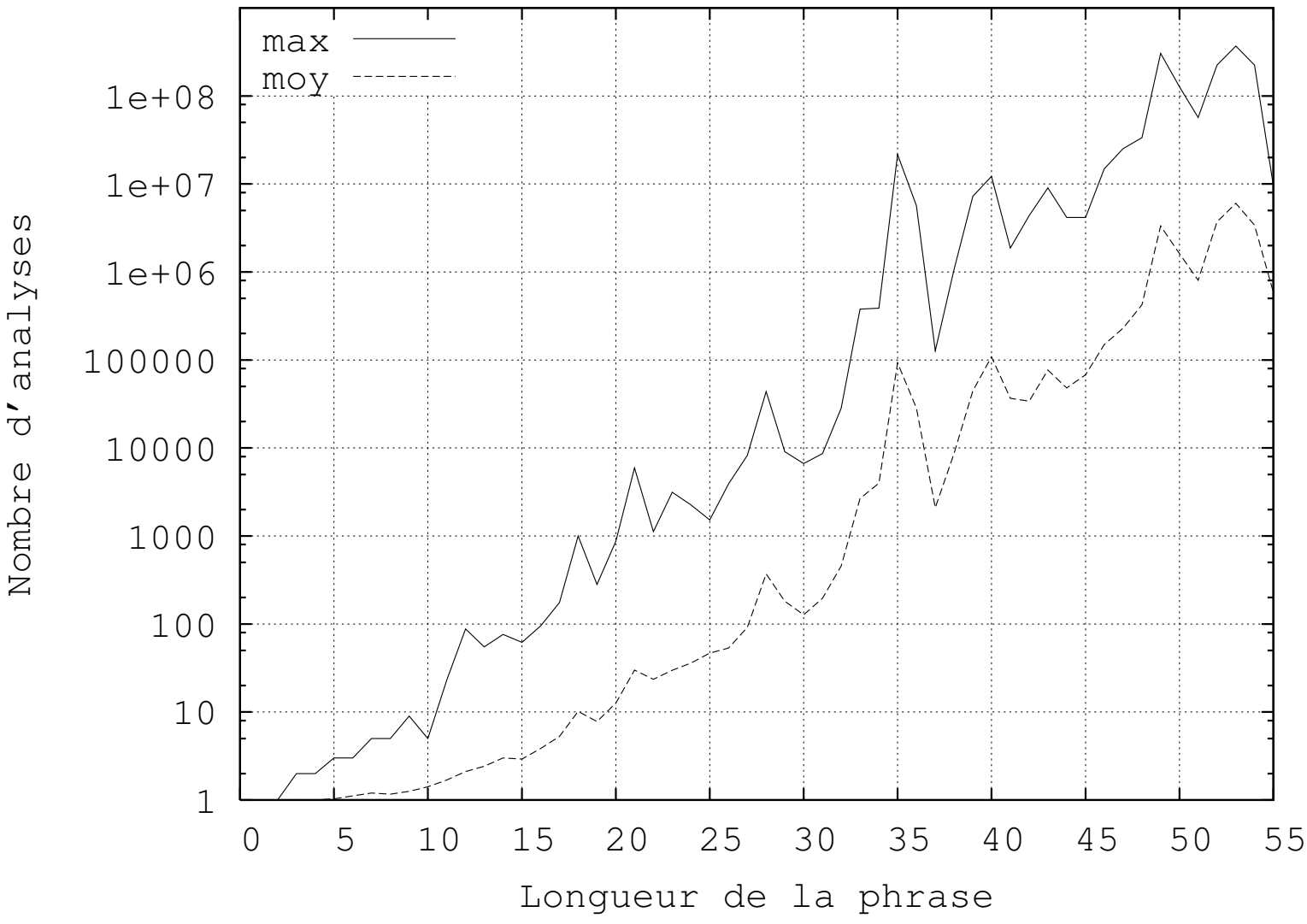
- L'accord entre le sujet et le verbe est modélisé par une même règle dans les différentes phrases.
- La règle est insensible à la longueur du groupe nominal
- La grammaire peut facilement attribuer une meilleure probabilité à la phrase correcte

$$P(S \rightarrow GNS \quad GVS) > P(S \rightarrow GNS \quad GVP)$$

- La grammaire probabiliste peut être utilisée comme modèle de langage.
- Elle permet de calculer la probabilité de toute phrase $S \in L(G)$

$$P(S|G)$$

Désambiguïsation syntaxique



Grammaire proba. vue comme un modèle de désambiguïsation

- Parmi les analyses automatiques d'une phrase, la très grande majorité est aberrante.
- On aimerait pouvoir classer les analyses selon leur plausibilité.
- La grammaire probabiliste peut être utilisée comme modèle de désambiguïsation
- Elle permet de calculer la probabilité d'un arbre T étant donné une phrase S :

$$P(T|S,G)$$

- et de retrouver l'arbre le plus probable :

$$\hat{T} = \arg\max_{T} P(T|S,G)$$

Notions de base de la théorie des langages

- Les symboles sont des éléments indivisibles qui vont servir de briques de base pour construire des mots.
- Un *alphabet* est un *ensemble* fini de symboles. On désigne conventionnellement un alphabet par la lettre grecque Σ.
- Une suite de symboles, appartenant à un alphabet Σ, mis bout à bout est appelé un *mot* (ou une *chaîne*) sur Σ. Le mot de longueur zéro est noté ε.
- L'ensemble de tous les mots que l'on peut construire sur un alphabet Σ est noté Σ^*.
- Un *langage* sur un alphabet Σ est un ensemble de mots construits sur Σ. Tout langage défini sur Σ est donc une partie de Σ^*.

Grammaires de réécriture

Une grammaire de réécriture est un 4-uplet $\langle N, \Sigma, P, S \rangle$ où :

- N est un ensemble de symboles non terminaux, appelé l'alphabet non terminal.
- Σ est un ensemble de symboles terminaux, appelé l'alphabet terminal, tel que N et Σ soient disjoints.
- P est un sous ensemble fini de :

$$(N \cup \Sigma)^* N (N \cup \Sigma)^* \times (N \cup \Sigma)^*$$

  un élément (α, β) de P, que l'on note $\alpha \to \beta$ est appelé une règle de production ou règle de réécriture.
- S est un élément de N appelé l'*axiome* de la grammaire.

Proto-phrases d'une grammaire

Les proto-phrases d'une grammaire $G = \langle N, \Sigma, P, S \rangle$ sont des mots construits sur l'alphabet $\Sigma \cup N$, on les définit récursivement de la façon suivante :

- S est une proto-phrase de G
- si $\alpha\beta\gamma$ est une proto-phrase de G et $\beta \to \delta \in P$ alors $\alpha\delta\gamma$ est une proto-phrase de G.

Une proto-phrase 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 une proto-phrase $\alpha\delta\gamma$ à partir d'une proto-phrase $\alpha\beta\gamma$ et d'une règle de production r de la forme $\beta \rightarrow \delta$ 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 :
  - $\alpha \overset{+}{\Rightarrow} \beta \equiv \alpha \overset{k}{\Rightarrow} \beta \text{ avec } k > 0$
  - $\alpha \overset{*}{\Rightarrow} \beta \equiv \alpha \overset{k}{\Rightarrow} \beta \text{ avec } k \geq 0$

Dérivation

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

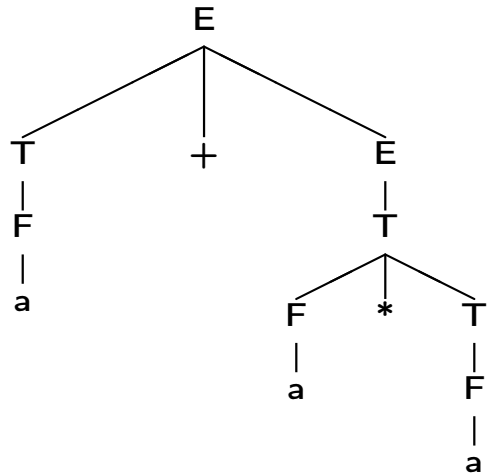
$$L(G) = \{m \in \Sigma^* | S \stackrel{*}{\Rightarrow} m\}$$

- Deux grammaires G et G' sont équivalentes si les langages $L(G)$ et $L(G')$ sont identiques.

Sens de dérivation

- Les proto-phrases générées lors d'une dérivation peuvent comporter plus d'un symbole non terminal :
  $\underline{E} \Rightarrow T + \underline{E} \Rightarrow \underline{T} + T \Rightarrow F + \underline{T} \Rightarrow F + \underline{F} * T \Rightarrow$
  $F + a * \underline{T} \Rightarrow \underline{F} + a * F \Rightarrow a + a * \underline{F} \Rightarrow a + a * a$
- Dérivation droite : on réécrit le non terminal le plus à droite :
  $\underline{E} \Rightarrow T + \underline{E} \Rightarrow T + \underline{T} \Rightarrow T + F * \underline{T} \Rightarrow T + F * \underline{F} \Rightarrow$
  $T + \underline{F} * a \Rightarrow \underline{T} + a * a \Rightarrow \underline{F} + a * a \Rightarrow a + a * a$
- Dérivation gauche : on réécrit le non terminal le plus à gauche :
  $\underline{E} \Rightarrow \underline{T} + E \Rightarrow \underline{F} + E \Rightarrow a + \underline{E} \Rightarrow a + \underline{T} \Rightarrow a + \underline{F} * T \Rightarrow$
  $a + a * \underline{T} \Rightarrow a + a * \underline{F} \Rightarrow a + a * a$

Arbre de dérivation



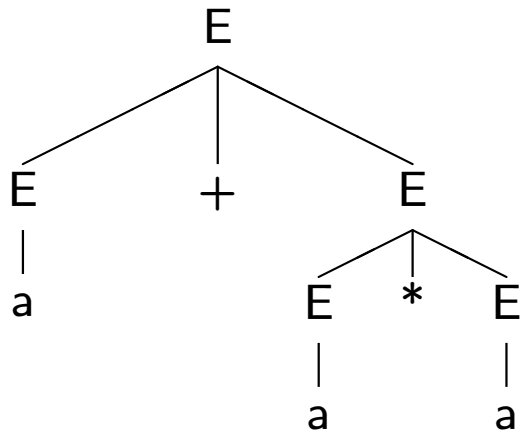
Un arbre de dérivation pour G ($G = \langle N, \Sigma, P, S \rangle$) est un arbre ordonné et étiqueté dont les étiquettes appartiennent à l'ensemble $N \cup \Sigma \cup \{\varepsilon\}$. Si un nœud de l'arbre est étiqueté par le non terminal A et ses fils sont étiquetés $X_1, X_2, ..., X_n$ alors la règle $A \rightarrow X_1, X_2, ..., X_n$ appartient à P.

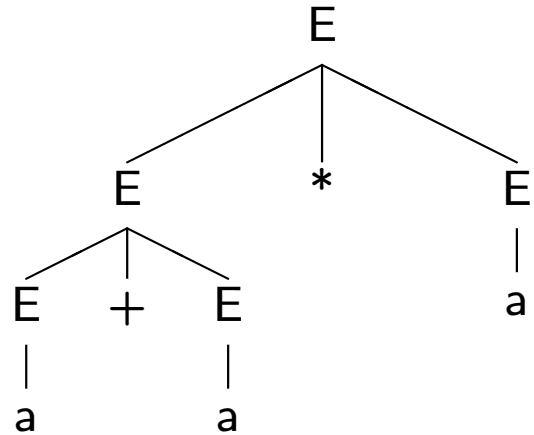
Arbre de dérivation

- Un arbre de dérivation indique les règles qui ont été utilisées dans une dérivation, mais pas l'ordre dans lequel elles ont été utilisées.
- A un arbre de dérivation correspondent une seule dérivation droite et une seule dérivation gauche.

Ambiguïté

Une grammaire G est **ambigüe** s'il existe au moins une chaîne c dans $L(G)$ à laquelle correspond plus d'un arbre de dérivation.
Exemple : $E \rightarrow E + E | E * E | a$





Types de règles

Les grammaires peuvent être classées en fonction de la forme de leurs règles de production. On définit cinq types de règles de production :

- règles régulières à gauche :
  Une règle est régulière à gauche si et seulement si elle est de la forme $A \to xB$ ou $A \to x$ avec $A, B \in N$ et $x \in \Sigma^*$.
- règles régulières à droite :
  Une règle est régulière à gauche si et seulement si elle est de la forme $A \to Bx$ ou $A \to x$ avec $A, B \in N$ et $x \in \Sigma^*$.
- règles hors-contexte :
  Une règle $A \to \alpha$ est un règle hors-contexte si et seulement si : $A \in N$ et $\alpha \in (N \cup \Sigma)^*$

Types de règles

- règles contextuelles :
  Une règle $\alpha \to \beta$ est une règle contextuelle si et seulement si : $\alpha = gAd$ et $\beta = gBd$ avec $g, d, B \in (N \cup \Sigma)^*$ et $A \in N$ le nom "contextuelle" provient du fait que A se réecrit B uniquement dans le contexte g_d.
- règles sans restrictions Une règle $\alpha \to \beta$ est une règle sans restriction si et seulement si : $|\alpha| \geq 1$

Type d'une grammaire

une grammaire G est :

- *régulière* si elle est régulière à droite ou régulière à gauche. Une grammaire est régulière à gauche si *toutes* ses règles sont régulières à gauche et une grammaire est régulière à droite si *toutes* ses règles sont régulières à droite.
- *hors contexte* si toutes ses règles de production sont hors contexte.
- *dépendante du contexte* si toutes ses règles de production sont dépendantes du contexte.
- *sans restrictions* si toutes ses règles de production sont sans restrictions.

Les types de grammaires définis ci-dessus forment une hiérarchie appelée *hiérarchie de Chomsky*

Type d'un langage

Un langage pouvant être généré par une grammaire de type x et pas par une grammaire d'un type supérieur dans la hiérarchie, est appelé un langage de type x.

Exemples

- langages réguliers
  $L = a^n$, avec $n \geq 0$
  $G = \langle \{S\}, \{a\}, \{S \rightarrow aS|\varepsilon\}, S \rangle$
- langages hors contextes
  $L = a^n b^n$, avec $n \geq 0$
  $G = \langle \{S\}, \{a, b\}, \{S \rightarrow aSb|\varepsilon\}, S \rangle$

  $L = mm$ avec $m \in \Sigma^*$ (langage mirroir)
  $G = \langle \{S\}, \{a, b\}, \{S \rightarrow aSa|bSb|aa|bb\}, S \rangle$
- langages contextuels
  $L = a^n b^n c^n$, avec $n \geq 0$
  $G = \langle \{S, S_1, S_2\}, \{a, b, c\}, \{S \rightarrow aS_1c, S_1 \rightarrow b|SS_2, cS_2 \rightarrow S_2c, bS_2 \rightarrow bb\}, S \rangle$.

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 $A, B \in N$ et $a \in \Sigma$. 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.

Conversion en forme normale de Chomsky

1. Création d'un nouvel axiome. On crée un nouveau symbole S_0 et on ajoute la règle $S_0 \to S$. Cette modification garantit que l'axiome n'apparaît pas dans une partie droite de règle.
2. Elimination des règles-ε. On élimine une règle de la forme $A \to \varepsilon \in P$, pour $A \neq S_0$ puis, pour toute occurrence de A dans une règle de P, on ajoute une nouvelle règle dans laquelle cette occurrence de A a été éliminée. La règle $X \to \alpha A \beta A \gamma$, par exemple, provoquera l'ajout des trois règles suivantes : $X \to \alpha \beta A \gamma$, $X \to \alpha A \beta \gamma$ et $X \to \alpha \beta \gamma$. Si $X \to A \in P$ alors on ajoute $X \to \varepsilon$ à moins que cette dernière n'ait déjà été éliminée. On recommence cette étape tant que P possède des règles-ε.

Conversion en forme normale de Chomsky

1. Elimination des règles $A \to B$. On élimine une règle de la forme $A \to B$. Pour toute règle de la forme $B \to \alpha$, on ajoute une règle $A \to \alpha$ à moins qu'il ne s'agisse d'une règle déjà éliminée. On recommence cette étape tant que P possède des règles de la forme $A \to B$.
2. Transformation des règles restantes. Toute règle de la forme $A \to \alpha_1 \alpha_2 \dots \alpha_k$ avec $k \geq 3$ et $\alpha_i \in \Sigma \cup N$, est remplacée par les règles $A \to \alpha_1 A_1$, $A_1 \to \alpha_2 A_2$, $\dots$, $A_{k-2} \to \alpha_{k-1} \alpha_k$ où $A_1 \dots A_k$ sont de nouveaux non terminaux. Si $k \geq 2$, on remplace tout symbole terminal α_i des règles précédentes par un nouveau symbole non terminal U_i et on ajoute la règle $U_i \to \alpha_i$

Exemple

1. Création d'un nouvel axiome.
   $S \to ASA|aB$
   $A \to B|S$
   $B \to b|\varepsilon$
   $S_0 \to S$
   $S \to ASA|aB$
   $A \to B|S$
   $B \to b|\varepsilon$
2. Elimination des règles-ε.
   $S_0 \to S$
   $S \to ASA|aB|a$
   $A \to B|S|\varepsilon$
   $B \to b$
   $S_0 \to S$
   $S \to ASA|aB|a|SA|AS|S$
   $A \to B|S$
   $B \to b$

Exemple

1. Elimination des règles $A \to B$.

   Elimination de $S \to S$ à gauche et de $S_0 \to S$ à droite :
   $S_0 \to S$
   $S \to ASA|aB|a|SA|AS$
   $A \to B|S$
   $B \to b$
   $S_0 \to ASA|aB|a|SA|AS$
   $S \to ASA|aB|a|SA|AS$
   $A \to B|S$
   $B \to b$

   Elimination de $A \to B$ à gauche et de $A \to S$ à droite :
   $S_0 \to ASA|aB|a|SA|AS$
   $S \to ASA|aB|a|SA|AS$
   $A \to S|b$
   $B \to b$
   $S_0 \to ASA|aB|a|SA|AS$
   $S \to ASA|aB|a|SA|AS$

Exemple

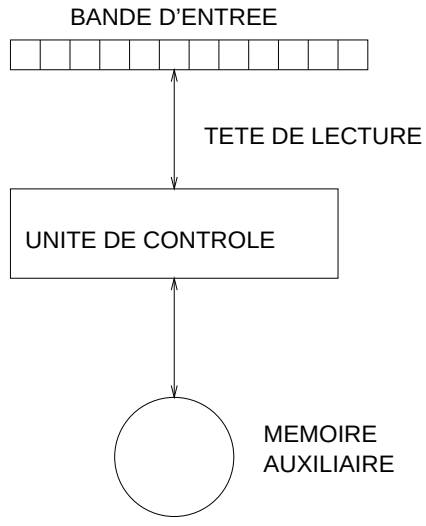
1. Transformation des règles restantes.

$$
\begin{aligned}
S_0 &\to AA_1|UB|a|SA|AS \\
S &\to AA_1|UB|a|SA|AS \\
A &\to b|AA_1|UB|a|SA|AS \\
A_1 &\to SA \\
U &\to A \\
B &\to b
\end{aligned}
$$

Grammaire v/s Reconnaisseur

- Une **grammaire** d'un langage L permet de générer toutes les chaînes appartenant à L.
- Un **reconnaisseur** pour un langage L est un programme qui prend en entrée une chaîne c et répond *oui* si c appartient à L et *non* sinon.
- Pour chaque classe de grammaire, il existe une classe de reconnaisseurs qui définit la même classe de langages.
  - Grammaires régulières $\Longleftrightarrow$ Automates à états finis
  - Grammaires hors contexte $\Longleftrightarrow$ Automates à pile

Reconnaisseur



Configuration et mouvement

- **Configuration** d'un reconnaisseur :
  - Etat de l'unité de contrôle
  - Contenu de la bande d'entrée et position de la tête
  - Contenu de la mémoire
- **Mouvement** : passage d'une configuration à une autre ($C_1 \vdash C_2$)

L'unité de contrôle est dite **déterministe** si à toute configuration correspond au plus un mouvement. S'il peut exister plus d'un mouvement, elle est dite **non déterministe**.

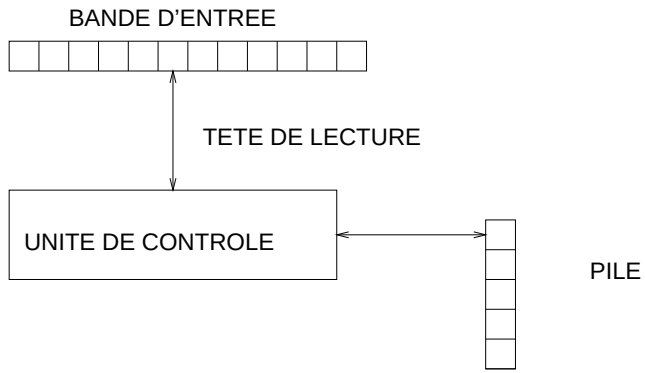
Configurations

- **configuration initiale**
  - L'unité de contrôle est dans un état initial
  - La tête est au début de la bande
  - La mémoire contient un élément initial.
- **configuration d'acceptation**
  - L'unité de contrôle est dans un état d'acceptation
  - La tête de lecture est à la fin de la bande
  - La mémoire se trouve dans un état d'acceptation.

Reconnaissance

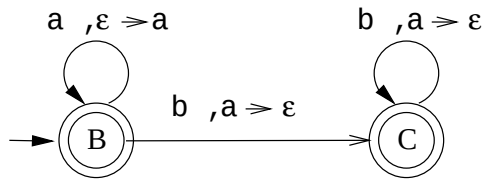
- Une chaîne w est **acceptée** par un reconnaisseur si, partant de l'état initial, avec w sur la bande d'entrée, le reconnaisseur peut faire une série de mouvements pour se retrouver dans un état d'acceptation.
- Le **langage accepté** par un reconnaisseur est l'ensemble de toutes les chaînes qu'il accepte.

Automates à pile



Une configuration d'un automate à pile est un triplet (q, w, α) où α représente le contenu de la pile, le symbole se trouvant au sommet de la pile est le symbole le plus à gauche.

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



$$(B, aaabbb, \$) \vdash (B, aabbb, a\$) \vdash (B, abbb, aa\$) \vdash (B, bbb, aaa\$) \vdash$$
$$(C, bb, aa\$) \vdash (C, b, a\$) \vdash (C, \varepsilon, \$)$$

Automate à pile : définition

Un automate à pile est un septuplet $\langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F \rangle$

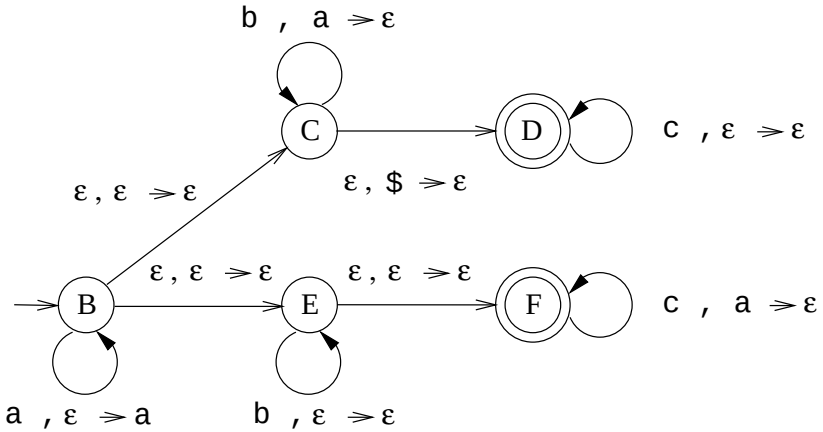
- Q est l'ensemble des états
- Σ est l'alphabet d'entrée
- Γ est l'alphabet de symboles de pile
- δ est la fonction de transition

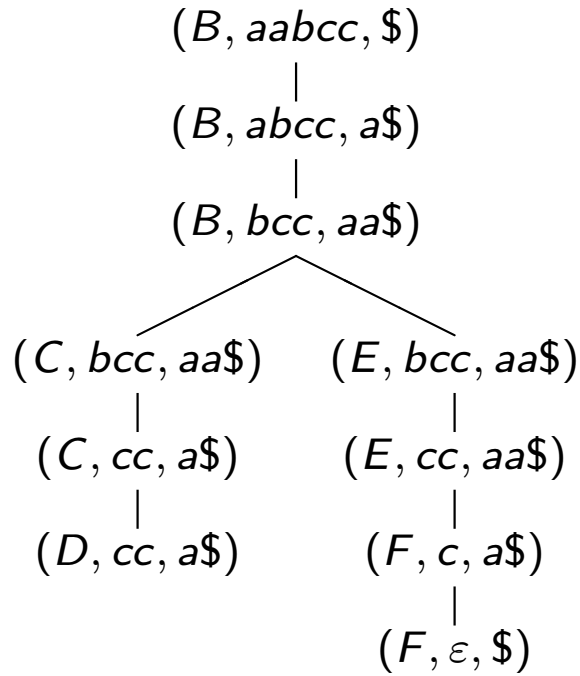
$$\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \to \wp(Q \times \Gamma^*)$$

- $q_0 \in Q$ est l'état initial
- $Z_0 \in \Gamma$ est le symbole de fond de pile
- $F \subseteq Q$ est l'ensemble des états d'acceptation

Reconnaissance avec un AP non déterministe

$$L = \{a^i b^j c^k \mid i, j, k \geq 0 \text{ et } i = j \text{ ou } i = k\}$$

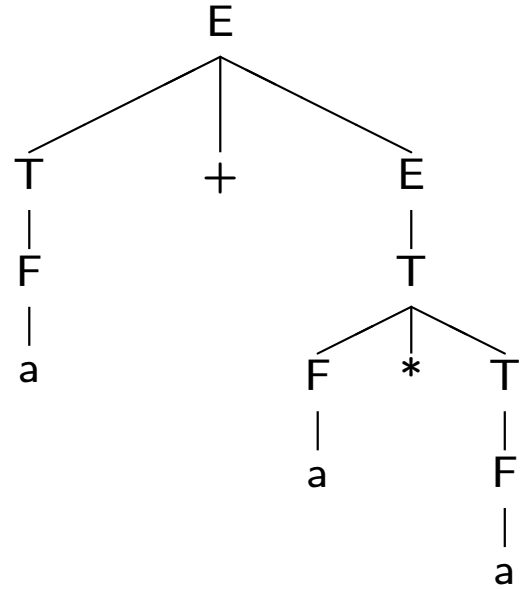




Analyse syntaxique

Etant donné $c \in \Sigma^*$ et $G = \langle \Sigma, N, P, A \rangle$, analyser c consiste à trouver pour c son (et éventuellement ses) arbre de dérivation.

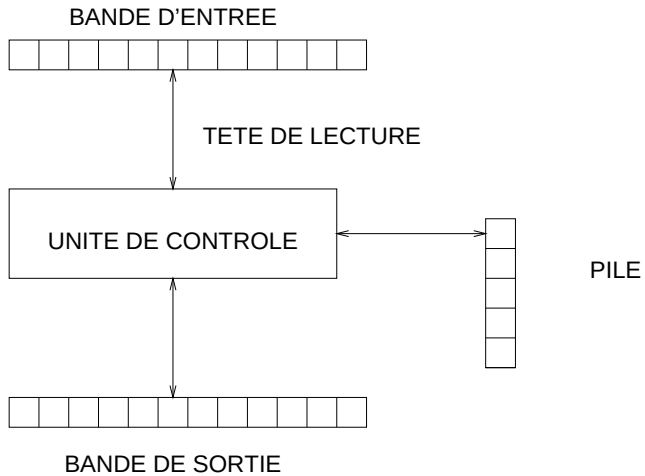
$$
\begin{array}{lcl}
E & \rightarrow & T + E | T \\
T & \rightarrow & F * T | F \\
F & \rightarrow & (E) | a
\end{array}
$$



Sens d'analyse

- Analyse descendante
  Séquence de dérivations gauches à partir de l'axiome
  $E \Rightarrow T + E \Rightarrow F + E \Rightarrow a + E \Rightarrow a + T \Rightarrow a + F * T \Rightarrow$
  $a + a * T \Rightarrow a + a * F \Rightarrow a + a * a$
- Dérivation ascendante
  Séquence de dérivation telle que la séquence inverse soit une dérivation droite de c.
  $a + a * a \Leftarrow F + a * a \Leftarrow T + a * a \Leftarrow T + F * a \Leftarrow$
  $T + F * F \Leftarrow T + F * T \Leftarrow T + E \Leftarrow E$

Transducteurs à pile



Un transducteur à pile est un automate à pile qui **émet**, à chaque déplacement, un suite finie de symboles de sortie.
Une configuration d'un transducteur à pile est un quadruplet (q, w, α, y) y étant une séquence de symboles de sortie.

Transducteur à pile : définition

Un transducteur à pile est un 8-uplet $\langle Q, \Sigma, \Gamma, \Delta, \delta, q_0, Z_0, F \rangle$

- Q est l'ensemble des états
- Σ est l'alphabet d'entrée
- Γ est l'alphabet de symboles de pile
- Δ est l'alphabet de sortie
- δ est la fonction de transition

$$\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \to \wp(Q \times \Gamma^* \times \Delta^*)$$

- $q_0 \in Q$ est l'état initial
- $Z_0 \in \Gamma$ est le symbole de fond de pile
- $F \subseteq Q$ est l'ensemble des états d'acceptation

Analyseur gauche

1 $E \to E + T$ 2 $E \to T$
3 $T \to T * F$ 4 $T \to F$
5 $F \to (E)$ 6 $F \to a$

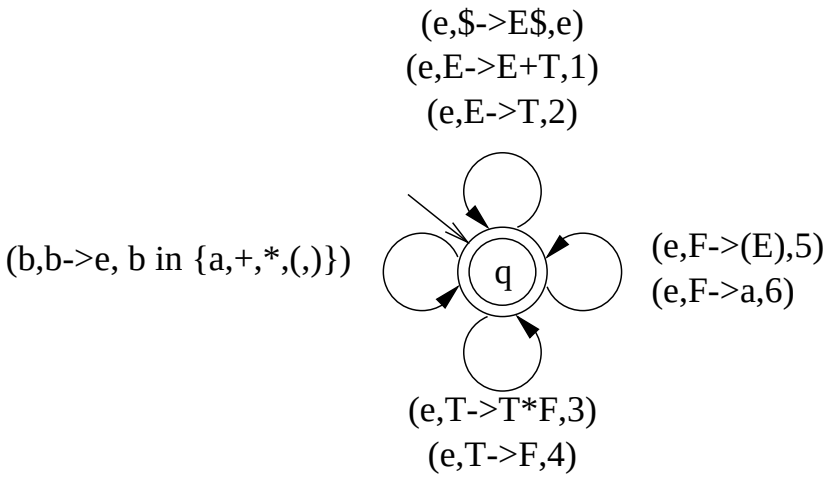
- Dérivation gauche de $a * (a + a)$:
  $E \overset{2}{\Rightarrow} T \overset{3}{\Rightarrow} T * F \overset{4}{\Rightarrow} F * F \overset{6}{\Rightarrow} a * F \overset{*}{\Rightarrow} a * (a * a)$
- Analyse gauche : 23465124646

Analyseur gauche

Soit une CFG G dont les règles ont été numérotées de 1 à p. On appelle un **analyseur gauche** de G, un transducteur à pile non déterministe T_G^g qui produit pour une entrée w, une dérivation gauche de w.
Performances :

- Espace : $\mathcal{O}(|w|)$
- Temps : $\mathcal{O}(c^{|w|})$

Analyseur gauche : Exemple



$$
\begin{aligned}
& (q, a + a * a, \$) \\
\vdash\ & (q, a + a * a, E\$) \\
\vdash\ & (q, a + a * a, E + T\$, 1) \\
\vdash\ & (q, a + a * a, T + T\$, 12) \\
\vdash\ & (q, a + a * a, F + T\$, 124) \\
\vdash\ & (q, a + a * a, a + T\$, 1246) \\
\vdash\ & (q, +a * a, +T\$, 1246) \\
\vdash\ & (q, a * a, T\$, 1246) \\
\vdash\ & (q, a * a, T * F\$, 12463) \\
\vdash\ & (q, a * a, F * F\$, 124634) \\
\vdash\ & (q, a * a, a * F\$, 1246346) \\
\vdash\ & (q, a, F\$, 1246346) \\
\vdash\ & (q, a, a\$, 12463466) \\
\vdash\ & (q, \varepsilon, \$, 12463466)
\end{aligned}
$$

Méthodes tabulaires

Programmation dynamique, les analyses partielles sont effectuées une seule fois et stockées dans une table.

Méthode	Espace	Temps
CYK	$\mathcal{O}(\|w\|^2)$	$\mathcal{O}(\|w\|^3)$
Earley	$\mathcal{O}(\|w\|^2)$	$\mathcal{O}(\|w\|^3)$

L'algorithme de Cocke-Younger-Kasami

Entrée:

- une grammaire hors-contexte sous forme normale de Chomsky sans ϵ-transitions.
- une chaine $w = a_1 a_2 \ ... \ a_n \in \Sigma^+$

Sortie:

- Une table d'analyse T pour w telle que la case $t_{i,j}$ contient A si et seulement si $A \stackrel{+}{\Rightarrow} a_i a_{i+1} ... a_j$

Exemple

- Grammaire initiale

$$E \longrightarrow F + E | F$$
$$F \longrightarrow T * F | T$$
$$T \longrightarrow (E) | a | b$$

- Grammaire sous forme normale de Chomsky

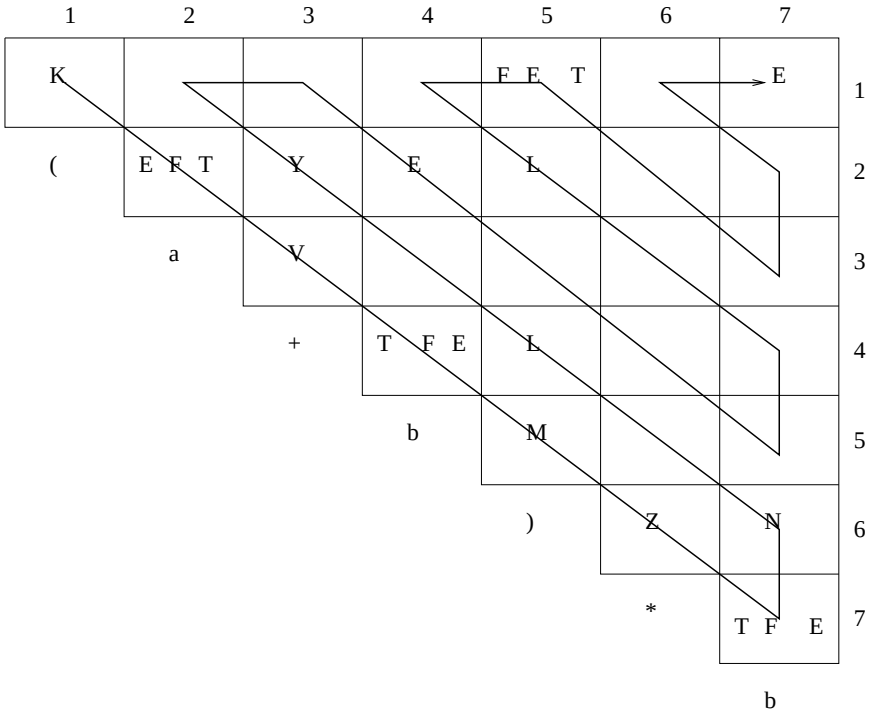
$E \rightarrow YE | TN | KL | a | b$ $\quad N \rightarrow ZF$ $\quad Z \rightarrow *$
$F \rightarrow TN | KL | a | b$ $\quad L \rightarrow EM$ $\quad K \rightarrow ($
$T \rightarrow KL | a | b$ $\quad V \rightarrow +$ $\quad M \rightarrow)$
$Y \rightarrow FV$

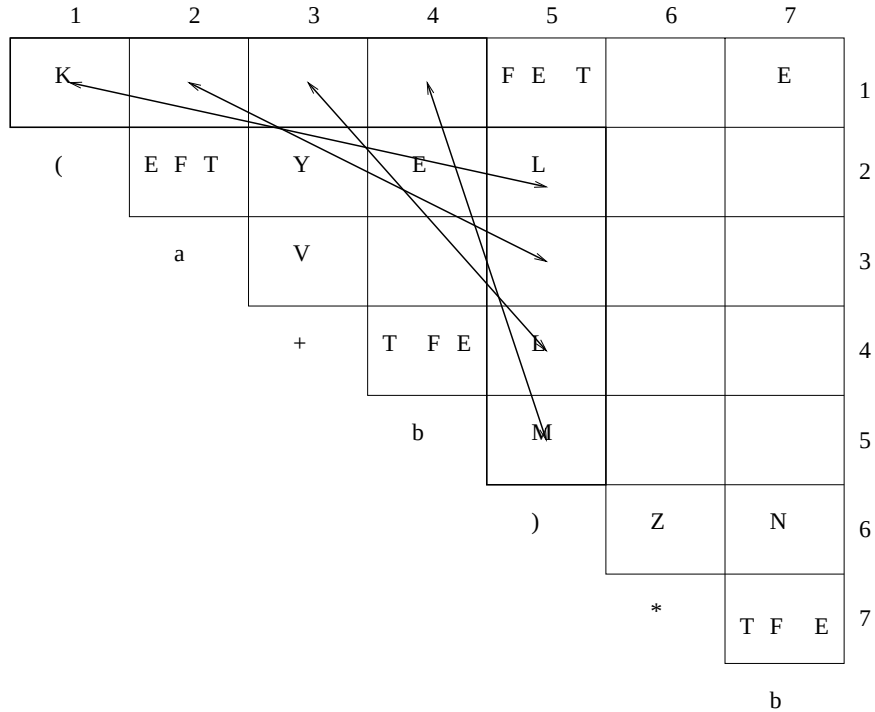
Analyse de la chaine $(a + b) * b$

$E \to YE|TN|KL|a|b \quad N \to ZF \quad Z \to *$
$F \to TN|KL|a|b \quad L \to EM \quad K \to ($
$T \to KL|a|b \quad V \to + \quad M \to)$
$Y \to FV$

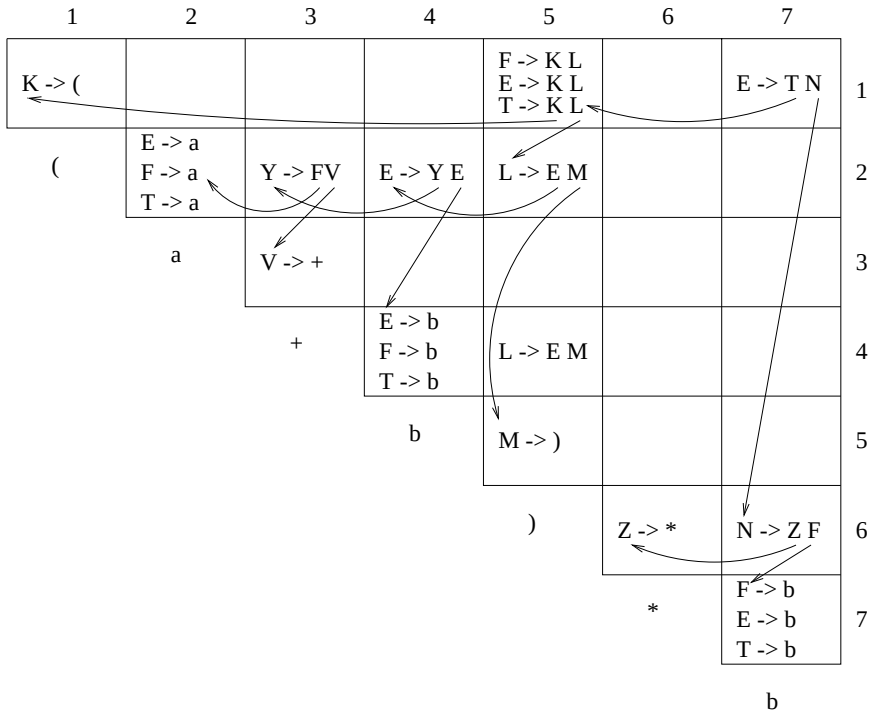
1	2	3	4	5	6	7	
K				F E T		E	1
(	E F T	Y	E	L			2
	a	V					3
		+	T F E	L			4
			b	M			5
				)	Z	N	6
					*	T F E	7
						b	

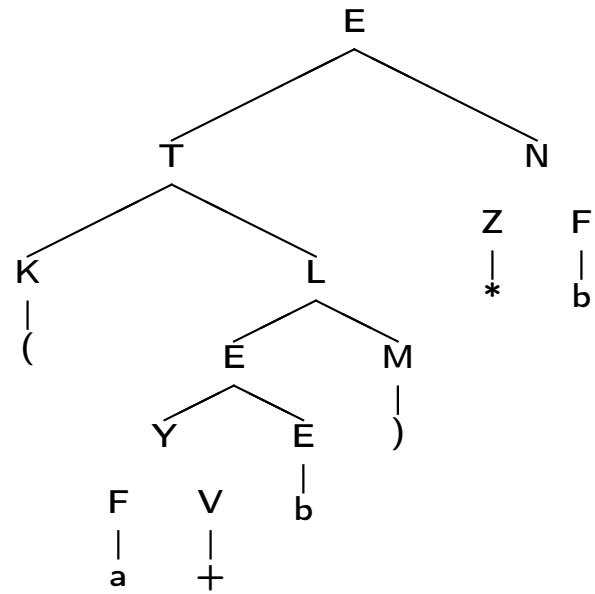
Analyse de la chaine $(a + b) * b$





Analyse de la chaine $(a + b) * b$





Algorithme CYK

pour $i = 1$ à n faire { initialisation }
  $t_{i,i} = \{A | A \to a_i\}$

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 $C \in t_{k+1,j}$

Grammaires hors-contexte probabiliste

Une grammaire hors-contexte probabiliste 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 $N^i \rightarrow \alpha$ avec $\alpha \in (\mathcal{N} \cup \mathcal{T})^*$
- Une distribution de probabilité associée à tout N^i :

$$\sum_j P(N^i \rightarrow \alpha^j) = 1$$

- On fera l'hypothèse que la grammaire est sous forme normale de Chomsky.

Probabilités

- La probabilité d'une règle est la probabilité de choisir cette règle pour réécrire le symbole de la partie gauche.

$$P(N^i \rightarrow \alpha^j) = P(N^i \rightarrow \alpha^j | N^i)$$

- Probabilité d'un arbre T :

$$P(T) = \prod_{n \in T} P(r(n))$$

où $n \in \mathcal{N}$ et $r(n)$ désigne la règle ayant permis de réécrire n.

- Probabilité d'une phrase S :

$$P(S) = \sum_{T \in \mathcal{T}(S)} P(T)$$

où $\mathcal{T}(S)$ est l'ensemble des analyses de S.

Trois problèmes à résoudre

- Calcul de la probabilité d'une phrase S :

$$P(S) = \sum_{T \in \mathcal{T}(S)} P(T)$$

- Construction de l'arbre d'analyse de S le plus probable :

$$\hat{T} = \arg \max_{T \in \mathcal{T}(S)} P(T)$$

- Estimation des probabilités de G à partir de données D :

$$\hat{G} = \arg \max_{G} P(D|G)$$

Parallèle avec les chaînes de Markov cachées

- Calcul de la probabilité d'une séquence d'observables o :

$$P(o) = \sum_{x \in \mathcal{C}_T} P(o, x)$$

où $\mathcal{C}_T$ est l'ensemble des séquences de T états et x une de ces séquences.

- Calcul du chemin le plus probable :

$$\hat{x} = \arg\max_{x \in \mathcal{C}_T} P(x|o)$$

- Estimation des probabilités du HMM :

$$\hat{\lambda} = \arg\max_{\lambda} P(o|\lambda)$$

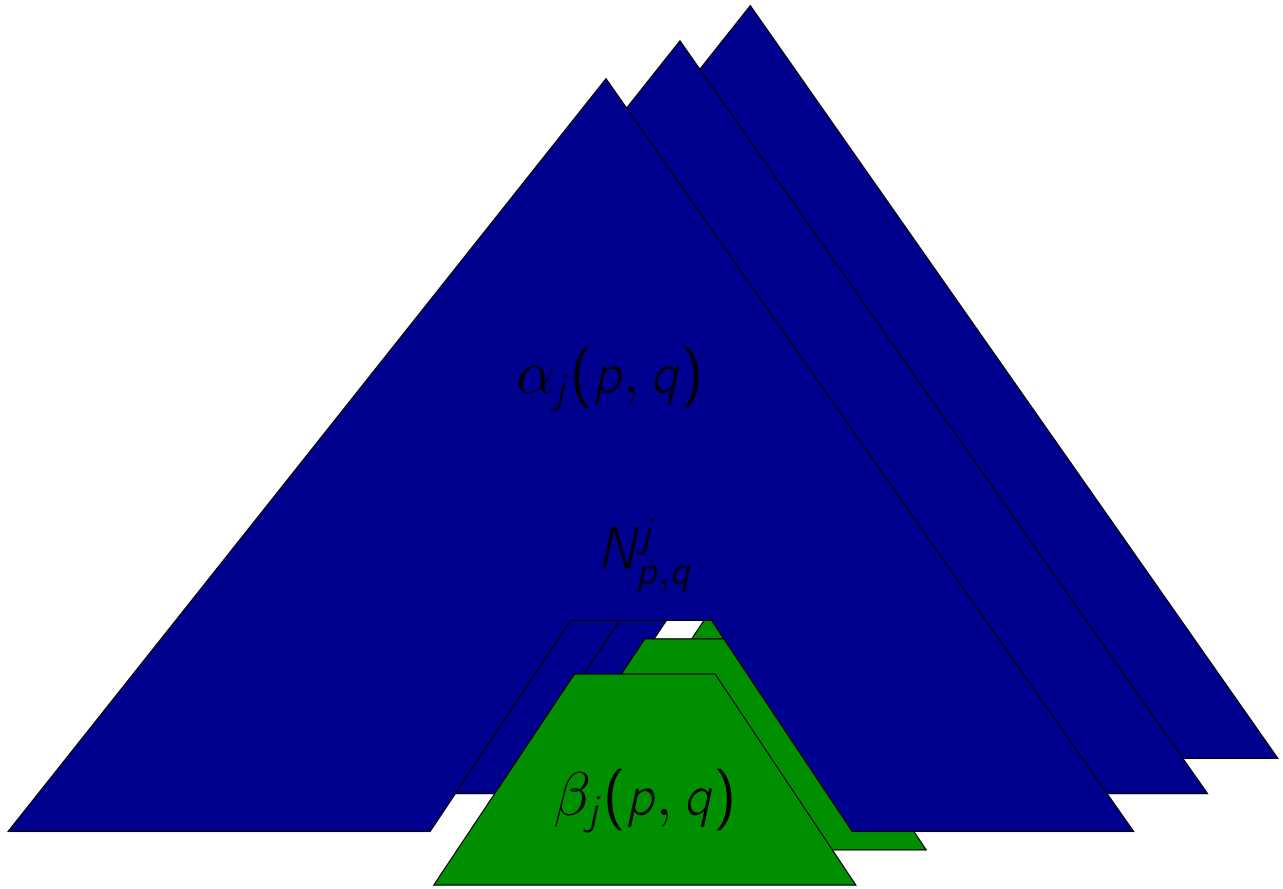
Différences

- Dans le cas d'un HMM une séquence d'états peut correspondre à plusieurs séquences d'observables alors qu'un arbre de dérivation correspond à une seule séquence d'observables (de terminaux).
- Etant donné un HMM et une séquence d'observables o, il est facile de déterminer tous les chemins ayant pu générer o. Dans le cas d'une grammaire probabiliste, il faut déterminer l'ensemble $\mathcal{T}(S)$: faire l'analyse syntaxique de S.

Notations

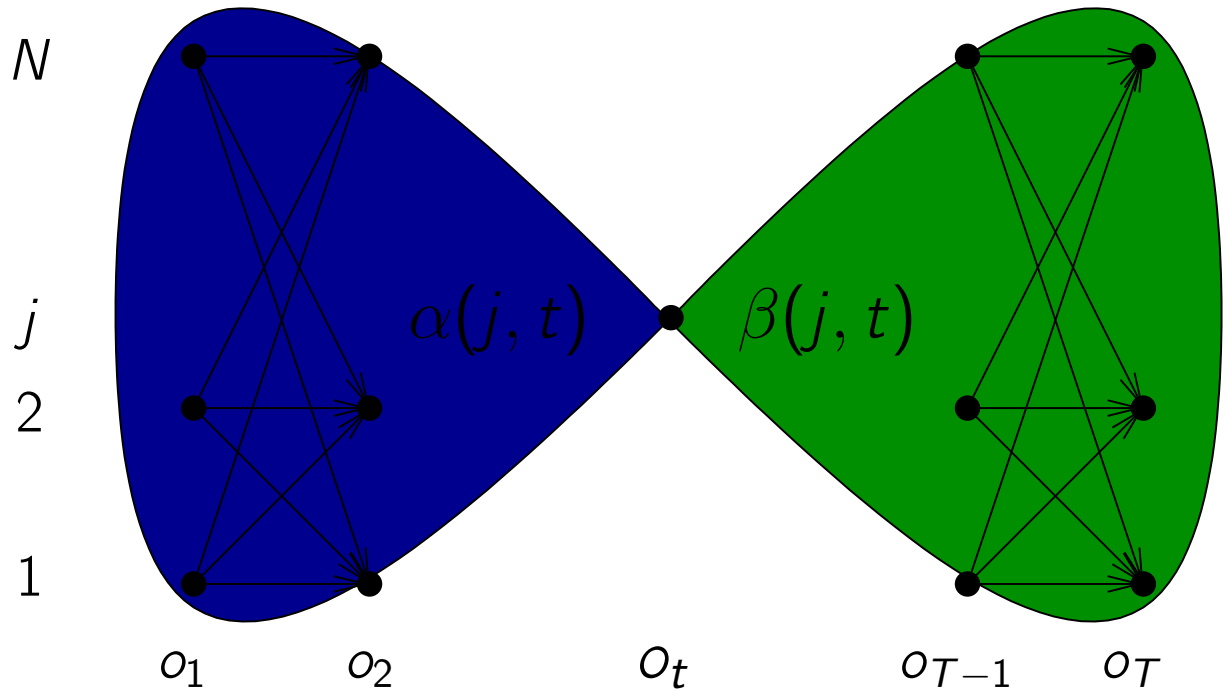
$m_1 \dots m_n$	phrase à analyser
$m_{p,q}$	segment $m_1 \dots m_n$ de la phrase
m^i	symbole de l'alphabet terminal
N^j	symbole de l'alphabet non terminal
$N^j_{p,q}$	le symbole N^j permet de dériver le segment $m_{p,q}$

Probabilités extérieures, probabilités intérieures



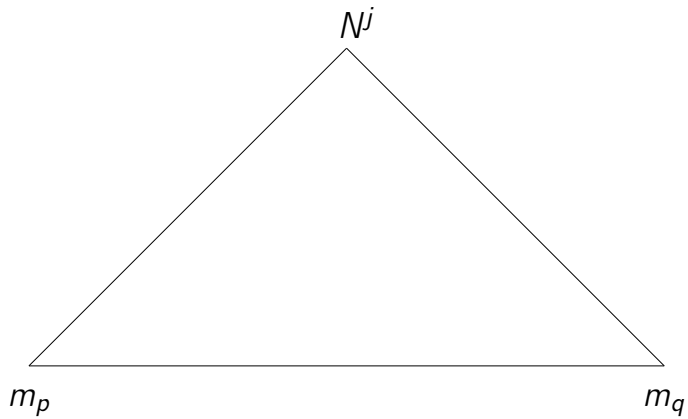
$$\beta_j(p,q) \stackrel{def}{=} P(m_{p,q}|N^j_{p,q}) \qquad \alpha_j(p,q) \stackrel{def}{=} P(m_{1,p-1}, N^j_{p,q}, m_{q+1,m})$$

Parallèle avec les HMM



Probabilité intérieure

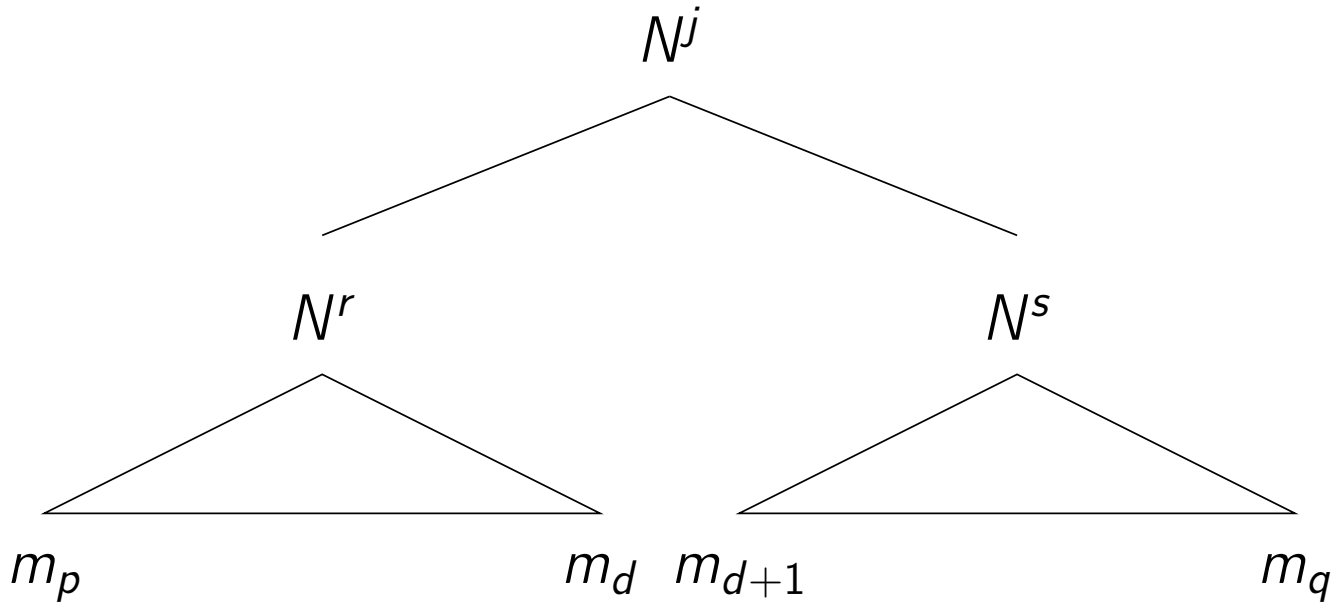
$$\beta_j(p,q) \stackrel{def}{=} P(m_{p,q}|N^j_{p,q})$$



Probabilité d'une phrase

$$
\begin{aligned}
P(m_{1,n}) &= P(N^1 \stackrel{*}{\Rightarrow} m_{1,n}) \\
&= P(m_{1,n}|N^1) \\
&= \beta_1(1,n)
\end{aligned}
$$

Calcul récursif des probabilités intérieures



- Calcul ascendant
- On calcule $\beta_j(p, q)$ à partir de $\beta_r(p, d)$ et $\beta_s(d + 1, q)$

Calcul récursif des probabilités intérieures

Relation de récurrence

$$
\begin{aligned}
\beta_j(p,q) &= P(m_{p,q}|N^j_{p,q}) \\
&= \sum_{r,s}\sum_{d=p}^{q-1} P(m_{p,d}, N^r_{p,d}, m_{d+1,q}, N^s_{d+1,q}|N^j_{p,q}) \\
&= \sum_{r,s}\sum_{d=p}^{q-1} P(N^r_{p,d}, N^s_{d+1,q}|N^j_{p,q}) P(m_{p,d}|N^r_{p,d}, N^s_{d+1,q}, N^j_{p,q}) \\
&\qquad\qquad P(m_{d+1,q}|N^r_{p,d}, N^s_{d+1,q}, N^j_{p,q}) \\
&= \sum_{r,s}\sum_{d=p}^{q-1} P(N^j \to N^r, N^s)\beta_r(p,d)\beta_s(d+1,q)
\end{aligned}
$$

Calcul récursif des probabilités intérieures

Cas terminal

$$\begin{aligned} \beta_j(k,k) &= P(m_k | N^j_{k,k}) \\ &= P(N^j \rightarrow m_k) \end{aligned}$$

Relation avec l'algorithme CYK

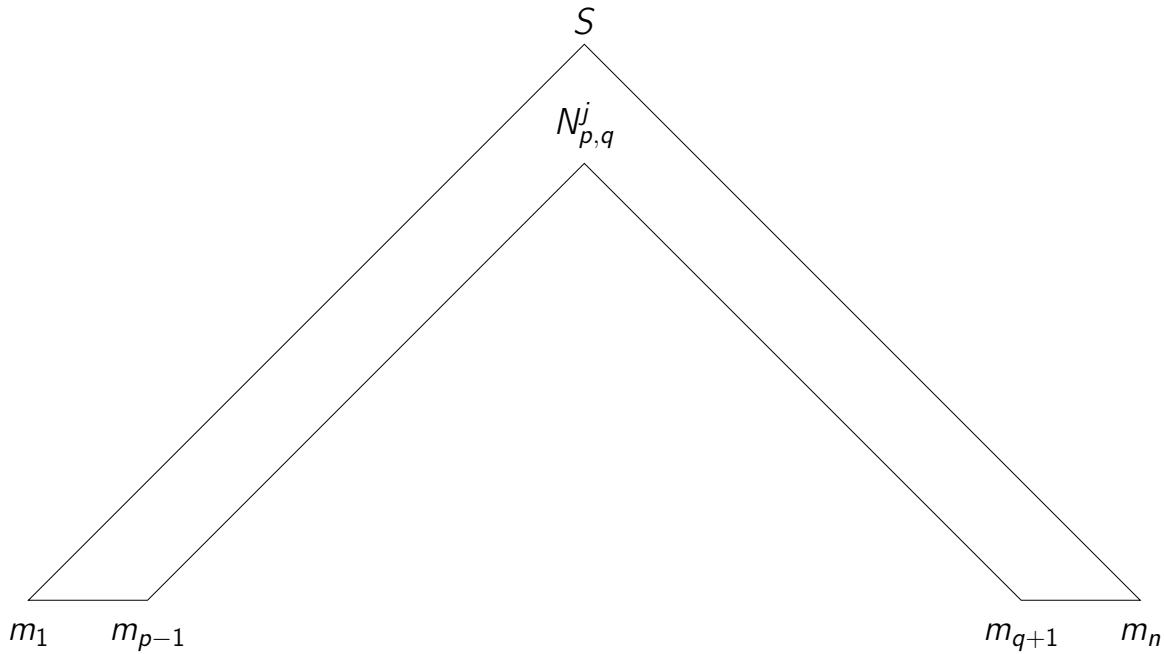
- $N^j_{p,q}$ correspond à la présence du symbole N^j dans la case $t_{p,q}$
- On peut calculer les $\beta(p, q)$ au fur et à mesure que l'on remplit la matrice d'analyse.

Calcul de $P(S)$ façon CYK

pour $q = 1$ à n faire { initialisation }
  pour $p = q$ à 1 faire
    si $(p == q)$
      $\beta_j(p, p) = P(N^j \to m_p)$
    sinon
      $\beta_j(p, q) = 0$
pour $q = 1$ à n faire
  pour $p = q - 1$ à 1 faire
    pour $d = p$ à $q - 1$ faire
      $\beta_j(p, q) = \beta_j(p, q) + P(N^j \to N^r, N^s)\beta_r(p, d)\beta_s(d + 1, q)$
      avec $N^r \in t_{p,d}$ et $N^s \in t_{d+1,q}$
$P(S) = \beta_1(1, n)$

Probabilité extérieure

$$\alpha_j(p,q) \stackrel{def}{=} P(m_{1,p-1}, N^j_{p,q}, m_{q+1,m})$$



Calcul de la probabilité d'une phrase

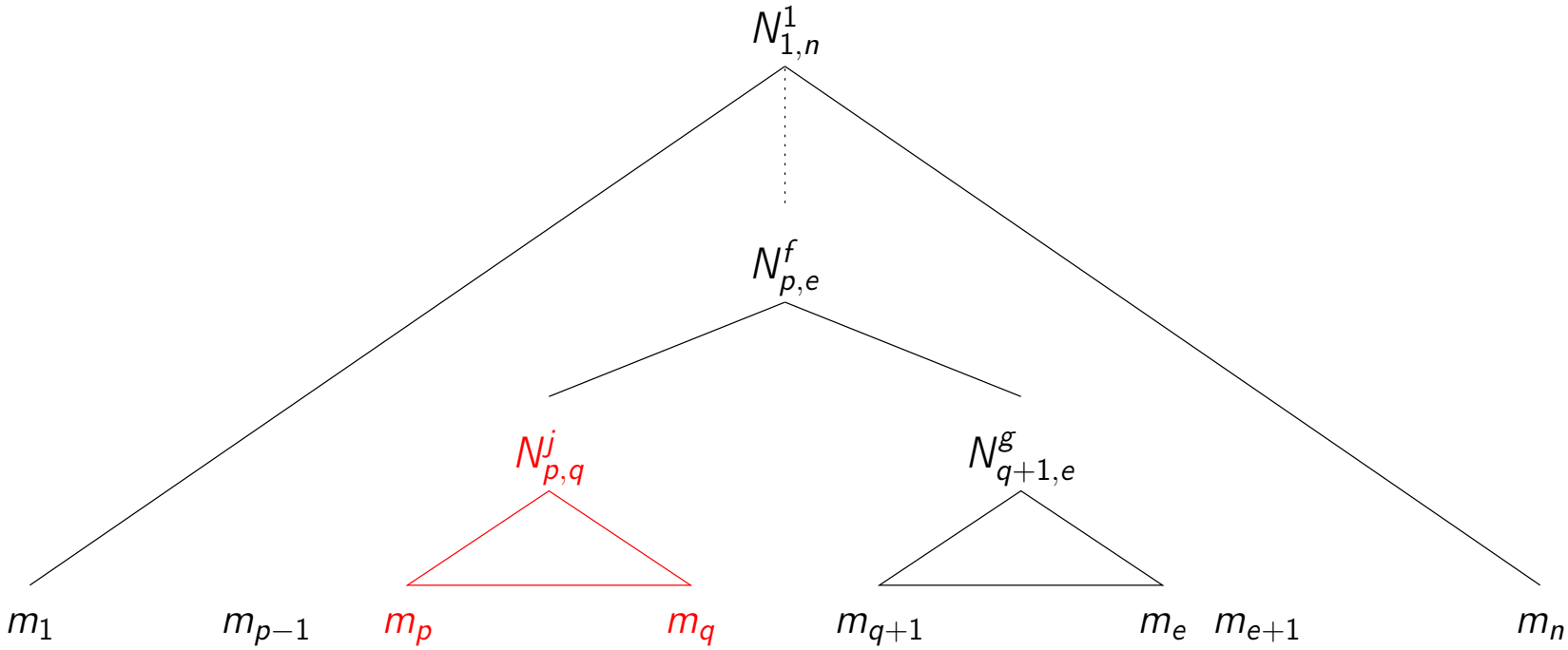
pour tout k, $1 \leq k \leq n$

$$
\begin{aligned}
P(m_{1,n}) &= \sum_j P(m_{1,k-1}, m_k, m_{k+1,n}, N^j_{k,k}) \\
&= \sum_j P(m_{1,k-1}, N^j_{k,k}, m_{k+1,n}) P(m_k | m_{1,k-1}, N^j_{k,k}, m_{k+1,n}) \\
&= \sum_j P(m_{1,k-1}, N^j_{k,k}, m_{k+1,n}) P(m_k | N^j_{k,k}) \\
&= \sum_j \alpha_j(k,k) P(N^j \rightarrow m_k)
\end{aligned}
$$

Calcul de la probabilité d'une phrase

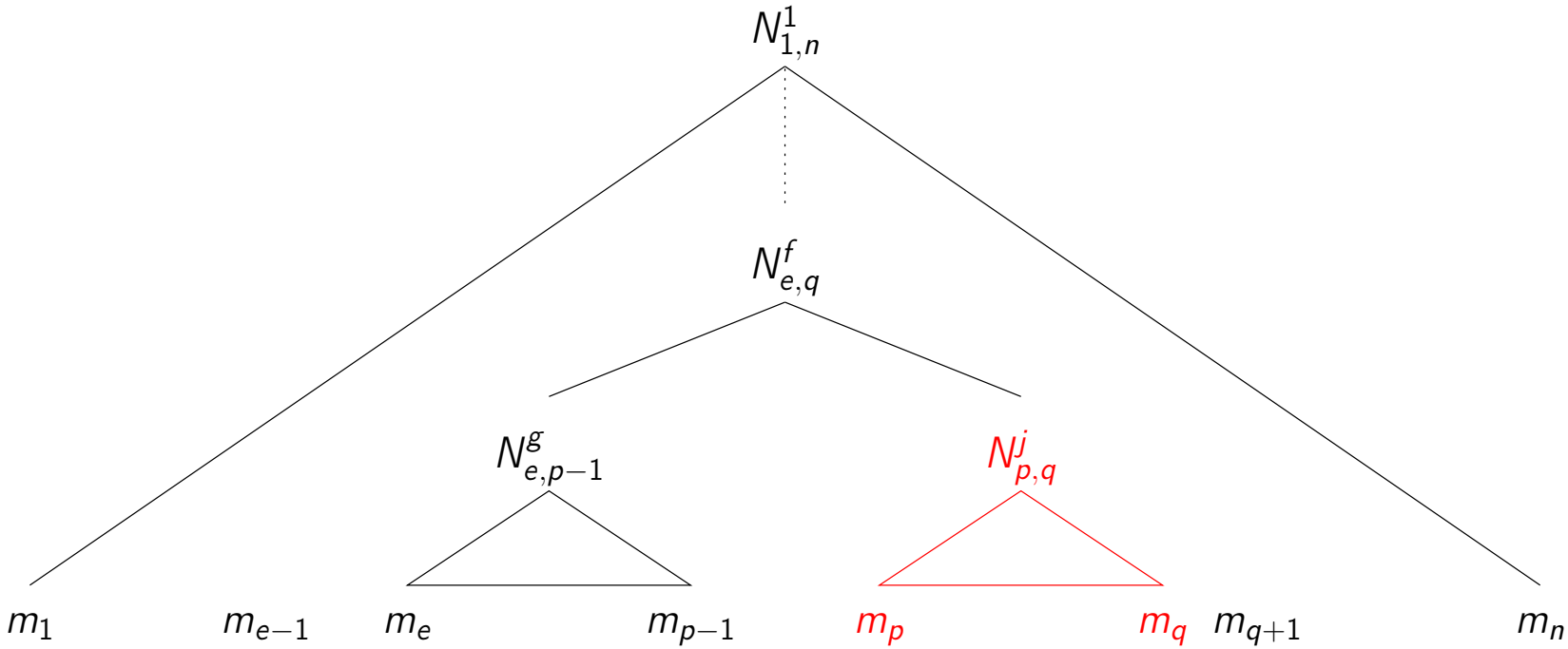
- Calcul descendant
- Le calcul des probabilités extérieues fait appel au calcul des probabilités intérieures, qui doivent avoir été préalablement calculées.

Calcul récursif des probabilités extérieures



$$\alpha_j^G(p,q) = \sum_{f,g} \sum_{e=q+1}^{n} P(m_{1,p-1}, m_{q+1,n}, N^f_{p,e}, N^j_{p,q}, N^g_{q+1,e})$$

Calcul récursif des probabilités extérieures



$$\alpha_j^D(p,q) = \sum_{f,g} \sum_{e=1}^{p-1} P(m_{1,p-1}, m_{q+1,n}, N^f_{e,q}, N^g_{e,p-1}, N^j_{p,q})$$

Calcul récursif des probabilités extérieures

$$
\begin{aligned}
\alpha_j(p,q) &= \alpha_j^G(p,q) + \alpha_j^D(p,q) \\
&= [\sum_{f,g \neq j} \sum_{e=q+1}^{n} P(m_{1,p-1}, m_{q+1,n}, N^f_{p,e}, N^j_{p,q}, N^g_{q+1,e})] \\
&\quad +[\sum_{f,g} \sum_{e=1}^{p-1} P(m_{1,p-1}, m_{q+1,n}, N^f_{e,q}, N^g_{e,p-1}, N^j_{p,q})]
\end{aligned}
$$

On impose la condition $g \neq j$ dans la première somme pour ne pas compter deux fois les comptes des règles de la forme $X \rightarrow N^j N^j$.

Calcul récursif des probabilités extérieures

$$
\begin{aligned}
\alpha_j(p,q) &= [\sum_{f,g\neq j}\sum_{e=q+1}^{n} P(m_{1,p-1}, m_{q+1,n}, N^f_{p,e}, N^j_{p,q}, N^g_{q+1,e})] \\
&\quad +[\sum_{f,g}\sum_{e=1}^{p-1} P(m_{1,p-1}, m_{q+1,n}, N^f_{e,q}, N^g_{e,p-1}, N^j_{p,q})] \\
&= [\sum_{f,g\neq j}\sum_{e=q+1}^{n} P(m_{1,p-1}, m_{q+1,e}, m_{e+1,n}, N^f_{p,e}, N^j_{p,q}, N^g_{q+1,e})] \\
&\quad +[\sum_{f,g}\sum_{e=1}^{p-1} P(m_{1,e-1}, m_{e,p-1}, m_{q+1,n}, N^f_{e,q}, N^g_{e,p-1}, N^j_{p,q})]
\end{aligned}
$$

Calcul récursif des probabilités extérieures

$$
\begin{aligned}
\alpha_j(p,q) &= [\sum_{f,g\neq j}\sum_{e=q+1}^{n} P(m_{1,p-1}, m_{q+1,e}, m_{e+1,n}, N^f_{p,e}, N^j_{p,q}, N^g_{q+1,e})] \\
&\quad +[\sum_{f,g}\sum_{e=1}^{p-1} P(m_{1,e-1}, m_{e,p-1}, m_{q+1,n}, N^f_{e,q}, N^g_{e,p-1}, N^j_{p,q})] \\
&= [\sum_{f,g\neq j}\sum_{e=q+1}^{n} P(m_{1,p-1}, m_{e+1,n}, N^f_{p,e}) P(N^j_{p,q}, N^g_{q+1,e} | m_{1,p-1}, m_{e+1,n}, N^f_{p,e}) \\
&\quad P(m_{q+1,e} | m_{1,p-1}, m_{e+1,n}, N^f_{p,e}, N^j_{p,q}, N^g_{q+1,e})] \\
&\quad +[\sum_{f,g}\sum_{e=1}^{p-1} P(m_{1,e-1}, m_{q+1,n}, N^f_{e,q}) P(N^g_{e,p-1}, N^j_{p,q} | m_{1,e-1}, m_{q+1,n}, N^f_{e,q}) \\
&\quad P(m_{e,p-1} | N^g_{e,p-1}, N^j_{p,q}, m_{1,e-1}, m_{q+1,n}, N^f_{e,q})
\end{aligned}
$$

Calcul récursif des probabilités extérieures

$$
\begin{aligned}
\alpha_j(p,q) &= [\sum_{f,g\neq j}\sum_{e=q+1}^{n} P(m_{1,p-1}, m_{e+1,n}, N^f_{p,e})P(N^j_{p,q}, N^g_{q+1,e}|m_{1,p-1}, m_{e+1,n}, N^f_{p,e}) \\
&\quad P(m_{q+1,e}|m_{1,p-1}, m_{e+1,n}, N^f_{p,e}, N^j_{p,q}, N^g_{q+1,e})] \\
&\quad +[\sum_{f,g}\sum_{e=1}^{p-1} P(m_{1,e-1}, m_{q+1,n}, N^f_{e,q})P(N^g_{e,p-1}, N^j_{p,q}|m_{1,e-1}, m_{q+1,n}, N^f_{e,q}) \\
&\quad P(m_{e,p-1}|N^g_{e,p-1}, N^j_{p,q}, m_{1,e-1}, m_{q+1,n}, N^f_{e,q})] \\
&= [\sum_{f,g\neq j}\sum_{e=q+1}^{n} P(m_{1,p-1}, m_{e+1,n}, N^f_{p,e})P(N^j_{p,q}, N^g_{q+1,e}|N^f_{p,e}) \\
&\quad \times P(m_{q+1,e}|N^g_{q+1,e})] + [\sum_{f,g}\sum_{e=1}^{p-1} P(m_{1,e-1}, m_{q+1,n}, N^f_{e,q}) \\
&\quad \times P(N^g_{e,p-1}, N^j_{p,q}|N^f_{e,q})P(m_{e,p-1}|N^g_{e,p-1})]
\end{aligned}
$$

Calcul récursif des probabilités extérieures

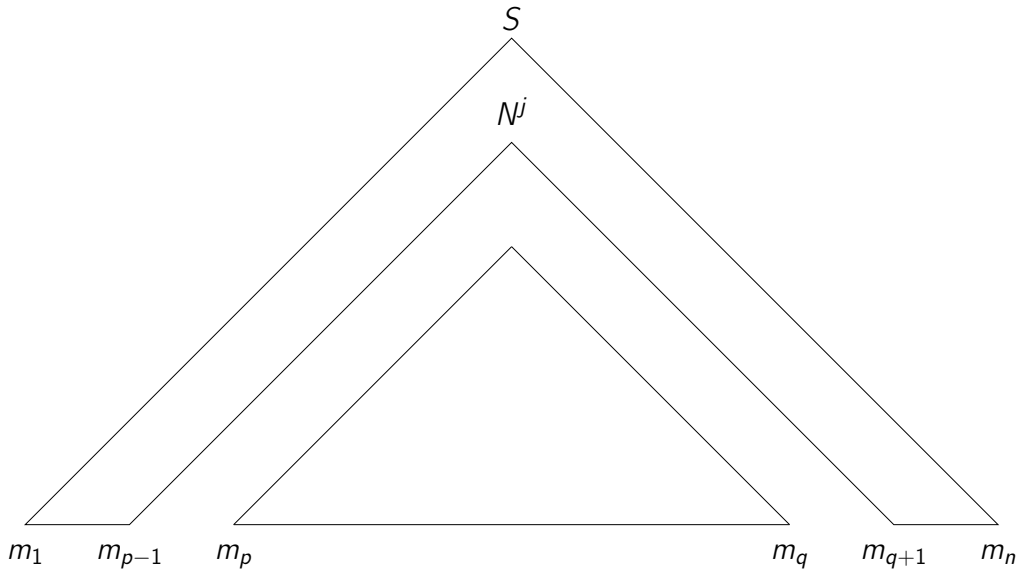
$$
\begin{aligned}
\alpha_j(p,q) &= [\sum_{f,g\neq j}\sum_{e=q+1}^{n} P(m_{1,p-1}, m_{e+1,n}, N^f_{p,e}) P(N^j_{p,q}, N^g_{q+1,e} | N^f_{p,e}) \\
&\quad \times P(m_{q+1,e} | N^g_{q+1,e})] + [\sum_{f,g}\sum_{e=1}^{p-1} P(m_{1,e-1}, m_{q+1,n}, N^f_{e,q}) \\
&\quad \times P(N^g_{e,p-1}, N^j_{p,q} | N^f_{e,q}) P(m_{e,p-1} | N^g_{e,p-1})] \\
&= [\sum_{f,g\neq j}\sum_{e=q+1}^{n} \alpha_f(p,e) P(N^f \rightarrow N^j N^g) \beta_g(q+1,e)] \\
&\quad + [\sum_{f,g}\sum_{e=1}^{p-1} \alpha_f(e,q) P(N^f \rightarrow N^j N^g) \beta_g(e,p-1)]
\end{aligned}
$$

Calcul récursif des probabilités extérieures

Cas terminal :

$$\alpha_1(1, m) = \begin{cases} 1 & \text{si } j = 1 \\ 0 & \text{sinon} \end{cases}$$

Combinaison des probabilités extérieures et intérieures



$$\begin{aligned}
\alpha_j(p,q)\beta_j(p,q) &= P(m_{1,p-1}, N^j_{p,q}, m_{q+1,n})P(m_{p,q}|N^j_{p,q}) \\
&= P(m_{1,p-1}, N^j_{p,q}, m_{q+1,n})P(m_{p,q}|m_{1,p-1}, N^j_{p,q}, m_{q+1,n} \\
&= P(m_{1,n}, N^j_{p,q})
\end{aligned}$$

Probabilité d'un syntagme

- Etant donné une phrase $m_{1,n}$, et une grammaire G, on peut calculer la probabilité que le segment $m_{p,q}$ constitue un syntagme de type N^j :

$$P(m_{1,n}, N^j_{p,q}) = \alpha_j(p,q)\beta_j(p,q)$$

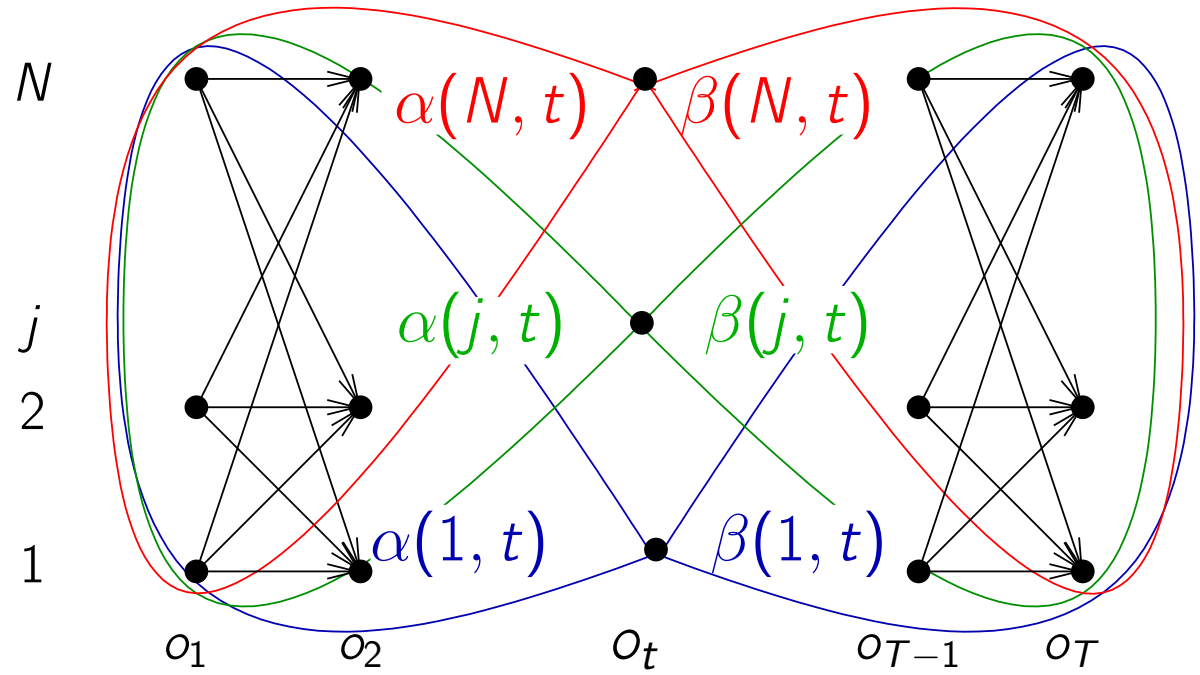
- et la probabilité que le segment $m_{p,q}$ constitue un syntagme de type quelconque :

$$P(m_{1,n}, N_{p,q}) = \sum_j \alpha_j(p,q)\beta_j(p,q)$$

- On peut aussi calculer la probabilité d'occurrence d'un syntagme de type N^j dans la phrase :

$$P(m_{1,n}, N^j) = \sum_{1 \leq p \leq q \leq n} \alpha_j(p,q)\beta_j(p,q)$$

Probabilité d'une phrase (HMM)



$$P(o) = \sum_{i=1}^{N} \alpha(i,t)\beta(i,t) \ \forall t \ 1 \leq t \leq T$$

on est sûr que le HMM sera dans un état quelconque à tout instant t

Probabilité d'une phrase

- dans le cas d'une PCFG, il est nécessaire de postuler la présence d'un non terminal.
  Deux cas sont sûrs :
  - $N^1_{1,n}$ existe :

$$P(m_{1,n}) = \alpha_1(1,n)\beta_1(1,n)$$

  - tout terminal est dominé par un préterminal :

$$P(m_{1,n}) = \sum_j \alpha_1(k,k)\beta_1(k,k) \ \forall \ 1 \leq k \leq n$$

Calcul de la probabilité de $\hat{T}$

$\delta_i(p,q) =$ la probabilité du sous-arbre $N^j_{p,q}$ le plus probable.

1. Initialisation

$$\delta_i(p,p) = P(N^i \rightarrow m_p)$$

2. Récurrence

$$\delta_i(p,q) = \max_{1 \leq j,k \leq n,\ p \leq d < q} P(N^i \rightarrow N^j N^k)\delta_j(p,d)\delta_k(d+1,q)$$

3. Fin

$$P(\hat{T}) = \delta_1(1,n)$$

Calcul de $P(\hat{T})$

pour $q = 1$ à n faire { initialisation }
  pour $p = q$ à 1 faire
    si $(p == q)$
      $\delta_j(p, p) = P(N^j \rightarrow m_p)$
    sinon
      $\delta_j(p, q) = 0$
pour $q = 1$ à n faire
  pour $p = q - 1$ à 1 faire
    pour $d = p$ à $q - 1$ faire
      $\delta_j(p, q) = \max(\delta_j(p, q), P(N^i \rightarrow N^j N^k)\delta_j(p, d)\delta_k(d + 1, q))$
      avec $N^r \in t_{p,d}$ et $N^s \in t_{d+1,q}$
$P(\hat{T}) = \beta_1(1, n)$

Construction de $\hat{T}$

$\psi_i(p,q) = \langle j,k,r \rangle$ où j,k,r désignent l'application de la règle ayant réalisé le maximum $\delta_i(p,q)$

$$\psi_i(p,q) = \arg\max_{j,k,d} P(N^i \rightarrow N^j N^k)\delta_j(p,d)\delta_k(d+1,q)$$

- racine($\hat{T}$) = $N^1_{1,n}$
- si $\psi_i(p,q) = \langle j,k,r \rangle$ alors
  - fils gauche($N^i_{p,q}$) = $N^j_{p,r}$
  - fils droit($N^i_{p,q}$) = $N^k_{r+1,q}$

Beam search

- Il est possible d'utiliser les probabilités calculées pour élaguer la forêt d'analyse au fur et à mesure de sa construction.
- Principe : dans chaque case $t_{p,q}$ on ne garde que les $N^j_{p,q}$ les plus probables.

Beam search

- A l'issue du remplissage de $t_{p,q}$
- On détermine :

$$\beta_{p,q} = \max_j \beta^j_{p,q}$$

- On calcule le seuil :

$$\tau_{p,q} = \frac{\beta_{p,q}}{K}$$

  où K est une constante déterminée empiriquement
- On élimine $N^j_{p,q}$ si $\beta^j_{p,q} < \tau_{p,q}$
- On n'est plus garanti de retrouver $\hat{T}$!

Estimation des probabilités de la grammaire

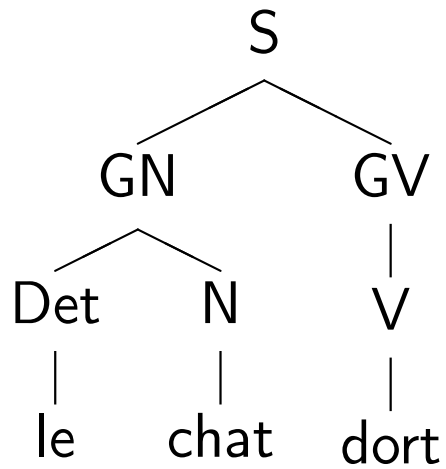
Deux cas :

- Données complètes : on dispose d'un ensemble de phrases et de leur analyse syntaxique (banque d'arbres).
- Données incomplètes : on ne dispose que d'un ensemble de phrases.

Exemples de banques d'arbres

corpus	Penn Treebank	corpus Paris 7
mots	1 000 000	400 000
phrases	45 000	15 000
cat. syntag.	26	13
parties de discours	36	14
règles	9 657	

Construction de la grammaire



$$
\begin{array}{lcl}
S & \rightarrow & GN\ GV \\
GN & \rightarrow & Det\ N \\
GV & \rightarrow & V \\
Det & \rightarrow & le \\
N & \rightarrow & chat \\
V & \rightarrow & dort
\end{array}
$$

Estimation des probabilités des règles

- On compte le nombre d'occurrences des symboles non terminaux A dans la banque d'arbres : $C(A)$
- On compte le nombre d'occurrences des règles $A \rightarrow \alpha$ dans la banque d'arbres : $C(A \rightarrow \alpha)$
- On estime $P(A \rightarrow \alpha)$ par maximum de vraisemblance :

$$P(A \rightarrow \alpha) = \frac{C(A \rightarrow \alpha)}{C(A)}$$

Estimation à partir de données incomplètes

- On fixe la partie algébrique de la grammaire (alphabets, règles)
- On recherche les distributions de probabilités de la grammaire qui maximise la probabilité des données d'apprentissage (des phrases dont on dispose)
- On ne sait calculer ces distributions directement, on fait appel à un algorithme itératif du type Expectation Maximization appelé algorithme intérieur-extérieur (inside-outside).

Principe de l'algorithme intérieur extérieur

- On fixe des distributions de probabilités initiales
- On calcule la probabilité des différents symboles et règles étant donné une phrase S
- On estime le nombre d'occurrences des différents symboles et règles lors des dérivations de S
- On calcule de nouvelles distributions de probabilités
- On réitère tant que les nouvelles probabilités augmentent $P(S)$

Estimation de $C(N^j)$

$$\begin{aligned}\alpha_j(p,q)\beta_j(p,q) &= P(N^1 \overset{*}{\Rightarrow} m_{1,n}, N^j \overset{*}{\Rightarrow} m_{p,q}) \\ &= P(N^1 \overset{*}{\Rightarrow} m_{1,n}) P(N^j \overset{*}{\Rightarrow} m_{p,q} | N^1 \overset{*}{\Rightarrow} m_{1,n})\end{aligned}$$

On sait comment calculer $P(N^1 \overset{*}{\Rightarrow} m_{1,n})$, que l'on appelle π.
On a :

$$P(N^j \overset{*}{\Rightarrow} m_{p,q} | N^1 \overset{*}{\Rightarrow} m_{1,n}) = \frac{\alpha_j(p,q)\beta_j(p,q)}{\pi}$$

On utilise cette probabilité pour estimer le nombre de fois que N^j a été utilisé dans la dérivation :

$$C(N^j) = \sum_{p=1}^{n} \sum_{q=p}^{n} \frac{\alpha_j(p,q)\beta_j(p,q)}{\pi}$$

Estimation de $C(N^j \rightarrow N^r N^s)$

$$P(N^j \rightarrow N^r N^s \overset{*}{\Rightarrow} m_{p,q} | N^1 \overset{*}{\Rightarrow} m_{1,n}) =$$

$$\frac{\sum_{d=p}^{q-1} \alpha_j(p,q) P(N^j \rightarrow N^r, N^s) \beta_r(p,d) \beta_s(d+1,q)}{\pi}$$

Pour estimer $C(N^j \rightarrow N^r N^s, N^j)$, on fait la somme sur tous les segments $m_{p,q}$ que N^j peut dominer :

$$C(N^j \rightarrow N^r N^s, N^j) =$$

$$\frac{\sum_{p=1}^{n-1} \sum_{q=p+1}^{n} \sum_{d=p}^{q-1} \alpha_j(p,q) P(N^j \rightarrow N^r, N^s) \beta_r(p,d) \beta_s(d+1,q)}{\pi}$$

Calcul de $\hat{P}(N^j \rightarrow N^r N^s)$

$$\hat{P}(N^j \rightarrow N^r N^s) = \frac{C(N^j \rightarrow N^r N^s)}{C(N^j)}$$

$$= \frac{\sum_{p=1}^{n-1} \sum_{q=p+1}^{n} \sum_{d=p}^{q-1} \alpha_j(p,q) P(N^j \rightarrow N^r, N^s) \beta_r(p,d) \beta_s(d+1,q)}{\sum_{p=1}^{n} \sum_{q=p}^{n} \alpha_j(p,q) \beta_j(p,q)}$$

Calcul de $\hat{P}(N^j \rightarrow m^k)$

$$P(N^j \overset{*}{\Rightarrow} m_{p,q} | N^1 \overset{*}{\Rightarrow} m_{1,n}) = \frac{\alpha_j(p,q)\beta_j(p,q)}{\pi}$$

$$\begin{aligned} P(N^j \rightarrow m^k | N^1 \overset{*}{\Rightarrow} m_{1,n}) &= \frac{\sum_{h=1}^{n} \alpha_j(h,h) P(N^j \rightarrow m_h, m_h = m^k)}{\pi} \\ &= \frac{\sum_{h=1}^{n} \alpha_j(h,h)\beta_j(h,h) P(m_h = m^k)}{\pi} \end{aligned}$$

où $P(m_h = m^k)$ vaut 0 ou 1

$$\hat{P}(N^j \rightarrow m^k) = \frac{\sum_{h=1}^{n} \alpha_j(h,h) P(N^j \rightarrow m_h), P(m_h = m^k)\beta_j(h,h)}{\sum_{p=1}^{n}\sum_{q=p}^{n} \alpha_j(p,q)\beta_j(p,q)}$$

Formules de réestimation

$$\hat{P}(N^j \rightarrow m^k) = \frac{\sum_{h=1}^{n} \alpha_j(h,h) P(N^j \rightarrow m_h), P(m_h = m^k) \beta_j(h,h)}{\sum_{p=1}^{n} \sum_{q=p}^{n} \alpha_j(p,q) \beta_j(p,q)}$$

$$\hat{P}(N^j \rightarrow N^r N^s) = \frac{\sum_{p=1}^{n-1} \sum_{q=p+1}^{n} \sum_{d=p}^{q-1} \alpha_j(p,q) P(N^j \rightarrow N^r, N^s) \beta_r(p,d) \beta_s(d+1,q)}{\sum_{p=1}^{n} \sum_{q=p}^{n} \alpha_j(p,q) \beta_j(p,q)}$$

Problèmes de l'algorithme extérieur intérieur

- Efficacité : pour chaque phrase du corpus d'apprentissage et chaque itération, la complexité est $O(n^3 V^3)$ où n est la longueur de la phrase et V le nombre de non terminaux de la grammaire.
- Maximum local : l'algorithme est très sensible aux distributions de probabilités initiales. Des distributions différentes aboutissent à des maxima différents.
- Choix du nombre de non terminaux : on ne sait pas combien de non terminaux choisir, les expériences ont montré qu'un nombre important de non terminaux améliore les résultats, ce qui augmente le problème d'efficacité.

Limites des PCFG (1)

- Indépendance lexicale
  - La réecriture d'un symbole pré-terminal X ne dépend pas du contexte d'occurrence de X
  - mise en défaut : la préposition introduisant un complément d'un verbe dépend de la nature lexicale du verbe
    Exemple : *Jean* **pense à** *Marie*
  - cette dépendance n'est pas modélisée :
    $GV \rightarrow V\ GP$
    $V \rightarrow pense$
    $GP \rightarrow P\ GN$
    $P \rightarrow à$

Limites des PCFG (2)

- Indépendance structurale
  - le choix d'une règle pour la réécriture d'un symbole X est indépendant du contexte d'occurrence de X
  - mise en défaut : Dans le corpus switchboard, 91% des sujets sont des pronoms alors que seule 34% des objets le sont (Francis et al 99)
  - Or il n'y a qu'une règle de la forme $GN \rightarrow Pro$
  - et une seule probabilité lui correspondant

Lexicalisation (Charniak,1997) (Collins, 1999)

- tête lexicale : un syntagme est construit autour d'une **tête lexicale**
- les catégories non terminales de la grammaire sont constituées de couples (cat, lex) où lex est la tête du syntagme de catégorie cat
- Exemple : $(GV, penser) \rightarrow (V, penser)(GP, à)$
- Explosion du nombre de catégories et du nombre de règles
- Problèmes sévères d'estimation des probabilités des règles.

Grammaires fondées sur l'historique (Black et al, 1992)

- la probabilité d'appliquer une règle à une étape de la dérivation dépend de la structure dérivée jusque là

$$S = T_1 \overset{R_1}{\Longrightarrow} T_2 \overset{R_2}{\Longrightarrow} \ldots \overset{R_{n-2}}{\Longrightarrow} T_{n-1} \overset{R_{n-1}}{\Longrightarrow} T_n = T$$

$$P(T) = \prod_{i=1}^{n-1} P(R_i|T_i)$$

- modèle non réaliste : la partie conditionnelle est beaucoup trop importante, l'estimation de la probabilité des règles est difficile.
- regroupement des historiques en classes d'équivalence.

$$P(R_i|T_i) \equiv P(R_i|E[T_i])$$

mesure d'évaluation : PARSEVAL (Black et al, 1991)

- étant donné une phrase s, un arbre candidat C pour s et l'arbre référence R pour s.
- un constituent dans C est correct s'il existe un constituent dans R étiqueté par la même catégorie et recouvrant le même segment de phrase.
- labeled recall (LR) $= \frac{\#\ constituents\ corrects\ dans\ C}{\#\ constituents\ dans\ R}$
- labeled precision (LP) $= \frac{\#\ consituents\ corrects\ dans\ C}{\#\ constituents\ dans\ C}$
- $F1$: moyenne harmonique de LR et LP ($F1 = \frac{2 \times LR \times LP}{LR + LR}$)
- Résultats sur le PTB : $F1 = 72.62$

Markovisation des règles (Klein Manning 03)

- Markovisation verticale : prise en compte des v premiers ancêtres dans la partie conditionnelle
- Markovisation horizontale : prise en compte des h frères de gauche dans la partie conditionnelle

	ordre horizontal			
ordre vertical	$h = 0$	$h = 1$	$h = 2$	$h = \infty$
$v = 1$	71.72	72.50	72.96	72.62
$v = 2$	74.68	77.42	77.50	76.81
$v = 3$	76.74	79.18	79.07	78.72

Divers résultats sur le Penn Tree Bank

	LP	LR	F1
Magerman 1995	84.9	84.6	84.7
Collins 1996	86.3	85.8	85.9
Charniak 1997	87.4	87.5	87.4
Collins 1999	88.7	88.6	88.6