جامعة الأخوين

AL AKHAWAYN
UNIVERSITY

Artificial Intelligence

Wumpus Game Logical Agent Project

Oumayma El Ghamrasni

Supervised by Dr. Tajjeeddine Rachidi

March 21st, 2022

I. Introduction:

In this project, we were asked to code a logical agent for the Wumpus game. This game s agent is an example of a knowledge-based agent that links its general knowledge with its perceptions to infer the current state and select the best action. Due to time constraints, we were not able to code a perfect logical agent that solves the whole game. However, we were able to code an agent that respects the Wumpus game's rules such as deducing points after each move, updating the score after taking the gold, losing after falling into a pit or a Wumpus, etc. However, our agent only assumes the rooms where the Wumpus can be, and thus shoots randomly in one of the possible Wumpus rooms. Even though our agent is not perfectly coded, we were able to tackle most of the Wumpus game's rules in our code. The **key predicates used in our code:**

r(X,Y): r is the room in the position X and Y.

breeze(r(X,Y)): the breeze is in room r(X,Y).

wumpus(r(X,Y)): the Wumpus is in room r(X,Y).

pit(r(X,Y)): the pit is in room r(X,Y).

stench(r(X,Y)): the stench is in room r(X,Y).

glitter(r(X,Y)): the glitter is in room r(X,Y).

gold(r(X,Y)): the gold is in room r(X,Y).

GrabGold (r(X,Y)): grab the gold in room r(X,Y).

agentLocation(r(X,Y)): Agent is in room r(X,Y).

perceiveBreeze(true,r(X,Y)): The agent perception of the breeze in r(X,Y).

perceiveBreeze(false,r(X,Y)): no perception of breeze in r(X,Y).

perceivestench(true,r(X,Y)): The agent perception of the stench in r(X,Y).

perceivestench(false,r(X,Y)): no perception of stench in r(X,Y).

perceiveGlitter(true,r(X,Y)): The agent perception of the Glitter in r(X,Y).

perceiveGlitter(false, r(X,Y)): no perception of Glitter in r(X,Y).

adjacentTo (r (X, Y), r(T,U)) : r(X, Y) is adjacent to r(T,U).

permitted(true,r(X,Y)): r(X,Y) belongs to the grid.

permitted(false,r(X,Y)): r(X,Y) does not belong to the grid.

preshootWumpus: Choose appropriate location to shoot the Wumpus.

shootWumpus(r(X,Y)): shoot Wumpus in room r(X,Y) only from an adjacent room.

scream(true,r(X,Y)): Scream from room r(X,Y).

scream(false,r(X,Y)): No scream from room r(X,Y).

tell_KB(L): assert all the percepts of from location L to the knowledge base.

backtrack(AL,PL,B): the agent moved from AL (agent location) to PL (previous location) B times.

getAdjacentRooms(r(X,Y),List): a list of the adjacent rooms to r(X,Y).

MoveFromTo(r(X,Y), r(Z,W), T): move from room r(X,Y) to room r(Z,W) at time T.

II. Experiments:

1) Rules of the game:

-The agent will start with 50 points in its score. After each move, we will reduce the score by -1.

-If the agents get the gold, the score will be increased by 1000.

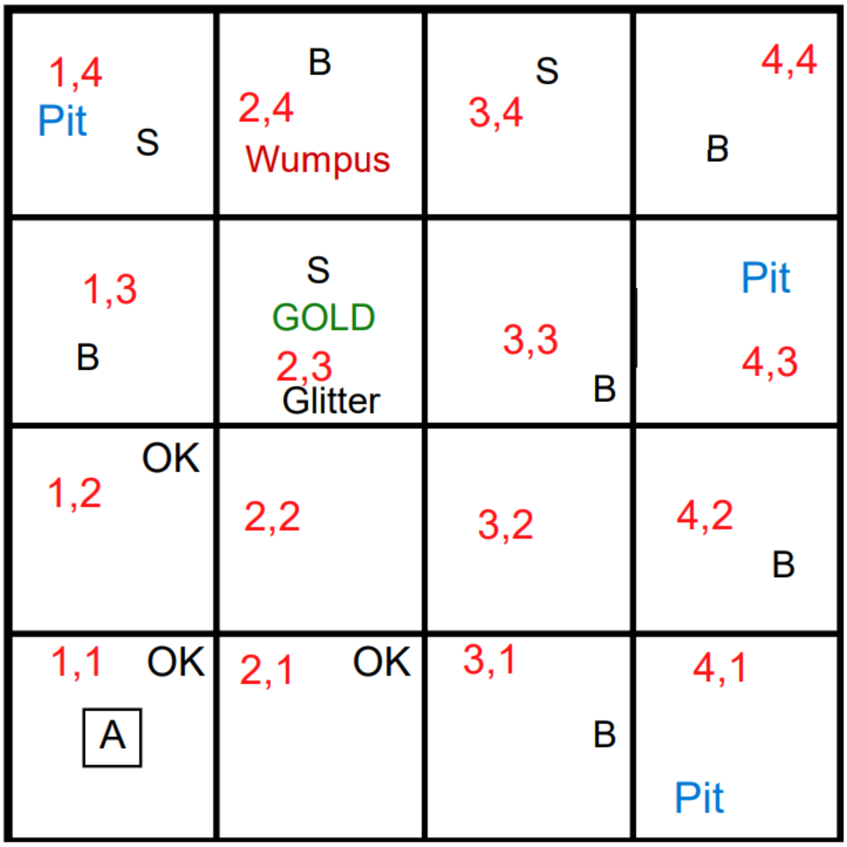
-If the agent falls in a pit the score will be decreased by 1000 and the game will be over.

-If the agent uses an arrow the score will be updated to S-10 and wins if it shoots the Wumpus and the game will be over if he does not.

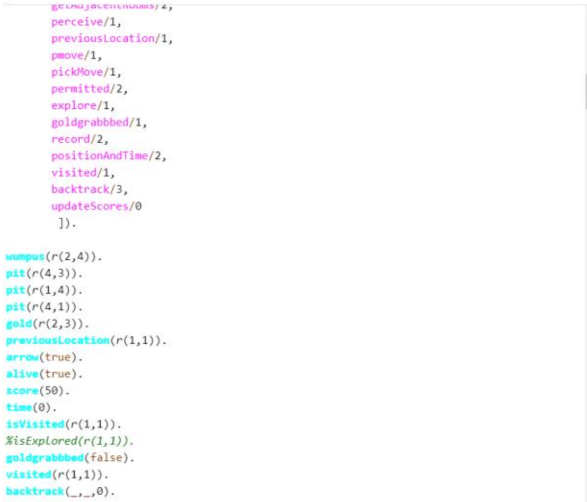
2) Overview:

In all the experiments, the agent will be following the same logic, but the results will be different. The first thing we will be calling walk where we get the agent location which is at the beginning (1,1). Then pmove will be called on the agent location to get the list of adjacent rooms to that location. After that, the list of adjacent rooms will be passed to move in which we either are going to choose a move from the list (permitted, not visited, and safe) or pick a random move in case the agent is stuck (surrounded by pits or wumpuses). In this case, if the agent is stuck, we need to backtrack to give it a chance to go back to a visited room where it is allowed to pick an already visited room. After exiting pmove, we call keepMoving, to check the state of the game, the move the agent does each time results either in a win or a failure state or none (the case where the agent should keep moving by calling move again). If the agent is alive, we keep calling walk. Each time the agent perceives a stench, glitter, or breeze it tells the knowledge base to mark that location and deduce conclusions from those perceptions such as the rooms of possible pits.

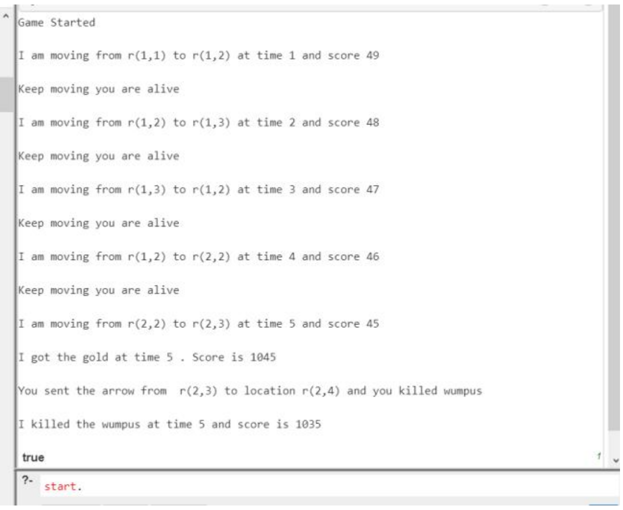
3) Experiment one of killing the Wumpus:



1,4 Pit S	B 2,4 Wumpus	S 3,4	4,4 B
1,3 B	S GOLD 2,3 Glitter	3,3 B	Pit 4,3
OK 1,2	2,2	3,2	4,2 B
1,1 OK A	2,1 OK	3,1 B	4,1 Pit



```
        getAdjacentRooms/2,
        perceive/1,
        previousLocation/1,
        pmove/1,
        pickMove/1,
        permitted/2,
        explore/1,
        goldgrabbbed/1,
        record/2,
        positionAndTime/2,
        visited/1,
        backtrack/3,
        updateScores/0
         ]).

wumpus(r(2,4)).
pit(r(4,3)).
pit(r(1,4)).
pit(r(4,1)).
gold(r(2,3)).
previousLocation(r(1,1)).
arrow(true).
alive(true).
score(50).
time(0).
isVisited(r(1,1)).
%isExplored(r(1,1)).
goldgrabbbed(false).
visited(r(1,1)).
backtrack(_,_,0).
```

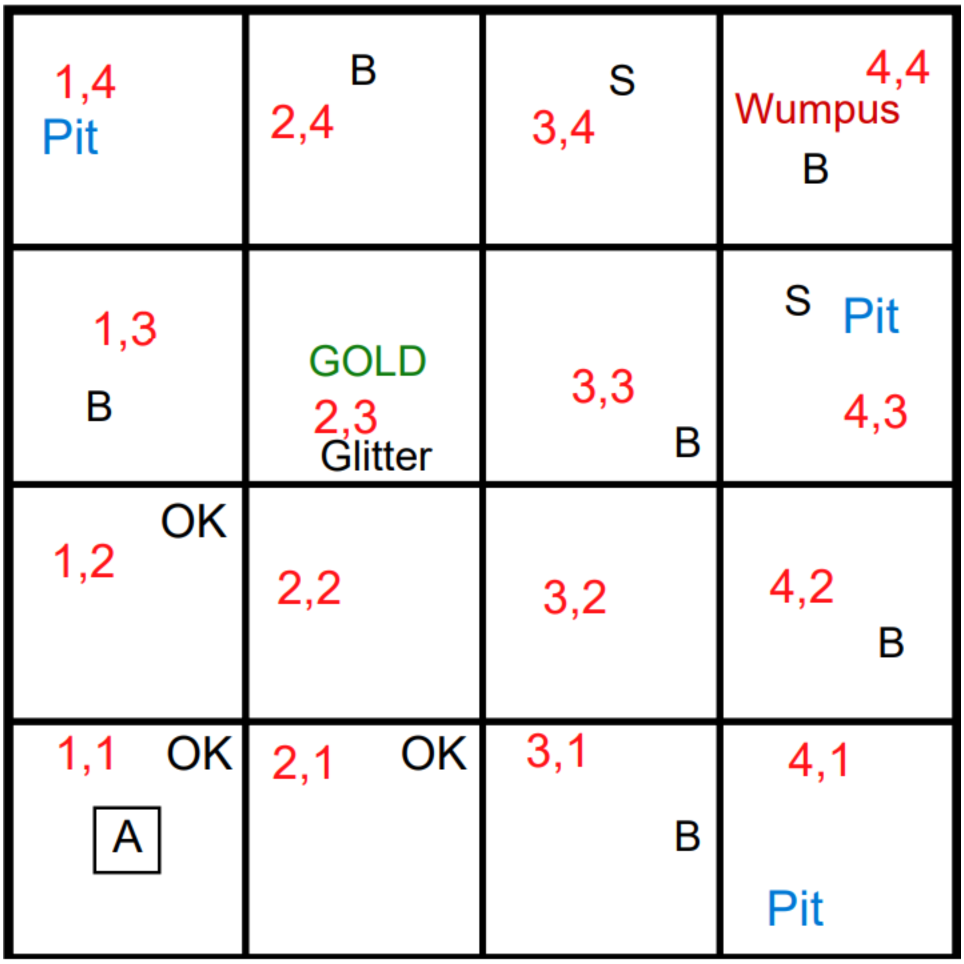


```
Game Started

I am moving from r(1,1) to r(1,2) at time 1 and score 49

Keep moving you are alive

I am moving from r(1,2) to r(1,3) at time 2 and score 48

Keep moving you are alive

I am moving from r(1,3) to r(1,2) at time 3 and score 47

Keep moving you are alive

I am moving from r(1,2) to r(2,2) at time 4 and score 46

Keep moving you are alive

I am moving from r(2,2) to r(2,3) at time 5 and score 45

I got the gold at time 5 . Score is 1045

You sent the arrow from  r(2,3) to location r(2,4) and you killed wumpus

I killed the wumpus at time 5 and score is 1035

true

?- start.
```

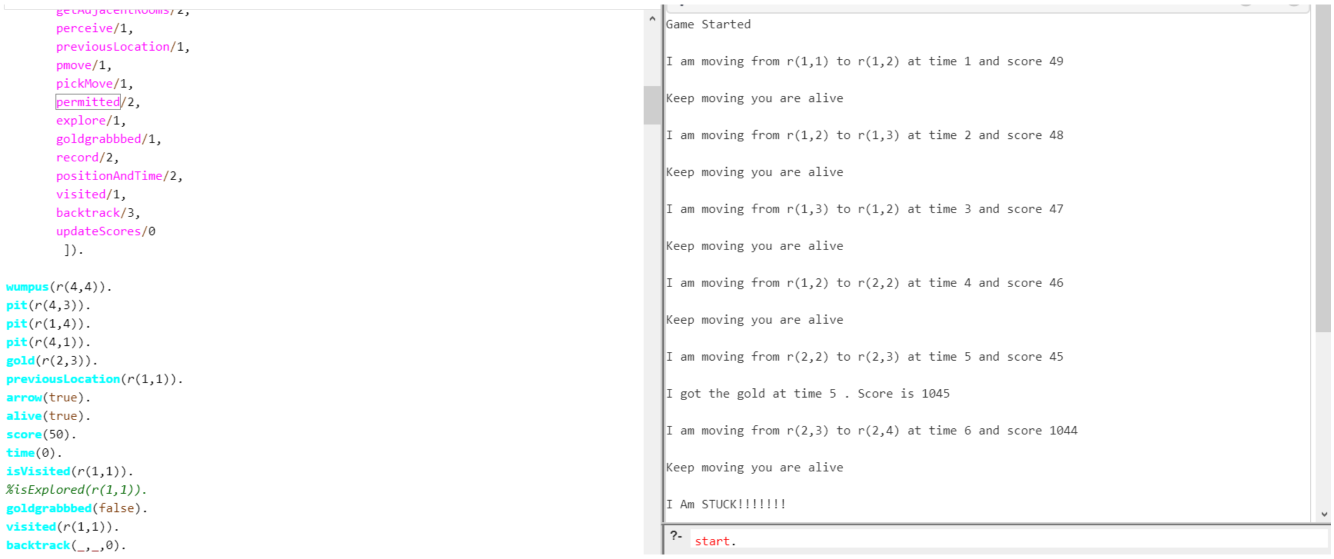
In this experiment, the agent keeps moving until it reached (1,3) where there is a breeze. The agent tells the knowledge base about its perceptions and based on reasoning made it concludes that there are possible pits in the adjacent rooms (1,4) and (1,3) except for (1,2) because it was already visited. In this case, the agent will backtrack and choose (1,2) then it will keep moving and gets the gold on its way until it reaches (2,3) where there is a stench. The agent will shoot in one of the adjacent rooms randomly, which in this case is (2,4), and will kill the Wumpus at a time of 5 and score 1035.

An Experiment with the same previous results:

I got the same results in this experiment, but it takes more time since the agent backtracks and takes random moves most of the time when it gets stuck:

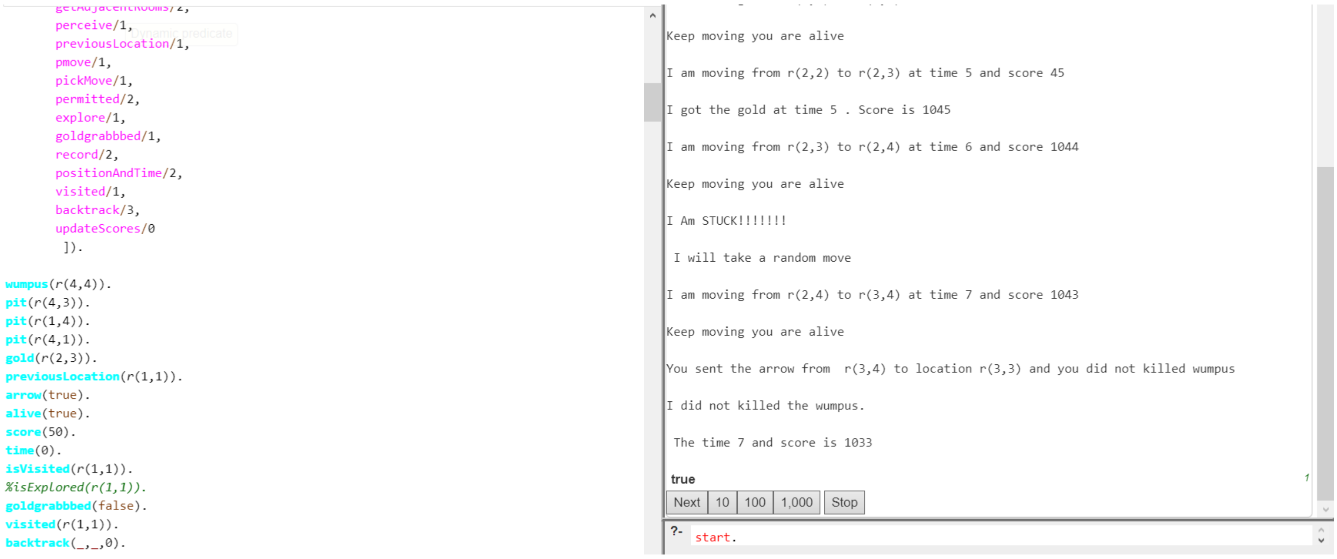


1,4 Pit	B 2,4	S 3,4	4,4 Wumpus B
1,3 B	GOLD 2,3 Glitter	3,3 B	S Pit 4,3
OK 1,2	2,2	3,2	4,2 B
1,1 OK A	2,1 OK	3,1 B	4,1 Pit



```
        getAdjacentRooms/2,
        perceive/1,
        previousLocation/1,
        pmove/1,
        pickMove/1,
        permitted/2,
        explore/1,
        goldgrabbbed/1,
        record/2,
        positionAndTime/2,
        visited/1,
        backtrack/3,
        updateScores/0
         ]).

wumpus(r(4,4)).
pit(r(4,3)).
pit(r(1,4)).
pit(r(4,1)).
gold(r(2,3)).
previousLocation(r(1,1)).
arrow(true).
alive(true).
score(50).
time(0).
isVisited(r(1,1)).
%isExplored(r(1,1)).
goldgrabbbed(false).
visited(r(1,1)).
backtrack(_,_,0).
```

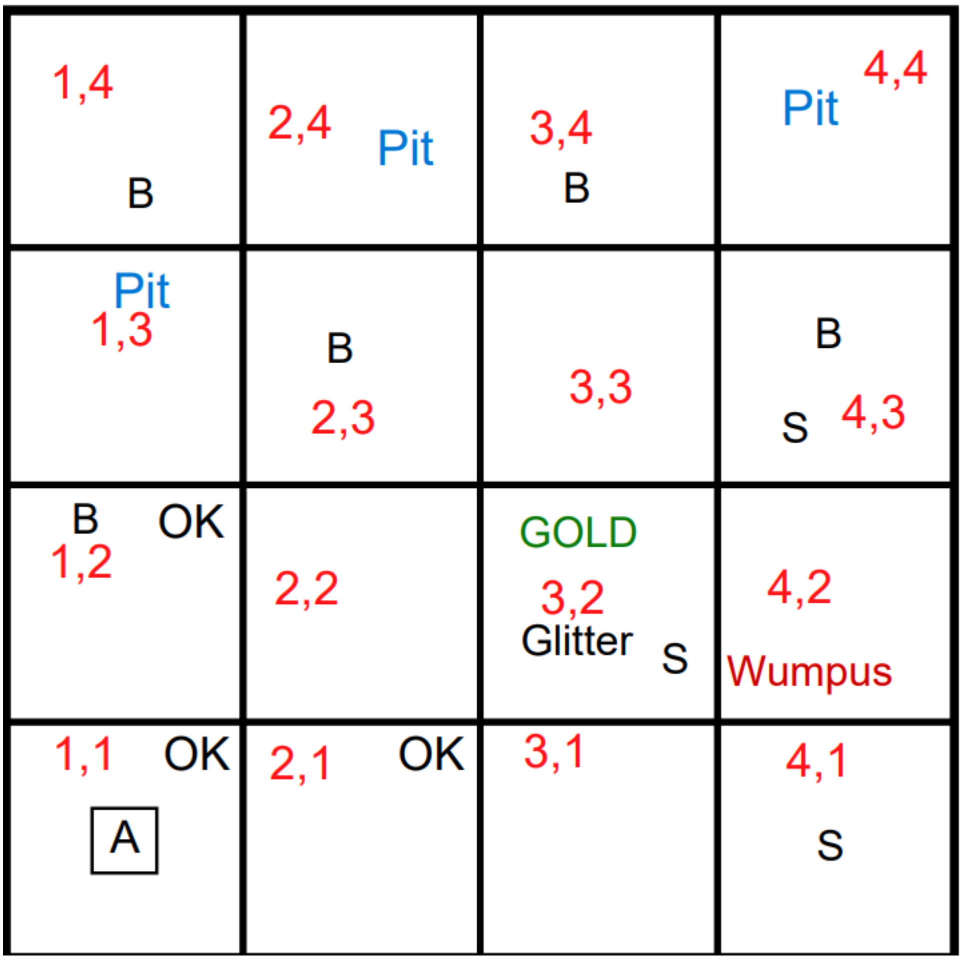
```
Game Started

I am moving from r(1,1) to r(1,2) at time 1 and score 49

Keep moving you are alive

I am moving from r(1,2) to r(1,3) at time 2 and score 48

Keep moving you are alive

I am moving from r(1,3) to r(1,2) at time 3 and score 47

Keep moving you are alive

I am moving from r(1,2) to r(2,2) at time 4 and score 46

Keep moving you are alive

I am moving from r(2,2) to r(2,3) at time 5 and score 45

I got the gold at time 5 . Score is 1045

I am moving from r(2,3) to r(2,4) at time 6 and score 1044

Keep moving you are alive

I Am STUCK!!!!!!!

?- start.
```



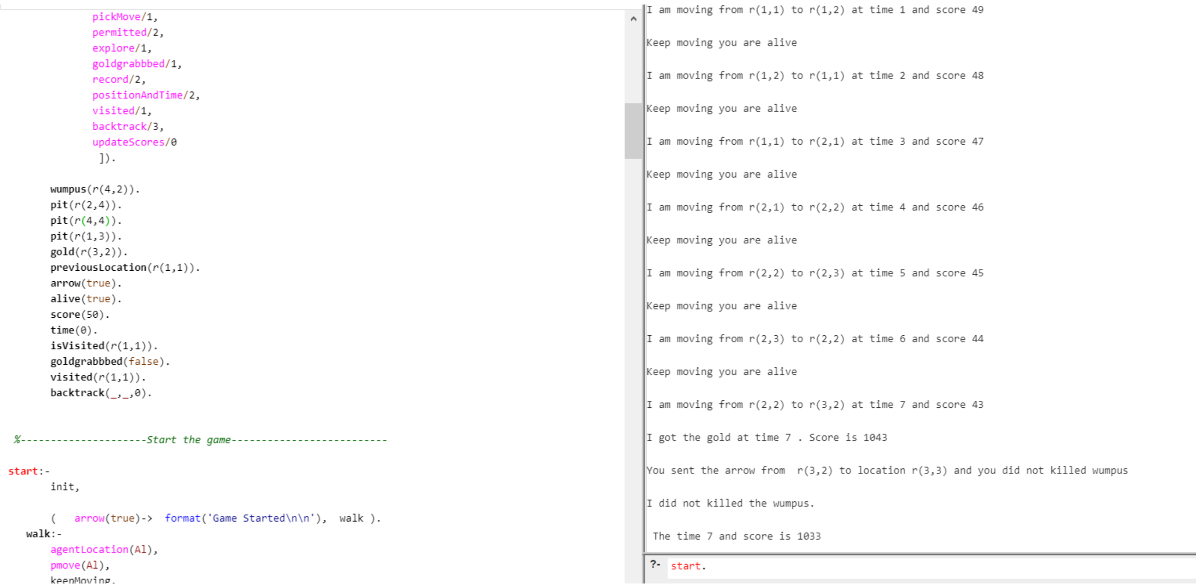
```
        getAdjacentRooms/2,
        perceive/1,
        previousLocation/1,
        pmove/1,
        pickMove/1,
        permitted/2,
        explore/1,
        goldgrabbbed/1,
        record/2,
        positionAndTime/2,
        visited/1,
        backtrack/3,
        updateScores/0
         ]).

wumpus(r(4,4)).
pit(r(4,3)).
pit(r(1,4)).
pit(r(4,1)).
gold(r(2,3)).
previousLocation(r(1,1)).
arrow(true).
alive(true).
score(50).
time(0).
isVisited(r(1,1)).
%isExplored(r(1,1)).
goldgrabbbed(false).
visited(r(1,1)).
backtrack(_,_,0).
```

```
Keep moving you are alive

I am moving from r(2,2) to r(2,3) at time 5 and score 45

I got the gold at time 5 . Score is 1045

I am moving from r(2,3) to r(2,4) at time 6 and score 1044

Keep moving you are alive

I Am STUCK!!!!!!!

 I will take a random move

I am moving from r(2,4) to r(3,4) at time 7 and score 1043

Keep moving you are alive

You sent the arrow from  r(3,4) to location r(3,3) and you did not killed wumpus

I did not killed the wumpus.

 The time 7 and score is 1033

true

?- start.
```

4) Experiment three of taking a random move and perceiving possible Wumpus and shooting in the wrong direction:



1,4 B	2,4 Pit	3,4 B	Pit 4,4
Pit 1,3	B 2,3	3,3	B S 4,3
B OK 1,2	2,2	GOLD 3,2 Glitter S	4,2 Wumpus
1,1 OK A	2,1 OK	3,1	4,1 S



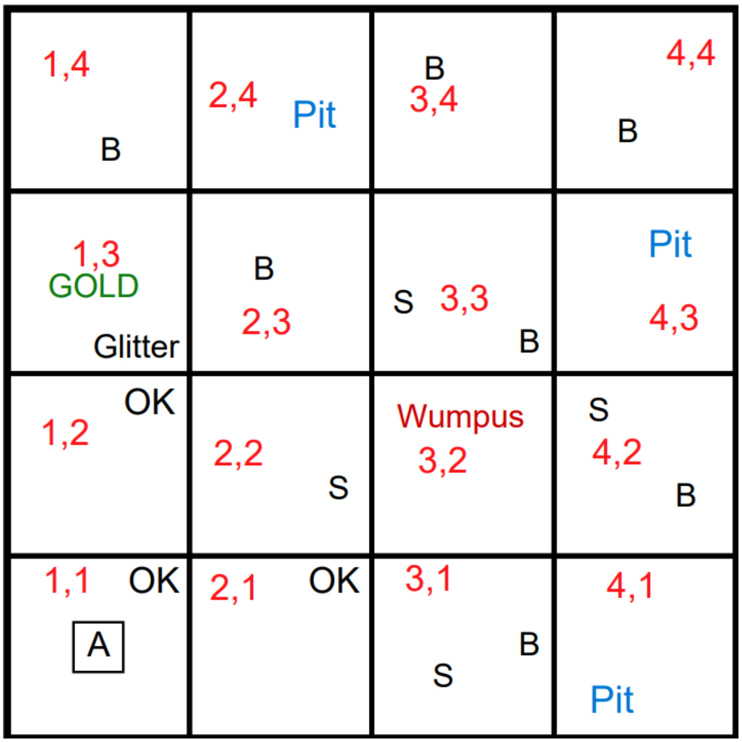
```
                pickMove/1,
                permitted/2,
                explore/1,
                goldgrabbbed/1,
                record/2,
                positionAndTime/2,
                visited/1,
                backtrack/3,
                updateScores/0
                 ]).

        wumpus(r(4,2)).
        pit(r(2,4)).
        pit(r(4,4)).
        pit(r(1,3)).
        gold(r(3,2)).
        previousLocation(r(1,1)).
        arrow(true).
        alive(true).
        score(50).
        time(0).
        isVisited(r(1,1)).
        goldgrabbbed(false).
        visited(r(1,1)).
        backtrack(_,_,0).


 %--------------------Start the game-------------------------

start:-
        init,

        (   arrow(true)->  format('Game Started\n\n'),  walk ).
   walk:-
        agentLocation(Al),
        pmove(Al),
        keepMoving.
```

```
I am moving from r(1,1) to r(1,2) at time 1 and score 49

Keep moving you are alive

I am moving from r(1,2) to r(1,1) at time 2 and score 48

Keep moving you are alive

I am moving from r(1,1) to r(2,1) at time 3 and score 47

Keep moving you are alive

I am moving from r(2,1) to r(2,2) at time 4 and score 46

Keep moving you are alive

I am moving from r(2,2) to r(2,3) at time 5 and score 45

Keep moving you are alive

I am moving from r(2,3) to r(2,2) at time 6 and score 44

Keep moving you are alive

I am moving from r(2,2) to r(3,2) at time 7 and score 43

I got the gold at time 7 . Score is 1043

You sent the arrow from  r(3,2) to location r(3,3) and you did not killed wumpus

I did not killed the wumpus.

 The time 7 and score is 1033

?- start.
```

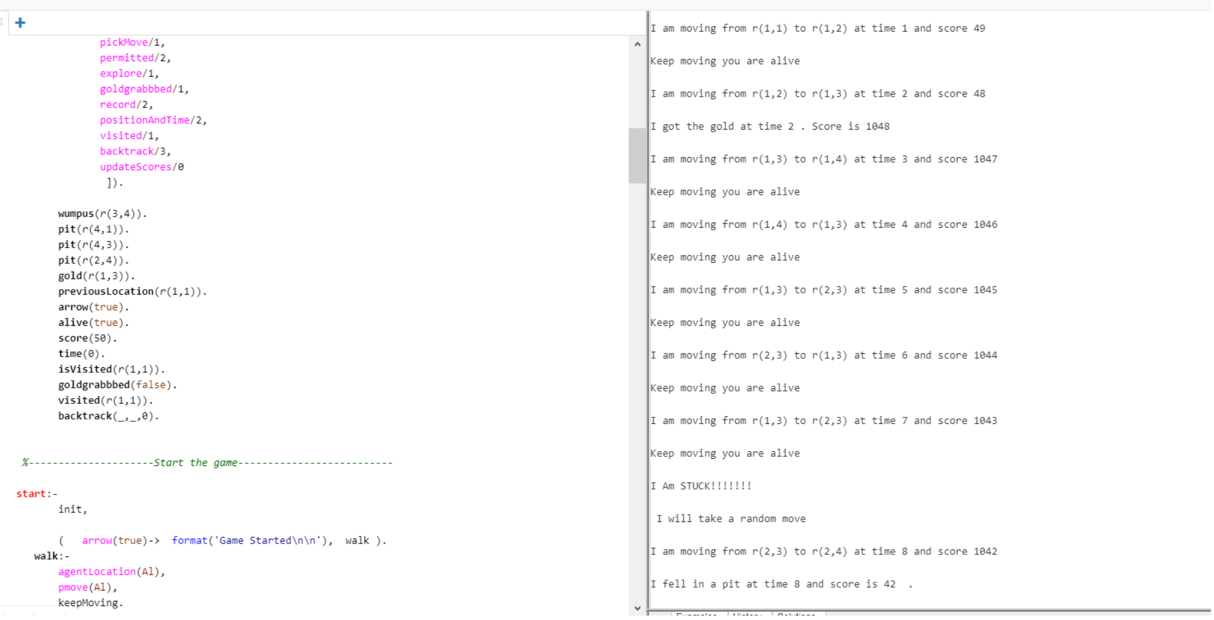
The agent starts from (1,1) then calls walk where we get the agent location and call pmove to get the location of the next move among the list of adjacent rooms to the agent location. However, from this list, we get only the move that is permitted, not visited yet, and safe.

Following this logic, the agent moves from (1, 1) to (1,2), and then it finds a breeze in this room which means that its adjacents are possible pits. Therefore, it backtracks and goes back to (1,1) to choose a better room which is (2,1). Then the agent goes to (2,2) and then to (2,3). Since (2,3) has a breeze then the knowledge base will tell the agent that there is a possible pit in its adjacents and therefore it will backtrack and go to (2,2). The agent will go to (3,2) where there is glitter and a stench. It will get the gold and then shoot randomly in one of the adjacents marked as possible Wupuses based on the perceptions provided to the knowledge base. It shoots (3,3) but since the Wumpus is in (4,2) the game is over.

5) Experiment two of taking a random move falling in a pit:



1,4 B	2,4 Pit	B 3,4	4,4 B
1,3 GOLD Glitter	B 2,3	S 3,3 B	Pit 4,3
OK 1,2	2,2 S	Wumpus 3,2	S 4,2 B
1,1 OK A	2,1 OK	3,1 B S	4,1 Pit

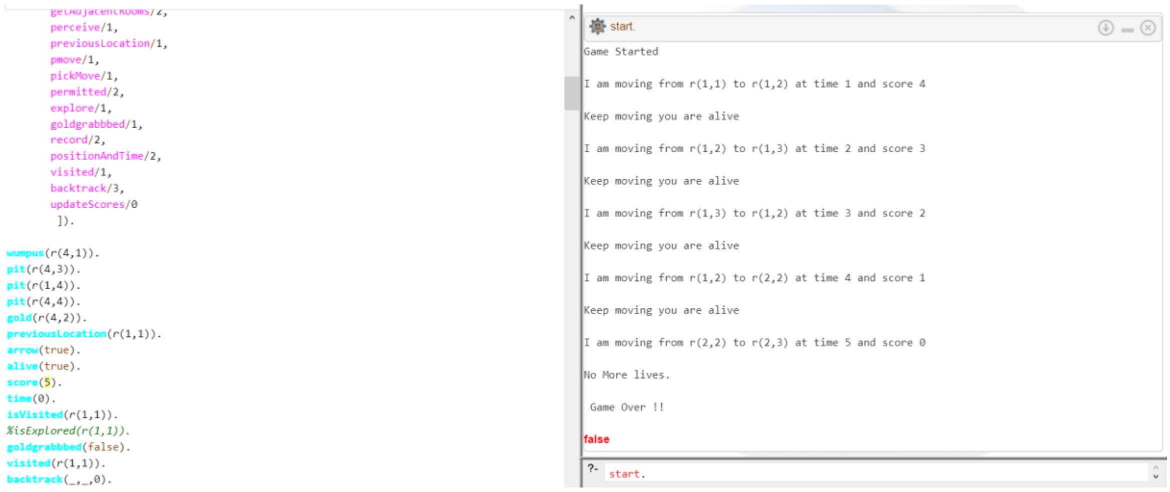


```
+
              pickMove/1,
              permitted/2,
              explore/1,
              goldgrabbbed/1,
              record/2,
              positionAndTime/2,
              visited/1,
              backtrack/3,
              updateScores/0
               ]).

       wumpus(r(3,4)).
       pit(r(4,1)).
       pit(r(4,3)).
       pit(r(2,4)).
       gold(r(1,3)).
       previousLocation(r(1,1)).
       arrow(true).
       alive(true).
       score(50).
       time(0).
       isVisited(r(1,1)).
       goldgrabbbed(false).
       visited(r(1,1)).
       backtrack(_,_,0).


 %--------------------Start the game-------------------------

start:-
       init,

       (   arrow(true)->  format('Game Started\n\n'),  walk ).
   walk:-
       agentLocation(Al),
       pmove(Al),
       keepMoving.
```

```
I am moving from r(1,1) to r(1,2) at time 1 and score 49

Keep moving you are alive

I am moving from r(1,2) to r(1,3) at time 2 and score 48

I got the gold at time 2 . Score is 1048

I am moving from r(1,3) to r(1,4) at time 3 and score 1047

Keep moving you are alive

I am moving from r(1,4) to r(1,3) at time 4 and score 1046

Keep moving you are alive

I am moving from r(1,3) to r(2,3) at time 5 and score 1045

Keep moving you are alive

I am moving from r(2,3) to r(1,3) at time 6 and score 1044

Keep moving you are alive

I am moving from r(1,3) to r(2,3) at time 7 and score 1043

Keep moving you are alive

I Am STUCK!!!!!!!

 I will take a random move

I am moving from r(2,3) to r(2,4) at time 8 and score 1042

I fell in a pit at time 8 and score is 42  .
```

In this experiment, when the agent goes to (1,4) it will be surrounded by possible pits and therefore backtracks to (1,3) to find a better action. Thus, it will select (2,3) however the agent will face the same problem and goes back to (1,3) again. The agent will go back to (2,3) again since it cannot select another room and therefore get stuck there where there is a breeze and therefore surrounded by possible pits. The knowledge base will suggest to the agent to take a random move. When it will move to (2,4) it will fall in a pit and thus the game will be over and the score of the agent will be reduced from 1042 (since it got the gold) to 42 (since it fell in a pit).

6) Experiment four of reducing the score to 5 instead of 50 and see what will happen:

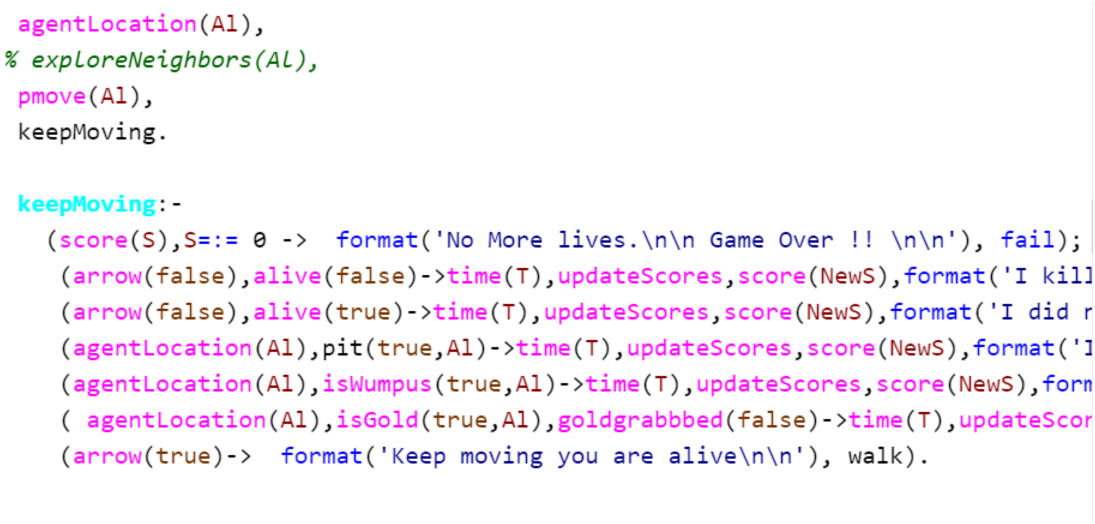


```
            getAdjacentRooms/2,
            perceive/1,
            previousLocation/1,
            pmove/1,
            pickMove/1,
            permitted/2,
            explore/1,
            goldgrabbbed/1,
            record/2,
            positionAndTime/2,
            visited/1,
            backtrack/3,
            updateScores/0
             ]).

wumpus(r(4,1)).
pit(r(4,3)).
pit(r(1,4)).
pit(r(4,4)).
gold(r(4,2)).
previousLocation(r(1,1)).
arrow(true).
alive(true).
score(5).
time(0).
isVisited(r(1,1)).
%isExplored(r(1,1)).
goldgrabbbed(false).
visited(r(1,1)).
backtrack(_,_,0).
```

```
Game Started

I am moving from r(1,1) to r(1,2) at time 1 and score 4

Keep moving you are alive

I am moving from r(1,2) to r(1,3) at time 2 and score 3

Keep moving you are alive

I am moving from r(1,3) to r(1,2) at time 3 and score 2

Keep moving you are alive

I am moving from r(1,2) to r(2,2) at time 4 and score 1

Keep moving you are alive

I am moving from r(2,2) to r(2,3) at time 5 and score 0

No More lives.

 Game Over !!

false
```

```
?- start.
```

In this experiment, we can see that the agent keeps moving from one room to another until it has no more lives. This can be ensured by the part of code shown bellow:



```
 agentLocation(Al),
% exploreNeighbors(AL),
 pmove(Al),
 keepMoving.

 keepMoving:-
   (score(S),S=:= 0 ->  format('No More lives.\n\n Game Over !! \n\n'), fail);
    (arrow(false),alive(false)->time(T),updateScores,score(NewS),format('I kil
    (arrow(false),alive(true)->time(T),updateScores,score(NewS),format('I did r
    (agentLocation(Al),pit(true,Al)->time(T),updateScores,score(NewS),format('I
    (agentLocation(Al),isWumpus(true,Al)->time(T),updateScores,score(NewS),form
    ( agentLocation(Al),isGold(true,Al),goldgrabbbed(false)->time(T),updateScor
    (arrow(true)->  format('Keep moving you are alive\n\n'), walk).
```

- The first thing the agent does after the game starts is to call walk.
- In walk we check the agent location and call pmove where we get the adjacent rooms to the agent location and put them in a move list.
- When we call move, the agent will ask the knowledge base for the next move and it will get as an answer the room that is permitted, safe, and not visited yet.
- When we finish with pmove we exit it, and we go back to the call of walk and call keepMoving where the first thing we do is to check if the score is equal to 0.

- After some moves the agent score became 0 and thus when it called keepMoving to move to the next room, the score was equal to 0 and thus could not continue the game.

III. Limitations of the code:

- One of the limitations of our agent is that it shoots randomly in one of the adjacent rooms of the stench, as I mentioned in experiment 3. Thus, our agent will fail to shoot the Wumpus. This can be solved by using heuristics to keep track of how far we are from the Wumpus, and the agent will shoot the room with the highest ranking.
- I think that our solution would have been better if we used informed search instead of uninformed search. In other words, if we used heuristics as I mentioned above, the agent will be able to conclude how close it is to the Wumpus and therefore will shoot in the right room. However, since we used uninformed search, when the agent perceives a stench, it shoots randomly in one of the adjacent rooms.
- In our solution, the agent gets stuck when it is surrounded by possible pits because it did not make enough perceptions to conclude the right room of the pit. Therefore, our solution would have been better if we coded our agent in a way that will allow it to give more facts to the knowledge base and perform more reasoning to choose the best action.

IV. Conclusion:

To conclude, this project was challenging especially since my teammate, and I was using prolog for the first time. Regardless of how hard and time-consuming this project was, it was very enriching because we put hands on prolog and had the opportunity to implement a logical agent even though it is not perfect. Even though it was frustrating to implement the code from scratch multiple times before we agreed on a final version, getting the expected results from our logical agent was satisfying.