Feuille de TD PLC

Michel Rueher

1 n Reines

Le problèmes des n reines consiste à placer n reines sans que deux reines soient en prise.

En Prolog
a) Ecrire un programme Prolog qui calcule les positions des reines sur un échiquier de taille n (on écrira d'abord une version totalement naïve, puis on essayera de l'optimiser);
b)Tester ces programmes pour n=8, 16, 24.

En Prolog avec contraintes
a) Adaptez le programme `Prolog` pour utiliser les contraintes sur les domaines finis lors du calcul des positions des reines.
b) Comparer les performances avec celles du programme `Prolog` pour n=8, 16, 24.

2 send+more=money

Le problème "send+more=money" consiste à trouver les chiffres associés aux lettres [S,E,N,D,M,O,R,Y] pour que l'équation *send+more=money* soit vérifiée. (toutes les lettres doivent avoir une valeur différente; S et M doivent être différents de 0).

En Prolog
a) Ecrire un programme Prolog qui calcule la valeur des lettres [S,E,N,D,M,O,R,Y](on écrira d'abord une version totalement naïve, puis on essayera de l'optimiser);
b) Tester ces programmes et comparer les performances.

En Prolog avec contraintes

Exécutez incrémentalement le programme ci-dessous qui utilise les contraintes sur les domaines finis pour calculer la valeur des lettres [S,E,N,D,M,O,R,Y]. Comparer les performances avec un programme `Prolog` qui calcule la valeur des lettres [S,E,N,D,M,O,R,Y].

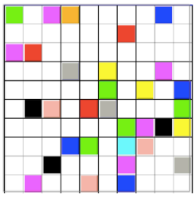
```
mm([S,E,N,D,M,O,R,Y]) :-
     fd_domain([S,E,N,D,M,O,R,Y], 0, 9),      % step 1
     S#>0, M#>0,
     fd_all_different([S,E,N,D,M,O,R,Y]),     % step 2
     sum(S,E,N,D,M,O,R,Y),
     fd_labelingff([S,E,N,D,M,O,R,Y]).      % step 3

sum(S, E, N, D, M, O, R, Y) :-
                  1000*S + 100*E + 10*N + D
     +            1000*M + 100*O + 10*R + E
     #= 10000*M + 1000*O + 100*N + 10*E + Y.

%  ?- mm([S,E,N,D,M,O,R,Y]).
%  D = 7, E = 5,  M = 1, N = 6,  O = 0,  R = 8,  S = 9, Y = 2 ?
```

3 Carré latin

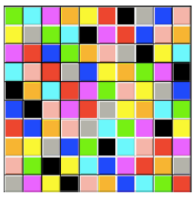
Etant donnée n couleurs, un carré latin (ou quasigroup) d'ordre n est un carré $n \times n$ colorié tel que :

- Toute cellule est coloriée
- Chaque couleur apparaît exactement une fois sur chaque ligne

- Chaque couleur apparaît exactement une fois sur chaque colonne



32% de cellule colorées



10^{68} complétions possibles

Carré latin

Lorsque une affectation partielle de couleurs est donnée, le problème est de savoir si le carré latin partiel peut être complété en un carré latin complet.

1. Ecrire un programme qui génère un carré latin de n couleurs
2. Ecrire un programme qui complète (lorsque c'est possible) un carré latin de n couleurs partiellement colorié.
3. Modifiez ce programme pour colorier une carte avec 4 couleurs (on représentera la carte par un graphe de voisinage).

4 Coloriage

On suppose qu'on dispose d'une liste de pays de la forme :

```
nom_pays([allemagne,france,italie,autriche,suisse]).
```

et de relations de voisinage de la forme :

```
adj(allemagne,france).  adj(allemagne,suisse).  ....
```

Écrire un programme qui colorie la carte correspondante avec 4 couleurs au plus en garantissant que la même couleur ne soit pas associée à deux pays voisins.

5 sudoku

Les règles du sudoku: un plateau de sudoku est composé d'une grille de 9 cases de coté. Ce plateau est divisé en "carrés" de 3 cases de coté. Certaines des cases du plateau sont vides, d'autres ont déjà des valeurs associées.

Exemple:

8		4				2		9
		9				1		
1			3		2			7
	5		1		4		8	
				3				
	1		7		9		2	
5			4		3			8
		3				4		
4		6				3		1

Le but du jeu de sudoku est simple: il faut compléter la grille en respectant les règles suivantes :

- Chaque case du plateau peut prendre une valeur allant de 1 à 9
- Les cases de chaque ligne ont toutes une valeur différente
- Les cases de chaque colonne ont toutes une valeur différente
- Les cases de chaque carré ont toutes une valeur différente

Ecrire un programme `Prolog` utilisant des contraintes pour résoudre ce problème.