Université Paris 7-Denis Diderot
L3
d'Informatique

Examen du 18 juin 2008
— Durée 3 h —
tous documents autorisés

Avertissement : Il est demandé de faire des preuves rigoureuses. Une explication en français correct, claire et précise de ce qui est à faire et de comment le faire pour mener à bien une preuve sera appréciée. On peut évidemment sauter une question en indiquant qu'on admet le résultat.

Exercice 1

Ici A désigne l'alphabet $\{a, b\}$. Pour $f \in A^*$, si $f = x_1x_2 \dots x_k$ où $x_i \in A$, $\tilde{f}$ désigne le mot miroir de f, soit : $\tilde{f} = x_k \dots x_2x_1$, et pour $L \subset A^*$, $\tilde{L} = \{\tilde{f} \mid f \in L\}$.

On rappelle que le langage de Lukaziewicz Ł est l'unique solution de l'équation :

$$v \quad = \quad avv + b$$

Question 0 : Dire pourquoi cette équation possède une unique solution. Vérifier que Ł est solution.

On définit les langages suivants : $D_1 = a.$Ł et $\hat{D}_1 = D_1 \bigcup \tilde{D}_1$.

Question 1 : Construire un automate à pile déterministe qui reconnaît $\hat{D}_1$ par pile vide et états d'acceptation.

Question 2 : En utilisant la construction des triples vue en cours, en déduire une grammaire non-ambiguë G engendrant $\hat{D}_1$. Donner une grammaire non-ambiguë H engendrant $\hat{D}_1{}^*$.

Exercice 2

On considère la grammaire algébrique $G =< A, V, P >$ où $A = \{a, b\}$, $V = \{S\}$ et P est défini par :

$$\begin{array}{lcl} S & \longrightarrow & aSbS \\ S & \longrightarrow & bSaS \\ S & \longrightarrow & 1 \end{array}$$

Question 0 : Donner un arbre de dérivation selon G pour le mot $aabbba$.

Question 1 : Démontrer que cette grammaire est non-ambiguë.

Question 2 : Construire un automate fini déterministe et complet ayant un ensemble d'états Q et une fonction de transition δ qui reconnaît $K = aa^*bb^*$.

Soit $W = Q \times V \times Q$. On définit une nouvelle grammaire $H =< A, W, P' >$ où P' est défini par :

$$\begin{array}{lcl} (q, S, q') & \longrightarrow & a(p, S, p')b(r, S, q') \iff \delta(q, a) = p \text{ et } \delta(p', b) = r \\ (q, S, q') & \longrightarrow & b(p, S, p')a(r, S, q') \iff \delta(q, b) = p \text{ et } \delta(p', a) = r \\ (q, S, q') & \longrightarrow & 1 \iff q = q' \end{array}$$

Question 3 : Ecrire explicitement toutes les règles de H. Réduire cette grammaire.

Question 4 : Montrer que cette grammaire est non-ambiguë.

Problème

Soit $G =< A, V, P >$ une grammaire algébrique pré-réduite.

Une variable $v \in V$ est dite *récursive* si il existe deux mots f et g tels que $fg \in A^+$ (*i.e.* f et g sont tous deux sur l'alphabet A et ne sont pas tous les deux simultanément vides) et $fvg \in \hat{L}_G(v)$.

Question 1 : Vérifier que toute variable récursive engendre un langage infini.

On définit sur V une relation R par :
$(v, v') \in R \Longleftrightarrow \exists f, g$ tels que $fg \in (A \cup V)^+$ et $v \longrightarrow fv'g \in P$,
et on note pour tout $U \subset V$:
$R(U) = \{v' \in V \mid \exists v \in U, (v, v') \in R\}$.

Question 2 : Montrer que l'application $\hat{R} : \mathcal{P}(V) \longrightarrow \mathcal{P}(V)$ définie par : $\hat{R}(U) = U \bigcup R(U)$ possède pour tout $v \in V$ un plus petit point fixe qui contient $\{v\}$, que l'on notera dans la suite $R^*(v)$.

Question 3 : Montrer que l'on peut calculer ce plus petit point fixe, et écrire un programme qui utilise ce calcul pour obtenir *in fine* pour chaque variable v l'ensemble $R^*(v)$.

Question 4 : En supposant que la grammaire est stricte (*i.e.* si $v \longrightarrow m \in P$, alors $m = 1$ ou bien m contient au moins une lettre de A), démontrer que v est une variable récursive si et seulement si $v \in R(R^*(v))$. Démontrer qu'il en est de même si la grammaire est propre.

Question 5 : Vérifier que la relation R' sur V définie par : $(v, v') \in R'$ si et seulement si $v = v'$ ou il existe deux mots f et g tels que $fg \in (A \cup V)^+$ et $fv'g \in \hat{L}_G(v)$ est une relation de préordre.

Question 6 : Expliquer comment calculer l'équivalence canonique associée à ce préordre, et l'ordre sur l'ensemble quotient.

Question 7 : Donner un algorithme permettant de calculer l'ensemble des variables engendrant un langage infini d'une grammaire stricte ou propre.

Question 8 : Que faut-il changer pour résoudre ce problème pour une grammaire pré-réduite quelconque ?