CZ3005 Artificial Intelligence

Team KJS

Members & Contributions:

Members	Contributions
Satini Sankeerthana (U1922470G)	Agent Code, Report
Yap Xuan Ying (U2021297G)	Agent Code, Report
Tok Jeng Wen (U1922943D)	Driver Code, Report

Agent Code

The Agent Code was coded in Prolog and constituted the major actions that the agent could do and percepts that it could receive. The Agent implements the following terms:

1. **Reborn/0**

Task: Implements Agent's reset due to arriving into a cell inhabited by the Wumpus.
Approach: Included asserting all the dynamic variables and changing the values of X,Y in current(X,Y,D) to 0 and the direction to North. At the same time, any previous sequences of percepts are also retracted.
Output: The agent is placed in the starting grid cell (0,0) and has no knowledge of its surroundings.

2. **Move(A,L)/2**

Task: Carries out the action to be done given the list of the percepts by the Driver.
Approach: We have splitted the logic for this term into different cases along with using the Implication Reasoning for this term. For example, if the action to be done is moveForward then update the sensors according to the percept sequence L that is also sent by the driver.
Output: This would result in the agent carrying out the action as queried in by the driver and at the same time makes sense of the percepts sent in by the driver and changes its understanding of the environment accordingly.

3. **Reposition(L)/1**

Task: Implements Agent Reset due to the agent stepping into the same grid as the Confundus Portal or at the beginning of the game.
Approach: The reposition(L) is triggered when the position of the confundus and the current position of the agent (current(X,Y)) are the same. As a result the sensors are updated via UpdateSensor and knowledge is cleared.
Output: This would result in the agent being reborn again.

4. **hasArrow/0**

Task: Returns True if the arrow has not been shot at the wumpus yet, else returns False.
Approach: We have implemented hasArrow as a dynamic object that is retracted from the existing world when the agent shoots at the wumpus. We have ensured that the agent is only able to shoot at the wumpus if it is in the same row or the same column as the wumpus and facing the wumpus.
Output: As a result of the shoot command and hasArrow being retracted, if the wumpus has been killed, a scream will be generated.

5. **Current(X,Y,D)**

Task: X and Y tells us the current relative position while the D tells us the current relative direction the agent is facing.
Approach: We have implemented this as a dynamic object that is changed every time the moveForward action is commanded. Based on the direction the agent is facing either it's X or Y value will be changed.
Output: This is the key command that allows the game to progress by enabling the agent to traverse around the grid while getting the percepts of the environment.

6. **Localisation and Mapping**

    a. **Visited(X,Y)**

    Task: Check whether the cell has been visited previously or not.
    Approach: We carry out a test to check if there is a cell in the grid (ie: the agent has not gone off world) and use the conjunction operator with the set of unvisited cells that we have been keeping track of. This results in an outcome of whether the current cell is an unvisited cell, which is then negated to obtain the outcome of whether a cell is visited.
    Output: Returns True or False depending on whether the cell has been visited or not.

    b. **Wumpus(X,Y)**

    Task: Check for the presence of a wumpus.
    Approach: We have implemented this by taking the conjunction of the outcomes of the percept stench(a,b), where a and b are each of the grid cells values that

are adjacent to cell(X,Y). This tells us whether there is a wumpus present in the cell (X,Y) only after we have gotten 4 precepts of stench in the specific cells.
`stench(X+1,Y),stench(X,Y+1), stench(X,Y-1), stench(X-1,Y)`
Output: Returns True if the agent knows or guesses that there is a wumpus in that cell (X,Y).

c. **Confundus(X,Y)**
Task: Check for the presence of a Confundus in the cell (X,Y).
Approach: We have implemented confundus(X,Y) similar to how we have implemented Wumpus(X,Y) where we take the conjunction of the outcome of the percept tingle(a,b), where a and b refer to the adjacent cells of (X,Y) to check whether there is a confundus or not. We conclude that a confundus exists only after we get 4 percepts of tingle(a,b) = True for the specific values of a and b.
`tingle(X+1,Y),tingle(X,Y+1), tingle(X,Y-1), tingle(X-1,Y).`
Output: Returns True if the agent reasons that the cell (X,Y) may contain a Confundus.

d. **Safe(X,Y)**
Task: Check whether the cell is safe.
Approach: We have implemented this by taking the conjunction of the outcomes of:
1. Whether Cell(X,Y) is a legitimate cell
2. Negation of whether there is a confundus present
3. Negation of whether there is a wall present
4. Negation of whether there is a wumpus present

`\+wumpus(X,Y);\+confundus(X,Y),cell(X,Y),\+wall(X,Y).`
Output: Returns True if the cell(X,Y) is safe if it is a legitimate cell and has no confundus, wall and wumpus in it.

e. **Wall(X,Y)**
Task: Check whether there is a wall in the grid cell (X,Y)
Approach: We implemented wall(X,Y) if the agent is able to percieveBum
Output: Returns True if the agent can reason that there is a wall in cell(X,Y)

7. **Explore(L)**

Task: Return a sequence of actions that can guide the agent into a safe cell that has not been visited.
Approach: The explore term makes use of the heuristic(H) term, which takes its outcome and appends it into a list to form a sequence of actions. The Heuristic(H) term, uses Implication Reasoning, where if there is a certain percept on the current cell(X,Y,) its corresponding string value is assigned to H. This H is then returned to Explore(L) and is appended into a list. For example: `(current(X,Y,_),glitter(X,Y) -> H is ["pickup"]);`

Output: Returns True if L contains a sequence of actions that can guide the agent into a safe, unvisited cell.

Driver Code

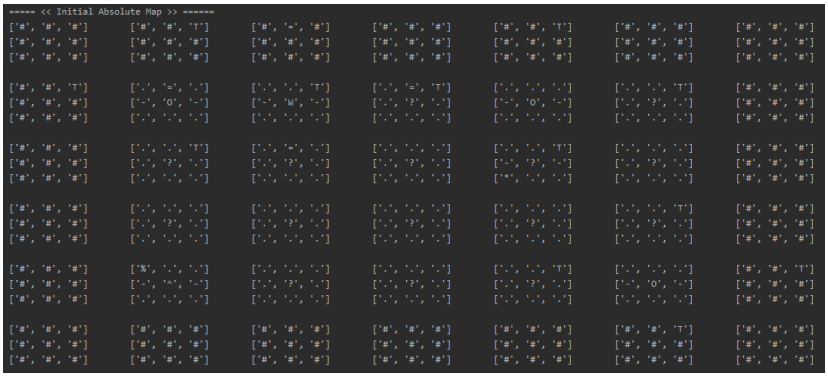
The purpose of the driver code is to simulate the movement of the agent and update it onto the absolute map. Even though the random placement has been coded out: def **generate_elements_random**(self), we are using a fixed map to better document our results:

#	#	#	#	#	#	#
#	P	W		P		#
#				C		#
#						#
#	A				P	#
#	#	#	#	#	#	#

P = Portal - A= Agent - C = Coin, W = Wumpus

Within each map cell, there is a 3x3 matrix which contains the symbols as stated in the assignment requirements. In each of the map cells, it contains the respective percepts that is to be sent to the agent. (eg. glitter= False, Stench = False). The stench and tingle symbols are set in the cell adjacent to the Wumpus and Portal respectively during the generation of the absolute map.

Array output with their symbols (*Please refer to "TeamKJS-testPrintout-Self.txt" in the zip file for clearer output)*



Given every move, the driver queries move(A,L) to the agent with L being a list of on/off strings.

Test Actions	# *Will kill the wumpus* = ['moveforward', 'moveforward', 'pickup', 'turnright', 'moveforward', 'turnleft', 'shoot'] # *Pickup_coin* = ['moveforward', 'moveforward', 'turnleft', 'moveforward', 'turnright', 'turnright', 'moveforward', 'moveforward', 'moveforward', 'pickup']
Successful moveforward: During the execution of each successful 'moveforward', the Agent leaves behind an "S" in symbol 5, marking it as a visited and safe cell. **Not Successful moveforward** | bump wall :	

The current location of the agent will stay the same, and the bump indicator will be ON for the current cell.

Not Successful moveforward | Step into Portal:
The agent will be informed of the repositioning & the agent is removed from the absolute map and placed into another cell that does not contain the wumpus / a portal.

Not Successful moveforward | Wumpus :
The entire map is reset and the agent is sent back to the starting point.

Successful shoot:
The driver checks if the agent is facing the wumpus, if yes, the scream indicator will be triggered and the wumpus is removed from the map. Else, the same old precepts will be fed back to the agent.

Successful pickup:
If the glitter indicator is ON, the agent will successfully pick up the coin, and the coin will be removed from the map. Else, the same old precepts will be fed back to the agent.

Example (Bump scenario) :

Initial position >	moveforward >	moveforward>	turnleft>	moveforward
['#', '#', 'T'] ['#', '#', '#'] ['#', '#', '#']	['#', '#', 'T'] ['#', '#', '#'] ['#', '#', '#']	['#', '#', 'T'] ['#', '#', '#'] ['#', '#', '#']	['#', '#', 'T'] ['#', '#', '#'] ['#', '#', '#']	['#', '#', 'T'] ['#', '#', '#'] ['#', '#', '#']
['.', '=', '.'] ['-', 'O', '-'] ['.', '.', '.']	['.', '=', '.'] ['-', 'O', '-'] ['.', '.', '.']	['.', '=', '.'] ['-', 'O', '-'] ['.', '.', '.']	['.', '=', '.'] ['-', 'O', '-'] ['.', '.', '.']	['.', '=', '.'] ['-', 'O', '-'] ['.', '.', '.']
['.', '.', 'T'] ['.', '?', '.'] ['.', '.', '.']	['.', '.', 'T'] ['.', '?', '.'] ['.', '.', '.']	['.', '.', 'T'] ['-', '^', '-'] ['.', '.', '.']	['.', '.', 'T'] ['-', '<', '-'] ['.', '.', '.']	['.', '.', 'T'] ['-', '<', '-'] ['.', 'B', '.']
['.', '.', '.'] ['.', '?', '.'] ['.', '.', '.']	['.', '.', '.'] ['-', '^', '-'] ['.', '.', '.']	['.', '.', '.'] ['-', 'S', '-'] ['.', '.', '.']	['.', '.', '.'] ['-', 'S', '-'] ['.', '.', '.']	['.', '.', '.'] ['-', 'S', '-'] ['.', '.', '.']
['%', '.', '.'] ['-', '^', '-'] ['.', '.', '.']	['%', '.', '.'] ['-', 'S', '-'] ['.', '.', '.']	['%', '.', '.'] ['-', 'S', '-'] ['.', '.', '.']	['%', '.', '.'] ['-', 'S', '-'] ['.', '.', '.']	['%', '.', '.'] ['-', 'S', '-'] ['.', '.', '.']
['#', '#', '#'] ['#', '#', '#'] ['#', '#', '#']	['#', '#', '#'] ['#', '#', '#'] ['#', '#', '#']	['#', '#', '#'] ['#', '#', '#'] ['#', '#', '#']	['#', '#', '#'] ['#', '#', '#'] ['#', '#', '#']	['#', '#', '#'] ['#', '#', '#'] ['#', '#', '#']

Conclusion

Through this project we have learnt how to translate propositional logic and reasoning into Prolog Code. Doing this helped us better understand the relationship between different logics and allowed us to translate it into code. Another new learning experience would be the connection of Prolog and Python. This was an eye-opening task for us as we previously were unaware that something as abstract and less calculation heavy as Logic and Reasoning can be combined with Python to solve stochastic and complex problems.