Licence ST Informatique – 2005/2006

ASC–Examen 1ère session

jeudi 10 juin 2004

Durée 3h. Documents autorisés

---

1 URLs

On se propose d'étudier dans cet exercice une syntaxe simplifiée d'URLs. On se donne la grammaire G_1 suivante :
$G_1 = (\{\texttt{id}\ ,\ \texttt{/}\ ,\ \ \texttt{:}\}, \{\text{URL}, \text{SITE}, \text{PROTOCOLE}, \text{CHEMIN}\}, \text{URL}, \mathcal{P})$ avec $\mathcal{P} =$

$$\left\{\begin{array}{lcl} \text{URL} & \longrightarrow & \text{SITE} \mid \text{PROTOCOLE} \\ \text{SITE} & \longrightarrow & \texttt{id}\ \text{CHEMIN} \\ \text{PROTOCOLE} & \longrightarrow & \texttt{id}\ \texttt{:}\ \text{SITE} \\ \text{CHEMIN} & \longrightarrow & \text{CHEMIN}\ \texttt{/}\ \texttt{id} \mid \varepsilon \end{array}\right\}$$

Question 1.1 : Cette grammaire est-elle LL(1), LR(0), SLR(1), ambiguë ? Justifiez chaque réponse.

Question 1.2 : Donnez une grammaire LL(1) équivalente à G_1. Justifiez votre réponse en construisant la table d'analyse LL(1).

Question 1.3 : Détaillez le fonctionnement de l'analyseur itératif LL(1) correspondant sur l'entrée `id:id/id/`

2 Expressions de contours de figures

Dans certains systèmes graphiques, comme des tables traçantes, des contours 2D sont décrits par des opérations sur des vecteurs correspondant aux quatre déplacements élémentaires *nord*, *sud*, *est* et *ouest* ; ces vecteurs seront représentés respectivement par les lettres n, s, e et o. Ainsi, un mot sur l'alphabet

$X = \{n, s, e, o\}$ correspond à un contour qui peut être tracé sur la grille 2D, à partir d'un point, appelé *origine*. La figure 1 montre un tel contour, décrit par le mot *ooseooo* et tracé à partir du point origine $(3, 2)$.

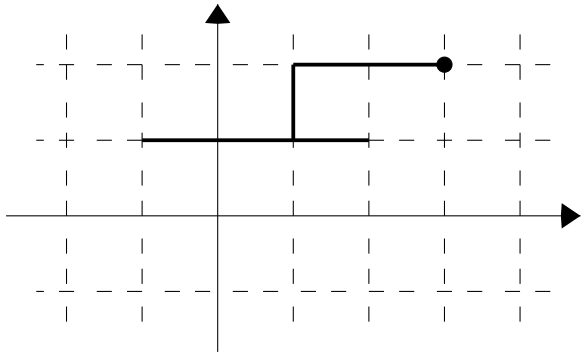

FIG. 1 – La figure d'origine $(3, 2)$, associée au mot *ooseooo*.

Pour préciser les choses, on appelle *point* tout couple d'entiers ; on appelle *mot de contour* tout mot de X^* et on appelle *tracé* la donnée d'un couple $< u, p >$, où u est un mot de contour et p est le point origine du tracé. Ainsi, la figure 1 représente-t-elle le tracé $<$ *ooseooo*$, (3, 2) >$. Enfin, on appelle *extrémité* d'un tracé $< u, p >$ le point atteint en traçant le contour u à partir de l'origine p. Par exemple, l'extrémité du tracé $<$ *ooseooo*$, (3, 2) >$, donné figure 1, est le point $(-1, 1)$ et l'extrémité d'un tracé $< \varepsilon, p >$ (de mot de contour vide) pour un point origine p quelconque est ce point p.

Question 2.1 : Calculez l'extrémité du tracé $<$ *nnnesesos*$, (1, 1) >$.

Question 2.2 : Donnez une grammaire algébrique qui engendre le langage de tous les mots de contour u pour lesquels l'extremité de tout tracé $< u, (0, 0) >$ a une abscisse impaire[1]. L'alphabet terminal est bien-sûr l'alphabet $X = \{n, s, e, o\}$.

Question 2.3 : Donnez un automate à pile pour le langage de tous les mots de contour pour lesquels l'extremité de tout tracé associé a même ordonnée que l'origine.

[1] L'abscisse d'un point est sa première coordonnée et l'ordonnée d'un point est sa deuxième coordonnée.

On décide d'enrichir nos langages de mots de contour à l'aide d'un opérateur binaire associatif de superposition, noté $+$, et dont l'effet est de tracer les contours, opérandes du $+$, à partir de la même origine. On précise que cet opérateur est de plus faible priorité que la concaténation des mots de contour, et que les deux opérations, concaténation et superposition, sont associatives gauche. On définit enfin l'extrémité d'une superposition comme étant l'extrémité du dernier opérande de la superposition. Par exemple, le tracé $< ooseooo + sesso, (3, 2) >$ est donné figure 2. Son extrémité est le point $(3, -1)$.

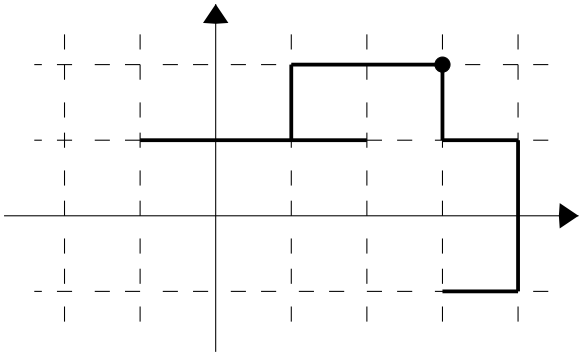

FIG. 2 – La figure d'origine $(3, 2)$, associée à l'expression de contour $ooseooo + sesso$.

On définit syntaxiquement les expressions de contour à l'aide de la grammaire G suivante, d'alphabet terminal $T = \{n, s, e, o, (,), +\}$, d'axiome E' et de règles :

$$E' \longrightarrow E$$
$$E \longrightarrow E + E \mid EE \mid (E) \mid n \mid s \mid e \mid o$$

et on étend la notion de tracé comme étant maintenant un couple expression de contour – point origine.

Question 2.4 : Donnez une grammaire, sous forme normale de Chomsky, équivalente à la grammaire G.

Question 2.5 : Donnez un arbre de dérivation, dans la grammaire G, pour l'expression de contour $(se + on)e$.

Question 2.6 : La grammaire G est-elle ambiguë ? Justifiez.

Question 2.7 : Construisez la table action $SLR(1)$ pour la grammaire G

Question 2.8 : En utilisant la priorité respective des opérateurs et leur associativité, résolvez les conflits apparaissant dans la table action de la question précédente (des conflits y apparaissent pour 2 états).

Question 2.9 : Donnez une grammaire $LL(1)$ équivalente à la grammaire G. Justifiez du caractère $LL(1)$ en construisant les ensembles *premiers* et *suivants*, ainsi que la table d'analyse LL(1)..

Pour enrichir votre grammaire d'actions sémantiques qui permettent de calculer les extrémités des tracés, on dispose du type `Point` suivant :

```
public class Point {
  private int x , y ;
  public Point(int x , int y) {this.x = x ; this.y = y ;}
  public int getX() {return x ;}
  public int getY() {return y ;}
}
```

Question 2.10 : Définissez les attributs nécessaires pour chaque variable de votre grammaire en précisant pour chacun s'il s'agit d'attribut synthétisé ou hérité.

Question 2.11 : Décorez votre grammaire d'actions sémantiques permettant de calculer, au cours de l'analyse syntaxique d'une expression de contour, l'extrémité d'un tracé correspondant, d'origine $(0, 0)$.

Question 2.12 : En supposant disposer d'un analyseur lexical à votre convenance et de la méthode `consommer`, écrivez un analyseur récursif descendant, qui étant donnée une expression de figure, affiche sur la sortie standard si l'expression est syntaxiquement correcte et dans ce cas affiche les coordonnées de l'extrémité d'une figure correspondante d'origine $(0, 0)$