Projet cluedo

Gaétan Richard

M2 informatique 2023-2024

Le projet est à rendre pour le jeudi 26 octobre.

1 Introduction

Le but de ce projet est de réaliser une IA utilisant une formalisation logique afin de jouer à une variante (sans plateau) de Cluedo.

L'ensemble du travail effectué sera à rendre via e-campus sous une forme d'archive. Cette archive aura pour nom `login.tgz` où *login* sera remplacé par votre login. Cette archive contiendra (au moins) un dossier `doc` contenant un fichier `rapport.pdf` contenant les réponses aux questions et un dossier `bin` contenant des exécutables linux (ou des scripts) pour les différentes versions du programme. Le code sera également fourni dans un dossier `src`.

1.1 Les règles

On fixe un nombre N de joueur.

Il y a trois types de cartes : Lieux, Armes et Personnages. Chaque type de cartes comporte $N + 1$ cartes différentes.

Au début de la partie, on met de côté 1 carte aléatoire de chaque type. Le but du jeu est de deviner ces cartes.

On mélange ensuite les cartes restantes et on en distribue 3 par joueurs.

Les joueurs jouent à tour de rôle.

À leur tour, chaque joueur peux faire une des deux actions suivantes:

- **Hypothèse**: le joueur annonce à tous un lieu, une arme et un personnage.
  - si voisin possède au moins l'une des trois cartes, il montre l'une des cartes annoncées uniquement au joueur ayant émis l'hypothèse;
  - sinon, on demande au voisin suivant.
- **Accusation**: le joueur indique secrètement un lieu, une arme et un personnage:
  - si ceux-ci correspondent, il gagne la partie;

– sinon, il devient inactif et ne fait plus que répondre aux hypothèses des autres.

1.2 L'environnement technique

Les agents (joueurs) communiquent via une socket sur un port fixé.

Les objets sont décrits de la façon suivante:

- Les cartes `c` sont numérotées de 0 à $3N + 2$ (par type);
- Les actions sont décrites par des lettres en majuscule;
- les joueurs `j` sont numérotés de 0 à $N - 1$.

Les messages sont composés (sauf exception) du caractère indiquant l'action suivi des entiers correspondants aux arguments séparés par des espaces et terminés par un retour chariot.

1.2.1 Messages d'initialisation

Le client se connecte au serveur.

Le serveur envoie au client le message de bonjour `"B j N"` où j indique le numéro de joueur attribué et N le nombre de joueurs total.

Le client répond `"B login"` où *login* est le login persopass du créateur du client.

Une fois tous les joueurs présents, le serveur envoie le message `"C c1 c2 c3"` où dont les arguments correspondent aux trois cartes attribuées au joueur.

1.2.2 En cours de jeu

Lorsque le tour du joueur arrive, le serveur lui envoie le message de tour `"T"`.

Le client doit répondre par `"H c1 c2 c3"` s'il fait une hypothèse ou `"A c1 c2 c3"` s'il veut faire une accusation.

Si une hypothèse est demandée, le serveur envoie à tous les clients le message `"H c1 c2 c3"` reçu. Tant que le voisin v ne possède pas de carte, il envoie alors à tous le message passe `"P v"`. Si un voisin v possède une des trois cartes, le serveur demande alors le choix du joueur par le message `"C"`. Le client concerné doit alors montrer sa carte en répondant `"M c"` où c est une carte qui est demandé et que le client possède. Le serveur transmet alors au joueur ayant fait l'hypothèse le message `"M v c"` et il donne l'information `"M v"` (sans arguments) aux autres joueurs. Si jamais aucun joueur ne peux montrer de carte, le serveur envoie un message `"M"` sans argument.

1.2.3 Autres messages

Lorsque la partie est finie, le serveur envoie à tous les clients le message `"F j"` où `j` désigne le joueur gagnant. Il peut relancer directement une nouvelle partie avec les même joueurs en redonnant des cartes à chaque joueur.

En cas d'erreur, le serveur envoie un message d'erreur `"E e"` où e est le numéro d'erreur. Une liste de ces numéros est disponible en annexe. Dans ce cas, le serveur peut relancer une partie.

2 Mise en place

On vous donne un serveur sous forme de fichier python.

Pour toutes vos réalisations, on demande que le programme soit au bon nom dans le dossier `bin` et puisse être lancé avec comme argument le port de connexion (on se connectera à la machine locale).

2.1 Bot minimal

1. Faire une programme `bot0` qui se connecte, récupère ses cartes et propose lorsque c'est son tour l'accusation $(0, N, 2N)$
2. Enrichir ce programme pour fournir un programme `bot1` qui réponde correctement aux demandes `M` du serveur. Pour le moment, vous pourrez choisir comme vous voulez la carte à montrer s'il y en a plusieurs valides.

2.2 Version simple

Dans cette première version, votre IA va mémoriser les cartes correspondant à vos demandes (et uniquement celles-là). Pour cela, on stockera l'ensemble des cartes que l'on a vu. Au départ, il s'agit donc des trois cartes de la main.

Lorsque l'on souhaitera formuler une hypothèse, on choisira trois cartes (une de chaque type) que l'on a pas encore vu. Si on nous montre une des cartes, on mémorise que celle-ci est vue. Si aucune carte n'est montrée, c'est que l'on a trouvé la solution et on la donnera au tour suivant.

3 Logique

On souhaite maintenant améliorer le système de connaissance pour prendre en compte plus d'information. Pour cela, on va ajouter une représentation logique permettant de modéliser la connaissance et on va ajouter des règles de déduction.

On utilisera la notation $\mathbb{M}_i(c_1, c_2, c_3)$ pour indiquer que l'on sait que le joueur i possède une des cartes parmi c_1, c_2 ou c_3. Cette notation existe aussi pour une ou deux cartes ($\mathbb{M}_i(c_1)$ indique donc que le joueur i a la carte c_1). On utilisera aussi la notation $\mathbb{S}(c)$ pour indiquer que la carte c fait partie du secret.

3.1 Système de base

Notre système logique repose donc sur des littéraux de la forme:

- $\mathbb{M}_i(c)$, $\neg\mathbb{M}_i(c)$
- $\mathbb{M}_i(c_1, c_2)$, $\mathbb{M}_i(c_1, c_2, c_3)$

3.2 Ajout de connaissance

On ajoute ces littéraux en fonction des messages:

- On connait les cartes que l'on possède;
- Si un joueur *Passe* sur une hypothèse (c_1, c_2, c_3) alors on peut en déduire les littéraux $\neg\mathbb{M}_j(c_i)$ (pour $i \in [1, 3]$);
- Si un joueur j nous montre la carte c, on peux en déduire $\mathbb{M}_j(c)$
- Si un joueur j montre une carte sur une hypothèse (c_1, c_2, c_3) alors on peut en déduire $\mathbb{M}_j(c_1, c_2, c_3)$

3.3 Traitements élémentaires de la base de connaissances

On a aussi des règles de création de littéral:

- $\mathbb{M}_j(c) \rightarrow \neg\mathbb{M}_i(c)$ pour tout $i \neq j$
- $\mathbb{M}_j(C)$, $\neg\mathbb{M}_j(c)$ avec $c \in C \rightarrow \mathbb{M}_j(C \setminus c)$
- $\mathbb{M}_j(C_1) \quad \mathbb{M}_j(C_2), \quad \mathbb{M}_j(C_3), \quad \mathbb{M}_j(C_4) \rightarrow \mathbb{M}_j(\{c | c \in C_a \cap C_b, a \neq b\})$ (Cette dernière règle se base sur le fait que chaque joueur a au plus 3 cartes)
- $\neg\mathbb{M}_j(c)$ pour tout $j \rightarrow \mathbb{S}(c)$

On dispose de règle de simplification:

- Si on a $\mathbb{M}_j(C)$ et $\mathbb{M}_j(C')$ avec $C \subset C'$, alors on retire la règle inutile $\mathbb{M}_j(C')$.

1. Implémenter une IA sous forme de programme `bot2` qui utilise ce système logique. Faire en sorte que le programme affiche son état sur la sortie d'erreur lorsque l'on lui envoie le signal `SIGUSR1`.

2. Améliorer ce programme pour que l'IA fasse des hypothèses lui permettant d'obtenir des informations pertinentes et emette une accusation dès qu'elle a la réponse.

3.4 Optimisations des hypothèses et des réponses

On cherche maintenant à améliorer l'efficacité des hypothèses. Pour cela, nous allons utiliser des heuristiques.

L'idée est d'attribuer une valeur numériques aux cartes selon si on possède des informations partielles sur celle-ci. On fera alors des hypothèses sur les cartes ayant la plus petite valeur et on montrera les cartes avec le plus d'information.

- Indiquer dans votre rapport (en justifiant votre intuition) votre choix de valeur

- Implémenter ce système dans le programme `bot3`

4 Connaissance

À venir

5 Erreurs

- `101` : Erreur ou non réponse au bonjour
- `201` : Hypothèse ou accusation invalide
- `202` : Non réponse à la demande d'action
- `211` : Choix de carte à montrer invalide
- `212` : Absence de réponse pour une carte à montrer
- `301` : Limite du nombre de tour atteints