ISIMA 1ère année

30 août 2005
Durée : 2 heures
Documents autorisés

PROGRAMMATION FONCTIONNELLE

Une *machine de Turing* est une machine virtuelle, dite machine à états. Elle dispose d'un *ruban* qui est un ensemble dénombrable d'emplacements. Cet ensemble est ordonné, conventionnellement de gauche à droite, ce qui signifie que le *premier emplacement* est celui qui est le plus à gauche. Chaque emplacement peut contenir un symbole. Dans le cas où l'emplacement est vide, il est équivalent de considérer qu'il contient un *symbole blanc* que l'on représentera conventionnellement avec l'identificateur `blanc`. Le ruban n'est pas borné à droite, mais il contient toujours un nombre fini de symboles non blancs. L'ensemble ordonné de tous les symboles placés entre le premier emplacement et le *dernier emplacement* (*i.e.* le plus à droite) contenant un symbole non blanc, bornes comprises, est le *mot* qui est écrit sur le ruban. Ce mot sera représenté par une liste de symboles.

La machine de Turing peut lire les symboles de ce mot, symbole par symbole, en déplaçant une *tête de lecture* d'un symbole à droite ou à gauche (si possible) et peut également écrire sur le ruban, en remplaçant le symbole lu par tout autre symbole. Son fonctionnement peut alors être décrit de la manière suivante : la machine démarre dans un *état initial*, avec un mot *m* sur le ruban, et avec sa tête de lecture positionnée sur le premier emplacement du ruban ; elle répète ensuite une succession de *transitions*, appelée *exécution* de la machine sur le mot *m* ; à chaque transition, si la machine est dans un état `e1` et si sa tête de lecture est positionnée sur un symbole `s1`, alors elle pourra éventuellement passer dans un état `e2`, écrire `s2` à la place de `s1`, et déplacer la tête de lecture à droite ou à gauche ; l'exécution de la machine se termine lorsque celle-ci atteint un état particulier unique, appelé *état accepteur*.

Une transition sera donc représentée par une liste de cinq symboles, appelée *quintuplet*, du type `(e1 s1 e2 s2 DG)`, où `e1` et `e2` sont des symboles représentant respectivement l'état de la machine avant la transition et après la transition, où `s1` et `s2` sont respectivement le symbole lu par la tête de lecture avant la transition et le symbole écrit par celle-ci après la transition, et où `DG` est soit le symbole `D`, soit le symbole `G`, indiquant que la tête de lecture se

déplace respectivement soit à droite, soit à gauche. Une machine de Turing sera finalement représentée par une liste de tels quintuplets.
L'ensemble des états de la machine est implicitement défini par la liste de quintuplets, ainsi que l'ensemble des symboles qui peuvent être lus sur le ruban. L'état initial et l'état accepteur seront conventionnellement représentés par des symboles particuliers d'identificateurs respectifs `initial` et `accepteur`.
Le mot initialement présent sur le ruban sera représenté par une liste de symboles. Cette liste définit donc implicitement l'ensemble des symboles initialement présents sur le ruban.
On appelle *configuration* d'une machine de Turing une liste de trois éléments, le premier étant le symbole représentant l'état courant de la machine, le deuxième étant une liste de symboles représentant le mot compris entre le début du ruban et la tête de lecture non comprise (donc si celle-ci est en début de ruban, cette liste est vide), et le troisième étant une liste de symboles représentant le mot compris entre la tête de lecture comprise et le dernier symbole non blanc (par convention, si la tête de lecture se trouve après le dernier symbole non blanc, alors cette liste est vide).
L'exécution d'une machine de Turing sur un mot sera représentée, soit par la liste des configurations atteintes jusqu'à une configuration finale où la machine est dans l'état `accepteur`, soit par la liste des configurations atteintes jusqu'à une configuration qui réapparaît une deuxième fois (dans le cas où l'exécution boucle).

Le but de ce problème est d'écrire une fonction `executer` ayant comme paramètres une machine de Turing, `MT`, et un mot `mot` et telle que l'évaluation de l'expression `(executer MT mot)` retourne la représentation de l'exécution de `MT` sur `mot`.

1°) Ecrire une fonction `triplet` ayant comme arguments une liste de quintuplets représentant une machine de Turing, `MT`, un symbole représentant un état, `e1`, et un symbole représentant un symbole du ruban, `s1`, et telle que l'évaluation de l'expression `(triplet MT e1 s1)` retourne soit un triplet du type `(e2 s2 DG)` si le quintuplet `(e1 s1 e2 s2 DG)` existe dans `MT`, soit la liste vide sinon.
Solution :

```
(define triplet (lambda (MT e1 s1)
   (if MT
      (if (and (equal? e1 (caar MT)) (equal? s1 (cadar MT)))
          (cddar MT)
```

```
            (triplet (cdr MT) e1 s1) )
        () ) ))
```

2°) Ecrire une fonction `tronquer` ayant comme argument une liste non vide de symboles représentant un mot non vide, `m`, et telle que l'évaluation de l'expression `(tronquer m)` retourne, en un temps linéaire en fonction de la longueur de la liste `m`, une liste contenant en première position le mot `m` tronqué de son dernier symbole, et en deuxième position ce dernier symbole.

Par exemple, l'évaluation de `(tronquer '(a b c d))` doit retourner la liste :

`((a b c) d)`.

Indication : pour obtenir un temps d'évaluation qui soit linéaire en fonction de la longueur de la liste `m`, on peut par exemple écrire et utiliser une fonction qui retourne l'image miroir d'une liste en un temps linéaire.

Solution :

```
(define tronquer (lambda (m)
   (let* ((rm (miroir m)) (s (car rm)) (mt (miroir (cdr rm))))
      (list mt s) ) ))
(define miroir (lambda (L) (mir L ())))
(define mir (lambda (L LM)
   (if L (mir (cdr L) (cons (car L) LM)) LM) ))
```

3°) Ecrire (en utilisant la fonction `tronquer`) une fonction `nouvelle_conf` ayant comme arguments une liste représentant une configuration, `conf`, et une liste représentant un triplet de type `(e2 s2 DG)`, `t`, et telle que l'évaluation de l'expression `(nouvelle_conf conf t)` retourne, si celle-ci est possible, une nouvelle configuration dans laquelle l'état courant est `e2`, le symbole placé sous la tête de lecture a été remplacé par `s2`, et la tête de lecture a été déplacée conformément à la direction donnée par `DG`, et retourne la liste vide si cette nouvelle configuration est impossible (uniquement dans le cas d'un déplacement à gauche alors que la tête de lecture est déjà en début de ruban).

Exemples :

a) l'évaluation de `(nouvelle_conf '(e1 (a b) (c d)) '(e2 e G))` doit retourner `(e2 (a) (b e d))`, car avant la transition la tête de lecture se trouve sur le premier symbole (`c`) de la troisième sous-liste (`(c d)`) du premier argument (`'(e1 (a b) (c d))`), que ce symbole est remplacé par un `e` et que la tête de lecture

étant déplacée à gauche le deuxième symbole (b) de la deuxième sous-liste ((a b)) de ce même premier argument se retrouve sous la tête de lecture, donc en tête de la troisième sous-liste de la nouvelle configuration.

b) l'évaluation de (nouvelle_conf '(e1 () (a a)) '(e2 b G)) doit retourner (), parce que la deuxième sous-liste du premier argument étant vide, cela signifie que la tête de lecture est en début de ruban, et ne peut donc pas être déplacée vers la gauche.

Solution :

```
(define nouvelle_conf (lambda (conf t)
   (let ((e1 (car conf)) (m1 (cadr conf)) (m2 (caddr conf))
         (e2 (car t)) (s2 (cadr t)) (DG (caddr t)) )
   (if (equal? DG 'G)
      (if (null? m1) ()
          (let* ((L (tronquer m1))
                 (nm1 (car L)) (s (cadr L)) )
          (list e2 nm1 (cons s (cons s2 (if m2 (cdr m2) ()))))
           ) )
      (list e2 (append m1 (list s2)) (if m2 (cdr m2) ())) ) )
))
```

4°) Ecrire (en utilisant les fonctions triplet et tronquer) une fonction exec ayant comme arguments une liste de quintuplets représentant une machine de Turing, MT, et une liste de trois éléments représentant une configuration initiale, conf, et telle que l'évaluation de l'expression (exec MT conf) retourne la représentation de l'exécution de MT à partir de cette configuration initiale, c'est-à-dire, soit la liste des configurations atteintes jusqu'à une configuration finale où la machine est dans l'état accepteur, soit la liste des configurations atteintes jusqu'à une configuration qui réapparaît une deuxième fois (dans le cas où l'exécution boucle).

Solution :

```
(define exec (lambda (MT conf) (ex MT conf (list conf))))
(define ex (lambda (MT conf LC)
   (let* ((e1 (car conf)) (m1 (cadr conf)) (m2 (caddr conf)))
   (if (equal? e1 'accepteur)
      (miroir LC)
      (let* ((s1 (if m2 (car m2) 'blanc))
             (t (triplet MT e1 s1))
```

```
            (nconf (if t (nouvelle_conf conf t) ())) )
      (if (null? nconf) (miroir LC)
          (if (membre? nconf LC) (miroir (cons nconf LC))
              (ex MT nconf (cons nconf LC)) ) ) ) ) ) ))
```

5°) Déduire la fonction `executer` de la fonction `exec`.

Solution :

```
(definer excuter (lambda (MT mot)
   (exec MT (list 'initial () mot)) ))
```