TD 2 : Langages algébriques

Exercice 1

Considérons la grammaire définie par les deux productions suivantes :
S → (L) | a
L → L,S | S
Les quatre symboles terminaux de cette grammaire sont : a , ( ). Les deux symboles non terminaux sont : L et S. Et S est l'axiome.
Construire, si possible, une suite de dérivations gauches, une suite de dérivations droites et un arbre d'analyse pour chacune des phrases suivantes :

- (a,a)
- (a,(a,a))
- (a,(a,a),a)
- ((a))
- (a,a,a,a)))

Exercice 2

Soit la grammaire donnée par la production suivante :
S → aSbS | bSaS | ε
S est l'axiome et le seul symbole non terminal. a et b sont les symboles terminaux.

1) Montrer que cette grammaire est ambiguë.
2) Quel est le langage engendré par cette grammaire ?

Exercice 3

Soit la grammaire G1 suivante :
EXP → EXP + EXP | EXP – EXP | EXP * EXP | EXP / EXP | (EXP) | entier
Cette grammaire a un symbole non terminal, EXP, qui est l'axiome de la grammaire, et sept symboles terminaux (+, -, *, /, (,), entier).

1) Quel est le langage engendré par cette grammaire ?
2) Montrer que la grammaire G1 est ambiguë en identifiant deux arbres d'analyse pour le mot 1+2+3.
3) Soit la grammaire G2 suivante (qui reconnaît le même langage) :
   EXP → EXP + OPERANDE | EXP – OPERANDE | EXP * OPERANDE
   | EXP / OPERANDE | OPERANDE
   OPERANDE → (EXP) | entier

   Vérifier qu'il n'existe plus qu'un arbre d'analyse pour l'expression 1+2+3.

   Remarquer ainsi qu'en introduisant une simple récursion à gauche du symbole non terminal EXP (dans EXP → EXP + OPERANDE, au lieu de la récursion à gauche et à droite incluse dans EXP → EXP + EXP), on obtient un arbre d'analyse qui permet d'interpréter l'expression, par un parcours en profondeur de l'arbre, comme (1+2)+3.

   En fait, toute grammaire qui est récursive à gauche et à droite pour le même symbole non terminal est ambiguë. En remplaçant la double récursion de EXP dans les 4 premières productions de G1, par une simple récursion (à gauche ou à droite), on obtient une grammaire équivalente qui n'est plus ambiguë.
4) Vérifier que si la grammaire G1 avait été modifiée en introduisant une récursion à

droite de EXP plutôt qu'à gauche, l'arbre d'analyse (unique) obtenu pour 1+2+3 permettrait d'interpréter l'expression comme 1+(2+3).

5) Quel est l'arbre d'analyse pour l'expression 1+2*3 avec la grammaire G2 (celle de la question 3) ? Qu'en est-il pour 2*3+5 ?
6) Considérons enfin la grammaire G4 suivante :
   EXP → EXP + PROD | EXP - PROD | PROD
   PROD → PROD * FACT | PROD / FACT | FACT
   FACT → (EXP) | entier

   Quel est l'unique arbre d'analyse correspondant à 1+2*3 ? Remarquer que 1+2*3 peut maintenant être interprété comme 1+(2*3) et que 2*3+5 peut toujours être interprété comme (2*3)+5.

<u>En résumé</u> :

- L'association à gauche des opérateurs, qui permet d'interpréter par exemple 1+2+3 (respectivement 1-2-3, 1*2*3 et 1/2/3) comme (1+2)+3 (respectivement (1-2)-3, (1*2)*3 et (1/2)/3), a été obtenue par la récursion à gauche des symboles non terminaux EXP et PROD.
- Les priorités relatives des opérateurs, qui permettent d'interpréter par exemple 10/(1+2*3-4/5) comme 10/(1+(2*3))-(4/5) ont été obtenues en hiérarchisant la grammaire de telle sorte qu'une expression arithmétique soit définie comme une simple somme ou différence de produits, alors qu'un produit est défini comme une multiplication ou une division de facteurs, qui sont eux-mêmes des expressions entre parenthèses ou des entiers. Ainsi les opérateurs de multiplication et de division pourront être interprétés comme étant plus prioritaires que les opérateurs d'addition et de soustraction. De même les parenthèses permettent d'obtenir une sous-expression plus "prioritaire" encore que les opérateurs * et /, et donc que + et -. Vous pouvez ainsi vérifier que l'arbre d'analyse obtenu pour 10/(1+(2*3))-(4/5) réduit aux symboles terminaux correspond bien à l'interprétation attendue de l'expression.

Exercice 4

On considère la grammaire suivante :
INST → si c alors INST | si c alors INST sinon INST | a | b

1) Trouvez un mot de ce langage qui admet deux arbres d'analyse distincts. Que pouvez-vous en conclure ?
2) Trouvez une grammaire équivalente non ambiguë.

Exercice 5

1) Décrire une grammaire non contextuelle générant le langage suivant : $\{ a^n b^n c^p \ / \ n,p \in \mathbb{N} \}$ et donner un arbre d'analyse pour le mot aabbc.
2) Donner un automate à pile décrivant ce langage et vérifier son fonctionnement lors de la reconnaissance du mot aabbc.
3) La grammaire est-elle LL(1) ?

Exercice 6

1) Décrire une grammaire non contextuelle générant le langage suivant : { uv / u ∈

{a,b}* et v miroir de u }. Par exemple, le miroir de aabab est babaa, donc aababbabaa est un mot du langage. Donner un arbre d'analyse pour baab.

2) Donner un automate à pile décrivant ce langage et vérifier son fonctionnement lors de la reconnaissance du mot baab.
3) La grammaire est-elle LL(1) ?

Exercice 7

Soit la grammaire non contextuelle G1 suivante :
S → (L) | a
L → L-S | S

1) Pourquoi cette grammaire n'est-elle pas LL(1) ?
2) Trouver une grammaire G2 équivalente à G1 qui ne soit pas récursive à gauche.
3) G2 est-elle LL(1) ?

Exercice 8

Soit la grammaire non contextuelle suivante :
A → abB | ε
B → Aaa | b
Montrer que cette grammaire n'est pas LL(1).

Exercice 9

Soit la grammaire non contextuelle suivante :
S → AB
A → aAb | 0
B → aBbb | 1

1) Quel est le langage généré par cette grammaire ?
2) Montrer que la grammaire est LL(1).
3) Donner un automate à pile décrivant ce langage.
4) Donner le programme Java d'un analyseur récursif LL(1) de cette grammaire.
5) Donner le programme Java d'un analyseur non récursif LL(1) de cette grammaire.
6) Donner le programme Java d'un traducteur de cette grammaire qui transforme chaque a en b et chaque b en a.

Exercice 10

Soit la grammaire non contextuelle suivante :
A → abB | ε
B → Aaa | b
Cette grammaire n'est pas LL(1) (cf. exercice 8).
Est-il possible de construire de manière automatique un analyseur de cette grammaire ?