Report of Project #2

Yassine Drafate & Abderrazak Lamdouar

Pr. Tajjeeddine Rachidi

CSC 4301

SUMMER 2022

I. Introduction :

xThe following is a documentation showing/discussing a single iteration logical agent implemented using Prolog, and contains key predicates, meaning of the variables, screenshots of experiments using various configurations with the corresponding performance rate, and limitations of the proposed solution along with possible resolution.

II. List of Predicates and Functions and their Explanation:

•start: initializes the game based on user input.

•setHunterAt (): sets hunter position and facing direction based on user input.

•setPitsPos (): retracts all pits and asserts new pits into the KB.

•pitsProb (X, Y): adds a pit in (X, Y) with a probability of 25%.

•setPitsPos (): assign pits randomly with a 25% probability.

•setWumpusPos (): retracts Wumpus and sets a new one.

•setGoldPos(): retracts gold and sets a new one.

•room (X, Y): returns all the rooms, or if (X, Y) is specified; checks if room exists.

•getAdjacentRooms (r (X, Y), L): returns a list of adjacent rooms to the given (X, Y).

•adjacentTo (r (X, Y), r (Z, A)): checks whether the given rooms are adjacent.

•breeze (r (X, Y)): returns all rooms with breeze. If (X, Y) are specified, checks if breeze exists.

•pit (r (X, Y)): returns all rooms with pits. If (X, Y) checks if breeze exists.

•gold (r (X, Y)): returns all rooms with gold. If (X, Y) is specified check if gold exists.

•wumpus (r (X, Y)): returns Wumpus position. If (X, Y) is given check if it contains Wumpus.

•stench (r (X, Y)): returns all rooms with stench. If (X, Y) are given check if stench exists.

•Safe (): returns all the rooms that are both adjacent to the hunter's current room and safe.

•safeWumpus (): check if the Wumpus is safe (the hunter is not in an adjacent room).

•wumpusAlive (): check if the Wumpus is still alive.

•hasArrow (): check if the hunter has an arrow.

•wallCheck (r (X, Y)): check if (X, Y) is not a valid room.

•shootWumpus (): shoot Wumpus if possible.

•shootWumpusWithDirection (): shoot the Wumpus if and only if the hunter is adjacent to the Wumpus and it is facing it.

•grabGold (): grab gold if and only if the hunter is at the same position as the gold.

•TurnLeft (): change the direction of the hunter 90 degrees to the left.

•TurnRight (): change the direction of the hunter 90 degrees to the right.

III. Experiments with Different Configurations:

1. Configuration 1:

Hunter in position (3 , 1).
Breeze in : (1 , 2), (1 , 4), (2 , 2), (2 , 3), (2 , 4), (3 , 1), (3 , 2), (3 , 3), (3 , 4), (4 , 2), (4 , 3), (4 , 4).
Pit in : (1 , 3), (3 , 2), (3 , 3), (3 , 4), (4 , 3).
Gold in position (4 , 1).
Wumpus in position (2 , 1).
Stench in : (1 , 1), (2 , 2), (3 , 1).

=>Screens of the functions being executed in the SWI-Prolog CLI:

```
?- start.
Please enter the X coordinate of the hunter:
|: 1.
Please enter the Y coordinate of the hunter:
|: 3.
Please enter the initial direction of the wumpus
|: w.

true.

?- hunterAt(r(X,Y),Z).
X = 1,
Y = 3,
Z = w.

?- wumpus(r(X,Y)).
X = 1,
Y = 2.

?- pit(r(X,Y)).
X = 2,
Y = 3 ;
X = 3,
Y = 1 ;
X = Y, Y = 3 ;
X = 3,
Y = 4 ;
X = 4,
Y = 3.

?- gold(r(X,Y)).
X = 1,
Y = 4.
```

```
?- breeze(r(X,Y)).
X = 1,
Y = 3 ;
X = 2,
Y = 1 ;
X = Y, Y = 2 ;
X = 2,
Y = 3 ;
X = 2,
Y = 4 ;
X = 2,
Y = 4 ;
X = 3,
Y = 2 ;
X = 3,
Y = 2 ;
X = Y, Y = 3 ;
X = Y, Y = 3 ;
X = Y, Y = 3 ;
X = 3,
Y = 4 ;
X = 4,
Y = 1 ;
X = 4,
Y = 2 ;
X = 4,
Y = 3 ;
X = Y, Y = 4 ;
X = Y, Y = 4.

?- stench(r(X,Y)).
X = Y, Y = 1 ;
X = 1,
Y = 3 ;
X = Y, Y = 2 ;
false.

?- safeWumpus().
false.
?- wumpusAlive().
false.

?- hasArrow().
false.

?- setHunterAt().
Please enter the X coordinate of the hunter:
|: 1.
Please enter the Y coordinate of the hunter:
|: 4.
Please enter the initial direction of the wumpus
|: e.

true.

?- grabGold().
true.

?- gold(r(X,Y)).
false.
```

2. Configuration 2:

Hunter in position (4 , 4).
Breeze in : (1 , 1), (2 , 2), (3 , 1), (3 , 3), (4 , 2).
Pit in : (2 , 1), (3 , 2).
Gold in position (1 , 1).
Wumpus in position (2 , 3).
Stench in : (1 , 3), (2 , 2), (2 , 4), (3 , 3).

=> Screens of the functions being executed in the SWI-Prolog CLI:

```
?- start.
Please enter the X coordinate of the hunter:
|: 4.
Please enter the Y coordinate of the hunter:
|: 4.
Please enter the initial direction of the wumpus
|: n.

true.

?- hunterAt(r(X,Y),Z).
X = Y, Y = 4,
Z = n.

?- wumpus(r(X,Y)).
X = 3,
Y = 2.

?- pit(r(X,Y)).
X = 1,
Y = 2 ;
X = 2,
Y = 3.

?- golf(r(X,Y)).
Correct to: "gold(r(X,Y))"?
Please answer 'y' or 'n'? yes
X = Y, Y = 1.

?- breeze(r(X,Y)).
X = Y, Y = 1 ;
X = 1,
Y = 3 ;
X = 1,
Y = 3 ;
X = Y, Y = 2 ;
X = Y, Y = 2 ;
X = 2,
Y = 4 ;
X = Y, Y = 3 ;
false.

?- stench(r(X,Y)).
X = Y, Y = 2 ;
X = 3,
Y = 1 ;
X = Y, Y = 3 ;
X = 4,
Y = 2 ;
false.

?- setHunterAt().
Please enter the X coordinate of the hunter:
|: 1.
Please enter the Y coordinate of the hunter:
|: 1.
Please enter the initial direction of the wumpus
|: e.

true.

?- grabGold().
true.

?- gold(r(X,Y)).
false.

?- setHunterAt().
Please enter the X coordinate of the hunter:
|: 2.
Please enter the Y coordinate of the hunter:
|: 2.
Please enter the initial direction of the wumpus
|: w.

?- hunterAt(r(X,Y),Z).
X = Y, Y = 2,
Z = w.

?- turnRight().
true.

?- hunterAt(r(X,Y),Z).
X = Y, Y = 2,
Z = n.

?- turnRight().
true .

?- hunterAt(r(X,Y),Z).
X = Y, Y = 2,
Z = e.

?- shootWumpusWithDirection().
true .

?- wumpusAlive().
false.

?- hasArrow().
false.
```

=> In this setting, I tested practically every function. The new function in this case was turnRight (). The wumpus was facing west this time, so we rotated them right twice to face east (the Wumpus).

3. Configuration 3:

Hunter in position (2 , 2).
Breeze in : (1 , 1), (1 , 2), (1 , 3), (1 , 4), (2 , 2), (2 , 3), (2 , 4), (3 , 1), (3 , 3), (4 , 2).
Pit in : (1 , 2), (1 , 3), (1 , 4), (3 , 2).
Gold in position (3 , 4).
Wumpus in position (3 , 3).
Stench in : (2 , 3), (3 , 2), (3 , 4), (4 , 3).
Safe in : : (2 , 1), (2 , 3).

=> Screens of the functions being executed in the SWI-Prolog CLI:

```
?- start.
Please enter the X coordinate of the hunter:
|: 2.
Please enter the Y coordinate of the hunter:
|: 2.
Please enter the initial direction of the wumpus
|: e.

true.

?- hunterAt(r(X,Y),Z).
X = Y, Y = 2,
Z = e.

?- wampus(r(X,Y)).
Correct to: "wumpus(r(X,Y))"?
Please answer 'y' or 'n'? yes
X = Y, Y = 3.

?-
|    pit(r(X,Y)).
X = 2,
Y = 1 ;
X = 2,
Y = 3 ;
X = 3,
Y = 1 ;
X = 4,
Y = 1.

?- gold(r(X,Y)).
X = 4,
Y = 3.

?- breeze(r(X,Y)).
X = Y, Y = 1 ;
X = 1,
Y = 3 ;
X = 2,
Y = 1 ;
X = Y, Y = 2 ;
X = Y, Y = 2 ;
X = 2,
Y = 4 ;
X = 3,
Y = 1 ;
X = 3,
Y = 1 ;
X = 3,
Y = 2 ;
X = Y, Y = 3 ;
X = 4,
Y = 1 ;
X = 4,
Y = 2 ;
false.
```

```
?- stench(r(X,Y)).
X = 2,
Y = 3 ;
X = 3,
Y = 2 ;
X = 3,
Y = 4 ;
X = 4,
Y = 3 ;
false.

?- wumpusAlive().
true.

?- safeWumpus().
true.

?- hasArrow().
true.

?- safe().
Room at X: 1, Y:2 is safe
Room at X: 3, Y:2 is safe
true.
```

=>We start a setup by executing start, and then we set the hunter's location to (2,2) gazing east. We then look at the wumpus, pits, gold, and winds to see where they are. We can observe that the breeze locations are correct (as mentioneed in the configuration above). However, there are winds in places where there are pits, which is unneeded. Then we check to see if the Wumpus is still alive, and the answer is yes. Furthermore, the wumpus is safe since the hunter is not nearby, and because the wumpus is still alive, the hunter still has their arrow. The function safe printed all of the safe rooms (1, 2) and (3, 2), which are empty.

4. Configuration 4:

Hunter in position (2 , 4).
Pit in : (2 , 1), (3 , 1), (3 , 4).
Breeze in : (1 , 1), (2 , 1), (2 , 2), (2 , 4), (3 , 3), (3 , 2), (3 , 1), (4 , 1), (4 , 4).
Gold in position (1 , 2).
Wumpus in position (2 , 2).
Stench in : (1 , 2), (2 , 1), (2 , 3), (3 , 2).
Safe in : : (1 , 4), (2 , 3).

=> Screens of the functions being executed in the SWI-Prolog CLI:

```
?- start.
Please enter the X coordinate of the hunter:
|: 4.
Please enter the Y coordinate of the hunter:
|: 2.
Please enter the initial direction of the wumpus
|: n.

true.
```

```
?- hunterAt(r(X,Y),Z).
X = 4,
Y = 2,
Z = n.

?- wampus(r(X,Y)).
Correct to: "wumpus(r(X,Y))"? yes
X = Y, Y = 2.

?-
|    wumpus(r(X,Y)).
X = Y, Y = 2.

?- pit(r(X,Y)).
X = 1,
Y = 2 ;
X = 1,
Y = 3 ;
X = 4,
Y = 3.

?- gold(r(X,Y)).
X = 2,
Y = 1.

?- breeze(r(X,Y)).
X = Y, Y = 1 ;
X = 1,
Y = 2 ;
X = 1,
Y = 3 ;
X = 1,
Y = 4 ;
X = Y, Y = 2 ;
X = 2,
Y = 3 ;
X = Y, Y = 3 ;
X = 4,
Y = 2 ;
X = Y, Y = 4.

?- stench(r(X,Y)).
X = 1,
Y = 2 ;
X = 2,
Y = 1 ;
X = 2,
Y = 3 ;
X = 3,
Y = 2 ;
false.
```

```
?- setHunterAt().
Please enter the X coordinate of the hunter:
|: 2.
Please enter the Y coordinate of the hunter:
|: 3.
Please enter the initial direction of the wumpus
|: n.

true.

?- safeWumpus().
false.

?- shootWumpus().
true.

?- wumpusAlive().
false.

?- hasArrow().
false.

?- gold(r(X,Y)).
X = 2,
Y = 1.

?- setHunterAt().
Please enter the X coordinate of the hunter:
|: 2.
Please enter the Y coordinate of the hunter:
|: 1
|: .
Please enter the initial direction of the wumpus
|: n.

true.
?- grabGold().
true.

?- gold(r(X,Y)).
false.
```

=>The new functions utilized in this configuration testing are shootWumpus () and grabGold (). To shoot the wumpus, we had to move the hunter to a location close to the wumpus. As a result, we invoked the method safeWumpus, which returned false. We then fire the wumpus to see whether it is still alive and if the hunter still has their arrow. We then shifted the hunter's location to that of the gold in order to seize it.

5. Configuration 5:

Hunter in position (3 , 3).
Pit in : (1 , 1), (2 , 3), (4 , 1).
Breeze in : (1 , 2), (1 , 3), (2 , 1), (2 , 2), (2 , 4), (3 , 1), (3 , 3), (4 , 2).
Gold in position (3 , 2).
Wumpus in position (2 , 1).
Stench in : (1 , 1), (2 , 2), (3 , 1).

=> Screens of the functions being executed in the SWI-Prolog CLI:

```
|    start.
Please enter the X coordinate of the hunter:
|: 3.
Please enter the Y coordinate of the hunter:
|: 3.
Please enter the initial direction of the wumpus
|: e.

true.

?- hunterAt(r(X,Y),Z).
X = Y, Y = 3,
Z = e.

?- wumpus(r(X,Y)).
X = 1,
Y = 2.
?- pit(r(X,Y)).
X = Y, Y = 1 ;
X = 1,
Y = 4 ;
X = 3,
Y = 2.

?- pit(r(X,Y)).
X = Y, Y = 1 ;
X = 1,
Y = 4 ;
X = 3,
Y = 2.

?- gold(r(X,Y)).
X = 2,
Y = 3.

?- breeze(r(X,Y)).
X = 1,
Y = 2 ;
X = 1,
Y = 3 ;
X = 2,
Y = 1 ;
X = Y, Y = 2 ;
X = 2,
Y = 4 ;
X = 3,
Y = 1 ;
X = Y, Y = 3 ;
X = 4,
Y = 2 ;
false.

?- stench(r(X,Y)).
X = Y, Y = 1 ;
X = 1,
Y = 3 ;
X = Y, Y = 2 ;
false.

?- safeWumpus().
true
```

```
?- setHunterAt().
Please enter the X coordinate of the hunter:
|: 2.
Please enter the Y coordinate of the hunter:
|: 2.
Please enter the initial direction of the wumpus
|: e.

true.
```

```
?- shootWumpusWithDirection().
false.

?- turnLeft().
true .

?- hunterAt(r(X,Y),Z).
X = Y, Y = 2,
Z = n.

?- turnLeft().
true .

?- hunterAt(r(X,Y),Z).
X = Y, Y = 2,
Z = w.

?- shootWumpusWithDirection().
true .

?- wumpusAlive().
false.

?- hasArrow().
false.

?-
```

=>We tested the function shootWumpusWithDirection with this setup (). This function doesn't need only that the hunter be close to the wumpus, but also that the hunter confronts the latter. The hunter was in position (2, 2) and couldn't kill the Wumpus in position (1, 2) since the hunter was looking east and the Wumpus has to be facing west to be killed. As a result, we used turnLeft() twice to guide the hunter to the west. Indeed, when we used shootWumpusWithDirection (to kill the Wumpus, we were successful. However, we can observe that our code performs poorly since we have already fired one arrow, i.e., we do not have an arrow remaining to shoot the Wumpus.

IV. Performance rate :

To determine the game's performance, I will primarily measure the performance of four functions: safe, grabGold, shootWumpus, and shootWumpusWithDirection. The computations are summarized in the table below:

	Config 1.	Config 2.	Config 3.	Config 4.	Config 5.	Average
safe	1	1	1	1	1	1
grabGold	1	1	_	1	1	1
shootWumpus	0	_	_	1	_	0.5
shootWumpusWithDirection	_	1	_	1	0	0.67
Average	0.67	1	1	1	0.67	0.87

Comment : We can see that the bot routines shootWumpus and shootWumusWithDirection provide incorrect results in some cases when the hunter shot before killing the wumpus. Our code makes the incorrect assumption that the hunter has limitless arrows.

*V. **Limitations :***

=>Limitation: the very first limitation of the proposed solution is that the hunter has unlimited arrows, which should not be true according to the rules of the game.

Resolution: while initializing the game, we can include the predicate hasArrow(), and retract it from the KB after the first shoot attempt.

=>Limitation: we percept breeze and stench although we know it contains a pit.

Resolution: Adding a condition of no pit, \+pit(r (X, Y)), if a breeze is detected.