Analyse Syntaxique

Analyseur Syntaxique -Parser-

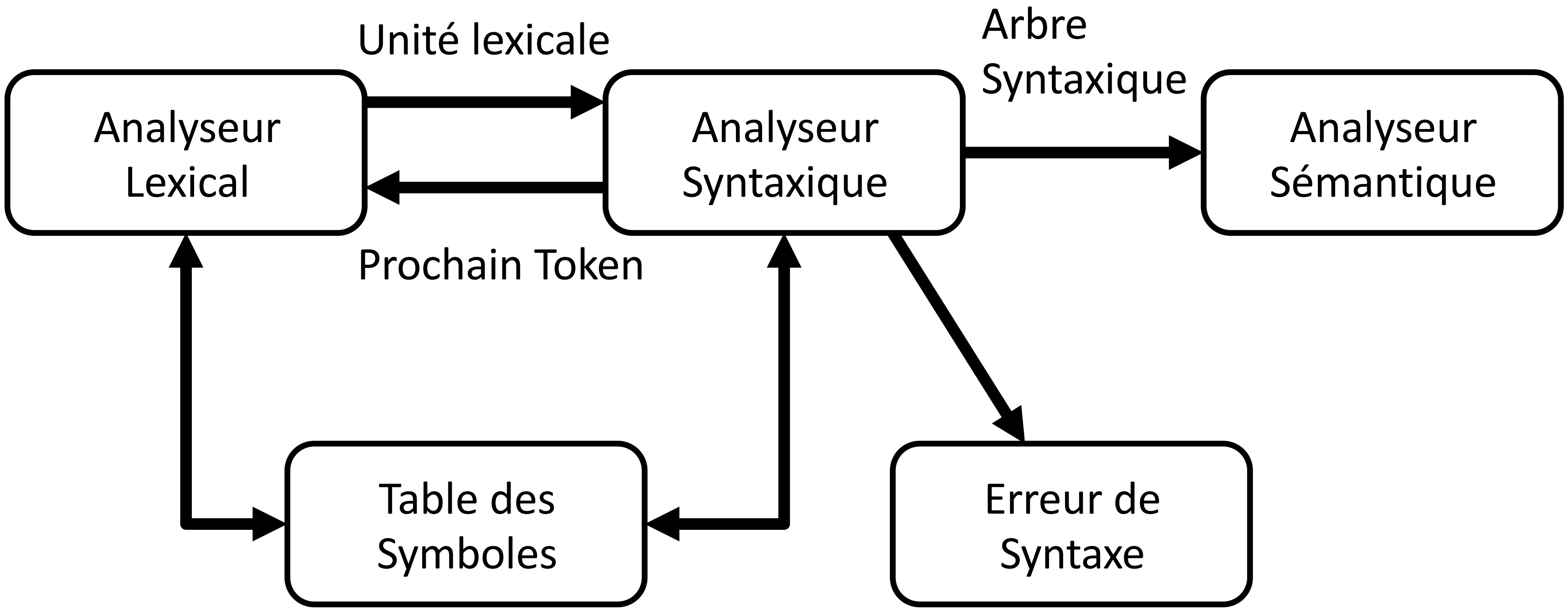
- L'analyseur syntaxique vérifie que l'ordre des tokens correspond à l'ordre définit pour le langage. On dit que l'on vérifie la syntaxe du langage à partir de la définition de sa grammaire.

- L'analyse syntaxique produit une représentation sous forme d'arbre de la suite des tokens obtenus lors de l'analyse lexicale

Analyseur Syntaxique

- Pour effectuer efficacement une analyse syntaxique, le compilateur nécessite :
  - Une définition formelle du langage source,
  - Une fonction indicatrice de l'appartenance d'un programme au langage source,
  - Un plan de gestion des entrées illégales.

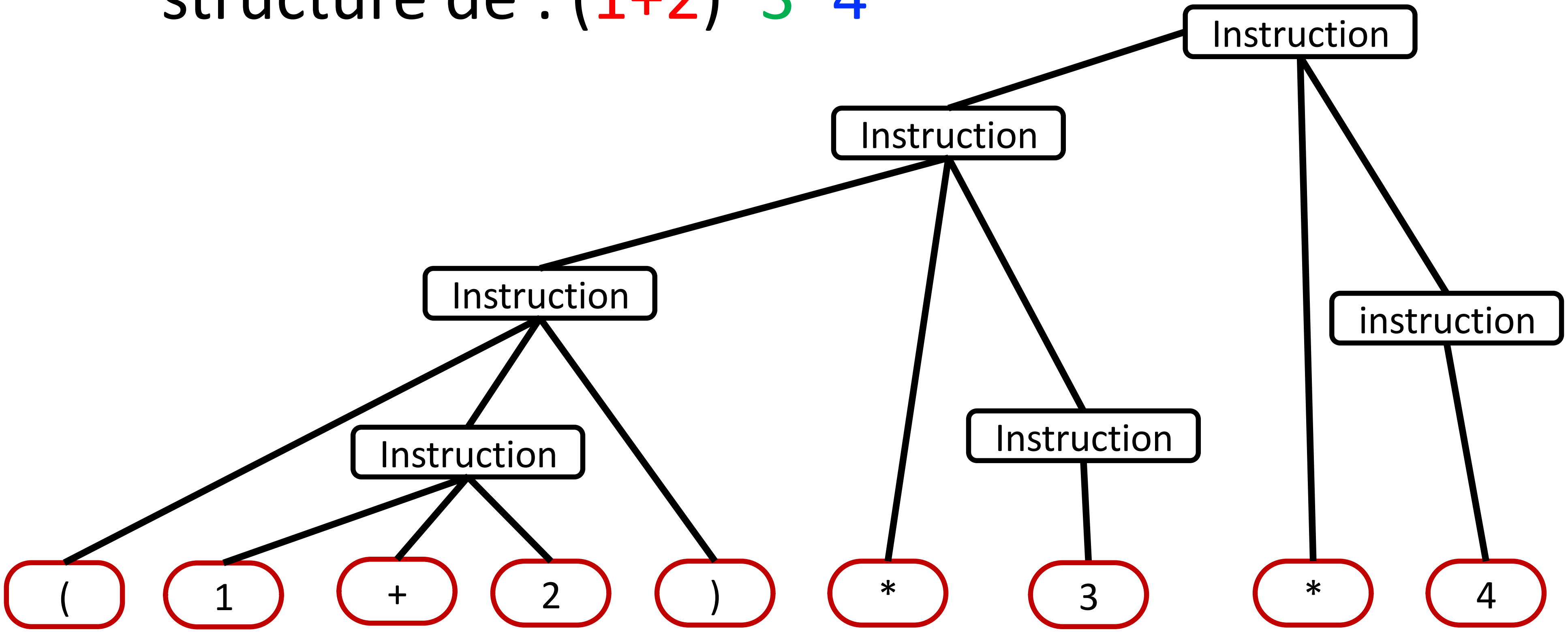
Analyseur Syntaxique



Analyseur Syntaxique

Arbre syntaxique - Exemple

- Arbre syntaxique suivant représente la structure de : (1+2)*3*4



Syntaxe et Grammaire

- La syntaxe est traditionnellement exprimée à l'aide d'une ***grammaire***

- Une ***grammaire*** **G** est une ***collection de règles de réécriture*** qui définissent mathématiquement quand une suite de symbole d'un certain alphabet constitue un mot d'un langage

- *L'ensemble des mots* pouvant être dérivées de ***G*** est appelé le *langage défini par* **G**, noté ***L(G).***

Grammaire

- Une **grammaire** est formellement définie par :
  - Un ensemble de symboles **terminaux** (token) : Les symboles élémentaires du langage.
  - Un ensemble de symboles **non-terminaux**
  - Un ensemble de **règles syntaxiques** (ou de productions). Tête → terminaux et/ou non-terminaux
  - Un **axiome** (symbole initial, un non-terminal).
- Une **grammaire** définit un langage formé par l'ensemble des séquences finies de symboles **terminaux** qui peuvent être dérivées de l'**axiome** par des applications successives des productions.

Grammaire
Exemple

- **<u>Exemple d’une structure du if en langage C :</u>**

- <structure_if> ::= if «(» <condition> «)» «{» <instruction> «}»

  - **<structure_if>, <condition>**, **<instruction>** : **non-terminaux**.
  - **::=** : est un **méta-symbole** (symbole de la grammaire) signifiant «est défini par».
  - **if**, **«(»** **,** **«)»** **,** **«{»** et **«}»** : **des terminaux**. Lorsque les **terminaux** ne font qu’un caractère, ou qu’ils contiennent des caractères non alphanumériques, ou qu’ils peuvent être confondus avec des méta-symboles, ils sont écrits entre guillemets.

Grammaire
Exemple

```
G1 = (  {« a », « b », « c », « d »,       /* terminaux */
        « + », « - », « * », « / »,
        « ( » , « ) », « ^ »},
        {<expression>, <facteur>  },/* non-terminaux */
        {<expression> ::= <facteur>,         /* productions */
         <expression> ::= <expression> « + » <expression>,
         <expression> ::= <expression> « - » <expression>,
         <expression> ::= <expression> « * » <expression>,
         <expression> ::= <expression> « / » <expression>,
         <facteur> ::= « a »,
         <facteur> ::= « b »,
         <facteur> ::= « c »,
         <facteur> ::= « d »,
         <facteur> ::= « ( » expression « ) »,
         <facteur> ::= <facteur> « ^ » <facteur> },
      <expression>                                /* axiome */)
```

Grammaire

Résumé

- Une grammaire dérive des chaînes en commençant par l'axiome et en remplaçant de façon répétée les non-terminaux décrits par les productions de la grammaire.
- Les chaînes de terminaux dérivables à partir de l'axiome forment le langage défini par la grammaire.

Grammaire hors-Contexte (GHC)

- Une Grammaire hors-contexte (GHC) ***G** est un 4-uplet* **$G = \langle T, NT, S, P \rangle$** *où* :
  - ***T*** est l'ensemble des symboles *terminaux* (concrets) ou lettres de l'alphabet
  - ***NT*** est l'ensemble des symboles non-*terminaux* (abstraits)
  - **$S \in NT$**, *appelé symbole initial (Start ou axiome)*
    - Toute dérivation d'un mot de **L(G)** débute par **S**
    - À partir de ***S***, on dérive l'ensemble des mots de ***L*(*G*)**
  - ***P*** est l'ensemble des règles de réécriture ou de production. Formellement, une règle de ***P*** est sous la forme : $NT \rightarrow (T \cup NT)^*$

Langage dérivé

- Soit $G = <T, NT, S, P>$ une grammaire. On appelle **langage engendré** par ***G*** l'ensemble $L(G) = \{w \in T^* \,/\, S \Rightarrow_{r \in P}^{+} w\}$

  - *où $\Rightarrow_{r \in P}$, est appelée dérivation et dénote l'application d'une règle de production **r** de **P***
  - ***et** $\Rightarrow_{r \in P}^{+}$ dénote la répétition de règles $\Rightarrow_{r \in P}$*

Grammaire hors-Contexte (GHC)
Exemple

- *Soit* G=<T,NT,S,P> *avec:*
  - ✓ T = {a,b}
  - ✓ NT={S,A,B}
  - ✓ S : *l'axiome.*
  - ✓ P ={**S→AB|aS|A**, **A→Ab|ε**, **B→AS**}

- *Pour cette grammaire, les mots* **AB**, **aaS** *et* **ε** *sont des formes sur* G.

Grammaire hors-Contexte (GHC)

Exemple

- Une grammaire hors-contexte qui engendre les palindromes sur {*a*, *b*} :

  - $S \rightarrow aSa$
  - $\quad | bSb$
  - $\quad | \varepsilon$

Grammaire hors-Contexte (GHC)
Ecriture

- G = <T={0,1}, NT={S}, S, P={r1,..,r5} >

- **1ère écriture des règles**
  - r1: $S \rightarrow \varepsilon$
  - r2: $S \rightarrow 0$
  - r3: $S \rightarrow 1$
  - r4: $S \rightarrow 0\ S\ 0$
  - r5: $S \rightarrow 1\ S\ 1$

- **2ème écriture des règles**
  - r1: $S \rightarrow \varepsilon$
  - r2: | 0
  - r3: | 1
  - r4: | 0 S 0
  - r5: | 1 S 1

- **3ème écriture des règles (BNF -Backus-Naur Form)**
  - r1: **<S> ::=** (ε n'est pas représentable : c'à dire : un vide = ε)
  - r2: | 0
  - r3: | 1
  - r4: | 0 **<S>** 0
  - r5: | 1 **<S>** 1

BNF-Backus-Naur Form
Exemple

Grammaire BNF pour la construction d'un langage naturel simple

```
<Phrase> ::= <sujet> <verbe> <complément>
<sujet> ::= <article> <adjectif> <nom> |
            <article> <nom> <adjectif> |
            <article> <nom>
<article>  ::= «le» | «la» | «l'» |
               «les» | «un» | «une» | «des»
<adjectif> ::= «grand» | «petit» | <couleur>
<couleur> ::= «bleu» | «vert» | «rouge»
<verbe> ::= (etc)
```

Grammaire hors-Contexte (GHC)

Exercice 1

- Quel est le langage ***L(G)*** décrit par la grammaire hors-contexte suivante ?

- ***G = < T ={a}, NT ={S}, S, P = {r1, r2} >***
  - **r1: $S \rightarrow aS$**
  - **r2: $\mid a$**

Grammaire hors-Contexte (GHC)
Solution 1

- **G:**
  - **r1:** $\mathbf{S \rightarrow aS}$
  - **r2:** $\mathbf{\mid\ a}$

- Raisonnons par induction sur la taille des mots w de L(G)
- $|w| = 1 : S \Rightarrow_{r2} a$
- $|w| = 2 : S \Rightarrow_{r1} aS \Rightarrow_{r2} aa = a^2$
- $|w| = 3 : S \Rightarrow_{r1} aS \Rightarrow_{r1} aaS \Rightarrow_{r2} aaa = a^3$
- $|w| = 4 : S \Rightarrow_{r1} aS \Rightarrow_{r1} aaS \Rightarrow_{r1} aaaS \Rightarrow_{r2} aaaa = a^4$
- $|w| = i : S \Rightarrow_{r1} aS \Rightarrow_{r1} aaS \Rightarrow_{r1} aaS \Rightarrow_{r1} \ldots \Rightarrow_{r1} aaa..aaS \Rightarrow_{r2} aaa..aaa = a^i$

- $L(G) = \cup_{1\leq i} \{w \in T^* \,/\, w = a^i \} = \{ w \in T^* \,/\, w = a+\}$
- ***L(G)*** est le langage de mots formés d'une suite non vide de la lettre 'a'

Grammaire hors-Contexte (GHC)

Exercice 2

On considère la grammaire G = <T,NT,S,P> où

$$T = \{b,c\}$$

$$NT = \{S\}$$

$$P = \{\ S \rightarrow bS \mid cc\}$$

Déterminer L(G).

Grammaire hors-Contexte (GHC)
Solution 2

En effet, partant de l'axiome S, toute dérivation commencera nécessairement par appliquer 0, 1 ou plusieurs fois la première règle puis se terminera en appliquant la deuxième règle.

On représentera cela en écrivant le schéma de dérivation suivant :

$$S \Longrightarrow_{r1}^{*} b^n S \Longrightarrow_{r2} b^n cc \quad n \in \mathbb{N}$$

Alors L(G) = $\{b^n cc \ / \ n \in \mathbb{N}\}$

Grammaire hors-Contexte (GHC)

Exercice 3

On considère la grammaire G = <T,NT,S,P> où

$T = \{ a,b,0\}$

$NT = \{ S,U\}$

$P = \{ S \rightarrow aSa \mid bSb \mid U$

$\quad U \rightarrow 0U \mid \varepsilon \}$

Déterminer L(G).

Grammaire hors-Contexte (GHC)

Solution 3

Prenons les cas suivants

$$S \Longrightarrow_{r1} aSa \Longrightarrow_{r1} aUa \Longrightarrow_{r2}^{n} a0^{n}a$$

$$S \Longrightarrow_{r1} aSa \Longrightarrow_{r1} abSba \Longrightarrow_{r2}^{n} ab0^{n}ba$$

ab0^n**ba**

On définit inverse(u) est le mot inverse de u, tel que v=inverse(u)

Alors L(G) = $\{u0^{n}v / u \in \{a, b\}^{*}, v = inverse(u), n \in \mathbb{N}\}$

Grammaire hors-Contexte (GHC)
Exercice 4

On considère la grammaire G = <T,NT,Ph,P> où

T = { un , une , le , la , enfant , garçon , fille , cerise , haricot , cueille , mange }
NT = {Ph,Gn,Gv,Df,Dm,Nf,Nm,V }
P = {
    Ph→Gn Gv
    Gn→Df Nf | Dm Nm
    Gv → V Gn
    Df → une | la
    Dm→ un | le
    Nf→ fille | cerise
    Nm → enfant | garçon | haricot
    V → cueille | mange

}

- La phrase "**une cerise cueille un enfant**" appartient-elle au langage L(G)?

Grammaire hors-Contexte (GHC)
Solution 4

Pour montrer qu'une phrase appartient au langage, on construit une dérivation de l'axiome Ph jusqu'à la phrase.

On souligne à chaque fois le symbole non terminal qui est remplacé par la dérivation.

<u>Ph</u> ⇒ <u>Gn</u> Gv ⇒ Df Nf <u>Gv</u> ⇒ Df Nf V <u>Gn</u>

⇒ <u>Df</u> Nf V Dm Nm ⇒ une <u>Nf</u> V Dm Nm

⇒ une cerise <u>V</u> Dm Nm

⇒ une cerise cueille <u>Dm</u> Nm

⇒ une cerise cueille un <u>Nm</u>

⇒ une cerise cueille un enfant

P = {Ph→Gn Gv
Gn→Df Nf | Dm Nm
Gv → V Gn
Df → une | la
Dm→ un | le
Nf→ fille | cerise
Nm → enfant | garçon | haricot
V → cueille | mange }

GHC Linéaire Droite

- Une grammaire ***G* = <*T, NT, S, P*>** *HC est dite* :
  - ***Linéaire Droite :*** si l'ensemble de ses règles de réécriture ***P*** *sont de la* forme : $NT \rightarrow (T \cup T.NT)$
    - La partie droite des règles de récriture contient un symbole terminal OU un symbole terminal suivi d'un symbole non-terminal
    - e.g. **G : S → aS | a**

GHC Linéaire Gauche

- Une grammaire ***G = <T, NT, S, P>*** *HC est dite* :
  - ***Linéaire Gauche :*** si l'ensemble de ses règles de réécriture ***P*** *sont de la* forme : $NT \rightarrow (T \cup NT.T)$
    - La partie droite des règles de récriture contient un symbole terminal OU un symbole non-terminal suivi d'un symbole terminal
    - e.g. **G : S → Sa | a**

Langages réguliers et Grammaire

- **Théorèmes :**

  - Toute **grammaire HC Linéaire Droite G** génère un **langage régulier L(G)** (L(G) est reconnu par un automate d'état fini)
    - e.g. **G : S $\rightarrow$ aS | a**

  - Tout **langage régulier L** possède une **grammaire HC Linéaire Droite G** (L(G) = L)

Algorithme DFA → GHC

Principe de la construction :

- ❑ *Soit le DFA* $M = \langle \mathbf{Q}, T, \delta, q_0, F \rangle$
  - ➢ *si* $q_0 \notin F$ *: on définit* $G = \langle T, \mathbf{Q}, q_0, P \rangle$ *équivalent avec*
    - ✓ $p \rightarrow aq \in P \iff \delta(p,a)=q$
    - ✓ $p \rightarrow a \in P \iff \delta(p,a) \in F$
  - ➢ *si* $q_0 \in F$ *: on fait comme le cas précédant + on rajoute la variable S et*
    - ✓ $S \rightarrow q_0 \mid \varepsilon$

Algorithme DFA → GHC
Autrement dit

Input : A = <S, Σ, δ, s_0, F >
Output : G = <T, NT, S, P>

- On fait correspondre
  - A chaque **s** de **S**, un élément de **NT** (NT(s))
  - A chaque élément **l** de Σ, un élément de **T** (T(l))
  - A chaque élément **(s,l,s')** de δ, un élément de P (P(s,l,s'))
  - A s_0 le non terminal **S**
  - A chaque élément **s** de **F**, une règle **NT**(**s**)--> ε