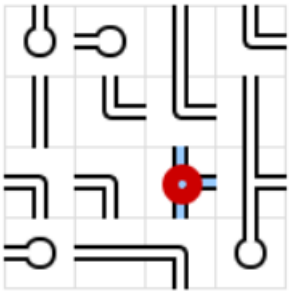
Projet Prolog : Résolution de Pipes

Le jeu de Pipes est un jeu (très) connu où le but est de faire tourner les tuyaux afin que la source atteigne la ou les destination(s) finale(s).

Dans le cadre du projet, nous avons développé un programme en « prolog » qui donne toutes les solutions possibles pour un puzzle de pipes en 4x4.

Nous sommes partis de ce puzzle ci-dessous pour élaborer notre solveur.



Voici, en 3 étapes, comment nous nous y sommes pris :

1° - Affichage cases et tuyaux :

```
1 print_lA([]):- write("caseA1:[1,0,0,0] / case A2: [0,0,0,1] / caseA3:
  [1,0,1,0] / caseA4: [1,1,0,0] \n").
2 print_lB([]):- write("caseB1: [1,0,1,0] / case B2: [1,1,0,0] / case B3:
  [1,1,0,0] / case B4: [1,0,1,0] \n").
3 print_lC([]):- write("caseC1: [0,0,1,1] / caseC2: [0,0,1,1] / caseC3:
  [1,1,1,0] / caseC4: [1,1,1,0] \n").
4 print_lD([]):- write("caseD1: [0,0,0,1] / caseD2: [0,1,0,1] / caseD3:
  [0,0,1,1] / caseD4: [1,0,0,0] \n").
5 print_all([]):- print_lA([]), write("\n"),
6  print_lB([]), write("\n"),
7  print_lC([]), write("\n"),
8  print_lD([]), write("\n").
```

/*Attribution à chaque case une liste de 4 éléments, représentant respectivement l'orientation du tuyau (Nord/Est/Sud/Ouest) où 0 = false et 1 = true. Création de prédicats affichant ligne par ligne les cases du tableau. Print_all([]) fait appel à tous les prédicats définis précédemment.*/

2° - Système de jeu :

```
10 /*rotation des tuyaux + sorties*/
11 rotat([N,E,S,O],[N,E,S,O],0):- write("0 rotation, case suivante : \n").
12 rotat([N,E,S,O],[O,N,E,S],1):- write("1 rotation vers la droite, case
   suivante \n").
13 rotat([N,E,S,O],[S,O,N,E],2):- write("2 rotations vers la droite, case
   suivante : \n").
14 rotat([N,E,S,O],[E,S,O,N],3):- write("3 rotations vers la droite, case
   suivante : \n").
15
16 /*les tuyaux avant et après X fois*/
17 cases_turn([],[],[]).
18 cases_turn([A|Ab], [B|Bc],Y):- rotat(A,B,Y1), cases_turn(Ab,Bc,Y2),
   append([Y1],Y2,Y).
```

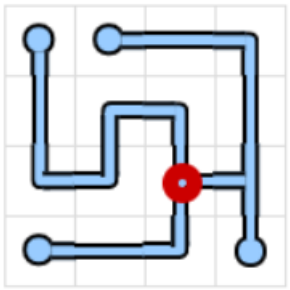
/*Définition du prédicat « rotat » (pour rotation) où l'on passe en argument la liste correspondant à l'orientation du tuyau avant et après rotation et combien a-t-il fait de quarts de tours sur la droite. Quant au prédicat « cases_turn » il sert à déplacer le dernier élément de la liste en tête, ce qui signifie qu'on a fait un quart de tour sur la droite.*/

3° -Implémentation du Puzzle :

```
20 pipes(A,B,C,D):- print_all([]), cases_turn(
21  [[1,0,0,0],[0,0,0,1],[1,0,1,0],[1,1,0,0]],
22  [[0,0,1,0],[0,1,0,0],[0,1,0,1],[0,0,1,1]],A),
23  cases_turn(
24  [[1,0,1,0],[1,1,0,0],[1,1,0,0],[1,0,1,0]],
25  [[1,0,1,0],[0,1,1,0],[0,0,1,1],[1,0,1,0]],B),
26  cases_turn(
27  [[0,0,1,1],[0,0,1,1],[1,1,1,0],[1,1,1,0]],
28  [[1,1,0,0],[1,0,0,1],[1,1,1,0],[1,0,1,1]],C),
29  cases_turn(
30  [[0,0,0,1],[0,1,0,1],[0,0,1,1],[1,0,0,0]],
31  [[0,1,0,0],[0,1,0,1],[1,0,0,1],[1,0,0,0]],D).
```

/*Développement du prédicat de résolution « pipes » avec 4 arguments qui représentent les 4 lignes du tableau, dans ce prédicat, on appelle l'affichage (« print_all »), puis le système de rotation (« cases_turn ») et enfin on intègre le puzzle en spécifiant l'état initial et final de chaque tuyau.*/

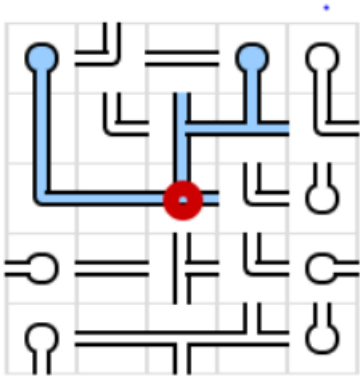
Solution finale du puzzle:

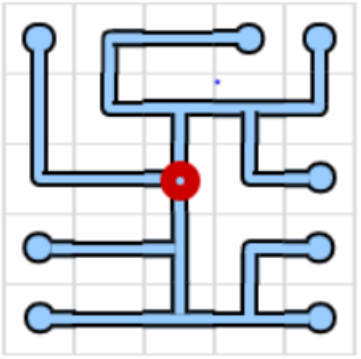


```
A = [2, 2, 1, 2],
B = [0, 1, 2, 0],
C = [2, 1, 0, 2],
D = [2, 0, 1, 0]
```

*/*Le résultat du programme renvoi le nombre de rotations à effectuer pour chaque tuyau pour parvenir à la solution. Notre programme a également trouvé 15 autres solutions pour résoudre le même puzzle.*/*

Voici un autre puzzle résolu en 5x5





```
A = [0, 2, 0, 1, 0],
B = [0, 0, 1, 2, 3],
C = [0, 0, 3, 0, 3],
D = [2, 0, 2, 1, 2],
E = [3, 0, 2, 0, 3]
```

*/*Le résultat du programme renvoi le nombre de rotations à effectuer pour chaque tuyau pour parvenir à la solution. Notre programme a également trouvé 31 autres solutions pour résoudre le même puzzle.*/*

Aymeric Béguinot & Ba Souleymane L1Z