Programmation Logique et Fonctionnelle - TP4
PROLOG

1 Coupure

1.1

Spécifiez le prédicat `supprimeDoublons/2` tel que si `L` est une liste complètement instanciée et `S` est une variable non instanciée alors `supprimeDoublons(L,S)` est satisfait par l'instanciation de `S` avec une liste comportant les mêmes éléments que `L`, mais avec une seule occurrence de chacun de ces éléments.
Peut on utiliser ce prédicat pour vérifier, dans le cas où `L` et `S` sont complètement instanciées, que `S` contient exactement une occurrence de chaque élément de `L` ?

1.2

Soit la spécification suivante du prédicat `retire/3` :

```
retire(_,[],[]).
retire(X,[X|Q],R) :- retire(X,Q,R).
retire(X,[T|Q],[T|R]) :- T\=X, retire(X,Q,R).
```

Essayez le but `retire(1,[1,2,1,3],R).`, puis recommencez l'expérience en supprimant `T\ =X` de la troisième clause. Déterminez où placer une coupure pour retrouver le comportement souhaité, à savoir que R doit être instanciée avec la liste donnée en deuxième argument dans laquelle les occurrences du premier argument ont été retirées.

1.3

Spécifiez un prédicat monnaie/3 qui, étant donnée une liste de pièces et/ou billets, dont les valeurs sont représentées par des entiers, et un prix, également représenté par un entier, donne toutes les manières possibles d'obtenir le prix en utilisant tout ou partie de la monnaie disponible.
Par exemple, le but `monnaie([1,1,2,2,5,6],6,R).` devrait produire les listes `[1,1,2,2]` et `[1,5]` (chacune de ces listes pouvant être produite plusieurs fois).
Placez une coupure de manière à limiter la recherche à la première solution trouvée.

2 Négation

2.1

On se propose de représenter des ensembles à l'aide de listes, en ne tenant pas compte de l'ordre des éléments et en supposant que chaque élément a une seule occurrence.
Spécifiez un prédicat `diff/3` tel que `diff(A,B,D)` permette de calculer la différence ensembliste `D` entre `A` et `B`, i.e., l'ensemble des éléments de `A` qui ne sont pas dans `B`.
Proposez une version utilisant une coupure et une version utilisant l'opérateur `not`.

2.2

Reprenez le premier exercice du TP1 de programmation logique et proposez un moyen de rechercher tous les ancêtres d'asterix qui n'ont pas de frère.

3 Approfondissement

Spécifiez un ensemble de prédicats permettant de réaliser le tri d'une liste d'entiers en vous basant sur l'algorithme du tri rapide. Lorsque la liste à trier comporte plus d'un élément, cet algorithme la sépare en deux sous-listes contenant respectivement les éléments inférieurs ou égaux et les éléments supérieurs à la valeur du premier élément. Chaque liste est ensuite triée récursivement, et les résultats sont concaténés.