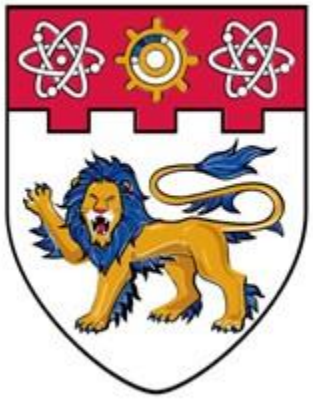

NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

CZ3005 Artificial Intelligence

LAB 2 Wumpus World (SSP6)
Group members:

Name	Matriculation Number	Contribution
Andre Lim	U2020397A	Agent and Driver Code, Report explanations
Goh Zheng Yang	U2020170H	All of Report

1. Approach

1.1. Overview

Two files were mainly used for our project, a driver and an agent, both of which are written in Python and Prolog respectively but connected via the PYSWIP library. The driver file implements the Wumpus World map's details and rules while the agent file implements the agent's intelligent decision logic and exploration capabilities.

1.2. Driver.py

Driver.py is the main file used to feedback what is going on in the game (map state, action choice, game state such as number of arrows, coins collected, sensory information, etc.)

- Wumpus and Confundus Portal are denoted by W and P respectively
- Player/agent is denoted by an arrow on the map, depending whether they are facing north,south, east or west (n,s,e,w).

- The player's orientation (n,s,e,w) is determined by the absDir variable, which will cause theplayer's 'avatar' printed on the map to be (^, v, >, <) respectively.

Wumpus, coin and portal positions in the map are fixed while player spawn point/orientation are random (but always in a safe spot). When the game starts, the driver calls the reborn() function which resets all states to default. It also uses the random_spawn() and random_direction() functions to randomise the player's spawn point and orientation respectively.

Maps are printed, entities/objects are spawned and the player can now perform actions by calling the controls() function.

The player can perform one of 6 actions, 5 to pass an action/sensory info after the action, and the 6th to let the agent self-explore. They result in a function where the driver performs a **Prolog query (move(A,L) / explore(L))**, being called, resulting in the game entering a different state:

1) Move Forward, 2) Turn Left, 3) Turn Right, 4) Pick Up Coin, 5) Shoot, 6) Self Explore

The default knowledge base at the start of the game is that there are entities/objects in the map cells in the form of Wumpus, Confundus Portals and Coins, and unpopulated cells are considered to be safe. The agent's knowledge base is expanded/updated as they traverse the world, and all knowledge is stored in a list data structure, which is updated throughout the exploration of the game map.

The user is able to use the driver in a game mode (manually input their choice of actions) or auto mode (let the agent decide its own action).

Every action the agent takes will result in a different game state, which is reflected by the map. There are two maps in the game:

- Relative Map
    - The relative map starts as a 3x3 grid, with the origin always at the center. It expands as the agent begins exploration.

- Absolute Map
    - An absolute map showing all the cells in the game is required to ensure the agent's relative map is working correctly.

1.3. Agent.pl

Default Gameplay

- reborn/0 retracts all information, asserts relative_position, numcoins, relative_direction, etc.to default settings at the start of a game.
- reposition(L) retracts only positional information but not others such as numcoins, arrows, wumpus_dead etc.
- percept uses asserts to update the agent's Knowledge Base (possible portal and wumpus locations to avoid, etc.) using knowledge of perceived sensations.
- move(A,L) takes in an action resulting in a change of state in the game

The action can be chosen by the user or by the agent's preprogrammed move suggestions. Actions taken are passed into the agent.pl file by the driver into the move(A, L) function. Depending on the action (A) passed, the outcome can be different.

Self-Exploring Agent (Action #6)

This is mainly done using the explore(L) function. The explore(L) function is an algorithm that ensures the agent always makes the most logical move using the game state and its knowledge base. It uses depth-first search (DFS) for path traversal, where the starting node is the current cell while goal nodes are safe unvisited cells, cells with coins (or origin cell after a full exploration). In Prolog, necessary actions required to travel within nodes in the path derived by DFS are appended and simulated. These actions are then returned to the Driver.

While explore(L) always picks up from cells with coins, it always shoots **as a last resort** when the agent has no other moves to make and is facing a potential wumpus cell

There is also a possibility of explore(L) being unable to come to a decision, such as when the agent is surrounded completely by possible Confundus Portals/Wumpu locations (e.g. due to a bad spawn or multiple portals/wumpus arranged in a way such that all cells adjacent to the agent are unsafe. When this happens, an empty list is returned to the driver which **triggers a need for the user to manually input the next move**.

Agent's Survival Deductions

While the agent can make rational decisions for most cases, it can be tricky for them to **deduce the exact cells containing the Wumpus or Confundus Portals (the only obstacles in the game)** due to the sensations being emitted in multiple adjacent cells. To do this, the agent makes use of Tingle and Stench sensations in several cells. The agent also further utilises this knowledge to identify the safe cells. The Agent uses the ***'assert'*** and ***'retract'*** commands in Prolog to update its knowledge base. Other rules are used as well (please refer to code).

Sensation	**Agent's Knowledge Base**	**Course of Action**
Tingle → Portal	As there can be >1 Confundus Portal, all cells adjacent to the agent could possibly contain a Confundus Portal, with the exception of the cell they previously moved forward from. (Unless agent spawned between multiple portals)	Store the cells adjacent to the agent that could possibly contain a portal in the agent's knowledge base and avoid stepping into them.
Stench → Wumpus	Upon first detection of stench, the agent will conclude that all adjacent cells could contain a Wumpus (apart from the cell it just came from). All subsequent stench detection will not result in new Wumpus locations being placed on the map because the game only has **1** Wumpus. Therefore, the agent can conclude which cells have the highest likelihood of containing the actual wumpus and remove previously concluded Wumpus cells from its knowledge base.	Store the cells adjacent to the agent that could possibly contain a Wumpus in the agent's knowledge base. If the agent still has the arrow and has no other moves to make, shoot. If a scream was heard, refer to the next row in this table. Otherwise, remove the cell the agent shot the arrow at from the list of possible Wumpus locations in the agent's knowledge base.

Safe	A cell is safe if either of the following holds true: 1. It is not populated by any entities or objects 2. It has been visited before 3. Arrow has been shot in that cell and it is not a potential confundus portal location 4. A cell that was previously thought to be occupied by the Wumpus **only** will be deemed safe once a Scream is heard. 5. Cells that the agent has already concluded are not Wumpus or Portal locations.	**When the agent is at a safe cell and does not experience a tingle or stench, it knows that all adjacent cells cannot contain the Confundus Portal or Wumpus. This allows the agent to deduce the exact location of the entities by traveling to as many safe cells as possible.**

2. Output

3 printout files are used(*BestOfSSP6-testPrintout-Self-Self, BestOfSSP6-testPrintout-Self-Self, and BestOfSSP6-testPrintout-Friend-Self) to illustrate accuracy of Agent.pl and Driver.py.*

2.1. Game Start

Note - All 3 files use the same format, so even though 'BestOfSSP6-testPrintout-Self-Self' is stated, the same method can be applied to the other files.

World Map is printed and shown immediately at the start of the game.	*BestOfSSP6-testPrintout-Self-Self (CTRL+F -> _worldmap_)*
Initial Absolute Map	*BestOfSSP6-testPrintout-Self-Self (CTRL+F -> _absolutemap_)*
Initial Relative Map	*BestOfSSP6-testPrintout-Self-Self (CTRL+F -> _relativemap_)*
Initial Sensory Information	*BestOfSSP6-testPrintout-Self-Self (CTRL+F -> _initialsenses_)*
Controls	*BestOfSSP6-testPrintout-Self-Self (CTRL+F -> _controls_)*

2.2. Test Cases

Move Forward (Has before and after)	*BestOfSSP6-testPrintout-Self-Self (CTRL+F -> _moveforward_)* Agent position changed after
Turn Left (Has before and after)	*BestOfSSP6-testPrintout-Self-Self (CTRL+F -> _turnleft_)* Agent orientation moved relative left
Turn Right (Has before and after)	*BestOfSSP6-testPrintout-Self-Self (CTRL+F -> _turnright_)* Agent orientation moved relative right

Tingle	*BestOfSSP6-testPrintout-Self-Self (CTRL+F -> _tingle_)* 'T' indicated in the corresponding cell. Adjacent cells indicate 'O'
Stench (First Detection)	*BestOfSSP6-testPrintout-Self-Self (CTRL+F -> _stench1_)* '=' indicated in the corresponding cell. Adjacent cells are 'W'.
Subsequent Stench Detection *Note - Even though a stench is detected, adjacent cells are not updated to contain Wumpus as previously, the Agent has already concluded the potential Wumpus cells. Additionally, some of the previous Wumpus cells have been removed and deemed safe because based on this new stench location, those older cells cannot possibly contain the Wumpus.*	*BestOfSSP6-testPrintout-Self-Self (CTRL+F -> _stench2_)* Only cells previously 'W' will be re-evaluated to confirm Wumpus' location.
Stench and Tingle	*BestOfSSP6-testPrintout-Self-Self (CTRL+F -> _stench_tingle_)* 'U' indicated in adjacent cells. 'T' and '=' indicated in agent's cell
Glitter	*BestOfSSP6-testPrintout-Self-Self (CTRL+F -> _glitter_)* '*' indicated in corresponding cell
Bump	*BestOfSSP6-testPrintout-Self-Self (CTRL+F -> _bump_)* 'B' indicated in the corresponding cell.
Pick up coin (Successful) (Has before and after) ***Note - Glitter gone***	*BestOfSSP6-testPrintout-Self-Self (CTRL+F -> _pickup_success_)*
Shoot (Missed Wumpus) (Has before and after)	*BestOfSSP6-testPrintout-Self-Self (CTRL+F -> _shoot_miss_)*
Note - No scream	
Shoot (Slain Wumpus) (Has before and after) *Note - Previous wumpus cells removed*	*BestOfSSP6-testPrintout-Self-Self (CTRL+F -> _shoot_hit_)* '@' indicated in agent's cell

Killed by Wumpus	*BestOfSSP6-testPrintout-Self-Self (CTRL+F -> _killed_)*
Confounded (Agent entered portal) (Has before and after) *Note - Agent spawned in a random spot, confounded, and senses tingle and stench. However, it only assumes surrounding cells to be portal because it has already slain the Wumpus as shown previously.*	*BestOfSSP6-testPrintout-Self-Self (CTRL+F -> _enter_portal_)* '%' indicated in the agent's cell only on spawn.

2.3. Agent Self-Exploration

Refer to https://youtu.be/umlp6RaMv_4 for a video demonstration of this

Do note that Agent returns to the origin cell after exploration is completed!

2.4. Agent in Dilemma (All moves unsafe)

When the agent is in a situation where all possible moves are unsafe (e.g. surrounded by all 4 adjacent cells being potential wumpus/portal locations, explore(L) returns an empty list resulting in the driver needing the user to manually input their next move.

Refer to submission file BestOfSSP6-testPrintout-Self-Self (CTRL+F -> _dilemma_)

2.5. Friend's Driver

Refer to https://youtu.be/qCZqMSi3Sjc for a video demonstration of this.

For the printout, do refer to submission file 'BestOfSSP6-testPrintout-Self-Friend'

2.6. Friend's Agent on Self Driver

Refer to submission file 'BestOfSSP6-testPrintout-Friend-Self'. Note - Friend's Agent's explore function does not work as it implements additional actions not stated in the lab manual. All other Driver functions are functioning.

3. References

DFS - https://stackoverflow.com/questions/27065774/depth-first-search-algorithm-prolog