TD de *Programmation logique et par contraintes* n° 5

MinMax en Prolog: le morpion

"Morpion" est un jeu à deux jouers, appelés dans cet énoncé *max* et *min*, qui se joue sur un matrice de dimension 3. Au début, les 9 cases de la matrice sont libres. Chaque joueur, à son tour, choisit une case libre et la marque (disons que *max* joue des croix et *min* des cercles) ; la case en question devient alors occupée et ne peut plus être choisie. Un joueur gagne s'il arrive a marquer les trois cases d'une ligne, d'une colonne ou d'une des deux diagonales de la matrice.

Le but de cet exercice est d'implementer ce jeu, en utilisant l'algorithme MinMax.

Les positions du jeu sont réprésentées par des listes de longueur 9, avec l'énumération des cases suivante:

1	2	3
4	5	6
7	8	9

Par exemple, la liste
`[l, l, ○, × ,×, ○, l, l, l]` (`l` désignant une case libre)
représente la position:

		○
×	×	○

Exercice 1 Définir un prédicat `move(+pos_courante,+joueur,-pos_suivante)` qui réussit si la position `pos_suivante` est atteignable à partir de la position `pos_courante` quand `joueur` a la main.

Exercice 2 Définir un prédicat `finale(+pos,-joueur)` qui réussit si `pos` est une position finale du jeu, et dans ce cas `joueur` vaut `max`, `min` ou `nul` selon les cas (à savoir: le joueur *max* a gagné - le joueur *min* a gagné - match nul).

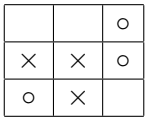
Exercice 3 Définir un prédicat `h(+pos,-valeur)` qui évalue une position, du point de vue de *max*..

Voici une des spécifications possibles de ce prédicat (mais vous pouvez définir votre propre fonction heuristique):

la valeur d'une position finale est:

- 100 si le gagnant est `max`.
- -100 si le gagnant est `min`.
- 0 pour un match nul.

La valeur d'une position p non finale est le nombre de possibilités de gain de *max* (c'est à dire le nombre de lignes, colonnes et diagonales de p qui ne contiennent pas le symble o) moins le nombre de possibilités de gain de *min*. Par exemple, la valeur de la position



est 0 (restent possibles pour *max* la colonne centrale et la diagonale principale, et pour *min* la première ligne et la troisième colonne).

Exercice 4 Définir un prédicat `minimax(+joueur,+pos_courante,-valeur,-pos_suivante,+profondeur)` qui choisit le prochain coup de `jouer` utilisant l'algorithme minmax .

Rappel:
`minimax(+joeur,+pos_courante,-valeur,-pos_suivante,+profondeur)`
est tel que:

- si `pos_courante` est une position finale, ou bien si `profondeur` est 0, alors `valeur` est calculée en utilisant le prédicat `h` de l'exercice précédent (et `pos_suivante` n'est pas calculée).
- sinon, on calcule l'ensemble P des positions atteignables par un coup de `joueur` à partir de `pos_courante`, on applique récursivement à chacune de ces position le prédicat `minimax`, en décrementant `profondeur` et en changeant de joueur, puis, on affecte à `pos_suivante` l'élément de P dont la valeur est maximale, si `joueur` est *max* (*resp.* minimale, si `joueur` est *min*). On affecte la valeur de cette position à `valeur`.

Exercice 5 Définir un prédicat `jeu(+Niveau,+Joueur)` qui permet à l'utilisateur de jouer au morpion contre la machine. La machine joue en utilisant le prédicat `minimax` à profondeur `Niveau`. Si `Joueur` vaut `min`, l'utilisateur commence, si `Joueur` vaut `max`, la machine commence. Vérifier quel sont les niveaux de jeu auxquels il est possible de gagner contre la machine.