Calculabilité et décidabilité : TD2

Licence Info 3 - Michele Pagani
31 janvier 2013

---

1 Le langage WHILE

WHILE a un seul type de données : le type $\mathbb{D}$ des arbres binaires, défini par la grammaire suivante.

$$\mathtt{d}, \mathtt{e} ::= \mathtt{nil} \mid (\mathtt{d}.\mathtt{e})$$

La syntaxe des programmes est donnée comme suit.

Expressions	E,F	::= X	(pour X variable)
		| d	(pour $\mathtt{d} \in \mathbb{D}$)
		| cons E F	
		| hd E F	
		| tl E F	
		| =? E F	
Commands	C,D	::= X := E	
		| C; D	
		| while E do {C}	
Programs	P	::= read X; C; write Y	

1.1 Expressions booléennes

On rappelle les abréviations vues en cours.

Valeurs booléennes	false	=	nil
	true	=	(nil.nil)
Conditionnels	if E then C	=	Z:= E; while Z do {Z := false; C};
	if E then C1 else C2	=	Z:= E; W:=true; while Z do {Z := false; W:= false; C1}; while W do {W := false; C2};

Exercice 1. Écrire les expressions correspondant aux opérateurs booléens suivants :

$$\neg \mathtt{X}, \quad \mathtt{X} \wedge \mathtt{Y}, \quad \mathtt{X} \vee \mathtt{Y}, \quad \mathtt{X} \to \mathtt{Y}, \quad \mathtt{X} \leftrightarrow \mathtt{Y}.$$

Exercice 2. Écrire un programme P reconnaissant les valeurs booléennes, c.à.d. P écrit soit true soit false sur Y selon que l'entrée lue sur X est une valeur booléenne ou pas.

1.2 Listes

La liste des n éléments $\mathtt{d}_1, \ldots, \mathtt{d}_n$ de $\mathbb{D}$ est représentée par l'arbre

$$(\mathtt{d}_1.(\mathtt{d}_2.(\cdots.(\mathtt{d}_{n-1}.(\mathtt{d}_n.\mathtt{nil})))))$$

Par commodité de lecture, on utilise la notation suivante vue en cours :

$$\underline{\mathtt{d}} = \begin{cases} (\underline{\mathtt{d}_1} \cdots \underline{\mathtt{d}_n}) & \text{si } \mathtt{d} = (\mathtt{d}_1.(\mathtt{d}_2.(\cdots.(\mathtt{d}_{n-1}.(\mathtt{d}_n.\mathtt{nil}))))), \\ \mathtt{d} & \text{sinon.} \end{cases}$$

Exercice 3. Écrire un programme `append` qui lit en entrée un couple de listes $(\underline{\mathtt{d}_1}.\underline{\mathtt{d}_2})$ et renvoie en sortie la concaténation de la liste $\underline{\mathtt{d}_2}$ à la liste $\underline{\mathtt{d}_1}$.

Exercice 4. Écrire un programme `pair` qui lit en entrée une liste $(\underline{\mathtt{d}_1} \cdots \underline{\mathtt{d}_n})$ et renvoie en sortie la liste des éléments en position paire $(\underline{\mathtt{d}_2} \cdots \underline{\mathtt{d}_{2(n/2)}})$.

Exercice 5. Écrire un programme `listepair` qui decide l'ensemble des listes contenant un nombre pair d'éléments, c.à.d. `listepair` écrit soit `true` soit `false` sur `Y` selon que l'entrée lue sur `X` est une liste $(\underline{\mathtt{d}_1} \cdots \underline{\mathtt{d}_n})$ avec n pair.

1.3 Entiers naturels

Pour chaque entier $n \in \mathbb{N}$ on définit le numéral $\underline{n} \in \mathbb{D}$ comme suit :

$$\underline{0} = \mathtt{nil} = (\,)$$
$$\underline{n+1} = (\mathtt{nil}.\underline{n}) = (\underbrace{\underline{\mathtt{nil}} \cdots \underline{\mathtt{nil}}}_{n+1 \text{ fois}})$$

Exercice 6. Écrire un programme `P` qui decide l'ensemble des numéraux, c.à.d. `P` écrit soit `true` soit `false` sur `Y` selon que l'entrée lue sur `X` a la forme $(\mathtt{nil} \cdots \mathtt{nil})$ ou pas.

Exercice 7. Écrire un programme `length` qui calcule la longueur d'une liste, c.à.d. `length` lit en entrée une liste $(\underline{\mathtt{d}_1} \cdots \underline{\mathtt{d}_n})$ et renvoie en sortie le numeral $\underline{n}$.

Exercice 8. Écrire les programmes qui calculent les fonctions suivantes (utiliser l'appel aux sous-procédures interne à un programme) :

1. l'addition ;
2. la soustraction (avec $x - y = 0$ si $x \leq y$) ;
3. la multiplication ;
4. la division entière.

Exercice 9. Écrire les programmes qui décident les propriétés suivantes (en renvoyant soit `true` soit `false`) :

1. n divise m ;
2. n est un nombre premier ;

Exercice 10 (Résumé)**.** Écrire un programme `eratosthene` qui met en œuvre le crible d'Ératosthène, c.à.d. `eratosthene` lit en entrée un numeral $\underline{n}$ et renvoie en sortie la liste $(\underline{\mathtt{d}_1} \cdots \underline{\mathtt{d}_k})$ de numéraux correspondant aux nombres premiers plus petits ou égaux à n.