COMP9414/9814 Artificial Intelligence

Session 1, 2017

Project 3, Option 2: Prolog (BDI Agent)

Due: Sunday 28 May, 11:59 pm
Marks: 18% of final assessment

Introduction

In this Assignment, you will be implementing an agent to move around in a rectangular environment, picking truffles (exotic mushrooms) and selling them to local restaurants. In doing so, you will implement the basic functions of a simple BDI Agent that operates in a Gridworld, and learn about the ideas underlying BDI agents.

Gridworld

The Gridworld consists of a two-dimensional grid of locations, extending to infinity in both directions. Truffles and restaurants appear at certain locations. The agent is able to move to a place where truffles are located and execute a `pick` action. After collecting a sufficient number of truffles, it can move to the location of a restaurant and execute a `sell` action. The agent can stay where it is, or move one square at a time, either horizontally or vertically. The world is dynamic in that truffles and restaurants can appear spontaneously at random locations at any time.

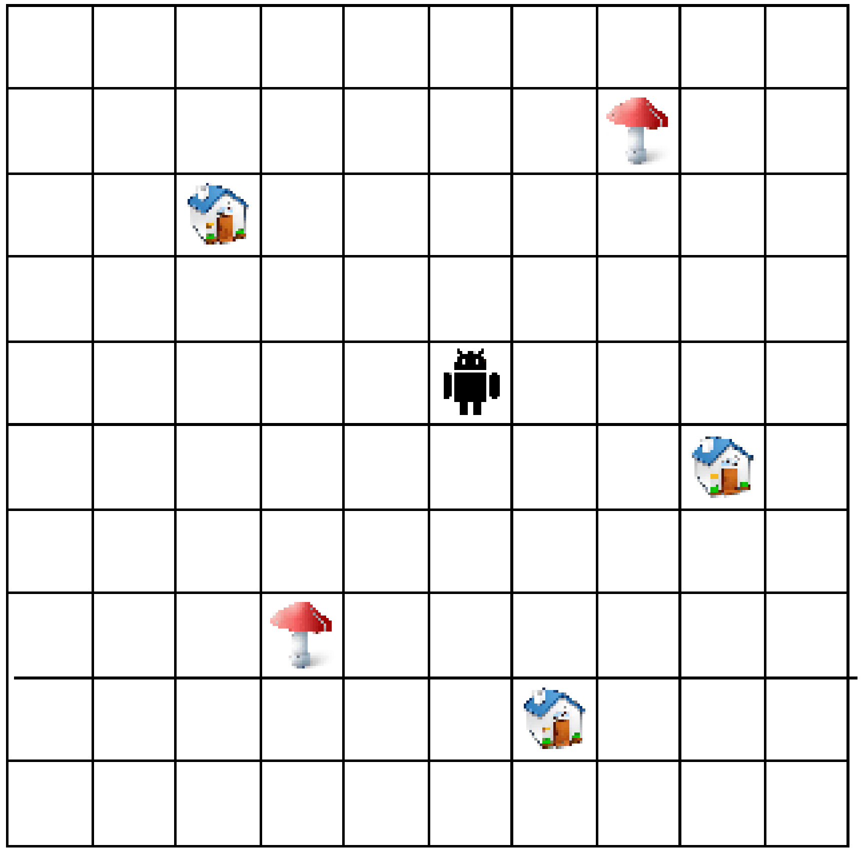

Figure 1: Gridworld State: Agent with truffles and restaurants

The supplied Prolog program `gridworld.pl` implements a system for conducting an experimental trial consisting of an agent in the Gridworld that repeatedly executes the BDI interpretation cycle for 20 iterations (this is a deliberately small number for ease of writing and debugging the program). The initial state of the world is always that there are no truffles or restaurants, and the agent is at location `(5,5)` holding no truffles.

The agent's *beliefs* at any time are in the form `beliefs(at(X,Y),stock(T))` meaning that the agent is at location `(X,Y)` and is currently holding a stock of `T` truffles. The initial belief state of the agent is represented by `beliefs(at(5,5),stock(0))`.

The agent's *goals* at any time are in the form `goals(Goals_rest,Goals_truff)` where `Goals_rest` is a list of locations of restaurants and the number of truffles they wish to buy, and `Goals_truff` is a list of locations of truffles, and the number of truffles at that location. Each goal of the agent is represented as a term `goal(X,Y,S)`, where `(X,Y)` is the location of either a truffle or a restaurant, and `S` is the number of truffles.

The agent's *intentions* are in the form `intents(Intents_sell,Intents_pick)` where `Intents_sell` and `Intents_pick` each consist of a list of pairs of the form `[Goal, Plan]`, representing a goal with an associated plan (which may be the empty plan), ordered according to some priority.

Each plan is a list of actions. To fulfil an intention, the agent executes the plan associated with its goal, which will make the agent move along a path towards the goal and then either `pick` or `sell` truffles. If, when the agent chooses an intention to fulfil, the plan associated with the goal of that intention is empty or cannot be executed, the agent creates a new plan for the goal and then begins to execute this plan.

In each cycle the agent executes one action. There are three types of action the agent can execute:

`move(X,Y)` - the agent moves to location `(X,Y)`
`pick(X,Y)` - the agent picks up the truffles at `(X,Y)`
`sell(X,Y)` - the agent sells truffles to the restaurant at `(X,Y)` and scores the associated points

BDI Interpreter

In each time cycle, the agent executes the interpreter shown abstractly in the table below. The new external events on each cycle are represented as a list of terms of the form `truffle(X,Y,S)` or `restaurant(X,Y,S)`, within some viewing distance of the agent. The agent will repeatedly perceive any truffle or restaurant so long as it remains in viewing range. It is not assumed that the agent can see all of the grid, so a new external event may occur as the agent is moving towards another target. Each new perceived event `truffle(X,Y,S)` or `restaurant(X,Y,S)` should trigger a goal for the agent, represented as a term of the form `goal(X,Y,S)`. Any new goal is incorporated into the agent's current intentions according to the agent's prioritization strategy (see below). The agent then selects one action for execution from the current set of intentions. Here the agent always selects the first intention on the list if there is one, creates or modifies the associated plan if necessary, then selects the first action in that plan, removes the selected action from the chosen plan, executes the action, and updates the list of intentions by removing any successfully achieved goals.

```
Abstract BDI Interpreter:
 initialize-state();
 do
   Percepts = get_new_external_events();
   G = trigger(Percepts);
   I = incorporate_goals(G, B, I);
   (I, A) = get_action(B, I);
   execute(A);
   Observation = observe(A);
   B = update_beliefs(Observation);
   I = update_intentions(Observation);
 until quit
```

The agent maintains separate lists of `sell` and `pick` intentions. Within each list, its prioritization strategy is very simple: without reordering existing goals, each new goal is inserted into the list of intentions in order of value (higher values before lower values), but if the new goal has the same value as existing goal(s), the new goal is inserted into the list of goals of the same value in order of distance from the current position (closer before further away). This means the agent maintains a "commitment" to pursuing its goals (the agent only changes its intention to pick or sell a higher value item or a closer item with the same value).

Assignment [18 marks]

You are supplied with a Prolog program in a file `gridworld.pl` that implements the experimental setup, including the generation of events (appearance of truffles and restaurants) and the execution of actions, and the agent's BDI interpretation cycle and observation functions.

[2 marks] Write a Prolog procedure `trigger(Events, Goals)` which takes a single list of events, each of the form `truffle(X,Y,S)` or `restaurant(X,Y,S)`, and computes the `Goals` for the agent in the form of two separate lists of items in the form `goal(X,Y,S)`. Your precedure should return `Goals` in the form `goals(Goals_rest,Goals_truff)` where `Goals_rest` and `Goals_truff` are both lists of items in the form `goal(X,Y,S)`.

[5 marks] Write a Prolog procedure
`incorporate_goals(Goals, Beliefs, Intentions, Intentions1)`

This procedure should take three inputs, as follows:

1. a set of `Goals` in the form `goals(Goals_rest,Goals_truff)`
2. a set of `Beliefs` in the form `beliefs(at(X,Y),stock(T))`
3. the current `Intentions` of the agent, in the form `intents(Int_sell,Int_pick)` where `Int_sell`, `Int_pick` are lists of intentions in the form `[goal(X,Y,S), Plan]`

Your procedure should return the updated `Intentions` of the agent after inserting the new goals from `Goals_rest` and `Goals_truff` into `Int_sell` and `Int_pick`, respectively. In each case, the new goals should be inserted into the existing list in decreasing order of `S`, using the Manhattan distance from the agent's current position to break ties. More precisely, a new goal should be placed immediately before the first goal in the list that has a lower value of `S`, or which has an equal value of `S` and is further away from the agent's current position (without reordering the current list of goals). Note that because of repeated perception of the same event, only new goals should be inserted into the list of intentions. The Plan associated with each new goal should be the empty plan (represented as the empty list `[]`).

[5 marks] Write a Prolog procedure
`get_action(Beliefs, Intentions, Intentions1, Action)`
which takes the agent's `Beliefs` in the form `beliefs(at(X,Y),stock(T))` and its current `Intentions` in the form `intents(Int_sell,Int_pick)` (as described above), and computes an action to be taken by the agent as well as the updated `Intentions`. The agent should select an intention as follows:

- If the list `Int_sell` of selling intentions is not empty, and its first item `[goal(X,Y,S), Plan]` satisfies the property that `S ≤ T` (i.e. the number of truffles in the agent's stock is greater than or equal to the number of truffles that the restaurant wants to buy) then this intention is selected;
- Otherwise, if the list `Int_pick` of picking intentions is not empty, then its first item `[goal(X,Y,S), Plan]` is selected;
- Otherwise, no intention is selected; in this case, the agent's `Intentions` should remain as they are, and it should stay in its current location (i.e. action is `move(X,Y)` if it is currently `at(X,Y)`).

The file `gridworld.pl` includes an `applicable()` predicate for testing whether an action is applicable. If the first action in the selected plan is applicable, the agent selects this action and updates the plan to remove the selected action. If there is no associated plan (i.e. the plan is the empty list) or the first action in the plan for the selected intention is not applicable in the current state, the agent should construct a new plan to go from its current position to the goal location and then either pick or sell truffles at that location. The plan will be a list of `move` actions followed by either a `pick` or `sell` action. The agent should then select the first action in this new plan, and update the list of intentions to incorporate the new plan (minus the selected first action). Due to the fact that there are no obstacles in the world, the exact path the agent takes towards the goal does not matter, so choose any convenient way of implementing this procedure.

[1 mark] Write a Prolog procedure
`update_beliefs(Observation, Beliefs, Beliefs1)`
to compute the new beliefs resulting from the agent's observations, as follows:

- `at(X,Y)` - the agent should believe it is `at(X,Y)`
- `picked(X,Y,S)` - `stock(T)` changes to `stock(T1)` where `T1 is T+S`
- `sold(X,Y,S)` - `stock(T)` changes to `stock(T1)` where `T1 is T-S`

[1 mark] Write a Prolog procedure
`update_intentions(Observation, Intentions, Intentions1)`
to update the agent's intentions, based on observation. An `at(X,Y)` observation should not change the agent's intentions. In the case of a `picked()` or `sold()` observation, the agent should remove the corresponding plan from its

list of intentions (since this plan has now successfully been executed).

There are 4 marks allocated for comments and programming style.

In general, a program that attempts a substantial part of the job but does that part correctly will receive more marks than one attempting to do the entire job but with many errors.

You can see an example of the output of a trial run by clicking here. Note: there was previously an error in Cycles 18 and 19 of this file. Please download the new version (updated 18 May 2017). The truffle (8,0,9) first appears in Cycle 2, but does not become visible to the agent until Cycle 18, at which point it generates an intention that is placed in front of the truffle at (5,4,3). Similarly, the new restaurant (8,6,5) should be placed in front of (3,1,0). The agent abandons its plan to move towards (3,1,0) and instead generates a new plan to move toward (8,6,5).

Submission

Submit one file called `agent.pl` using the command

```
give cs9414 hw3prolog agent.pl
```

Your solution should work with the supplied file `gridworld.pl`. **Do not change any of the procedures in this file and do not include the code from this file with your submission.**

The submission deadline is Sunday 28 May, 11:59 pm.
15% penalty will be applied to the (maximum) mark for every 24 hours late after the deadline.

Questions relating to the project can be posted to the Forums on the course Web site.

If you have a question that has not already been answered on the Forum, you can email it to `blair@cse.unsw.edu.au`

Plagiarism Policy

Your program must be entirely your own work. Plagiarism detection software will be used to compare all submissions pairwise (including submissions for any similar projects from previous years) and serious penalties will be applied, particularly in the case of repeat offences.

DO NOT COPY FROM OTHERS; DO NOT ALLOW ANYONE TO SEE YOUR CODE

Please refer to the UNSW Policy on Academic Honesty and Plagiarism if you require further clarification on this matter.

Good luck!
