Rattrapage de programmation logique

29 juin 2016

La clarté et la précision de la rédaction seront prises en compte dans l'évaluation.
Le barème est indicatif.
Aucun document autorisé.
Machines (ordinateurs, téléphones, montres connectées et calculatrices) interdites.

Aucune question ne pourra être posée durant l'examen. En cas de doute concernant le sujet, vous expliquerez vos hypothèses.

Les réponses non justifiées ne seront pas prises en compte

Durée : 2h

1 Mise en jambes /4 pt

Écrire les 2 prédicats suivants :

1. `avantdernier` qui retourne vrai si la valeur en sortie (deuxième paramètre) est l'avant dernier élément d'une liste en entrée (premier paramètre).

   ```
   ?- avantdernier([1,3,6,3,2],E).
   E = 3
   ```

2. `inverse` qui retourne vrai si la valeur en sortie (deuxième paramètre) est la liste inversée des éléments d'une liste en entrée (premier paramètre).

   ```
   ?- inverse([1,3,6,3,2],L).
   L = [2,3,6,3,1]
   ```

2 Fizz Buzz /5 pt

Écrire un prédicat `fizzbuzz` qui retourne vrai si la valeur en sortie (deuxième paramètre) est une liste contenant la suite de valeur correspondant au jeu du Fizz Buzz jusqu'à une valeur passée en entrée (premier paramètre). Le jeu du Fizz Buzz consiste à énumérer les entiers strictement positifs en remplaçant les multiples de 3 par fizz, les multiples de 5 par buzz et les multiples de 3 et 5 par fizzbuzz.

```
?- fizzbuzz(16,L).
L = [1,2,fizz,4,buzz,fizz,7,8,fizz,buzz,11,fizz,13,14,fizzbuzz,16]
```

3 Coloration /4 pt

Écrire un prédicat `coloration` qui retourne vrai si la liste en sortie (second paramètre non instancié) représente une coloration d'un graphe de N sommets numérotés de 1 à N (premier paramètre) avec le minimum de couleurs. Deux sommets adjacents ne peuvent pas avoir la même couleur. On considèrera que le graphe est modélisé par les prédicats `arete(X,Y)`, indiquant qu'il y a une arête entre les sommets `X` et `Y`. Les couleurs seront codées par des entiers consécutifs commençant à 1.

```
?- listing(arete),coloration(4,C).
arete(1, 2).
arete(2, 3).
arete(3, 4).
arete(4, 1).
C = [1, 2, 1, 2]
```

4 Google Code Jam/7 pt

Après des années d'études, les scientifiques du Google Labs ont découvert un langage alien transmis depuis une planète lointaine. Le langage alien est vraiment unique, puisque chaque mot est constitué d'exactement **L** lettres minuscules. De plus, il y a exactement **D** mots dans ce langage.

Une fois le dictionnaire de tous les mots du langage alien construit, il fallut décrypter tous les messages reçus depuis des décennies. Malheureusement, en raison de la distance, le signal fut parfois altéré et quelques mots risquent d'être mal interprétés. Pour aider au décodage, les scientifiques vous demandent de leur écrire un programme prolog qui détermine le nombre d'interprétations possibles pour un message donné.

Un message est constitué d'exactement **L** motifs devant correspondre aux **L** lettres. Chaque motif correspond à la liste des lettres possibles. Par exemple, si **L=3** : `[[a,b],[d],[d,c]]` signifie que la première lettre est soit `a` soit `b`, la seconde est `d` et la dernière soit `d` soit `c`. Donc le message `[[a,b],[d],[d,c]]` peut correspondre à une des 4 possibilités : `add,adc,bdd,bdc.`

1. Proposer une solution pour stocker le dictionnaire des mots du langage alien.
2. Écrire un prédicat `enum_alien/2` qui génère un mot possible à partir d'un message composé de motifs.

   ```
   Exemple:
   ?- enum_alien([[a,b],[d],[d,c]],E).
   E = [a, d, d] ;
   E = [a, d, c] ;
   E = [b, d, d] ;
   E = [b, d, c].
   ```

3. Écrire un prédicat `count_alien/2` qui détermine le nombre de mots possibles appartenant au langage alien à partir d'un message constitué de motifs.

   Exemple si `add` et `bdd` appartiennent au langage alien, mais ni `adc`, ni `bdc` :

   ```
   ?- count_alien([[a,b],[d],[d,c]],C).
   C = 2.
   ```