Expression Logique et Fonctionnelle . . . Évidemment

DS de Programmation Logique

Durée 1h30. Documents de cours autorisés.

Exercice 1: *Augmentation*

Question 1. On donne le prédicat suivant :

```
%ajoute(Objet,L1,L2) vrai ssi L2 peut etre obtenue en ajoutant
% l'Objet a L1 en n'importe quelle position
%regle 1
ajoute(Objet,L1,[Objet|L1]).
%regle 2
ajoute(Objet,[X|L1],[X|L2]):- ajoute(Objet,L1,L2).
```

Avec quelles têtes de règles s'unifient les termes suivants ? Précisez les substitutions permettant d'obtenir l'unificateur le plus général.

1. `?- ajoute(o,Y,[p,r,o,l,o,g]).`

   `?- ajoute(e,[e,l,f],Z).`

3. `?- ajoute(a,Y,Z).`

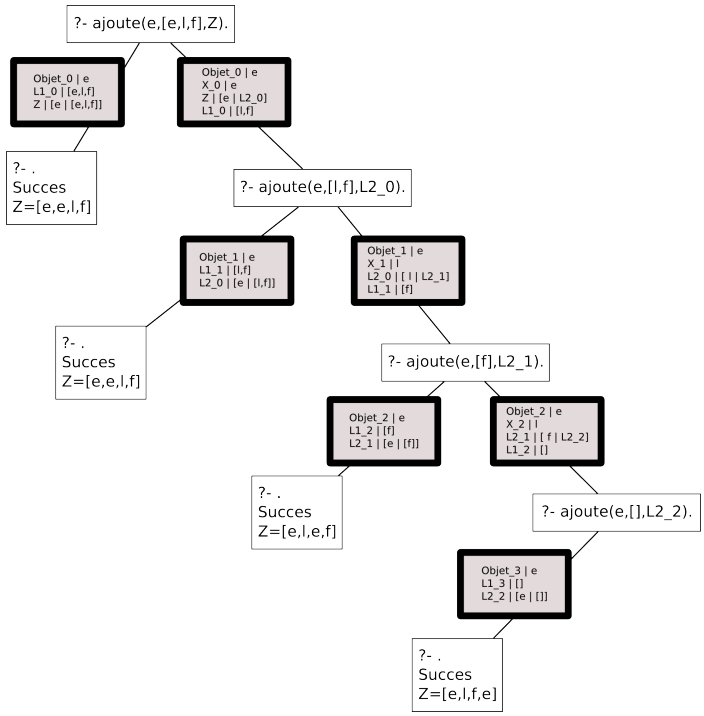
Réponse 1.0.

- ce terme s'unifie avec la tête de la seconde règle.
  `Objet | o`
  `Y | [p | L1]`
  `X | p`
  `L2 | [r,o,l,o,g]`
- ce terme s'unifie avec la règle 1.
  `Objet | e`
  `L1 | [e,l,f]`
  `Z | [e,e,l,f]`
  ce terme s'unifie avec la règle 2
  `Objet | e`
  `X | e`
  `L1 | [l,f]`
  `Z | [ e | L2 ]`
- ce terme s'unifie avec la règle 1.
  `Objet | a`
  `Y | L1`
  `Z | [a | L1]`
  Ce terme s'unifie avec la règle 2.
  `Objet | a`
  `Y | [X | L1]`
  `Z | [X | L2]`

Question 2. Donnez l'arbre de résolution du but :

```
?- ajoute(e,[e,l,f],Z).
```

Réponse 1.0.



Question 3. Que dire de la résolution du but ci-dessous ? Justifiez votre réponse.

```
?- ajoute(a,Y,Z).
```

Réponse 1.0. La résolution sera infinie, on a vu précédement que ce terme s'unifie avec la règle 2. Le nouveau but à résoudre sera `?- ajoute(a,L1,L2).` qui est le même que `?- ajoute(a,X,Z).` au renommage des variables près !

Question 4. On dit qu'une liste $M2$ augmente la liste $M1$, s'il existe un objet X, et une liste L tels que L soit une permutation de $M1$, et $M2$ soit obtenue en ajoutant X en position quelconque de L.

Implantez le prédicat `augmente(M1,M2)`.

Voici un exemple de comportement attendu :

```
?- augmente([t,a,s],[s,t,a,r]).
Yes
?- augmente([p,i,e],[s,t,a,r]).
No
```

On pourra utiliser le prédicat `permutation` vu en tp, mais il faut rappeler son code.

Réponse 1.0.

```
permutation([],[]).
permutation([X|L1],R):-permutation(L1,Aux),ajoute(X,Aux,R).
augmente(M1,M2):- permutation(M1,Aux),ajoute(_,Aux,M2).
```

Question 5. On dispose d'un dictionnaire formé par une liste de faits de la forme :

```
dicos([a,s]).      dicos([r,a,t,e,s]).
dicos([t,a,s]).    dicos([t,a,r,e,e,s]).
dicos([s,t,a,r]).  dicos([a,r,r,e,t,e,s]).
dicos([r,a,s,e]).  dicos([t,e,r,r,a,s,s,e]).
```

Donnez un prédicat `mot_longueur(+N,?X)` où N est un entier et où X est unifié avec les mots du dictionnaire de longueur N.

Par exemple :

```
?- mot_longueur(4,X).
X=[s,t,a,r];
X=[r,a,s,e];
No.
```

Réponse 1.0.

```
mot_longueur(N,X):-dicos(X),length(X,N).
```

Question 6. On souhaite maintenant savoir s'il existe au moins un mot de longueur 7. Quel but doit-on faire résoudre par `Prolog` ?

Réponse 1.0.

```
?- mot_longueur(7,_).
```

Question 7. On souhaite maintenant obtenir le premier mot rencontré dans le dictionnaire de longueur 4. Quel but doit-on faire résoudre par `Prolog` ?

Réponse 1.0.

```
?- mot_longueur(4,X),!.
```

Question 8. Réalisez un prédicat `cherche(+Mot1,+Mot2,?Liste)` qui unifie la liste avec une liste des mots du dictionnaire (s'il en existe) telle que

- la tête est $Mot1$
- la liste se termine par $Mot2$
- tout couple de mots consécutifs X,Y de la liste soient augmentés, c'est-à-dire que `augmente(X,Y)` soit satisfait.

Par exemple :

```
?-cherche([a,s],[t,e,r,r,a,s,s,e],X).
X=[[a,s],[t,a,s],[s,t,a,r],[r,a,t,e,s],[t,a,r,e,e,s],[a,r,r,e,t,e,s],[t,e,r,r,a,s,s,e]];
No.
```

Réponse 1.0.

```
%cherche(Motdepart,MotArrive,Liste)
cherche(Motdepart,MotArrive,Liste):-cherche_aux(Motdepart,MotArrive,[],Liste).

cherche_aux(MotCourant,MotCourant,ListeEnConstruction,Lres):-
  reverse([MotCourant|ListeEnConstruction],Lres),!.
cherche_aux(MotCourant,MotArrive,ListeEnConstruction,Lres):-
  augmente(MotCourant,MotSuivant),
  dicos(MotSuivant),
  cherche_aux(MotSuivant,MotArrive,[MotCourant|ListeEnConstruction],Lres).
```