Prolog – TP 1

2017-2018

Exercice 1 : arbre généalogique

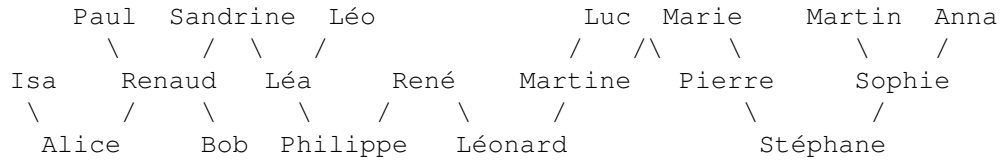
1. Soumettez à Prolog le programme suivant :

```
femme(sandrine).
femme(alice).
homme(paul).
homme(renaud).

%% pere(Fils, Pere).
pere(renaud, paul).
pere(alice, renaud).
```

2. Quel est le résultat de la requête `pere(X,Y).` ?

3. Saisissez les faits correspondant à l'arbre généalogique suivant :



4. Définissez les prédicats suivants : `parent`, `mere`, `enfant`, `grandparent`, `frere`/`soeur`, `oncle`/`tante`, `cousin`, etc.
   La requête `frere(X,Y)` doit retourner `X = martine,Y = pierre`.
   La requête `oncle(X,Y)` doit retourner `X = leonard,Y = pierre`.
   Que retournent les requêtes `tante(X,Y)` et `cousin(X,Y)` ?

Exercice 2 : manipulation de listes

Nous rappelons qu'une liste vide se note `[]` et qu'une liste non vide se note `[ X | Y ]` (où `X` est le premier élément), `[ X,Y | Z ]` (où `X` et `Y` sont les deux premiers éléments), etc.

1. Définissez le prédicat `member(Elt,L)` qui est vrai si `Elt` est présent au moins une fois dans `L`.
   `member(X,[])` doit renvoyer `false`.
   `member(X,[a,Y,[]])` doit renvoyer `X = a; X = Y; X = []`.
   `member(c,[a,b,c])` doit renvoyer `true`.
   `member(k,[a,b,c])` doit renvoyer `false`.

2. Définissez le prédicat `concat(L1,L2,L3)` qui est vrai si `L3` est la concaténation des listes `L1` et `L2`.
   `concat(X,Y,[])` doit retourner `X = Y,Y = []`.
   `concat([a,b,c],[],L)` doit retourner `L = [a,b,c]`.
   `concat(A,B,[e,4])` doit retourner `A = [],B = [e,4] ; A = [e],B = [4] ; A = [e,4],B = []`.

3. Définissez le prédicat `sum(Liste,Sum)` qui est vrai si `Sum` est la somme des nombres contenus dans `Liste`.

4. Définissez le prédicat `occ(Elt,Liste,N)` qui est vrai si `Elt` est présent `N` fois dans `Liste`.
5. Définissez un prédicat `liste_entiers(N,L)` qui est vrai si `L` est la liste des `N` premiers entiers.
6. Définissez le prédicat `inverse(L1,L2)` qui est vrai si la liste `L1` est l'inverse de la liste `L2`.
7. Redéfinissez `inverse` en utilisant une liste intermédiaire.
8. Définissez le prédicat `palindrome(L)` qui est vrai si `L` est un palindrome, de 2 manières différentes : avec et sans `append`.

Exercice 3 : tri de listes

1. Définissez un prédicat `inserer(Elt,L1,L2)` qui insère `Elt` dans la liste triée `L1`, `L2` est la liste triée obtenue après insertion. Définir un prédicat `tri_insertion(L,L_triee)` en utilisant `inserer`.
2. Définissez un prédicat de tri par sélection `tri_selection(L,L_triee)`. les plus petits éléments seront successivement placés au début de la liste.
3. Définissez un prédicat de tri fusion `tri_fusion(L,L_triee)` : la liste à trier est divisée en 2, chaque partie est triée, puis les 2 parties triées sont fusionnées.

Exercice 4 : contraintes

1. Trouvez une affectation entre 0 et 9 pour les lettres S,E,N,D,M,O,R,Y de sorte que :

```
    SEND
+   MORE
= MONEY
```

   Chaque lettre doit avoir une valeur différente des autres. M et S ne peuvent pas avoir la valeur 0.

2. Générez un carré magique normal d'ordre 3.

   > Un carré magique d'ordre n est composé de n^2 entiers strictement positifs, écrits sous la forme d'un tableau carré. Ces nombres sont disposés de sorte que leurs sommes sur chaque rangée, sur chaque colonne et sur chaque diagonale principale soient égales. Un carré magique normal est un cas particulier de carré magique, constitué de tous les nombres entiers de 1 à n^2, où n est l'ordre du carré. (Wikipedia)