

Programmation Logique - TP3 – 2001 - 2002

Et si on jouait ?

N. Jussien (d'après un sujet de R. Debruyne)

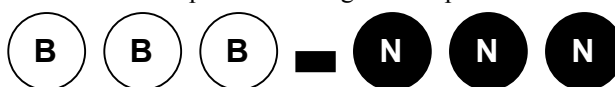
1. Des dominos pour s'échauffer les doigts

Etant donné un ensemble de dominos sous la forme d'une liste de couples d'entiers, on souhaite construire une liste ordonnée de ces dominos telle que deux faces en contact portent le même chiffre (on peut bien évidemment « tourner » un domino si besoin).

- Résoudre le problème en utilisant une stratégie « générer et tester », c'est-à-dire en générant ici l'ensemble des listes ordonnées possibles et en testant pour chacune d'elle si elle possède la propriété de concordance des faces. Essayez votre solution sur des ensembles de dominos de différentes tailles (7, 8 ou 9 dominos par exemple) et critiquez cette approche.
- Trouvez une meilleure approche pour résoudre ce problème.

2. Pions noirs et blancs

On dispose de 3 pions blancs et de 3 pions noirs alignés et séparés :



On souhaite connaître la suite d'opérations à réaliser pour parvenir à un état NNN-BBB sachant qu'on dispose des 4 opérations suivantes :

- glissement à gauche d'un pion noir : BNB-NNB → BNBN-NB
- glissement à droite d'un pion blanc : BNB-NNB → BN-BNNB
- saut à gauche d'un pion noir par dessus un pion blanc : BN-BNNB → BNNB-NB
- saut à droite d'un pion blanc par dessus un pion noir : BBN-NNB → B-NBNNB

Ecrire le prédicat `pions(L)` qui ne réussit que si `L` est une liste d'opérations (désignées par 1, 2, 3 et 4) permettant de passer de l'état BBB-NNN à l'état NNN-BBB.

Suggestion : représentez un état par deux listes spécifiant les pions figurant à gauche et les pions figurant à droite respectivement. Il est plus simple d'ordonner ces listes de telle façon qu'elles commencent par les pions les plus proches de la séparation. Une des possibilités pour résoudre le problème est de passer par un prédicat `transit(LG, LD, S)` qui réussit si la suite d'opération `S` permet à partir de l'état `(LG, LD)` d'aboutir à l'état NNN-BBB.

3. Les carrés magiques

On veut placer sur une matrice $N \times N$ les nombres de 1 à N^2 de telle manière que la somme des nombres d'une ligne quelconque corresponde à celle de n'importe qu'elle autre ligne. De plus, la somme des nombres d'une ligne doit correspondre à la somme des nombres de n'importe qu'elle colonne ou diagonale. Une matrice $N \times N$ remplie de cette façon est appelée un carré magique.

6	1	8
7	5	3
2	9	4

On souhaite résoudre le problème en utilisant la méthode « générer et tester ». On représente une solution potentielle par une permutation de la liste $[1, 2, 3, \dots, N^2]$, les N premiers nombres correspondant à la première ligne, les N suivants à la deuxième, etc.

- Ecrire le prédicat `genere(N, L)` qui n'est vérifié que si `L` est la liste $[1, 2, 3, \dots, N^2]$.
- Ecrire le prédicat `nombreMagique(N, X)` qui n'est vérifié que si `X` est la somme d'une colonne/ligne/diagonale d'un carré magique de côté `N`.
- Vous trouverez dans le fichier `tp-3-prolog.pl` la définition du prédicat `sommeLigne(L, N, I, X)` qui n'est vérifié que si `X` est la somme de la $I^{\text{ème}}$ ligne de la matrice $N \times N$ `L`. Ecrivez le prédicat `lignesOK(L, N, V)` qui n'est vérifié que si pour chaque ligne de la matrice $N \times N$ `L`, la somme de ses nombres vaut `V`.
- Dans le fichier `tp-2-prolog.pl` vous trouverez les définitions des prédicats `sommeColonne(L, N, I, X)`, `sommeDiagonale1(L, N, X)` et `sommeDiagonale2(L, N, X)`. Ecrire le prédicat `carreMagique(N, L)` qui n'est vérifié que si `L` est un carré magique de taille $N \times N$.
- Comment améliorer la rapidité du `carreMagique` ?