Prolog – TP n°2
— Les listes —

Pierre-Alexandre FAVIER

2005 - 2006

1 Rappel : la syntaxe des listes Prolog

Les listes utilisent les crochets `[ ]` comme délimiteurs.
Les différents éléments de la liste sont séparés pas des virgules.
La liste vide est notée par une paire de crochets vides. Voici des exemples de listes valides :

```
[ ]
[a,b,c]
[a,b(56,54),c]
[a,[b,c,[] ], [[e],f,g] ]
[12.3,4.5]
```

L'opérateur *pipe* `|` permet de séparer la tête de la liste de sa queue. Exemples :

```
?- [a,b,c] = [T|Q].
T = a
Q = [b,c]
yes

?- [a] = [T|Q].
T = a
Q = [ ]
yes

?- [a,b,c] = [T1, T2 | Q]
T1 = a
T2 = b,
Q = [c]
yes
```

2 Représentations de listes

1. Représenter par un arbre la liste [a,b,c].
2. Ecrire de toutes les manières possibles la liste [a,b,c].

3 Informations sur les listes

Définir les prédicats suivants :

1. **liste(L)** *% **L** est une liste*
2. **premier(X,L)** *% **X** est le premier élément de la liste **L***
3. **dernier(X,L)** *% **X** est le dernier élément de la liste **L***
4. **longueur(N,L)** *% **N** est le nombre d'éléments de la liste **L***
5. **dans(X,L)** *% **X** appartient à la liste **L***

4 Manipulation de listes

Définir les prédicats suivants :

1. **conc(L1,L2,L3)** % $\mathbf{L_1} + \mathbf{L_2} = \mathbf{L_3}$.
2. **ajouter(X,L1,L2)** *% **X** est ajouté en fin de **L1** pour donner **L2**.*
3. **supprimer(X,L1,L2)** *% **X** est supprimé de **L1** pour donner **L2**.*
4. **substituer(X,Y,L1,L2)** *% substituer **X** par **Y** dans **L1** donne **L2**.*
5. **convertir(L,LC)** *% **LC** est la liste **L** (**= [t1,t2,...,tn]**) écrite sous la forme* **t1 et t2 et ... et tn et fin**.
6. **aplatir(L,LA)** *% **LA** est la liste **L** qui ne contient plus qu'un seul niveau de crochets* (**[...]**).

5 Tris de listes

Définir les prédicats suivants :

1. **triFusion(L,LT)** *% **LT** est la liste **L** triée selon la méthode de tri par fusion.*

6 Ensembles

Un ensemble est représenté ici par une liste dont les éléments sont tous différents les uns des autres.

Définir les prédicats suivants :

1. **ensemble(L)** *% la liste **L** repésente un ensemble.*
2. **listeEnsemble(L,E)** *% la liste **L** est transformée en un ensemble **E***

3. **intersection(E1,E2,E3)** $\% \ \mathbf{E_3 = E_1 \cap E_2}$
4. **union(E1,E2,E3)** $\% \ \mathbf{E_3 = E1 \cup E_2}$
5. **sousEnsemble(E1,E2)** $\% \ \mathbf{E_1 \subset E_2}$

7 Listes et arithmétique

Définir le prédicat **somme(+Liste, ?N)** vrai si et seulement si la somme des éléments de la **Liste** vaut **N**.

8 Traitements génériques

Définir le prédicat **map(+Terme,+Liste1, ?Liste2)** vrai si et seulement si **Liste2** représente la **Liste1** où chaque élément a été traité à l'aide du prédicat de symbole fonctionnel **Terme**.

Exemple :

?- map(abs,[-1,2,-3],L).

```
L = [1,2,3]
true
```

?- map(sin,[1,2,3],L).

```
L = [-0.841470984807897,0.909297426825682,-0.141120008059867]
true
```

9 Ensembles de solutions

On considère les faits suivants :

```
triplet(a,b,2).
triplet(a,b,1).
triplet(b,c,3).
triplet(b,c,3).
triplet(b,b,1).
triplet(c,c,5).
triplet(c,c,2).

idem(X) :- triplet(X, X, X).
```

Interprétez le résultat des requêtes suivantes :

```
?- findall( X, idem(X), L).

?- findall( C, triplet(A,B,C), L).
```

```
?- bagof(X, idem(X), L).

?- bagof(C, triplet(A,B,C), L).

?- bagof(C, A^triplet(A,B,C), L).

?- bagof(C, B^triplet(A,B,C), L).

?- bagof(C, A^B^triplet(A,B,C), L).

?- setof(X, idem(X), L).

?- setof(C, triplet(A, B, C), L).

?- setof(C, triplet(A, B, C), L).

?- setof(C, B^triplet(A, B, C), L).

?- setof(C, A^B^triplet(A, B, C), L).
```

10 Coloriage de carte plus complet

Soit une base de fait contenant les régions adjacentes d'une carte sous la forme suivante :

```
adjacent(r1,r2).
adjacent(r2,r3).
adjacent(r3,r1).
adjacent(r4,r45).
adjacent(r45,r3).
adjacent(r4,r3).
..
..
```

Ecrire le prédicat **colorier/1** vrai si son argument est une liste contenant les termes **(X-Y)** ou **X** est une région et **Y** est sa couleur. Toutes les régions adjacentes ayant des couleurs différentes.

11 Questions subsidiaires

1. **dans(X,L)** % **X** *appartient à la liste* **L**
2. **hors(X,L)** % **X** *n'appartient pas à la liste* **L**
3. **unique(X,L)** % **X** *n'apparaît qu'une seule fois dans la liste* **L**
4. **occurence(X,N,L)** % **X** *apparaît* **N** *fois dans la liste* **L**
5. **prefixe(P,L)** % **P** *est un préfixe de la liste* **L**
6. **suffixe(S,L)** % **S** *est un suffixe de la liste* **L**
7. **sousListe(SL,L)** % **SL** *est une sous–liste de la liste* **L**
8. **inserer(X,L1,L2)** % **X** *est inséré en tête de* **L1** *pour donner* **L2**.
9. **inverser(L1,L2)** % *inverser* **L1** *donne* **L2**.
10. **decaler(L,LD)** % **LD** *est la liste* **L** *où les éléments ont été décalés à droite.*