UE ELFE - Programmation Fonctionnelle

durée 1h – documents autorisés

Programmation Logique - Interrogation 2

Exercice 1 : *Unification*

Q 1 . Pour chacun des ensembles de termes suivants, dire si les termes qu'il contient s'unifient. Justifier votre réponse, en donnant si besoin l'unificateur le plus général.

a. $E_1 = \{p(Y, X, h(Y)); p(g(a) + Z, f(V), V)\}$

b. $E_2 = \{p(1 + X + 7, Z + (2 + V)); p(Y + Z, Y)\}$

c. $E_4 = \{p([X|Y], [Y|Z], V); p([a, b, c], W, [X, W])\}$

Q 2 . Donnez un unificateur de E_1 qui n'est pas l'unificateur *le plus général*.

Exercice 2 : *Anagrammes*

Une *anagramme* est un mot obtenu par permutation des lettres d'un mot. Par exemple le mot *elisa* possède comme anagrammes *ailes,asile,esial,salies,...*. Bien sûr, parmi ces anagrammes certains mots n'ont aucun sens en français, comme par exemple *aeils*

On suppose que le prédicat $mot(+M)$ est défini par une ensemble de *faits* qui représente l'ensemble des mots d'un dictionnaire. Chaque mot est représenté par la liste de ses lettres.

```
mot([a,i,l,e,s]).
mot([a,i,m,e,r]).
mot([a,s,i,l,e]).
mot([e,l,i,s,a]).
mot([m,a,r,i,e]).
```

Q 1 . Réalisez le prédicat $select(?Liste, ?Element, ?ListeSansElement)$: ce prédicat doit être vrai si la suppression d'une occurrence de l'élément $Element$ dans la liste $Liste$ donne la liste $ListeSansElement$. Vous n'utiliserez aucun prédicat prédéfini.

Q 2 . Construire l'arbre de résolution du but $? - select(L, x, [a, b])$.

Q 3 . Proposez un prédicat $permutation(+L, ?LP)$: ce prédicat doit être vrai si la liste LP est une permutation de L. Vous utiliserez uniquement comme prédicat extérieur le prédicat $select$.

Q 4 . Proposer un prédicat $anagramme(+Mot, ?X)$ satisfait si X est un mot et une anagramme du mot Mot.

Le prédicat $findall(+Var, +But, -Listedobjets)$ produit une liste de toutes les substitutions portant sur Var qui satisfont le But. Par exemple $findall(E, pere(pierre, E), L)$ unifie L avec la liste des enfants de $pierre$. On peut également écrire la différence entre deux ensemble $E_1 \backslash E_2$ comme

```
difference(E1,E2,Res) :- findall(X, (member(X,E1),\+ member(X,E2)),Res).
```

Q 5 . Proposez un prédicat $dico(-D)$ qui unifie D avec la liste des mots du dictionnaire.

Q 6 . Proposez un prédicat $anagrammes(+Dico, -Groupes)$ qui unifie $Groupes$ avec la liste des groupes d'anagrammes des mots du dictionnaire $Dico$. Il faut que tous les mots d'un groupe soient anagrammes entre eux. Un groupe sera représenté par une liste de mots. Le résultat ne doit pas être redondant. Par exemple le but suivant donne :

```
?- dico(D),anagrammes(D,X).
D = [[a,i,l,e,s], [a,i,m,e,r], [a,s,i,l,e], [e,l,i,s,a], [m,a,r,i,e]]
X = [[[a,i,l,e,s], [a,s,i,l,e], [e,l,i,s,a]], [[a,i,m,e,r], [m,a,r,i,e]]];
```

Solutions

Exercice 1

Q 1 .

Solution

a. $\{Y \mid g(a) + Z \; ; \; X \mid f(h(g(a) + Z)) \; ; \; V \mid h(g(a) + Z)\}$

b. $\{pasd'unificationpossible\}$ si Z variable unique liée entre les deux prédicats, sinon $\{X \; ; \; 2 + V \; ; \; Y \mid 1 + (2 + V) \; ; \; Z_1 \mid 1 \; ; \; Z_2 \mid 7\}$

c. $\{X \mid a \; ; \; Y \mid [b, c] \; ; \; V \mid [a, [[b, c]|Z] \; ; \; W \mid [[b, c]|Z]\}$ si X variable unique liée entre les deux prédicats, sinon $\{X_1 \mid a \; ; \; Y \mid [b, c] \; ; \; V \mid [X_2, [[b, c]|Z] \; ; \; W \mid [[b, c]|Z]\}$

Q 2 .

Solution il suffit en plus de substituer la seule variable libre Z par un terme quelconque.

Exercice 2

Q 1 .

Solution Ce prédicat s'appelle "insert" dans le TP4, à une permutation de paramètres prés ... il est facile de l'obtenir...

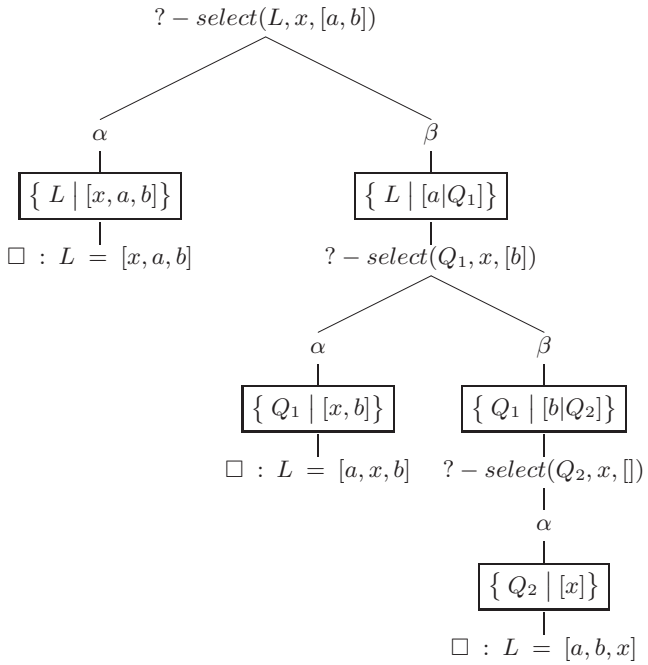
```
%select(?Liste,?Element,?ListeMoinsElement)
select([Element|Queue],Element,Queue).
select([Tete|Queue],Element,[Tete|QueueRes]) :- select(Queue,Element QueueRes).
```

Q 2 .

Solution Cet arbre de résolution a l'avantage d'avoir une liste de but à chaque noeud ne comportant qu'un seul élément, ça évitera les erreurs d'étourderie par oubli.

$\alpha \quad select([E_i|Q_i], E_i, Q_i).$

$\beta \quad select([T_i|Q_i], E_i, [T_i|Qr_i]) : -select(Q_i, E_i, Qr_i).$



Q 3 .

Solution

```
%permutation(+Liste,-ListePermutee).
permutation([], []).
permutation(Liste, [Element|RestePermute]) :-
select(Liste,Element,Reste), permutation(Reste, RestePermute).
```

Q 4 .
Solution

```
anagramme(Mot,X) :- mot(X), permutation(X,Mot).
```

Q 5 .
Solution

```
dico(D) :- findall(M,mot(M),D).
```

Q 6 .
Solution

```
% on a besoin de la difference entre ensembles
% E1 \ E2
difference(E1,E2,Res) :- findall(M,(member(M,E1),\+ member(M,E2)),Res).
%anagrammes(+Dico,-Groupes)
anagrammes([],[]).
anagrammes([MotDico|ResteDico],[Anagrammes|ResteListegrammes]):-
    findall(Mot,anagramme(MotDico,Mot),Anagrammes),
    difference(ResteDico,Anagrammes,ResteDicoClean),
    anagrammes(ResteDicoClean,ResteListeAnagrammes).
```