Examen de Logique et de Prolog — ESSI2

(Documents autorisés: *tous*)

Hélène Collavizza et Michel Rueher

Mai 2000

1 Logique

1. Dans la théorie τ des listes avec les constructeurs suivants (les c_i représentent des atomes, nil et l correspondent à des listes):

   - nil
   - $c.nil$
   - $c_1.c_2.l$

   (a) (3 points) Donner les axiomes définissant :

   - $pair(l)$: les éléments de rang pair de l
   - $impair(l)$: les éléments de rang impair de l
   - $alterne(l_1, l_2)$: la liste contenant les éléments pris alternativement dans l_1 et l_2. (remarque : si les listes l_1 et l_2 sont de longueurs différentes on ajoute à la fin les éléments de la plus longue des listes).

   Exemple:

   - $pair(a.b.c.d.nil) = b.d.nil$
   - $impair(a.b.c.d.e.nil) = a.c.e.nil$
   - $alternee(a.b.c.d.nil, 1.2.3.4.5.nil) = a.1.b.2.c.3.d.4.5.nil$

   (b) (3 points) Montrer que $alternee(impair(l), pair(l)) = l$ est vrai dans les modèles initiaux de τ.

2. On considère les prédicats:

   - $milieu(p_1, p_2, m)$: m est le milieu du segment $[p_1, p_2]$
   - $perpendiculaire(p_1, p_2, p_3, p_4)$: le segment $[p_1, p_2]$ est perpendiculaire au segment $[p_3, p_4]$
   - $isocelle(p_1, p_2, p_3)$: le triangle défini par les points (p_1, p_2, p_3) est isocelle

   (a) (3 points) Ecrire à l'aide de ces seuls prédicats les énoncés suivants sous forme de formules en logique du premier ordre :

i. Si la hauteur en B du triangle ABC passe par le milieu du segment$[A, C]$ alors ABC est isocelle.

ii. Pour tout couple de points (AB), il existe un point C tel que AB soit perpendiculaire au segment déterminé par C et le milieu du segment$[A, B]$

iii. Pour tout couple de points (AB), il existe un point C qui est le milieu du segment$[A, B]$

(b) (3 points) Utiliser les énoncés précédents pour montrer que pour tout couple de points (AB) il existe un point C tel que le triangle ABC soit isocelle. Détailler les différentes étapes de la preuve par résolution (distinguer clairement les variables des constantes).

2 Prolog

1. (2 points) Expliquer pourquoi le prédicat `renverser_1` est moins efficace que le prédicat `renverser_2`.

```
renverser_1([],[]).
renverser_1([A|L],Z):-
        renverser_1(L,L1),
        append(L1,[A],Z).
```

```
renverser_2(L1,L2):- renverser(L1,[],L2)
renverser([],L,L).
renverser([T|Q],L_aux,L):-
    renverser(Q,[T|L_aux],L
```

2. Soit le programme Prolog qui résoud “l'équation” `SEND + MORE = MONEY` :

```
:- use_module(library(lists)).

mm1([S,E,N,D,M,O,R,Y]) :-
     generate1([S,E,N,D,M,O,R,Y]),
     S>0, M >0,
     alldiff([S,E,N,D,M,O,R,Y]),
     test_sum(S,E,N,D,M,O,R,Y).
```

```
mm2([S,E,N,D,M,O,R,Y]) :-
     generate2([S,E,N,D,M,O,R,Y]),
     S>0, M >0,
     alldiff([S,E,N,D,M,O,R,Y]),
     test_sum(S,E,N,D,M,O,R,Y)
```

```
test_sum(S, E, N, D, M, O, R, Y) :-
     V  is            1000*S + 100*E + 10*N + D
         +            1000*M + 100*O + 10*R + E,
     V is 10000*M + 1000*O + 100*N + 10*E + Y,
```

```
generate1([]).
generate1([X|Xs]):-
    member(X,[0,1,2,3,4,5,6,7,8,9]),
    generate1(Xs).
```

```
generate2([]).
generate2([X|Xs]):-
      digit(X),
      generate2(Xs).
```

```
alldiff([]).

alldiff([X|L]):-
    diff(X,L),
    alldiff(L).
diff(X,[]).
diff(X,[V|L]):-
     X ¯ V,
     diff(X,L).
```

```
digit(0).
digit(1).
digit(2).
...
digit(9).
```

(a) (2 points) Comparer et expliquer les différences de performance des appels `mm1([S,E,N,D,M,O,R,Y])` et `mm2([S,E,N,D,M,O,R,Y])` (syntaxiquement ces prédicats ne diffèrent que par `generate1/1` et `generate2/1`).

(b) (2 points) Donner 2 ordonnancements de la liste [S,E,N,D,M,O,R,Y] pour le prédicat `generate1/1` qui accélèrent considérablement les performances de `mm1/`. Expliquer.

3. Machine de Türing déterministe

(a) (3 points) Ecrire un interprète $t\_machine/2$ pour une machine de Türing déterministe. Cet interprète fera appel à un prédicat $transition/5$ qui sera défini pour chaque langage.

- $t\_machine(Tape, Answer)$:
  - $Tape$: liste binaire qui contient le mot à lire sur le ruban,
  - $Answer$: liste binaire qui contient le résultat de l'interprétation (ok ou ko), l'état de la machine et l'état du ruban.
- $transition(Q_0, W_0, Q_1, W_1, D)$:
  - Q_0: l'état de la machine avant la transition,
  - W_0: le caractère se trouvant sous la tête de lecture,
  - Q_1: l'état de la machine après la transition,
  - W_1: le caractère qui sera écrit sur le ruban à la place du caractère se trouvant sous la tête de lecture,
  - D: le sens de déplacement de la tête de lecture.

(b) (3 points) Ecrire les règles de transition pour les langages suivants:

- $a^n b^n$
- mots sur l'alphabet {a,b} dont la longueur est une puissance de 2.

Annexe

Définition d'une machine de Türing déterministe(rappel)

Une machine de türing déterministe est un automate défini par $M = (Q, \Gamma, \Sigma, \delta, s, B, F)$ où:

- Q est un ensemble fini d'états,
- Γ est l'alphabet utilisé sur le ruban,
- $\Sigma \subseteq \Gamma$ est l'alphabet utilisé pour le mot d'entrée,
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ est la fonction de transition,
- $s \in Q$ est l'état initial,
- $B \in \Gamma - \Sigma$ est le symbole blanc (noté #),
- $F \subseteq Q$ est l'ensemble des états accepteurs.

Une configuration contient toute l'information nécessaire pour poursuivre l'exécution (i.e., l'état, le mot écrit sur le ruban avant la tête de lecture, le mot se trouvant entre la position de la tête de lecture et le dernier caractère non blanc). Les configurations dérivable à partir de $(q, \alpha_1, b\alpha_2)$ sont :

- $(q', \alpha_1 b', \alpha_2)$ si $\delta(q, b) = (q', b', R)$
- $(q, \alpha_1', ab'\alpha_2)$ si $\delta(q, b) = (q', b', L)$ et $\alpha_1 = \alpha_1' a$