CMPE480 – 2021-2022 Fall – Homework 2

In this project, you will implement the logical rules for the domain below, and verify them using Prolog.

You will submit a single file: rules.pl

Your program will be evaluated using by

- concatanating rules.pl and experience.pl in kb.pl
- loading kb.pl into swipl
- with isClear() and isWall() in Q1
- with hasNotWumpus() and hasNotPit() in Q2
- with hasWumpus(), hasPit() and hasDeadWumpus() in Q3
- with hasFood() and hasNotFood() in Q4

Some sample experience.pl files and required answers to sample queries will be provided.

Introduction to the problem

The environment includes:

- Agent can locate in a cell, can face to one of the four directions,
- One or more wumpus,
- One or more pits,
- One or more foods,
- Walls enclose the environment. Any square might also include a wall.

Directions:

- North, East, South, West

Actions:

- Forward: Move forward one grid in the current direction
- Eat: Eat the food in the current grid
- ClockWise: Turn clockwise
- CounterClockWise: Turn counter-clockwise
- Attack: Move forward one grid and kill the wumpus if there is an alive wumpus in the new grid.

Environment Model:

- Wumpus, food, and pit create smell, smell and breeze in the 4-neighbourhood, respectively.
- Wumpus, food, and pit does not create smell, smell and breeze in their own grid.
- If agent steps in a grid with a pit or an alive wumpus, it dies.
- If agent steps in a grid with a dead wumpus, nothing happens.
- Food and its smell disappear after food being eaten.
- Wumpus and its smell does not disappear after wumpus being killed.
- If agent hits the wall as a result of forward or attack actions, "bump" is sensed.
- If agent successfully eats the food, "full" is sensed.

In this project, you need to implement the following predicates (depending on Q1-3):

- isClear(T,X,Y).
- hasFood(T,X,Y).
- hasNotFood(T,X,Y).
- hasWumpus(T,X,Y).
- hasNotWumpus(T,X,Y).
- hasPit(T,X,Y).
- hasNotPit(T,X,Y).

- hasDeadWumpus(T,X,Y).

isClear(T,X,Y) is already partially implemented for you.

- X corresponds to the row index (starting from top, starting from 0)
- Y corresponds to the column index (starting from left, starting from 0)

Q1: Bump no more!
Improve isWall predicate and update isClear if necessary, so that the agent does not bump to the already bumped wall.

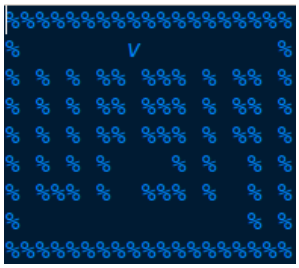
Q2: Avoid dangers!
Implement hasNotWumpus and hasNotPit predicates. Your agent should avoid all the dangers.

Q3: Make sure dangers and kill enemies
Implement hasWumpus, hasPit and hasDeadWumpus predicates. Your agent should avoid all the dangers, make sure the location of the enemies and pits when possible, and therefore be able to kill the enemies.

Q4: Eat food
Implement hasFood and hasNotFood predicates. Your agent should be able to locate the grid with food (when possible), step in and eat the food.

Example runs for Q1:

Assume the following environment unknown to the agent [%: wall, top-left:(0,0), V: location and direction of the agent]



experience.pl:location(1,1,8).
dir(1,south).
action(1,forward).
action(2,forward).
action(3,forward).
action(4,counterClockWise).
action(5,forward).
bump(6).

```
?- isWall(4,9).
true.

?- isWall(4,7).
false.

?- isWall(1,1).
false.

?- isWall(6,6).
false.
```

experience.pl:

location(1,1,8).dir(1,south).
action(1,forward).
action(2,forward).
action(3,forward).
action(4,counterClockWise).
action(5,forward).
bump(6).
action(6,clockWise).
action(7,clockWise).
action(8,forward).
bump(9).

```
?- isWall(4,9).
true.

?- isWall(4,7).
true.

?- isWall(1,1).
false.

?- isWall(6,6).
false.
```

experience.pl:

location(1,1,8).dir(1,south).action(1,clockW
action(2,forward).
action(3,forward).
action(4,forward).
action(5,forward).
action(6,forward).
action(7,forward).
action(8,counterClockWise).
action(9,forward).
bump(10).
action(10,counterClockWise).
action(11,forward).
action(12,clockWise).
action(13,forward).
action(14,forward).
action(15,forward).
action(16,forward).
action(17,forward).
bump(18).

```
?- isWall(2,2).
true.

?- isWall(7,3).
false.

?- isWall(8,3).
false.

?- isWall(6,3).
true.

?- isWall(4,7).
false.

?- isWall(4,9).
false.
```

Example run for Q2:

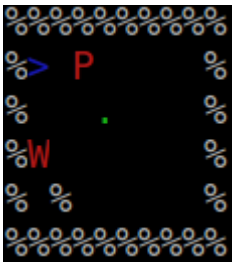
Assume the following environment unknown to the agent [%: wall, top-left:(0,0), V: location and direction of the agent, P: pit, W: wumpus]

experience.pl:

location(1,4,2).dir(1,north).
action(1,forward).
action(2,forward).
action(3,forward).
action(4,counterClockWise).
action(5,forward).
action(6,counterClockWise).
action(7,forward).
action(8,forward).
action(9,forward).
action(10,counterClockWise).
action(11,forward).
action(12,forward).
action(13,forward).
action(14,forward).
action(15,forward).
action(16,forward).
action(17,forward).
action(18,forward).
action(19,counterClockWise).
action(20,forward).
action(21,counterClockWise).
action(22,forward).
pitBreeze(10).
pitBreeze(11).
bump(19).
wumpusSmell(21).
wumpusSmell(22).

```
?- hasNotWumpus(1,5,2).
true.

?- hasNotWumpus(1,6,2).
false.

?- hasNotPit(1,4,1).
true.

?- hasNotPit(1,5,1).
false.

?- hasNotWumpus(1,6,6).
false.

?- hasNotWumpus(1,2,8).
false.

?- hasNotPit(1,2,8).
true.

?- hasNotPit(1,8,1).
false.
```

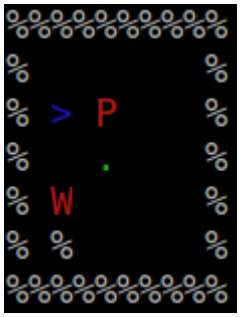
Example run for Q2&Q3:
Assume the following environment unknown to the agent [%: wall, top-left:(0,0), V: location and direction of the agent, P: pit, W: wumpus, green-dot: food]



experience.pl:

location(1,1,1).
dir(1,east).
action(1,forward).
action(2,counterClockWise).
action(3,counterClockWise).
action(4,forward).
action(5,counterClockWise).
action(6,forward).
pitBreeze(2).
pitBreeze(3).
pitBreeze(4).
WumpusSmell(7).

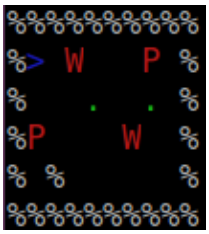
```
?- hasPit(1,1,2).
false.

?- hasNotPit(1,1,2).
true.

?- hasPit(1,1,3).
true.

?- hasNotPit(1,1,3).
false.

?- hasWumpus(1,2,1).
false.

?- hasNotWumpus(1,2,1).
true.

?- hasWumpus(1,3,1).
true.

?- hasNotWumpus(1,3,1).
false.
```



experience.pl:

location(1,2,2).
dir(1,east).
action(1,forward).
action(2,counterClockWise).
action(3,counterClockWise).
action(4,forward).
action(5,counterClockWise).
action(6,forward).
pitBreeze(2).
pitBreeze(3).
pitBreeze(4).
wumpusSmell(7).

```
?- hasPit(1,2,3).
false.

?- hasNotPit(1,2,3).
true.

?- hasPit(1,2,4).
false.

?- hasNotPit(1,2,4).
false.

?- hasWumpus(1,3,2).
false.

?- hasNotWumpus(1,3,2).
true.

?- hasWumpus(1,4,2).
false.

?- hasNotWumpus(1,4,2).
false.
```



experience.pl:

```
location(1,1,1).
dir(1,east).
action(1,forward).
action(2,counterClockWise).
action(3,counterClockWise).
action(4,forward).
action(5,counterClockWise).
action(6,forward).
action(7,counterClockWise).
action(8,forward).
action(9,forward).
action(10,counterClockWise).
action(11,attack).
wumpusSmell(2).
wumpusSmell(3).
wumpusSmell(4).
pitBreeze(7).
pitBreeze(8).
foodSmell(10).
wumpusSmell(10).
foodSmell(11).
wumpusSmell(11).
```

```
?- hasDeadWumpus(11,1,3).
true.

?- hasDeadWumpus(1,1,3).
true.
```

Example run for Q4:

Assume the following environment unknown to the agent [%: wall, top-left:(0,0), V: location and direction of the agent, P: pit, W: wumpus, green-dot: food]

experience.pl:

```
location(1,1,1).
dir(1,east).
foodSmell(1).
```

```
?- hasFood(1,1,2).
false.

?- hasNotFood(1,1,2).
false.
```

experience.pl:

```
location(1,1,1).
dir(1,east).
action(1,forward).
action(2,forward).
action(3,clockWise).
action(4,clockWise).
action(5,forward).
action(6,counterClockWise).
foodSmell(1).
bump(3).
foodSmell(6).
foodSmell(7).
```

```
?- hasFood(1,1,2).
false.

?- hasNotFood(1,1,2).
false.
```

experience.pl:

```
location(1,1,1).
dir(1,east).
action(1,forward).
action(2,forward).
action(3,clockWise).
action(4,clockWise).
action(5,forward).
action(6,counterClockWise).
action(7,forward).
foodSmell(1).
bump(3).
foodSmell(6).
foodSmell(7).
bump(8).
foodSmell(8).
```

```
?- hasFood(1,1,2).
true.

?- hasNotFood(1,1,2).
false.
```

experience.pl:

```
location(1,1,1).
dir(1,east).
action(1,forward).
action(2,forward).
action(3,clockWise).
action(4,clockWise).
action(5,eat).
action(6,forward).
action(7,counterClockWise).
action(8,forward).
foodSmell(1).
bump(3).
bump(9).
```

```
?- hasFood(8,1,2).
false.

?- hasNotFood(8,1,2).
true.
```

experience.pl:

```
location(1,1,1).
dir(1,east).
action(1,forward).
action(2,forward).
action(3,clockWise).
action(4,clockWise).
action(5,eat).
foodSmell(1).
bump(3).
```

```
?- hasFood(1,1,2).
false.

?- hasNotFood(1,1,2).
false.
```