COMP9414/9814 Artificial Intelligence

Session 1, 2018

Project 3, Option 2: Prolog (BDI Agent)

Due: Sunday 3 June, 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 up stones from the land and dropping them in the water, thus building a path to the location of a dangerous sea monster. The final stone must be dropped on the head of the sea monster in order to kill it.

In doing this assignment, 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. Some locations are specified as being on land, while the others are assumed to be water. In the first round of the simulation, the agent must construct a minimal list of locations in which stepping stones need to be dropped, in order to create a path to the location of the monster. In subsequent rounds, stones will appear randomly at certain locations. The agent is able to move to a place where a stone is located and execute a `pick` action. It can then move to one of the pre-computed drop locations and execute a `drop` 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 stones can appear spontaneously at random locations at any time.

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 100 iterations (you may like to reduce this number while debugging your program). The initial state of the world is always that there are no stones, and the agent is at location `(1,1)` holding no stones.

The agent's *goals* at any time are in the form `[goal(`X_1`,`Y_1`), ... , goal(`X_n`,`Y_n`)]` where `(`X_1`,`Y_1`), ... , (`X_n`,`Y_n`)` are locations where stones can be picked up.

The agent's *intentions* are in the form `intents(Intents_drop,Intents_pick)` where `Intents_drop` and `Intents_pick` each consist of a list of pairs of the form `[goal(X,Y), 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 `drop` a stone. 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 stone at `(X,Y)`
`drop(X,Y)` - the agent drops a stone at `(X,Y)`

`move(X,Y)` can be executed when the Manhattan distance from the agent's current location to `(X,Y)`

is either 0 or 1, and `land_or_dropped(X,Y)` is `true`.

`pick(X,Y)` can be executed if the Manhattan distance from the agent's current location to `(X,Y)` is exactly 1, `have_stones(0)` is `true` and `stone_at(X,Y)` is `true`.

`drop(X,Y)` can be executed if the Manhattan distance from the agent's current location to `(X,Y)` is exactly 1, `have_stones(1)` is `true` and `land_or_dropped(X,Y)` is `false`.

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 `stone(X,Y)`, within some viewing distance of the agent. The agent will repeatedly perceive any stone 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 `stone(X,Y)` should trigger a goal for the agent, represented as a term of the form `goal(X,Y)`. 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 `drop` 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 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 up a stone if new stone appears at a closer location.

Assignment

You are supplied with a Prolog program in a file `gridworld.pl` that implements the experimental

setup, including the generation of events (appearance of stones) and the execution of actions, and the agent's BDI interpretation cycle and observation functions.

When executed, the run command loads a file `land.pl` containing the location of the monster in the form `monster(X,Y)`, and a list of locations which are on land in the form `land(X,Y)`. The current location of the agent is specified by a dynamic predicate `agent_at(X,Y)`.

[4 marks] Write a Prolog procedure

```
initial_intentions(Intentions)
```

which binds `Intentions` to `intents(L,[])` with `L` in the form `[[goal(X1,Y1),[]], ... , [goal(Xn,Yn),[]]]`. Here `(Xn,Yn)` is the location of the monster and `(X1,Y1)`, ... , `(Xn-1,Yn-1)` are places where the mininum number of stones need to be dropped in order to allow the agent to move along some path from its current location to that of the monster.

[1 mark] Write a Prolog procedure

```
trigger(Percepts, Goals)
```

which takes a list of percepts, each of the form `stone(X,Y)`, and converts it into a corresponding list of goals, each of the form `goal(X,Y)`.

[4 marks] Write a Prolog procedure


```
incorporate_goals(Goals, Intentions, Intentions1)
```


This procedure should take two inputs, as follows:

1. a set of `Goals` in the form of a list `[goal(X1,Y1), ... , goal(Xn,Yn)]`
2. the current `Intentions` of the agent, in the form `intents(Int_drop,Int_pick)` where `Int_drop`, `Int_pick` are lists of intentions in the form `[goal(X,Y), Plan]`

Your procedure should return the updated `Intentions` of the agent after inserting the new goals into `Int_pick`. The new goals should be inserted into the existing list in decreasing order of the length of the shortest valid path from the agent's current position. A valid path is one which passes through only locations `(X,Y)` for which `land_or_dropped(X,Y)` is true. More precisely, a new goal should be placed immediately before the first goal in the list whose path length is longer than that of the new goal (without reordering the current list of goals). If no such valid path exists, then the new goal should not be inserted. Note that because of repeated perception of the same event, only goals not already in the list 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 `[]`).

[4 marks] Write a Prolog procedure

```
get_action(Intentions, Intentions1, Action)
```

which takes the agent's current `Intentions` in the form `intents(Int_drop,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 agent is currently holding a stone, indicated by `agent_stones(1)`, then the first intention `[goal(X,Y), Plan]` in the list `Int_drop` of dropping intentions is selected;
- otherwise, if the list `Int_pick` of picking intentions is not empty, then its first item `[goal(X,Y), 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 drop a stone at that location. The plan will be a list of `move` actions followed by either a `pick` or `drop` 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).

[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 `dropped()` 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 that `goal(9,2)` is not inserted into the list of intentions until after a stone has been dropped at `(5,3)` in Cycle 8 (thus greating a viable path). At Cycle 38, the agent abandons its Plan for `goal(2,3)` and instead gives priority to the new stone that just appeared at `(3,7)`.

Path Search Code

For this assignment, you are free to copy any of the path search code supplied for Assignment 2, and adapt it for the current task. You might find `pathsearch.pl` and `ucsdijkstra.pl` particularly useful.

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 3 June, 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!
