

# Contents

---

<b>1</b>	<b>Projet compilo</b>	<b>2</b>
1.1	Définitions . . . . .	2
1.2	Schémas . . . . .	2
1.3	Construction de la grammaire G0 (il doit manquer pas mal de trucs) . . . . .	3
1.3.1	Notation B.N.F. . . . .	3
1.3.2	Regle 1 . . . . .	3
1.3.3	Regle 2 . . . . .	4
1.3.4	Regle 3 . . . . .	4
1.3.5	Regle 4 . . . . .	4
1.3.6	Regle 5 . . . . .	4
1.4	Structure de donnees . . . . .	4
1.5	Construction des 5 Arbres . . . . .	5
1.5.1	Fonctions Gen* . . . . .	5
1.5.2	Arbres . . . . .	6
<b>2</b>	<b>Grammaires LL(k)</b>	<b>8</b>
2.1	Premier(N) . . . . .	8
2.2	Suivants . . . . .	8
2.3	Grammaire LL(1) . . . . .	8
<b>3</b>	<b>Opérateurs <math>\doteq</math>, <math>&gt;</math>, et <math>&lt;</math></b>	<b>9</b>
<b>4</b>	<b>Types des grammaires</b>	<b>10</b>

# 1 Projet compilo

## 1.1 Définitions

**GPL** Grammaire Petit Langage

**Scanner** analyse lexicale

**Analyseur** autres analyses (syntaxique et semantique)

## 1.2 Schémas

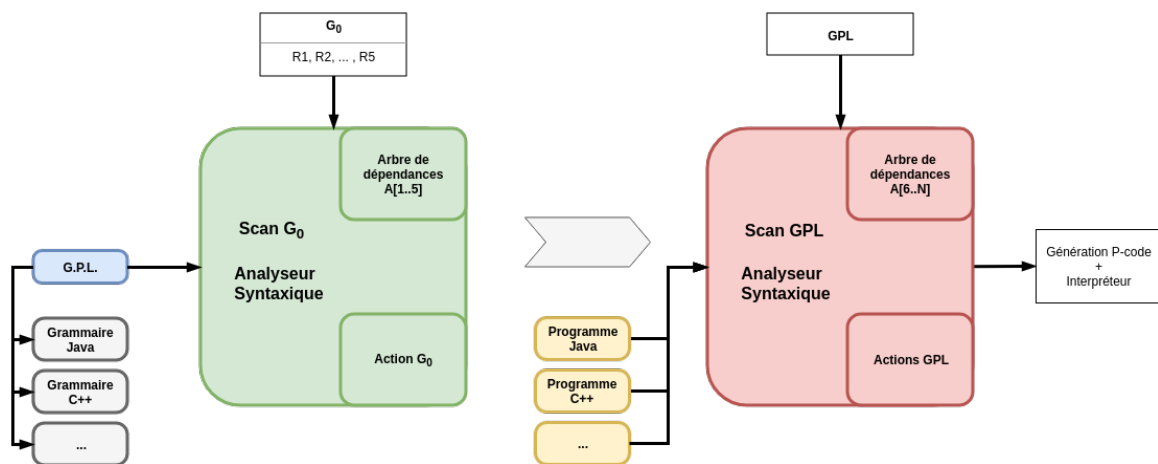
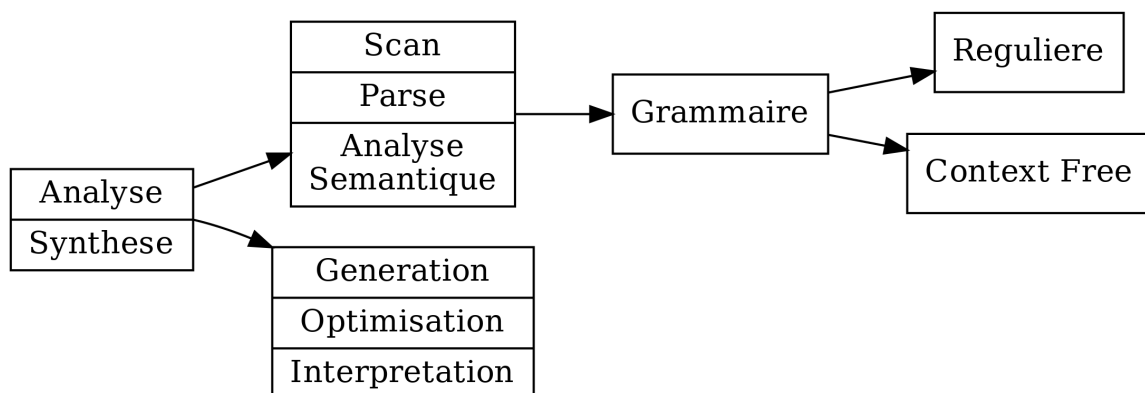


Figure 1.1: Projet Compilo





### 1.3.3 Regle 2

---

$$N \rightarrow' INDTER',$$

### 1.3.4 Regle 3

---

$$E \rightarrow R.[\text{'}.T],$$

### 1.3.5 Regle 4

---

$$T \rightarrow F.[\text{'}.F],$$

### 1.3.6 Regle 5

---

$$F \rightarrow' INDTER' + ' ELTER' + ' (.E.)' + ' [.E.]' + ' (/E./),;$$

## 1.4 Structure de donnees

---

Syntaxe maison...

```
Type Atomtype = (Terminal, Non-Terminal);
  Operation = (Conc, Union, Star, UN, Atom);
PTR = \uparrow{} Node

Node = Enregistrement
  case operation of
    Conc: (left, right : PTR);
    Union: (left, right : PTR);
    Star: (stare: PTR);
    UN: (UNE : PTR);
    ATOM: (COD, Act : int ; AType: Atomtype);
  EndEnregistrement

A: Array [1..5] of PTR;
```

## 1.5 Construction des 5 Arbres

---

### 1.5.1 Fonctions Gen\*

---

```
Fonction GenConc(P1, P2 : PTR) : PTR;
  var P : PTR;
debut
  New(P, conc);
  P \uparrow{}.left := P1;
  P \uparrow{}.right := P2;
  P \uparrow{}.class := conc;
  GenConc := P;
fin
```

```
Fonction GenUnion(P1, P2 : PTR) : PTR;
  var P : PTR;
d[U+FFFD]but
  New(P, union);
  P \uparrow{}.left := P1;
  P \uparrow{}.right := P2;
  P \uparrow{}.class := union;
  GenUnion := P;
fin
```

```
Fonction GenStar(P1 : PTR) : PTR; //0 ou n fois
  var P:PTR;
d[U+FFFD]butStar
  New(P, star);
  P \uparrow{}.stare := P1;
  P \uparrow{}.class := star;
  GenStar := P;
fin
```

```
Fonction GenUn(P1 : PTR) : PTR; //0 ou une fois
  var P:PTR;
d[U+FFFD]but
  New(P, un);
  P \uparrow{}.une := P1;
  P \uparrow{}.class := un;
  GenUn := P;
fin
```

```
Fonction GenAtom(COD, Act : int, AType : Atomtype) : PTR
  var P:PTR;
d[U+FFFD]but
  New(P, atom);
  P \uparrow{}.COD := COD;
  P \uparrow{}.Act := Act;
  P \uparrow{}.AType := AType;
  GenAtom := P;
fin
```

### 1. S

```
A[S] :=
  GenConc(
    GenStar(
      GenConc(
        GenConc(
          GenConc(GenAtom('N', \varnothing, NonTerminal),
            GenAtom('→', 5, Terminal)
          ),
          GenAtom('E', \varnothing, NonTerminal)
        ),
        GenAtom(',', , Terminal)
      ),
      GenAtom(';', , Terminal)
    );
```

### 2. N

//Ajouts de ma part, je ne suis pas sûr des résultats :

```
A[N] := GenAtom('IDNTER', , Terminal);
```

### 3. E

```
A[E] := GenConc(
  GenAtom('T', \varnothing, NonTerminal),
  GenStar(
    GenConc(
      GenAtom('+', ?, Terminal),
      GenAtom('T', \varnothing, Terminal)
    )
  )
);
```

### 4. T

```
A[T] := GenConc(
  GenAtom('F', \varnothing, NonTerminal),
  GenStar(
    GenConc(
      GenAtom('.', ?, Terminal),
      GenAtom('T', \varnothing, Terminal)
    )
  )
);
```

5. F

```
A[F] := GenUnion(  
  GenUnion(  
    GenUnion(  
      GenUnion(  
        GenAtom('IDNTER', , Terminal),  
        GenAtom('ELTER', , Terminal)  
      ),  
      GenConc(  
        GenConc(  
          GenAtom('(', ?, Terminal),  
          GenAtom('E', \varnothing, NonTerminal)  
        ),  
        GenAtom(')', ?, Terminal)  
      )  
    ),  
    GenConc(  
      GenConc(  
        GenAtom('[', ?, Terminal),  
        GenAtom('E', \varnothing, NonTerminal)  
      ),  
      GenAtom(']', ?, Terminal)  
    )  
  ),  
  GenConc(  
    GenConc(  
      GenAtom('(', ?, Terminal),  
      GenAtom('E', \varnothing, NonTerminal)  
    ),  
    GenAtom(')', ?, Terminal)  
  )  
)
```

## 2 Grammaires LL(k)

---

$k$  est une mesure de l'ambiguïté. Représente le nombre de caractères qu'il est nécessaire de regarder pour déterminer quelle règle utiliser. Bien entendu, les règles LL(1) sont préférables.

### 2.1 Premier(N)

---

- Si  $N \rightarrow A \dots$  alors  $Premier(N) = Premier(A)$
- Si  $N \rightarrow c \dots$  alors  $Premier(N) = \{c\}$
- Si  $N \rightarrow A.B \dots \wedge A \Rightarrow \epsilon$  alors  $Premier(N) = Premier(B)$

Avec " $\Rightarrow$ " signifiant "se derivant en".

Il ne s'agit pas d'appliquer une règle à chaque fois, mais plutôt d'appliquer toutes les règles possibles.

### 2.2 Suivants

---

- Si  $A \rightarrow \dots Nc \dots$  alors  $Suiv(N) = \{c\}$
- Si  $A \rightarrow \dots NB \dots$  alors  $Suiv(N) = Prem(B)$
- Si  $A \rightarrow N \dots$  alors  $Suiv(N) = Suiv(A)$

### 2.3 Grammaire LL(1)

---

- si  $A \rightarrow \alpha_1 / \alpha_2 / \dots / \alpha_n$  alors
$$Prem(\alpha_i) \cap Prem(\alpha_j) = \Phi, \forall i \neq j$$
- si  $A \Rightarrow \epsilon$  on doit avoir  $Prem(A) \cap Suiv(A) = \Phi$

Si une règle ne possède qu'une dérivation, la règle 1 ne s'applique pas. Si une règle ne possède pas de suiv, la règle 2 ne s'applique pas.



### 3 Opérateurs $\doteq$ , $\succ$ , et $\triangleleft$

---

—  $X \doteq Y$  si

$$A \rightarrow \dots X.Y \dots \in \mathcal{P}$$

—  $X \triangleleft Y$  si

$$A \rightarrow \dots X.Q \dots \in \mathcal{P}$$

$$\text{et } Q \xRightarrow{*} Y$$

—  $X \succ Y$  si

$$A \doteq Y$$

$$\text{et } A \xRightarrow{*} X$$

On peut remplir le tableau SR à partir des relations  $\doteq$ ,  $\succ$  et  $\triangleleft$  :

- (ligne  $\doteq$  colonne) et (ligne  $\triangleleft$  colonne) se traduisent en (ligne Shift colonne)
- (ligne  $\succ$  colonne) se traduit en (ligne Reduce colonne)

## 4 Types des grammaires

---

0 type c

1 type context sensitive CS  $\gamma \rightarrow \beta$  avec  $\|\gamma\| \leq \|\beta\|$

2 type context free CF  $A \rightarrow B$  avec  $A \in V_N, B \in V^+$

3 type reguliere  $\begin{cases} A \rightarrow aB \\ A \rightarrow a \end{cases}$  ou  $\begin{cases} A \rightarrow Ba \\ A \rightarrow a \end{cases}$

$$L(G) = \{x \in V_T^* / S \Rightarrow x\}$$

l'intersection de deux languages de type x n'est pas forcement de type x.