Document et Web sémantique - TP Prolog

Quelques prédicats utiles

Les prédicats suivants peuvent vous être utiles (Cf. la documentation de swi-prolog) :

— *append/3*
— *atomic/1*
— *var/1*
— *nonvar/1*

1 Prédicats sans cut

1.1 *premier/2*

Développez le prédicat *premier/2* tel que `premier(X,L)` est vrai lorsque `X` est le premier élément de `L`.

Voici un exemple de résultats voulus :

```
?- premier(1,[1,2,3]).
true.

?- premier(1,[3,1,3]).
fail.
```

1.2 *dernier/2*

Développez le prédicat *dernier/2* tel que `dernier(X,L)` est vrai lorsque `X` est le dernier élément de `L`.

Voici un exemple de résultats voulus :

```
?- dernier(1,[1,2,1]).
true ;
fail.

?- dernier(1,[1,2,3]).
fail.
```

1.3 *inverser/2*

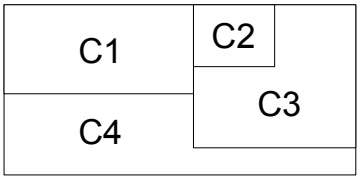
Développez le prédicat *inverser/2* tel que `inverser(L1,L2)` est vrai lorsque `L2` représente la liste inverse de `L1`.

Voici un exemple d'utilisation :

```
?- inverser([1,2,3],L).
L = [3, 2, 1].
```

1.4 Coloriage (issu du livre Objectif Prolog)

On se propose de définir un prédicat permettant de coloriser la carte suivante :



Les règles sont les suivantes :
— On dispose de trois couleurs (vert, rouge, bleu) ;
— Deux zones contiguës doivent avoir des couleurs différentes.

Développez le prédicat *colorier/4* pour colorier cet exemple (et uniquement cet exemple, nous généraliserons le problème par la suite).

1.5 Tri par fusion

1.5.1 fusionner

Développez le prédicat *fusionner/3* qui permet de fusionner deux listes ordonnées : `fusionner(L1,L2,L3)` est vrai lorsque `L3` représente la fusion des listes `L1` et `L2`.

Voici un exemple de résultats attendus :

```
?- fusionner([1,4,5],[2,3,7],L).
L = [1, 2, 3, 4, 5, 7] ;
false.
```

1.5.2 decouperEnDeux

Développez le prédicat *decouperEnDeux/3* qui permet de découper une liste en deux parties de même longueur (à un près).

Voici un exemple de résultats attendus :

```
?- decouperEnDeux([1,2,3],L1,L2).
L1 = [1],
L2 = [2, 3] ;
false.

?- decouperEnDeux([1,2],L1,L2).
```

```
L1 = [1],
L2 = [2] ;
false.

?- decouperEnDeux([1],L1,L2).
L1 = [],
L2 = [1] .

?- decouperEnDeux([4,5,7,9,32,4,6,4],L1,L2).
L1 = [4, 5, 7, 9],
L2 = [32, 4, 6, 4] ;
false.
```

1.5.3 triParFusion

Développez le prédicat *triParFusion/2* qui permet de trier une liste à l'aide de l'algorithme du tri par fusion.

Voici un exemple de résultats attendus :

```
?- triParFusion([4,7,2,1,5,6,3,4,8],L).
L = [1, 2, 3, 4, 4, 5, 6, 7, 8] ;
false.
```

2 Prédicats avec cut

2.1 *aplat/2*

Développer le prédicat *aplat/2* qui permet « d'aplatir » une liste de listes.

Voici un exemple de résultats voulus :

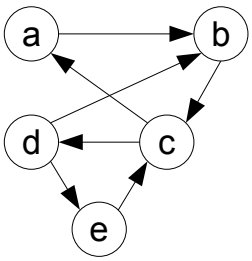
```
?- aplat([1,[2],3],L).
L = [1, 2, 3].

?- aplat([1,[2,[4,5]],3],L).
L = [1, 2, 4, 5, 3].
```

2.2 Recherche des chemins acycliques (inspiré du livre Objectif Prolog)

Soit un graphe orienté identifié à l'aide du prédicat *arc/2*. Développez un prédicat *cheminAcyclique/3* qui permet d'obtenir les listes de tous les nœuds formant le chemin permettant d'aller d'un nœud source à un nœud destination sans cycle.

Par exemple pour le graphe suivant :



On a :

```
?- cheminAcyclique(d,a,L).
L = [d, b, c, a] ;
L = [d, e, c, a] ;
fail.

?-
```

2.3 Coloriage (bis)

On se propose de généraliser le problème du coloriage de zones. On suppose posséder des faits définissant :

— les couleurs disponibles, *couleur/1*

— que deux zones soient contigües, *voisin/2*

Par exemple, pour l'exemple précédent nous avons les faits suivants :

```
couleur(vert).
couleur(rouge).
couleur(bleu).

voisin(c1,c2).
voisin(c1,c4).
voisin(c2,c3).
voisin(c3,c4).
```

L'objectif général est d'avoir un predicat *colorier/2* qui à partir d'une liste de zones détermine une liste d'arbres binaires (nommés couleur) qui associe une couleur à chacune des zones de la première liste.

Par exemple, pour l'exemple précédent on aura le résultat suivant :

```
?- colorier([c1,c2,c3,c4],L).
L = [couleur(c1, vert), couleur(c2, rouge), couleur(c3, vert), couleur(c4, rouge)] ;
L = [couleur(c1, vert), couleur(c2, rouge), couleur(c3, vert), couleur(c4, bleu)] ;
L = [couleur(c1, vert), couleur(c2, rouge), couleur(c3, bleu), couleur(c4, rouge)] ;
L = [couleur(c1, vert), couleur(c2, bleu), couleur(c3, vert), couleur(c4, rouge)] ;
L = [couleur(c1, vert), couleur(c2, bleu), couleur(c3, vert), couleur(c4, bleu)] ;
L = [couleur(c1, vert), couleur(c2, bleu), couleur(c3, rouge), couleur(c4, bleu)] ;
L = [couleur(c1, rouge), couleur(c2, vert), couleur(c3, rouge), couleur(c4, vert)] ;
L = [couleur(c1, rouge), couleur(c2, vert), couleur(c3, rouge), couleur(c4, bleu)] ;
L = [couleur(c1, rouge), couleur(c2, vert), couleur(c3, bleu), couleur(c4, vert)] ;
L = [couleur(c1, rouge), couleur(c2, bleu), couleur(c3, vert), couleur(c4, bleu)] ;
L = [couleur(c1, rouge), couleur(c2, bleu), couleur(c3, rouge), couleur(c4, vert)] ;
L = [couleur(c1, rouge), couleur(c2, bleu), couleur(c3, rouge), couleur(c4, bleu)] ;
L = [couleur(c1, bleu), couleur(c2, vert), couleur(c3, rouge), couleur(c4, vert)] ;
```

```
L = [couleur(c1, bleu), couleur(c2, vert), couleur(c3, bleu), couleur(c4, vert)] ;
L = [couleur(c1, bleu), couleur(c2, vert), couleur(c3, bleu), couleur(c4, rouge)] ;
L = [couleur(c1, bleu), couleur(c2, rouge), couleur(c3, vert), couleur(c4, rouge)] ;
L = [couleur(c1, bleu), couleur(c2, rouge), couleur(c3, bleu), couleur(c4, vert)] ;
L = [couleur(c1, bleu), couleur(c2, rouge), couleur(c3, bleu), couleur(c4, rouge)] ;
fail.
```

1. Proposer le prédicat *voisins/2* qui permet d'obtenir la liste des voisins d'une zone. Par exemple :

   ```
   ?- voisins(c2,L).
   L = [c3, c1].
   ```

2. Proposer le prédicat *zonesAvecCouleur/3* qui permet d'obtenir à partir d'une liste d'arbres *couleur*, les zones qui ont la même couleur qu'une couleur donnée. Par exemple :

   ```
   ?-  zonesAvecCouleur(rouge,[couleur(c1, rouge), couleur(c2, bleu), couleur(c3, blanc),
       couleur(c4, rouge)],L).
   L = [c1, c4] ;
   fail.
   ```

3. Proposer le prédication *intersectionVide/2* qui permet de savoir si deux listes de zones ont une intersection vide. Par exemple :

   ```
   ?- intersectionVide([c1,c2,c3],[c2,c4]).
   fail.

   ?- intersectionVide([c1,c3],[c2,c4]).
   true.
   ```

4. Proposer le prédicat *colorier/2*.

3 Prolog et le Web des données liées

Rappels

Comme vu en cours, SWI prolog propose plusieurs bibliothèques (`semweb/`) permettant de gérer des triplets RDF [1]. Le chargement d'une bibliothèque est réalisé grâce au prédicat *use_module/1* avec comme paramètre un arbre *library*.

La question suivante charge la bibliothèque `semweb/rdf_db` :

```
?- use_module(library(semweb/rdf_db)).
```

Pour qu'un programme prolog puisse chargé un module, il faut donc qu'il pose une question. Pour cela il fait la préfixer par un `:-`.

Dans cet exercice nous avons besoin des bibliothèques `semweb/rdf_db` et `semweb/rdf_turtle`.

Il est possible d'enregistrer des préfixes d'URI non standard (autres que `rdf:`, `rdfs:`, ...) à l'aide du prédicat *rdf_register_prefix/2* (les préfixes déclarés dans un fichiers turtle ne sont pas par défaut créés). Une fois un préfixe enregistré, on peut utiliser une URI préfixée comme paramètre des prédicats en entourant l'URI de simples guillemets. Pour qu'un prédicat puisse utiliser la notation préfixé, il faut utiliser le prédicat *rdf_meta/1*.

1. `http://www.swi-prolog.org/pldoc/man?section=semweb-rdf-db`)

Questions

Soit la base RDF suivante disponible sur moodle :

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix dom: <https://purl.oclc.org/a#> .

dom:A
    rdf:type rdfs:Class .

dom:B
    rdf:type rdfs:Class .

dom:pAB
    rdf:type rdf:Property;
    rdfs:domain dom:A ;
    rdfs:range dom:B .

dom:pAA
    rdf:type rdf:Property ;
    rdfs:domain dom:A ;
    rdfs:range dom:A .

dom:iA1
    rdf:type dom:A ;
    dom:pAA dom:iA2 .

dom:iA2
    rdf:type dom:A ;
    dom:pAA dom:iA3 .

dom:iA3
    rdf:type dom:A.

dom:iB1
    rdf:type dom:B;
    dom:pAB dom:iB2.
```

1. Après avoir étudier le prédicat *rdf/3*, donnez une expression prolog qui permet d'obtenir toutes les instances de la classe `dom:A`. Pourquoi `dom:iB1` n'apparait pas ?
2. idem avec la classe `dom:B`. Pourquoi `dom:iB2` n'apparait pas ?
3. Proposez le prédicat *est_une_classe/1* qui permet de savoir si une ressource est une classe, par exemple :

   ```
   ?- est_une_classe('https://purl.oclc.org/a#A').
   true.
   ```

```
?- est_une_classe(U).
U = 'https://purl.oclc.org/a#A' ;
U = 'https://purl.oclc.org/a#B'.
```

4. Proposez le prédicat *est_une_instance_de/2* qui permet de savoir si une ressource est instance d'une classe (utilisez l'opérateur *rdf_meta/1*) en prenant en compte les propriétés `rdf:type`, `rdfs:domain` et `rdfs:range`.

```
?- est_une_instance_de(U,dom:'A').
U = 'https://purl.oclc.org/a#iA1' ;
U = 'https://purl.oclc.org/a#iA2' ;
U = 'https://purl.oclc.org/a#iA3' ;
U = 'https://purl.oclc.org/a#iA1' ;
U = 'https://purl.oclc.org/a#iA2' ;
U = 'https://purl.oclc.org/a#iB1' ;
U = 'https://purl.oclc.org/a#iA2' ;
U = 'https://purl.oclc.org/a#iA3' ;
```

5. Après avoir étudié le prédicat *rdf_assert/3*, proposez le précidat *ajouter_liens_instanciation/2* qui permet d'ajouter un lien d'instanciation rdf entre chaque ressource d'une liste et une classe donnée.
6. Proposez le prédicat *completer_liens_instanciation/0* qui permet de completer la base rdf avec des liens d'instanciation.
7. Après avoir complété la base, vérifiez que maintenant `dom:iB1` est bien une instance de `dom:A`.

4 Suppléments

Voici des exercices pour vous entrainer à développer des prédicats Prolog.

4.1 Le compte est bon

« Le but de ce jeu est d'obtenir un nombre (de 100 à 999) à partir d'opérations élémentaires $(+, -, *, /)$ sur des entiers naturels, en partant de nombres tirés au hasard (de 1 à 10, 25, 50, 75 et 100). »

Nous allons développer une prédicat *leCompteEstBon/3* qui donnera la liste des opérations à effectuer pour résoudre le problème ou qui répondra faux lorsqu'il n'y a pas de solution, comme par exemple :

```
?- leCompteEstBon([2,75,3,10,25,7],136,L).
L = ['75+2=77', '77+7=84', '25-3=22', '22*10=220', '220-84=136', '
   Le compte est bon'].
```

1. Étudiez les prédicat *concat_atom/2*.
2. Proposez un prédicat *supprimer/3* qui supprime le première occurrence d'un élément d'une liste.
3. Proposez un prédicat *tirerDeuxNombres/4* qui permet d'obtenir deux éléments d'une liste (tel que le premier est supérieur ou égal au deuxième) ainsi que la liste initiale avec les deux éléments choisis en moins.
4. Proposez le prédicat *leCompteEstBon/3*.

4.2 Tri rapide

1. Développez le prédicat *partitionner/4* qui permet de partitionner en 2 listes $L1$ et $L2$ une liste L tels que tous les éléments de $L1$ soient plus petits ou égaux à une valeur pivot VP et que ceux de $L2$ soient plus grands. Par exemple :

```
?- partitionner([2,4,9,1,3,4,7,6],4,L1,L2).
L1 = [2, 4, 1, 3, 4],
L2 = [9, 7, 6].
```

2. Développez le prédicat *trirapide/2* qui permet d'obtenir une liste $L2$ à partir d'une liste $L1$ tel que $L2$ soit un tri de $L1$ (algorithme du tri rapide). Par exemple :

```
?- trirapide([4,2,4,9,1,3,4,7,6],L).
L = [1, 2, 3, 4, 4, 4, 6, 7, 9].
```