UNIVERSITÉ BORDEAUX I — U.F.R. Mathématiques et Informatique
Année 1991-1992 — Session de Février 1992

Maîtrise d'Informatique
Module Compilation 1
(Durée: 3 heures)

Documents autorisés: notes de cours et de T.D.
Autres documents interdits.

Les trois exercices sont indépendants.
Barême envisagé: exercice 1: 5 points, exercice 2: 7 points, exercice 3: 8 points.

Exercice 1

Voici la spécification d'un analyseur lexical, dans le format reconnu par `lex`:

```
%%
[A-Z][a-z]*     printf ("IDENT: %s\n", yytext);
\'[a-z]*\'      printf ("CHAINE: %s\n", yytext);
:=              printf ("AFFECTATION\n");
\;              printf ("SEPARATION\n");
.               ; /* ignorer le reste */
%%
```

On suppose cette spécification compilée, et on appelle `lexico` l'analyseur obtenu. Lorsque le texte fourni en entrée est:

```
Alpha := 'alpha'; Beta := 'beta';
```

voici le résultat de l'exécution de `lexico`:

```
IDENT: Alpha
AFFECTATION
CHAINE: 'alpha'
SEPARATION
IDENT: Beta
AFFECTATION
CHAINE: 'beta'
SEPARATION
```

1. Donner le résultat de l'exécution de `lexico` lorsque le texte fourni en entrée est:

   ```
   Alpha := 'Alpha'; Beta := 'Beta';
   ```

2. Même question lorsque le texte fourni en entrée est:

   ```
   ALPHA := 'alpha'; BETA := 'beta';
   ```

3. Dans la spécification, on remplace la ligne:

   ```
   \'[a-z]*\'      printf ("CHAINE: %s\n", yytext);
   ```

   par:

   ```
   \'.*\'          printf ("CHAINE: %s\n", yytext);
   ```

   Quel est maintenant le résultat de l'exécution lorsque le texte fourni en entrée est celui du préambule, à savoir:

   ```
   Alpha := 'alpha'; Beta := 'beta';          ?
   ```

Exercice 2

On considère la grammmaire G des expressions postfixées:

$E \to EE+ \mid EE- \mid EE* \mid \texttt{id}$

où `id` désigne n'importe lequel des symboles terminaux $a, b, c, \ldots$

1. Ecrire la grammaire G' obtenue à partir de G en appliquant l'algorithme d'élimination de la récursivité gauche.
2. Donner les arbres de dérivation, dans G puis dans G', des expressions suivantes:

   $$ab+c* \ , \quad abc+* \ , \quad abc+ab*-*$$

3. Montrer que G' n'est pas LL(1).

Exercice 3

On introduit dans les expressions l'opérateur unaire $*$ pour indiquer, comme en langage C, une indirection; on s'intéresse donc à la grammaire simplifiée G suivante:

$E \to E+T \mid T$
$T \to *T \mid F$
$F \to (E) \mid \texttt{id} \mid \texttt{num}$

où `id` désigne n'importe lequel des symboles terminaux $a, b, \ldots$ et `num` une constante numérique.

1. Donner l'arbre de dérivation de l'expression:

   $$*a+**b+*(a+1)$$

2. Construire l'automate déterministe (dont les états sont des ensembles de 0-items), associé à un analyseur ascendant pour G.
3. Montrer que G est LR(0).
4. Construire la table d'analyse ascendante de G; appliquer l'analyseur à l'expression de la première question, et retrouver ainsi son arbre de dérivation.