1 Background: Wumpus World II

Environment: A finite grid world of size $N \times M$, where N,M are integer numbers greater than or equal to 6 (six). Outermost cells contain Walls[1]. A Wall can also appear elsewhere on the grid.

Agent and NPCs: The world is also populated by an Agent, a Wumpus, a set of Confundus Portals, and a set of Coins. While the Agent is mobile, all other items remain fixed throughout the game and cannot co-inhabit a grid cell.

Goal: The goal of the Agent is to find all coins that are accessible and return to the initial position.

Actions: The Agent has *orientation*, so that its movements are: Forward, TurnLeft, and TurnRight. The Agent can also Pickup a gold coin if it is in the same cell as the Agent. The Agent can also shoot an arrow in the direction it faces. The Agent has only one arrow.

Percepts: Prior to any movement, the Agent receives an vector of sensory indicators: Confounded, Stench, Tingle, Glitter, Bump, Scream. All indicators are Off, except in the following situations:

- Confounded indicator is On at the start of the game and after the Agent stepped into a cell with a Confundus
  Portal.
- Stench indicator is On in the cell next to one inhabited by the Wumpus.
- Tingle indicator is On in the cell next to one inhabited by a Confundus Portal.
- Glitter is On in the cell inhabited by a Coin.
- Bump is On if the Agent attempted an action Forward and the next cell in that direction contained a Wall.
- Scream indicator is On after the Agent shot an arrow and the Wumpus was in the direction of the shot.

Notice that the agent does not perceive its exact location on the grid.

Action Effects and Agent Capabilities: All actions taken by the Agent are deterministic in nature. The Agent is capable of counting its steps and turns. Thus, the Agent is capable of orientation in space *relatively* to its initial position and direction. The initial direction is termed Relative North. The direction to the left of Relative North is Relative West, the direction to the right of Relative North is Relative East, and the direction opposite to Relative North is termed Relative South. The initial position of the agent is the Origin with relative coordinates (0,0). Actions have the following effects:

- Executing action Forward changes the position of the Agent. If the agent's orientation is Relative North,the second coordinate of its relative position will increase by 1: $(x,y) \rightarrow (x,y+1)$. Executing Forward in Relative South orientation will decrease the second relative coordinate: $(x,y) \rightarrow (x,y-1)$. Similarly, executing Forward in Relative West (East) orientation will decrease (increase) the first relative coordinate. Exceptions to the rule:
  - The intended new location is inhabited by a Wall. In this case the relative location of the agent is not changed.
  - The intended new location is inhabited by a Confundus Portal. The Agent is randomly relocated to a safe location in the Wumpus World. The relative position is reset to (0,0) and relative orientation is reset to become the Relative North. All memory of previous steps and sensory readings is lost, except the fact of existence of uncollected coins, whether the Wumpus is alive and whether the Agent has the arrow.

[1] Notice that Walls are not between the cells, but *within* the cells completely filling them.

  - The intended new location is inhabited by the Wumpus. Game ends. Relative position reset occurs, as with the case of stepping into a Confundus Portal, to prepare for a new game. All memory of previous steps and sensory reading is lost without exceptions, the arrow is returned to the Agent.

- TurnLeft changes the Agent's current orientation by 90 degrees counter-clockwise. E.g. if the Agent'scurrent orientation is Relative East, then after executing TurnLeft the orientation will become Relative North.
- TurnRight changes the Agent's current orientation by 90 degrees clockwise. E.g., if the Agent's currentorientation is Relative South, then after executing TurnRight the orientation will become Realative West.
- Pickup action removes a Coin, if present, from the current location of the Agent. Percepts change accordingly.
- Shoot action removes the Wumpus from the world, if the Wumpus inhabits any of the cells ahead of theagent.

2 The Assignment

In this assignment you will implement the Agent's logic in a Prolog program. Your Prolog implementation will allow the Agent to analyse the sensory data, navigate and map the current world. You will also implement a basic Wumpus World Driver to test your (and other's) Agent using PySWIP library.

2.1 The Agent

Your Proglog Agent program will implement the following terms:

- reborn/0 – implements Agent's reset due to arriving into a cell inhabited by the Wumpus.
- move(A,L) – implement Agent's reasoning response to executing action A and receiving sensory input L.
  - Action constants are {*shoot,moveforward,turnleft,turnright,pickup*}
  - Sensory input is a list of six indicators, one for each percept: Confounded, Stench, Tingle, Glitter, Bump, Scream. In that order. Each indicator can have one of the two values {*on, off*}.
- reposition(L) – implements Agent's reset due to game start or arrival to a cell inhabited by a ConfundusPortal. The argument includes the initial sensory information. The format of L is the same as with the move/2 function. Confundus indicator should be on.
- Localisation and mapping.
  - visited(X,Y), where X,Y are integers, returns true if the Agent has visited relative position (X,Y). Otherwise it returns false.
  - wumpus(X,Y), where X,Y are integers, returns true if the Agent knows (or reasons) that the Wumpus may inhabit relative position (X,Y). Otherwise it returns false. Notice that there may be more than one positions where true is returned. It is also possible that all positions return false, either because there is no Wumpus or no relevant information exists.
  - confundus(X,Y), where X, Y are integers, returns true if the Agent knows (or reasons) that a Confundus Portal may inhabit relative position (X,Y). Otherwise false is returned.
  - tingle(X,Y), where X, Y are integers, returns true if the Agent knows (or reasons that) it will experience a Tingle indicator beging *on* at the relative position (X,Y). Otherwise, false is returned.
  - glitter(X,Y), where X, Y are integers, returns true if the Agent knows (or reasons that) it will experience Glitter indicator being *on* at the relative position (X,Y). Otherwise false is returned. Notice that this knowledge changes, once the Agent picks up the coin.

  - stench(X,Y), where X,Y are integers, returns true if the Agent knows it will experience Stench indicator being *on* at the relative position (X,Y). Otherwise, false is returned. Notice that this knowledge changes, once the Wumpus is killed.
  - safe(X,Y), where X,Y are integers, returns true if the Agent knows or can reason that the relative position (X,Y) contains neither a Wumpus nor a Confundus Portal.
  - wall(X,Y), where X, Y are integers, returns true if the Agent knows or can reason that the relative position (X,Y) contains a Wall.

- explore(L) is true if the list L contains a sequence of actions that leads the Agent to inhabit a safe andnon-visited location.
  - if there are no more safely accessible un-visited portions of the map left and there are no Coins on the explored portion of the map, the explore(L) should return true on an action sequence that returns the Agent to the relative Origin.
- current(X,Y,D) is true if (X,Y) is the current relative position and D is the current relative orientation of theAgent. Possible relative orientations are described by constants {rnorth,rwest,reast,rsouth}, and the relative position consists of two integers.
- hasarrow/0 returns true if the Agent has the arrow, and begins to return false after *shoot* action has been executed by the Agent.

2.2 Wumpus World Driver

The Wumpus World Driver has to implement a map of size 7×6 with the outer cells inhabited by Walls, and the inner cells populated by at least one Coin, one Agent, one Wumpus and 3 (three) Confundus Portals. The initial position and direction of the Agent can be random, but should be safe. All sensory input generated for the Agent should refer to this map of the Wumpus World.

The Driver is to ask the Agent to reset (reborn/0 call), and then create a feedback loop with the Agent that tests:

- Correctness of Agent's localisation and mapping abilities
  - by asking the Agent to perform a test-sequence of actions
  - using localisation and mapping terms to confirm that the Agent correctly represents its relative position on its relative map.
- Correctness of Agent's sensory inference
  - by asking the Agent to perform a test-sequence of (actions, observations)
  - using localisation and mapping terms to confirm that the Agent correctly absorbed and interpreted its sensory input
- Correctness of Agent's memory management in response to stepping though a Confundus Portal.
  - by feeding the Agent an (action,observation) sequence that creates a non-trivial map within the Agent's knowledge.
  - ask the Agent to reset as if it stepped through a Confundus Portal (reposition/1 call)
  - confirming that the knowledge base of the Agent has been correctly cleaned via the use of localisation and mapping terms.
- Correctness of Agent's exploration capabilities
  - Repeatedly call explore(L), confirm the generated path correctness (safe and ends at a non-visited location or Origin).
  - Feed the exploration plan back to the Agent via a sequence of move(A,L) calls

  - If the newly found cell contains a Coin, order the Agent to pick it up
  - Confirm that the Agent returns *and* stays at the origin once there are no more safely accessible unvisited parts of the map and all discovered coins have been collected.

- Correctness of the Agent's end-game reset in a manner similar to that of Confundus Portal reset.

Hint: Because you can always restart the Prolog, thus hard-resetting the Agent, you can run multiple Wumpus World scenarios in each correctness test. This would make such a test more reliable and challenging. The positioning of NPCs and the Agent does not need to be purely random, thus creating a challenging exploration and planning scenarios. You can also test the system by using empty or partially filled maps, where some NPCs are missing.

2.2.1 Driver Printout:

Correctness (or lack-of) has to be demonstrated by a printout of the following:

- Initial absolute map of the world depicted by an array of map cells. Rows correspond to the second coordinate of the absolute position and the columns to the first coordinate of the absolute position. The map begins at a new line of the printout and finishes by a newline.
- Iteration of the following as many times as necessary
  - Action sequence printout (e.g., "moveforward, turnleft, moveforward, pickup") finished by a newline
  - Sequence of relative positioning maps that depict the complete knowledge of the agent, as the consequence of applying the chosen action sequence. The relative Origin cell is always depicted at the centre of the map. The sequence begins with the map of the initial knowledge before the first actions is applied.
  - Each map (except the initial knowledge map) is preceded by the percepts that were provided to the Agent by the Driver as a sensory input after an action application. If a particular indicator is On, then the full name of the indicator should be printed, otherwise the first letter of the indicator should be printed. Dashes are used as the separators between indicators. E.g. "Confounded-Stench-T-Glitter-BS" is printed, ending with a newline, to denote that Confounded, Stench and Glitter indicators are on, while Tingle, Bump and Scream indicators are off, within the percept provided by the Driver to the Agent.

2.2.2 Map format:

Map cells consist of 9 symbols forming a 3x3 square. Map cells are separated by a single space within a row of the grid map and rows are separated by an new (empty) line. Let's enumerate our symbols 1-9 within a single map cell, like the following. Then:

1	2	3
4	5	6
7	8	9

Symbol 1: Corresponds to the Confounded indicator of the sensory input in this map cell. If it is on, the symbol will printout as "%", otherwise a dot will be printed

Symbol 2: Stench indicator. "=" is printed if the indicator is on, and a dot will be printed otherwise.

Symbol 3: Tingle indicator. "T" is printed if the indicator is on, and a dot otherwise.

Symbol 4: Prints "−" if the cell contains the Agent or an NPC, otherwise prints as a single space.

Symbol 5:

- Prints ”W” if the map cell is known or reasoned to (possibly) contain the Wumpus
- Prints ”O” if the map cell is known or reasoned to (possibly) contain a Confundus Portal
- Prints ”U” if the map cell is both: a) *reasoned to possibly* contain a Confundus Portal and b) *reasoned to possibly* contain the Wumpus. Notice that no cell can *factually* contain both of those NPCs.
- Prints ”∧”,”<”,”>” or ”∨” to denote that the Agent inhabits the cell and faces, correspondingly, (r)north, (r)west, (r)east or (r)south. The meaning of the direction is according to the map that is being printed. Absolute in the actual Wumpus World map, and relative in the Agent’s localisation map.
- Prints small ”s” for a non-visited safe map cell without the Agent inhabiting it
- Prints capital ”S” for a visited safe map cell without the Agent inhabiting it – If none of the above, a single question mark, ”?”, is printed.

Symbol 6: Prints ”−” if the cell contains the Agent or an NPC, otherwise prints as a single space.

Symbol 7: Glitter indicator. ”*” (star) is printed if the indicator is on and a dot otherwise.

Symbol 8: Bump indicator. ”B” is printed if the indicator is on and a dot otherwise.
Caution: Bump indicator is transitory. That is, it will appear only if the agent tried to go forward and met a Wall.

Symbol 9: Scream indicator. ”@” is printed if the indicator is on and a dot otherwise.
Caution: Scream indicator is transitory. That is, it will appear only if the agent shot its Arrow and the Wumpus was killed/removed.

Exclusion: If a map cell is inhabited by a (known) Wall, then all symbol are #

3 Marking Criteria

3.1 Agent capabilities (72 points)

Three different Driver implementations will be used to generate (and confirm) reports. Self-Driver, a ”friend” Driver and a School chosen Driver.

- Correctness of Agent’s localisation and mapping abilities.12 points (4 per Driver that was run on the Agent).
- Correctness of Agent’s sensory inference. 24 points (8 per Driver that was run on the Agent).
- Correctness of Agent’s memory management in response to stepping though a Confundus Portal.9 points (3 per Driver that was run on the Agent).
- Correctness of Agent’s exploration capabilities
  - Finding a safe path to a safe unexplored location. 12 points (4 per Driver that was run on the Agent)
  - Confirmation of a return (and stay) path solution. 9 points (3 per Driver that was run on the Agent)
- Correctness of the end-game reset. 6 points (2 per Driver that was run on the Agent).

3.2 Driver capabilities (28 points)

- Differentiation between the Absolute and Relative Map printouts. 3 points
- Ability to test viable Agents. 5 points per Agent test (2 points for self-Agent, 2 points for ”friend” Agent”, 1 point for a School chosen Agent), totalling at 25 points for all tests.

4 Team Formation and Inter-Team Cooperation

You are required to form a team of up to 3 people to complete this project. A team with only a single person is also allowed. For team size that is larger than one, you need to clearly state the contribution of each team member in your solution report.

For the purposes of completing this Assignment you will need to exchange your Driver code with other teams. However, this is done for test-run purposes only. Do not use someone else code as a clue/hint/inspiration – you never konw where it will lead you.

You cannot exchange Agent code. If the ”A-team” needs a printout of their Driver running on the Agent of the team ”FBI”, then the ”A-team” should send their A-Team-Driver to the team ”FBI”. The team ”FBI” will then run the ”A-team-Driver” Driver on the ”FBI-Agent” and revert with the outcome.

5 Honour Code

If you use any existing code, libraries, etc. and consult any papers, books, online references, etc. for your project, you must cite your sources in your report and clearly indicate which parts of the project are your contributions and which parts were implemented by others. Using others' code/ideas without acknowledgement will lead to failure of your project. You should not even attempt to submit another team's Driver code (partially or in its entirety) as your own. Plagiarism testing tools are readily available and ruthless. You cannot exchange Agent code.

6 Deliverables

You are required to produce:

1. A final report of no more than 5 pages. Name your report as <team name>-final-report.pdf.
2. Source codes for your project. Your source codes are to be submitted in .zip format. Name the zip file as<team name>-final-code.zip.
    - The Agent code should be named <team name>-Agent.pl
    - The Driver code should be named <team name>-Driver.py
    - Friend Driver code should be named <team name>-Friend-Driver.py
    - Printout of Driver reports <team name>-testPrintout-<agent source>-<driver source>.txt
        – source is either ”Self” or ”Friend”. Thus, team ”Frodo” running their own Driver on a friendly Agent should be named ”Frodo-testPrintout-Friend-Self.txt”.