

Contents

1	Projet compilo	2
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	3
1.3.4	Regle 3	3
1.3.5	Regle 4	3
1.3.6	Regle 5	4
1.4	Structure de donnees	4
1.5	Construction des 5 Arbres	4
2	Grammaires LL(k)	5
2.1	Premier(N)	5
2.2	Suivants	5
2.3	Grammaire LL(1)	5
3	Opérateurs \doteq, $>$, et $<$	6
4	Types des grammaires	7

1 Projet compilo

1.1 Définitions

GPL Grammaire Petit Langage

Scanner analyse lexicale

Analyseur autres analyses (syntaxique et semantique)

1.2 Schémas

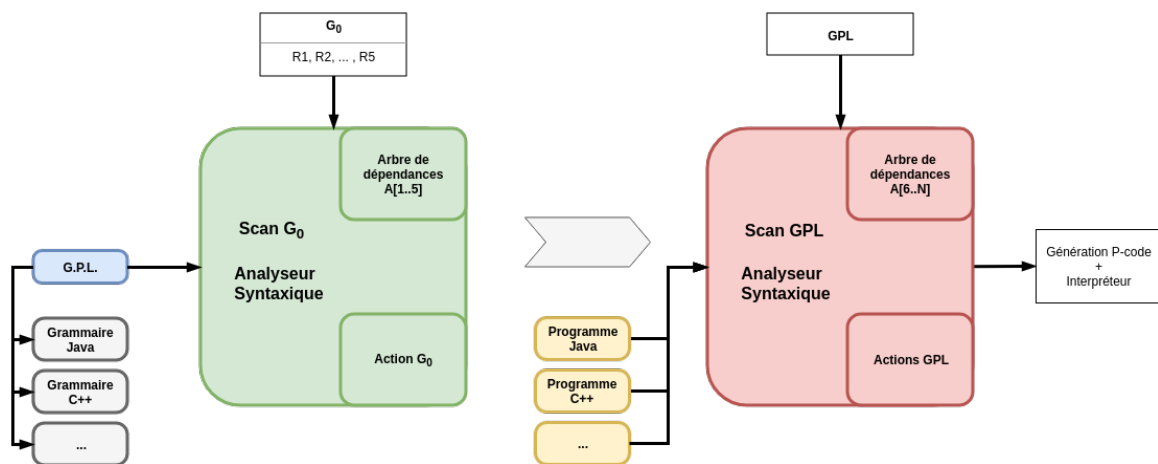
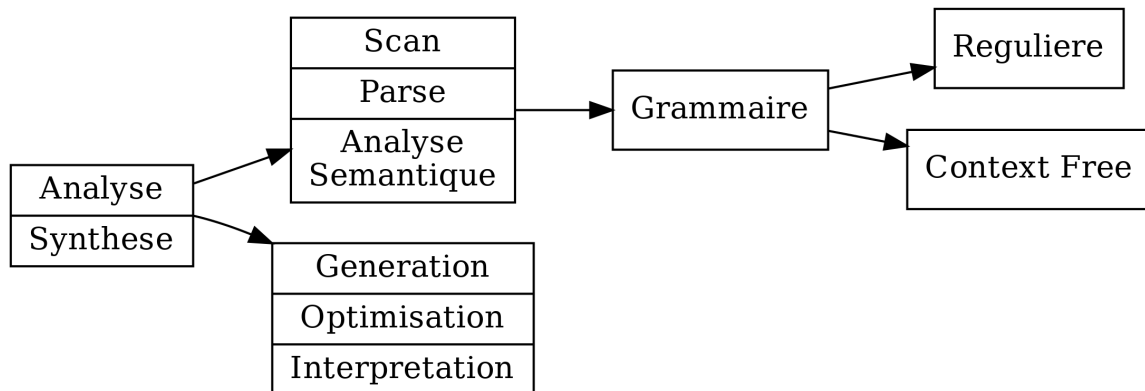
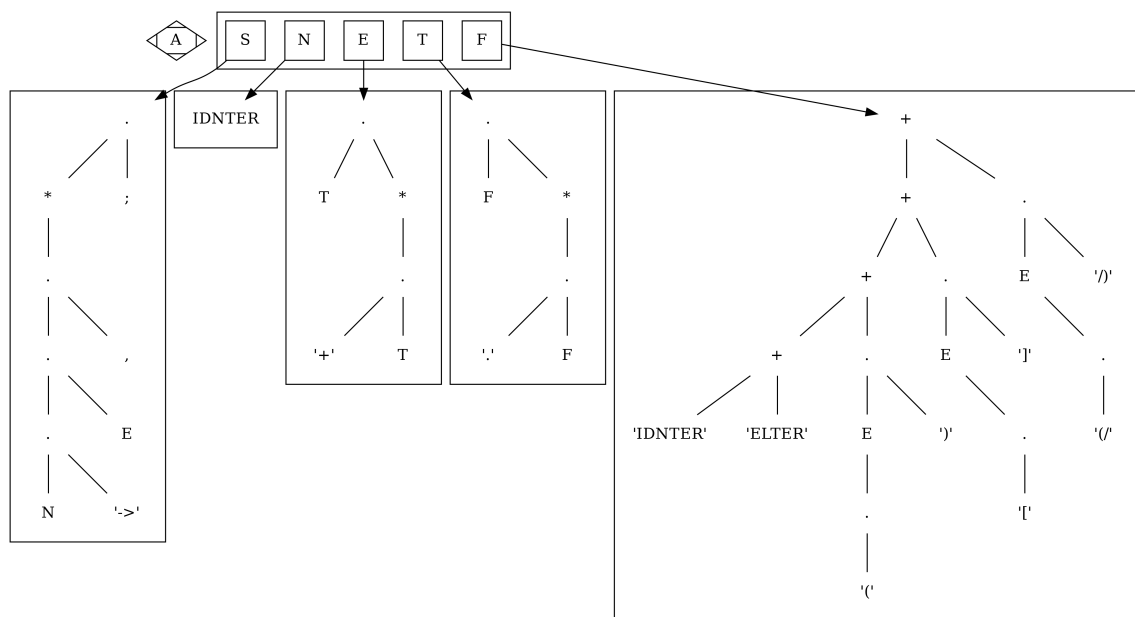


Figure 1.1: Projet Compilo





1.3 Construction de la grammaire G0 (il doit manquer pas mal de trucs)

1.3.1 Notation B.N.F.

- $::= \iff \rightarrow$
- $[X] \iff X.X.X...X(n \text{ fois}), n \geq 0$
- $x \iff \cdot$

1.3.2 Regle 1

$$S \rightarrow [N.' \rightarrow' .E.'',']';',$$

- concatenation $\iff \cdot$
- pour differencier les terminaux et les non terminaux, on met les terminaux entre guillemets

1.3.3 Regle 2

$$N \rightarrow' INDTER',$$

1.3.4 Regle 3

$$E \rightarrow R.['+' .T],$$

1.3.5 Regle 4

$$T \rightarrow F.[' ' .F],$$

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

```
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;
  Conc := P;
fin

GenUnion,
GenStar, //0 ou n fois
GenUn, //0 ou une fois
GenAtom

A[S] :=
  GenConc(
    GenStar(
      GenConc(
        GenConc(
          GenConc(GenAtom('N', , NonTerminal)),
          GenAtom('->', , Terminal)
        ),
        GenAtom('E', , )
      ),
      GenAtom(',', , Terminal)
    ),
    GenAtom(';', , Terminal)
  );
```

2 Grammaires LL(k)

k est une mesure de l'ambigüité. 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 $\text{Premier}(N) = \text{Premier}(A)$
- Si $N \rightarrow c \dots$ alors $\text{Premier}(N) = \{c\}$
- Si $N \rightarrow A.B \dots \wedge A \Rightarrow \epsilon$ alors $\text{Premier}(N) = \text{Premier}(B)$

Avec " \Rightarrow " signifiant "se dérivant 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 $\text{Suiv}(N) = \{c\}$
- Si $A \rightarrow \dots NB \dots$ alors $\text{Suiv}(N) = \text{Prem}(B)$
- Si $A \rightarrow N \dots$ alors $\text{Suiv}(N) = \text{Suiv}(A)$

2.3 Grammaire LL(1)

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

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.