

Université Paris 7 — IUP informatique 1<sup>ère</sup> année  
Examen de compilation — 25 janvier 2000  
Durée : 2h

*La notation tiendra compte de la clarté des explications et des justifications fournies.*

**Exercice** Écrire un programme en utilisant `lex`, `yacc` et le langage C pour tester si une chaîne lue sur l'entrée standard est une expression régulière de lex syntaxiquement correcte qui n'utilise que des lettres (majuscules et minuscules), ainsi que la concaténation (notée par simple juxtaposition), l'étoile `*`, l'union `|` et les parenthèses.

Le programme devra par exemple accepter l'expression `(ab|ab)*`. Il devra rejeter `ab|ab)*` (expression syntaxiquement incorrecte), ainsi que `(ab|ab)+` qui utilise un opérateur autre que ceux auxquels on s'intéresse.

L'analyseur écrit en `yacc` ne devra comporter aucun conflit réduction-réduction. Votre analyseur comporte-t-il des conflits empilement-réduction? Comment et dans quel ordre doit-on compiler les fichiers?

**Problème** On considère la grammaire  $G$  suivante :

$$P \longrightarrow B\$ \quad (1)$$

$$B \longrightarrow \{ LI \} \quad (2)$$

$$LI \longrightarrow LI ; I \quad (3)$$

$$LI \longrightarrow I \quad (4)$$

$$I \longrightarrow B \quad (5)$$

$$I \longrightarrow V = E \quad (6)$$

$$V \longrightarrow \mathbf{id} \quad (7)$$

$$V \longrightarrow \mathbf{id} [ E ] \quad (8)$$

$$E \longrightarrow V \quad (9)$$

$$E \longrightarrow \mathbf{nb} \quad (10)$$

Dans cette grammaire, `$`, `}`, `{`, `;`, `=`, `]`, `[`, `nb` et `id` sont des terminaux.

1. Décrire de façon informelle le langage engendré par  $G$ .
2. La grammaire  $G$  est-elle ambiguë? Justifier précisément la réponse.
3. Donner l'arbre de dérivation de la chaîne `{ID[nb]=ID}$`.
4. Montrer que  $G$  n'est pas LL(1).
5. Calculer pour chaque symbole non-terminal de  $G$  les ensembles Premier et Suivant correspondants.
6. Trouver une grammaire  $G_1$  qui engendre le même langage que  $G$  mais qui est LL(1). On demande une justification précise du fait que  $G_1$  est LL(1).
7. Préciser pour chaque non-terminal de  $G_1$  quelle règle est appliquée par l'analyseur LL(1) en fonction du symbole d'avance.
8. Calculer l'automate des items LR(0) de  $G$ .
9. La grammaire  $G$  est-elle LR(0)?
10. La grammaire  $G$  est-elle SLR(1)?
11. Simuler le comportement d'un analyseur syntaxique montant (en précisant à chaque étape l'entrée, la pile des symboles, l'action choisie) sur l'entrée `{ID[nb]=ID}$`.
12. Écrire un source `yacc` permettant de
  - compter le nombre total d'instructions dans le programme (les instructions étant séparées par le symbole `;`).
  - compter et afficher la profondeur de chaque bloc entre accolades (le bloc le plus externe est de profondeur 0, et les blocs à l'intérieur d'un bloc de profondeur  $i$  sont de profondeur  $i + 1$ );

Les attributs utilisés sont-ils synthétisés? hérités?