Langages et Traducteurs

TD7 : Syntaxe abstraite – Ambiguités sémantiques

Exercice 1.

Soit G une grammaire décrivant des affectations dans laquelle les terminaux c et i désignent respectivement une constante entière et un identificateur :

```
Z -> A ;  
A -> i := E  
E -> E + T | T  
T -> T * F | F  
F -> c | i | (E)
```

On considère les deux affectations suivantes (x et y sont des identificateurs) :

$$x := 5 + x * 2 + y; \quad (1)$$

$$y := (5 + x) * 2 + y; \quad (2)$$

1. Construisez les arbres de dérivation correspondant aux affectations (1) et (2).

2. On suppose que l'on dispose d'une machine à 3 adresses, comportant 16 registres ($R_0, R_1, \dots, \widehat{R}_{15}$), et disposant des instructions suivantes :

- LD R_i, x : charge la valeur de x dans le registre R_i
- ST R_i, x : affecte à x la valeur contenue dans le registre R_i
- ADD $R_i, op1, op2$ range dans R_i la somme $op1 + op2$ ($op1, op2 =$ registre ou constante)
- MULT $R_i, op1, op2$ range dans R_i le produit $op1 * op2$ ($op1, op2 =$ registre ou constante)

Ecrivez les séquences d'instruction correspondant à la traduction de (1) et (2).

3. Décorez les arbres de dérivation de (1) et (2) avec les séquences d'instructions obtenus à la question précédente : chaque instruction doit décorer le nœud de l'arbre qu'elle traduit (certains nœuds n'étant pas décorés).

4. En vous inspirant du résultat de la question 3, proposez une *syntaxe abstraite* pour G (sous la forme d'une grammaire hors-contexte G').

Exercice 2

Les fragments de programme ci-dessous correspondent aux instructions "for" en ADA et C (sous une forme simplifiée) :

```
for i in <expression 1>..<expression2> loop  
-- instructions  
end loop ;
```

```
for (<instruction> ; <condition d'arrêt> ; <instruction>) {  
/* instructions */  
}
```

1. Proposez une syntaxe concrète de l'instruction `for` pour chacun de ces deux langages.
2. Proposez une syntaxe abstraite correspondante
3. Intuitivement, quelles vérifications doit faire le compilateur dans chacun de ces deux cas pour s'assurer que l'instruction `for` est "correcte" ?

Exercice 3

Quel est la valeur des variables après exécution des fragments de pgm C suivants :

```
x = 3 ;
y = 2 ;
z = 3*x + x=2 + (x+1) ; /* en C x=2 rend la valeur 2 */
```

```
x = 3 ;
y = 2 ;
z = x + f(y) ; avec f(int a) {return a+1}
```

```
x = 3 ;
y = 2 ;
z = y + f(*y) ; avec f(int *a) {return *a=*a+1}
```

```
x = 3 ;
y = 2 ;
z = x + f(y) ; avec f(int a) {x=4 ; return a}
```

Exercice 4

On cherche à définir un petit langage fonctionnel. Quelles sont les valeurs possibles (raisonnables) pour les expressions suivantes :

1. soit $x=3$ dans $x+1$
2. soit $x=3$ dans soit $y=x+1$ dans $y+x$
3. soit $x=2$ dans $x +$ soit $x=3, y=x+1$ dans $x+y$
4. soit $x=2$ dans soit $x=x+1, y=x+1$ dans $x+y$
5. soit $x=3$ dans soit $f(y) = y+1$ dans $x + f(5)$
6. soit $x=3$ dans soit $f(y) = y+x$ dans $x + f(5)$
7. soit $x=3$ dans soit $f(y)=y+x$ dans soit $x=7$ dans $x + f(5)$