Analyse syntaxique

Grammaires hors-contexte

Une grammaire hors-contexte 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 :

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

un élément (α, β) de P, que l'on note $\alpha \to \beta$ est appelé une règle de production ou règle de réécriture.
α est appelé partie gauche de la règle
β est appelé partie droite de la règle

- S est un élément de N appelé l'axiome de la grammaire.

Automate à pile

Un automate à pile est un 6-uplet $\langle Q, \Sigma, \Gamma, \delta, q_0, F \rangle$

- 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 :

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

- $q_0 \in Q$ est l'état initial
- $F \subseteq Q$ est l'ensemble des états d'acceptation

Grammaires hors-contexte $\Leftrightarrow$ Automate à pile

Un langage est hors-contexte si et seulement si il existe un automate à pile qui le reconnaît.

- Si un langage est hors-contexte alors il existe un automate à pile qui le reconnaît.
- Si un langage est reconnu par un automate à pile alors il est hors-contexte.

Grammaires hors-contexte $\Rightarrow$ Automate à pile

- Soit $G = \langle N, \Sigma, P, S \rangle$ une grammaire hors-contexte, on construit un automate à pile A qui accepte un mot m s'il existe une dérivation pour m dans G ($S \overset{+}{\Rightarrow} m$).
- A est conçu de telle sorte à déterminer une dérivation conduisant de S à m.
- Idée clef : écrire dans la pile de A les proto-mots qui constituent la dérivation recherchée.

Principe

1. Empiler l'axiome S
2. Remplacer S par la partie droite d'une règle de la forme $S \rightarrow \alpha$ de telle sorte que le premier symbole x de α se trouve en sommet de pile.
   - Si x est un terminal alors on le compare avec le caractère se trouvant sous la tête de lecture. S'ils sont égaux alors on dépile.
   - Si x est un non terminal alors on le remplace par la partie droite d'une règle de P de la forme $x \rightarrow \beta$.

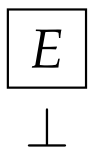
Exemple

Reconnaissance du mot :

$$a + a * a$$

avec la grammaire :

$$\begin{aligned} E &\rightarrow T + E \mid T \\ T &\rightarrow F * T \mid F \\ F &\rightarrow (E) \mid a \end{aligned}$$



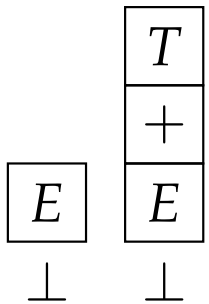
Exemple

Reconnaissance du mot :

$$a + a * a$$

avec la grammaire :

$$\begin{aligned} E &\to T + E \mid T \\ T &\to F * T \mid F \\ F &\to (E) \mid a \end{aligned}$$



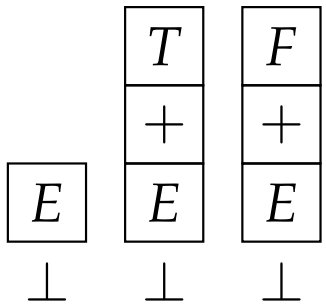
Exemple

Reconnaissance du mot :

$$a + a * a$$

avec la grammaire :

$$E \to T + E \mid T$$
$$T \to F * T \mid F$$
$$F \to (E) \mid a$$



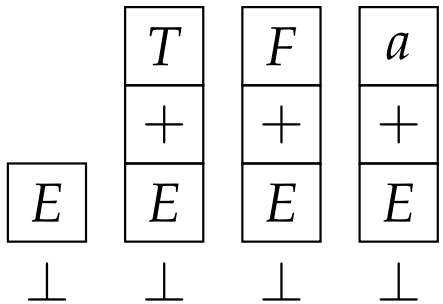
Exemple

Reconnaissance du mot :

$$a + a * a$$

avec la grammaire :

$$E \to T + E \mid T$$
$$T \to F * T \mid F$$
$$F \to (E) \mid a$$



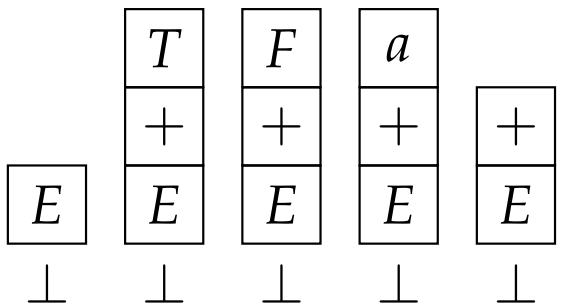
Exemple

Reconnaissance du mot :

$$a + a * a$$

avec la grammaire :

$$E \to T + E \mid T$$
$$T \to F * T \mid F$$
$$F \to (E) \mid a$$



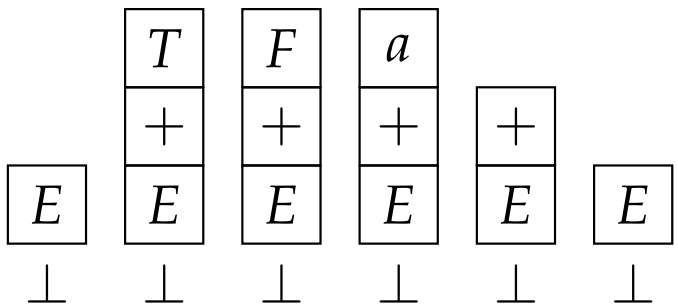
Exemple

Reconnaissance du mot :

$$a + a * a$$

avec la grammaire :

$$E \to T + E \mid T$$
$$T \to F * T \mid F$$
$$F \to (E) \mid a$$



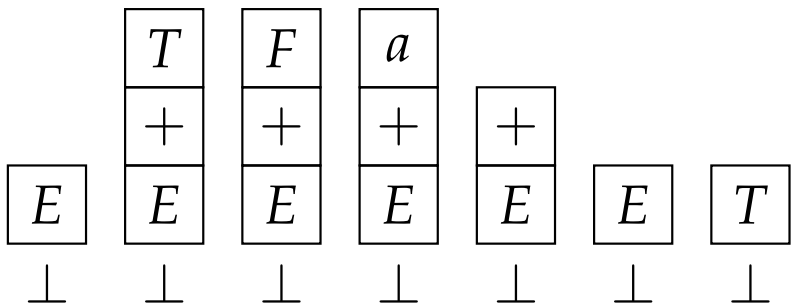
Exemple

Reconnaissance du mot :

$$a + a * a$$

avec la grammaire :

$$E \to T + E \mid T$$
$$T \to F * T \mid F$$
$$F \to (E) \mid a$$



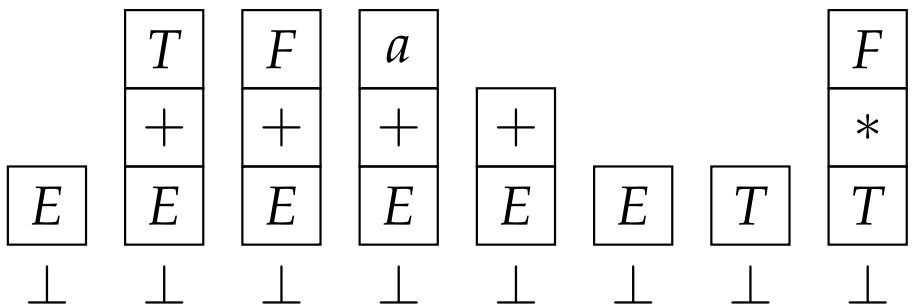
Exemple

Reconnaissance du mot :

$$a + a * a$$

avec la grammaire :

$$E \to T + E \mid T$$
$$T \to F * T \mid F$$
$$F \to (E) \mid a$$



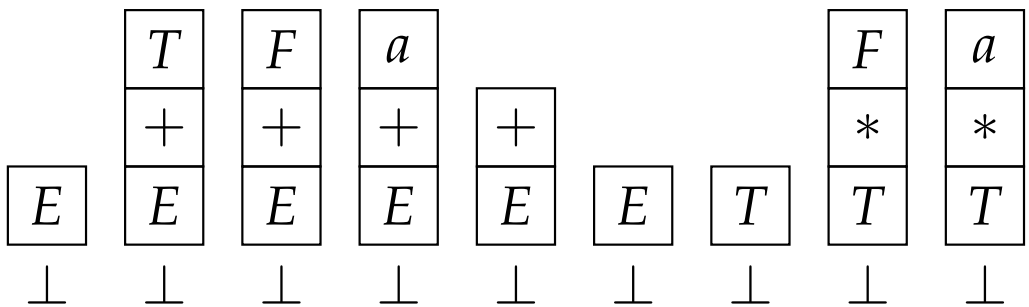
Exemple

Reconnaissance du mot :

$$a + a * a$$

avec la grammaire :

$$E \to T + E \mid T$$
$$T \to F * T \mid F$$
$$F \to (E) \mid a$$



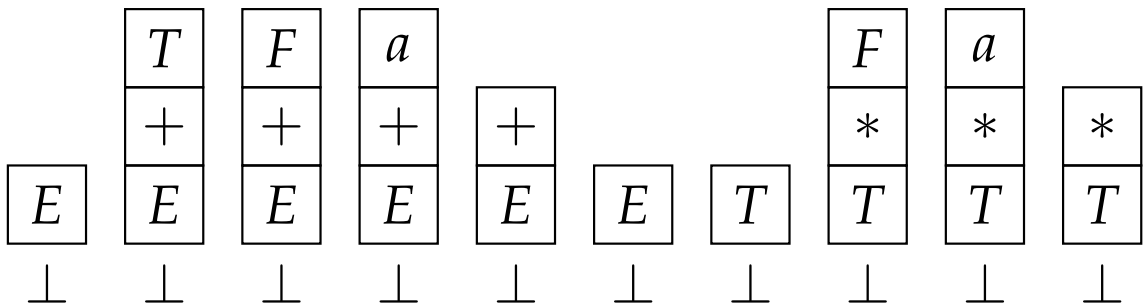
Exemple

Reconnaissance du mot :

$$a + a * a$$

avec la grammaire :

$$E \to T + E \mid T$$
$$T \to F * T \mid F$$
$$F \to (E) \mid a$$



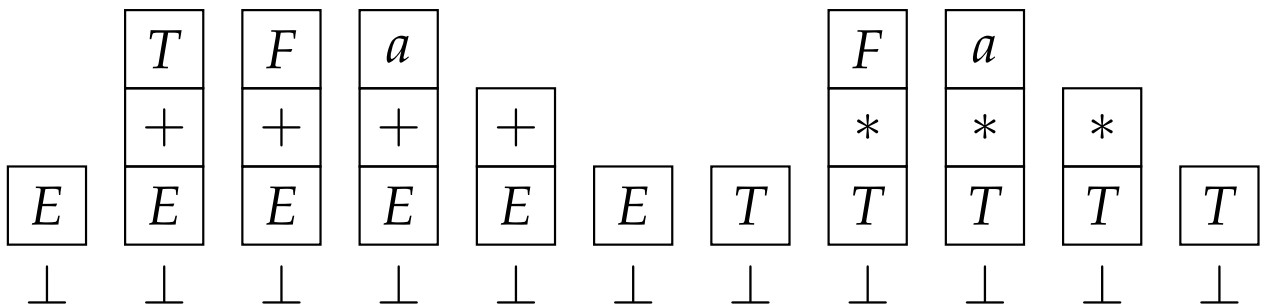
Exemple

Reconnaissance du mot :

$$a + a * a$$

avec la grammaire :

$$E \to T + E \mid T$$
$$T \to F * T \mid F$$
$$F \to (E) \mid a$$



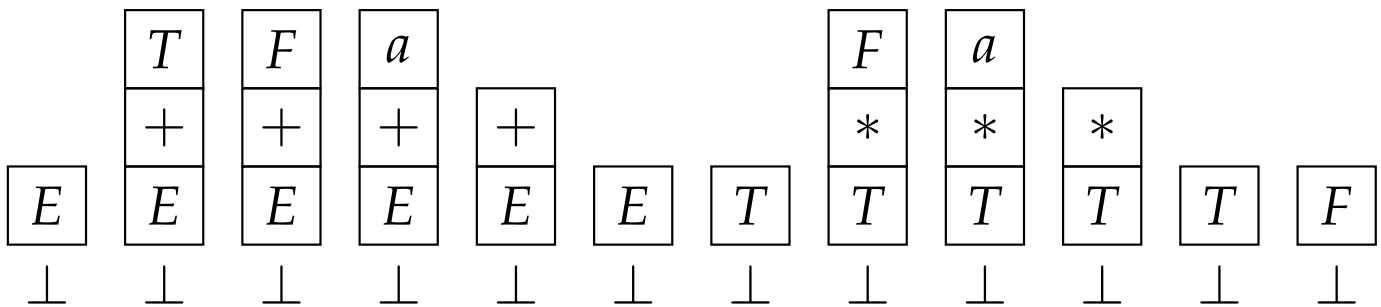
Exemple

Reconnaissance du mot :

$$a + a * a$$

avec la grammaire :

$$E \to T + E \mid T$$
$$T \to F * T \mid F$$
$$F \to (E) \mid a$$



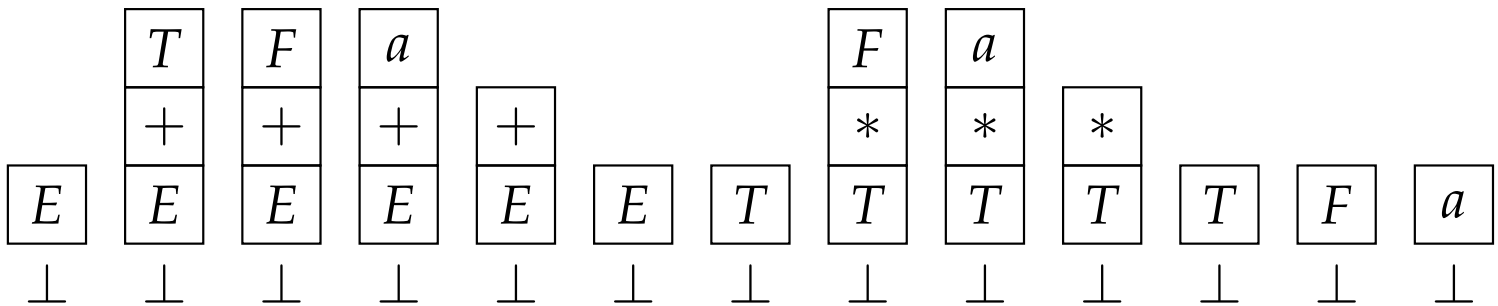
Exemple

Reconnaissance du mot :

$$a + a * a$$

avec la grammaire :

$$E \to T + E \mid T$$
$$T \to F * T \mid F$$
$$F \to (E) \mid a$$



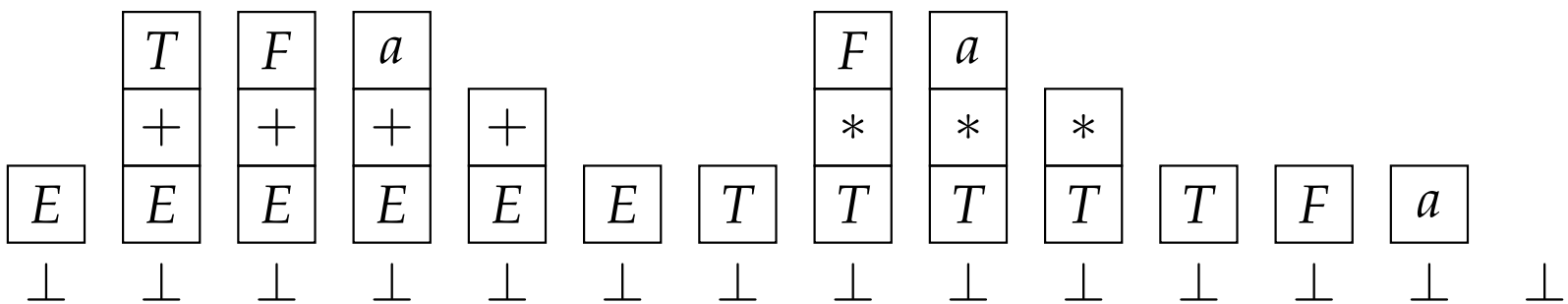
Exemple

Reconnaissance du mot :

$$a + a * a$$

avec la grammaire :

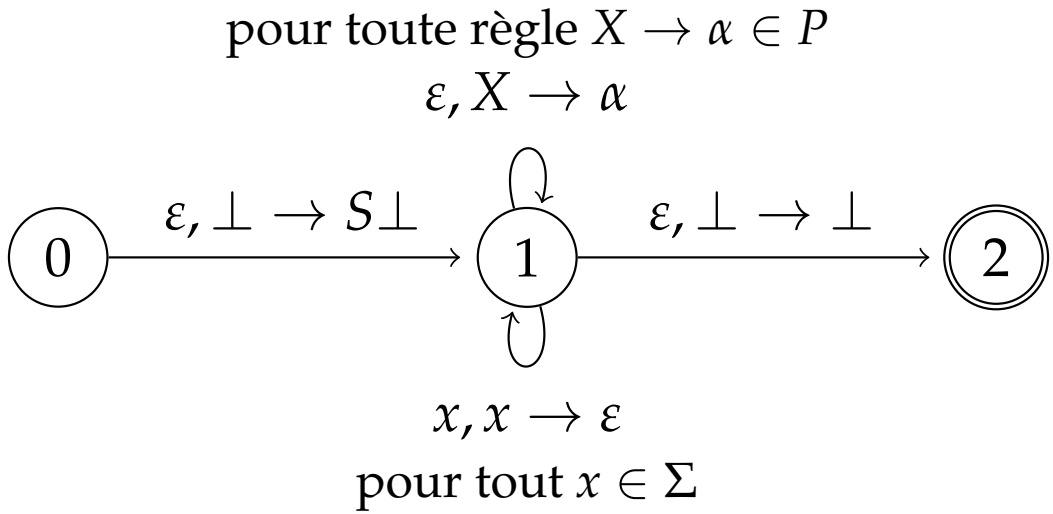
$$E \to T + E \mid T$$
$$T \to F * T \mid F$$
$$F \to (E) \mid a$$



Non déterminisme

- Lorsqu'un non terminal X doit être remplacé au sommet de la pile, il peut l'être par la partie droite d'une règle de la forme $X \to \beta$.
- Plusieurs règles de cette forme peuvent exister dans la grammaire.
- L'automate correspondant est généralement non déterministe.

Automate correspondant à la grammaire $G = \langle N, \Sigma, P, S \rangle$



Construction de l'automate

Automate à pile A correspondant à la grammaire $G = \langle N, \Sigma, P, S \rangle$:

$$A = \langle \{0, 1, 2\}, \Sigma, N \cup \Sigma \cup \{\bot\}, \delta, 0, \{2\} \rangle$$

La fonction de transition δ est définie de la façon suivante :

- $\delta(0, \varepsilon, \bot) = \{(1, S\bot)\}$ On empile l'axiome.
- $\delta(1, \varepsilon, X) = \{(1, \alpha)$ pour tout $X \to \alpha \in P\}$
  Si un symbole non terminal X occupe le sommet de la pile, on le remplace par la partie droite α d'une règle $X \to \alpha$.
- $\delta(1, a, a) = \{(1, \varepsilon) \mid$ avec $a \in \Sigma\}$
  Si le même symbole terminal occupe le sommet de la pile et la case courante de la bande d'entrée, on dépile.
- $\delta(1, \varepsilon, \bot) = \{(2, \bot)\}$
  Si le mot en entrée a été reconnu et que la pile ne contient que le symbole de fond de pile, on passe à l'état d'acceptation.

Construction — Exemple

Grammaire :

$$\langle \{E, T, F\}, \{a, +, *, (,)\}, P, E \rangle$$

avec :

$$P = \left\{ \begin{array}{lcl} E & \rightarrow & T + E \mid T, \\ T & \rightarrow & T * F \mid F, \\ F & \rightarrow & (E) \mid a \end{array} \right\}$$

Automate :

$$A_1 = \langle \{0, 1, 2\}, \{a, +, *, (,)\}, \{a, +, *, (,), E, T, F, \perp\}, \delta, 0, \perp, \{2\} \rangle$$

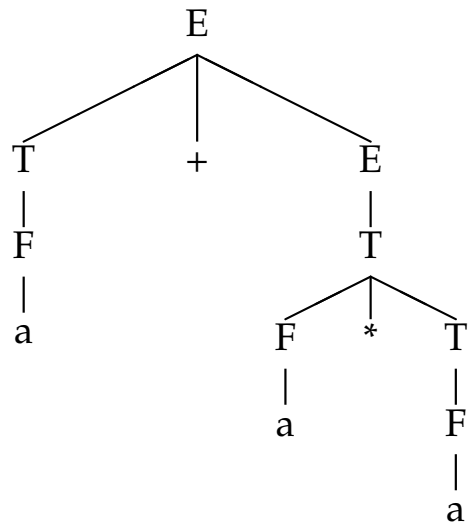
avec :

$$\begin{array}{rcl} \delta(0, \varepsilon, \perp) & = & \{(1, E\perp)\} \\ \delta(1, \varepsilon, E) & = & \{(1, T + E), (1, T)\} \\ \delta(1, \varepsilon, T) & = & \{(1, F * T), (1, F)\} \\ \delta(1, \varepsilon, F) & = & \{(1, (E)), (1, a)\} \\ \delta(1, \varepsilon, \perp) & = & \{(2, \perp)\} \end{array} \qquad \begin{array}{rcl} \delta(1, +, +) & = & \{(1, \varepsilon)\} \\ \delta(1, *, *) & = & \{(1, \varepsilon)\} \\ \delta(1, (, () & = & \{(1, \varepsilon)\} \\ \delta(1,),)) & = & \{(1, \varepsilon)\} \\ \delta(1, a, a) & = & \{(1, \varepsilon)\} \end{array}$$

Analyse syntaxique

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

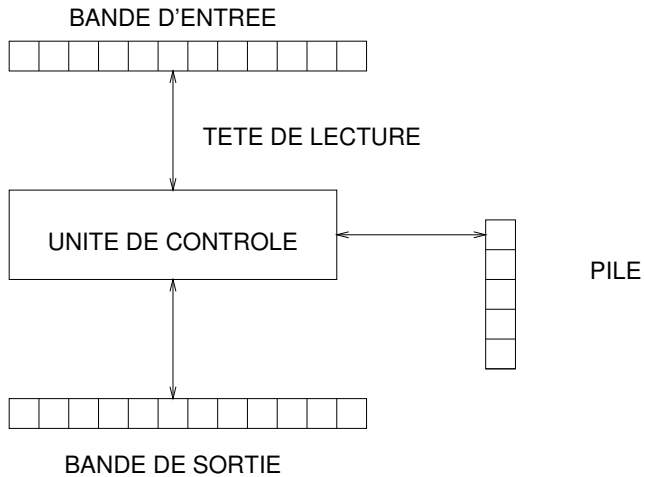
$$
\begin{array}{lcl}
E & \rightarrow & T + E \mid T \\
T & \rightarrow & F * T \mid F \\
F & \rightarrow & (E) \mid a
\end{array}
$$



Sens d'analyse

- Analyse descendante
  L'arbre de dérivation est construit depuis la racine vers les feuilles
  Séquence de dérivations gauches à partir de l'axiome
  $E \Rightarrow T + E \Rightarrow F + E \Rightarrow a + E \Rightarrow a + T \Rightarrow a + F * T \Rightarrow a + a * T \Rightarrow a + a * F \Rightarrow a + a * a$
- Analyse ascendante
  L'arbre de dérivation est construit des feuilles vers la racine
  Séquence de dérivation telle que la séquence inverse soit une dérivation droite de m.
  $a + a * a \Leftarrow F + a * a \Leftarrow T + a * a \Leftarrow T + F * a \Leftarrow T + F * F \Leftarrow T + F * T \Leftarrow T + E \Leftarrow E$

Transducteur 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) où y est une séquence de symboles de sortie.

Transducteur à pile — définition

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

- 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

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

- $q_0 \in Q$ est l'état initial
- $F \subseteq Q$ est l'ensemble des états d'acceptation

Analyseur gauche

$$
\begin{array}{llll@{\qquad}llll}
1: & E & \to & T+E & 2: & E & \to & T \\
3: & T & \to & F*T & 4: & T & \to & F \\
5: & F & \to & (E) & 6: & F & \to & a
\end{array}
$$

- Dérivation gauche de $a + a * a$:
  $E \Rightarrow T + E \Rightarrow F + E \Rightarrow a + E \Rightarrow a + T \overset{*}{\Rightarrow} a + a * a$
- Elle correspond à l'application des règles suivantes :
  $1, 4, 6, 2, 3, 6, 4, 6$

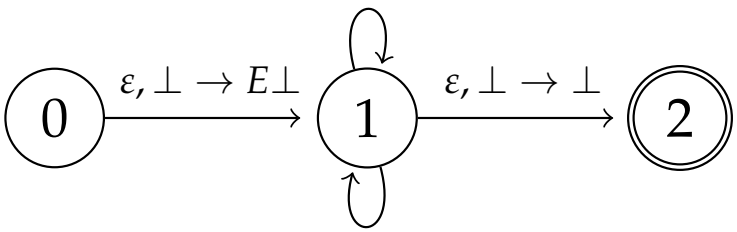
Analyseur gauche

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 un mot $m \in L(G)$, une dérivation gauche de m.
Performances :

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

Analyseur gauche : Exemple

$$
\begin{array}{lcl}
\varepsilon, E & \to & T + E, 1 \\
\varepsilon, E & \to & T, 2 \\
\varepsilon, T & \to & F * T, 3 \\
\varepsilon, T & \to & F, 4 \\
\varepsilon, F & \to & (E), 5 \\
\varepsilon, F & \to & a, 6
\end{array}
$$



$x, x \to \varepsilon$
pour tout $x \in \{a, +, *, (,)\}$

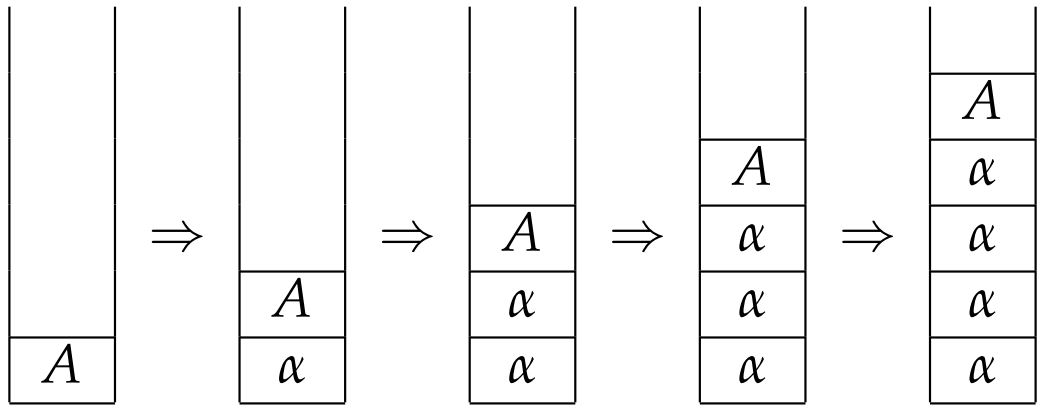
$$
\begin{array}{ll}
 & (0, a + a * a, \perp) \\
\vdash & (1, a + a * a, E\perp) \\
\vdash & (1, a + a * a, T + E\perp, 1) \\
\vdash & (1, a + a * a, F + E\perp, 14) \\
\vdash & (1, a + a * a, a + E\perp, 146) \\
\vdash & (1, +a * a, +E\perp, 146) \\
\vdash & (1, a * a, E\perp, 146) \\
\vdash & (1, a * a, T\perp, 1462) \\
\vdash & (1, a * a, F * T\perp, 14623) \\
\vdash & (1, a * a, a * T\perp, 146236) \\
\vdash & (1, *a, *T\perp, 146236) \\
\vdash & (1, a, T\perp, 1462364) \\
\vdash & (1, a, F\perp, 14623646) \\
\vdash & (1, a, a\perp, 14623646) \\
\vdash & (1, \varepsilon, \perp, 14623646) \\
\vdash & (2, \varepsilon, \varepsilon, 14623646)
\end{array}
$$

Analyse descendante prédictive : idée générale

- Rendre déterministe un analyseur gauche en s'autorisant à regarder les k symboles suivant le caractère courant dans le mot à analyser.
- Une grammaire dont l'analyseur gauche peut être rendu déterministe en regardant les k symboles suivant le caractère courant est dite $LL(k)$.
- Certaines grammaires hors contexte ne sont pas $LL(k)$, en particulier :
  - les grammaires ambiguës.
  - les grammaires récursives à gauche,
  - certaines grammaires non factorisées à gauche,

Problème de la récursivité à gauche

Si la grammaire possède une règle de la forme $A \to A\alpha$, l'automate à pile ou l'analyseur gauche correspondant bouclera !



Récursivité à gauche

- Un symbole non terminal A est dit récursif si $A \overset{*}{\Rightarrow} \alpha A \beta$ avec $\alpha, \beta \in (N \cup \Sigma)^*$.
  - Si $\alpha = \varepsilon$, A est dit récursif à gauche.
  - Si $\beta = \varepsilon$, A est dit récursif à droite.
- Une grammaire comportant au moins un symbole récursif à gauche est dite grammaire récursive à gauche.
- Une grammaire comportant au moins un symbole récursif à droite est dite grammaire récursive à droite.

Récursivité à gauche

- récursivité gauche directe : la récursivité à gauche apparaît à l'issue d'une seule dérivation.
- Exemple : application de la règle $A \rightarrow AB$ au symbole A :

$$A \Rightarrow AB$$

- récursivité gauche indirecte : la récursivité à gauche apparaît après plusieurs dérivations.
- Exemple : application sucessive des deux règles $A \rightarrow BC$ et $B \rightarrow AE$ à A :

$$A \Rightarrow BC \Rightarrow AEC$$

Bonne nouvelle !

Tout langage hors-contexte peut être engendré par une grammaire hors-contexte non récursive à gauche.

Idée générale :

$$A \to A\alpha \mid \beta \Leftrightarrow \begin{cases} A & \to & \beta A' \\ A' & \to & \alpha A' \mid \varepsilon \end{cases}$$

Elimination de la récursivité à gauche directe

Soit $G = \langle N, \Sigma, P, S \rangle$ une grammaire hors contexte, et soit

$$A \to A\alpha_1 \mid A\alpha_2 \mid \ldots \mid A\alpha_m \mid \beta_1 \mid \beta_2 \mid \ldots \mid \beta_n$$

toutes les règles de P ayant A pour partie gauche.
G engendre le même langage que la grammaire G' définie de la façon suivante :

$$G' = \langle N \cup \{A'\}, \Sigma, P', S \rangle$$

où P' est égale à P avec les règles ayant A pour partie gauche remplacées par :

$$A \to \beta_1 A' \mid \beta_2 A' \mid \ldots \mid \beta_n A'$$
$$A' \to \alpha_1 A' \mid \alpha_2 A' \mid \ldots \mid \alpha_m A' \mid \varepsilon$$

Alternative sans ε-production

$$A \to A\alpha_1 \mid A\alpha_2 \mid \ldots \mid A\alpha_m \mid \beta_1 \mid \beta_2 \mid \ldots \mid \beta_n$$

Devient

$$A \to \beta_1 \mid \beta_2 \mid \ldots \mid \beta_n \mid \beta_1 A' \mid \beta_2 A' \mid \ldots \mid \beta_n A'$$
$$A' \to \alpha_1 \mid \alpha_2 \mid \ldots \mid \alpha_m \mid \alpha_1 A' \mid \alpha_2 A' \mid \ldots \mid \alpha_m A'$$

Exemple

$$
\begin{array}{lcll}
E & \to & E+T & \mid T \\
T & \to & T*F & \mid F \\
F & \to & (E) & \mid a
\end{array}
\quad \Rightarrow \quad
\begin{array}{lcll}
E & \to & T & \mid TE' \\
E' & \to & +T & \mid +TE' \\
T & \to & F & \mid FT' \\
T' & \to & *F & \mid *FT' \\
F & \to & (E) & \mid a
\end{array}
$$

Elimination de la récursivité à gauche

- Principe : On procède de manière incrémentale en considérant des ensembles de règles de plus en plus important, jusqu'à avoir traité toute les règles.
- On ordonne les non terminaux de la grammaire : $A_1, \ldots, A_n$ et on commence par éliminer la récursivité directe des règles de la forme $A_1 \rightarrow \alpha$.
- Puis on traite les règles de la forme $A_2 \rightarrow \beta$ et ainsi de suite, jusqu'à avoir transformé toute la grammaire.

Elimination de la récursivité à gauche

Entrée : une grammaire $G = \langle N, \Sigma, P, S \rangle$
Sortie : une grammaire G' non récursive à gauche
Méthode :

- Numéroter les non terminaux de G : $N = \{A_1, \ldots, A_n\}$
- éliminer les récursivités à gauche directes des règles ayant A_1 pour partie gauche.
- Pour $i = 2$ à n faire
  - pour $j = 1$ à $i - 1$ faire
    1. remplacer chaque règle de la forme $A_i \to A_j\gamma$ par les règles $A_i \to \delta_1\gamma \mid \ldots \mid \delta_k\gamma$, où $A_j \to \delta_1 \mid \ldots \mid \delta_k$ sont toutes les règles ayant A_j pour partie gauche.
    2. éliminer les récursivités à gauche directes des règles ayant A_i pour partie gauche.

Elimination de la récursivité à gauche

- La raison pour laquelle l'algorithme ci-dessus produit l'effet voulu est qu'après la $(i-1)$ème itération de la boucle la plus externe (en i), chaque règle de la forme $A_j \rightarrow A_l\alpha$, où $j < i$ doit être telle que $l > j$.
- Il en résulte qu'à l'itération suivante dans la boucle interne (en j), les remplacements successifs de A_j dans les règles de la forme $A_i \rightarrow A_j\alpha$ va avoir pour conséquence que les règles de la forme $A_i \rightarrow A_l\alpha$ seront telles que $l \geq i$ et l'élimination de la récursivité directe sur A_i va faire que $l > i$.

Exemple

$$
\begin{array}{lcl}
A & \rightarrow & BC \mid a \\
B & \rightarrow & CA \mid Ab \\
C & \rightarrow & AB \mid CC \mid a
\end{array}
$$

- On pose $A < B < C$.
- On commence par éliminer la récursivité directe sur A puis on remplace, dans $B \rightarrow Ab$, A par $BC \mid a$ puis on élimine la récursivité directe sur B.
- On remplace alors, dans $C \rightarrow AB$, A par $BC \mid a$, ce qui donne $C \rightarrow BCB \mid aB \mid CC \mid a$.
- Puis on remplace B par $CA \mid ab \mid CAB' \mid abB'$ et on termine en éliminant la récursivité directe sur C.

Exemple

$$
\begin{array}{lcl}
A & \to & BC \mid a \\
B & \to & CA \mid Ab \\
C & \to & AB \mid CC \mid a
\end{array}
$$

- On pose $A_1 = A$, $A_2 = B$ et $A_3 = C$.

$i = 1$		pas de changements
$i = 2$	$j = 1$	$B \to CA \mid BCb \mid ab$
$i = 2$		$B \to CA \mid ab \mid CAB' \mid abB'$
		$B' \to CbB' \mid Cb$
$i = 3$	$j = 1$	$C \to BCB \mid aB \mid CC \mid a$
$i = 3$	$j = 2$	$C \to CACB \mid abCB \mid CAB'CB \mid abB'CB \mid aB \mid CC \mid a$
$i = 3$		$C \to abCB \mid abB'CB \mid aB \mid a \mid abCBC' \mid abB'CBC' \mid aBC' \mid aC'$
		$C' \to ACBC' \mid AB'CBC' \mid CC' \mid ACB \mid AB'B \mid C$

Grammaire factorisée à gauche

Une grammaire G est dite factorisée à gauche si les parties droites de deux règles ayant la même partie gauche n'ont pas de préfixe commun propre : ($A \to \alpha\beta_1 \mid \alpha\beta_2$ avec $\alpha \neq \varepsilon$).

Factorisation à gauche

Entrée : une grammaire G
Sortie : une grammaire équivalente factorisée à gauche
Méthode : Pour chaque symbole non terminal A, trouver le plus long préfixe $\alpha \neq \varepsilon$ commun à deux règles ou plus ayant A pour partie gauche. Remplacer toutes les règles ayant A pour partie gauche :

$$A \to \alpha\beta_1 \mid \alpha\beta_2 \mid \ldots \mid \alpha\beta_n \mid \gamma$$

où γ représente toutes les parties droites qui ne commencent pas par α, par :

$$A \to \alpha A' \mid \gamma$$

$$A' \to \beta_1 \mid \beta_2 \mid \ldots \mid \beta_n$$

Exemple

$$G = \langle \{E, S\}, \{i, t, e, a, b\}, \left\{ \begin{array}{c} S \to iEtS \mid iEtSeS \mid a, \\ E \to b \end{array} \right\}, S \rangle$$

Factorisée à gauche, cette grammaire devient :

$$G = \langle \{E, S, E'\}, \{i, t, e, a, b\}, \left\{ \begin{array}{c} S \to iEtSS' \mid a, \\ S' \to eS \mid \varepsilon, \\ E \to b \end{array} \right\}, S \rangle$$

Exemples

$LL(1)$

$$\begin{array}{lcl} A & \to & aB \\ A & \to & bC \end{array}$$

$LL(1)$

$$\begin{array}{lcl} A & \to & BC \\ A & \to & DE \\ B & \to & a \\ D & \to & b \end{array}$$

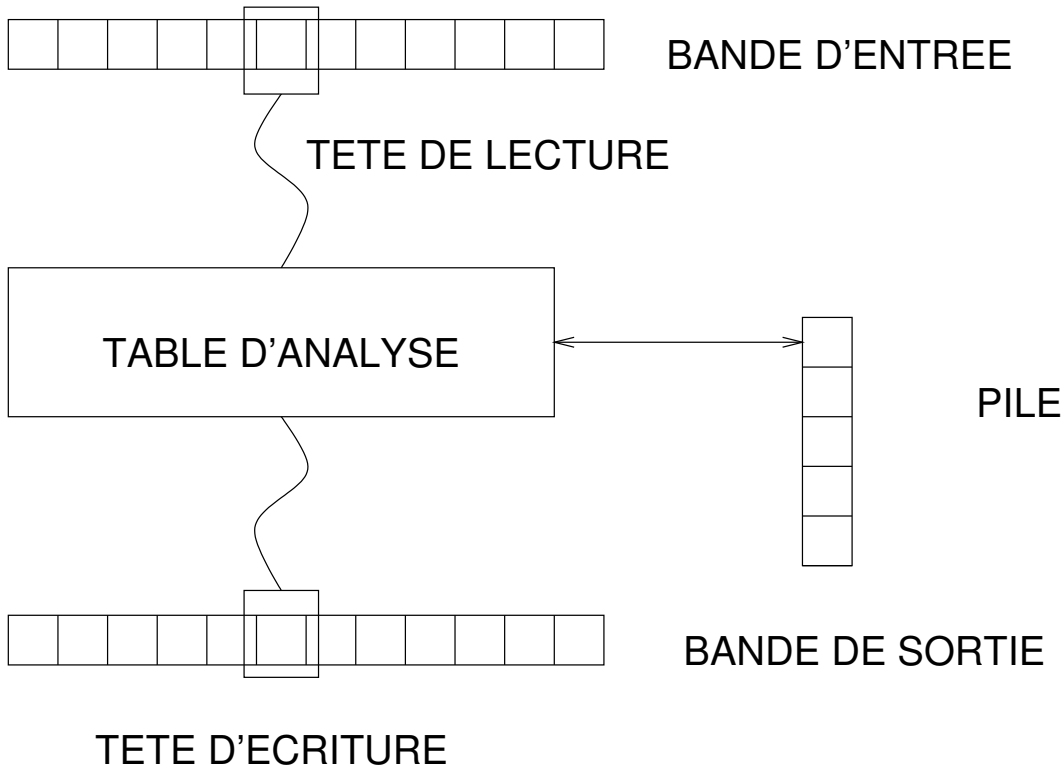
$LL(2)$

$$\begin{array}{lcl} A & \to & BC \\ A & \to & DE \\ B & \to & a \\ D & \to & a \\ C & \to & c \\ E & \to & e \end{array}$$

Grammaires $LL(1)$

- Soit $G = \langle N, \Sigma, P, S \rangle$ une grammaire hors-contexte non ambiguë et $m = a_1 \dots a_n$ un mot de $L(G)$.
- On sait qu'il existe une unique dérivation gauche du mot m composée des proto-mots $\alpha_1 \dots \alpha_k$ avec $\alpha_1 = S$ et $\alpha_k = m$.
- Idée de l'analyse $LL(1)$: construire cette suite de proto-mots en ne lisant m qu'une fois, de gauche à droite.
- Principe : si $\alpha_i = a_1 \dots a_j A \beta$ alors α_{i+1} doit pouvoir être déterminée de façon unique en fonction du symbole non terminal A et du symbole a_{j+1}.
- Une grammaire possédant cette propriété est dite grammaire $LL(1)$.

Analyseurs $LL(1)$



Configuration

Une configuration d'un analyseur LL est un triplet $(au, X\alpha, \pi)$ où :

- au représente la partie du mot d'entrée non encore lue, a est le symbole terminal se trouvant sous la tête de lecture.
- $X\alpha$ représente le contenu de la pile (avec X au sommet de cette dernière)
- π représente le mot produit sur la bande de sortie.

Si $m \in \Sigma^*$ est le mot à analyser,

- la configuration initiale de l'analyseur est : $(m\perp, S\perp, \varepsilon)$ $\perp$ étant le symbole de fond de pile qui sert aussi à marquer la fin de la chaîne à analyser.
- Une configuration d'acceptation se présente sous la forme : $(\perp, \perp, \pi)$ π étant l'analyse gauche de m.

Mouvements

Trois cas possibles à partir de la configuration $(au, X\alpha, \pi)$:

1. Si $X = a = \bot$, l'analyseur s'arrête et annonce le succès de l'analyse.
2. Si $X = a \neq \bot$, l'analyseur enlève X de la pile et avance la tête de lecture :

$$(au, a\alpha, \pi) \vdash (u, \alpha, \pi)$$

3. Si X est un symbole non terminal, l'analyseur consulte l'entrée $M(X, a)$ de la table d'analyse M. Deux cas sont possibles :
   1. $M(X, a) = i$ où i est le numéro d'une règle ayant X pour partie gauche $(X \to \beta)$. Dans ce cas, X est dépilé, β est empilé et i est écrit sur la bande de sortie.

$$(au, X\alpha, \pi) \vdash (au, \beta\alpha, \pi i)$$

   2. $M(X, a) =$ `erreur`, l'analyse s'arrête et annonce l'échec de l'analyse.

Exemple

Grammaire :

$1 : E \to TE'$
$2 : E' \to +TE' \quad 3 : E' \to \varepsilon$
$4 : T \to FT'$
$5 : T' \to *FT' \quad 6 : T' \to \varepsilon$
$7 : F \to (E) \quad 8 : F \to a$

	a	$($	$)$	$+$	$*$	$\perp$
E	1	1	$\times$	$\times$	$\times$	$\times$
E'	$\times$	$\times$	3	2	$\times$	3
T	4	4	$\times$	$\times$	$\times$	$\times$
T'	$\times$	$\times$	6	6	5	6
F	8	7	$\times$	$\times$	$\times$	$\times$

Exécution :

$(a * a)\perp$

E
$\perp$

Exemple

Grammaire :

$1 : E \to TE'$
$2 : E' \to +TE' \quad 3 : E' \to \varepsilon$
$4 : T \to FT'$
$5 : T' \to *FT' \quad 6 : T' \to \varepsilon$
$7 : F \to (E) \quad 8 : F \to a$

	a	$($	$)$	$+$	$*$	$\perp$
E	1	1	$\times$	$\times$	$\times$	$\times$
E'	$\times$	$\times$	3	2	$\times$	3
T	4	4	$\times$	$\times$	$\times$	$\times$
T'	$\times$	$\times$	6	6	5	6
F	8	7	$\times$	$\times$	$\times$	$\times$

Exécution :

$(a * a)\perp$

T
E'
$\perp$

Exemple

Grammaire :

$1 : E \to TE'$
$2 : E' \to +TE' \quad 3 : E' \to \varepsilon$
$4 : T \to FT'$
$5 : T' \to *FT' \quad 6 : T' \to \varepsilon$
$7 : F \to (E) \quad 8 : F \to a$

	a	$($	$)$	$+$	$*$	$\perp$
E	1	1	$\times$	$\times$	$\times$	$\times$
E'	$\times$	$\times$	3	2	$\times$	3
T	4	4	$\times$	$\times$	$\times$	$\times$
T'	$\times$	$\times$	6	6	5	6
F	8	7	$\times$	$\times$	$\times$	$\times$

Exécution :

$(a * a)\perp$

F
T'
E'
$\perp$

Exemple

Grammaire :

$1 : E \to TE'$
$2 : E' \to +TE' \quad 3 : E' \to \varepsilon$
$4 : T \to FT'$
$5 : T' \to *FT' \quad 6 : T' \to \varepsilon$
$7 : F \to (E) \quad 8 : F \to a$

	a	$($	$)$	$+$	$*$	$\perp$
E	1	1	$\times$	$\times$	$\times$	$\times$
E'	$\times$	$\times$	3	2	$\times$	3
T	4	4	$\times$	$\times$	$\times$	$\times$
T'	$\times$	$\times$	6	6	5	6
F	8	7	$\times$	$\times$	$\times$	$\times$

Exécution :

$(a * a)\perp$

$($
E
$)$
T'
E'
$\perp$

Exemple

Grammaire :

$1 : E \to TE'$
$2 : E' \to +TE' \quad 3 : E' \to \varepsilon$
$4 : T \to FT'$
$5 : T' \to *FT' \quad 6 : T' \to \varepsilon$
$7 : F \to (E) \quad 8 : F \to a$

	a	$($	$)$	$+$	$*$	$\perp$
E	1	1	$\times$	$\times$	$\times$	$\times$
E'	$\times$	$\times$	3	2	$\times$	3
T	4	4	$\times$	$\times$	$\times$	$\times$
T'	$\times$	$\times$	6	6	5	6
F	8	7	$\times$	$\times$	$\times$	$\times$

Exécution :

$$a * a) \perp$$

E
$)$
T'
E'
$\perp$

Exemple

Grammaire :

$1 : E \to TE'$
$2 : E' \to +TE' \quad 3 : E' \to \varepsilon$
$4 : T \to FT'$
$5 : T' \to *FT' \quad 6 : T' \to \varepsilon$
$7 : F \to (E) \quad 8 : F \to a$

	a	$($	$)$	$+$	$*$	$\perp$
E	1	1	$\times$	$\times$	$\times$	$\times$
E'	$\times$	$\times$	3	2	$\times$	3
T	4	4	$\times$	$\times$	$\times$	$\times$
T'	$\times$	$\times$	6	6	5	6
F	8	7	$\times$	$\times$	$\times$	$\times$

Exécution :

$a * a) \perp$

T
E'
$)$
T'
E'
$\perp$

Exemple

Grammaire :

$1 : E \to TE'$
$2 : E' \to +TE' \quad 3 : E' \to \varepsilon$
$4 : T \to FT'$
$5 : T' \to *FT' \quad 6 : T' \to \varepsilon$
$7 : F \to (E) \quad 8 : F \to a$

	a	$($	$)$	$+$	$*$	$\perp$
E	1	1	$\times$	$\times$	$\times$	$\times$
E'	$\times$	$\times$	3	2	$\times$	3
T	4	4	$\times$	$\times$	$\times$	$\times$
T'	$\times$	$\times$	6	6	5	6
F	8	7	$\times$	$\times$	$\times$	$\times$

Exécution :

$a * a) \perp$

F
T'
E'
$)$
T'
E'
$\perp$

Exemple

Grammaire :

$1 : E \to TE'$
$2 : E' \to +TE' \quad 3 : E' \to \varepsilon$
$4 : T \to FT'$
$5 : T' \to *FT' \quad 6 : T' \to \varepsilon$
$7 : F \to (E) \quad 8 : F \to a$

	a	$($	$)$	$+$	$*$	$\perp$
E	1	1	$\times$	$\times$	$\times$	$\times$
E'	$\times$	$\times$	3	2	$\times$	3
T	4	4	$\times$	$\times$	$\times$	$\times$
T'	$\times$	$\times$	6	6	5	6
F	8	7	$\times$	$\times$	$\times$	$\times$

Exécution :

$a * a) \perp$

a
T'
E'
$)$
T'
E'
$\perp$

Exemple

Grammaire :

$1 : E \to TE'$
$2 : E' \to +TE' \quad 3 : E' \to \varepsilon$
$4 : T \to FT'$
$5 : T' \to *FT' \quad 6 : T' \to \varepsilon$
$7 : F \to (E) \quad 8 : F \to a$

	a	$($	$)$	$+$	$*$	$\perp$
E	1	1	$\times$	$\times$	$\times$	$\times$
E'	$\times$	$\times$	3	2	$\times$	3
T	4	4	$\times$	$\times$	$\times$	$\times$
T'	$\times$	$\times$	6	6	5	6
F	8	7	$\times$	$\times$	$\times$	$\times$

Exécution :

$$* a) \perp$$

T'
E'
$)$
T'
E'
$\perp$

Exemple

Grammaire :

$1 : E \to TE'$
$2 : E' \to +TE' \quad 3 : E' \to \varepsilon$
$4 : T \to FT'$
$5 : T' \to *FT' \quad 6 : T' \to \varepsilon$
$7 : F \to (E) \quad 8 : F \to a$

	a	$($	$)$	$+$	$*$	$\perp$
E	1	1	$\times$	$\times$	$\times$	$\times$
E'	$\times$	$\times$	3	2	$\times$	3
T	4	4	$\times$	$\times$	$\times$	$\times$
T'	$\times$	$\times$	6	6	5	6
F	8	7	$\times$	$\times$	$\times$	$\times$

Exécution :

$*\, a) \perp$

$*$
F
T'
E'
$)$
T'
E'
$\perp$

Exemple

Grammaire :

$1 : E \to TE'$
$2 : E' \to +TE' \quad 3 : E' \to \varepsilon$
$4 : T \to FT'$
$5 : T' \to *FT' \quad 6 : T' \to \varepsilon$
$7 : F \to (E) \quad 8 : F \to a$

	a	$($	$)$	$+$	$*$	$\perp$
E	1	1	$\times$	$\times$	$\times$	$\times$
E'	$\times$	$\times$	3	2	$\times$	3
T	4	4	$\times$	$\times$	$\times$	$\times$
T'	$\times$	$\times$	6	6	5	6
F	8	7	$\times$	$\times$	$\times$	$\times$

Exécution :

$a)\perp$

F
T'
E'
$)$
T'
E'
$\perp$

Exemple

Grammaire :

$1 : E \to TE'$
$2 : E' \to +TE' \quad 3 : E' \to \varepsilon$
$4 : T \to FT'$
$5 : T' \to *FT' \quad 6 : T' \to \varepsilon$
$7 : F \to (E) \quad 8 : F \to a$

	a	$($	$)$	$+$	$*$	$\perp$
E	1	1	$\times$	$\times$	$\times$	$\times$
E'	$\times$	$\times$	3	2	$\times$	3
T	4	4	$\times$	$\times$	$\times$	$\times$
T'	$\times$	$\times$	6	6	5	6
F	8	7	$\times$	$\times$	$\times$	$\times$

Exécution :

$a)\perp$

a
T'
E'
$)$
T'
E'
$\perp$

Exemple

Grammaire :

$1 : E \to TE'$
$2 : E' \to +TE' \quad 3 : E' \to \varepsilon$
$4 : T \to FT'$
$5 : T' \to *FT' \quad 6 : T' \to \varepsilon$
$7 : F \to (E) \quad 8 : F \to a$

	a	$($	$)$	$+$	$*$	$\perp$
E	1	1	$\times$	$\times$	$\times$	$\times$
E'	$\times$	$\times$	3	2	$\times$	3
T	4	4	$\times$	$\times$	$\times$	$\times$
T'	$\times$	$\times$	6	6	5	6
F	8	7	$\times$	$\times$	$\times$	$\times$

Exécution :

$)\perp$

T'
E'
$)$
T'
E'
$\perp$

Exemple

Grammaire :

$1 : E \to TE'$
$2 : E' \to +TE' \quad 3 : E' \to \varepsilon$
$4 : T \to FT'$
$5 : T' \to *FT' \quad 6 : T' \to \varepsilon$
$7 : F \to (E) \quad 8 : F \to a$

	a	$($	$)$	$+$	$*$	$\perp$
E	1	1	$\times$	$\times$	$\times$	$\times$
E'	$\times$	$\times$	3	2	$\times$	3
T	4	4	$\times$	$\times$	$\times$	$\times$
T'	$\times$	$\times$	6	6	5	6
F	8	7	$\times$	$\times$	$\times$	$\times$

Exécution :

$)\perp$

E'
$)$
T'
E'
$\perp$

Exemple

Grammaire :

$1 : E \to TE'$
$2 : E' \to +TE' \quad 3 : E' \to \varepsilon$
$4 : T \to FT'$
$5 : T' \to *FT' \quad 6 : T' \to \varepsilon$
$7 : F \to (E) \quad 8 : F \to a$

	a	$($	$)$	$+$	$*$	$\perp$
E	1	1	$\times$	$\times$	$\times$	$\times$
E'	$\times$	$\times$	3	2	$\times$	3
T	4	4	$\times$	$\times$	$\times$	$\times$
T'	$\times$	$\times$	6	6	5	6
F	8	7	$\times$	$\times$	$\times$	$\times$

Exécution :

$)\perp$

$)$
T'
E'
$\perp$

Exemple

Grammaire :

$1 : E \to TE'$
$2 : E' \to +TE' \quad 3 : E' \to \varepsilon$
$4 : T \to FT'$
$5 : T' \to *FT' \quad 6 : T' \to \varepsilon$
$7 : F \to (E) \quad 8 : F \to a$

	a	$($	$)$	$+$	$*$	$\perp$
E	1	1	$\times$	$\times$	$\times$	$\times$
E'	$\times$	$\times$	3	2	$\times$	3
T	4	4	$\times$	$\times$	$\times$	$\times$
T'	$\times$	$\times$	6	6	5	6
F	8	7	$\times$	$\times$	$\times$	$\times$

Exécution :

$\perp$

T'
E'
$\perp$

Exemple

Grammaire :

$1 : E \to TE'$
$2 : E' \to +TE' \quad 3 : E' \to \varepsilon$
$4 : T \to FT'$
$5 : T' \to *FT' \quad 6 : T' \to \varepsilon$
$7 : F \to (E) \quad 8 : F \to a$

	a	$($	$)$	$+$	$*$	$\perp$
E	1	1	$\times$	$\times$	$\times$	$\times$
E'	$\times$	$\times$	3	2	$\times$	3
T	4	4	$\times$	$\times$	$\times$	$\times$
T'	$\times$	$\times$	6	6	5	6
F	8	7	$\times$	$\times$	$\times$	$\times$

Exécution :

$\perp$

E'
$\perp$

Exemple

Grammaire :

$1 : E \to TE'$
$2 : E' \to +TE' \quad 3 : E' \to \varepsilon$
$4 : T \to FT'$
$5 : T' \to *FT' \quad 6 : T' \to \varepsilon$
$7 : F \to (E) \quad 8 : F \to a$

	a	$($	$)$	$+$	$*$	$\perp$
E	1	1	$\times$	$\times$	$\times$	$\times$
E'	$\times$	$\times$	3	2	$\times$	3
T	4	4	$\times$	$\times$	$\times$	$\times$
T'	$\times$	$\times$	6	6	5	6
F	8	7	$\times$	$\times$	$\times$	$\times$

Exécution :

$\perp$

$\boxed{\perp}$

Construction d'une table $LL(1)$ à partir d'une grammaire

La construction d'une table d'analyse $LL(1)$ pour une grammaire $G = \langle N, \Sigma, P, S \rangle$ est facilitée par les deux fonctions PREMIER et SUIVANT.
Ces deux fonctions pemettent, quand c'est possible, de remplir les entrées de la table d'analyse $LL(1)$ de G.

PREMIER

Si α est un proto-mot de G, $\text{PREMIER}(\alpha)$ est l'ensemble des terminaux qui commencent les chaînes se dérivant de α :

$$\text{PREMIER}(\alpha) = \{a \in \Sigma \mid \alpha \stackrel{*}{\Rightarrow} au\}$$

Si $\alpha \stackrel{*}{\Rightarrow} \varepsilon$ alors ε appartient aussi à $\text{PREMIER}(\alpha)$.

PREMIER(X)

Pour calculer PREMIER(X) avec $X \in N \cup \Sigma$, on applique les règles suivantes jusqu'à ce qu'aucun terminal ni ε ne puisse être ajouté aux ensembles PREMIER.

1. Si $X \in \Sigma$, PREMIER(X) = $\{X\}$.
2. Si $X \to \varepsilon \in P$, on ajoute ε à PREMIER(X).
3. Si $X \in N$ et $X \to Y_1 \dots Y_k \in P$, mettre a dans PREMIER(X) s'il existe i tel que a est dans PREMIER(Y_i) et que ε est dans tous les PREMIER(Y_1) ... PREMIER(Y_{i-1}).
   Si $\varepsilon \in$ PREMIER(Y_j)$\forall j$, $1 \leq j \leq k$, on ajoute ε à PREMIER(X).

PREMIER$(X_1 \dots X_n)$

On calcule PREMIER$(X_1 \dots X_n)$ de la façon suivante :

1. Ajouter à PREMIER$(X_1 \dots X_n)$ tous les symboles de PREMIER(X_1) différents de ε.
2. Si $\varepsilon \in$ PREMIER(X_1), ajouter également les symboles de PREMIER(X_2) différents de ε.
   Si $\varepsilon \in$ PREMIER(X_2), ajouter également les symboles de PREMIER(X_3) différents de ε, etc.
3. Finalement, si ε appartient à PREMIER(X_j) pour tous les $j = 1, 2, \dots n$, on ajoute ε à PREMIER$(X_1 \dots X_n)$.

SUIVANT(X)

Si $X \in N$, SUIVANT(X) est l'ensemble des symboles $a \in \Sigma$ qui peuvent apparaître immédiatement à droite de X dans un proto-mot :

$$\text{SUIVANT}(X) = \{a \in \Sigma \mid S \overset{*}{\Rightarrow} \alpha X a \beta\}$$

Si X peut être le symbole le plus à droite d'un proto-mot alors $\perp$ est dans SUIVANT(X).

SUIVANT(X)

Pour calculer SUIVANT(X) pour tous symbole non terminal X, on applique les règles suivantes jusqu'à ce qu'aucun symbole terminal ne puisse être ajouté aux ensembles SUIVANT :

1. Mettre $\perp$ dans SUIVANT(S).
2. si $X \to \alpha B \beta$, le contenu de PREMIER(β), excepté ε, est ajouté à SUIVANT(B).
3. s'il existe une règle $X \to \alpha B$ ou une règle $X \to \alpha B \beta$ telle que $\varepsilon \in$ PREMIER(β) (c'est à dire $\beta \overset{*}{\Rightarrow} \varepsilon$), les éléments de SUIVANT(X) sont ajoutés à SUIVANT(B).

Exemple

Soit la grammaire $G = \langle \{E, E', T, T', F\}, \{a, +, *, (,), a\}, P, E \rangle$ non récursive à gauche où P est composé des règles suivantes :

$$
\begin{array}{ll}
1\ E \to TE' & 2\ E' \to +TE' \\
3\ E' \to \varepsilon & 4\ T \to FT' \\
5\ T' \to *FT' & 6\ T' \to \varepsilon \\
7\ F \to (E) & 8\ F \to a
\end{array}
$$

Alors :

$\text{PREMIER}(E) = \text{PREMIER}(T) = \text{PREMIER}(F) = \{(, a\}$
$\text{PREMIER}(E') = \{+, \varepsilon\}$
$\text{PREMIER}(T') = \{*, \varepsilon\}$
$\text{SUIVANT}(E) = \{), \bot\}$
$\text{SUIVANT}(E') = \text{SUIVANT}(E) = \{), \bot\}$
$\text{SUIVANT}(T) = \{\text{PREMIER}(E') - \{\varepsilon\}\} \cup \text{SUIVANT}(E) = \{+,), \bot\}$
$\text{SUIVANT}(T') = \text{SUIVANT}(T) = \{+,), \bot\}$
$\text{SUIVANT}(F) = \{\text{PREMIER}(T') - \{\varepsilon\}\} \cup \text{SUIVANT}(T) = \{+, *,), \bot\}$

Construction de la table $LL(1)$

Entrée : $G = \langle N, \Sigma, P, S \rangle$ Une grammaire dont les règles sont numérotées.
Sortie : M Une table d'analyse $LL(1)$ pour G.
Méthode :

1. pour chaque regle $i \in P$ de la forme $A \to \alpha$, procéder aux étapes 2 et 3.
2. Pour chaque symbole terminal $a \in \text{PREMIER}(\alpha)$, ajouter i à $M(A, a)$.
3. Si $\varepsilon \in \text{PREMIER}(\alpha)$, ajouter i à $M(A, b)$ pour chaque symbole terminal $b \in \text{SUIVANT}(A)$. Si $\varepsilon \in \text{PREMIER}(\alpha)$ et $\perp \in \text{SUIVANT}(A)$, ajouter i à $M(A, \perp)$.
4. Mettre `erreur` ($\times$) dans toutes les entrées restées vides.

Grammaires non $LL(1)$

Si G n'est pas $LL(1)$, en particulier si elle est récursive à gauche, non factorisée à gauche ou ambiguë, M peut avoir des entrées qui sont définies de façons multiples.
On peut montrer qu'une grammaire G est $LL(1)$ si et seulement si, pour toute règle disctincte $A \to \alpha$ et $A \to \beta$ de G, les conditions suivantes s'appliquent :

1. Pour aucun symbole terminal a, α et β ne se dérivent toutes les deux en des mots commençant par a.
2. Un des deux proto-mots α et β peut se dériver en ε.
3. Si $\beta \overset{*}{\Rightarrow} \varepsilon$, α ne se dérive pas en un mot commençant par un élément de $\text{SUIVANT}(A)$ et vice-versa.

Réalisation simple d'un analyseur $LL(1)$ en C

Principes généraux :

- G est une grammaire $LL(1)$.
- Une fonction en langage C est associée à tout symbole non terminal de G.
- Le graphe des appels de fonctions représente l'arbre de dérivation.

Cas de base

$A \rightarrow B$

```
void A(void){
  if(est_premier(cc, _B_)){
    B();
    return;}
  erreur();}
```

$A \rightarrow a$

```
void A(void){
  if(cc == 'a'){
    cc = yylex();
    return;}
  erreur();}
```

Parties droites complexes

$A \rightarrow BC$

```
void A(void){
  if(est_premier(cc, _B_)){
    B();
    if(est_premier(cc, _C_)){
      C();
      return;}}
    erreur();}
```

$A \rightarrow aB$

```
void A(void){
  if(cc == 'a'){
    cc = yylex();
    if(est_premier(cc, _B_)){
      B();
      return;}}
    erreur();}
```

Redondance

Certains tests sont effectués plusieurs fois
$A \to B$ et $B \to b$ avec $\text{PREMIER}(B) = \{b\}$

```
void A(void){
  if(est_premier(cc, _B_)){ /* premiere fois */
    B();
    return;}
  erreur();}

void B(void){
  if(cc == 'b'){ /* deuxieme fois */
    cc = yylex();
    return;}
  erreur();}
```

ça permet de détecter au plus tôt une erreur de syntaxe

Symboles ambigüs

$A \rightarrow B \mid C$

```
void A(void){
  if(est_premier(cc, _B_)){
    B();
    return;}
  if(est_premier(cc, _C_)){
    C();
    return;}
  erreur();}
```

Règles vides

$A \rightarrow B \mid \varepsilon$

```
void A(void){
  if(est_premier(cc, _B_)){
    B();
    return;}
  if(est_suivant(cc, _A_)){
    return;}
  erreur();}
```

Règles vides

De manière plus générale
$A \rightarrow B$ avec $\varepsilon \in \text{PREMIER}(B)$

```
void A(void){
  if(est_premier(cc, _B_)){
    B();
    return;}
  if(est_suivant(cc, _A_)){
    return;}
  erreur();}
```

est_premier

```
int premiers[NB_NON_TERMINAUX+1][NB_TERMINAUX+1];

void initialise_premiers(void){
  int i,j;
  for(i=0; i <= NB_NON_TERMINAUX; i++)
    for(j=0; j <= NB_TERMINAUX; j++)
      premiers[i][j] = 0;

  premiers[_B_][b] = 1;
  ...
  }

int est_premier(int non_terminal, int terminal){
  return premiers[terminal][non_terminal];
}
```

Arbre de dérivation

- Il n'est pas nécessaire de construire explicitement l'arbre de dérivation sous-jacent à une analyse syntaxique.
- L'arbre abstrait sera constuit directement lors de l'analyse, par ajout d'actions sémantiques.
- Il est cependant utile de pouvoir visualiser l'arbre de dérivation pour des raisons de mise au point.
- On peut pour cela produire de manière simple un fichier XML qui représente la structure de l'arbre de dérivation, qu'il suffira d'ouvrir à l'aide d'un outil de visualisation de fichiers XML.

Affichage d'un arbre de dérivation XML

```
void A(void){
  fprintf(sortie_xml, "<A>\n");
  if(est_premier(cc, _B_){
    B();
    fprintf(sortie_xml, "</A>\n");
    return;}
  if(est_suivant(cc, _A_)){
    fprintf(sortie_xml, "</A>\n");
    return;}
  erreur();}
```

Un peu plus joli

```
void A(void){
  balise_ouvrante(__FUNCTION__);
  if(est_premier(cc, _B_){
    B();
    balise_fermante(fonction);
    return;}
  if(est_suivant(cc, _A_)){
    balise_fermante(__FUNCTION__);
    return;}
  erreur();}

void balise_ouvrante(char *fonction) {
  fprintf(sortie_xml, "<%s>\n", fonction);}

void balise_fermante(char *fonction) {
  fprintf(sortie_xml, "</%s>\n", fonction);}
```