Bonnefon Romain - Desclaux Nicolas - Duboz Emma - Garlatti Laura - Lorgue Paul

---

Intelligence artificielle symbolique

Projet Prolog

Quoridor

Promo 2022

Encadrant : Pierre-Alexandre Favier

Vendredi 9 avril 2021

Table des matières

Introduction

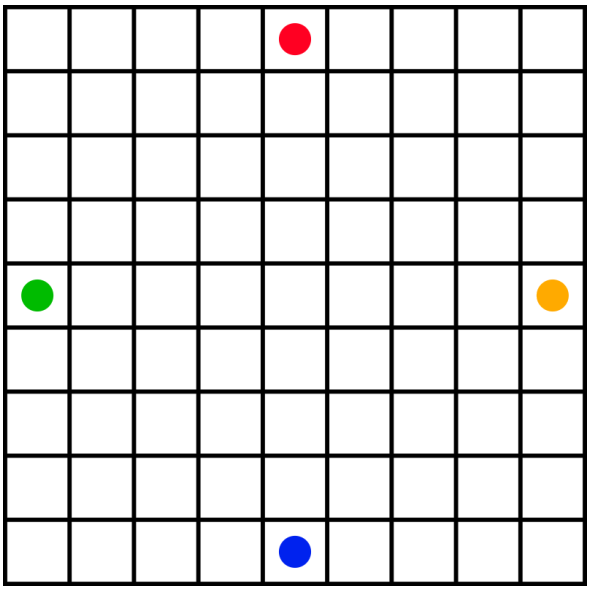
Dans le cadre du module d'intelligence artificielle, nous devions réaliser un projet en exploitant un nouveau langage de programmation, Prolog. L'approche de ce langage est différente des langages orientés objets tels que C#. A l'aide de règles logiques et d'instructions, nous devons concevoir une version informatisée du jeu **Quoridor**. Dans le contexte du projet, seulement deux joueurs seront impliqués dans la partie. Ce jeu de plateau joué sur un plateau de 81 cases, a pour but de mettre en place des stratégies et d'empêcher le joueur adverse d'atteindre le côté du plateau face à lui. Des contraintes sont à prendre en compte tel que la disposition de mur pour chaque joueur afin de restreindre le chemin voulu.

Afin d'élaborer le Quoridor, il est nécessaire de traduire les règles de ce jeu en un système logique par l'intermédiaire de prédicats. Cependant d'autres aspects sont à prendre en compte tels que la dynamique du jeu, l'affichage, et la partie intelligence artificielle. Dans la suite de ce rapport, nous allons détailler les axes principaux pour la compréhension du code avec les choix techniques et fonctionnalités présentes.

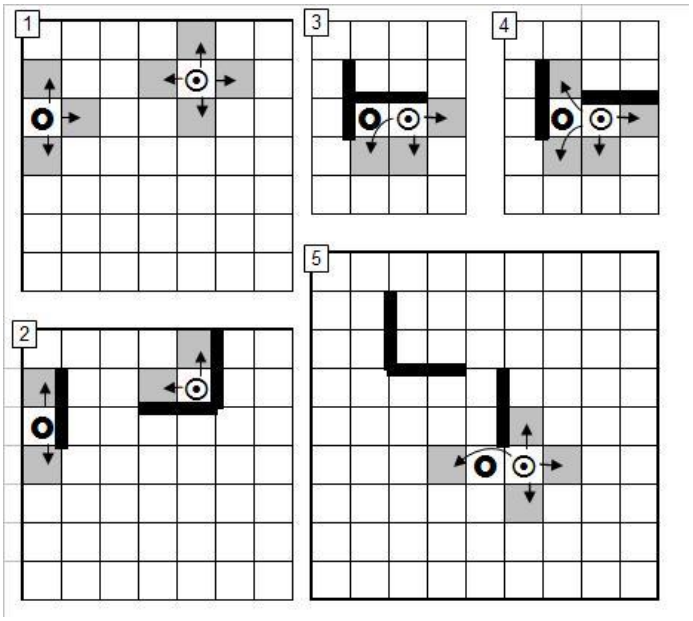
Mode d'emploi

1. Description du jeu

Quoridor est joué sur un plateau de 81 tuiles carrées (9x9). Chaque joueur y est représenté par un pion qui part de la tuile centrale du côté du plateau situé face à lui. L'objectif est d'être le premier joueur à atteindre n'importe laquelle des neuf tuiles situées sur le bord opposé du plateau.



Le jeu Quoridor comporte aussi vingt murs. Ce sont des plaquettes de bois longues de deux tuiles qui peuvent être placées dans l'interstice entre les tuiles. Les murs restreignent le déplacement de tous les pions, qui doivent les contourner. Les murs sont distribués également entre les joueurs au début de la partie ; une fois placés sur le plateau, ils ne peuvent plus être retirés ni bougés de la partie. Quand vient son tour de jouer, un joueur peut choisir soit de déplacer son pion, soit (si possible) de placer un mur.



Ci-dessus une image représentant les déplacements possibles dans ce jeu de plateau. Notre code illustre et gère ces divers déplacements des pions.

2. Exemple d'utilisation

```
_____________joueurA_____________

Mur(s) du joueur A : 5
Mur(s) du joueur B : 5

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

3 +   +   +   +   +   +   +   +   +   +

4 +   +   +   +   +   +   +   +   +   +

5 +   +   +   +   +   +   +   +   +   +

6 +   +   +   +   +   +   +   +   +   +

7 +   +   +   +   +   +   +   +   +   +

8 +   +   +   +   +   +   +   +   +   +

9 +   +   +   +   +   +   +   +   +   +
                B
  +   +   +   +   +   +   +   +   +   +

Au tour de joueurA de jouer :

Que voulez vous faire ? Entrez mur ou deplacer
|: deplacer.

Vos coordonnees actuelles: (5;1)
Ou voulez vous placer vous deplacer ?
 Entrez la coordonnee X (colonne) :
|: 5.

 Entrez la coordonnee Y (ligne):
|: 2.

Deplacement valide !!
```

Initialisation du jeu :

Le joueur doit taper ***start*** pour entamer une partie. En suivant, le programme lui propose deux actions : se déplacer ou placer un mur. Sur ce premier écran, le joueur A décide de se déplacer, il tape ***deplacer.***, il doit maintenant choisir les coordonnées de son déplacement. Celui-ci sera validé ou non par le programme.

C'est ensuite au tour du joueur B de faire son action.

```
_____________joueurB_____________

Mur(s) du joueur A : 5
Mur(s) du joueur B : 5

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

2 +   +   +   +   +   +   +   +   +   +
                A
3 +   +   +   +   +   +   +   +   +   +
```

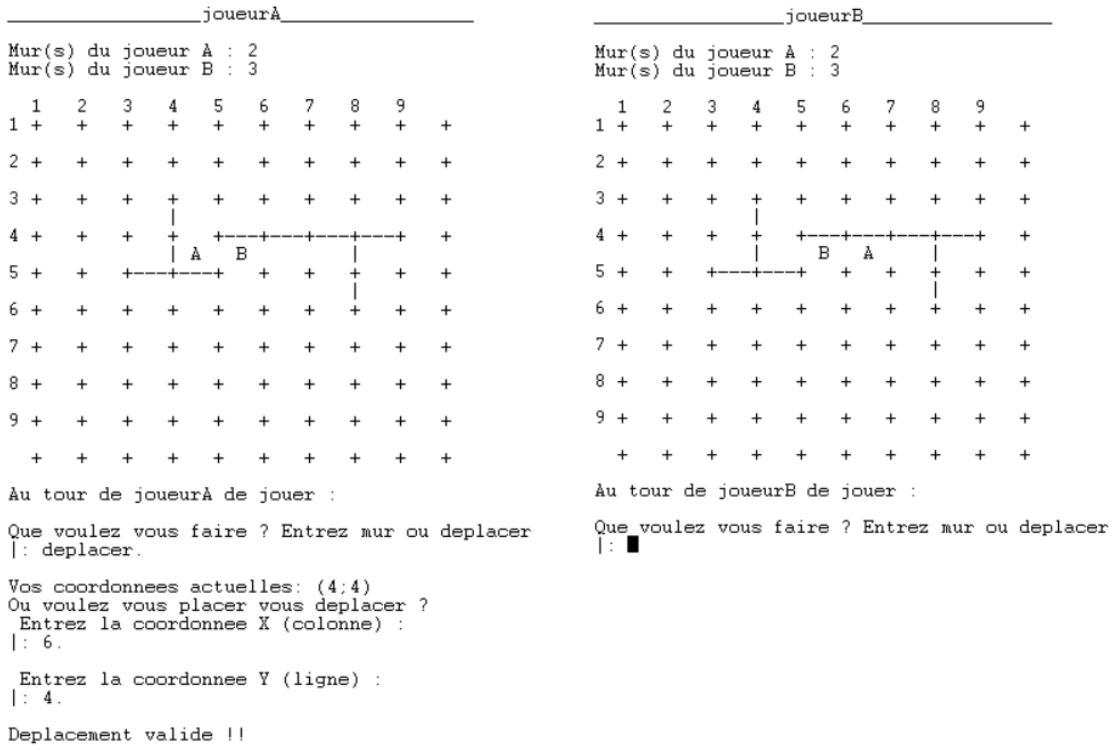
Gestion des murs :

Le joueur A reprend la main et décide de placer un mur pour bloquer son adversaire, il tape ***mur.*** Le joueur doit donc entrer les coordonnées du point de départ du mur. Enfin, il choisit l'orientation de son mur. Celui-ci sera validé ou non par le programme.

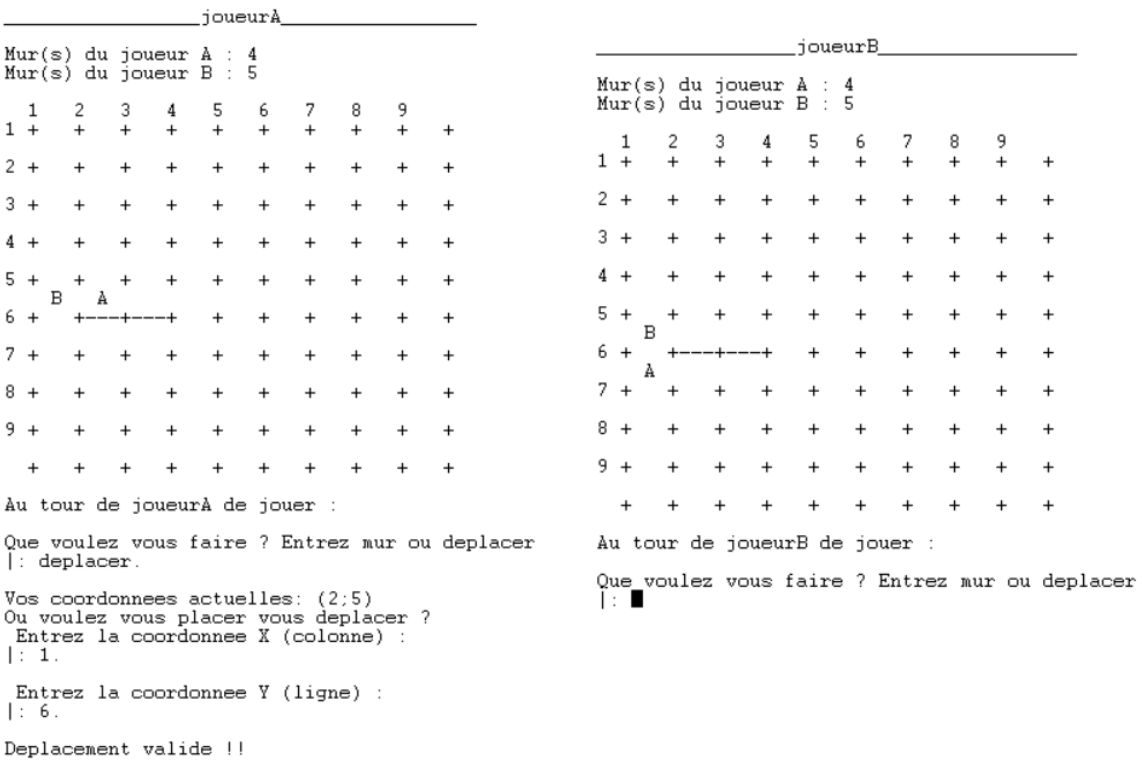
```
__________________joueurA_________________

Mur(s) du joueur A : 5
Mur(s) du joueur B : 5

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

2 +   +   +   +   +   +   +   +   +   +

3 +   +   +   +   +   +   +   +   +   +
              A
4 +   +   +   +   +   +   +   +   +   +

5 +   +   +   +   +   +   +   +   +   +

6 +   +   +   +   +   +   +   +   +   +

7 +   +   +   +   +   +   +   +   +   +
                      B
8 +   +   +   +   +   +   +   +   +   +

9 +   +   +   +   +   +   +   +   +   +

  +   +   +   +   +   +   +   +   +   +

Au tour de joueurA de jouer :

Que voulez vous faire ? Entrez mur ou deplacer
|: mur.

Ou voulez vous placer votre mur ?
 Entrez la coordonnee X (colonne) :
|: 6.

 Entrez la coordonnee Y (ligne) :
|: 7.

 Entrez l'orientation de votre mur : vertical ou horizontal :
|: horizontal.

Le mur a bien ete place !
```

```
__________________joueurB_________________

Mur(s) du joueur A : 4
Mur(s) du joueur B : 5

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

2 +   +   +   +   +   +   +   +   +   +

3 +   +   +   +   +   +   +   +   +   +
              A
4 +   +   +   +   +   +   +   +   +   +

5 +   +   +   +   +   +   +   +   +   +

6 +   +   +   +   +   +   +   +   +   +

7 +   +   +   +   +   +---+---+   +   +
                      B
8 +   +   +   +   +   +   +   +   +   +

9 +   +   +   +   +   +   +   +   +   +

  +   +   +   +   +   +   +   +   +   +

Au tour de joueurB de jouer :

Que voulez vous faire ? Entrez mur ou deplacer
|: 
```

Après plusieurs tours, les joueurs A et B sont à côté si le joueur A veut se déplacer sur la case d'après il peut sauter le joueur B.



C'est au tour du joueur A de jouer, pour avancer vers le côté gagnant le joueur A peut se déplacer en diagonale car B et le mur bloquent le chemin.



Après plusieurs tours, le joueur B se trouve sur l'avant dernière case de la grille, il se déplace donc sur la case en face de lui pour atteindre la dernière ligne. Le joueur B a gagné la partie !

```
_________________joueurB_________________

Mur(s) du joueur A : 5
Mur(s) du joueur B : 5

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

2 +   +   +   +   +   +   +   +   +   +
                    B
3 +   +   +   +   +   +   +   +   +   +
                    A
4 +   +   +   +   +   +   +   +   +   +

5 +   +   +   +   +   +   +   +   +   +

6 +   +   +   +   +   +   +   +   +   +

7 +   +   +   +   +   +   +   +   +   +

8 +   +   +   +   +   +   +   +   +   +

9 +   +   +   +   +   +   +   +   +   +

  +   +   +   +   +   +   +   +   +   +

Au tour de joueurB de jouer :

Que voulez vous faire ? Entrez mur ou deplacer
|: deplacer.

Vos coordonnees actuelles: (5;2)
Ou voulez vous placer vous deplacer ?
 Entrez la coordonnee X (colonne) :
|: 5.

 Entrez la coordonnee Y (ligne):
|: 1.

Deplacement valide !!

Fin du jeu, le joueurB a gagnÃ© !
true.
```

Choix techniques

Nous avons découpé notre programme en plusieurs fichiers:

Fichier	Fonction
main.pl	fichier principal contenant *start*
murs.pl	fichier contenant le code de gestion des murs.
deplacement.pl	fichier contenant le code de gestion des déplacements.
affichage.pl	fichier contenant le code permettant l'affichage
tools.pl	fichiers contenant du code utilitaire

Nous avons principalement utilisé la structure *(condition) -> si vrai ; si faux* pour factoriser le code.

Le plateau:

L'état du plateau est unifié pour la première fois dans start à un tableau contenant les coordonnées des joueurs et le nombre de mûrs qu'ils leur restent.

```
NombreMurs is 3,
Plateau = [
    [5, 1, NombreMurs],
    [5, 9, NombreMurs]],
```

Les murs:

Les murs sont décomposés en deux demi-mûrs. Les demi-mûrs sont définis par des prédicats ajoutés dynamiquement au cours du jeu.

```
assert(estPlaceMur(4,4,vertical)),
```

Exemple pour placer un demi-mur vertical de coordonnées (4;5)

Initialement nous avions choisis de placer les murs dans la liste du Plateau mais pour une exécution plus rapide le prédicat dynamique nous a paru plus pertinent.

Nous utilisons le prédicat: *estPlacableDoubleMur/4* pour vérifier si les coordonnées du murs rentrées par l'utilisateur sont correctes.

Enfin, nous avons fait en sorte que le joueur ne puisse pas placer un mur si celui-ci bloque le jeu. Grâce aux prédicats du fichier *chemin.pl* , nous vérifions qu'il existe toujours un chemin entre deux cases. Nous utilisons donc ce prédicat pour vérifier qu'il existe un chemin entre la position des joueurs et leurs cases d'arrivée avant de pouvoir placer un mur.

L'affichage:

L'affichage du terrain complet se fait à partir du prédicat *dessinerTerrain/1.* auquel on passe en paramètre le plateau actuel (coordonnées des joueurs et leurs nombres de murs restant entre crochets).

L'affichage se fait en ligne de commande à l'aide de la fonction prolog *write/1*. Un affichage préliminaire s'effectue pour afficher les informations liées aux joueurs (le nombre de murs restant), puis le numéro des cases. Enfin, la boucle du dessin du terrain se lance.

En lançant le prédicat *dessinerLigne/3,* on commence à dessiner la ligne J (entre 1 et 9) qui va se rappeler lui-même pour dessiner la ligne suivante, jusqu'à s'arrêter à la ligne 10 (fin).

- En faisant ça on appelle le prédicat *dessinerPionsMurs /4* qui dessine le pions/case vide de coordonnée I,J puis se rappelle elle-même pour dessiner la ligne jusqu'à son extrémité (9 cases)
- On appelle de la même manière le prédicat *dessinerMursHorizontaux/2* qui va de manière similaire dessiner une ligne de mur/espace

Le déplacement:

Nous utilisons le prédicat: *estDeplacementPossible/4* pour vérifier si les coordonnées rentrées par l'utilisateur sont correctes.

Fonctionnalités développées

N°	Fonctionnalités
F_1	Affichage du plateau dans la console
F_2	Possibilité de placer un mur (sans bloquer totalement le passage.)
F_3	Les murs ne peuvent chevaucher un autre
F_4	Possibilité de sauter par dessus un joueur

F_5	Possibilité de jouer à deux
F_6	Possibilité de se déplacer sur le plateau
F_7	Un mur ne peut pas être placé s'il encercle un joueur
F_8	Le jeu s'arrête quand un des joueurs à atteint le coté opposé à son coté de départ

Gestion de projet

Au début du projet nous avons défini plusieurs étapes. Pour avancer par étapes nous nous sommes fixés des missions, ces missions étaient définies lors de réunions hebdomadaires sur un salon Discord. Evidemment ces missions varient en fonction de l'avancement de la mission précédente et des problèmes rencontrés. Ces réunions nous permettaient aussi de faire un point sur notre avancement, d'adapter le planning, de discuter de nos problématiques. Certaines tâches ont été plus conséquentes et ont demandé plus de temps.

La gestion des versions du code s'est faite avec Git, et nos avancements sur le code source ont été partagés via la plateforme en ligne GitHub. Le travail en parallèle est une étape délicate à gérer, ainsi GitHub nous a permis de réaliser l'intégralité du code tout en travaillant en simultané, en récupérant le travail et en comparant le code généré.

Bilan

Ce projet nous a permis de réutiliser les connaissances acquises en TD sur le langage Prolog. Nous avons pu réinvestir différents aspects comme la gestion de liste, de prédicats etc..

Nous avons rencontré certaines difficultés dues au travail en groupe à distance et à la charge de travail. En effet, il était compliqué de se regrouper tous ensemble pour travailler en dehors des créneaux prévus, il était donc difficile de connecter les différentes parties.

Nous sommes tout de même satisfait du résultat car nous avons pu réaliser les fonctionnalités principales du jeu.