Grenoble INP - Ensimag $1^{\text{ière}}$ année mai 2013

Théorie des Langages 2

Durée : 3 heures.
Documents : notes de cours et poly autorisés, livres interdits.

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

Exercice (6 points)

Soit $<\cdot,\cdot>$ une fonction totale calculable bijective de $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$

Soit U une variante de machine de Turing universelle telle que $\mathsf{U}(<m,n>)$ calcule le résultat éventuel (dans $\mathbb{N}$) de la machine de Turing de code m sur l'entrée de code n

▷ **Question 1 (3 points)**

Chacune des 2 fonctions totales suivantes est-elle calculable ? Justifier la réponse.

Pour m, n fixés : $f_{<m,n>}(i) = 0$ si $\mathsf{U}(<m,n>)$ termine en au plus i pas d'exécution
$= 1$ sinon

$h(<m,n>) = 0$ si $\mathsf{U}(<m,n>)$ termine
$= 1$ sinon

▷ **Question 2 (3 points)**

Soit g la fonction partielle de $\mathbb{N} \to \mathbb{N}$ telle que

$\mathrm{Dom}(g) = \{m \mid \exists n : \mathsf{U}(<m,n>) \text{ termine}\}$

$\forall m \in \mathrm{Dom}(g) : g(m) = \mathrm{Min}\{\mathsf{U}(<m,n>) \mid \mathsf{U}(<m,n>) \text{termine}\}$

- Justifier que cette définition a un sens (i.e. que g est bien une fonction...)
- Soit le polynôme $P(X) = X^2 - 6X + 16$. Montrer que $\forall n \in \mathbb{N}, P(n) \in \mathbb{N}$
- Soit m le code d'une MT calculant P ; que vaut $g(m)$?
- En utilisant les réponses de la question 1, montrer que g n'est pas calculable.

Problème (14 points)

On s'intéresse à un langage d'instructions (non-terminal *inst*) constituées d'une suite d'affectations (non-terminaux *seq* et *aff*), défini par la grammaire G_1 suivante (l'axiome est le non-terminal *prog*) :

prog	$\to$	**begin** *inst* **end**		
inst	$\to$	*seq*	| *aff*	
seq	$\to$	*inst* ; *aff*		
aff	$\to$	**idf** := *exp*		
exp	$\to$	(*exp* + *exp*)	| **idf**	| **num**

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

▷ **Question 3 (1 point)**

Justifier en quoi cette grammaire n'est pas LL(1).

▷ **Question 4 (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 directeurs).

▷ **Question 5 (2 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 *aff*, 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 6 (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.** Par exemple on voudra refuser l'affectation multiple x, y, x := 1, 2, 3.

Ajouter à la grammaire précédente un calcul d'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 7 (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** $= V_T \cup \{\$\}$, 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 8 (3 points)**

On s'intéresse ici à «concaténer» des grammaires LL(1) définies sur le même vocabulaire terminal V_T. Soit $G_1 = (V_T, V_{N_1}, S_1, R_1)$ et $G_2 = (V_T, V_{N_2}, S_2, R_2)$ deux grammaires LL(1) avec $V_{N_1} \cap V_{N_2} = \emptyset$. On construit la grammaire G suivante : $(V_T, \{S\} \cup V_{N_1} \cup V_{N_2}, S, \{S \to S_1S_2\} \cup R_1 \cup R_2)$, avec S un nouveau symbole ($S \notin (V_{N_1} \cup V_{N_2})$). On rappelle qu'on définit l'ensemble des directeurs d'une règle par (S étant l'axiome) :

$$Dir(A \to w) = \{\mathbf{x} \in (V_T \cup \{\$\}) \mid \exists w_1, w_2, w_3 \ : \ S\$ \Rightarrow^* w_1Aw_2 \Rightarrow w_1ww_2 \Rightarrow^* w_1\mathbf{x}w_3\}$$

- Avec $R_1 = \{S_1 \to aS_1b \mid \varepsilon\}$ et $R_2 = \{S_2 \to cS_2 \mid b\}$ montrer que la «concaténation» de G_1 et G_2 est LL(1).
- Donner un contre-exemple montrant que G n'est pas toujours LL(1).
- Donner une condition suffisante, la moins restrictive possible, sur les ensembles *Dir*, *Prem* et *Suiv* déjà calculés pour G_1 et G_2 permettant de garantir que G est LL(1).
- (*Bonus*) vos conditions sont-elles nécessaires ?