Université de Technologie de Compiègne

IA02

Rapport de TP

Prolog Stock Exchange

Printemps 2015

Damien MARIÉ - Kyâne PICHOU

16 juin 2015

Table des matières

1 Introduction

Dans le cadre des TP de l'UV IA02, nous devons implémenter une version Prolog du jeu Chicago Stock Exchange, un jeu de plateau sur le thème de la bourse et de la finance. A chaque tour, les joueurs déplacent un pion « trader » et récoltent 2 marchandises. Ils en gardent une et jettent l'autre, faisant évoluer la valeur des ces marchandises. Le but étant d'avoir le stock de marchandises ayant le plus de valeur à la fin du jeu.

1.1 Structures de données

Les différentes structures de données qui composent ce jeu seront implémentées sous forme de listes.

Bourse La bourse est simplement une liste contenant 6 sous-listes (une par élément) formant un couple [marchandise,valeur].

Marchandises Le jeu est composé de 9 piles de 4 marchandises (à l'état initial). Chaque pile forme une sous-liste de 4 éléments et ces sous-listes composent une grande liste représentant l'état de toutes les piles du jeu.

Trader Le trader a une position qui est simplement représentée par le numéro de la pile où il se trouve.

Réserve Le joueur accumule des ressources tout au long de la partie. Chaque jouer a donc une liste dédié contenant toutes les marchandies qu'il possède.

Coup de jeu Lorsqu'un joueur joue, plusieurs paramètres sont à prendre en compte. Ces paramètres sont placés dans une liste contenant :

- le joueur qui effectue l'action
- la valeur du déplacement effectué (1,2 ou 3)
- la ressource qui est conservée
- la ressource qui est jetée

2 Principaux prédicats

2.1 Gestion du plateau

2.1.1 Affichage

2.1.2 Génération du plateau

En début de partie il faut générer les différentes structures qui composent un plateau. On crée donc un prédicat retournant un nouveau plateau.
Le prédicat va générer l'ensemble des piles en formant une liste, contenant des sous-listes de 4 éléments. Pour générer ces sous-listes, on tire un nombre au hasard, correspondant à l'indice d'un élément dans une grande liste contenant tout les jetons du jeu.

```
take_random(All,El,NewAll):-
       random_el(All,N),
       nth(N,All,El),
       select(El,All,NewAll)
```

Ensuite on génère la bourse, qui est toujours la même à l'état initial. On récupère donc simplement une liste hardcodée correspondant à l'état initial de la bourse.
De plus on génère un chiffre entre 1 et 9 correspondant à la position du trader.

2.2 Déroulement du jeu

2.2.1 Vérification de coup

Il est important de vérifier qu'un coup respecte les règles du jeu lorsqu'il est executé. Pour cela on utilise un prédicat *coup_possible* qui va prendre en entrée un plateau de jeu et qui va vérifier que le coup proposé est valide.
On commence par tester que la valeur du déplacement est bien de 1, 2 ou 3. Puis on effectue réellement le déplacement afin de récupérer la nouvelle position du trader. A partir de la nouvelle position du trader, on en déduit les éléments se trouvant sur le dessus des piles adjacentes. Enfin, on vérifie que les éléments à jeter et garder correspondent bien à ceux présents sur le haut des piles.
Concernant le déplacement du trader, on additionne simplement sa position avec le valeur du déplacement. On prend cependant garde à ne pas dépasser le nombre de pile, si c'est la cas alors on effectue un bouclage pour revenir au début du plateau.
Pour la vérification des éléments, on appelle un prédicat *choix* qui va retourner les 2 éléments disponibles en fonction de la position du trader. Ce prédicat récupère le nombre de pile puis la position des piles autour du trader. Puis il récupère simplement les éléments en haut des piles corespondantes.

2.3 Intelligence artificielle

Nous avons programmé plusieurs intelligences artificielles afin de pouvoir avoir un benchmark de celles-ci :

— Une intelligence purement aléatoire, qui choisit un coup aléatoire parmis les coups possibles (ai_random)

— Une intelligence qui regarde à un seul niveau les scores possibles (ai_simple_best)
— Une intelligence qui utilise l'algorithme minimax (ai_minimax)

Pour commencer on déplace le trader, puis on ajoute l'élément qui est conservé par le joueur dans sa réserve. Pour cela on appelle les prédicats *bouger_trader* (déjà utilisé dans le test de coup) et *add_to_player* qui ajoute tout simplement l'objet *Keep* dans la réserve du joueur *J*.
Le prédicat *remove_items* permet de supprimer les éléments sur le dessus des piles adjacentes au trader.
Pour terminer, il faut dévaluer l'élément qui est jeté. Pour cela on parcourt l'ensemble des élément de la bourse en appellant récursivement un prédicat *downgrade*. Lorsque l'élément est celui jeté, on décrémente sa valeur et on réinsert le couple dans la liste.
Dans notre traitement du coup, il faut prendre en cas la situation où une pile devient vide. Dans ce cas il faut la supprimer, ce qui implique que la position du trader devient fausse (puisqu'il ,n'y a une ou deux pile en moins). De ce fait on effectue un test sur la position du trader après avoir joué, afin de s'assurer que sa position ne dépasse pas le nombre de pile.

3 Conclusion

Nous avons bien réussi à implémenter une version Prolog du jeu Chicago Exchange. Le jeu est complet, c'est à dire qu'il se déroule correctement et répond au cahier des charges :

— Partie Humain vs Humain

— Partie IA vs Humain

— Partie IA vs IA

Entre autre la gestion des marchandises, de la bourse, des réserves et de la position du trader sont implémentés, fonctionnels et sans failles. Notre intelligence artificielle reste cependant rudimentaire, c'est à dire qu'elle n'implémente pas l'algorithme *minimax* et ne fait ses calculs que en fonction de l'état courant du plateau. Dans un sens, cette technique est la plus logique étant donné que les joueurs ne peuvent savoir que la marchandise se trouvant sur le dessus de la pile, regarder en dessus et donc calculer les prochains états du jeu relève donc de la triche.
Néanmoins nous avons essayé d'implémenter l'algorithme *minimax*, mais celui-ci ne fonctionne qu'avec une profondeur de 1. Notre implémentation n'était donc pas opérationnelle, nous avons conservé notre IA basique.