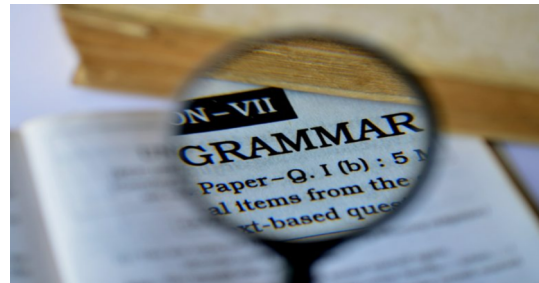


ANALYSE SYNTAXIQUE - Partie 1 : Principes de base

Chapitre 2 : Analyse syntaxique
SYNT



Chebieb - ESI

Table des matières



I - A- Concepts de base	3
1. A- L'analyse syntaxique ? C'est quoi, pourquoi et comment ?	4
1.1. Les activités du processus de compilation	4
1.2. L'étape de l'analyse syntaxique	4
1.3. Grammaire	6
1.4. Grammaire algébrique	8
1.5. Reconnaissance des mots par une grammaire algébrique	9
1.6. 1 - Définition formelle d'un automate à pile et grammaire algébrique	10

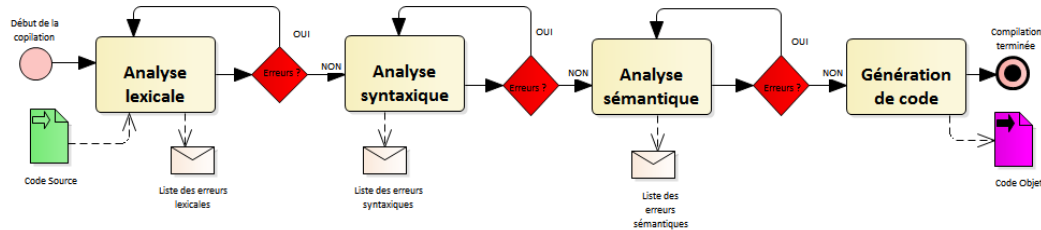
A- Concepts de base

I

1. A- L'analyse syntaxique ? C'est quoi, pourquoi et comment ?

1.1. Les activités du processus de compilation

L'enchaînement des étapes de la compilation (traduction)



Étapes de Compilation

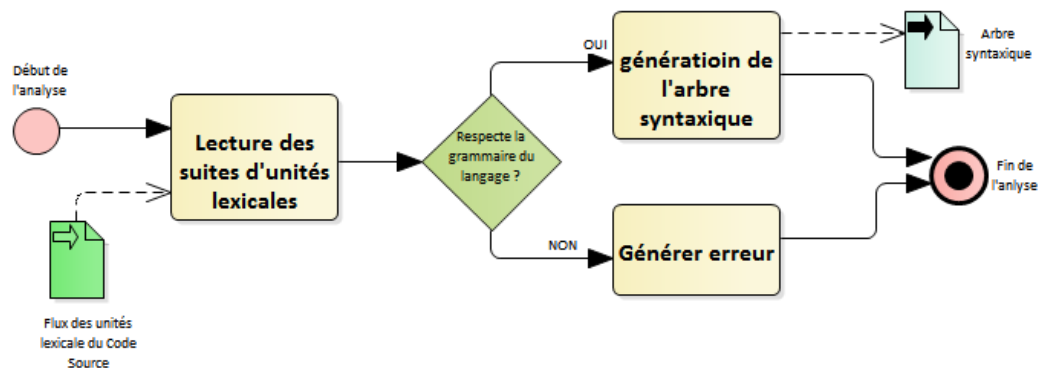
1.2. L'étape de l'analyse syntaxique

En quoi consiste cette étape ?

Vérifier si les unités lexicales obtenues du code source forme bien les structures du langage utilisé. C'est à dire :

- Est ce que **la grammaire du langage** utilisé est respectée par le flux des lexème du code source ?
 - SI oui une structure de donnée dite "**ARBRE SYNTAXIQUE**" est générée
 - Sinon une erreur est signalée.

La figure ci-après montre les Phases de l'analyse syntaxique.



Phase analyse syntaxique

☞ **Exemple : Pour le flux : $A + B * C$**

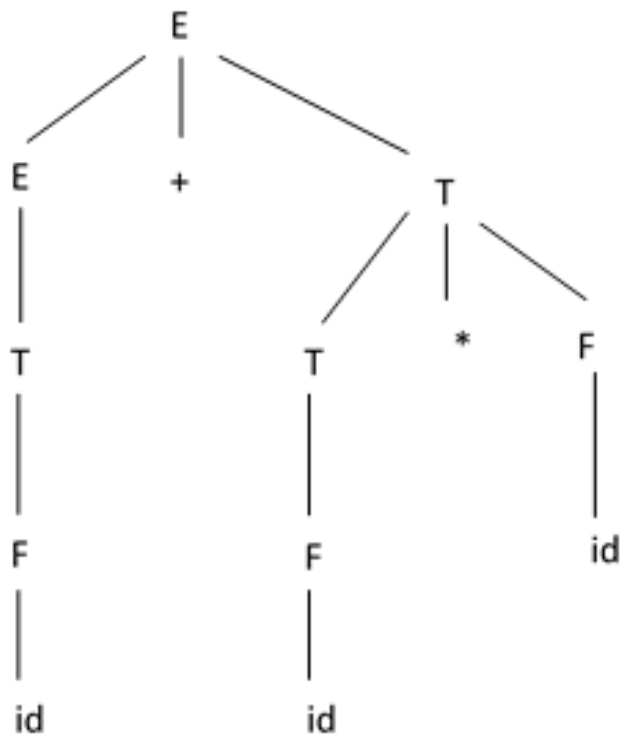
Soit la grammaire du langage :

$E \rightarrow E + T \mid T$

$T \rightarrow T * F \mid F$

$F \rightarrow (E) \mid id$

L'analyse lexicale retournera par exemple : $id1 + id2 * id3$ et l'analyse syntaxique retournera l'arbre syntaxique de cette chaîne.



Arbre syntaxique

👉 Exemple : La définition textuelle de la syntaxe de la fonction SUM de MS. EXCEL

Syntax:

SUM(number1,[number2],...)

Argument name	Description
number1 Required	The first number you want to add. The number can be like 4, a cell reference like B6, or a cell range like B2:B8.
number2-255 Optional	This is the second number you want to add. You can specify up to 255 numbers in this way.

La grammaire permettant de décrire la syntaxe de cette fonction

$E \rightarrow \text{SUM} (\text{PARAM})$


$\text{PARAM} \rightarrow \text{PARAM} ; \text{Number} \mid \text{Number}$

🔑 Définition : L'analyse syntaxique : c'est quoi au juste ?

- Opération qui consiste à reconnaître les mots d'un langage décrit par une grammaire algébrique.

- Cette reconnaissance se base sur la notion de **dérivation** en utilisant les règles de productions de la grammaire.

1.3. Grammaire

 *Définition : Rappelons que :*

Une grammaire est un quadruplet $G = \langle N, T, P, S \rangle$ où :

T est l'ensemble de symboles dits **TERMINAUX**

N est l'ensemble de symboles dits **NON TERMINAUX**, $N \cap T = \emptyset$

$S \in T$ est **L'AXIOME** de G ,

P est l'ensemble des règles de production.

 *Complément : Règle de production*

Une règle de production est l'opération qui permet la génération des mots par réécriture (substitution).
l'ensemble P peut être défini comme suit :

$$P \subset ((T \cup N)^* \cdot T \cdot (T \cup N)^*) \times (T \cup N)^*$$

$$p \in P, p = (\alpha, \beta) \text{ avec :}$$

$$\alpha \in (T \cup N)^* \cdot T (T \cup N)^*$$

$$\beta \in (T \cup N)^*$$

α est le membre Gauche de la production p

β est le membre Droit de la production p

On dit :

α est réécrit en β ou β est généré par α

La production p est noté comme suit :

$$p : \alpha \longrightarrow \beta$$


Il existe une autre forme de notation dite "**forme de Backus-Naur**" (**BNF de l'anglais Backus-Naur Form**) utilisée souvent dans la description des grammaires des langages de programmation.

Dans les productions de la notation BNF :

Les symboles non terminaux sont mis en "<" et ">"

Les symboles terminaux sont mis entre " "

Le MGP (membre gauche de production) est séparé du MDP (membre droit de production) par " : := "

 *Exemple : Soit G une grammaire $G = \langle N, T, P, S \rangle$*

$$G = \langle \{a, b, \epsilon\}, \{S, A\}, S, \{S \longrightarrow abSA, B \longrightarrow Aa, A \longrightarrow Ba\} \rangle$$

Dans G :

- Les terminaux sont $T = \{a, b, \epsilon\}$,
- Les non terminaux sont $N = \{S, A\}$,
- L'axiome est S
- Les règles de production sont :

$$P = \{ \begin{array}{l} S \longrightarrow abSA \\ B \longrightarrow Aa \\ A \longrightarrow Ba \end{array} \}$$

- La notation BNF des règles de productions est :

$$\langle S \rangle ::= \text{"a"} \text{"b"} \langle S \rangle \langle A \rangle$$

$$\langle B \rangle ::= \langle A \rangle \text{"a"}$$

$$\langle A \rangle ::= \langle B \rangle \text{"a"}$$

Complément : La dérivation et la série de dérivations

Une dérivation est l'application d'une seule règle de production pour passer d'une chaîne de symboles de la grammaire $(T \cup N)^*$ à une autre chaîne de symboles.

La dérivation est notée :

$$\omega_1 \Rightarrow \omega_2$$

La série de dérivations est l'application de plusieurs règles consécutivement pour récrire une chaîne en une autre. Elle est notée

$$\omega_1 \Rightarrow^* \omega_2$$

Complément : Mot dérivé par une grammaire

Soit une grammaire $G = \langle N, T, P, S \rangle$

On dit qu'un mot $\omega \in T^*$ est généré par G s'il existe une série de dérivation de ω à partir de l'axiome S :

$$\exists p_1 p_2 \cdots p_n \in P \text{ tq :}$$

$$S \Rightarrow^{P_1} \dots \Rightarrow^{P_2} \dots \Rightarrow^{P_n} \dots \Rightarrow \omega$$

Une dérivation pour ce mot à partir de l'axiome est notée : $S \Rightarrow^* \omega$.

Complément : Dérivation la plus à gauche

On parle de dérivation **la plus à gauche** (**LEFTMOST DERIVATION**) lorsqu'à chaque étape d'une série de dérivations on applique une production au non-terminal le plus à gauche.

Complément : Dérivation la plus à droite

On parle de dérivation **la plus à droite** (**RIGHTMOST DERIVATION**) lorsqu'à chaque étape d'une série de dérivations on applique une production au non-terminal le plus à droite.

Exemple

Soit la chaîne $aabb$ à dériver de l'axiome S d'une grammaire dont les productions

$$P_1 : S \longrightarrow aXS ; P_2 : S \longrightarrow aSX ; P_3 : S \longrightarrow ab ; P_4 : X \longrightarrow bX ; P_5 : X \longrightarrow b$$

- La série de dérivation la plus à gauche est :

$$S \Rightarrow^{P_2} aSX \Rightarrow^{P_3} aabX \Rightarrow^{P_4} aabb$$

- La série de dérivation la plus à droite est :

$$S \Rightarrow^{P_2} aSX \Rightarrow^{P_5} aSb \Rightarrow^{P_3} aabb$$

Complément : Langage généré par une grammaire

Soit une grammaire $G = \langle N, T, P, S \rangle$

Le langage L généré ou dérivé par une grammaire $G = \langle X, V, P, S \rangle$ est défini comme suit :

$$L(G) = \{ \omega \in T^* \mid \exists p_i \in P \text{ tq } S \Rightarrow^* \omega \}$$

1.4. Grammaire algébrique

Définition : Grammaires algébriques

Des grammaires classées par Chomsky dans le type 2. Elles ont une restriction au niveau des membres droits et gauches des productions.

Elles sont dites aussi "**Grammaires hors contexte ou à contexte libre**"

Une grammaire algébrique est un quadruplet $G = \langle N, T, P, S \rangle$ où :

T est l'ensemble de symboles dits **TERMINAUX**

N est l'ensemble de symboles dits **NON TERMINAUX**, $N \cap T = \emptyset$

$S \in T$ est **L'AXIOME** de G ,

P est l'ensemble des règles de production tel que

$$\forall p \in P : p = A \longrightarrow \omega \text{ avec :}$$

$$A \in N \wedge \omega \in (T \cup N)^*$$

Exemple : Grammaire des expressions arithmétiques

Soit $G = \langle \{ E \}, \{ +, -, *, /, (,), id \}, P, E \rangle$ avec P :

$$E \longrightarrow E + E$$

$$E \longrightarrow E - E$$

$$E \longrightarrow E * E$$

$$E \longrightarrow E / E$$

$$E \longrightarrow (E)$$

$$E \longrightarrow id$$

Les productions ayant plusieurs MDPs peuvent s'écrire comme suit :

$$E \longrightarrow E + E \mid E - E \mid E * E \mid E / E \mid (E) \mid id$$

Fondamental : Les propriétés des grammaires algébriques utilisables :

Un analyseur syntaxique déterministe se base sur une grammaire propre. Une grammaire dite propre possède les propriétés suivantes:

- (1) Pas de symboles inutiles : Chaque non terminal et chaque terminal de la grammaire apparaît au moins dans la dérivation d'un mot du langage
- (2) Pas de productions unitaires: absence de productions de la forme $A \longrightarrow B$ avec $(A, B) \in N^2$
- (3) Absence d' ϵ -productions : Si $\epsilon \notin L$, alors éliminer les productions de la forme $A \longrightarrow \epsilon$ avec $A \in N$

☞ *Exemple : G1 : Grammaire non propre*

$G_1 = \langle \{a, b, \epsilon\}, \{S, A\}, S, P \rangle$ Tel que P :

$$S \rightarrow abSA$$

$$B \rightarrow Aa | A$$

$$A \rightarrow Ba | \epsilon$$

$$C \rightarrow aBa$$

☞ *Exemple : G2 : Grammaire propre*

$G_2 = \langle \{a, b, \epsilon\}, \{S, A\}, S, P \rangle$ Tel que P :

$$S \rightarrow abA | baB | \epsilon$$

$$B \rightarrow Aa | a$$

$$A \rightarrow Ba | b$$

1.5. Reconnaissance des mots par une grammaire algébrique

🔑 *Définition : Ambiguïté des grammaires algébriques*

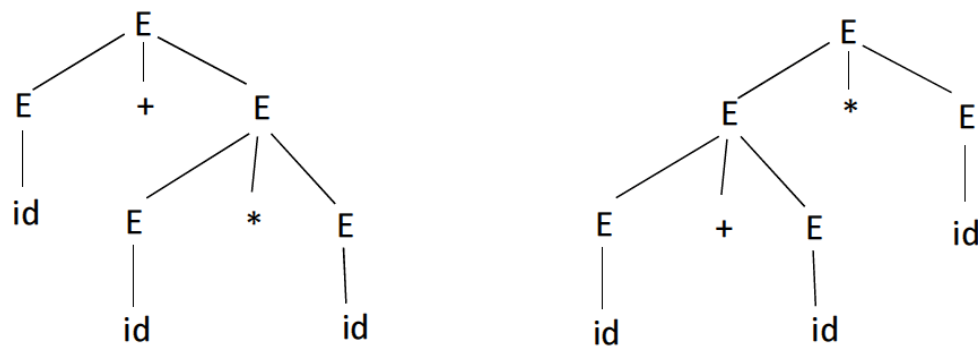
Une grammaire à contexte libre (type 2) est ambiguë si elle génère (dérive) un langage L ayant au moins un mot (chaîne de symboles terminaux) possédant deux ou plusieurs arbres syntaxiques.

☞ *Exemple : G est une grammaire ambiguë et la dérivation du mot $id + id * id$ donne deux arbres syntaxique*

Soit $G = \langle \{E\}, \{+, -, *, /, (,), id\}, P, E \rangle$ avec P :

$$E \rightarrow E + E | E - E | E * E | E / E | (E) | id$$

Deux arbre syntaxique (voir figure suivante) pour le mot : $id + id * id$



⚠ *Attention : Prouver la non ambiguïté ou transformer la grammaire*

- L'existence de deux suites de dérivations pour un mot $\omega \in T^*$ n'induit pas l'existence de deux arbres syntaxiques.
- On dispose en général des conditions nécessaires mais pas suffisantes pour prouver la non ambiguïté d'une grammaire (problème indécidable)

Nous pouvons donner une grammaire non ambiguë si on connaît le langage à générer : le langage des expressions arithmétiques en est un cas.

Attention : Ambiguïté et non déterminisme

Une grammaire ambiguë induit une analyse syntaxique non déterministe

Fondamental : PRINCIPE DE FONCTIONNEMENT DE L'ANALYSEUR SYNTAXIQUE

L'analyseur syntaxique doit être déterministe :

- Soit le flux d'entrée (le programme) est correct
- Soit il y a une ou plusieurs erreurs

Complément : Méthodes d'analyse syntaxique

Il existe deux démarches pour l'analyse syntaxique d'un flux :

1. Analyse descendante
2. Analyse ascendante

Méthode : PRINCIPE DE L'ANALYSE DESCENDANTE

- Retrouver le flux entrée en effectuant des dérivations successives en commençant par l'axiome de la grammaire.
- Cela revient à produire l'arbre syntaxique en partant du nœud racine (qui représente l'axiome) et descendre vers les feuilles (les symboles terminaux)

Méthode : PRINCIPE DE L'ANALYSE ASCENDANTE

- Retrouver l'axiome de la grammaire en effectuant des dérivations successives inverses (dites réductions)
- Cela revient à produire l'arbre syntaxique en partant des feuilles (les symboles terminaux) pour remonter au nœud racine (qui représente l'axiome)

1.6. 1 - Définition formelle d'un automate à pile et grammaire algébrique

Définition : Automate à pile et grammaire

Une grammaire algébrique est un quadruplet $G = \langle N, T, P, S \rangle$ où :

T est l'ensemble de symboles dits **TERMINAUX**

N est l'ensemble de symboles dits **NON TERMINAUX**, $N \cap T = \emptyset$

$S \in T$ est **L'AXIOME** de G ,

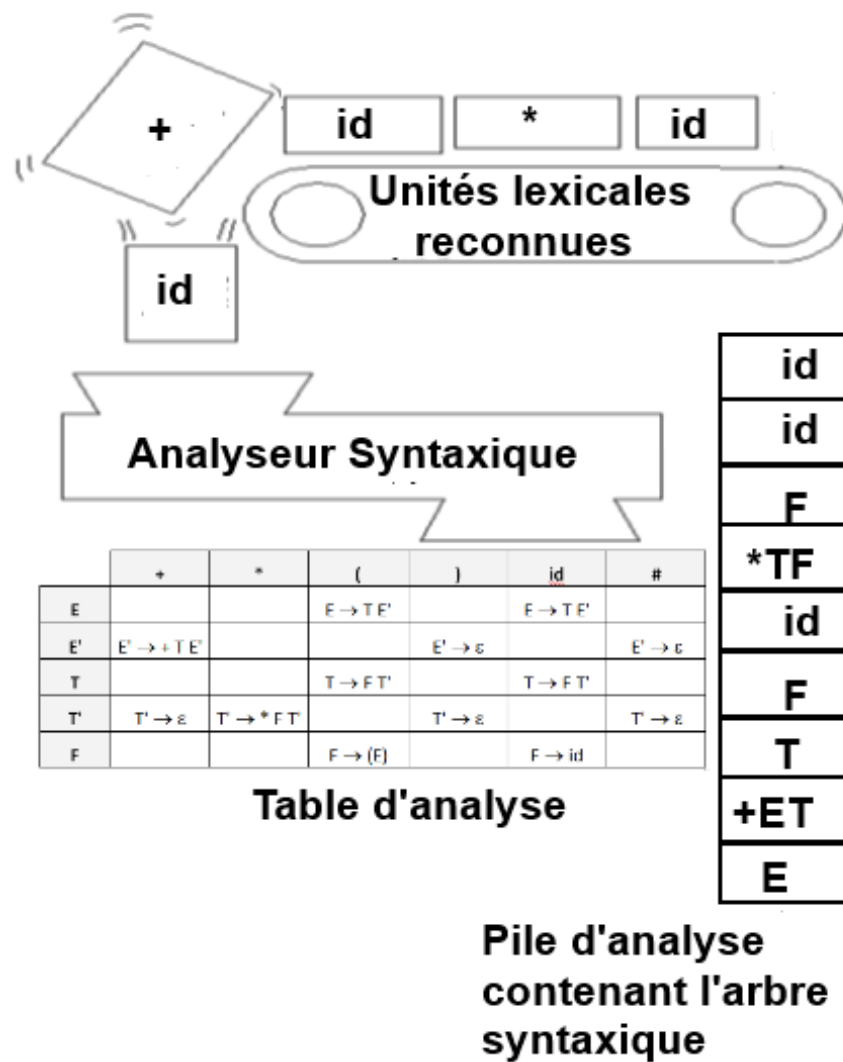
P est l'ensemble des règles de production tel que

$\forall p \in P : p = A \rightarrow \omega$ avec :

$A \in N \wedge \omega \in (T \cup N)^*$

Un automate à pile est un 7-uplets $\langle X, Y, S, S_0, F, A, \# \rangle$:

- X : un alphabet d'entrée qui correspond à l'ensemble des terminaux (unités lexicales)
- Y : un alphabet auxiliaire (de la pile) qui correspond à l'ensemble des non terminaux et terminaux ($T \cup N$)
- S : ensemble des états de l'automate qui correspond aux étapes de dérivations
- F : ensemble des états finaux qui correspond à l'acceptation et c'est un seul état unique
- A : ensembles des actions possibles de l'automate suite à la lecture d'une unité lexicale et qui correspond à :
 - Avancer : lire la prochaine unité lexicale du flux d'entrée.
 - Appliquer une règle de production (dérivé) : Empiler / Dépiler un ou plusieurs symboles $mp \in (N \cup T)$ membres de la production (mp).
- $\#$: symbole représentant à la fois la fin du flux d'entrée et la pile vide



* *

*

Résumons la partie :

Dans cette partie 1 du chapitre Analyse Syntaxique, nous avons abordé les principes de bases de l'analyse syntaxique permettant de comprendre l'implémenter des analyseurs syntaxiques :

L'analyse Syntaxique est donc l'étape de la compilation qui permet de vérifier la conformité du code source écrit par le programmeur à la syntaxe du langage utilisé. Ce langage est décrit par une grammaire de type 2. Les mots du langages peuvent donc être vérifiés et/ou reconnus par un automate à pile qui permet de recalculer l'arbre syntaxique du mot en appliquant les règles de production de la grammaire qui décrit le langage. Ce dernier est composé des éléments suivants:

1. Une tête de lecture du flot d'entrée : suite d'unités lexicales renvoyées par l'analyseur lexical à la demande de l'analyseur syntaxique
2. Une pile d'analyse : la mémoire de l'automate à pile où sont empilés les symboles des différentes règles de production de la grammaire qui servent à construire l'arbre syntaxique du flot d'entrée.
3. Une table d'analyse : la matrice qui permet à l'automate de choisir l'action à entreprendre, à savoir avancer dans le flot en demandant une autre unité lexicale ou/et agir sur la pile (dépiler et/ou empiler).

Si l'automate arrive à la fin du flot d'entrée avec une pile vide alors le flot d'unités lexicales est reconnu conforme à la syntaxe du langage et l'historique de la pile révèle son arbre syntaxique. Dans le cas contraire, si l'automate n'arrive pas à la fin du flot et ne trouve pas d'action à entreprendre dans sa table alors le flot d'unités lexicales est signalé non conforme à la syntaxe du langage et l'historique de la pile révèle l'endroit de l'erreur syntaxique.

La conception de l'analyseur revient à concevoir sa table d'analyse. Il existe deux méthodes majeures d'analyse qui sont :

- **L'analyse descendante** : Consiste à démarrer de l'axiome de la grammaire et essayer de retrouver le mot en effectuant des dérivations en appliquant les règles de production de la grammaire. Cette méthode possède deux techniques d'implémentation à savoir l'analyse LL (**Left to right scanning using the Leftmost derivation**) par table prédictive et l'analyse LL par descente récursive.
- **L'analyse ascendante** : Consiste à démarrer du mot et essayer de retrouver l'axiome de la grammaire en effectuant des dérivations inverses dites réductions en appliquant les règles de production de la grammaire. Cette méthode possède deux techniques d'implémentation à savoir l'analyse par précédences et l'analyse par contextes dite LR (**left to right scanning using Rightmost derivation in reverse -reductions-**). Cette dernière est la plus utilisée par les produits du marché.