ESIAL 3A IL LE

Partiel de Programmation Logique

February 18, 2010

Mise en route:

La durée de l'examen est de **2h**. **Les exercices sont indépendants** et peuvent être faits dans le désordre. **Tous documents papier autorisés, l'échange de documents est interdit**.

Certaines indications vous suggèrent de créer certains prédicats intermédiaires. Il n'est pas obligatoire de les utiliser et **vous pouvez introduire vos propres prédicats auxiliaires**. Dans ce cas vous n'hésiterez pas à décrire leur fonctionnement et la manière de s'en servir.

Nous considérons l'implantation Gnu-Prolog vu en cours et en TD.

La notation tiendra compte de la présence d'explications dans vos réponses (sans pour autant paraphraser les programmes écrits) ainsi que de la qualité de la rédaction et de la présentation. Le barème sur 20 est donné à titre indicatif.

Exercice 1 Résolution (5 points)

On charge le programme suivant :

```
p(X,Y,X) :- q(X,Y).
p(X,X,_) :- r(X,Y), !, s(Y).
p(X,Y,Z) :- q(Y,Z).

q(X,Y) :- X < 3, Y is 3 * X.
q(Y,X) :- s(X), X < 3, r(X,Y).

r(X,X).
r(X,Y) :- s(Y), Y < X.

s(2).
s(3).
s(5).
```

1. Quelle est la réponse Prolog aux questions suivantes ?
   On vous demande :
   - si la preuve réussit (Prolog indique yes)
   - la valeur des valeurs des variables lorsque c'est possible ou vous expliquerez brièvement pourquoi la preuve échoue
   - dans le cas où il y a une erreur vous l'indiquerez et expliquerez pourquoi Pour la premières question, on vous demande en plus l'arbre de résolution Prolog

```
1-1 ?- p(1,2,X).
1-2 ?- p(1,X,2).
1-3 ?- p(5,5,1).
1-4 ?- p(X,2,X).
1-5 ?- p(Y,X,s(Y)) = p(q(r(a),Z),sf(V),V).
1-6 ?- p([X|Y],[Y|Z],V) = p([a,b,c],W,[X,W]).
```

Solution de l'Exercice 1

```
1-1 ?- p(1,2,X).
X = 1 ? ;
X = 6 ? ;
X = 2 ? ;
no

1-2 ?- p(1,X,2).
no

1-3 ?- p(5,5,1).
yes

1-4 ?- p(X,2,X).
uncaught exception: error(instantiation_error,(<)/2)

1-5 ?- p(Y,X,s(Y)) = p(q(r(a),Z),sf(V),V).
V = s(q(r(a),Z))
X = sf(s(q(r(a),Z)))
Y = q(r(a),Z)
yes

1-6 ?- p([X|Y],[Y|Z],V) = p([a,b,c],W,[X,W]).
V = [a,[[b,c]|Z]]
W = [[b,c]|Z]
X = a
Y = [b,c]
yes
```

Exercice 2 Puzzle (2 points)

Soit le problème de cryptographie suivant, où il faut associer à chaque lettre un chiffre unique (compris entre 0 et 9), de telle manière qu'en remplaçant les lettres par les chiffes correspondants, l'équation suivante soit vérifiée.

```
  D O N A L D
+ G E R A L D
-------------
  R O B E R T
```

1. Proposez un code Prolog permettant de trouver quel chiffre associer à quelle lettre pour que cette équation fonctionne (Vous pouvez utiliser les fonctions `fd_*`).

Solution de l'Exercice 2

```
main([D,O,N,A,L,G,E,R,B,T]) :- fd_domain([D,O,N,A,L,G,E,R,B,T],0,9),
       fd_all_different([D,O,N,A,L,G,E,R,B,T]),
                              fd_labeling([D,O,N,A,L,G,E,R,B,T]),
        ((D * 100000 + O * 10000 + N * 1000 + A * 100 + L * 10 + D * 1) +
         (G * 100000 + E * 10000 + R * 1000 + A * 100 + L * 10 + D * 1)) =:=
        (R * 100000 + O * 10000 + B * 1000 + E * 100 + R * 10 + T * 1).
```

Exercice 3 Compression (4 points)

1. Proposez un code Prolog permettant de compresser une liste de lettres. Un groupe d'occurrences adjacentes d'une même lettres est compressé en un terme `[N,L]`, où `L` est la lettre et `N` le nombre d'occurrences adjacentes de celle-ci. Par exemple on veut avoir:

   ```
   | ?- encode([a,a,a,a,b,c,c,a,a,d,e,e,e,e],X).
   X = [[4,a],[1,b],[2,c],[2,a],[1,d][4,e]]
   ```

Solution de l'Exercice 3

```
compresser([],_,0,[]).
compresser([X|Y],X,N,Z) :- compresser(Y,X,M,Z), N is M + 1.
compresser([DX|Y],X,0,[[N,DX]|Z]) :- compresser(Y,DX,M,Z), N is M + 1.

compresser([X|Y],[[N,X]|Z]) :- compresser(Y,X,M,Z), N is M + 1.

% compresser([a,a,a,a,b,c,c,a,a,d,e,e,e,e],X).
```

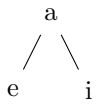
Exercice 4 Arbres binaires (9 points)

Soit la structure d'arbre binaire définie à partir de la structure `bin/3` ayant comme arguments un nom de nœud, un fils gauche et un fils droit, et de la structure `nil\0` pour représenter les nœuds vides. Une feuille est représentée par un arbre sans fils (e.g. `bin(a,nil,nil)`).

Voici quelques exemples d'arbres binaires:

Arbre 1:

```
bin(a,bin(e,nil,nil),
      bin(i,nil,nil))
```

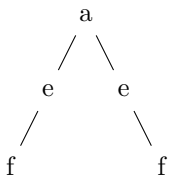


Arbre 2:

```
bin(a,nil,nil)
```

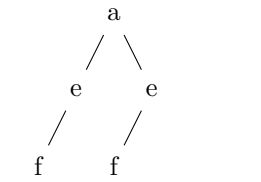

Arbre 3:

```
bin(a,bin(e,bin(f,nil,nil),
            nil),
      bin(e,nil,
            bin(f,nil,nil)))
```



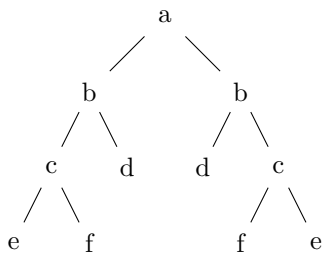
Arbre 4:

```
bin(a,bin(e,bin(f,nil,nil),
          nil),
      bin(e,bin(f,nil,nil),
          nil))
```



Arbre 5:

```
bin(a,bin(b,bin(c,bin(e,nil,nil),
                  bin(f,nil,nil)),
            bin(d,nil,nil)),
      bin(b,bin(d,nil,nil),
            bin(c,bin(f,nil,nil),
                  bin(e,nil,nil)))
```



1. (2 points) Écrivez le prédicat `membre/2` permettant de savoir si un élément fait parti d'un arbre binaire. Par exemple:

```
| ?- membre(E,bin(a,bin(e,nil,nil),bin(i,nil,nil))).
E = a ? ;
E = e ? ;
E = i ? ;
no
| ?- membre(d,bin(a,bin(e,nil,nil),bin(i,nil,nil))).
no
```

2. (2 points) Écrivez le prédicat `nbfeuilles/2` permettant de connaitre le nombre de feuilles d'un arbre. Par exemple:

```
| ?- nbfeuilles(bin(a,bin(e,nil,nil),bin(i,nil,nil)), E).
E = 2 ? ;
no
| ?- nbfeuilles(bin(a,bin(e,bin(f,nil,nil),nil),bin(e,nil,bin(f,nil,nil))), E).
E = 2 ? ;
no
```

3. (2 points) Écrivez le prédicat `hauteur/2` permettant de connaitre la hauteur d'un arbre. Par exemple:

```
| ?- hauteur(bin(a,bin(e,nil,nil),bin(i,nil,nil)), E).
E = 2
yes
| ?- hauteur(bin(a,bin(e,bin(f,nil,nil),nil),bin(e,nil,bin(f,nil,nil))), E).
E = 3
yes
```

4. (3 points) Soit le prédicat `symetrique/1` permettant de savoir si un arbre est symétrique ou non. Un arbre binaire est symétrique si en pliant l'arbre sur la verticale passant par la racine, les nœud et les arcs correspondent. Dans les cinq arbres définit précédemment seul les arbres 2, 3 et 5 sont symétriques. On veut avoir par exemple:

```
| ?- symetrique(bin(a,bin(e,bin(f,nil,nil),nil),bin(e,nil,bin(f,nil,nil)))).
true ?
yes
```

```
| ?- symetrique(bin(a,bin(e,bin(f,nil,nil),nil),bin(e,bin(f,nil,nil),nil))).
no
```

Nous vous proposons l'implantation suivante du prédicat symetrique:

```
symetrique(bin(_,nil,nil)).
symetrique(bin(_,bin(S,LX,RX),bin(S,LY,RY))):- sym(bin(S,LX,RX)),
                                                sym(bin(S,LY,RY)),!.
```

a) Quel est le résultat de la question:
`sym(bin(a,bin(e,bin(f,nil,nil),bin(f,nil,nil)),E)).`

b) Est-ce bien le résultat attendu ? Si non, expliquez pourquoi Prolog ne retourne pas le bon résultat et proposez des modifications pour corriger le problème.

Solution de l'Exercice 4

```
membre(X,bin(X,_,_)).
membre(X,bin(_,L,R)):- membre(X,L);membre(X,R).

nbfeuilles(nil,0).
nbfeuilles(bin(_,nil,nil),1):-!.
nbfeuilles(bin(_,X,Y),Z):- nbfeuilles(X,NX), nbfeuilles(Y,NY), Z is NX + NY.

max(X,Y,X):- X >= Y, !.
max(X,Y,Y):- X < Y.

hauteur(nil,0).
hauteur(bin(_,X,Y),Z):- hauteur(X,HX), hauteur(Y,HY), max(HX,HY,T), Z is T + 1.

sym(bin(_,nil,nil)).
sym(bin(_,bin(S,LX,RX),bin(S,LY,RY))):- sym(bin(S,LX,RX)), sym(bin(S,LY,RY)),!.

sym2(bin(_,nil,nil)).
sym2(bin(_,bin(S,LX,RX),bin(S,LY,RY))):- sym2(bin(S,LX,RY)), sym2(bin(S,LY,RX)),!.

% membre(E,bin(a,nil,nil)).
% membre(E,bin(a,bin(e,nil,nil),bin(i,nil,nil))).
%
% nbfeuilles(bin(a,nil,nil), E).
% nbfeuilles(bin(a,bin(e,nil,nil),bin(i,nil,nil)), E).
% nbfeuilles(bin(a,bin(e,nil,nil),E),2).
%
% hauteur(bin(a,nil,nil), E).
% hauteur(bin(a,bin(e,nil,nil),bin(i,nil,nil)), E).
% hauteur(bin(a,bin(e,nil,nil),E),2).
%
% bin(a,bin(e,bin(f,nil,nil),nil),bin(e,nil,bin(f,nil,nil)))
%
% sym(bin(a,nil,nil)).
% sym(bin(a,bin(e,nil,nil),bin(i,nil,nil))).
% sym(bin(a,bin(e,bin(f,nil,nil),nil),bin(e,nil,bin(f,nil,nil)))).
% sym2(bin(a,bin(e,bin(f,nil,nil),nil),bin(e,nil,bin(g,nil,nil)))).
```

```
% sym2(bin(a,bin(e,bin(f,nil,nil),nil),bin(e,bin(f,nil,nil),nil))).
```