Analyseur descendant récursif

Descente récursive LL(1)

Préambule

Vous avez découvert dans ce module deux types d'analyseurs syntaxiques descendants :

- les analyseurs récursifs
- les analyseurs non-récursifs

L'analyse syntaxique par descente récursive est une méthode d'analyse syntaxique descendante dans laquelle on exécute un ensemble de procédures récursives pour traiter la chaîne d'entrée. Une procédure est associée à chaque non-terminal d'une grammaire. Une analyse syntaxique récursive LL(1) se décompose en deux parties :

(*i*) **construction de l'analyseur** : on écrit le programme principal, la procédure d'analyse d'un terminal ainsi qu'une procédure d'analyse pour chaque non-terminal

(*ii*) **reconnaissance d'un mot** : on appelle la procédure principale. Le résultat de l'exécution fournit, si le mot est reconnu, l'arbre syntaxique du mot analysé en notation postfixée

Construction de l'analyseur

Construire l'analyseur syntaxique récursif LL(1) d'une grammaire consiste à effectuer successivement les étapes suivantes :

- vérifier que la grammaire n'est pas ambiguë
- réduire la grammaire si nécessaire
- dérecursiver la grammaire si nécessaire
- factoriser la grammaire
- déterminer les non-terminaux produisant le vide
- calculer les premiers et les suivants des non-terminaux
- en déduire les symboles directeurs des règles
- écrire le programme principal et la procédure d'analyse d'un terminal[1]
- écrire la procédure d'analyse associée à chaque non-terminal

Programme principal

Algorithme 1 Procédure Principale

```
Initialisation : erreur ← faux ; res ← ϵ ;
  lire(y) ;
  Ana-X() ; // si X est l'axiome
  si erreur alors
    afficher "le mot n'est pas engendré par la grammaire" ;
  sinon
    si y = $ alors
      afficher "le mot est engendré par la grammaire" ;
      afficher res ;
    sinon
      afficher "le mot n'est pas engendré par la grammaire" ;
    fin si
  fin si
```

[1] Étape identique quelque soit la grammaire.

Procédure d'analyse d'un terminal

Algorithme 2 Procédure Ana(x :T)

si $\neg erreur$ **alors**
  si $y = x$ **alors**
    $res \leftarrow res \oplus x$;
    lire(y) ;
  sinon
    $erreur \leftarrow vrai$
  fin si
fin si

Procédure d'analyse d'un non-terminal N

Pour rester général, on note les k règles dont N est le membre gauche de la façon suivante :

$$N \rightarrow \alpha_1\alpha_2 \ldots \alpha_n \ \textcircled{1} \ \mid \ \beta_1\beta_2 \ldots \beta_p \ \textcircled{2} \ \mid \ \ldots \ \mid \ \gamma_1 \ldots \gamma_m \ \ \textcircled{k}$$

et $SD(\textcircled{i})$ les symboles directeurs de la règle numéro i.

Algorithme 3 Procédure Ana-N()

si $\neg erreur$ **alors**
  cas
    $y \in SD(\textcircled{1})$: Ana-α_1 ; ... ; Ana-α_n ; $res \leftarrow res \oplus 1$
    $y \in SD(\textcircled{2})$: Ana-β_1 ; ... ; Ana-β_p ; $res \leftarrow res \oplus 2$
    ...
    $y \in SD(\textcircled{k})$: Ana-γ_1 ; ... ; Ana-γ_m ; $res \leftarrow res \oplus k$
    autres cas : $erreur \leftarrow vrai$
  fin cas
fin si