|  |  |
| --- | --- |
| |  | | --- | |  | |

**Devoir #2 : Bataille Navale**

Session d'hiver 2015

*Système multi-agent*

*(8INF844)*

*Département d’informatique et de mathématiques*

*Présenté à : Professeur: Sébastien Guillet*

*Travail de : AMAMOU Houssem*

*Code Permanent : AMAH10029004*

Date de remise : 27-02-2015

Nous allons présenter l’application que nous avons implémentée en mettant l'accent sur les points les plus importants, par la suite nous allons présenter un exemple qui va permettre d'expliquer les principes de l'implémentation de notre moteur d'inférence ainsi que les règles.

## Les faits :

On a choisi de représenter les faits sous forme d’un objet composé d’une chaîne de caractère et de la valeur booléenne de ce fait.

## Le moteur d’inférence:

Le moteur d’inférence utilise un chaînage avant qui parcourt l’arbre des règles en profondeur, c’est à dire que des que la base des fait contient un fait qui fait partie de la solution de notre problème le moteur d’inférence va s’arrêter pour donner le fait solution. Afin de générer la solution de ce problème le moteur d’inférence va parcourir les règles qui se retrouve dans le fichier Battleship.rules l’une après l’<autre et vérifie si l’une d’entre elles est déclenchable autrement dit si les prémisses d'une règle satisfait les faits contenu dans la base de fait du moteur d’inférence. Lorsque les prémisses d’une règle est vérifié, on ajoute les conclusions de cette règle a notre base de fait, soit en ajoutant tout simplement de nouveau fait, soit on modifiant la valeur d’un fait déjà présent. L’objet MoteurInference contient deux variables principales qui sont: baseFaits et règles. La première variable va contenir tous les faits de notre moteur, cette variable est sous la forme d’une HashMap contenant le fait et sa valeur qui est vrai ou faux. La base des faits est mises à jour à chaque itération du moteur d’inférence. baseFait contient initialement des faits permettant l’exécution de notre moteur pour la première itération. En ce qui concerne la deuxième variable règles, cette dernière va représenter les règles présentes dans le fichier Battleship.rules, ce fichier va être lu dans le constructeur de notre moteur afin de charger les règles dans ce dernier. Les règles doivent être composées de permisse et de conclusion séparé par «=>». Les faits présents dans les prémisses et les conclusions sont séparé par«,». Si l’on retrouve un «¬» avant un fait cela voudra dire que l’on doit évaluer le complémentaire de ce fait. Lors de l’exécution du moteur d’inférence, on a besoin de marquer les règles qui ont été précédemment utilisé, c’est pour cela que l’on retrouve la variable check dans notre objet règle. Nous avons implémenté un système permettant de retourner en arriéré lorsqu’un certain scénario se produit car il est nécessaire d’avoir ce type de mécanisme lorsqu’une décision retourne un résultat négatif. Nous allons détailler cela par la suite.

## Les règles

Afin de mieux comprendre comment on a choisi d’implémenter nos règles et notre moteur d’inférence, on va présenter les règles utilisé afin de les expliquer. Pour cela nous devons expliquer ce que représente chaque fait utilisé pour construire nos règles.

* dernierCoup est un fait permettant de voir si le dernier coup qui a été tiré a touché une cible dans ce cas sa valeur sera vrai.
* touche est un fait permettant de savoir si on a touché un bateau lors des coups précédant, il se met à vrai lorsqu’on heurte un bateau pour la première fois et ne change de valeur que lorsque le bateau qui été heurté est coulé
* random est un fait qui permet d’indiquer au moteur d’inférence qu’il peut tirer un coup au hasard.
* purge est un fait utilisé permettant de mettre à jour la base des faits dans certaine situation par exemple lorsqu’un bateau a coulé.
* coule est un fait permettant de savoir si un bateau qui a été touché a coulé.
* bouger Haut, bougerBas, bougerGauche, bougerDroite sont des faits permettant d’indiquer si un mouvement vers la direction indiquée est possible. Le mouvement n’est pas possible lorsqu’on se retrouve près d’un bord du plateau, ou lorsqu’on a déjà tiré sur l’endroit sur lequel on s’apprête à tirer au prochain coup.
* haut, bas, gauche, droite sont des faits qui servent à indiquer au moteur qu’il a ou pas la possibilité de tirer vers ce sens. un exemple qui peut expliquer cela est lorsqu’un bateau a été touché et qu’on a choisi de tirer un coup vers le haut mais ce tire est tombé dans l’eau, on a plus la possibilité de tirer vers le haut. C’est un système de sécurité permettant d’éviter que ce genre de scénario arrive.
* tirerHaut, tirerDroite, tirerGauche, tirerBas sont des faits permettant d’indiquer la direction vers lequel va se faire le prochain tir.
* revenir est un fait permettant d’indiquer au moteur d’inférence qu’il doit revenir en arriéré car on se trouve dans la situation ou notre dernier coup est tombé dans l’eau alors que notre avant dernier coup a touché donc nous devons localiser la position du bateau afin de le couler le plus rapidement possible pour cela certain fait doivent être rechargé dans la base des faits et des règles doivent être soit marquée soit démarquée.

Le principe de nos règles est que tant qu’on n’a pas coulé le bateau on doit frapper aux alentours de ce dernier et ceci en changeant de direction de tir si cela s’avère nécessaire. Le changement de direction s’effectue lors qu’on rate le tir et pour être sûr de couler le bateau en un minimum de coup on doit orienter nos tirs dans les 4 directions différentes si cela est possible et en changeant de direction lorsque certaines conditions se réunissent

Les faits qui font office de solutions dans notre problème sont: tirerHaut, tirerBas, tirerGauche, tirerDroite et random. Cela veut dire que si aucun de ces fait n’est présent dans la base des faits et a une valeur positif le moteur d’inférence doit encore faire une itération.

Afin de mieux comprendre comment marche notre moteur, nous allons procéder à une exécution a la main et voir quelles sont les règles utilisé

## Exemple d’exécution

Pour commencer on tir un coup au hasard en utilisant la règle si dessous, dernierCoup est à faux et on a encore touché aucun navire donc touche est à faux. Comme résultant de l’exécution de cette règle on ajoute random avec la valeur vrai, purge avec vrai et coule avec la valeur faux. on ajoute purge et coule pour prévenir de l’exécution d’autre règles qui tromperait notre moteur. La décision qui a été donc prise est de tirer un coup au hasard

1. Règle utilisée: ¬dernierCoup,¬touche=>random,purge,¬coule

Décision: random

Le coup qu’on a tiré précédemment a touché un bateau donc la fait dernierCoup est devenu vrai et comme coule est à faux et on a la possibilité de bougerHaut et haut donc tirerHaut est a vrai

1. Règle utilisée: dernierCoup,¬coule,bougerHaut,haut=>tirerHaut,¬random,touche,¬purge

Décision: tirerHaut

On tire encore un coup en haut car cela est permis

1. Règle utilisée:dernierCoup,¬coule,bougerHaut,haut=>tirerHaut,¬random,touche,¬purge

Décision: tirerHaut

Le bateau a coulé avec les deux tirs précédents donc le prochain coup va être tiré au hasard.

1. Règle utilisée: coule=>random,¬dernierCoup,haut,bas,gauche,droite,¬touche,purge

Décision: random

On touche un autre bateau avec notre coup.

1. Règle utilisée:¬dernierCoup,¬touche=>random,purge,¬coule

Décision:random

La case qui a été touché se trouve dans le rangée du haut donc on ne peut pas tirer en haut car bougerHaut est à faux donc la règle activée est la suivante

1. Règle utilisée: dernierCoup,¬coule,bougerBas,bas=>tirerBas,¬random,touche,¬purge,¬haut

Décision: tirerBas

le coup tiré en bas a fini dans l’eau donc on doit procédé à un rollback permettant d’activer le fait tirerGauche, on utilise ici trois règle afin d’arriver à une conclusion grâce au fait revenir

1. Règle utilisée:touche,¬coule,tirerBas=>revenir,¬tirerBas,¬bas,¬random,¬haut

Règle utilisée: revenir=>dernierCoup,¬revenir

Règle utilisée:dernierCoup,¬coule,bougerGauche,gauche=>tirerGauche,¬random

,touche,¬purge,¬haut,¬bas

Décision: tirerGauche

Notre tir à gauche est encore tombé à l’eau donc on doit changer de direction comme précédemment

1. Règle utilisée: touche,¬coule,tirerGauche=>revenir,¬tirerGauche,¬gauche,¬random,

¬haut,¬bas

Règle utilisée: revenir=>dernierCoup,¬revenir

Règle utilisée: dernierCoup,¬coule,bougerDroite,droite=>tirerDroite,¬random,

touche,¬purge,¬haut,¬bas,¬gauche

Décision : tirerDroite

Le bateau a coulé dont on peut tirer un coup au hasard grâce à la règle suivante.

9. Règle utilisée: coule=>random,¬dernierCoup,haut,bas,gauche,droite,

¬touche,purge

Décision : random