Ensimag 1ière année | Le 17 mai 2010

Théorie des Langages 2

Durée : 3 heures.
Documents : tout document autorisé.

Pensez au lecteur. Commentez, utilisez des noms explicites et ne renommez pas les noms employés dans l'énoncé.

Exercice (6 points)

Soient $f : \mathbb{N} \to \mathbb{N}$ et $g : \mathbb{N} \to \mathbb{N}$ deux fonctions calculables quelconque. On se propose dans cet exercice de montrer que le problème "$\forall x \in \mathbb{N}, f(x) = g(x)$" n'est pas décidable.

Considérons les fonctions $f_i : \mathbb{N} \to \mathbb{N}$ et $g_{i,e} : \mathbb{N} \to \mathbb{N}$ définies ci-dessous, avec i l'indice d'une machine de Turing et e un entier naturel quelconque.

$$f_i(x) = \begin{cases} 1 & \text{si } MTU(i,x) \text{ s'arrête} \\ \text{indéfinie} & \text{sinon} \end{cases}$$

$$g_{i,e}(x) = \begin{cases} f_i(x) & \text{si } x \neq e \\ 1 & \text{si } x = e \end{cases}$$

▷ **Question 1 (2 points)**

Montrer que "$MTU(i,e)$ s'arrête" si et seulement si "$\forall x \in \mathbb{N}, f_i(x) = g_{i,e}(x)$".

▷ **Question 2 (2 points)**

Justifier brièvement que les fonctions f_i et $g_{i,e}$ sont calculables.

▷ **Question 3 (2 points)**

Déduire l'indécidabilité du problème "$\forall x \in \mathbb{N}, f(x) = g(x)$", pour f et g calculables.

PROBLEME (14 points)

On s'intéresse à un langage de commandes, défini par la grammaire G_1 suivante (l'axiome est le non-terminal programme) :

programme	$\rightarrow$	**begin** commande **end**		
commande	$\rightarrow$	sequencement	| affectation	
sequencement	$\rightarrow$	commande ; affectation		
affection	$\rightarrow$	**idf** := exp		
exp	$\rightarrow$	(exp + exp)	| **idf**	| **num**

Le vocabulaire terminal est VT={ **begin**, **end**, **idf**, **num**, **;**, **:=**, **(**, **)**, **+** }.

▷ **Question 4 (1 point)**
Justifier en quoi cette grammaire n'est pas LL(1).

▷ **Question 5 (3 points)**
Donner une grammaire LL(1) pour le langage L(G_1). On fera bien attention à préserver le langage. On prouvera le caractère LL(1) de la grammaire proposée (on pourra numéroter les règles pour les calculs de directeur).

▷ **Question 6 (3 points)**
On veut étendre la notation pour permettre l'affectation simultanée de plusieurs variables. Exemples :

1. x, y, z := a, (a+1), 0
2. x, y := y, x

Proposer une nouvelle définition du non-terminal **affectation**, sous forme LL(1), prenant en compte cette extension. La grammaire proposée devra garantir qu'il y a autant d'éléments de chaque côté du signe :=. On étend le vocabulaire terminal à l'aide du symbole **,**.

▷ **Question 7 (2 points)**
On ajoute la contrainte suivante pour l'affectation simultanée : **les identificateurs apparaissant en partie gauche du signe := doivent être distincts deux à deux.**

Ajouter à la grammaire précédente des attributs permettant de vérifier cette contrainte. Les attributs manipulés représenteront des ensembles de noms et on pourra utiliser les opérations classiques sur les ensembles ($\cup, \cap, \in$ $, \notin, \ldots$).

▷ **Question 8 (3 points)**
Ecrire un analyseur LL(1) qui reconnaît les affectations simultanées et vérifie la contrainte de la question précédente. On utilisera la fonction **lire_mot return token** avec **token** = $VT \cup \{\$\}$, où *$* représente le marqueur de fin de texte, ainsi que la fonction **nom_idf return string** qui renvoie le nom d'un identificateur lorsque le mot reconnu est de catégorie **idf**. On n'écrira pas la procédure **exp**, qui analyse les expressions.

On supposera aussi disposer d'un type abstrait **Ens** qui décrit les ensembles de string ainsi que les opérations classiques sur ces ensembles.

▷ **Question 9 (2 points)**

Soit `x1, ..., xn := e1, ..., en` une affectation simultanée. Cette commande peut être traduite en une séquence d'affectation simple de la manière suivante :

```
a1 := x1 ; ... an := xn ; x1 := e1'; ...  xn := en'
```

avec `ai` des identificateurs n'apparaissant pas dans le programme. L'expression `ei'` désigne l'expression obtenue à partir de `ei` en remplaçant les occurrences des identificateurs `xi` par `ai` (`ei' = ei [x1← a1] ... [xn← an]`).

Appliquer cette traduction aux 2 exemples de la question 6.

On s'intéresse au cas où les identificateurs `x1, ..., xn` n'apparaissent pas dans les expressions `e1, ..., en`. Expliquer en quoi cette hypothèse permet de simplifier la traduction ci-dessus. Donner un calcul d'attributs sur la grammaire de la question 6 permettant de déterminer si la condition ci-dessus est vérifiée.