INPG — **ENSIMAG**
Année Spéciale — **Année 2005-06**

Compilation

Fiche de cours - analyse syntaxique descendante

1 Principe

L'analyse descendante s'effectue en partant de l'axiome de la grammaire, que l'on dérive *systématiquement* par le non-terminal le plus à gauche, jusqu'à obtenir l'arbre de dérivation de la phrase à analyser.

Lorsque l'on a dérivé jusqu'à un non-terminal et qu'il ne correspond pas au lexème recherché, l'analyse doit revenir en arrière (en anglais, *backtracking*), *i.e.* remonter d'une application dans l'arbre de dérivation et essayer une autre production.

1.1 Grammaires LL(1)

LL(1) signifie : *Left-to-right scanning, Leftmost derivation, use 1 symbol lookahead.*

Définition 1 *$First(\alpha)$ ($\alpha \in (V_T \cup V_{NT})^*$) est l'ensemble des symboles terminaux qui peuvent apparaître comme premier symbole dans une phrase dérivée du symbole α.*

pour tout $\alpha \in V_T$
  $First(\alpha) \leftarrow \alpha$
pour tout $A \in V_{NT}$
  $First(A) \leftarrow \emptyset$
tant que (les ensembles $First$ changent)
  pour tout $p \in P$ tq p est de la forme $A \rightarrow \beta_1\beta_2...\beta_k$
    $First(A) \leftarrow First(A) \cup (First(\beta_1) \setminus \{\epsilon\})$
    $i \leftarrow 1$
    tant que $(\epsilon \in First(\beta_i) \wedge i \leq k-1)$
      $First(A) \leftarrow First(A) \cup (First(\beta_{i+1}) \setminus \{\epsilon\})$
      $i \leftarrow i+1$
    si $(i = k \wedge \epsilon \in First(\beta_k))$ alors
      $First(A) \leftarrow First(A) \cup \{\epsilon\}$

Tab. 1 – Algorithme de calcul des ensembles $First$

On notera que $First(\alpha)$ peut contenir ϵ. Ce cas est problématique, car on ne peut pas savoir quel non-terminal sera rencontré une fois que A aura été dérivé totalement. On définit alors les ensembles $Follow$, qui n'ont de sens que définis sur les non-terminaux, qui vont permettre au *parser* de savoir qu'est-ce qui apparaît immédiatement après une dérivation nulle.

Définition 2 *$Follow(A)$ ($A \in V_{NT}$) est l'ensemble des symboles terminaux qui peuvent apparaître immédiatement après A dans un* mot *du langage.*

pour tout $A \in V_{NT}$
  $Follow(A) \leftarrow \emptyset$
tant que (les ensembles $Follow$ changent)
  pour tout $p \in P$ tq p est de la forme $A \rightarrow \beta_1\beta_2...\beta_k$
    si ($\beta_k \in V_{NT}$) alors $Follow(\beta_k) \leftarrow Follow(\beta_k) \cup Follow(A)$
    $After \leftarrow Follow(A)$
    pour i de k a 2
      si ($\epsilon \in First(\beta_i)$) alors
        si ($\beta_{i-1} \in V_{NT}$) alors $Follow(\beta_{i-1}) \leftarrow Follow(\beta_{i-1}) \cup (First(\beta_i) \setminus \{\epsilon\}) \cup After$
      sinon
        si ($\beta_{i-1} \in V_{NT}$) alors $Follow(\beta_{i-1}) \leftarrow Follow(\beta_{i-1}) \cup First(\beta_i)$
        $After \leftarrow First(\beta_i)$

Tab. 2 – Algorithme de calcul des ensembles $Follow$

À l'aide de ces deux ensembles on peut écrire la condition que doit remplir une grammaire pour être LL(1).

Notons $First^+(\alpha)$ l'ensemble $First(\alpha)$ si $\epsilon \notin First(\alpha)$ ou $(First(\alpha) \cup Follow(\alpha))$ sinon. la grammaire doit vérifier que pour tout non-terminal A ayant plusieurs dérivations possibles $A \longrightarrow \beta_1 \mid \beta_2 \mid ... \mid \beta_n$ on doit avoir :

$$\forall i, j \quad 1 \leq i < j \leq n \qquad First^+(\beta_i) \cap First^+(\beta_j) = \emptyset$$