School of Science and Engineering

Project #2 Report: Wumpus Game Logican Agent.

CSC 4301
Artificial Intelligence
Summer 2022

Imane Ettabaa
82345

Teammate: Mahmoud Erradi

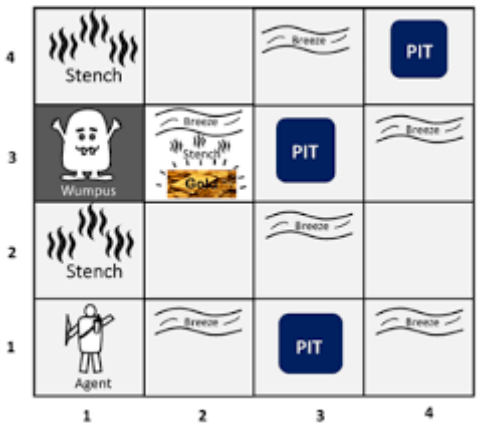
Supervised by:

Dr. Tajjeeddine Rachidi

June 29th, 2022

Introduction:

The Wumpus world is a cave with 4/4 identical rooms connected by passages. Consequently, there are 16 rooms over all, which are all connected to one another. We have a knowledge based agent that will move forward in this world. A beast known as Wumpus lives in a room of the cave and eats anyone who enters the room. The Wumpus can be shot by the agent but the agent has a single arrow. If an agent falls into one of the bottomless Pits rooms he will be imprisoned there indefinitely. The intriguing aspect of this cave is the potential discovery of a gold mine in one of the rooms. So the agent's goal is to find the gold and climb out the cave without falling into Pits or being eaten by Wumpus.



In this project, we had to write a single iteration of a logical agent to handle a few features in a prolog-coded program for the Wumpus game. An initial position is provided to the agent, and only one action is requested. For instance, questions concerning the room's security, grabbing the gold, or shooting the wumpus can be directed at the agent. Additionally, we were charged with testing our agent's performance in various setups.

Key predicated used in our project:

a. r (X,Y) the room in position (x,y).
b. adjacentTo (r(X,Y), r(Z,T)) room r(T,Z) is adjacent to room r(X,Y)
c. breeze(r(X,Y)) there is a breeze in room r(X,Y).
d. stench(r(X,Y)) room R(X,Y) stenches
e. glitter (r (X,Y)) there is gold in room r(X,Y)
f. safe(r(x,y)) room r(X,Y) is safe
g. grabGold () if there is glitter grab the gold in the current room
h. shootWumpus(R(X,Y)) shoots the Wumpus in room R(X,Y) only from adjacent rooms.
i. currentRoom (r(X,Y)) it returns the current room.
j. move (r(Z,T) it is for testing the configuration in the same world
k. getAdjacentRooms (r(X,Y),L)

Overview:

In all the experiments, the agent will be following the same logic but different results for each configuration. The agent location will be always (1,1). Then safe will be called to get the list of adjacent rooms to that location in which we are going to have either safe, stench, glitter or breeze. If it is safe we move to another room. If there is glitter we grab the gold. If there is a stench we shoot the wumpus, and if there is a breeze we check a different room. The agent will have 2000 different configurations that we can evaluate using the link provided in the project requirement
https://github.com/alexroque91/wumpus-world-prolog/blob/master/worldBuilder.pl

Let's play!

```
% Starts the game
load(N):-
    recreateWorld(N),
    currentRoom.
```

```
%Clears and builds 4x4 world
recreateWorld(N) :-
    clearWorld,
    buildWorld(N).
```

```
%Removes all objects from world
clearWorld :-
    retractall(gold(_)),
    retractall(wumpus(_)),
    retractall(pit(_)).
```

- In our code, the load predicate takes N number of configurations which is between 1 and 2000. When it is correct, it returns true and the message game starts and the current room. The load calls recreate which by itself calls clearworld and the clearworld removes all the objects from the current starting configuration.

```
buildWorld(30) :-
    asserta(pit(r(3,2))),
    asserta(pit(r(4,3))),
    asserta(gold(r(2,2))),
    asserta(wumpus(r(3,4))).
```

- BuildWorld takes the value given to N to load in the beginning. In my report I chose to test using the configuration number 30 represented above.

```
?- load(30).
Game Started:

You are at room [1,1].
true.
```

- Now after creating the world, we get a message that the agent is in position (1,1) which is the case for all configurations.

```
% Agent Position
r(1,1). %Start at cave entry r(1,1)
```

```
%move (works like a teleport)
move(r(Z,T)) :- r(X,Y) -> retract(r(X,Y)), asserta(r(Z,T)), currentRoom.
```

```
?- move(r(2,4)).
Game Started:

You are at room [2,4].
true.
```

- We can change the agent position by using the move predicate as you can see in the above screenshot the agent moved from position (1,1) to position (2,4). The move

predicate works like a teleport. It allows for testing without having to change the configuration or the code.

```
%Returns current room
currentRoom :- r(X,Y), pit(r(X,Y)) -> format('Oops, it appears you moved into a pit!\n');
    r(X,Y), wumpus(r(X,Y)) -> format('Oops, it appears you moved into the Wumpus!\n');
    r(X,Y), format('Game Started:\n\nYou are at room [~w,~w].', [X,Y]).
```

- The currentRoom predicate checks the position of the agent and it compares it with the position of the wumpus and the pit. Whenever the agent moves into the pit it gets or the wumpus it gets a warning message. Otherwise, the message that the game started appears.

```
?- stench(r(2,4)).
true.

?-
```

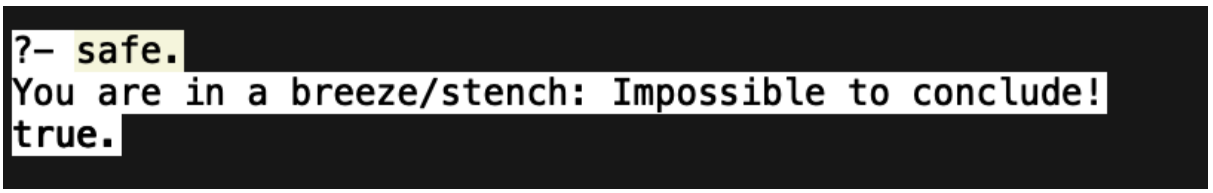
- After moving to (2,4) position the agent asked if there is a stench in that position and it returned true.

```
?- shootWumpus.
You shot the Wumpus at [3,4]!!!
true.
```

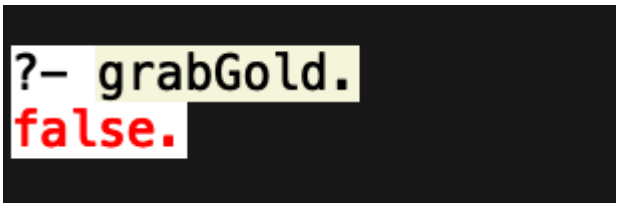
- After checking that there is a stench the agent shot the wumpus. The wumpus is shot successfully and the room location of the wumpus is returned.

```
?- move(r(4,4)).
Game Started:

You are at room [4,4].
true.

?- breeze(r(4,4)).
true .
```

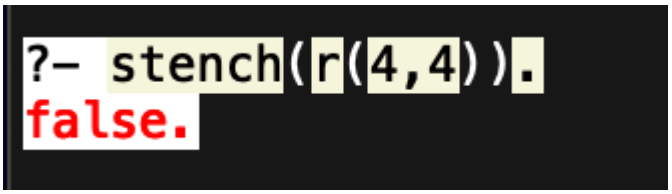
- Now the agent moved to room (4,4) successfully then checked for the breeze and it returned true which means there is a pit in an adjacent cell on (4,4).



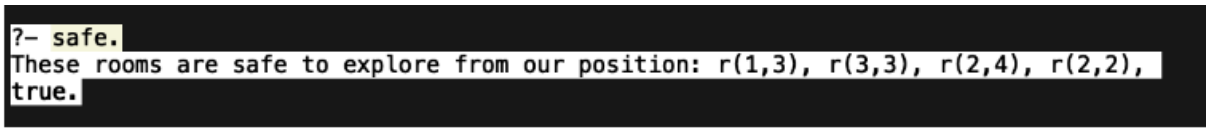
- To check the performance of the agent in the different configurations we checked the safety of the room (4,4) that has a breeze and it returned the above message.



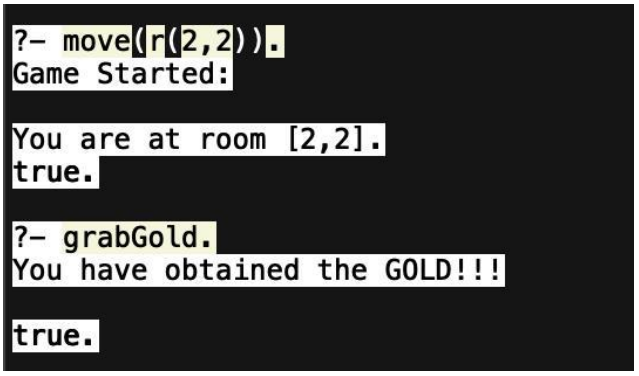
- Again we checked if there is gold in room (4,4) it returned false.



- And to check the performance of the configurations we asked if there is stench in position (4,4) and it returned false which is true because the wumpus is on position (3,4), and the stench can not be on position (4,4).



- After that we checked for the safe rooms we got (1,3), (3,3), (2,4), (2,2) are safe.



- Finally, we moved to room (2,2) and we grabbed the gold. The gold is obtained successfully.

→ Now moving on to a different world which is 75 and it has the following configurations.

```
buildWorld(75) :-
    asserta(pit(r(1,3))),
    asserta(pit(r(3,4))),
    asserta(pit(r(4,4))),
    asserta(gold(r(3,2))),
    asserta(wumpus(r(3,4))).
```

```
?- load(75).
Game Started:

You are at room [1,1].
true.
```

- As mentioned before we started the game using load we are in room (1,1)

```
?- move(r(2,2)).
Game Started:

You are at room [2,2].
true.
```

- We moved to room (2,2) using move predicate.

```
?- safe.
These rooms are safe to explore from our position: r(1,2), r(3,2), r(2,3), r(2,1),
true.

?-
```

- We checked the safety of the rooms around the (2,2) room and the following (1,2), (3,2), (2,3), (2,1) are safe.

```
?- move(r(2,4)).
Game Started:

You are at room [2,4].
true.

?- safe.
You are in a breeze/stench: Impossible to conclude!
true.
```

- To check the performance we checked room (2,4) which is not between the safe rooms if it is safe and we got that it has breeze/stench.

```
You are at room [2,4].
true.

?- shootWumpus.
You shot the Wumpus at [3,4]!!!
true.
```

- Because we had a stench in room (2,4) we shot the wumpus. The wumpus is shot successfully and the room of its position is returned which is (3,4).

```
?- move(r(3,2)).
Game Started:

You are at room [3,2].
true.

?- grabGold.
You have obtained the GOLD!!!
```

- Now moving to position (3,2) we grabbed gold. Gold is grabbed successfully.

```
?- move(r(1,1)).
Game Started:

You are at room [1,1].
true.

?- grabGold.
false.
```

- To check the performance even though we grabbed the gold in position (3,2). We checked if there was gold in room (1,1) and it returned false.

```
?- shootWumpus.
false.
```

- We shot the wumpus from position (1,1) and it returned false because the wumpus is already killed in room (3,4).

→ Finally, we are going to test world 114 with the following configurations.

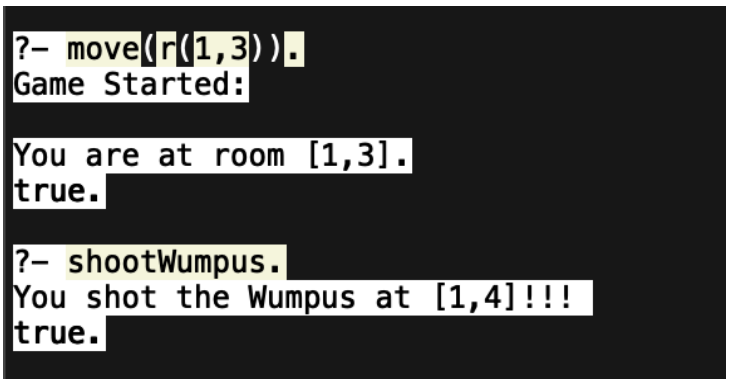
```
buildWorld(114) :-
    asserta(pit(r(4,4))),
    asserta(gold(r(1,3))),
    asserta(wumpus(r(1,4))).
```

```
?- load(114).
Game Started:

You are at room [2,3].
true.

?- safe.
These rooms are safe to explore from our position: r(1,3), r(3,3), r(2,4), r(2,2),
true.
```

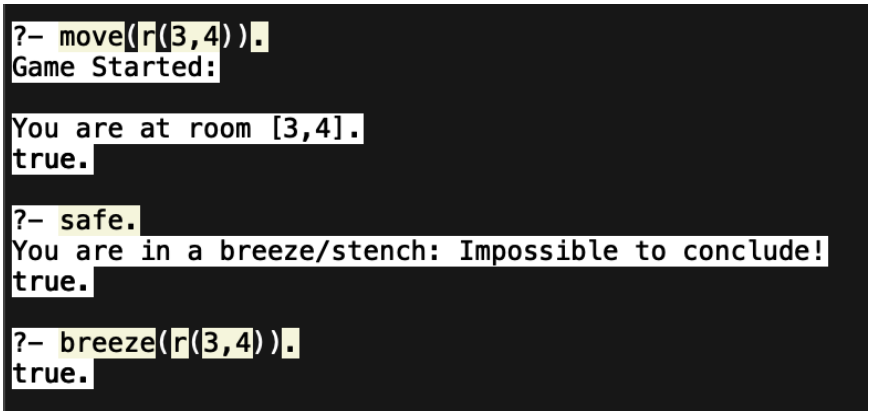
- We started the game and we checked the safe rooms around (2,3) and we got (1,3), (3,3), (2,4), (2,2) are the safe rooms

```
?- grabGold.
false.

?-
```

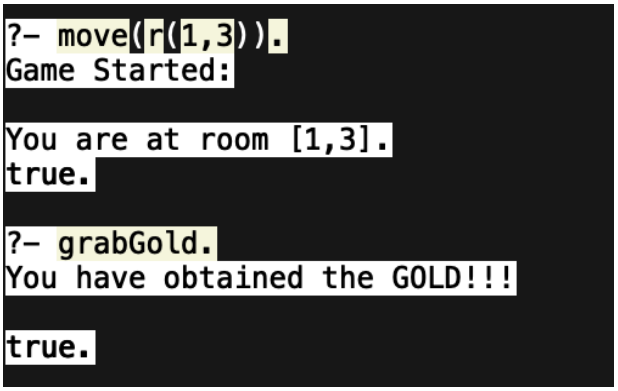
- We grabbed the gold in room (2,3) and it returned false because there is no gold in this cell.



- We shot the wumpus from position (1,3). The wumpus is shot successfully and the position of the wumpus is returned which is



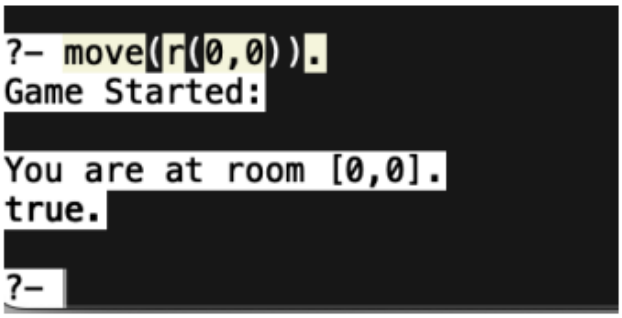
- We moved to cell (3,4) and we checked its safety. We found that there is breeze in that cell.

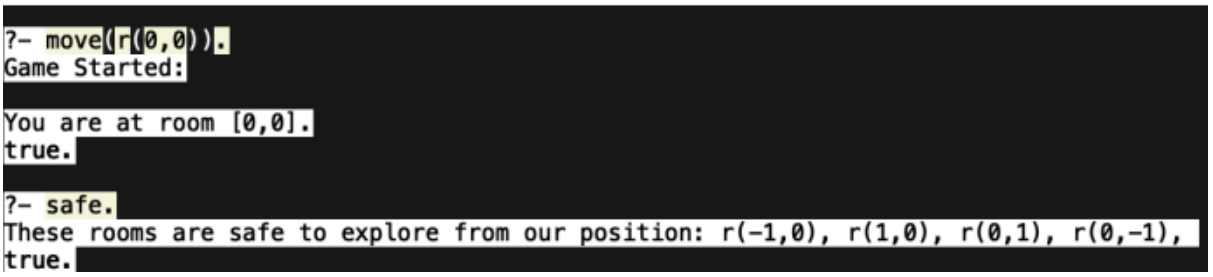


- Finally, We moved to room (1,3) and we grabbed it. The gold was successfully grabbed.

Limitations of the solution:

The results seen in the screenshots above are consistent, meaning that they always yield the same results given the same starting configurations. However, there are some ways to make the code fail: Using the move/1 predicate, it is possible to go out of bounds, as such:





There is no point in going out of bounds, as nothing is generated there. Nevertheless, it is possible. This problem can be solved with more time by implementing wall-like features to limit the scope of the game. Otherwise, every other feature works as intended and the results remain consistent with our expectations.