Programmation agents

Le but de ce projet est de coder un agent qui va, de manière autonome, poursuivre des buts. Pour cela, il devra observer son environnement et réagir de façon adéquate. L'agent aura peut être plusieurs façon d'arriver à ses fins. Il devra aussi être capable de réagir à des échecs.

La métaphore que nous avons choisi pour ce projet est un garçon de café ou un serveur dans un restaurant. Celui-ci doit s'occuper de plusieurs clients en même temps. Son but est de satisfaire le besoin de chaque client. On a en tête un petit restaurant qui ne possède qu'un seul serveur. Celui-ci doit donc s'occuper de tout :

- lorsqu'un client arrive, il doit lui trouver une table disponible
- une fois que le client est installé, il faut lui donner un menu
- après avoir donné un peu de temps, il faut prendre la commande
- il faudra apporter les plats à chaque fois que ceux-ci seront prêts
- quand le client le désirera, il faudra apporter l'addition et encaisser le payment

Comme le serveur doit s'occuper de plusieurs clients en même temps, à chaque fois qu'il a accompli une tâche (par exemple pris une commande ou faire assoir un client), il devra se demander quelle sera la tâche suivante et le faire en fonction du contexte le plus actuel. Par exemple, mieux vaut peut être s'attacher à accueillir un client qui vient d'arriver ou d'apporter un plat qui vient de sortir de la cuisine plutôt que de prendre la commande d'une autre table.

On va utiliser un modèle simplifié d'un restaurant. Le but est de programmer l'agent serveur. Le serveur a pour tâche de trouver une table lorsqu'un client arrive, lui apporter le menu, prendre sa commande, lui apporter sa commande dès que le l'assiette est prête, lui apporter l'addition quand le client la désire et encaisser le payment. On fera les hypothèses simplificatrices suivantes pour la modélisation :

- les clients arrivent un à un
- une table peut être occupée par au plus un client
- le client passe une commande à la fois, mais peut passer plusieurs commandes au cours de son repas.

On fera l'hypothèse que 10 clients vont entrer dans le restaurant qui ne contient que 5 tables.

1 Modèle sans communication – utilisation de percepts pour communiquer

Dans cette partie, il n'y a qu'un seul agent : le garçon, et donc, il n'y a pas de communication directe entre le garçon et ses clients. On va utiliser l'observation du garçon (donc des *percepts*) pour déduire ses actions. Par exemple si un client veut commander, l'agent recevra le percept `wait_to_order(P)` et l'agent devra alors réagir. Tout se passe comme si on avait déjà implémenté le mécanisme qui interprète un signal du client (le client lève la main, parle au garçon, ou a un comportement qui indique qu'il est prêt à commander), et l'agent reçoit donc un percept que le client est prêt à passer commande. Dans la suite, on va donner la liste de percepts que l'agent observera et la liste d'actions qui se trouvent à sa

disposition.

Liste des percepts

- `patron(P)` : un client `P` se trouve dans le restaurant
- `wait_for_seating(P)` : un client `P` cherche à s'assoir à une table.
- `table(T,P)` : décrit l'état de la table `T` : soit la variable `P` est associé au mot clé `free`, ce qui signifie que la table est libre, soit `P` est associé à un client `T`.
- `wait_for_menu(P)` : le client `P` désire avoir un menu
- `wait_to_order(P)` : le client `P` désire passer une commande
- `bill(P)` : le client `P` désire qu'on lui apporte l'addition
- `pay(P)` : le client `P` désire effectuer le payment de son addition.
- `dish(D,P)` : le plat `D` est prêt à être servi au client `P`
- `leave_patron(P)` : le client `P` quitte le restaurant

Au début de la simulation, la croyance initiale de l'agent sera que chaque table est libre.

Liste des actions

- `seat_patron(P)` : le serveur cherche une table libre pour le client `P`.
- `bring_menu(P)` : le serveur apporte un menu au client `P`.
- `take_order(P)` : le serveur prend la commande du client `P`. Lors de l'action, le client communique au serveur un des plats du menu (et seulement un seul).
- `bring_dish(P)` : le serveur apporte le plat au client `P`.
- `bring_bill(P)` : le serveur apporte l'addition au client `P`.
- `make_payment(P)` : le serveur encaisse le payment du client `P`.

Travail

Ecrire dans un fichier `garcon.asl` le code en AgentSpeak qui permet à l'agent de fonctionner.

2 Avec communication

Dans la partie précédente, on avait un seul agent qui recevait des signaux : un client veut commander, un client attend, un plat est prêt, etc... On veut maintenant passer à un système multiagent avec des agents qui modélise des clients et un agent qui modélise le serveur. Les agents vont pouvoir communiquer entre eux. Par exemple, lorsqu'un client entre dans le restaurant, il va demander au serveur à quelle table s'installer.

Pour cette partie, on va simplifier l'interaction au scénario suivant :

- un client entre
- il demande à s'asseoir, et le garçon lui indiquera une table dès qu'une table sera libre.
- une fois assis, il demande le menu
- une fois qu'il obtient le menu, il quitte la salle

Téléchargez le code source correspondant à la seconde partie. L'environnement est simplifié car l'agent ne reçoit plus certains percepts qui indique un changement d'environnement (par exemple, le serveur ne se rendra compte de la présence d'un client que si celui si demande à s'asseoir). L'environnement va envoyer des percepts à chaque agent, i.e. à chaque client individuellement ou au serveur.

- Pour le serveur : `table(T)` : la table numéro `T` existe (et est libre initialement)
- Pour le client : `entering(Id)` : le client `Id` entre dans le restaurant

On a trois actions disponibles pour l'environnement :

- actions du serveur
  - `seat_patron(P,T)` : le serveur fait asseoir le client `P` à la table `T`. Cette fois-ci, le serveur doit décider de lui même quelle table libre il va donner au client.
  - `bring_menu(P)` : le serveur apporte le menu au client `P`.
- action du client
  - `leave_room(Id)` : le client `Id` quitte le restaurant

Ecrivez le code du client et du serveur pour modéliser le scénario présenté ci-dessus. Vous devrez 1) utiliser de la communication entre les agents et 2) gérer vous-même les croyances des agents.

Travail

Ecrire dans les fichiers `garcon.asl` et `client.asl` le code en AgentSpeak pour les agents `garcon` et `client`.