Examen de Prolog

6 juin 2014

La clarté et la précision de la rédaction seront prises en compte dans l'évaluation.
Aucun document autorisé.
Machines (ordinateurs, calculatrices et téléphones) interdites.
Durée : 3h

Égalités

Donner et justifier les réponses de Prolog aux questions suivantes :

```
?- 2+2=4.
?- 2+2=:=4.
?- X=2+2.
?- x=X.
?- 2+2 is 4.
?- 4 is 2+2.
?- X is 2+2.
?- 2+2 is X.
```

Nombres premiers

Écrire un prédicat `premier(N)` qui est vrai si `N` est un entier naturel premier.

Écrire un prédicat `entiers_premiers(B,L)` qui est vrai si `L` est la liste des entiers premiers de 1 à `B`.

Nombres parfaits

On appelle entier parfait un entier dont la somme des diviseurs autre que lui est égale à lui-même. Par exemple, 6=1+2+3 et 28=1+2+4+7+14 sont deux entiers parfaits.

Écrire un prédicat `parfait(N)` qui est vrai si `N` est un entier parfait.

Nombre amicaux

On appelle nombre amicaux un couple de nombres dont la somme des diviseurs autre que lui est égale à l'autre et réciproquement.

Le plus petit couple de nombres amicaux est :

- 220 dont la somme des diviseurs vaut 1+2+4+5+10+11+20+22+44+55+110=284
- 284 dont la somme vaut 1+2+4+71+142=220

Écrire un prédicat `amicaux(N1,N2)` qui est vrai si `N1` et `N2` sont un couple d'entiers amicaux.

Tri rapide

On rappelle que le tri rapide consiste à choisir un élément (ici on prendra le premier) et à l'utiliser comme pivot, *i.e.* à séparer les autres éléments en une liste d'éléments plus petits que le pivot et une autre d'éléments plus grand que le pivot. On trie ensuite ces deux listes récursivement.

Écrire un prédicat `trirapide(L,LTriee)` qui est vrai si `LTriee` contient les mêmes éléments (ici des entiers) que `L` et est triée par la méthode du tri rapide.

Carré magique

On appelle carré magique une matrice 3×3 qui contient exactement une fois chacun des entiers de 1 à 9 de telle manière que la somme des lignes, des colonnes et des deux diagonales principales soit la même.

Exemple :

```
492
357
816
```

Écrire un programme prolog qui donne tous les carrés magiques sans utiliser la programmation par contraintes.

Écrire un programme prolog qui donne tous les carrés magiques en utilisant un module de programmation par contraintes que l'on précisera.

Triangle

Écrire un prédicat `triangle/1` qui appelé avec l'argument `N`, entier positif, dessine un triangle de taille `N` à l'écran. Par example, un triangle de taille 5 donne :

```
?- triangle(5).
     *
    * *
   * * *
  * * * *
 * * * * *
true.
```

Il y a un espace entre chaque paire d'étoiles horizontales. Entre le haut du triangle de taille `N` et le côté gauche de l'écran, il ne peut pas y avoir plus de `N+2` espaces.

Spanning spider

Un arbre couvrant en étoile (spanning spider) est un arbre contenant tous les sommets d'un graphe dont au maximum un seul sommet a un degré supérieur ou égal à 3. Tous les graphes n'ont pas de "spanning spider", et vous devez donc écrire un prédicat `spanspid/0` qui retourne vrai si un graphe possède un "spanning spider" et faux sinon. Le graphe, comme nous l'avons vu en TP est donné par une succession de prédicats représentant des faits, `edge/2`, ayant pour arguments les sommets connectés par une arête.

Nous considérons les graphes non orientés et connexes suivants :

```
edge(d,a).
edge(d,b).
edge(d,c).
edge(d,e).
edge(a,x).
edge(b,y).
edge(c,z).
edge(a,b).
edge(y,c).

?- spanspid.
Yes
```

```
edge(a,e).
edge(b,e).
edge(e,f).
edge(f,d).
edge(f,c).
edge(d,c).

?- spanspid.
No
```

```
edge(a,b).
edge(b,c).
edge(a,c).
edge(d,a).
edge(d,b).

?- spanspid.
Yes
```

1. Dessiner les 3 graphes ci dessus et vérifiez le resultat attendu.
2. Écrire le prédicat `spanspid`. L'algorithme consiste à générer tous les sous-graphes contenant un nombre d'arêtes égal au nombre de sommets du graphe initial moins un, qui couvrent tous les sommets. Un tel sous graphe est forcément un arbre couvrant. Il suffit dès lors de tester pour chacun des ces arbres couvrants s'il a au plus un sommet de degré 3 ou plus.
   Aide : l'utilisation du prédicat `findall/3` est fortement recommandée.
   Nous vous rappelons l'existence des prédicats suivant qui peuvent également être utiles :
   (a) `sublist/2` qui retourne vrai si le second argument est une sous liste du premier argument.
   ```
   ?- sublist([1,2,3,4],[1,3]).
   true.
   ```
   (b) `list_to_set/2` qui retourne vrai si le second argument correspond à la liste des éléments du premier argument sans doublons.
   ```
   ?- list_to_set([1,2,3,3,4],[1,2,3,4]).
   true.
   ```
   (c) `sort/2` qui retourne vrai si le second argument correspond à la liste des éléments du premier argument triés par ordre croissant.
   ```
   ?- sort([3,2,4,1],[1,2,3,4]).
   true.
   ```