Master d'Ingéniérie Informatique de Paris Diderot - Paris 7
M1: Prolog et programmation par contraintes

Examen partiel du 4 novembre 2008 - Durée: 1h45

Documents autorisés; le barème est donné à titre indicatif.

Exercice 1 (**3 points**)

Pour chacun des buts suivants, dire quel est le résultat renvoié par l'interpreteur Prolog :

1. `f(X,Y)==f(4,7).`
2. `f(X,Y)=f(4,7).`
3. `not(!);fail.`
4. `!,not(fail).`
5. `[X,0|[X|L]]=[1,0,1].`

Exercice 2 (**5 points**)

On considère les trois prédicats suivants :

```
if_1(T,P,Q):-T,!,P.
if_1(T,P,Q):-Q.

if_2(T,P,Q):-T,P,!.
if_2(T,P,Q):-Q.

if_3(T,P,Q):-!,T,P.
if_3(T,P,Q):-Q.
```

1. Pour chaque paire `j,k`, ($1 \leq$`j`$<$`k`≤ 3) trouver des arguments `T_jk,P_jk,Q_jk` tels que les buts `if_j(T_jk,P_jk,Q_jk)` et `if_k(T_jk,P_jk,Q_jk)` donnent des résultats différents[1].
2. Dessiner les arbres de dérivation de deux de ces buts, pour une paire `j,k` au choix.
3. Lequel de ces prédicats implémente correctement le `if-then-else`? (motiver brièvement).

Exercice 3 (**6 points**)

Dans cet exercice, on suppose que toutes les listes passées en argument aux différents prédicats sont des listes d'entiers (aucun contrôle n'est donc nécessaire). N'utilisez pas la coupure dans votre solution. Vous pouvez par contre utiliser la négation.

1. Ecrire un prédicat `filtre_negatifs(+liste_arg,-liste_res)` qui renvoie, dans `liste_res`, la liste `liste_arg` purgée de ses éléments négatifs.

   Par exemple:

   ```
   ?- filtre_negatifs([1,0,-6,7,-1],L).
   L = [1, 0, 7] ;
   No
   ```

[1] Il s'agit d'exhiber trois triplets: `T_12,P_12,Q_12`, puis `T_13,P_13,Q_13` et finalement `T_23,P_23,Q_23`.

2. Ecrire un prédicat `map_carre(+liste_arg,-liste_res)` qui renvoie, dans `liste_res`, la liste des carrés des éléments de `liste_arg`.

   Par exemple:

   ```
   ?- map_carre([2,-3,7],L).
   L = [4, 9, 49] ;
   No
   ```

3. Rappel: le prédicat prédefini `=..` transforme une liste en un terme. En voici un exemple d'utilisation:

   ```
   ?- T=..[p,5,7].
   T = p(5,7) ;
   No
   ```

   Ecrire un prédicat `filtre(+liste_arg,+nom_pred,-liste_res)` qui renvoie, dans `liste_res`, la liste `liste_arg` purgée de ses éléments `X` pour lesquels le but `nom_pred(X)` echoue. Le prédicat `nom_pred/1` doit être défini, au moment de l'appel de `filtre`.

   Par exemple, en supposant que `test` soit défini par `test(N):-N >= 0.`

   ```
   ?- filtre([-6,7,-1,0],test,L).
   L = [7,0] ;
   No
   ```

4. Ecrire un prédicat `map(+liste_arg,+nom_pred,-liste_res)` qui renvoie, dans `liste_res`, la liste des résultats obténus en appliquant le prédicat `nom_pred(+arg,-res)` aux éléments de `liste_arg`.

   Par exemple, en supposant que `test` soit défini par `test(N,R):-R is N*N.`

   ```
   ?- map([3,5,-2],test,L).
   L = [9, 25, 4] ;
   No
   ```

5. En utilisant `filtre` et `map`, écrire un prédicat `f(+liste_arg,-liste_res)` qui transforme une liste d'entiers en la liste des carrés de ses éléments non négatifs.

Exercice 4 (**6 points**)

Le jeu suivant est de la famille des jeux de Nim (deux joueurs qui s'alternent, information parfaite, somme nulle). Une configuration est un liste de listes. Un coup consiste en :

- retirer une des listes éléments de la configuration, ou bien :
- retirer le premier élément d'une des listes éléments de la configuration.

Ecrire un prédicat `move(+config_courante, -nouvelle_config)` qui implémente la rêgle du jeu.

Le joueur qui vide la configuration a perdu.

Ecrire un prédicat `gagne(+config_courante)` qui réussit si le joueur qui a la main à la configuration donnée a une stratégie gagnante.

Les feuilles des arbres de jeu sont des listes vides. Une feuille de niveau `max` vaut 1, une feuille de niveau `min` vaut -1. Dessiner l'arbre de jeu de racine `[[un,deux,trois]]`, et décorer les noeuds de cet arbre avec leur valeur selon l'algorithme `minimax` (la racine étant de niveau `max`). Donner une condition suffisante pour que une configuration soit gagnante.