Nada Bounajma 77807

Spring 2022 - Artificial Intelligence Project

The Wumpus World :

Creation of a Knowledge base Agent using Prolog

1. Programming Language Used

The purpose of the project was to create a Knowledge Base Agent that will interact logically with the Wumpus World environment. For that matter, the agent is coded solely using Prolog. This logic programming language will allow us represent the state of our program as a set of logical relations (e.g., Wumpus is in room (1, 3), gold is in room (2, 2)) that uses natural language understanding. Also, Prolog is nonprocedural, meaning we as programmers only specify what goals are to be accomplished but not how to accomplish those goals.

2. How to Run the Code

- The prolog files should be in the same directory.
- In the Prolog environment, compile the code using the following query :

```
?- [main2],[heuristic2],[world].
```

- Then run the code using the following query :

```
?- run.
```

3. Run Using Different Worlds

To modify the Wumpus World configuration, you can go to *world.pl* file, then add, remove, or change the location of the pits or Wumpus.

4. Code Description

Some Key Predicates of our Code

note : R(X, Y) is the same thing as X, Y.

`hunter(X, Y)` ==> hunter is in room (X, Y)
`wumpus(X, Y)` ==> wumpus is in room (X, Y)
`pit_at(X, Y)` ==> a pit is in room (X, Y)
`gold(X, Y)` ==> gold is in room (X, Y)
`visited(X, Y)` ==> a room (X, Y) can be marked visited
`score(X)` ==> represents the total score X of the hunter
`steps(X)` ==> represents the total steps X done by the hunter
`grab(X, Y)` ==> means that the action grab was done on room (X, Y)
`shooted(X, Y)` ==> the room (X, Y) has been shot
`in_bounds(X, Y)` ==> checks if a certain room is within our world

`isAdjacent((X, Y), (XT, YT))` ==> checks if two rooms are adjacent to each other

State of the Players

`hunter_is()` returns Dead if hunter and wumpus, or hunter and pit are in the same room
`wumpus_is()` returns Dead if wumpus is in a room that has been shot

What the Hunter Currently Holds

`has_arrow()` returns no if a room has been shot
`has_gold()` returns yes if the hunter was in a room that has gold and already performed the grab action in it

Actions that will be done

`move(X, Y)` ==> move to room (X, Y)
`shoot(X, Y)` ==> shoot room (X, Y)
`grab` ==> perform the action Grab to get the gold

Perceptions

`has_glitter()` returns Yes if the hunter and gold are in the same room
`has_stench()` returns Yes if the wumpus is adjacent to the hunter
`has_breeze()` returns Yes if a pit is adjacent to the hunter
`has_scream()` returns Yes if wumpus is Dead

Cases where it will fail

- **If Wumpus is in a Corner room** : we did not know how to implement the walls in our project due to lack of time (we did try to implement them, but since we left it as the very last element to add, we struggled to integrate it properly in our code)

In this example, the Wumpus is in Room(4x1), which is a corner room. But since our heuristic did not detect that there is a Wumpus in that room, it got chosen as a safe room by our heuristic and informed the Agent that the room is Safe.

```
At time 42: hunter is in room (2x3), and senses [no,no,no,no].

Nothing is perceived, Hunter will move to a random adjacent room...
Hunter is Moving to room (2x4)

At time 43: hunter is in room (2x4), and senses [no,no,no,no].

Nothing is perceived, Hunter will move to a random adjacent room...
Hunter is Moving to room (3x4)

At time 44: hunter is in room (3x4), and senses [no,no,no,no].

Nothing is perceived, Hunter will move to a random adjacent room...
Hunter is Moving to room (3x3)

At time 45: hunter is in room (3x3), and senses [no,no,no,no].

Nothing is perceived, Hunter will move to a random adjacent room...
Hunter is Moving to room (4x3)

At time 46: hunter is in room (4x3), and senses [no,no,no,no].

Nothing is perceived, Hunter will move to a random adjacent room...
Hunter is Moving to room (4x2)

At time 47: hunter is in room (4x2), and senses [yes,no,no,no].
The Wumpus is near.
The Wumpus is near.

> Costs [40,40,20] for [[4,3],[3,2],[4,1]] selected 20
Hunter is Moving to room (4x1)

You Lost, Wumpus is sill Alive...
----Game has Finished---
```

Cases where it will win

Hopefully will work in the rest of the configurations. The Wumpus in this case is in room (1x3) :

```
?- run.
At time 0: hunter is in room (1x1), and senses [no,no,no,no].

Nothing is perceived, Hunter will move to a random adjacent room...
Hunter is Moving to room (2x1)

At time 1: hunter is in room (2x1), and senses [no,no,no,no].

Nothing is perceived, Hunter will move to a random adjacent room...
Hunter is Moving to room (1x1)

At time 2: hunter is in room (1x1), and senses [no,no,no,no].

Nothing is perceived, Hunter will move to a random adjacent room...
Hunter is Moving to room (2x1)

At time 3: hunter is in room (2x1), and senses [no,no,no,no].

Nothing is perceived, Hunter will move to a random adjacent room...
Hunter is Moving to room (1x1)

At time 4: hunter is in room (1x1), and senses [no,no,no,no].

Nothing is perceived, Hunter will move to a random adjacent room...
Hunter is Moving to room (2x1)

At time 5: hunter is in room (2x1), and senses [no,no,no,no].

Nothing is perceived, Hunter will move to a random adjacent room...
Hunter is Moving to room (2x2)

At time 6: hunter is in room (2x2), and senses [no,no,no,no].

Nothing is perceived, Hunter will move to a random adjacent room...
Hunter is Moving to room (1x2)

At time 7: hunter is in room (1x2), and senses [yes,no,no,no].
The Wumpus is near.
Hunter Shot an arrow at room (1x3)
 !You Won! You killed the Wumpus!
----Game has Finished---
```

Struggles Faced

The lack of familiarity we have with Prolog played a major role in determining the correctness of our code (we went through it very briefly in Languages and Compilers course) especially when

creating our heuristic function as it required a very deep understanding of how to implement lists. Also, the Error messages and order of execution that the Prolog's runtime environment uses was very confusing and does not make it easy for us debug our codes in the first place. But recently we found out about the Trace Mode (displays the order of the execution line by line) and only then that we were able to find Errors and Debug our files.