MASTER M1 INFORMATIQUE

REPRÉSENTATION DES CONNAISSANCES *(IA SYMBOLIQUE)*

TP 1 – PARTIE 2

RÉALISATION D'UN PROGRAMME D'OTHELLO ET D'UN JOUEUR ARTIFICIEL POUR LE JEU D'OTHELLO.

Le programme devra être déposé sur la plateforme UMTICE à la fin de chaque TP.

PRÉSENTATION GÉNÉRALE

L'objectif du projet de Tps (TP 1, 2 et 3) sera de réaliser un programme d'Othello en Prolog. Votre programme final devra jouer selon deux modes possibles:

- un mode *superviseur*: Le programme devra permettre à deux joueurs humains de jouer l'un contre l'autre. Le programme vérifiera que les coups des 2 joueurs sont corrects et affichera l'othellier tout au long de la partie en mettant à jour les coups.
- un mode *moteur*: La partie doit permettre à un joueur humain de jouer contre la machine, le programme jouant lui-même et répondant aux coups de l'humain en utilisant un algorithme de recherche de coup plus ou moins élaboré (simple, minimax, alpha-bêta,...). La couleur du joueur humain devra pouvoir être choisie au départ.

Par convention, la position de départ sera donnée dans les règles du jeu. Pour l'affichage, le noir sera représenté par 'x' et le blanc par 'o'.

MODULES A IMPLEMENTER

Le programme final comportera au minimum les différents fichiers (modules) prolog suivants:

- othello.pl: **module principal**, qui chargera tous les modules nécessaires au déroulement du jeu et dans lequel est implémenté le prédicat principal **lanceJeu/0** qui lance une partie, permet de choisir le mode, la couleur, ...
- representation.pl: **module de représentation** du jeu, avec la définition des prédicats de manipulation de la grille de jeu (son affichage, la saisie des coups,...).
- regles.pl: **module de modélisation des règles du jeu**, dans lequel on trouvera toutes les fonctions dépendantes des règles du jeu d'Othello (coup valide, prise,...).
- evaluation.pl: **module d'évaluation d'une position** qui implémente tous vos prédicats nécessaires pour cette évaluation (cases libres, nombres de pions dans chaque couleur,...).
- minimax.pl: module de recherche du coup à jouer par l'ordinateur par l'**algorithme minimax**.

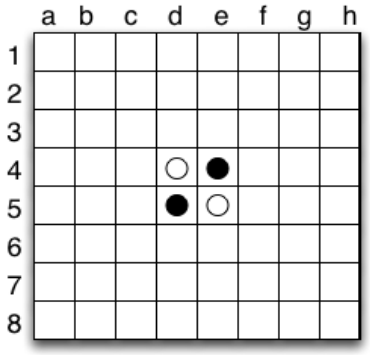
D'autres modules *pourront* être implémentés :

- alpha.pl: module de recherche du coup à jouer par l'ordinateur, par l'**algorithme alpha-bêta**, d'une optimisation de l'alpha-bêta, voire d'un autre algorithme de recherche.
- optimise.pl: module des versions optimisées: optimisation de l'alpha-bêta, voire d'un autre algorithme de recherche. Implémentation de la possibilité que le programme joue contre lui-même. Possibilité de faire jouer vos programmes l'un contre l'autre.

Il n'y aura pas de rapport écrit supplémentaire à rendre mais vous devrez commenter votre code et donnez en commentaires des exemples de requêtes illustrant vos prédicats. Sauvegardez votre travail régulièrement...

RÈGLES DU JEU D'OTHELLO

Le jeu Othello se joue sur une othellier (un damier) de 8 cases sur 8 avec des pions de deux couleurs (Noir er Blanc). Au départ, l'othellier est dans la position suivante :



Chaque case est repérée par ses coordonnées : colonne / ligne.

Les pions du jeu comportent 2 faces : une blanche et une noire. La face visible donne la couleur actuelle du pion. Le joueur qui joue les pions noirs commence à poser son premier pion. Les règles du jeu sont les suivantes :

- Un joueur peut jouer si, en posant son pion, il prend en sandwich sur une ligne, une colonne ou une diagonale, un ou plusieurs pions de l'adversaire.
- Si le joueur peut jouer, tous les pions pris en sandwich sur les lignes, colonnes ou diagonales, sont retournés et deviennent des pions du joueur.
- Si le joueur ne peut pas jouer, il passe son tour et son adversaire peut jouer.
- La partie est finie s'il n'y a plus de cases jouables pour les 2 joueurs (si les deux joueurs doivent passer leur tour ou l'othellier n'a plus de case libre).
- Le gagnant est le joueur qui a le plus de pions de sa couleur sur l'othellier à la fin de la partie.

PLANNING DES SÉANCES DE TP

- Etape 1 (TP1 et 2): Réalisation d'un othellier permettant de superviser une partie entre deux humains : définir la représentation du jeu, sa visualisation, la gestion des coups, la fin du jeu.
- Etape 2 (TP2 et 3): Fonction d'évaluation et algorithme minimax.

- Etape 3 (TP3 et +): Optimisation et alpha-beta pour ceux qui veulent aller plus loin...

MASTER M1 ISI

177 EN 005 - REPRÉSENTATION DES CONNAISSANCES *(IA SYMBOLIQUE)*

RÉALISATION D'UN OTHELLIER EN PROLOG – ETAPE 1

PRÉSENTATION

Dans cette première étape de la réalisation de votre programme d'Othello, vous allez réaliser un programme permettant de superviser une partie de jeu entre deux joueurs humains. Votre programme affichera la grille de jeu, saisira les coups, vérifiera qu'ils sont corrects et modifiera la grille de jeu en conséquence. A la fin de la partie, il affichera le score et le vainqueur.

Remarque : Dans questions qui suivent, la description des prédicats n'est pas forcément complète. Dans certains cas, il vous faudra écrire des prédicats intermédiaires afin de réaliser des prédicats demandés.

REPRÉSENTATION DU JEU

Nous allons représenter la grille de jeu à l'aide de listes Prolog.

Une ligne de la grille de jeu sera représentée par une liste à 8 éléments représentant chacun une case de la ligne. Une case peut être vide (représentée par '-') ou occupée par un pion noir ('x') ou par un pion blanc ('o').

La grille de jeu complète sera représentée par une liste de 8 listes, chacune de ces listes représentant une ligne de la grille, de la ligne supérieure à la ligne inférieure.

1. Ecrivez la clause fait **`grilleDeDepart(+Grille)`** qui est satisfait si la **`Grille`** est la grille correspondant à la position de départ du jeu.

Dans tout le texte qui suit, nous parlerons de *grille* pour faire référence à la liste contenant toute la grille de jeu et nous parlerons de *ligne* pour faire référence à une liste représentant une ligne quelconque.

Une ligne sera repérée par un nombre entier de 1 à 8, son numéro depuis le haut de la grille. Une colonne sera repérée par une lettre de *a* à *h,*de la gauche vers la droite. Ainsi la case en haut à gauche de la grille sera la case *a1* et celle en bas à droite la case *h8*.

GESTION DE LA GRILLE ET DES COUPS

Première étape: visualiser la grille de jeu.

2. Ecrivez un ensemble de clauses faits **`succNum(?Val1,?Val2)`** pour définir le successeur de chaque entier de *1* à *8*. De la même manière, écrivez un ensemble de clauses faits **`succAlpha(?Alpha1,?Alpha2)`** pour définir le successeur de chaque lettre de l'alphabet entre *a* et *h*. Ces prédicats nous permettront de passer d'une case à l'autre ou d'une ligne à

l'autre.

3. Ecrivez le prédicat **`afficheLigne/1`** qui écrit une ligne d'une grille. Ce prédicat prend en paramètre la liste à afficher. Vous pouvez utiliser le prédicat prédéfini **`tab/1`** qui permet d'écrire un certain nombre d'espaces. Par exemple, tab(3) écrit 3 caractères espace.
4. Ecrivez le prédicat **`afficheGrille/1`** qui écrit la grille de jeu à l'écran. Ce prédicat prendra en paramètre une grille de jeu (une liste de listes). On souhaiterait aussi avoir les indications relatives aux coordonnées de la grille (affichage en première ligne de la liste des coordonnées des colonnes, et au début de chaque ligne, le numéro de la ligne correspondante).
5. Ecrivez le prédicat récursif **`ligneDansGrille(+NumLigne, +Grille, ?Ligne)`** qui est satisfait si **`Ligne`** est la ligne numéro **`NumLigne`** dans la **`Grille`**. De la même façon, écrivez le prédicat récursif **`caseDansLigne(+Col, +Liste, ?Valeur)`** qui est satisfait si **`Valeur`** est dans la **`Liste`** en position **`Col`**.
6. Ecrivez le prédicat **`donneValeurDeCase(+Grille, +NumColonne, +NumLigne,?Valeur)`** qui est satisfait si il y a une valeur **`Valeur`** dans la case de coordonnées **`[NumColonne, NumLigne]`** de la **`Grille`**.
7. Ecrivez le prédicat **`caseVide(+Grille, +NumColonne, +NumLigne)`** qui est satisfait si la case de la grille correspondante est vide.

Deuxième étape: validité des coups.

Maintenant il faut pouvoir vérifier la légalité d'un coup. Pour cela, à partir de la case jouée, il va falloir examiner, dans chaque direction la possibilité de réaliser une prise. On rappelle qu'une prise par le camp x est possible, selon une direction donnée, si dans la direction donnée, on trouve au moins un pion o et ensuite un pion x.

8. Ecrivez l'ensemble des 8 clauses faits **`direction(?Dir,?DiffLig, ?DiffCol)`** qui est satisfait si, pour obtenir la case adjacente dans la direction **`Dir`** il faut ajouter **`DiffLig`** à l'indice de la ligne et **`DiffCol`** à l'indice de la colonne.

Il y a différentes façons d'utiliser les clauses **`direction`**, les prédicats suivants en décrivent une mais vous pouvez en déterminer une autre.

9. Ecrivez le prédicat **`donneListeCasesDansDirection(+Dir, +Grille, +NumLigne, +NumColonne,?ListeCases)`** qui est satisfait si **`ListeCases`** contient la liste des contenus des cases de la **`Grille`**, situées entre la case de coordonnées (**`NumLigne`**, **`NumColonne`**) et le bord de l'othellier, dans la direction **`Dir.`** Cette liste peut être vide.
10. Ecrivez le prédicat **`faitPrise(+Camp, +ListeCases)`** qui est satisfait si, étant donné une liste de cases (supposées être dans la même direction), **`Camp`** réalise une prise.
11. Ecrivez le prédicat **`leCoupEstValide(+Grille, +Camp,+NumLigne, +NumCol)`** qui s'efface si le fait de jouer dans la case (NumCol, NumLigne) de Grille est légal pour le joueur Camp.

Troisième étape: jouer effectivement un coup.

12. En vous aidant des questions précédentes, écrivez le prédicat **`retournePionsDansDirection (+Dir, +GrilleDep, +NumLig, +NumCol, +Valeur,?GrilleArr)`** qui est satisfait si **`GrilleArr`** est obtenue à partir de **`GrilleDep`** dans laquelle on a retourné les pions autres que **`Valeur`** depuis la case (**`NumCol, NumLig`**), et cela dans la direction **`Dir.`**
13. Ecrivez le prédicat **`coupJoueDansGrille(GrilleDep, NumLig, NumCol,`**

Valeur,?GrilleArr) qui est satisfait si **GrilleArr** est obtenue à partir de **GrilleDep** dans laquelle on a joué **Valeur** en **NumCol, NumLig,** et cela étant d'autre part un coup valide.

FIN DE PARTIE ET COMPTABILISATION DES POINTS

On va s'intéresser maintenant à savoir si une partie est terminée et qui l'a gagnée.

14. Ecrivez le prédicat **comptePions(+Grille,?NbPionsX,?NbPionsO)** qui est satisfait si dans **Grille** il y a **NbPionsX** pions x et **NbPionsO** pions o.
15. Afin de simplifier l'écriture, écrivez un prédicat **coordonneesOuListe/3** qui prend en argument un nom de colonne **NomCol**, un numéro de ligne **NumLig** et une liste composée des 2 éléments **NomCol** et **NumLig.** Par exemple la requête **coordonneesOuListe(a, 2, [a,2])** sera vérifiée. Ce prédicat nous permettra donc de passer à un mode de coordonnées séparées en X et Y en un mode de représentation où les 2 coordonnées sont sous forme de liste, et réciproquement. Une case sera donc maintenant une liste **[NomCol, NumLig]**.

Pour vérifier que la partie est terminée, il faut vérifier qu'il ne reste aucun coup valide pour chacun des deux camps. Pour cela, une solution est de construire la liste de toutes les cases libres de l'othellier. A partir de cette liste, on peut construire la liste de toutes les cases 'légales' où un joueur donné peut jouer.

16. Ecrivez les prédicats **listeCasesVides(+Grille,?ListeCasesVides)** et **listeCasesValides(+Grille,+Camp,+ListeCasesVides, +ListeCases).**
17. Ecrivez le prédicat listeDesCoupsCamp(+Grille, +Camp,?ListeCases) qui est satisfait si ListeCases est une liste de cases de Grille où Camp peut jouer.
18. Ecrivez le prédicat partieGagnee/2 qui prend en arguments une valeur (x, o, -) et une grille et qui se vérifie si la partie est gagnée par le joueur jouant avec la valeur donnée.

MOTEUR DU SUPERVISEUR

19. Ecrivez le prédicat **duneListeALautre/3** qui prend en argument une liste de cases **LC1**, une case **C** et une autre liste de case **LC2** et qui vérifie si la liste **LC1** est composée de toutes les cases de la liste **LC2** plus de la case **C**. Par exemple, **duneListeALautre([[a,1], [a,2],[a,3]], [a,2], [[a,1],[a,3]])** sera vérifiée.
20. Ecrivez un ensemble de clauses **campAdverse/2** qui permet de trouver le camp adverse d'un camp donné.
21. Ecrivez le prédicat **joueLeCoup(Case, Valeur, GrilleDep, GrilleArr)** qui se vérifie si la grille **GrilleArr** est obtenue en jouant la valeur **Valeur** dans **Case** dans la grille **GrilleDep.** Rajoutez un **AfficheGrille** dans ce prédicat afin d'afficher le coup joué.
22. Ecrivez le prédicat **saisieUnCoup(NumLig,NumCol)** qui permet de saisir par des **read** les coordonnées à jouer.
23. Ecrivez le prédicat **moteur(Grille, ListeCoups, Camp)** qui a pour paramètres une Grille, dans laquelle tous les coups de la listeCoups sont jouables, et pour laquelle Camp doit jouer (c'est son tour). On écrira différentes clauses pour ce prédicat en particulier il

faudra distinguer :

- le cas où l'un des deux joueurs a gagné
- le cas où la grille est remplie et qu'il n'y a pas de gagnant
- le cas où c'est à l'ordinateur de jouer
- le cas où c'est au joueur humain de jouer.

24. Ecrivez le prédicat **`lanceJeu/0`** qui permet de lancer la partie.