Ensimag 1ère année Le 12 mai 2011

Théorie des Langages 2

Durée : 3h.
Documents : tous documents autorisés.

Exercice 1 (6 points)

▷ Question 1 (3 points)

Soit $f : \mathbb{N} \to \mathbb{N}$ la fonction définie comme suit :

$$f(x) = \begin{cases} i & \text{si } x = 2 * i \\ i & \text{si } x = (2 * i) + 1 \end{cases}$$

Donner une machine de Turing qui réalise f. On utilisera un codage unaire pour l'entrée x. On pourra utiliser une machine à un ou plusieurs rubans. Expliquer la solution proposée.

▷ Question 2 (3 points)

On rappelle qu'un ensemble est récursivement énumérable si et seulement si il est le domaine d'une fonction calculable. Les ensembles suivants sont-ils récursivement énumérables ? Justifiez votre réponse.

1. $L_1 = \{a^{2^n} \mid n \in \mathbb{N}\}$
2. $L_2 = \{x \in \mathbb{N} \mid MTU(x, x) \text{ ne s'arrête pas } \}$

Exercice 2 (14 points)

Les grammaires étendues sont une notation plus compacte pour les grammaires hors-contexte. Elles permettent de décrire dans les parties droites de règles des parties optionnelles ou itérées.

Dans la définition classique des grammaires une partie droite de règle est une chaîne de l'ensemble $L_D = (V_T \cup V_N)^*$. Dans une grammaire étendue le langage des parties droites de règles est défini par les constructions suivantes :

(1) $\epsilon \in L_D$
(2) si $X \in (V_T \cup V_N)$ alors $X \in L_D$
(3) si $w \in L_D$ alors $[w] \in L_D$ — w : partie optionnelle
(4) si $w \in L_D$ alors $\{w\} \in L_D$ — w : partie itérée
(5) si $w_1 \in L_D$ et $w_2 \in L_D$ alors $w_1 w_2 \in L_D$

Les symboles [,], { et } sont des méta-symboles, i.e. des symboles appartenant au langage de description des règles.

On peut construire de manière systématique une grammaire " classique " à partir d'une grammaire étendue en appliquant (récursivement) les transformations suivantes :

1. Remplacer dans les parties droites de règles chaque sous-terme de la forme $[w]$ par un nouveau symbole non-terminal X et ajouter les deux règles :

$$X \rightarrow \epsilon \qquad X \rightarrow w$$

2. Remplacer dans les parties droites de règles chaque sous-terme $\{w\}$ par un nouveau symbole Y et ajouter les deux règles :

$$Y \rightarrow \epsilon \qquad Y \rightarrow wY$$

On notera $\mathcal{T}(G)$ la grammaire classique associée à la grammaire étendue G. On dira qu'une grammaire étendue G est LL(1) si et seulement si sa transformée $\mathcal{T}(G)$ est LL(1).

▷ **Question 1 (2 points).**

Soit la grammaire étendue G suivante :

(1)	SI	→	I {I}
(2)	I	→	affect
(3)	I	→	**if** exp **then** SI [**else** SI] **end** ;
(4)	affect	→	**idf** := exp ;

Le non-terminal SI est l'axiome et les éléments du vocabulaire terminal sont notés en gras. Le non-terminal exp n'est pas défini ici. Donner la grammaire transformée $\mathcal{T}(G)$ associée à G.

▷ **Question 2 (2 points).**

Appliquer les calculs LL(1) sur la grammaire $\mathcal{T}(G)$ et proposer une grammaire équivalente LL(1), si besoin est. Soit G' la grammaire LL(1) obtenue.

▷ **Question 3 (3 points).**

Certaines normes de développement imposent des règles de programmation permettant de maîtriser la complexité du code. Une règle classique est de limiter la profondeur d'imbrication des conditionnelles. Par exemple la profondeur maximale d'imbrication des conditionnelles pour l'exemple suivant est 2 :

```
if e1 then
   if e2 then x:=3 ; end ;
else x:=1 ; end ;
if e3 then  x:=4 ; else x:=0 ; end ;
```

Ajouter un calcul d'attributs sur la grammaire G' de la question 2 permettant de calculer la profondeur maximale d'imbrication des conditionnelles dans une suite d'instructions (non-terminal SI). On expliquera précisément la signification des attributs utilisés.

▷ **Question 4 (4 points).**

A partir de la grammaire G' écrire un analyseur LL(1) qui reconnait une instruction (non-terminal I) et calcule le nombre maximal d'imbrication de conditionnelles. On supposera écrites les procédures d'analyse SI, exp et affect. On écrira toute autre procédure d'analyse nécessaire en fonction de la grammaire G' proposée.

▷ **Question 5 (3 points).**

On s'intéresse maintenant à étendre le principe de l'analyse LL(1) directement sur les grammaires étendues. Pour cela nous devons étendre les définitions $Eps(w)$, $Premier(w)$ et les conditions LL(1). On peut définir le prédicat $Eps(w)$ qui calcule si il existe une dérivation de m vers ϵ de la manière suivante :

$$
\begin{array}{lllll}
(1) & Eps(\epsilon) & = & true & \\
(2) & Eps(xw) & = & false & x \in V_T \\
(3) & Eps(Aw) & = & (\vee_{A \to w_i} Eps(w_i)) \wedge Eps(w) & A \in V_N \\
(4) & Eps(\{r\}w) & = & Eps(w) & \\
(5) & Eps([r]w) & = & Eps(w) &
\end{array}
$$

avec r et w des mots de L_D.

1. Étendre le calcul vu en cours des ensembles $Premier(w)$ pour les deux nouvelles constructions $[r]w$ et $\{r\}w$).
2. Pour les grammaires étendues l'ensemble des directeurs d'une règle se calcule de la même manière que pour les grammaires classiques. Par contre des conditions supplémentaires sont introduites par les règles contenant des occurrences de la forme $[w]$ ou $\{w\}$.

   Soit une règle de la forme $A \to w_1[w_2]w_3$. Donner des conditions, portant sur les chaînes w_2, w_3 et A, qui garantissent le caractère LL(1) de la grammaire étendue.
3. Même question pour une règle de la forme $A \to w_1\{w_2\}w_3$.