جامعة الأخوين

ⵜⴰⵙⴷⴰⵡⵉⵜ ⵏ ⵍⴰⵅⴰⵡⴰⵢⵏ

AL AKHAWAYN

UNIVERSITY

Assignment 2: logical agent for the Wumpus game.

Mohammed Khalil Ghali: 76368
Noureddine Amraoui : 81051

Dr. Tajjeeddine Rachidi

Link to my GitHub Repository:
https://github.com/khalil-ghali/WumpusWorld

Project Objective:

The Wumpus world is a cave with *16* rooms *(4×4)*. Each room is connected to others through walkways (no rooms are connected diagonally). The knowledge-based agent starts from *Room[1, 1]*. The cave has – some **pits**, a **treasure** and a beast named **Wumpus**. The Wumpus can not move but eats the one who enters its room. If the agent enters the pit, it gets stuck there. The goal of the agent is to take the treasure and come out of the cave. The agent is rewarded, when the goal conditions are met. The agent is penalized, when it falls into a pit or being eaten by the Wumpus.

Some elements support the agent to explore the cave, like -The wumpus's adjacent rooms are stenchy. -The agent is given one arrow which it can use to kill the wumpus when facing it (Wumpus screams when it is killed). – The adjacent rooms of the room with pits are filled with breeze. -The treasure room is always glittery.

## I- Key Predicates and the meaning of variables:

- Key predicates

  map_size: Which implies the size of the world
  room: it refers to the location of a certain room by x,y coordinates
  wumpus: possible Wumpus location
  noPit: the agent is sure there is no pit at location x,y
  noWumpus: the agent is sure there is no wumpus at location x,y
  maybeVisitLater: **if no** adjacent cell to go to, add the current cell as a probable point to visit after
  init_Wumpus: initializes the position of Wumpus in board
  init_agent: initializes the agent in the board to [1,1]
  init_cave: initializes the position of pits and gold
  wumpusPath: gives us the path to the Wumpus room
  stenchy
  breezy
  shootWumpus
  Explore: refers to moving to an adjacent cell to explore and keeps track of visited cells.
  printR: it prints the final status of the game
  printStatus: prints the continuing flow of the game
  Adjacent: generate adjacent rooms of a given room
  ValidRoom: checks if the board room is valid or not

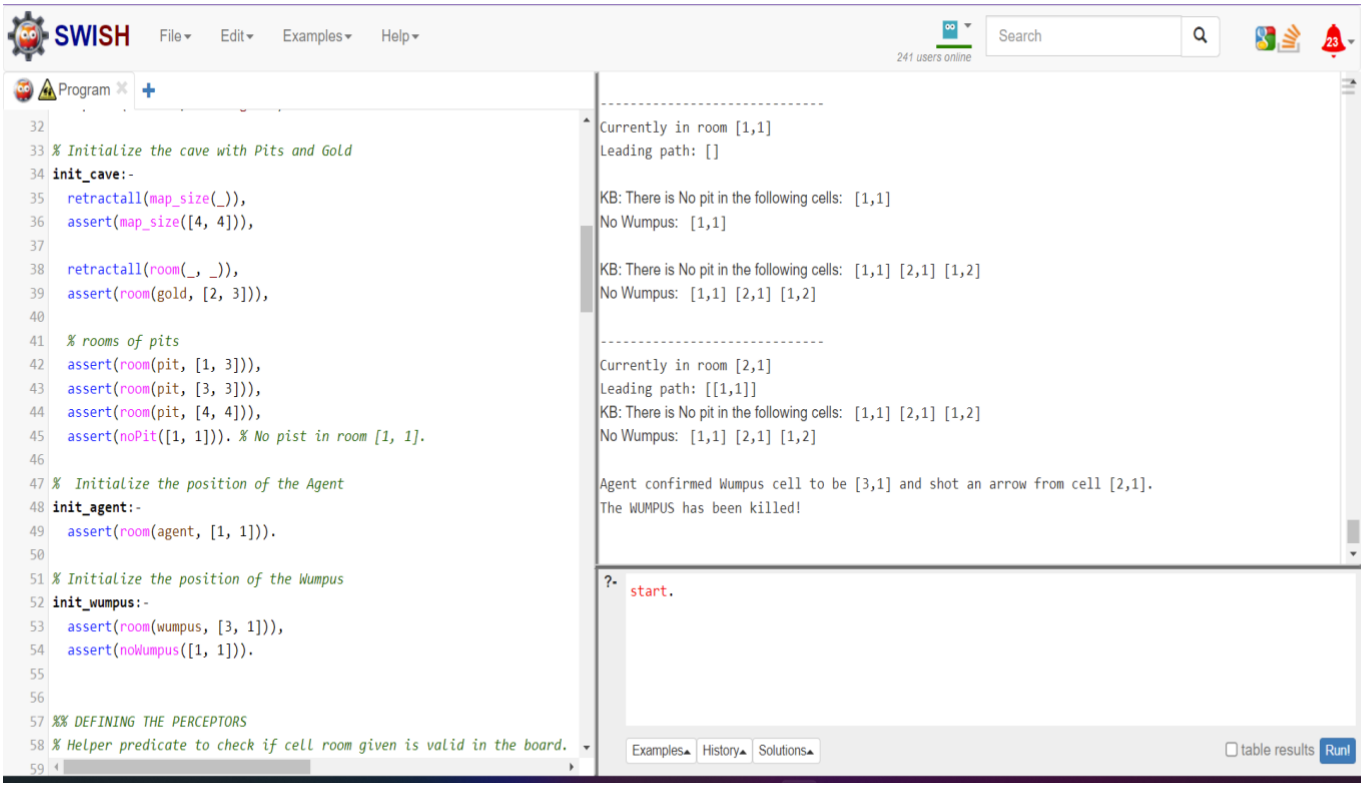
- Variables and their meaning

  X: the abscissa of the map
  Y: the ordinate of the map
  OldCell: an already visited cell
  Leading Path:
  L: refers to the left of the current cell
  R: refers to the right of the current cell
  A: refers to above the current cell
  B: refers to below the current cell
  Cell: the current cell.

II- Snapshots of our Experiments

First configuration where pits locations are as shown in the screenshot

And the gold is in (2,3), the Wumpus in (3,1)

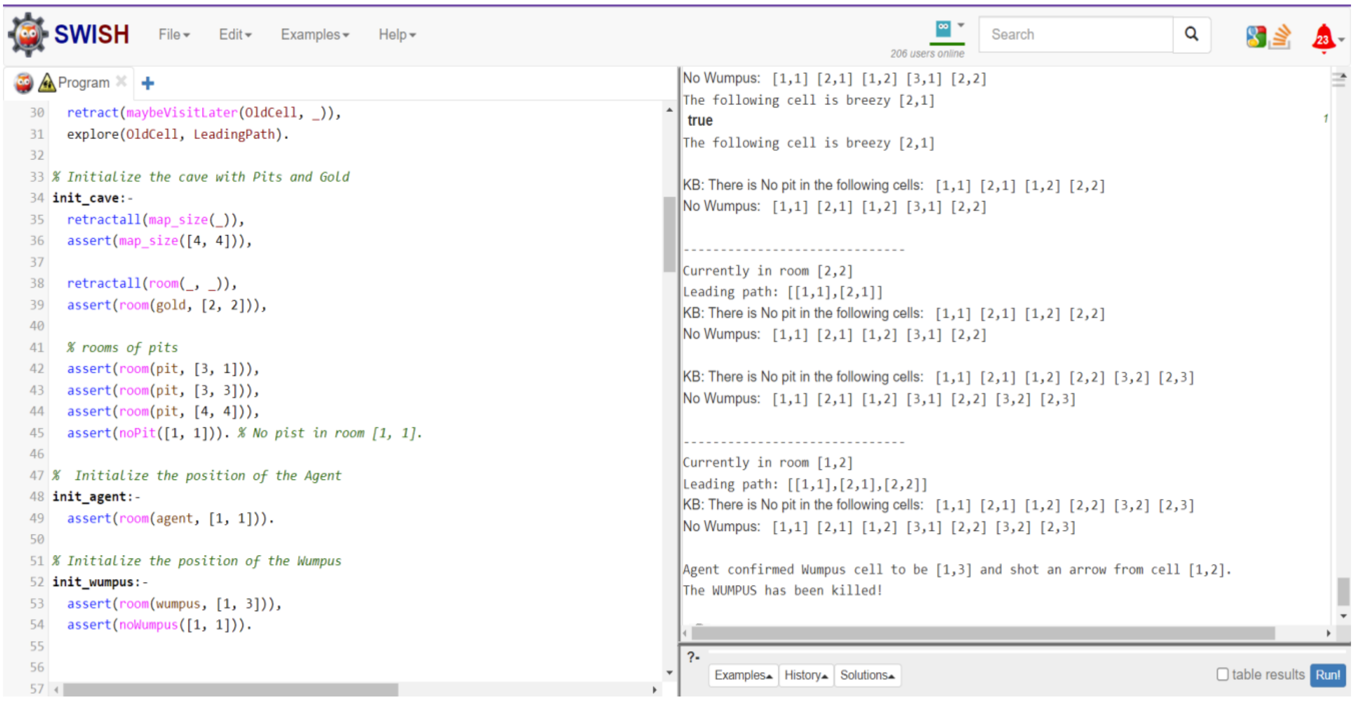


```prolog
% Initialize the cave with Pits and Gold
init_cave:-
    retractall(map_size(_)),
    assert(map_size([4, 4])),

    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.
```

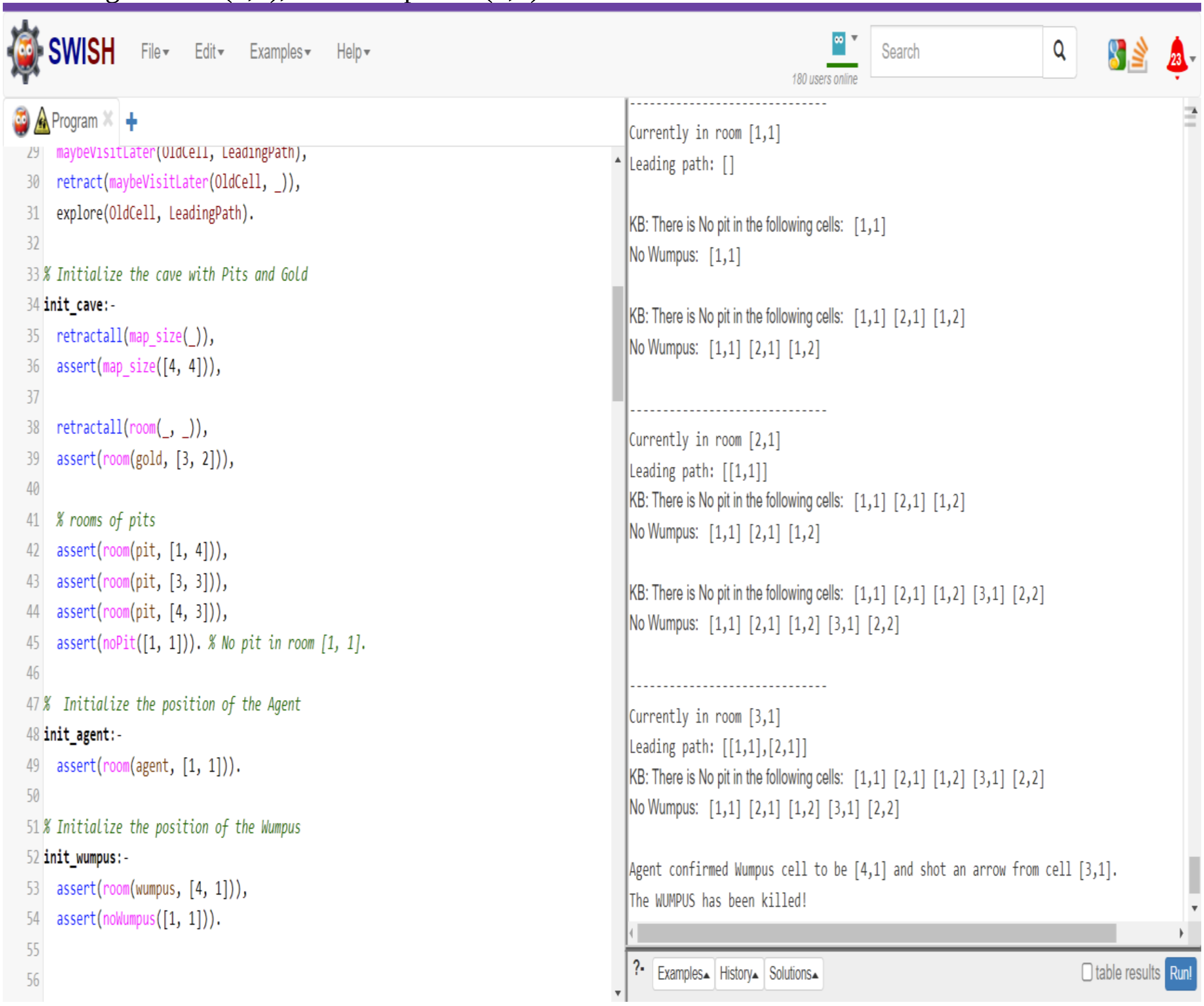
Second Configuration where pits locations are as shown in the screenshot
And the gold is in (2,2), the Wumpus in (1,3)



```prolog
    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                                                              1
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!
```

3rd Configuration where pits locations are as shown in the screenshot
And the gold is in (3,2), the Wumpus in (4,1)



```
    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])).
```

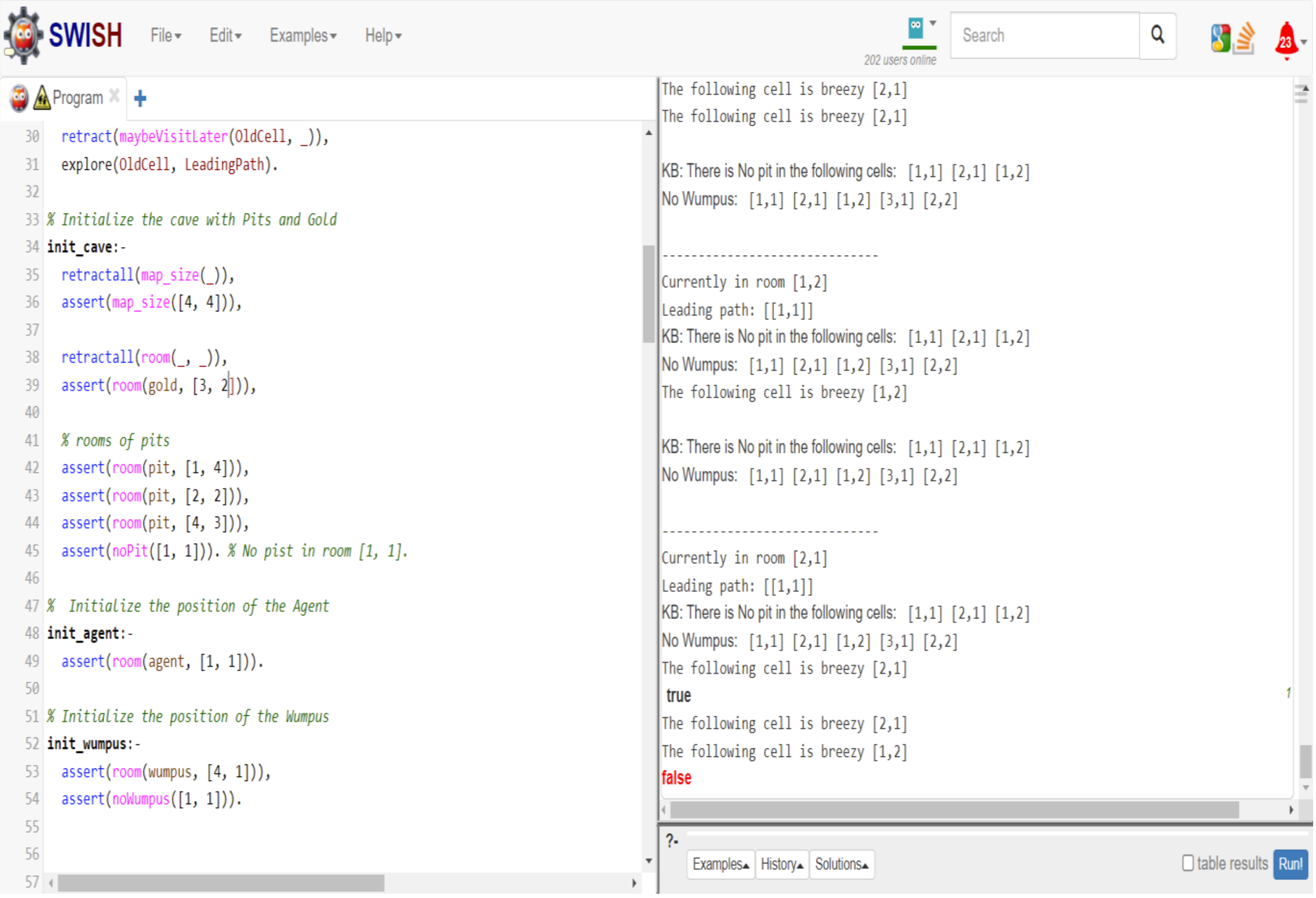
```
-----------------------------
Currently in room [1,1]
Leading path: []

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!
```

## III- Limitation to the code:

Configuration where pits locations are as shown in the screenshot

And the gold is in (3,2), the Wumpus in (4,1)

But it clearly shows that it fails in this configuration as the agent is stuck in room [2,2] since breeze is in both [2,1] and [1,2] the thing that perturbs the agent and fails.



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

Due to the time limitation we could not further implement several features of the game such as setting up the initial positions automatically thing that we hardcoded by giving initial cells, Plus the agent when it fails it stops instead of trying to remediate the problem. One last thing is the score feature, it is true that it is minor in this game nut if we had more time and more prolog knowledge we could have implemented it.