Contents

1 Projet compilo

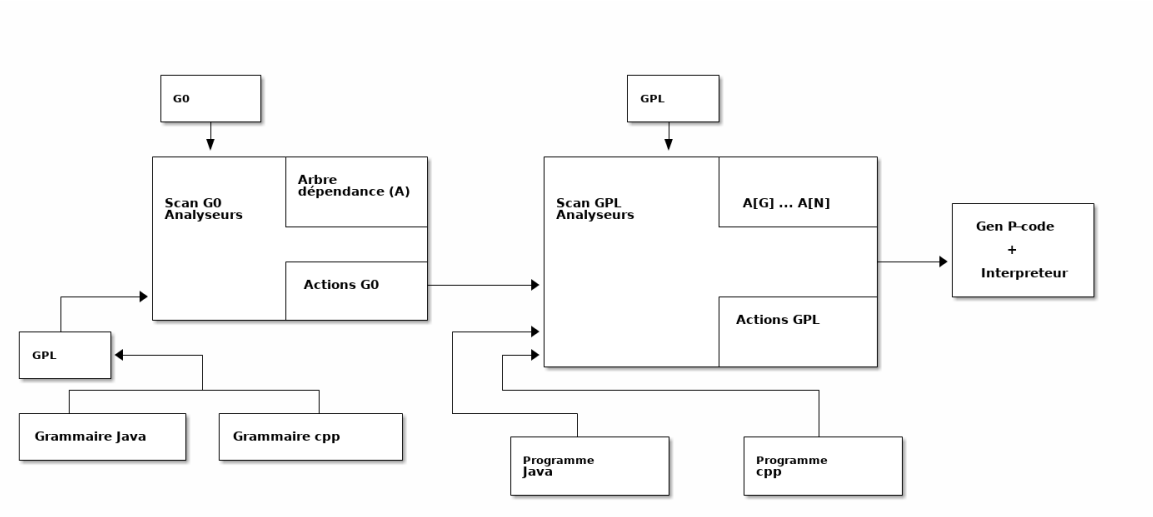
1.1 Définitions

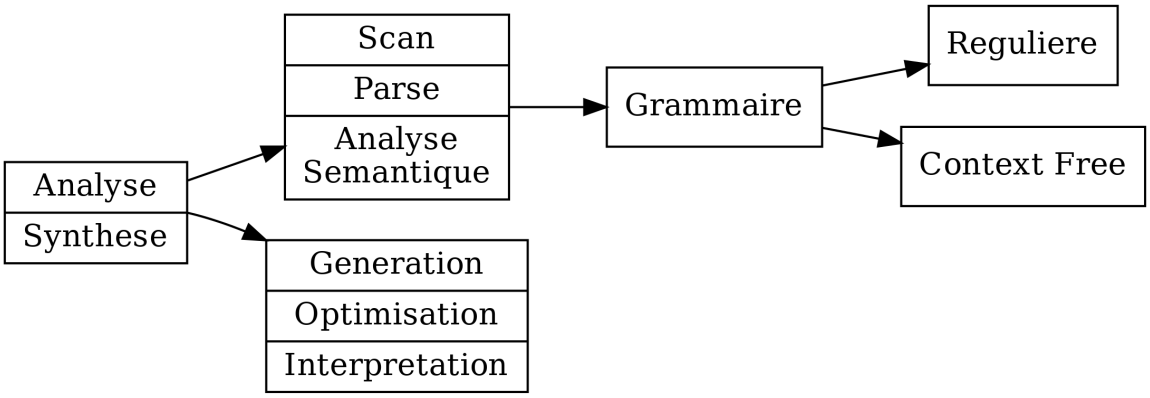
GPL Grammaire Petit Langage

Scanner analyse lexicale

Analyseur autres analyses (syntaxique et semantique)

1.2 Schémas





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 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'ambiguite. Represente le nombre de caracteres qu'il est necessaire de regarder pour determiner quelle regle utiliser. Bien entendu, les regles LL(1) sont preferables.

2.1 Premier(N)

— Si $N \to A \ldots$ alors $Premier(N) = Premier(A)$
— Si $N \to c \ldots$ alors $Premier(N) = \{c\}$
— Si $N \to A.B \ldots \wedge A \Rightarrow \epsilon$ alors $Premier(N) = Premier(B)$

Avec "⇒" signifiant "se derivant en".

Il ne s'agit pas d'appliquer une regle a chaque fois, mais plutot d'appliquer toutes les regles possibles.

2.2 Suivants

— Si $A \to \ldots Nc \ldots$ alors $Suiv(N) = \{c\}$
— Si $A \to \ldots NB \ldots$ alors $Suiv(N) = Prem(B)$
— Si $A \to N \ldots$ alors $Suiv(N) = Suiv(A)$

2.3 Grammaire LL(1)

— si $A \to \alpha_1 / \alpha_2 / \ldots / \alpha_n$ 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 regle ne possede qu'une derivation, la regle 1 ne s'applique pas. Si une regle ne possede pas de suiv, la regle 2 ne s'applique pas.

3 Opérateurs $\doteq$, $\gtrdot$, et $\lessdot$

— $X \doteq Y$ si

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

— $X \lessdot Y$ si

$$A \to \dots X.Q \dots \in \mathcal{P}$$
$$\text{et } Q \overset{*}{\Rightarrow} Y$$

— $X \gtrdot Y$ si

$$A \doteq Y$$
$$\text{et } A \overset{*}{\Rightarrow} X$$

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

— (ligne $\doteq$ colonne) et (ligne $\lessdot$ colonne) se traduisent en (ligne Shift colonne)
— (ligne $\gtrdot$ colonne) se traduit en (ligne Reduce colonne)

4 Types des grammaires

0 type c

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

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

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

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

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