# Analyseur syntaxique

# Sommaire

- Problématique
- Grammaires et arbres de dérivation
- Mise en œuvre d'un analyseur syntaxique
- Analyse descendante
- Table d'analyse LL1
  - Calcul des premiers
  - Calcul des suivants
  - Construction de la table LL1
- Analyseur syntaxique

# Problématique

## Préambule

Tout langage de programmation possède des règles qui indiquent la structure syntaxique d'un programme bien formé. Par exemple, en Pascal, un programme bien formé est composé de blocs, un bloc est formé d'instructions, une instruction de ...

La **syntaxe** d'un langage peut être décrite par une **grammaire**. Cette grammaire dérits comment les unités lexicales doivent être agencées.

L'analyseur syntaxique reçoit une suite d'unités lexicales de la part de l'analyseur lexical et doit vérifier que cette suite peut être engendrée par la grammaire du langage.

Le problème est donc

$$\left\{ \begin{array}{l} \bullet \text{ étant donnée une grammaire } G \\ \bullet \text{ étant donné un mot } m \text{ (un programme)} \end{array} \right.$$

$\Longrightarrow$ est ce que $m$ appartient au langage généré par $G$ ?

Le principe est d'essayer de construire un **arbre de dérivation**. Il existe deux méthodes pour cette construction : méthode (analyse) descendante et méthode (analyse) ascendante.

# Grammaires et arbres de dérivation

## La grammaire par les exemples

Exemples (définitions informelles de grammaires) :
dans le langage naturel, une phrase est composée d'un sujet suivi d'un verbe suivi d'un complément (pour simplifier ...).
Par exemple : $\boxed{\text{L'étudiant}}$ $\boxed{\text{subit}}$ $\boxed{\text{un cours}}$
On dira donc :

*phrase* = *sujet verbe complément*

Ensuite, il faut expliquer ce qu'est un *sujet*, un *verbe*, un *complément*. Par exemple :

*sujet* = *article adjectif nom*
| *article nom adjectif*
| *article nom*
*article* = `le` | `la` | `un` | `des` | `l'`
*adjectif* = `malin` | `stupide` | *couleur*
*couleur* = `vert` | `rouge` | `jaune`
ainsi de suite ...

une expression conditionnelle en C est : `if (` *expression* `)` *instruction*
Par exemple : `if (` $\boxed{x < 10}$ `)` $\boxed{a = a + b}$
Il faut encore définir ce qu'est une *expression* et ce qu'est une *instruction* ...
On distingue les

- *symboles* **terminaux** : les lettres du langage (`le`, `la`, `if` ...dans les exemples)
- *symboles* **non-terminaux** : les symboles qu'il faut encore définir (ceux en italique dans les exemples précédents)

# Grammaires et arbres de dérivation

## Définition 1

*Une* **grammaire** *est la donnée de* $G = (V_T, V_N, S, P)$ *où*

- $V_T$ *est un ensemble non vide de symboles terminaux (alphabet terminal)*
- $V_N$ *est un ensemble de symboles non-terminaux, avec* $V_T \bigcap V_N = \emptyset$
- $S$ *est un symbole initial* $\in V_N$ *appelé* **axiome**
- $P$ *est un ensemble de* **règles de productions** *(règles de réécritures)*

## Définition 2

*Une* **règle de production** $\alpha \to \beta$ *précise que la séquence de symboles* $\alpha$ *(* $\alpha \in (V_T \bigcup V_N)^+$*)* *peut être remplacée par la séquence de symboles* $\beta$ *(*$\beta \in (V_T \bigcup V_N)^*$*).*
$\alpha$ *est appelée* **partie gauche** *de la production, et* $\beta$ **partie droite** *de la production.*

## Exemple 1

symboles terminaux (alphabet) : $V_T = \{a, b\}$
symboles non-terminaux : $V_N = \{S\}$
axiome : $S$

règles de production :

$$\begin{cases} S \to \varepsilon \\ S \to aSb \end{cases} \quad \text{qui se résument en } S \to \varepsilon \mid aSb$$

## Exemple 2

$G =< V_T, V_N, S, P >$ avec
$V_T$ = { il, elle, parle, est, devient, court, reste, sympa, vite }
$V_N$ = { PHRASE, PRONOM, VERBE,
COMPLEMENT, VERBETAT, VERBACTION }
$S$ = PHRASE
$P$ = { PHRASE → PRONOM VERBE COMPLEMENT
PRONOM → il | elle
VERBE → VERBETAT | VERBACTION
VERBETAT → est | devient | reste
VERBACTION → parle | court
COMPLEMENT → sympa | vite }

# Arbre de dérivation

## Définition 1

On appelle **dérivation** l'application d'une ou plusieurs règles à partir d'un mot de $(V_T \bigcup V_N)^+$.
On notera $\rightarrow$ une dérivation obtenue par application d'une seule règle de production, $\xrightarrow{*}$ une dérivation obtenue par l'application de $n$ règles de production, où $n \geq 0$ et $\xrightarrow{+}$ une dérivation obtenue par l'application de $n$ règles de production, où $n > 0$

## Exemple 1

sur la grammaire de l'exemple 1

$S \rightarrow \varepsilon$
$S \rightarrow aSb$
$aSb \rightarrow aaSbb$
$S \xrightarrow{*} ab$
$S \xrightarrow{*} aaabbb$
$S \xrightarrow{*} aaSbb$

## Exemple 2

sur la grammaire de l'exemple 2

PHRASE $\rightarrow$ PRONOM VERBE COMPLEMENT $\xrightarrow{*}$ elle VERBETAT sympa
PHRASE $\xrightarrow{*}$ il parle vite
PHRASE $\xrightarrow{*}$ elle court sympa

# Arbre de dérivation

## Définition 2

*Etant donnée une grammaire $G$, on note $L(G)$ le langage généré par $G$ et défini par*
$\{w \in (V_T)^* \ \ tq\ S \xrightarrow{*} w\}$
*(c'est à dire tous les mots composés uniquement de symboles terminaux (de lettres de l'alphabet) que l'on peut former à partir de $S$).*
L'exemple 1 nous donne $L(G) = \{a^n b^n, n \geq 0\}$

## Définition 3

*On appelle* **arbre de dérivation** *(ou arbre syntaxique) tout arbre tel que*

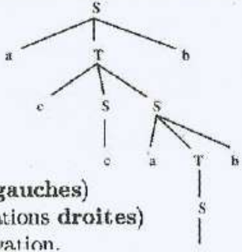
- *la racine est l'axiome*
- *les feuilles sont des symboles termianux*
- *les noeuds sont des symboles non-terminaux*
- *les fils d'un noeud $X$ sont $\beta_0, \ldots, \beta_n$ si et seulement si $X \rightarrow \beta_0 \ldots \beta_n$ est une production (avec $\beta_i \in V_T \bigcup V_N$)*

## Exemple

Soit la grammaire ayant $S$ pour axiome et pour règles de production

$$P = \begin{cases} S \rightarrow aTb \mid c \\ T \rightarrow cSS \mid S \end{cases}$$

Un arbre de dérivation pour le mot *accacbb* es




$S \rightarrow aTb \rightarrow acSSb \rightarrow accSb \rightarrow accaTbb \rightarrow accaSbb \rightarrow accacbb$ (dérivations **gauches**)
ou $S \rightarrow aTb \rightarrow acSSb \rightarrow acSaTbb \rightarrow acSaSbb \rightarrow acSacbb \rightarrow accacbb$ (dérivations **droites**)
Ces deux suites **différentes** de dérivations donnent le **même** arbre de dérivation.

# Arbre de dérivation

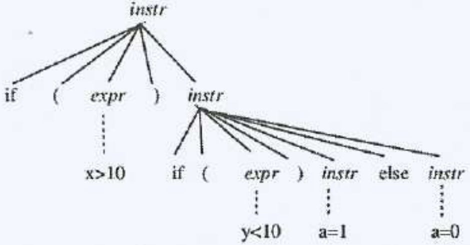
## Définition 3

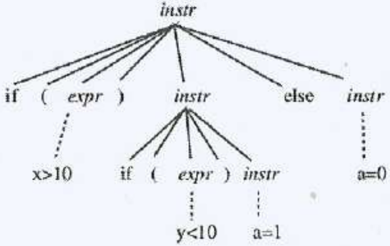
*Une grammaire est dite* **ambiguë** *s'il existe un mot de $L(G)$ ayant plusieurs arbres syntaxiques.*

Remarque : la grammaire précédente **n'est pas** ambiguë.

Exemple : Soit $G$ donnée par
$$\begin{cases} instr \rightarrow \texttt{if}\ (expr\ )\ \ instr\ \texttt{else}\ \ instr \\ instr \rightarrow \texttt{if}\ (expr\ )\ \ instr \\ instr \rightarrow \ldots \\ expr \rightarrow \ldots \end{cases}$$

Cette grammaire est ambiguë car le mot $m = \text{if}\ (x > 10\ )\ \text{if}\ (y < 0)\ a = 1\ \text{else}\ a = 0$ possède deux arbres syntaxiques différents :




car il y a deux interprétations syntaxiques possibles :

# Mise en œuvre d'un analyseur syntaxique

## Analyseur syntaxique

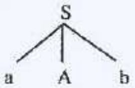
L'analyseur syntaxique reçoit une suite d'unités lexicales (de symboles terminaux) de la part de l'analyseur lexical. Il doit dire si cette suite (ce mot) est syntaxiquement correct, c'est à dire si c'est un mot du langage généré par la grammaire qu'il possède. Il doit donc essayer de construire l'arbre de dérivation de ce mot. S'il y arrive, alors le mot est syntaxiquement correct, sinon il est incorrect.
Il existe deux approches (deux méthodes) pour construire cet arbre de dérivation : une méthode descendante et une méthode ascendante.

# Analyse descendante

## Principe

construire l'arbre de dérivation du haut (la racine, c'est à dire l'axiome de départ) vers le bas (les feuilles, c'est à dire les unités lexicales).

On se retrouve avec




$w=\cancel{a}cb$

## Exemple 1

$$\left\{ \begin{array}{l} S \to aAb \\ A \to cd|c \end{array} \right. \quad \text{avec le mot } w = acb$$

En lisant le $c$, on ne sait pas s'il faut prendre la règle $A \to cd$ ou la règle $A \to c$. Pour le savoir, il faut lire aussi la lettre suivante ($b$). Ou alors, il faut se donner la possibilité de faire des **retour en arrière :** on essaye la 1ère règle ($A \to cd$), on aboutit à un échec, alors on retourne en arrière et on essaye la deuxième règle et là ça marche.

## Conclusion

ce qui serait pratique, ça serait d'avoir une table qui nous dit : quand je lis tel caractère et que j'en suis à dériver tel symbole non-terminal, alors j'applique telle règle et je ne me pose pas de questions. Ça existe, et ça s'appelle une **table d'analyse.**

# Analyse descendante

## Principe

construire l'arbre de dérivation du haut (la racine, c'est à dire l'axiome de départ) vers le bas (les feuilles, c'est à dire les unités lexicales).

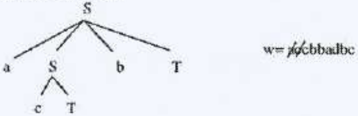
## Exemple 2

$$\begin{cases} S \to aSbT|cT|d \\ T \to aT|bS|c \end{cases}$$ avec le mot $w = accbbadbc$

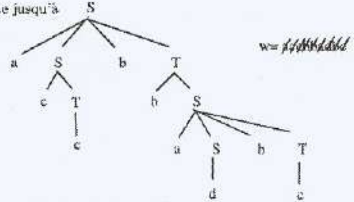
On part avec l'arbre contenant le seul sommet $S$

La lecture de la première lettre du mot ($a$) nous permet d'avancer la construction


puis la deuxième lettre nous amène à




et ainsi de suite jusqu'à




On a trouvé un arbre de dérivation, donc le mot appartient au langage.

# Table d'analyse LL1

## Définition

Pour construire une table d'analyse, on a besoin des ensembles PREMIER et SUIVANT

## Calcul de Premier

Pour toute chaîne $\alpha$ composée de symboles terminaux et non-terminaux, on cherche PREMIER($\alpha$) : l'ensemble de tous les **terminaux** (y compris $\varepsilon$) qui peuvent **commencer** une chaîne qui se dérive de $\alpha$
C'est à dire que l'on cherche toutes les lettres $a$ telles qu'il existe une dérivation $\alpha \xrightarrow{*} a\beta$ ($\beta$ étant une chaîne quelconque composée de symboles terminaux et non-terminaux). Cas particulier : $\varepsilon \in$ PREMIER($\alpha$) si et seulement si il existe une dérivation $\alpha \xrightarrow{*} \varepsilon$

## Exemple

$$\begin{cases} S \to Ba \\ B \to cP|bP|P|\varepsilon \\ P \to dS \end{cases}$$

$S \xrightarrow{*} a$, donc $a \in$ PREMIER($S$) $S \xrightarrow{*} cPa$, donc $c \in$ PREMIER($S$)
$S \xrightarrow{*} bPa$, donc $b \in$ PREMIER($S$) $S \xrightarrow{*} dSa$, donc $d \in$ PREMIER($S$)
Il n'y a pas de dérivation $S \xrightarrow{*} \varepsilon$
Donc PREMIER($S$)= $\{a, b, c, d\}$
$B \xrightarrow{*} dS$, donc $d \in$ PREMIER($B$)
$aB \xrightarrow{*} a\alpha$, donc PREMIER($aB$)= $\{a\}$
$BSb \xrightarrow{*} ab$, donc $a \in$ PREMIER($BS$)

# Table d'analyse LL1 – calcul de Premier –

## Algorithme de construction des ensembles Premier(X) pour $X \in ( V_T \cup V_N )$

1. Si $X$ est un non-terminal et $X \to Y_1 Y_2 \dots Y_n$ est une production de la grammaire (avec $Y_i$ symbole terminal ou non-terminal) alors
   - ajouter les éléments de PREMIER($Y_1$) sauf $\varepsilon$ dans PREMIER($X$)
   - s'il existe $j$ ($j \in \{2, \dots, n\}$) tel que pour tout $i = 1, \dots, j-1$ on a $\varepsilon \in$PREMIER($Y_i$), alors ajouter les éléments de PREMIER($Y_j$) sauf $\varepsilon$ dans PREMIER($X$)
   - si pour tout $i = 1, \dots, n$ $\varepsilon \in$PREMIER($Y_i$), alors ajouter $\varepsilon$ dans PREMIER($X$)
2. Si $X$ est un non-terminal et $X \to \varepsilon$ est une production, ajouter $\varepsilon$ dans PREMIER($X$)
3. Si $X$ est un terminal, PREMIER($X$) $= \{X\}$.

Recommencer jusqu'à ce qu'on n'ajoute rien de nouveau dans les ensembles PREMIER.

Algorithme de construction des ensembles Premier(α) pour $\alpha \in (V_T \cup V_N)^*$

```
On a α = Y1 Y2 ... Yn avec Yi ∈ VT ou Yi ∈ VN
si Y1 est un symbole terminal alors ajouter Y1 aux PREMIER(α)
sinon
    // Y1 est un symbole non terminal
    ajouter les éléments de PREMIER(Y1) sauf ε dans PREMIER(α)
    si ε ∈ PREMIER(Y1) alors
        // faire la même chose avec Y2
        si Y2 est un symbole terminal alors ajouter Y2 aux PREMIER(α)
        sinon
            // Y2 est un symbole non terminal
            ajouter les éléments de PREMIER(Y2) sauf ε dans PREMIER(α)
            si ε ∈ PREMIER(Y2) alors
                // faire la même chose avec Y3
                ⋮
                        // faire la même chose avec Yn
                        si Yn ∈ VT alors ajouter Yn aux PREMIER(α)
                        sinon
                            // Yn est un symbole non terminal
                            ajouter les PREMIER(Yn) sauf ε aux PREMIER(α)
                            si ε ∈ PREMIER(Yn) alors
                                ajouter ε aux PREMIER(α)
                            finsi
                        finsi
                ⋮
            finsi
        finsi
    finsi
finsi
```

# Table d'analyse LL1 – calcul de Premier –

Exemple 1

On considère la grammaire suivante, calculer les premiers de chaque symbole Non terminal

$$\begin{cases} E \to TE' \\ E' \to +TE' | -TE' | \varepsilon \\ T \to FT' \\ T' \to *FT' | /FT' | \varepsilon \\ F \to (E) | \text{nb} \end{cases}$$

Exemple 2

On considère la grammaire suivante, calculer les premiers de chaque symbole Non terminal

$$\begin{cases} S \to ABCe \\ A \to aA | \varepsilon \\ B \to bB | cB | \varepsilon \\ C \to de | da | dA \end{cases}$$

# Table d'analyse LL1 – calcul de Suivant –

## Définition

Pour tout non-terminal $A$, on cherche SUIVANT($A$) : l'ensemble de tous les symboles terminaux $a$ qui peuvent apparaître immédiatement à droite de $A$ dans une dérivation : $S \xrightarrow{*} \alpha A a \beta$

## Exemple

$$\begin{cases} S \to Sc|Ba \\ B \to BPa|bPb|P|\varepsilon \\ P \to dS \end{cases}$$

$a, b, c, d \in$SUIVANT($S$) car il y a les dérivations $S \xrightarrow{*} Sc$, $S \xrightarrow{*} dSa$, $S \xrightarrow{*} bdSba$ et $S \xrightarrow{*} dSdSaa$

$d \in$SUIVANT($B$) car $S \xrightarrow{*} BdSaa$

## Algorithme de construction des ensembles Suivant

1. Ajouter un marqueur de fin de chaîne (symbole \$ par exemple) à SUIVANT($S$) (où $S$ est l'axiome de départ de la grammaire)
2. Pour chaque production $A \to \alpha B \beta$ où $B$ est un non-terminal, alors ajouter le contenu de PREMIER($\beta$) à SUIVANT($B$), **sauf** $\varepsilon$
3. Pour chaque production $A \to \alpha B$, alors ajouter SUIVANT($A$) à SUIVANT($B$)
4. Pour chaque production $A \to \alpha B \beta$ avec $\varepsilon \in$PREMIER($\beta$), ajouter SUIVANT($A$) à SUIVANT($B$)

Recommencer à partir de l'étape 3 jusqu'à ce qu'on n'ajoute rien de nouveau dans les ensembles SUIVANT.

# Table d'analyse LL1 – calcul de Suivant –

**Exemple 1**

On considère la grammaire suivante, calculer les suivants de chaque symbole Non terminal

$$\begin{cases} E \rightarrow TE' \\ E' \rightarrow +TE' \mid -TE' \mid \varepsilon \\ T \rightarrow FT' \\ T' \rightarrow *FT' \mid /FT' \mid \varepsilon \\ F \rightarrow (E) \mid \mathrm{nb} \end{cases}$$

**Exemple 2**

On considère la grammaire suivante, calculer les suivants de chaque symbole Non terminal

$$\begin{cases} S \rightarrow aSb \mid cd \mid SAe \\ A \rightarrow aAdB \mid \varepsilon \\ B \rightarrow bb \end{cases}$$

# Table d'analyse – Construction de la table LL1 –

## Définition

Une table d'analyse est un tableau $M$ à deux dimensions qui indique pour chaque symbole non-terminal $A$ et chaque symbole terminal $a$ ou symbole $ la règle de production à appliquer.

## Algorithme de construction de la table d'analyse

- Pour chaque production $A \to \alpha$ faire
  1. pour tout $a \in$PREMIER$(\alpha)$ (et $a \neq \varepsilon$), rajouter la production $A \to \alpha$ dans la case $M[A, a]$
  2. si $\varepsilon \in$PREMIER$(\alpha)$, alors pour chaque $b \in$SUIVANT$(A)$ ajouter $A \to \alpha$ dans $M[A, b]$
- Chaque case $M[A, a]$ vide est une erreur de syntaxe

## Exemple : Compléter la table suivante

$$\begin{cases} E \to TE' \\ E' \to +TE' \mid -TE' \mid \varepsilon \\ T \to FT' \\ T' \to *FT' \mid /FT' \mid \varepsilon \\ F \to (E) \mid \text{nb} \end{cases}$$

| | nb | + | − | * | / | ( | ) | $ |
|---|---|---|---|---|---|---|---|---|
| $E$ | $E \to TE'$ | | | | | | | |
| $E'$ | | | | | | | | |
| $T$ | $T \to FT'$ | | | | | | | |
| $T'$ | | | | | | | | |
| $F$ | $F \to \text{nb}$ | | | | | | | |

# Analyseur syntaxique

## Principe

Maintenant qu'on a la table, comment l'utiliser pour déterminer si un mot $m$ donné est tel que $S \xrightarrow{*} m$ ?
On utilise une **pile.**

## Algorithme

données : mot $m$ terminé par \$, table d'analyse $M$
initialisation de la pile :

$\begin{array}{|c|} S \\ \$ \\ \hline \end{array}$ et un pointeur $ps$ sur la 1ère lettre de $m$

```
repeter
    Soit X le symbole en sommet de pile
    Soit a la lettre pointée par ps
    Si X est un non terminal alors
        Si M[X, a] = X → Y1 ... Yn alors
            enlever X de la pile
            mettre Yn puis Yn-1 puis ...puis Y1 dans la pile
            émettre en sortie la production X → Y1 ... Yn
        sinon         (case vide dans la table)
            ERREUR
        finsi
    Sinon
        Si X = $ alors
            Si a = $ alors ACCEPTER
            Sinon ERREUR
            finsi
        Sinon
            Si X = a alors
                enlever X de la pile
                avancer ps
            sinon
                ERREUR
            finsi
        finsi
    finsi
jusqu'à ERREUR ou ACCEPTER$
```

# Analyseur syntaxique

## Exemple 1

Sur la grammaire E,E',T,T' et F , on cherche à analyser l'instruction m = (7+3)5

## L'arbre syntaxique

| Pile | Entrée | Sortie |
|---|---|---|
| $E | (7+3)5$ | E → T E' |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

## Exemple

Sur la grammaire E,E',T,T' et F , on cherche à analyser l'instruction m=3+4*5

## L'arbre syntaxique

| Pile | Entrée | Sortie |
|---|---|---|
| $E | 3+4*5$ | E → T E' |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

1

| | nb | $+$ | $-$ | $*$ | $/$ | ( | ) | \$ |
|---|---|---|---|---|---|---|---|---|
| $E$ | $E \to TE'$ | | | | | $E \to TE'$ | | |
| $E'$ | | $E' \to +TE'$ | $E' \to -TE'$ | | | | $E' \to \varepsilon$ | $E' \to \varepsilon$ |
| $T$ | $T \to FT'$ | | | | | $T \to FT'$ | | |
| $T'$ | | $T' \to \varepsilon$ | $T' \to \varepsilon$ | $T' \to *FT'$ | $T' \to /FT'$ | | $T' \to \varepsilon$ | $T' \to \varepsilon$ |
| $F$ | $F \to \mathrm{nb}$ | | | | | $F \to (E)$ | | |

|  | nb | + | - | * | / | ( | ) | $ |
|---|---|---|---|---|---|---|---|---|
| $E$ | $E \to TE'$ | Err | Err | Err |  | 1 |  |  |
| $E'$ |  | 2 | 3 |  |  |  | 4 | 4 |
| $T$ | $T \to FT'$ |  |  |  |  | 5 |  |  |
| $T'$ |  | 8 | 8 | 6 | 7 |  |  |  |
| $F$ | $F \to nb$ |  |  |  |  | 9 |  |  |

3+4*5$

```
repeter
    Soit X le symbole en sommet de pile
    Soit a la lettre pointée par ps
1   Si X est un non terminal alors
1.1     Si M[X, a] = X → Y1 ... Yn alors
            enlever X de la pile
            mettre Yn puis Yn-1 puis ...puis Y1 dans la pile
            émettre en sortie la production X → Y1 ... Yn
1.2     sinon        (case vide dans la table)
            ERREUR
        finsi
2   Sinon
2.1     Si X = $ alors
a           Si a = $ alors ACCEPTER
b           Sinon ERREUR
            finsi
2.2     Sinon
c           Si X = a alors
                enlever X de la pile
                avancer ps
d           sinon
                ERREUR
            finsi
        finsi
    finsi
jusqu'à ERREUR ou ACCEPTER$
```

données : mot m terminé par $ table d'analyse M

initialisation de la pile :

| |
|---|
| $ |
| $ |

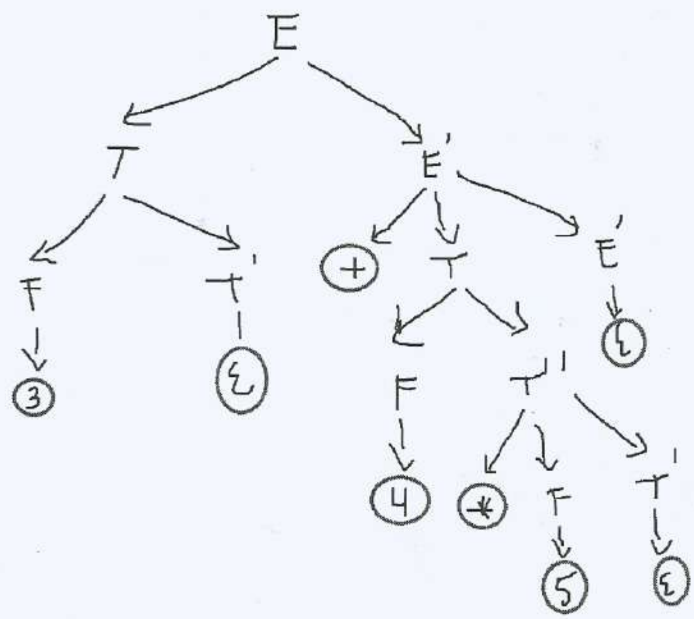
et un pointeur ps sur la 1ère lettre de m

| Pile | Entrée | Sortie |
|---|---|---|
| $ ~~E~~ | $\boxed{3+4*5\$}$ | E → TE' |
| $E' T | 3+4*5$ | T->FT' |
| $E' T' F | 3+4*5$ | F --> nb |
| $E' T' ~~nb~~ | ~~3~~+4*5$ | T' --> epsilon |
| $E' | +4*5$ | E' --> +TE' |
| $E'T ~~+~~ | ~~+~~4*5$ | T --> FT' |
| $E' T' F | 4*5$ | F -> nb |
| $E' T' ~~nb~~ | ~~4~~* 5 $ | T' -> * F T' |
| $E' T' F ~~*~~ | ~~*~~5$ | F->nb |
| $E' ~~T'~~ ~~nb~~ | ~~5~~$ | T' -> epsilon |
| $E' | $ | E' -> epsilon |
| $ | $ | ACCEPTER |
|  |  |  |
|  |  |  |
|  |  |  |

l'analyse de la phrase 3+4*5


| Pile | Entrée | Sortie |
|---|---|---|
| $ ~~E~~ | $\boxed{3+4*5\$}$ | E → TE' |
| $E' T | 3+4*5$ | T->FT' |
| $E' T' F | 3+4*5$ | F --> nb |
| $E' T' ~~nb~~ | ~~3~~+4*5$ | T' --> epsilon |
| $E' | +4*5$ | E' --> +TE' |
| $E'T ~~+~~ | ~~+~~4*5$ | T --> FT' |
| $E' T' F | 4*5$ | F -> nb |
| $E' T' ~~nb~~ | ~~4~~* 5 $ | T' -> * F T' |
| $E' T' F ~~*~~ | ~~*~~5$ | F->nb |
| $E' ~~T'~~ ~~nb~~ | ~~5~~$ | T' -> epsilon |
| $E' | $ | E' -> epsilon |
| $ | $ | ACCEPTER |


Arbre de dérivation de la phrase 3+4*5



3+4:

| | | nb | + | - | * | / | ( | ) | $ |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |
| $E$ | $E \to TE'$ | Err | Err | Err | | | 1 | | |
| $E'$ | | | 2 | 3 | | | | 4 | 4 |
| $T$ | $T \to FT'$ | | | | | | 5 | | |
| $T'$ | | | 8 | 8 | 6 | 7 | | 8 | 8 |
| $F$ | $F \to$ nb | | | | | | 9 | | |

3

**repeter**
- Soit $X$ le symbole en sommet de pile
- Soit $a$ la lettre pointée par $ps$
- 1 **Si** $X$ est un non terminal **alors**
  - 1.1 **Si** $M[X,a] = X \to Y_1 \ldots Y_n$ **alors**
    - enlever $X$ de la pile
    - mettre $Y_n$ puis $Y_{n-1}$ puis …puis $Y_1$ dans la pile
    - émettre en sortie la production $X \to Y_1 \ldots Y_n$
  - 1.2 **sinon** (case vide dans la table)
    - ERREUR
  - **finsi**
- 2 **Sinon**
  - 2.1 **Si** $X$ =$ **alors**
    - a **Si** $a$ = $ **alors** ACCEPTER
    - b **Sinon** ERREUR
    - **finsi**
  - 2.2 **Sinon**
    - c **Si** $X = a$ **alors**
      - enlever $X$ de la pile
      - avancer $ps$
    - d **sinon**
      - ERREUR
    - **finsi**
  - **finsi**
- **finsi**

**jusqu'à** ERREUR ou ACCEPTER$

données : mot $m$ terminé par $, table d'analyse $M$

initialisation de la pile :

| |
|---|
| $ |
| $ |

et un pointeur $ps$ sur la 1ère lettre de $m$

| Pile | Entrée | Sortie |
|---|---|---|
| $E | (7+3)5$ | E → T E' |
| $E' T | (7+3)5$ | T -> FT' |
| $E' T' F | (7+3)5$ | F -> ( E ) |
| $E' T' ) E ~~(~~ | ~~(~~7+3)5$ | E -> TE' |
| $E'T')E' T | 7+3)5$ | T -> F T' |
| $E' T' )E'T' F | 7+3)5$ | F -> nb |
| $E' T' )E'T' ~~nb~~ | ~~7~~+3)5$ | T' -> epsilon |
| $E' T' )E' | +3)5$ | E' -> + TE' |
| $E' T' ) E' T | 3)5$ | T-> FT' |
| $E' T' ) E' T' F | 3)5$ | F-> nb |
| $E' T' ~~)~~ ~~E'~~ ~~T'~~ | ~~)~~5$ | T' -> epsilon |
| | | Erreur car la case |
| | | M[T',nb] est vide |
| | | |
| | | |