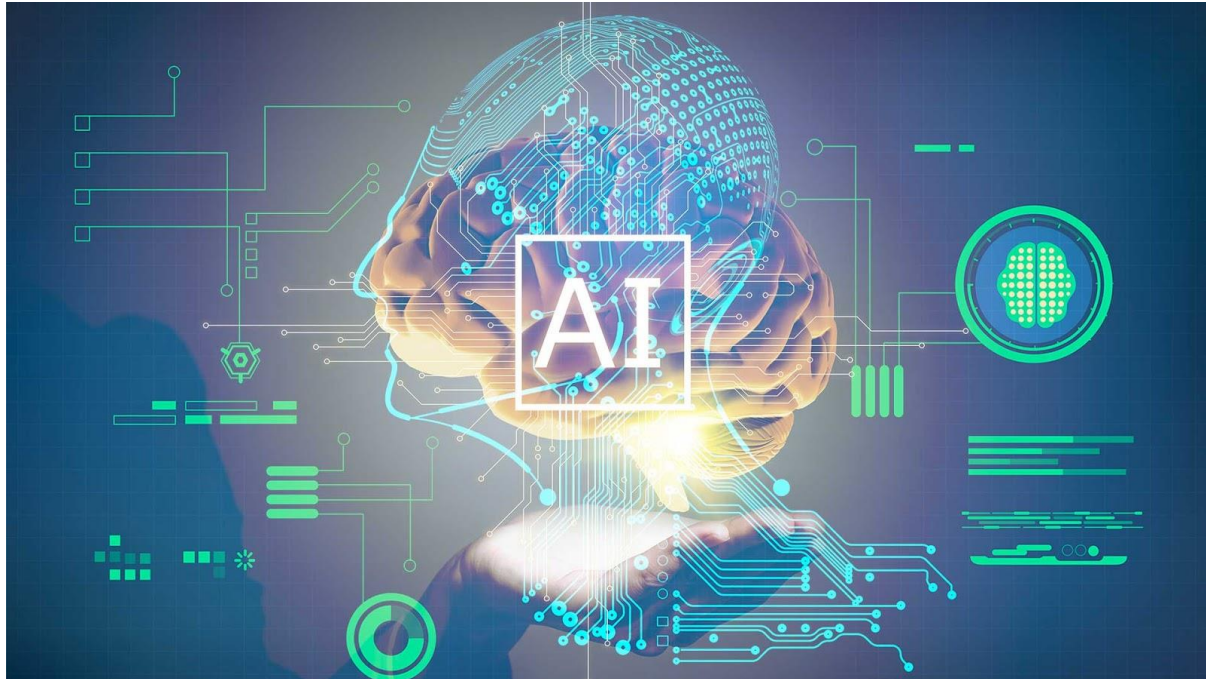


# CSEN 901: Artificial Intelligence

## Project (2) Report



Name : Hesham Sherif

ID : 34-5518

Name : Karim Walid El-Khafif

ID : 34-5263

Name : Ali Mohamed Hany

ID : 34-2035

## Problem Description:

A grid of size  $N \times M$  that consists of  $N$  rows and  $M$  columns is given. The grid contains cells which can be Jon's, or an obstacle, or a dragon stone where Jon can obtain dragon glass from to kill white walkers, or a white walker or lastly an empty cell. Jon is initially at the lower right corner of the grid so for example if the grid is of size (3,3) the Jon will be initially at position (2,2). At any point in the game Jon is able to do one of 6 actions, he can move on empty cells in one of the 4 directions (up, down, left, right) or he can take dragon glass from the dragon stone provided that he is standing on it. In addition, he can attack any white walker if it is adjacent to him provided, he has dragon glass. The goal of this project is to implement a logical agent who is capable of planning a sequence of actions to kill all the white walkers in the grid using the knowledge base (which is the initial representation of the grid) and the successor state axioms.

## GenGrid Modifications:

To accommodate for the change of using a logical agent instead of search agent, a modification was needed in the GenGrid function in java so it can be able to generate logical statements the agent is able to understand.

The grid generation was the same as project 1 but after it finishes a function was called which was responsible for looping over the grid represented in java and it generates logical statements from the grid into a file.

The grid representation in java is as follows

```
whi, emp, emp, whi,  
emp, obs, obs, whi,  
emp, dra, emp, obs,  
obs, obs, emp, Jon,
```

emp: empty cell

whi: white walker cell

obs: obstacle cell

Jon: Jon cell

dra: Dragon Stone cell

The method which generates this grid is the same as project 1

```
public void genGrid() throws IOException {
    Random r = new Random();
    //Generate a grid with random number of rows and cols not less than 4 and not greater than 10
    int randomX = r.nextInt(3) + 3;
    int randomY = r.nextInt(3) + 3;
    //Random number of dragons glass between 1 and 10 a dragon stone can provide at a time
    maxDragonGlass = r.nextInt(10) + 2;
    this.positionI = randomX - 1;
    this.positionJ = randomY - 1;
    grid = new Cell[randomX][randomY];
    //Initialize Jon Snow position
    grid[this.positionI][this.positionJ] = new Cell("JonSnow", 0);
    //Get a random position for the dragon stone not the same as Jon Snow position
    while(true) {
        int randomDragonX = r.nextInt(randomX);
        int randomDragonY = r.nextInt(randomY);
        if(randomDragonX != positionI && randomDragonY != positionJ) {
            grid[randomDragonX][randomDragonY] = new Cell("dragonStone", maxDragonGlass);
            positionIDragon = randomDragonX;
            positionJDragon = randomDragonY;
            break;
        }
    }
    //Generate the rest of the grid
    for (int i = 0; i < grid.length; i++) {
        for (int j = 0; j < grid[i].length; j++) {
            if(grid[i][j] == null) {
                int randomSpawn = r.nextInt(3);
                grid[i][j] = new Cell(generateCell(randomSpawn), 0);
            }
        }
    }
}
```

The method creates a random grid with random positions of dragon stone, white walkers and empty cells, and puts Jon into the lower right cell.

## Generating logical statements:

After generation of a random grid in java using the data structure of Cell. We call a function called generatePrologFile() which loops over the generated grid cell by cell and produce logical statements into a prolog file in addition to putting logical statements for things that are not represented in the 2d array such as the maximum number of dragon glass, the grid size and the initial number of dragon glass Jon is holding.

```

public void generatePrologFile() throws IOException {
    FileOutputStream outputStream = new FileOutputStream("facts.pl");
    String file = "";
    file += "gridSize(" + grid.length + ", " + grid[0].length + ").\n";
    file += "maxDragon(" + maxDragonGlass + ").\n";
    file += "currentDragonInit(" + currentDragonGlass + ").\n";
    for (int i = 0; i < grid.length; i++) {
        for (int j = 0; j < grid[0].length; j++) {

            if(grid[i][j].getType().equals("obstacle")) {
                file += "obstacle(" + i + ", " + j + ").\n";
            }
            else if(grid[i][j].getType().equals("whiteWalker")) {
                file += "whiteWalker(" + i + ", " + j + ").\n";
            }
            else if(grid[i][j].getType().equals("empty")) {
                file += "emptyInit(" + i + ", " + j + ").\n";
            }
            else if(grid[i][j].getType().equals("JonSnow")){
                file += "jonInit(" + i + ", " + j + ").\n";
                file += "emptyInit(" + i + ", " + j + ").\n";
            }
            else {
                file += "dragonStone(" + i + ", " + j + ").\n";
                file += "emptyInit(" + i + ", " + j + ").\n";
            }
        }
    }

    byte[] strToBytes = file.getBytes();
    outputStream.write(strToBytes);
    outputStream.close();
}

```

## How to generate a grid:

You just need to go to genGrid class in the java project and go to the main method.

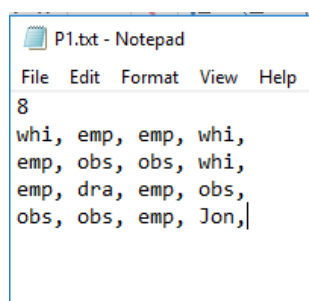
There are 2 options to generate a grid whether to generate a random grid or to generate a grid from a .txt file which is a user input.

To generate the logical statements of random grid, go to the main method and comment the try and catch statements and uncomment the g.genGrid() line.

```
public static void main(String[] args) throws IOException {
    genGrid g = new genGrid();
    g.deleteFile();
    // try {
    //     g.genGrid2("p1.txt");
    // } catch (Exception e) {
    //     System.out.println("File not Found");
    // }
    g.genGrid();
    g.printGrid(g.grid);
    g.generatePrologFile();
}
```

If you would like to generate logical statements from a .txt file just uncomment the try and catch statements and put the name of the file inside the g.genGrid2() which should be located in the root directory of the java project and comment the g.genGrid() line.

```
public static void main(String[] args) throws IOException {
    genGrid g = new genGrid();
    g.deleteFile();
    try {
        g.genGrid2("p1.txt");
    } catch (Exception e) {
        System.out.println("File not Found");
    }
    // g.genGrid();
    g.printGrid(g.grid);
    g.generatePrologFile();
}
```



```
P1.txt - Notepad
File Edit Format View Help
8
whi, emp, emp, whi,
emp, obs, obs, whi,
emp, dra, emp, obs,
obs, obs, emp, Jon,
```

Make sure if you are going to input your own created grid to make the .txt file in this format, where the number at the beginning represents the maximum number of dragon glass Jon can obtain from the dragon Stone.

You can find 2 examples for generating a prolog file from java existing in the root directory of the java project named facts1.pl and facts2.pl

Here is an example of a generated prolog file

### Example 1:

<code>gridSize(3, 3).</code>	Which corresponds to this grid
<code>maxDragon(8).</code>	
<code>currentDragonInit(0).</code>	Max Dragon:8
<code>emptyInit(0, 0).</code>	
<code>whiteWalker(0, 1).</code>	Current Dragon:0
<code>emptyInit(0, 2).</code>	
<code>emptyInit(1, 0).</code>	Grid size: 3x3
<code>dragonStone(1, 1).</code>	
<code>emptyInit(1, 1).</code>	emp, whi, emp
<code>obstacle(1, 2).</code>	
<code>whiteWalker(2, 0).</code>	emp, dra, obs,
<code>emptyInit(2, 1).</code>	
<code>jonInit(2, 2).</code>	whi, emp, Jon,
<code>emptyInit(2, 2).</code>	

### Example 2:

<code>1 gridSize(4, 4).</code>	Which corresponds to this grid
<code>2 maxDragon(2).</code>	
<code>3 currentDragonInit(0).</code>	Max Dragon: 2
<code>4 emptyInit(0, 0).</code>	
<code>5 whiteWalker(0, 1).</code>	
<code>6 whiteWalker(0, 2).</code>	Grid Size: 4x4
<code>7 obstacle(0, 3).</code>	
<code>8 obstacle(1, 0).</code>	Current Dragon: 0
<code>9 whiteWalker(1, 1).</code>	
<code>0 emptyInit(1, 2).</code>	
<code>1 whiteWalker(1, 3).</code>	Emp, whi, whi, obs,
<code>2 dragonStone(2, 0).</code>	
<code>3 emptyInit(2, 0).</code>	obs, whi, emp, whi,
<code>4 emptyInit(2, 1).</code>	
<code>5 whiteWalker(2, 2).</code>	dra, emp, whi, obs,
<code>6 obstacle(2, 3).</code>	
<code>7 emptyInit(3, 0).</code>	emp, emp, emp, Jon,
<code>8 emptyInit(3, 1).</code>	
<code>9 emptyInit(3, 2).</code>	
<code>0 jonInit(3, 3).</code>	
<code>1 emptyInit(3, 3).</code>	

## Predicates and Actions

### Predicates from the generated Prolog file from java (KnowledgeBase)

- gridSize(X, Y): where X is the number of rows and Y is the number of columns.
- maxDragon(D): where D is the maximum number of dragon glass Jon can obtain from the dragon stone.
- currentDragonInit(D): where D is the initial number of dragon glass Jon is holding
- whiteWalker(X, Y): which specifies that there is a white walker in row X and column Y.
- emptyInit(X, Y): which specifies that row X and column Y is an empty cell. Note: - Jon's Cell and the dragon Stone are considered empty cells as well.
- dragonStone(X, Y): specifies that at row X and column Y is the dragon Stone.
- jonInit(X, Y): specifies that the initial position of Jon is at row X , column Y.

### Actions in the logical agent program

#### action(name, DX, DY):

Actions are represented in the agent's program as facts where name is (up, down, left, right, take, attack) and DX and DY are the values that should be added to each coordinate value of the current position to get the new position of Jon.

```
%Actions%
action(attack, 0, 0).
action(take, 0, 0).
action(up, -1, 0).
action(right, 0, 1).
action(down, 1, 0).
action(left, 0, -1).
```



## Predicates in the logical agent program:

- `goal_test(S)`: which is true if all the white walkers in the grid have a corresponding empty cell with the same location in state `S` and there is no goal state before state `S`.

```
% Goal_test checks that in State S
% all white walkers are dead
goal_test(S):-
    foreach(whiteWalker(X, Y), (empty(X, Y, S))), noGoalStateBefore(S).
```

- `noGoalStateBefore(S)`: which is true if there is not goal state before State `S`.

```
% Checks that there doesn't
% occur any goal before State S
noGoalStateBefore(s0).
noGoalStateBefore(result(_, S)):-
    \+ goal_test(S),
    noGoalStateBefore(S).
```

- `checkValidMove(X, Y, DX, DY)`: which is true if the action Jon is trying to do is inside the grid.

```
%Checks that Jon can move to a certain cell
checkValidMove(X, Y, DX, DY):-
    gridSize(I, J),
    P is X + DX,
    P2 is Y + DY,
    P >= 0, P < I,
    P2 >= 0, P2 < J.
```

- `checkValidAttack(X, Y, W, Z)`: which is true if Jon is in a position adjacent to a white walker positioned at row X, column Y and inside the grid.

```
%Checks jon is able to attack a white Walker
```

```
checkValidAttack(X, Y, W, Z):-
    gridSize(I, J),
    P is W+1,
    P2 is W-1,
    P3 is Z-1,
    P4 is Z+1,
    (
        (X = P, Y = Z, X>=0, X<I);
        (X = P2, Y = Z, X>=0, X<I);
        (X = W, Y = P3, P3>=0, P3<J);
        (X = W, Y = P4, P4>=0, P4<J)
    ).
```

## Successor state Axioms:

### Empty Axiom:

`empty(X, Y, result(A, S))`: is true and only true if

- A = attack and at State S, there was a white walker in position (X, Y) and it was not empty and Jon was in a position adjacent to this white walker and he had more than zero dragon glass.
- At State S, location (X, Y) it was already empty.

### Jon Axiom:

`jon(X, Y, result(A, S))`: is true and only true if

- At State S, location (X, Y) is empty and doing action A is a valid move (inside the grid).

## currentDragon Axiom:

currentDragon(D, result(A, S)): is true and only true if

- A = take and D = maximum number of dragon glass and at state S Jon is standing over the dragon Stone cell and he doesn't have the maximum number of dragon glass.
- A = attack and D = (number of dragon glass in state S) - 1  
and at State S Jon was located in a cell adjacent to a white walker and it was not empty and he had more than zero dragon glass
- A is not equal to attack and A is not equal to take and D = number of dragon glass in state S.

## How to run the query:

In order to run the program, you need to put the grid file (Knowledge Base) from the java project in the same directory as run.pl file and you have to rename the grid file to facts.pl

To run the query, run the file run.pl using prolog and call the predicate plan(). Which can be done by typing in the prolog prompt plan()., The output will be a sequence of actions to reach a goal state.

```
Use :- disjointness emptyInit/2. to suppress this message
Warning: c:/users/hesham sherif/desktop/new folder/facts.pl:16:
Clauses of obstacle/2 are not together in the source-file
Earlier definition at c:/users/hesham sherif/desktop/new folder/facts.pl:
Current predicate: emptyInit/2
Use :- disjointness obstacle/2. to suppress this message
Warning: c:/users/hesham sherif/desktop/new folder/facts.pl:19:
Clauses of emptyInit/2 are not together in the source-file
Earlier definition at c:/users/hesham sherif/desktop/new folder/facts.pl:
Current predicate: obstacle/2
Use :- disjointness emptyInit/2. to suppress this message
Warning: c:/users/hesham sherif/desktop/new folder/facts.pl:21:
Clauses of emptyInit/2 are not together in the source-file
Earlier definition at c:/users/hesham sherif/desktop/new folder/facts.pl:
Current predicate: jonInit/2
Use :- disjointness emptyInit/2. to suppress this message
Welcome to SWI-Prolog (threaded, 64 bits, version 7.6.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit http://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- plan().
```

## 2 Examples:

### Example 1:

Max Dragon: 3

whi, obs, whi,  
whi, dra, emp,  
obs, obs, Jon,

Output:

```
?- plan().  
[up, left, take, attack, left, attack, right, right, attack]  
true ■
```

---

### Example 2:

Max Dragon: 4

obs, emp, whi,  
whi, emp, dra,  
whi, whi, Jon,

Output:

```
?- plan().  
[up, take, attack, left, attack, down, attack]  
true ■
```

### Example 3:

Max Dragon: 1

whi, emp, whi,  
whi, dra, emp,  
emp, whi, Jon,

Output:

```
?- plan().  
[up, left, take, attack, take, up, attack]  
true ■
```

---