Intelligent Systems

Assignment 2. Programming Intelligent Agents

Assignment description

An agent, as defined in our textbook is anything that can perceive its environment through sensors, and act upon that environment through actuators based on its agent program. In this AI programming assignment, you must create a jupyter notebook that codifies and describes the implementation of two intelligent agents that should find and pick up tools and treasures in two types of grid environments. The key difference between the agents is that one of them has to be programmed as a simple reflex agent while the other has to be programmed as a model-based reflex agent. The key difference between the environments is that one is a fully-observable while the other is partially-observable.

The grid environment (defined as a subclass of the AIMA class Environment – **do not use** XYEnvironment) should be created as a matrix of 6 rows and 6 columns. The location of Thing objects in the environment (treasures, tools, walls, and the agent) has to be described by their (r, c) coordinates, where r specifies the row position on the grid while c specifies the column position. The origin of the environment, with coordinates (1, 1), is located at the upper-left corner of the grid. An environment cell can contain only one thing and eventually the agent.

The environment can contain 2 types of treasures and 2 types of tools. These treasures and tools can be placed in any quantity and location within the environment. To grab a type 1 treasure, identified with a capital letter T, the agent requires to have previously grabbed and have a reusable tool, identified with a capital letter H. To grab a type 2 treasure, identified with a lowercase letter t, the agent requires to have previously grabbed and have a disposable tool, identified with a lowercase letter h. The agent can use the same reusable tool to grab as many type 1 treasures as he wants, but the disposable tools can only be used once. If you want to grab another type 2 treasure, the agent must find and grab another disposable tool.

The agents could move inside the environment with the actions “Right”, “Left”, “Up”, and “Down”, which modify the location of the agent by increasing the column number, decreasing the column number, decreasing the row number, and increasing the row number, respectively. The agent can also decide to do nothing with the action “NoOp”. The agent also has actions to grab each different type of thing: "Greuse", "Gdispos", "GTreas1" and "GTreas2" to grab a reusable tool, a disposable tool, a type 1 treasure and a type 2 treasure, respectively.

The result of executing an action is as follows:

- The performance measure of the agent starts with 50 points.
- Every intent to move reduces one point from the agent’s performance measure.
- If there is a wall Thing in the target position of the movement, the agent maintains its current position.
- If the agent intent to move outside the grid, it maintains its current position and reduces its performance in 5 points.
- To grab a treasure or tool, the agent must enter the cell that contains such things.
- Every intent to grab a tool reduces two points from the agent’s performance measure. A grabbed treasure must be eliminated from the environment.
- If the agent moves to a cell that contains a treasure Thing and the agent have an adequate tool, as its next action it can grab the treasure and add the treasure value to its performance measure. A type 1 treasure has a value of 20 and a type 2 treasure has a value of 40. A grabbed treasure must be eliminated from the environment.
- Every disposable tool used must be eliminated from the agent’s tools.
- The “NoOp” action maintains the agent’s location and performance measure.

You can add an adequate quantity of treasures, tools, and walls into the environment. Use a capital letter X to identify walls, and a character – to mark empty cells.

The perception, that the agent program receives, describes the current location of the agent's cell, the list of tools that the agent has, and a description of what the agent can see.

The first environment is completely observable, so the agent can perceive the complete grid with all the things it contains. The second environment is partially observable and the agent can only perceive the contents of its current cell and the eight cells adjacent to it.

The environment's step method must display the full state of the environment (grid). and internal state of the agent before and after executing the agent's action. It also must display the percept and the selected action by the agent's program.

Examples:

Full environment state containing 6 wall cells (X), 2 type 1 treasures (T), 2 type 2 treasures (t), 2 reusable tools (H), and 2 disposable tools (h), where the agent is in the 3rd row and 2nd column, and it already has a disposable tool:

```
Agent location: (3, 2)
Agent tools: [h]
\ 1 2 3 4 5 6 /
1 H - - - - T 1
2 - X X X h - 2
3 - - h - H - 3
4 - T - X - - 4
5 - - - X - t 5
6 t - - X - - 6
/ 1 2 3 4 5 6 \
```

The percept for both agents in the full observable environment includes all the above information.

For the case of the partially observable environment, the percept for the agent in the above state would be:

```
Agent location: (3, 2)
Agent tools: [h]
\ 1 2 3 /
2 - X X 2
3 - - h 3
4 - T - 4
/ 1 2 3 \
```

And the internal state for the model-based reflex agent that started at location (5, 2) and moved "Right" to location (5, 3) in the above environment:

```
\ 1 2 3 4 5 6 /          \ 1 2 3 4 5 6 /
1 ? ? ? ? ? ? 1          1 ? ? ? ? ? ? 1
2 ? ? ? ? ? ? 2          2 ? ? ? ? ? ? 2
3 ? ? ? ? ? ? 3    =>    3 ? ? ? ? ? ? 3
4 - T - ? ? ? 4          4 - T - X ? ? 4
5 - - - ? ? ? 5          5 - - - X ? ? 5
6 t - - ? ? ? 6          6 t - - X ? ? 6
/ 1 2 3 4 5 6 \          / 1 2 3 4 5 6 \
```

The question mark characters (?) represent the part of the grid that the agent haven't seen.

The grids that are shown are just the expected depictions of the environment, percepts and internal state of a model-based reflex agent, but internally you can represent them as you prefer.

Test both agents in both environments under different circumstances (different numbers of things in the environment with different locations and starting with the agent in different locations) to visualize their behaviors, compare their performance measures, generate and report some conclusions. For example, who behaves better, the simple reflex agent or the model based reflex agent? How do different types of environments affect different types of agents? Do you think the agents are behaving rationally? What did you have to program differently for agents to work in different environments? And so on…

Run the Environments for different numbers of time steps. Start displaying the initial state of the environment and the agent initial performance measure value. Something like the following:

MODEL-BASED REFLEX AGENT in PARTIALLY OBSERVABLE ENVIRONMENT
<STARTING>

```
Agent location: (5, 2)
Agent tools: []
\ 1 2 3 4 5 6 /
1 H - - - - T 1
2 - X X X h - 2
3 - - h - H - 3
4 - T - X - - 4
5 - - - X - t 5
6 t - - X - - 6
/ 1 2 3 4 5 6 \

Agent performance: 50
```

Then, for each of the steps, show the percept received by the agent's program, the internal state of the agent (only for the model-based reflex agent), the name of the selected action, and finally, the new performance measure for the agent and the new environment state. Something like the following for a partially observable environment:

```
<STEP 1>
PERCEPT
Agent location: (5, 2)
Agent tools: []
\ 1 2 3 /
4 - T - 4
5 - - - 5
6 t - - 6
/ 1 2 3 \
AGENT'S INTERNAL STATE
\ 1 2 3 4 5 6 /
1 ? ? ? ? ? ? 1
2 ? ? ? ? ? ? 2
3 ? ? ? ? ? ? 3
4 - T - ? ? ? 4
5 - - - ? ? ? 5
6 t - - ? ? ? 6
/ 1 2 3 4 5 6 \
```

```
SELECTED ACTION: Right
NEW AGENT'S PERFORMANCE: 49
NEW ENVIRONMENT STATE
Agent location: (5, 3)
Agent tools: []
\ 1 2 3 4 5 6 /
1 H - - - - T 1
2 - X X X h - 2
3 - - h - H - 3
4 - T - X - - 4
5 - - - X - t 5
6 t - - X - - 6
/ 1 2 3 4 5 6 \

<STEP 2>
PERCEPT
Agent location: (5, 2)
Agent tools: []
\ 2 3 4 /
4 T - X 4
5 - - X 5
6 - - X 6
/ 2 3 4 \
AGENT'S INTERNAL STATE
\ 1 2 3 4 5 6 /
1 ? ? ? ? ? ? 1
2 ? ? ? ? ? ? 2
3 ? ? ? ? ? ? 3
4 - T - X ? ? 4
5 - - - X ? ? 5
6 t - - X ? ? 6
/ 1 2 3 4 5 6 \
SELECTED ACTION: Up
NEW AGENT'S PERFORMANCE: 48
NEW ENVIRONMENT STATE
Agent location: (4, 3)
Agent tools: []
\ 1 2 3 4 5 6 /
1 H - - - - T 1
2 - X X X h - 2
3 - - h - H - 3
4 - T - X - - 4
5 - - - X - t 5
6 t - - X - - 6
/ 1 2 3 4 5 6 \
```

And so on...

Delivery instructions:

- AIMA files cannot be modified and must not be delivered with the assignment.
- Only one of the team members must upload the completed `assignment02.ipynb` jupyter notebook file to Blackboard.

- The solution must be contained within the compressed M.zip file, where M must be replaced by the student ids of the team members. For example, `A01111111_A00999999.zip` should contain the solution of the team whose members are A01111111 and A00999999.
- The notebook must include the team data.

Evaluation criteria:

As for the programming of the intelligent agent, the grading criteria is:

- The assignment was not delivered (0).
- The notebook lacks many elements and the program does not run (40).
- The notebook includes most of the elements but the program does not run (60).
- The notebook includes most of the elements, the program runs, but not as asked (80).
- The notebook includes all the elements and runs as asked (100).

The grade will be augmented (awarded) or reduced (penalized) depending on the quality of the notebook documentation with markdown text, the internal documentation of the Python code, and the writing of the conclusions.