Project 2: Logical Agent for the Wumpus Game

Nour Eddine Amraoui <81051>

Dr. Tajjeedine Rachidi

Teammate: Mohammed Khalil Ghali <76368>

Intro. to Artificial Intelligence

March 21st, 2022

I. The project objective

The aim of this project is to implement a logical agent to paly the Wumpus game. The game is a simple world example to illustrate the worth of a knowledge-based agent and to represent knowledge representation. The game is represented with a cave that has 16 rooms that are connected with each other. One room within the cave had the Wumpus, who is a beast that kills everyone. The goal of the game is to kill the beast, and this can be done using an only arrow that the player has. Withing the journey of the agent during the cave, there are some pits that are bottomless rooms in which one will be stuck there. There is also another chamber that contains gold, which shall increase the agent’s life.

II. Key Predicates, and the meaning of the Variables

➢ <u>Key Predicates:</u>

- `map_size`: The size of the cave, which is going to be 4x4.
- `room`: Room as (A, [X, Y]) meaning that the location of A is [X, Y]
- `Wumpus`: Room that has the Wumpus (detected using the stench).
- `noPit`: noPit([X, Y]) means that the agent is sure that there is no pit in the room [X, Y], it is known because there is no breeze on adjacent cells.
- `noWumpus`: noWumpus([X, Y])]) means that the agent is sure that there is no Wumpus in the room [X, Y], it is known because there is no breeze on adjacent cells.
- `maybeVisitLater`: When no adjacent cell is safe, this adds the current cell as a probable room to visit later and backtrack.
- `wumpusPath`: The agent stores each room from the starting pint which is [1, 1] to the Wumpus.
- `init_cave`: initialize the state of the cave, which is 4x4 map and the positions of Gold and Pits.

- init_agent: initialize the room where the agent is (the room [1,1]).
- init_wumpus: initialize the room of the Wumpus.
- explore: A predicate that moves one cell to another and it keeps track of the rooms that were visited to reach a certain position.
- adjacent: adjacent([X, Y], Z)is predicated that determines if a room with coordinated [X,Y] is adjacent to a room with coordinated Z= [A,B].
- stenchy: if true, means that there is a Wumpus in an adjacent room.
- breezy: if true, means that there is a Pit in an adjacent room.
- glittery: if true, means that there is Gold in an adjacent room.
- shootWumpus: kill the Wumpus if it exists in an adjacent room
- printP: print the final status of the game.
- printStatus: print the stages when playing, and also it prints the leading path.

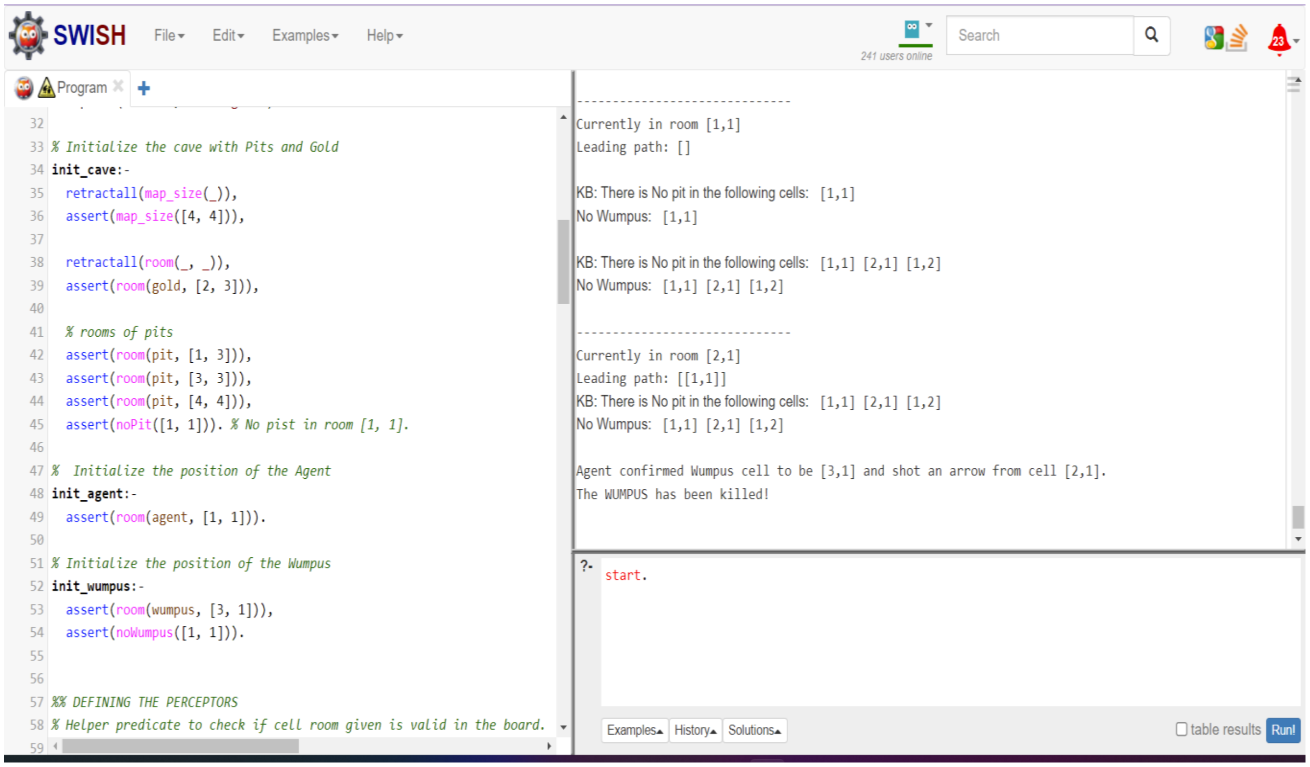
Variables Meaning:

- X: Within the coordinates of a room, it refers to the horizontal position withing the cave/map.
- Y: Within the coordinates of a room, it refers to the vertical position withing the cave/map.
- OldCell: A room that is visited and it was not the best one to go to.
- LeadingPath: List of visited rooms.
- When defining the "adjacent" predicate:
  - L: room left to the one on question.
  - R: room at the right to the one on question.
  - A: room above the one on question.
  - B: room below the one on question.

III. Snapshots of the experiments

A. The first configuration:

   a. We have 3 pits, they are in [1, 3], [3, 3], and [4, 4].
   b. The Wumpus is in [3, 1].
   c. The Gold is in [2, 3].



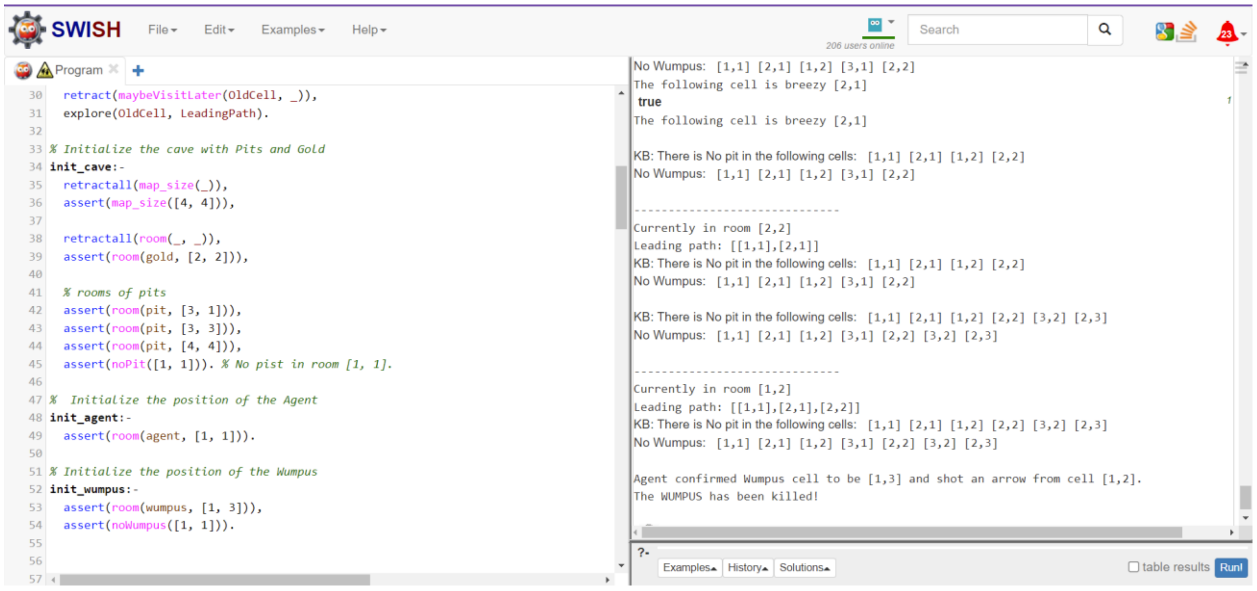
```prolog

% Initialize the cave with Pits and Gold
init_cave:-
  retractall(map_size(_)),
  assert(map_size([4, 4])),

  retractall(room(_, _)),
  assert(room(gold, [2, 3])),

  % rooms of pits
  assert(room(pit, [1, 3])),
  assert(room(pit, [3, 3])),
  assert(room(pit, [4, 4])),
  assert(noPit([1, 1])). % No pist in room [1, 1].

%  Initialize the position of the Agent
init_agent:-
  assert(room(agent, [1, 1])).

% Initialize the position of the Wumpus
init_wumpus:-
  assert(room(wumpus, [3, 1])),
  assert(noWumpus([1, 1])).


%% DEFINING THE PERCEPTORS
% Helper predicate to check if cell room given is valid in the board.
```

```
-----------------------------
Currently in room [1,1]
Leading path: []

KB: There is No pit in the following cells:  [1,1]
No Wumpus:  [1,1]

KB: There is No pit in the following cells:  [1,1] [2,1] [1,2]
No Wumpus:  [1,1] [2,1] [1,2]

-----------------------------
Currently in room [2,1]
Leading path: [[1,1]]
KB: There is No pit in the following cells:  [1,1] [2,1] [1,2]
No Wumpus:  [1,1] [2,1] [1,2]

Agent confirmed Wumpus cell to be [3,1] and shot an arrow from cell [2,1].
The WUMPUS has been killed!
```

```
?- start.
```

B. The second configuration:

   a. We have 3 pits, they are in [3, 1], [3, 3], and [4, 4].
   b. The Wumpus is in [1, 3].
   c. The Gold is in [2, 2].

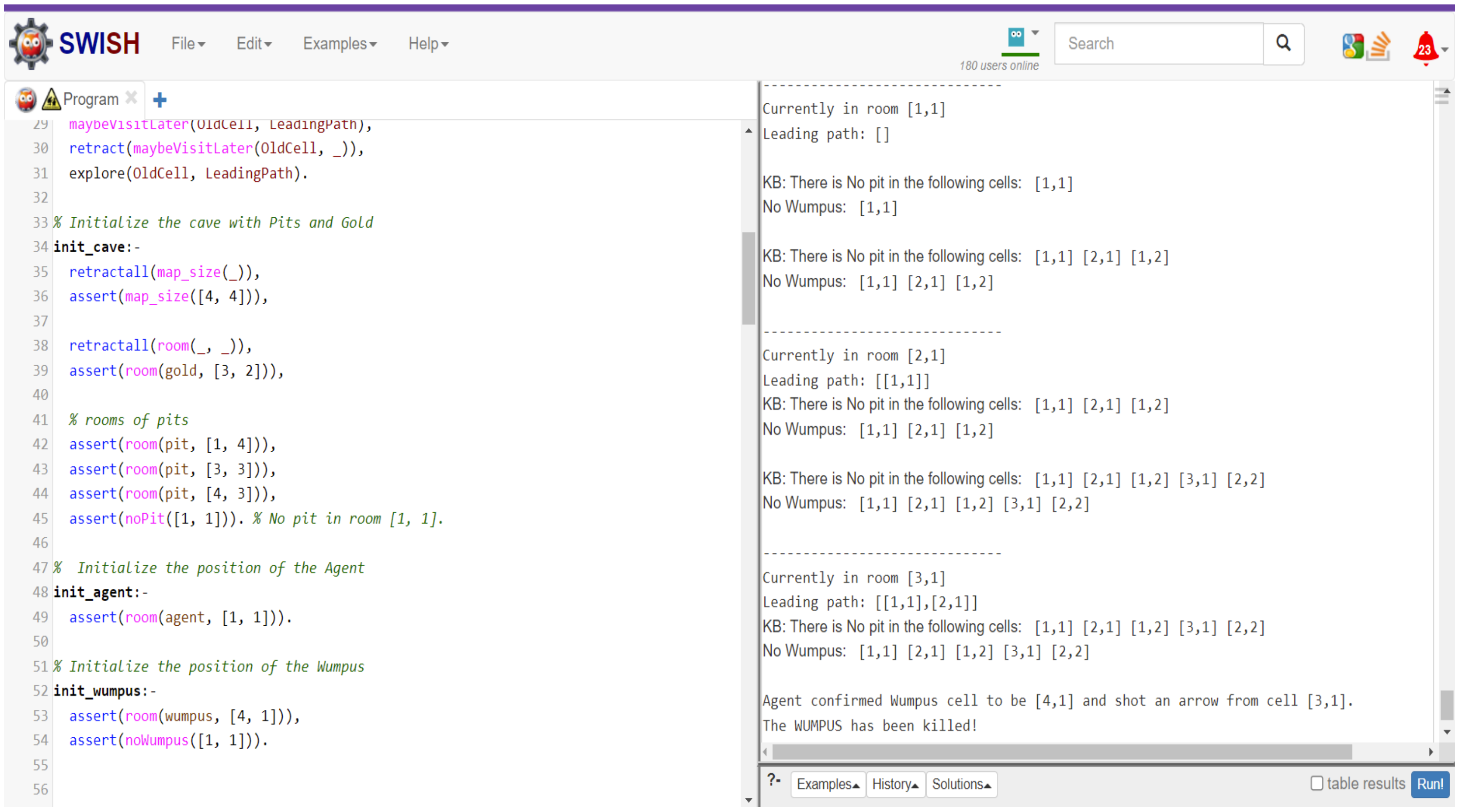


```
  retract(maybeVisitLater(OldCell, _)),
  explore(OldCell, LeadingPath).

% Initialize the cave with Pits and Gold
init_cave:-
  retractall(map_size(_)),
  assert(map_size([4, 4])),

  retractall(room(_, _)),
  assert(room(gold, [2, 2])),

  % rooms of pits
  assert(room(pit, [3, 1])),
  assert(room(pit, [3, 3])),
  assert(room(pit, [4, 4])),
  assert(noPit([1, 1])). % No pist in room [1, 1].

%  Initialize the position of the Agent
init_agent:-
  assert(room(agent, [1, 1])).

% Initialize the position of the Wumpus
init_wumpus:-
  assert(room(wumpus, [1, 3])),
  assert(noWumpus([1, 1])).
```

```
No Wumpus:  [1,1] [2,1] [1,2] [3,1] [2,2]
The following cell is breezy [2,1]
 true
The following cell is breezy [2,1]

KB: There is No pit in the following cells:  [1,1] [2,1] [1,2] [2,2]
No Wumpus:  [1,1] [2,1] [1,2] [3,1] [2,2]

-----------------------------
Currently in room [2,2]
Leading path: [[1,1],[2,1]]
KB: There is No pit in the following cells:  [1,1] [2,1] [1,2] [2,2]
No Wumpus:  [1,1] [2,1] [1,2] [3,1] [2,2]

KB: There is No pit in the following cells:  [1,1] [2,1] [1,2] [2,2] [3,2] [2,3]
No Wumpus:  [1,1] [2,1] [1,2] [3,1] [2,2] [3,2] [2,3]

-----------------------------
Currently in room [1,2]
Leading path: [[1,1],[2,1],[2,2]]
KB: There is No pit in the following cells:  [1,1] [2,1] [1,2] [2,2] [3,2] [2,3]
No Wumpus:  [1,1] [2,1] [1,2] [3,1] [2,2] [3,2] [2,3]

Agent confirmed Wumpus cell to be [1,3] and shot an arrow from cell [1,2].
The WUMPUS has been killed!
```

```
?-
```

C. The third configuration:
   a. We have 3 pits, they are in [1, 4], [3, 3], and [4, 3].
   b. The Wumpus is in [4, 1].
   c. The Gold is in [3, 2].



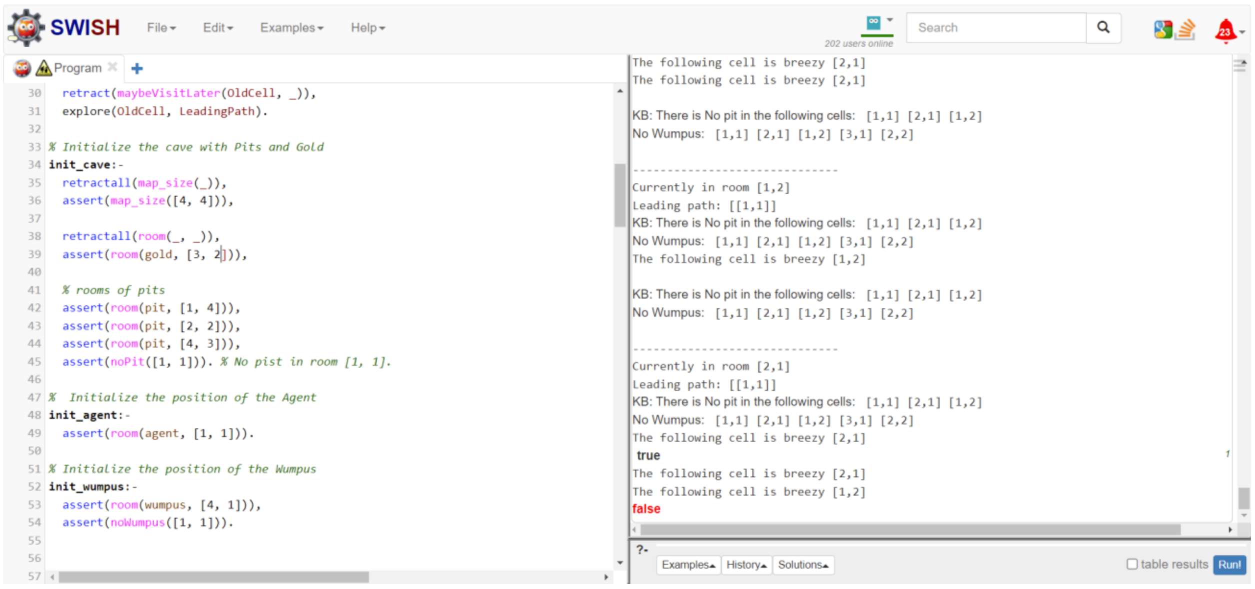
```
  maybeVisitLater(OldCell, LeadingPath),
  retract(maybeVisitLater(OldCell, _)),
  explore(OldCell, LeadingPath).

% Initialize the cave with Pits and Gold
init_cave:-
  retractall(map_size(_)),
  assert(map_size([4, 4])),

  retractall(room(_, _)),
  assert(room(gold, [3, 2])),

  % rooms of pits
  assert(room(pit, [1, 4])),
  assert(room(pit, [3, 3])),
  assert(room(pit, [4, 3])),
  assert(noPit([1, 1])). % No pit in room [1, 1].

%  Initialize the position of the Agent
init_agent:-
  assert(room(agent, [1, 1])).

% Initialize the position of the Wumpus
init_wumpus:-
  assert(room(wumpus, [4, 1])),
  assert(noWumpus([1, 1])).
```

```
------------------------------
Currently in room [1,1]
Leading path: []

KB: There is No pit in the following cells:  [1,1]
No Wumpus:  [1,1]

KB: There is No pit in the following cells:  [1,1] [2,1] [1,2]
No Wumpus:  [1,1] [2,1] [1,2]

------------------------------
Currently in room [2,1]
Leading path: [[1,1]]
KB: There is No pit in the following cells:  [1,1] [2,1] [1,2]
No Wumpus:  [1,1] [2,1] [1,2]

KB: There is No pit in the following cells:  [1,1] [2,1] [1,2] [3,1] [2,2]
No Wumpus:  [1,1] [2,1] [1,2] [3,1] [2,2]

------------------------------
Currently in room [3,1]
Leading path: [[1,1],[2,1]]
KB: There is No pit in the following cells:  [1,1] [2,1] [1,2] [3,1] [2,2]
No Wumpus:  [1,1] [2,1] [1,2] [3,1] [2,2]

Agent confirmed Wumpus cell to be [4,1] and shot an arrow from cell [3,1].
The WUMPUS has been killed!
```

IV. Limitations

A. The fourth configuration: Fails

d. We have 3 pits, they are in [1, 4], [2, 2], and [4, 3].
e. The Wumpus is in [4, 1].
f. The Gold is in [3, 2].



```
  retract(maybeVisitLater(OldCell, _)),
  explore(OldCell, LeadingPath).

% Initialize the cave with Pits and Gold
init_cave:-
  retractall(map_size(_)),
  assert(map_size([4, 4])),

  retractall(room(_, _)),
  assert(room(gold, [3, 2])),

  % rooms of pits
  assert(room(pit, [1, 4])),
  assert(room(pit, [2, 2])),
  assert(room(pit, [4, 3])),
  assert(noPit([1, 1])). % No pist in room [1, 1].

%  Initialize the position of the Agent
init_agent:-
  assert(room(agent, [1, 1])).

% Initialize the position of the Wumpus
init_wumpus:-
  assert(room(wumpus, [4, 1])),
  assert(noWumpus([1, 1])).
```

```
The following cell is breezy [2,1]
The following cell is breezy [2,1]

KB: There is No pit in the following cells:  [1,1] [2,1] [1,2]
No Wumpus:  [1,1] [2,1] [1,2] [3,1] [2,2]

------------------------------
Currently in room [1,2]
Leading path: [[1,1]]
KB: There is No pit in the following cells:  [1,1] [2,1] [1,2]
No Wumpus:  [1,1] [2,1] [1,2] [3,1] [2,2]
The following cell is breezy [1,2]

KB: There is No pit in the following cells:  [1,1] [2,1] [1,2]
No Wumpus:  [1,1] [2,1] [1,2] [3,1] [2,2]

------------------------------
Currently in room [2,1]
Leading path: [[1,1]]
KB: There is No pit in the following cells:  [1,1] [2,1] [1,2]
No Wumpus:  [1,1] [2,1] [1,2] [3,1] [2,2]
The following cell is breezy [2,1]
 true
The following cell is breezy [2,1]
The following cell is breezy [1,2]
false
```

At this configuration the agent is stuck at the room [2,1] because it perceives breeze there and in [1,2] and it does not conclude if the Pit exists in [2,2] or in [3,1].

Given the specific time to work on this project and the fact that personally this was my first interaction with Prolog, I believe that this code can be improved more and more. The first thing has to do with the configurations. The configurations can be randomly generated instead of having them hardcoded. Furthermore, our agent gets stuck when it is in a situation that is inconclusive, and it does not go further. Calculating and displaying the score could be added as well, hence, even if it is a minor thing to be implemented, we decided to go further and invest the time given in working in the logic of the game.