

Projet

Exercice 1 : Agence matrimoniale

Soit une agence matrimoniale qui possède un fichier de candidats au mariage. Ce fichier est organisé sous la forme d'un ensemble de faits de la forme :

homme(N, T, C, A).

femme(N, T, C, A).

où N est le nom d'un homme ou d'une femme, T sa taille (grande, moyenne ou petite), C la couleur de ses cheveux (blonds, bruns, roux, châains), A son âge (jeune, mur ou vieux).

Le prédicat `gout(N, M, L, S)` indique que la personne N aime le genre de musique M (classique, pop, jazz), le genre de littérature L (aventure, science-fiction, policier) et pratique le sport S (tennis, natation, jogging).

Le prédicat `recherche(N, T, C, A)` indique que la personne N recherche un partenaire de taille T, ayant des cheveux de couleur C et dont l'âge est A.

1. Définir un ensemble de faits représentant 10 hommes et 10 femmes.

Remarque : il est laissé à votre discrétion de choisir les noms et les caractéristiques des candidats.

On considère que deux personnes X et Y de sexes différents sont assorties si X convient à Y et si Y convient à X. On dira que X convient à Y si d'une part X convient physiquement à Y (la taille, l'âge et la couleur des cheveux de X sont ceux que Y recherche) et si d'autre part les goûts de X et Y en matière de musique, littérature et sport sont identiques.

2. Définir le prédicat `convient-physiquement(X, Y)` en suivant les informations données précédemment.
3. Définir le prédicat `meme_gout(X, Y)` en suivant les informations données précédemment.
4. Ecrire le programme permettant de déterminer les couples assortis.

Exercice 2 : Cryptarithmétique

On considère l'addition suivante : **LOGIC + LOGIC = PROLOG** où chaque lettre représente un chiffre différent (compris entre 0 et 9). On souhaite connaître la valeur de chaque lettre, sachant que la première lettre de chaque mot représente un chiffre différent de 0.

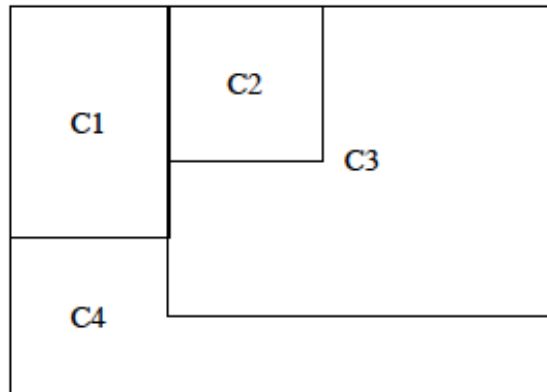
Résoudre ce problème en Prolog.

Indications : Chaque élément de l'addition est représenté sous la forme d'une liste : LOGIC est représenté par la liste [L,O,G,I,C].

Bonus : Vérifier que votre programme fonctionne aussi avec l'opération : ZERO + ZERO = RIEN

Exercice 3 : Coloriage de carte

On souhaite définir un prédicat permettant de colorier la carte suivante :



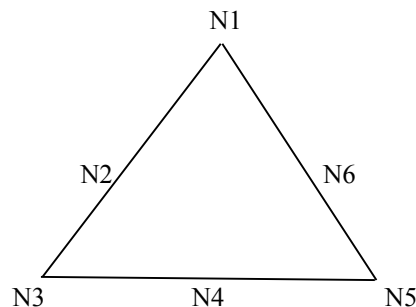
Les règles sont les suivantes :

- On dispose de trois couleurs qui sont : vert, jaune et rouge ;
- Deux zones contiguës doivent avoir des couleurs différentes.

1. Ecrivez un prédicat `coloriage(C1, C2, C3, C4)` qui comportera deux parties :
 - La première partie génère toutes les valeurs possibles de C1, C2, C3 et C4.
 - La seconde vérifie si les colorations obtenues sont conformes à la carte par l'utilisation du prédicat $X \neq Y$ sur les couleurs des zones contiguës.
2. Reprenez ce prédicat, et modifiez le programme en déplaçant les tests de différence de couleurs le plus tôt possible dans l'écriture du prédicat, c'est-à-dire en vérifiant les différences de couleurs dès que celles-ci sont instanciées. Quelle en est la conséquence ?

Exercice 4 : Nombres en triangle

On dispose les nombres de 1 à 6 en triangle, en n'utilisant qu'une fois chaque nombre



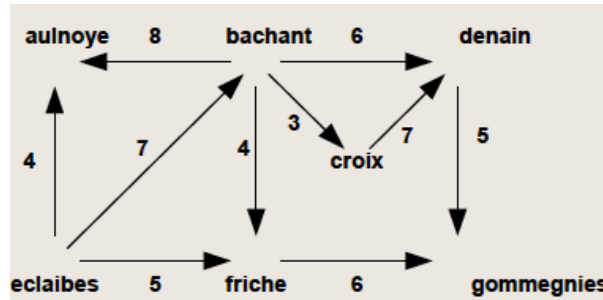
Il s'agit de trouver toutes les manières de placer les nombres sur le triangle de telle sorte que la somme des trois nombres figurant sur le côté soit la même pour les trois côtés du triangle.

1. Définir le prédicat `choisir(X, L)`, qui est vérifié si X est un élément de L.
2. Définir le prédicat `permutation(L, LP)`, qui est vérifié si LP est une version permutée de la liste L.
3. Définir le prédicat `generer(L)`, qui est vérifié si L est une permutation de la liste [1,2,3,4,5,6].
4. Définir le prédicat `tester(L)`, où L est de la forme [N1,N2,N3,N4,N5,N6], qui vérifie que la somme de chaque côté du triangle telle que définie sur la figure précédente est égale.

5. Définir le prédicat `solution(X)`, qui est vérifié si `X` définit un triangle « valide ».
6. Combien y a-t-il de solutions ?
Remarque : Pensez à utiliser le prédicat `bagof`.

Exercice 5 : Parcours de graphe

Supposons le graphe suivant :



1. Décrire la base de faits sous la forme `arc(bachant, aulnoye)` décrivant cette carte.
2. Décrire les règles permettant de répondre aux questions :
 - Peut-on aller de eclaiibes à croix ?
 - Comment aller de eclaiibes à croix ?
 - Quels sont les chemins menant à aulnoye ?
 - Comment aller de eclaiibes à gommegnies en passant par deux vias ?
 - Comment aller de eclaiibes à gommegnies avec un minimum de deux étapes ?

Remarque : On décrira la règle `chemin(A, B, Liste)`, `Liste` étant l'ensemble des villes entre `A` et `B`.

3. Définissez maintenant le prédicat `arc` en faisant intervenir le coût (exemple : `arc(friche, gommegnies, 6)`).
4. Définissez la règle `chemin(A, B, Liste, Cout)` donnant, pour chaque chemin, la liste des villes entre `A` et `B` ainsi que le coût du voyage.
5. Transcrire les questions :
 - Quels sont les chemins entre bachant et gommegnies passant par denain, et quels sont leurs coûts ?
 - Quels sont les chemins partant de eclaiibes avec un coût supérieur ou égal à 11 ?
6. Ecrire maintenant la règle `voisin(A, B)` permettant de définir si deux villages ou deux villes sont voisins.
7. Ecrire la règle `chemin(A, B, Trace, Liste)` où `Trace` est la liste des villes déjà visitées.
8. Afficher les chemins possibles entre aulnoye et croix.
9. Ajouter maintenant la prise en compte du coût.
10. Afficher les chemins entre aulnoye et gommegnies de coût inférieur à 20.

Consignes : Le projet se fait en binôme (les trinômes (ou plus) sont interdits).

Questions : pour toutes questions, n'hésitez pas à me contacter.

Si vous avez une mailing list pour la promo, n'hésitez pas à la mettre en copie afin que tout le monde puisse bénéficier de la réponse.

Soumission : Les soumissions doivent être **envoyées par mail** à ens.courtine@gmail.com. Votre archive doit impérativement être nommée de la manière suivante : **Nom1_Nom2.zip** (personnalisé avec les informations vous concernant) et se composera du fichier **PL** (Prolog) et d'un fichier **TXT** avec l'ensemble de vos tests.

Date de rendu : **Samedi 8 février 2014 à minuit.**

Toute soumission « en retard », incomplète, erroné, ... est équivalente à un 0.

Le fichier **Prolog** doit être dûment commenté, la clarté de votre programme est un critère important dans la notation. Pour commenter vos prédicats Prolog, vous pouvez utiliser la représentation suivante :

```
% Titre : Affichage du prédicat pere. (question 10)
% Prédicat : ecritPere/2
% Usage : ecritPere(?Pere, ?Fils) est satisfait si Pere est le père de Fils.
ecritPere(Pere, Fils):- ...
```

Le **fichier de tests** est un fichier TXT dans lequel vous montrez sur des exemples ***pertinents personnels*** que l'ensemble de vos prédicats fonctionne correctement. Ce fichier doit donc être commenté et contenir des questions et les réponses associées de Prolog.
