Université Bordeaux
Licence, module Compilation, 2016/2017

TD4 – Analyse syntaxique, analyse sémantique

Exercice 4.1

Écrire une définition dirigée par la syntaxe pour une expression de la logique propositionnelle. Nous rappelons que le langage de la logique propositionnelle est constitué de variables propositionnelles $p, q, \ldots$, d'opérateurs $\wedge, \vee, \neg, \rightarrow, \leftrightarrow$ et les constantes 0 et 1.

On écrira la grammaire de telle sorte que le programmeur puisse instancier une ou plusieurs variables avant de les utiliser comme opérandes dans l'expression. L'opérateur d'affectation est :=, le séparateur obligatoire pour l'affectation est le point-virgule.

Exemple :

$$p := 1; q := \neg p; (p \rightarrow q) \rightarrow p$$

Implémenter cette grammaire en *Yacc* de telle sorte que les affectations et l'expression soient données en entrée standard du programme et le résultat soit affiché en sortie standard.

Pour faciliter l'écriture de la grammaire en *Yacc*, on utilisera les propriétés de précédence d'opérateurs. Nous rappelons que les opérateurs $\vee, \rightarrow, \leftrightarrow$ sont prioritaires sur $\wedge$ et que $\neg$ est prioritaire sur tous les autres. Par ailleurs, tous les opérateurs binaires sont associatifs à gauche.

Exercice 4.2

Une grammaire attribuée est une extension d'une grammaire algébrique $G = (\Sigma, V_N, S, R)$ où

- Dans un arbre de dérivation quelconque, on attache à chaque noeud étiqueté par X un ensemble fini d'attributs : $A(X) = \{a_1(X), \ldots, a_n(X)\}$.
  $A(X)$ est partitionné en deux ensembles disjoints : $A_s(X)$ et $A_h(X)$, respectivement les attributs synthétisés, et les attributs hérités. Les attributs synthétisés $A_s(X)$ sont ceux dont le calcul dépend des attributs attachés aux noeuds en dessous de X dans l'arbre de dérivation. Les attributs hérités $A_h(X)$ sont ceux qui sont calculés à partir des noeuds frères ou père de X.
- On associe à chaque production $p = X_0 \rightarrow X_1 \ldots X_n$ un ensemble de règles $R(p) = \{a_n(X_i) = f(a_m(X_j), \ldots)\}$

Prenons un exemple simple inspiré de [Knuth, 1968], dans le domaine des grammaires attribuées et associant à chaque nombre binaire, sa valeur décimale.

- $\Sigma = \{0, 1\}$
- S, N, B sont les symboles de la grammaire (B pour les chiffres binaires et N pour les nombres exprimés avec ces chiffres). On distinguera N_1 et N_2, deux occurrences différentes du même symbole N dans une même règle.
- S est le symbol initial
- Pour tout noeud N ou B de l'arbre de dérivation, v est un attribut synthétisé et s un attribut hérité. Le premier contient la valeur du nombre binaire, le second correspond au décalage du chiffre à partir de la droite.

$N \rightarrow B$	$v(N) = v(B), s(B) = s(N)$
$N_1 \rightarrow N_2 B$	$v(N_1) = v(N_2) + v(B), s(N_2) = s(N_1) + 1, s(B) = s(N_1)$

$B \rightarrow 0$	$v(B) = 0$
$B \rightarrow 1$	$v(B) = 2^{s(B)}$
$S \rightarrow N$	$s(N) = 0$

Décrire dans le détail le calcul de la valeur décimale du nombre 1001. On dessinera un arbre de dérivation décoré d'un graphe des attributs hérités et d'un graphe des attributs synthétisés.

Exercice 4.3

Pour quelle(s) raison(s) n'est-il pas possible d'écrire une grammaire *YACC* qui implémente immédiatement la grammaire précédente ?

Tenter d'écrire une grammaire *YACC* qui implémente cette grammaire en utilisant les différents outils de la programmation impérative C.

Références

[Knuth, 1968] Knuth, D. E. (1968). Semantics of context-free languages. *Mathematical Systems Theory*, 2(2) :127–145.