Master Info - 2016-2017
MIF08 Compilation

Contrôle continu 1 - Durée 20 min

Éléments de correction

1 Une grammaire

Les expressions numériques du langage Mu suivent la grammaire (simplifiée) `.g4` suivante :

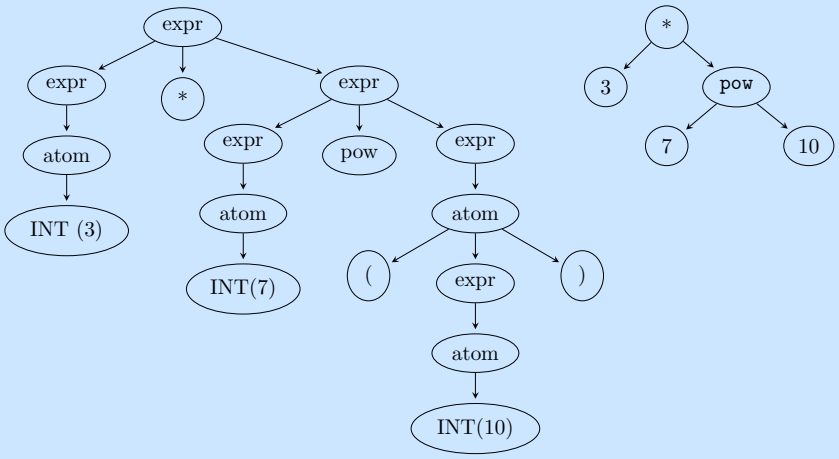
```
expr
 : <assoc=right> expr POW expr          #powExpr
 | expr '*'  expr                       #multiplicationExpr
 | expr '+' expr                        #additiveExpr
 | atom                                 #atomExpr
 ;
atom
 : OPAR expr CPAR #parExpr
 | INT  #numberAtom
 ;
```

On rappelle que les priorités sont exprimées par l'ordre des règles.

1. Quel est l'arbre de **dérivation** pour l'expression `3*7^(10)` ?
2. Quel est l'arbre de syntaxe abstrait correspondant ?

Solution:

L'arbre de dérivation (à gauche) comporte comme noeuds internes les non-terminaux de la grammaire, et aux feuilles les terminaux (y compris les parenthèses). L'arbre de syntaxe abstrait (à droite) quant à lui a pour noeuds internes les opérations, et n'a plus le sucre syntaxique comme les parenthèses :



Attention à la priorité des opérateurs, comme la puissance est prioritaire sur la multiplication, l'opération "pow" doit apparaître plus bas dans l'arbre (car elle doit être réalisée en premier).
En cas de confusion entre les deux arbres, j'ai quand même donné des points. Idem si la priorité des opérateurs a été mal traité. Idem pour les parenthèses pour l'AST.

2 Une attribution

On considère la grammaire (.g4) suivante :

```
grammar Tree;

tree:  INT                      #feuille
    |  NODE '(' INT tree+ ')'   #arbre
    ;

INT: [0-9]+;
NODE: 'node';
```

Cette grammaire permet de représenter des arbres n-aires, par exemple `node (42 12 1515 17)` représente l'arbre de racine 42 avec 3 fils $12, 1515, 17$. Écrire une attribution qui permet de décider si un arbre reconnu par la grammaire est binaire (une feuille est un arbre binaire, un noeud est un arbre binaire ssi il a deux 2 fils qui sont des arbres binaires). On vous fournit le code à remplir :

Solution:
J'invente une attribution qui renvoie true aux feuilles, et pour un arbre regarde si il y a *exactement* deux fils et qu'ils sont eux-mêmes binaires. Voici mon code :

```
from TreeVisitor import TreeVisitor
from TreeParser import *

class MyTreeVisitor(TreeVisitor):

    def visitFeuille(self, ctx):
        return True #une feuille est un A.B.

    def visitArbre(self, ctx):
        children=ctx.tree()
        if not (len(children)==2) : #+ de 2 fils -> non.
            return False
        else:
            for tree in children: #appel sur les 2 fils.
                if not self.visit(tree): #si 1 n'est pas binaire
                    -> non.
                    return False
            return True
```

et pour le main :

```
    visitor = MyTreeVisitor();
    b=visitor.visit(tree);
    print("Is␣it␣a␣binary␣tree␣?␣"+str(b));
```

— Pour l'attribution : accéder au contenu de la feuille est inutile, mais je n'ai pas sanctionné. J'ai mis des points même si votre attribution n'est pas cohérente mais qu'il y a des bouts (en particulier si on voit l'appel au visiteur des fils), alors qu'elle doit renvoyer une information du même type (ici, j'attendais quand même un booléen). J'ai aussi mis des points à des solutions partielles).

— Pour le main, je n'ai pas sanctionné de syntaxe incorrecte, mais il fallait que "tree" et "visitor" apparaissent dans votre appel. J'ai enlevé des points à l'utilisation de visitorArbre au lieu de visitor tout court.