CZ3005 Artificial Intelligence – Lab Assignment 2
Learning to Use Prolog as a Logic Programming Tool

Team members: Anagha S. Ani, Chin Kin Mun, Lau Chen Yi Wynne

1. For each task, explain your approach, output, etc. In particular, explain how your printouts demonstrate correctness. Enumerate maps within the reports when necessary.
   - **The Agent**
     - reborn/0
       - Used retractall() to remove all facts in the knowledge base (KB) of the agent, followed by using assert() to make sure that the agent spawns at cell (0,0) of the relative map and has an arrow.
       - Agent now has safe(0,0), current(0,0,rnorth), visited(0,0) and arrow(1) in its KB.
     - move(A,L)
       - For each movement, blocks adjacent to the final block will be updated based on the perceptions received from the driver. This is done through setblock(L).
       - shoot: checks if the agent has an arrow, if yes, remove the arrow. Also, if position of the wumpus has already been confirmed, all stench(X,Y) and wumpus(X,Y) will be retracted from KB, and existsWumpus(0) will be asserted, to indicate that the wumpus is dead, when the arrow is shot.
       - moveforward: moves the agent one cell forward according to the direction they are facing using current(X,Y,D).
       - turnleft: updates the direction the agent is facing to its left using current(X,Y,D).
       - turnright: updates the direction the agent is facing to its right using current(X,Y,D).
       - pickup: if glitter(X,Y) is present in the current cell, it will be retracted. Also, coin(X,Y) will be asserted, to indicate that the coin here has already been picked up.
     - setblock(L):
       - resets when Confundus is on.
       - assert(stench(X,Y)) when Stench is on.
       - assert(tingle(X,Y)) when Tingle is on.
       - assert(glitter(X,Y)) when Glitter is on.
       - assert(wall(X,Y)) through bump_update when Bump is on.
       - Evaluation of adjacent blocks is also done when receiving perceptions directly from the driver. surveyadj(X,Y) will do that evaluation of adjacent blocks. X is the x component of the current block and Y is the y component of the current block.
     - surveyadj(X,Y) checks the perception of the current block and evaluates the adjacent block based on the current block. The following are the states of the block where the adjacent blocks will be updated:
       - No tingle(X,Y) and no stench(X,Y) in the current cell. Will assert that the adjacent cells are safe(X,Y).

    - noConfundus(X,Y) and noWumpus(X,Y) in the current cell. Will assert that the adjacent cells are safe(X,Y).
    - Stench. Will assert that the adjacent cells are wumpus(X,Y) unless it is visited or it has noWumpus(X,Y)
    - Tingle. Will assert that the adjacent cells are confundus(X,Y) unless it is visited or it has noConfundus(X,Y)
    - No stench. Will assert that the adjacent cells are noWumpus(X,Y).
    - No Tingle. Will assert that the adjacent cells are noConfundus(X,Y).
- reposition(L)
    - Repositions the agent randomly into a safe spot. Retracts all the information from the agent except the the number of coins it has collected and whether the wumpus is dead.
- visited(X,Y)
    - Asserted whenever the agent visits an unvisited cell. If the cell has been visited, no assertion will be made.
- wumpus(X,Y)
    - Asserted when one of its adjacent cells has stench(X,Y) in the KB.
    - Removed or not asserted when cell is marked safe(X,Y) or noWumpus(X,Y) in KB. Mainly asserted in surveyadj(X,Y).
- confundus(X,Y)
    - Asserted when one of its adjacent cells has tingle(X,Y) in the KB.
    - Removed or not asserted when cell is marked safe(X,Y) or noConfundus(X,Y) in KB. Mainly asserted in surveyadj(X,Y).
- tingle(X,Y)
    - Asserted through setblock(L) when agent moves, when tingle is on.
- glitter(X,Y)
    - Asserted through setblock(L) when agent moves, when glitter is on.
- stench(X,Y)
    - Asserted through setblock(L) when agent moves, when stench is on.
- safe(X,Y)
    - Asserted when a cell has no stench(X,Y) and no tingle(X,Y).
    - Asserted when the cell has both noWumpus(X,Y) and noConfundus(X,Y). Mainly asserted in surveyadj(X,Y).
- wall(X,Y)
    - Asserted through setblock(L) when agent moves, when bump is on. When the agent moves forward to wall(X,Y), the agent will be pushed back to the previous cell.
- explore(L)
    - Takes in a list of actions and checks accordingly as long as the next cell the agent is moving into is safe. Once an unvisited block is detected, the full list of the actions will be returned back to the driver for the driver to give the perceptions for update.
- current(X,Y,D)
    - Constantly updated whenever the agent moves, by asserting the new current(X,Y,D) and retracting the previous current(X,Y,D).

  - hasarrow/0
    - Returns true when arrow(1) is still in KB.

- **Wumpus World Driver**
  - Correctness of Agent’s localisation and mapping abilities
    - Tested with the help of the functions:
    1. printout_localisation_mapping() which executes a series of predefined actions including: moveforward, turnleft, turnright and prints out the relative map generated by the agent after performing each of the actions in order to test the correctness. The absolute map is printed at the beginning of the function to validate the positioning.
    2. printout_glitter_visited_safe() which guides the agent towards a coin, which it then picks up.
    3. printout_wumpus_stench_bump_current() which guides the agent into a wall to test the bump indicator. It also tests the Shoot function, and the presence of the arrow before and after shooting.
  - Correctness of Agent’s sensory inference
    - Tested with the help of the functions: printout_localisation_mapping(), printout_glitter_visited_safe(), printout_wumpus_stench_bump_current(), printout_confundus_tingle(). After each action is executed, the relative map is printed to display the sensory inference of the agent. This can be validated against the initial absolute map printed at the beginning of each of the functions.
  - Correctness of Agent’s memory management in response to stepping through a Confundus Portal.
    - Tested using the function printout_confundus_tingle() which prints out the series of relative maps generated as the agent is guided towards a confundus portal.
  - Correctness of Agent’s exploration capabilities
    - Tested with the function printout_explore() which repeatedly calls the agent’s explore function and executes each of the actions returned by the agent until an empty list is returned.
  - Correctness of the Agent’s end-game reset in a manner similar to that of the Confundus Portal reset.
    - Tested with the function printout_reposition()

Results:
Localisation and mapping:
For localisation and mapping, all the sensory perceptions are taken in and asserted correctly. Subsequently, the inference of these perceptions are also correct. Inference of perceptions refers to what the agent infers with the given senses and knowledge base that it possesses. Results are shown in the test-Printout-Self-Self.txt.

Explore/path planning: Unfortunately due to the complexity of the assignment and the unfamiliarity of prolog with the limitation of the time given for the assignment, a fully working explore function is not available yet. Based on white box testing, we have been able to test specific movements to check if the movement brings the agent to somewhere safe and known based on the knowledge base given by the driver. Such search will stop when the agent detects that the agent is no longer in an unvisited spot or if the list of actions brings the agent to an unsafe and unknown cell.

Related to other drivers:
Driver design matters when it comes to what the driver perceives from the agent and what the agent gets from the driver. There are some variations of design that lead to different results. This may occur due to the difference in certain labeling or index or it could also be due to the difference in the base design, where the perception L in the "move" function might refer to the perception based on the previous cell instead of the current cell. This can lead to different outcomes from the driver.

2. Contribution: For teams with more than one person, please clearly state each team member's contribution.

- Agent by U2022912F is used to test our driver. The name of this text file is 3005_python-testPrintout-Friend-Self.txt.
- Driver by TeamVeryHandsome is used to test our agent. The name of this text file is 3005_python-testPrintout-Self-Friend.txt.

Anagha S. Ani	Driver
Chin Kin Mun	Agent
Lau Chen Yi Wynne	Agent

3. Conclusion: What have you learned?
    - Use “assert” and “retract” to manage facts in the knowledge base.
    - Asserting specific predicates for more than one time will create multiple predicates.
    - Retract all is used to ensure that all of a specific predicate is removed from the knowledge base.
    - List manipulation is mainly used with [H|T], where there can be more than one element in the H. H refers to Head and T refers to Tail.
    - Order of the justification of the predicate determines how the pl file act.
    - There can be prolog predicates of the same name, and prolog will be able to use them accordingly.
        - For example, when “move(moveforward,_)” is used, and there are 4 predicates named “moveforward”. If the agent is currently facing north at cell (0,0), it will move to cell (0,1). However, if the agent is currently facing east at cell (0,0), it will move to cell (1,0) instead.
    - “is” vs “=”.
        - Using “is” will result in the evaluation of an arithmetic expression on the right hand side and is then binded to the left hand side.
        - Using “=” will result in the binding of the unevaluated right hand side to the left hand side
        - For example, X = 2 + 4 will result in X = 2 + 4, while X is 2 + 4 will result in X = 6.

4. References: Please cite the sources you consulted in the report.

- https://github.com/Benjababe
- https://github.com/kckusal/wumpus-world