LOG3210- Éléments de langages et compilateur

Groupe 01

Travail pratique #1

Grammaire et analyseur syntaxique

Présenté à Karl JULIEN

David TREMBLAY 1748125

Alexandre Clark 1803508

Département de génie informatique et génie logiciel

Le 6 février 2018
École polytechnique de Montréal

Question 1

À partir du fichier Template.jjt, écrivez les règles de production relatives aux expressions entières selon le formalisme vu dans le cours (sans notation EBNF).

A: IntExpr()
B: IntAddExpr()
C: IntMultExpr()
D: IntNegExpr()
E: IntBasicExpr()
f: Identifier()
g: IntValue()

A -> B
B -> C ((+ | -)C)*
C -> D ((* | /) D)*
D -> -E() | E
E -> f | g | (A)

Question 2

On considère la définition actuelle du nœud IntAddExpr. Les définitions alternatives suivantes sont-elles acceptables? (justifier)

I.
Cette définition n'est pas acceptable puisqu'il manque * à la fin. Il est donc impossible d'additionner plus de 2 résultats. (2*3) + (2*3) + (2*3) est impossible par exemple.

II.
Cette définition n'est pas acceptable puisqu'elle permet la récursivité à gauche.

III.
Cette définition est acceptable. La récursivité à droite permet d'enchaîner les additions de multiplications.

IV.
Cette définition n'est pas acceptable puisqu'elle permet la récursivité à gauche.

V.
Cette définition est acceptable. Par contre, on perd la cohérence des classes.

La définition actuelle règle tous les problèmes rencontrés ici. La cohérence des classes est respectée puisqu'on n'autorise pas IntNegExpr() directement dans IntAddExpr. Aussi,

l'étoile permet l'enchaînement des additions et la récursivité à gauche n'est pas permise.

Question 3

Étant donnée la grammaire des expressions, pourquoi doit-on définir des assignations différentes selon le type (=, =., =*) ?

Il est nécessaire de définir des assignations différentes pour les opérations entières, réelles et sur les tableaux puisque la grammaire fournit des opérations différentes selon le type de variable traitée.

Question 4

Si vous avez utilisé des LOOKAHEAD(2), quels en sont l'intérêt dans votre contexte d'utilisation?

Le but d'utiliser des LOOKAHEAD(2) est d'aller observer les deux prochains tokens pour déterminer le chemin à suivre. Ils sont particulièrement utiles lorsque le premier token de plusieurs expressions séparées par l'opérateur logique "OU" est le même.