BE2 - Exercices sur les listes

Algorithme et raisonnement

Emeline GOT

Table des matières

1 Recettes

On souhaite réaliser des recettes avec différents ingrédients. On utilise un base de données de repas et d'ingrédients nécessaires pour les préparer. Pour réaliser un repas en suivant sa recette, on va donc chercher à déterminer les ingrédients nécessaires ainsi que les quantités d'ingrédients associés.

Dans un premier temps, on entre la base de données des ingrédients et des recettes fournie sur la plateforme pédagogie.

```
%predicat disponible(ingrédient)
disponible(eau).
disponible(sucre).
disponible(sel).
disponible(vinaigre).
disponible(huile).
disponible(beurre).
disponible(sachet_de_the).
disponible(cafe).
disponible(chocolat).
disponible(lait).
disponible(farine).
disponible(oeufs).
disponible(pate_a_crepes).
disponible(pates_nouilles).
disponible(laitue).
disponible(pommes_dt).
disponible(oignon).
disponible(tomates).
disponible(creme).
disponible(fromage).  % toutes sortes. Pour une fondue, pour une
    tartiflette, raclette...
disponible(lardons).  % pour tester pates_a_la_carbonara


%prédicat recette(mets, liste des ingrédients)
recette(expresso, [cafe, eau, sucre ]).
recette(gateau, [lait, farine, oeufs, sucre, beurre ]).
recette(the, [sachet_de_the, eau, sucre ]).
recette(crepes, [pate_a_crepes, sucre, beurre ]).
recette(salade, [laitue, vinaigre, huile, oignon]).
recette(puree, [lait, pommes_dt, beurre, sel ]).
recette(frites, [pommes_dt, huile, sel]).
recette(pates_a_la_carbonara, [pates_nouilles, oeufs, lardons, sel,
    oignon, creme]).
recette(carbona, [pates_nouilles, oeufs, lardons, sel, oignon ]).
recette(pitza, [farine, oeufs, lardons, sel, oignon, tomates]).
recette(tartiflette, [pommes_dt, lardons, sel, fromage]).
```

1.1 Relation *peut_preparer(R)*

La relation *peut_preparer(R)* est vraie si tous les ingrédients nécessaires pour préparer le repas R sont disponibles. Pour cela, on vérifie que pour chacun des ingrédients présent dans la recette est disponible avec le prédicat disponible. Comme la recette présente une liste d'ingrédient, on utilise la définition de la liste comme une tête et une queue.
Dans un premier temps, on récupère la liste des ingrédients pour le repas considérer, et on définit une relation *peut_preparer(R,L)* qui prend en argument un repas et sa liste d'ingrédients. Ensuite, on caractérise la relation *peut_preparer(R,L)* sur la tête et la queue de la liste ainsi que sur le cas particulier où la liste d'ingrédients est vide (cas d'arrête de la récurrence sur la liste).

```
peut_preparer(R):- recette(R,L),peut_preparer(R,L). %La liste L est la
    liste d'ingrédients du repas R.
peut_preparer(R,[T|Q]):- disponible(T),peut_preparer(R,Q). %Le premier
    ingrédient T de la liste doit être disponible et on applique la
    relation par récurrence sur le reste de la liste
peut_preparer(R,[]). %Si la liste est vide, alors tous les ingrédients
    sont disponibles et on peut preparer le repas
```

Par exemple, si on enlève l'eau des ingrédients disponibles, on obtient les traces suivantes :

```
| ?- peut_preparer(expresso).
no
| ?- peut_preparer(the).
no
| ?- peut_preparer(frites).
yes
```

1.2 Relation *a_besoin_de(R,I)*

La relation *a_besoin_de(R,I)* doit être vraie si la recette du repas R contient l'ingrédient I. Pour cela, on définit dans un premier temps la relation emphmembre(L,I) qui vérifie que l'élément I est dans la liste L.

```
membre([],_):-fail. %Si la liste est vide, il n'y a pas d'appartenance
    possible
membre([A],A). %Si la liste est constituée de l'élément testé, c'est bon
membre([T|R],A):- T\=A. %Si la liste est constituée de plusieurs éléments,
    on vérifie si la tête de la liste vaut l'élément cherché
membre([T|R],A):- membre(R,A). %Sinon, on vérifie que si l'élément cherché
    est dans le reste de la liste
```

Pour définir la relation *a_besoin_de(R,I)*, il ne nous reste plus qu'à tester si l'ingrédient I est dans la recette du repas R. On récupère la liste des ingrédients de R avec le prédicat *recette* et ensuite on teste l'appartenance avec la relation *membre*.

```
a_besoin_de(R,I):- recette(R,L),membre(L,I).
```

```
| ?- a_besoin_de(gateau,lait).
yes
```

```
| ?- a_besoin_de(gateau,eau).
no
```

2 Introduction des quantités

2.1 Changement de la base de données

On introduit la notion de quantité dans le prédicat *disponible*. C'est-à-dire qu'on considère désormais que les ingrédients sont disponibles en quantité finie et que chaque recette a besoin d'un nombre précis de chacun des ingrédients.

Il faut donc redéfinir la base de données en introduisant ces changements.

```
%predicat disponible(ingrédient, quantité en litre/Kilo/mètres selon l
    ingérdient)
disponible(eau,50).
disponible(sucre,100).
disponible(sel,50).
disponible(vinaigre,4).
disponible(huile,3).
disponible(beurre,8).
disponible(sachet_de_the,5).
disponible(cafe,10).
disponible(chocolat,23).
disponible(lait,10).
disponible(farine,300).
disponible(oeufs,12).
disponible(pate_a_crepes,300).
disponible(pates_nouilles,300).
disponible(laitue,2).
disponible(pommes_dt,10).
disponible(oignon,10).
disponible(tomates,40).
disponible(lardons,50).    % pour tester pates_a_la_carbonara
disponible(creme,4).
disponible(fromage,10). % toutes sortes. Pour une fondue, pour une
    tartiflette, raclette...

%prédicat recette(mets, liste des ingrédients + la quantité nécessaire)
recette(expresso, [(cafe,3),(eau,10), (sucre,1)]).
recette(gateau, [(lait,5), (farine,100), (oeufs,6), (sucre,30), (beurre,
    2)]).
recette(the, [(sachet_de_the,1), (eau,10), (sucre,1)]).
recette(crepes, [(pate_a_crepes,20), (sucre,33), (beurre,2)]).
recette(salade, [(laitue,1), (vinaigre,3), (huile, 2), (oignon,1)]).
recette(puree, [(lait,2), (pommes_dt,5), (beurre, 2), (sel,1)]).
recette(frites, [(pommes_dt,5), (huile, 3), (sel,2)]).
recette(pates_a_la_carbonara, [(pates_nouilles, 40), (oeufs, 5), (lardons,
    5),(sel,3), (oignon,2)]).
```

```
recette(carbona, [(pates_nouilles, 40), (oeufs, 5), (lardons, 5),(sel,3),
    (oignon,2)]).
recette(pitza, [(farine, 40), (oeufs, 5), (lardons, 5),(sel,3), (oignon,2)
    , (tomates,5)]).
recette(tartiflette, [(pommes_dt,5), (lardons,5), (sel,2), (fromage,3)])
    .
```

2.2 Relation *peut_preparer(X)*

Il faut également prendre en considération les quantités dans la préparation des repas et donc dans la relation *peut_preparer(X)*. Pour cela, on utilise une relation *tous_disponibles(L)* qui vérifie que tous les éléments de la liste L sont disponibles.

```
peut_preparer(R):- recette(R,L),tous_disponibles(L). %La recette R est
    réalisable avec les ingrédients de la liste L si tous les éléments de
    L sont disponibles.
tous_disponibles([]). %Si la liste L est vide alors c'est bon (cas d'arrêt
    de la récurrence sur la liste)
tous_disponibles([(T,S)|Q]):- disponible(T,X),X#>S,tous_disponibles(Q). %
    Chaque ingrédient est un couple (T,S) avec T la dénomination de l'
    ingrédient et S la quantité nécessaire. On vérifie que l'ingrédient
    est disponible avec la quantité nécessaire pour la recette S
    inférieure à la quantité X disponible our l'ingrédient T. On fait ça
    pour tous les éléments de la liste d'ingrédients par récurrence.
```

Si on passe la quantité de lait disponible à 3, on a les traces d'exécution suivantes :

```
| ?- peut_preparer(gateau).
no
| ?- peut_preparer(puree).

yes
```

On a besoin que de 2 unités de lait pour préparer du lait contre 5 pour préparer un gâteau. On ne peut donc pas préparer de gâteau avec 3 unités de lait.

3 Gestion des quantités

On modifie les fonctions pour prendre en compte la consommation d'ingrédients lors de la réalisation d'une recette. Pour cela, on crée une liste *qte_restantes* avec l'ensemble des ingrédients et leurs quantités respectives. Cette liste sera mise à jour lors de la réalisation d'une recette.

Nous avons tenté d'écrire le code suivant mais il ne fonctionne pas car les changements sur la liste *qte_restantes* ne sont pas pris en compte dans la fonction *tous_disponibles*.

```
qte_restantes([(eau,50),(sucre,100),(sel,50),(vinaigre,4),(huile,3),(
    beurre,8),(sachet_de_the,5),(cafe,10),(chocolat,23),(lait,5),(farine
    ,300),(oeufs,12),(pate_a_crepes,300),(pates_nouilles,300),(laitue,2),(
    pommes_dt,10),(oignon,10),(tomates,40),(lardons,50),(creme,4),(fromage
    ,10)]). %Liste des quantités restantes
```

```
peut_preparer(R):-recette(R,L),qte_restantes(Q),tous_disponibles(L,Q,0).
    %On récupère la liste d'ingrédients nécessaires à la recette et la
    liste des quantités restantes de chaque ingrédient
tous_disponibles([],_,_). %Cas d'arrêt
tous_disponibles([(T,S)|R],[(T,B)|C],_):- S#=<B, X#=B-S, tous_disponibles(
    R,[(T,X)|C],0), qte_restantes([(T,X)|C]). %Si l'ingrédient disponible
     l'est en quantité suffisante, on s'intéresse aux ingrédients suivants
tous_disponibles(L,[T|Q],N):- N1#=N+1, length([T|Q],N2), N1 #=< N2,
    tous_disponibles(L,R,N1). %Sinon, on continue de parcourir la liste d'
    ingrédients tant qu'on a pas atteint la fin de la liste
```

4 Exercice Bonus - Livres

4.1 Base de données livres

On entre la base de données correspondant aux titres des livres, aux auteurs et aux prix associés.

```
livres(auteur('Victor', 'Hugo'), [('Juliette␣Drouet', 32), ('Notre␣Dame␣de
    ␣Paris', 45), ('Les␣Misérables', 35),
('Quatre␣Vingt␣Treize', 24), ('Feuilles␣d␣automne', 30), ('Les␣
    Contemplations', 25)]).
livres(auteur('Léo', 'Ferré'), [('Testament␣Phonographe', 25), ('La␣
    méthode', 25), ('Benoit␣Misère', 30)]).
livres(auteur('Max', 'Weber'), [('Economie␣et␣Société', 24), ('Le␣savant␣
    et␣le␣Politique', 29),
('Théorie␣de␣la␣science', 34), ('La␣bourse', 25)]).
livres(auteur('Blaise', 'Pascal'), [('Pensées', 25), ('De␣l␣esprit␣
    Géométrique', 45)]).
livres(auteur('Confucius', 'Confucius'), [('Confucius', 35), ('La␣morale',
     30), ('Les␣entretiens', 25)]).
livres(auteur('Jacques', 'Lacan'), [('D␣un␣autre␣à␣l␣autre', 30), ('Mon␣
    enseignement', 50)]).
livres(auteur('Sigmund', 'Freud'), [('Sur␣le␣rêve', 30), ('Totem␣et␣Tabou'
    , 25), ('Métapsychologie', 40)]).
livres(auteur('Michel', 'Foucault'), [('Surveiller␣et␣punir', 34), ('
    Histoire␣de␣la␣folie', 25),
('L␣ordre␣du␣discours', 35)]).
livres(auteur('Jacques', 'Derrida'), [('Feu␣la␣cendre', 30), ('Mémoire␣d␣
    aveugle' , 20), ('Voiles', 25 ) ,
('Demeure' , 35), ('Position', 20) ]).
livres(auteur('Michel', 'Serres'), [('Atlas,␣Philosophie␣des␣réseaux', 30)
    , ('Tiers␣Instruit', 25)]).
```

4.2 Requêtes

1- Les auteurs dont les prénoms (ou les noms) sont identiques.

Pour vérifier si des auteurs ont le même prénom, on entre la commande suivante :

```
livres(auteur(X,Y),_),livres(auteur(W,Z),_),X=W,Y\=Z.
```

On souhaite connaître les auteurs ayant le même prénom mais pas le même nom de famille pour connaître les auteurs différents.

La trace d'exécution donne :

```
| ?- livres(auteur(X,Y),_),livres(auteur(W,Z),_),X=W,Y\=Z.
X = 'Jacques'
Y = 'Lacan'
W = 'Jacques'
Z = 'Derrida' ?;
X = 'Michel'
Y = 'Foucault'
W = 'Michel'
Z = 'Serres' ?;
X = 'Jacques'
Y = 'Derrida'
W = 'Jacques'
Z = 'Lacan' ?;
X = 'Michel'
Y = 'Serres'
W = 'Michel'
Z = 'Foucault' ?;
no
```

On obtient donc qu'il y a deux Jacques et deux Michel.

Pour obtenir de même avec les noms de famille, il suffit de changer les conditions à la fin de la commande :

```
livres(auteur(X,Y),_),livres(auteur(W,Z),_),X\=W,Y=Z.
```

Cela ne nous donne aucun résultat.

2- La somme des prix des livres d'un auteur dont on précise le prénom et le nom.

Pour connaître la somme des prix des livres d'un auteur, on part d'une relation *somme(X,Y)* qui prend en argument le nom et le prénom de l'auteur que l'on souhaite. On récupère la liste de ses livres à l'aide du prédicat *livres*. On crée ensuite une nouvelle relation *somme*\4 qui prend en argument le prénom et le nom de l'auteur mais aussi sa liste de livres et la valeur de la somme des prix. Par récurrence sur la liste, on va pouvoir calculer la somme des prix des livres.

```
somme(X,Y):- livres(auteur(X,Y),L),somme(X,Y,L,0). %On récupère la liste
    des livres de l'auteur et on initialise la somme à 0
somme(X,Y,[],S):- write(S). %Quand on arrive à la liste vide, on écrit la
    valeur de la somme (cas d'arrêt de la récurrence)
somme(X,Y,[(T,P)|Q],S):- S1#=S+P,somme(X,Y,Q,S1). %On calcule la somme de
    la valeur temporaire de la somme + du prix du premier livre de la
    liste. Par récurrence on calcule la somme de la même manière sur la
    queue de la liste restante.
```

Les traces d'exécution sont les suivantes :

```
| ?- somme('Victor','Hugo').
191
yes
| ?- somme('Blaise','Pascal').
70
yes
```

3- Le nombre de livres d'un auteur dont on précise le prénom et le nom.

Pour calculer le nombre de livres, on écrit une relation similaire à la précédente sauf qu'au lieu d'ajouter le prix d'un livre à la somme, on a ajoute la valeur 1 à chaque livre considéré.

```
nb_livres(X,Y):- livres(auteur(X,Y),L), nb_livres(X,Y,L,0).
nb_livres(X,Y,[],N):- write(N).
nb_livres(X,Y,[T|Q],N):- N1#=N+1, nb_livres(X,Y,Q,N1).
```

Les traces d'exécution sont les suivantes :

```
| ?- nb_livres('Victor','Hugo').
6
yes
| ?- nb_livres('Michel','Serres').
2
yes
```

4- Le maximum des prix des livres d'un auteur dont on précise le prénom et le nom.

Pour évaluer le maximum des prix des livres d'un auteur, on parcoure la liste des livres de cet auteur. On compare la valeur de chacun des prix des livres à une valeur référence M (initialisée à 0) et si le prix du livre considéré est supérieur à M, on remplace le maximum par cette valeur. À la fin, on obtient le prix maximum pour tous les livres de l'auteur.

Cela correspond au code suivant :

```
prix_max(X,Y):- livres(auteur(X,Y),L), prix_max(X,Y,L,0). %On récupère la
    liste des livres
prix_max(X,Y,[],M):- write(M). %Cas d'arrêt de la récurrence
prix_max(X,Y,[(T,P)|Q],M):- P#>M, prix_max(X,Y,Q,P). %Si le prix du livre
    est supérieur à M, alors P prend la place de M
prix_max(X,Y,[(T,P)|Q],M):- prix_max(X,Y,Q,M). %Sinon on continue de
    parcourir la liste
```

Les traces d'exécutions pour cette fonction :

```
| ?- prix_max('Victor','Hugo').
45
yes
| ?- prix_max('Michel','Foucault').
35
yes
```

5- Les livres d'un auteur dont le prix est inférieur à un prix donné

```
inf_prix(X,Y,R):- livres(auteur(X,Y),L), inf_prix(X,Y,L,R,[]).
inf_prix(X,Y,[],R,L):- write(L).
inf_prix(X,Y,[(T,P)|Q],R,L):- P#<R, inf_prix(X,Y,Q,R,[T|L]). %Si le prix
    est inférieur au seuil de prix fixé P, on ajoute le titre du livre à
    la liste
inf_prix(X,Y,[T|Q],R,L):- inf_prix(X,Y,Q,R,L). %Si le prix est supérieur,
    on ne fait rien et on continue avec la queue de la liste
```

Et les traces d'exécutions :

```
| ?- inf_prix('Victor','Hugo',30).
[Les Contemplations,Quatre Vingt Treize]
yes
| ?- inf_prix('Jacques','Lacan',30).
[]
yes
```

Les deux livres de Victor Hugo inférieurs à 30 euros sont Les Contemplations et Quatre Vingt Treize. Il n'y a pas de livre de Jacques Lacan inférieur à 30 euros.

6- Les titres des livres dont le prix = un certain prix donné en paramètre.

On fait la même relation que précédemment mais au lieu de faire une comparaison d'inégalité, on fait une comparaison d'égalité.

Cela donne le code suivant :

```
egal_prix(X,Y,R):- livres(auteur(X,Y),L), egal_prix(X,Y,L,R,[]).
egal_prix(X,Y,[],R,L):- write(L).
egal_prix(X,Y,[(T,P)|Q],R,L):- P#=R, egal_prix(X,Y,Q,R,[T|L]). %Seul
    changement par rapport au code précédent : le < en =
egal_prix(X,Y,[T|Q],R,L):- egal_prix(X,Y,Q,R,L).
```

Les traces d'exécutions sont les suivantes :

```
| ?- egal_prix('Victor','Hugo',30).
[Feuilles d automne]
yes
| ?- egal_prix('Victor','Hugo',45).
[Notre Dame de Paris]
yes
| ?- egal_prix('Victor','Hugo',42).
[]
yes
```

Les livres de Victor Hugo qui valent 30 et 45 euros sont Feuilles d'automne et Notre Dame de Paris. Il n'y a pas de livre de Victor Hugo qui vaut 42 euros.