Project 2: ∃a∃s[Escaped(R2D2,Result(a,s))]

Report

Ahmed Ashraf Nazih 31-10219
Yasmeen Mohammed Wafa 31-1008
Maha Ehab Elleci 31-11852

Brief description of the problem :

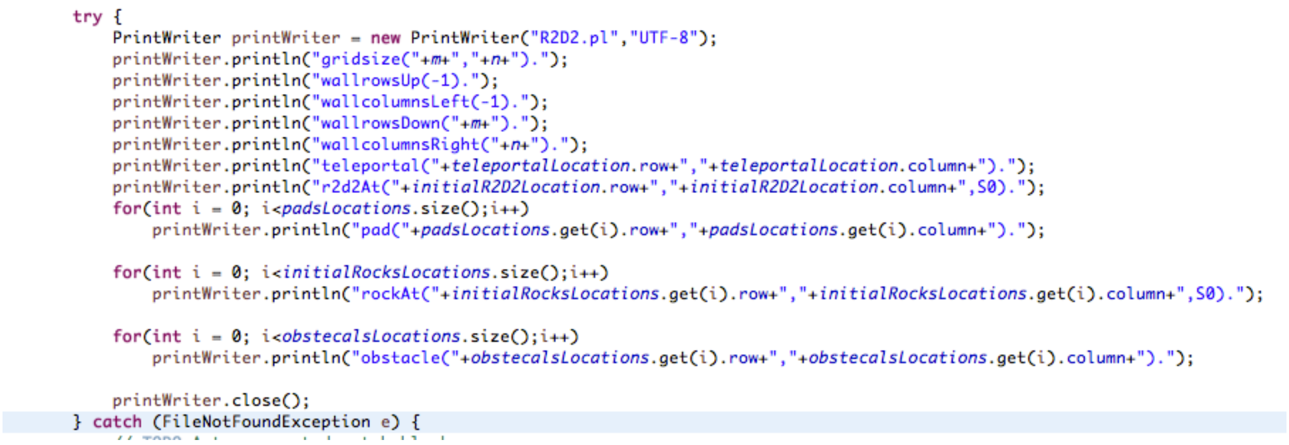
The problem is to specify the required sequence of actions the agent R2D2 should follow in order to reach the goal. The agent reaches the goal when every rock is pushed over every pressure pad and the agent itself reaches the teleportal location. The problem is represented and solved in a logic-based version by defining the successor state axioms for the two main fluents in the problem:

1. *r2d2At(X,Y,S)* :- which represents the location coordinates of the agent R2D2 in the grid in a situation S.

2. *rockAt(X,Y,S)* :- which represents the location coordinates of a certain rock in the grid in a situation S.

The modified version of GenGrid :

The *GenGrid* function in Project 1 implemented in Java was modified so that it generates the knowledge base of this logic-based version with the initial configuration of the grid. This modification included adding a *PrintWritter* instance that writes the grid's configuration -grid size, rocks initial locations, pads locations, agent initial location-, to a prolog file with valid syntax.



```
try {
    PrintWriter printWriter = new PrintWriter("R2D2.pl","UTF-8");
    printWriter.println("gridsize("+m+","+n+").");
    printWriter.println("wallrowsUp(-1).");
    printWriter.println("wallcolumnsLeft(-1).");
    printWriter.println("wallrowsDown("+m+").");
    printWriter.println("wallcolumnsRight("+n+").");
    printWriter.println("teleportal("+teleportalLocation.row+","+teleportalLocation.column+").");
    printWriter.println("r2d2At("+initialR2D2Location.row+","+initialR2D2Location.column+",S0).");
    for(int i = 0; i<padsLocations.size();i++)
        printWriter.println("pad("+padsLocations.get(i).row+","+padsLocations.get(i).column+").");

    for(int i = 0; i<initialRocksLocations.size();i++)
        printWriter.println("rockAt("+initialRocksLocations.get(i).row+","+initialRocksLocations.get(i).column+",S0).");

    for(int i = 0; i<obstecalsLocations.size();i++)
        printWriter.println("obstacle("+obstecalsLocations.get(i).row+","+obstecalsLocations.get(i).column+").");

    printWriter.close();
} catch (FileNotFoundException e) {
```

Running examples :

In this project, two grids configurations were added to the knowledge base for testing.

Grid 1 :

gridsize(3,3).
wallrowsUp(-1).
wallcolumnsLeft(-1).
wallrowsDown(3).
wallcolumnsRight(3).
teleportal(2,1).
r2d2At(1,2,s0).
pad(1,0).
rockAt(1,1,s0).
obstacle(0,0).

Obstacle		
Pad	Rock	R2D2
	Teleportal	

Grid 2 :

gridsize(4,3).
wallrowsUp(-1).
wallcolumnsLeft(-1).
wallrowsDown(4).
wallcolumnsRight(3).
teleportal(2,2).
r2d2At(0,1,s0).
pad(3,1).
rockAt(1,1,s0).
obstacle(0,0).

Obstacle	R2D2	
	Rock	
		Teleportal
	Pad	

Syntax and semantics of the action terms and predicates :

1. Knowledge Base Predicates :

- gridSize(X,Y) : which denotes the the size of the grid where x is the number of rows and y is the number of columns.
- *wallrowsUp(U)* : Which denotes the position upper wall of the grid.
- *wallcolumnsLeft(L)* : Which denotes the position left wall of the grid.
- *wallrowsDown(D)* : Which denotes the position lower wall of the grid.
- *wallcolumnsRight(R)* : Which denotes the position right wall of the grid.
- *teleportal(X,Y)* : which denotes the the position of the teleportal, where X is the row position and Y is the column position of the teleportal.
- *r2d2At(X,Y,s0)* : which denotes the the position of the r2d2, where X is the row position , Y is the column position of the r2d2 and the term s0 is the initial situation of r2d2 .
- *rockAt(X,Y,s0)* : which denotes the the position of the rock, where X is the row position , Y is the column position of the rock and the term s0 is the initial situation of the rock .
- *pad(X,Y)* : which denotes the the position of the pad, where X is the row position,Y is the column position of the pad.
- *obstacle(X,Y)* : which denotes the the position of the obstacle, where X is the row position , Y is the column position of the obstacle.

3. Predicates used for the axioms :

- *r2d2At(X,Y,result(A,S))* : where X and Y denote the position of r2d2 in the current situation S, and this situation is the result of performing the action A (moving east, west, north or south) .
- *rockAt(X,Y,result(A,S))* : where X and Y denote the position of the rock in the current situation S, and this situation is the result of performing the action A (moving east, west, north or south).

Description of the query :

The query is defined as follows for the first grid defined in the knowledge base :

call_with_depth_limit((r2d2At(2,1,S),rockAt(1,0,S)),4,C).

where, *r2d2At(2,1,S)* is the target situation where the agent reached the teleportal location *(2,1).*
rockAt(1,0,S) is the target situation where the rock is pushed in the pressure pad located at *(1,0).*

the result is the sequence of actions required to reach both situations stated in the query above :

S = result(south, result(west, s0)).

Implementation of the successor-state axioms :

The successor-state axiom is defined by the effect axioms and the frame axioms according to the problem.
In our model, 3 predicates were implemented for the successor-state axioms :

- the effect of the agent's movement *(r2d2At(X,Y,S))* (there is no obstacle or wall exactly next to the agent in the direction of movement),
- the effect axiom for the movement of the rock *(rockAt(X,Y,S)),*
- the persistence state of the rock where the action fails to change the rock's location in the grid (there is an obstacle or wall or another rock exactly next to the rock in the direction of movement) .