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

Ahmed Kamel (31-15431, T14), Mostafa Abdullah (31-15451, T14)

1. Introduction

Successor state axioms can be used to decide whether a certain predicate holds or not in a certain situation. A successor state axiom of a certain predicate is defined by the state of one or more predicates in the predecessor situation. From the successor state axioms, one can generate a plan for a search problem, by going from the goal situation, and following the predecessor situations until the initial situation (that we will denote as S_0) is encountered. In this project, the goal is to generate a plan for the search problem defined in Project 1, by writing the successor state axioms for the location of the R2D2 robot and the rocks, since they are the only predicates that might change in different situations. (The obstacles, the teleport and the pressure pads cannot move in any successor situtation). Prolog is chosen for encoding these axioms and for the the search plan generation.

2. GenGrid

Like in Project 1, the initial grid is randomly generated with random locations of the robot, the teleport, the obstacels, the rocks and the pads. The GenGrid method defined in Project 1 is modified to encode the grid definition in the form of Prolog facts. These facts are then written in a *knowledge base* file, which is loaded by the prolog file responsible for defining the successor state axioms. These facts act as the base cases (the initial situation S_0) for different predicates. The syntax for these facts and the successor state axioms are explained in the next section.

3. Syntax and Semantics

3.1. Actions

There are four possible actions that the robot can make, each corresponding a step in one of the main directions: up, down, left and right. Each action is represented by a unique integer corresponding its direction as follows:

- 1 → Up
- 2 → Down

- 3 → Left
- 4 → Right

So an action is defined by the following grammar:
$Action = 1 \,|\, 2 \,|\, 3 \,|\, 4$

3.2. Situations

A certain situation is defined by the following grammar:
$Situation = S_0 \,|\, result(Action, Situation)$
Where $result(a, s)$ means the resulting situation from applying action a on situation s.

3.3. Predicates

The following predicates are employed in our implementation:

- robot(X, Y, S). True iff the robot is in position (X, Y) in the grid in situation S.
- rock(X, Y, S). True iff a rock is in position (X, Y) in the grid in situation S.
- pad(X, Y). True iff a pressure pad is in position (X, Y) in the grid in any situation.
- obstacle(X, Y). True iff on obstacle is in position (X, Y) in the grid in any situation.
- teleport(X, Y). True iff the teleport is in position (X, Y) in the grid in any situation.
- height(H). True iff H is the height of the grid.
- width(W). True iff W is the width of the grid.
- inside_grid (X, Y). True iff location (X, Y) lies within the grid boundaries
- empty(X, Y). True iff location (X, Y) lies within the grid borders and is not occupied by any blocking object (a rock, an obstacle or a teleport).
- rocks_on_pads(S). True iff S is a situation in which every rock lies on a pressure pad.
- query(S). True iff S is the goal situation, that is goal(S) is true and the the robot is in the same position as the teleport.
- dx(Idx, Dx). True iff $[1, -1, 0, 0][\text{Idx}] = \text{Dx}$. Used for choosing one vertical direction to move to.
- dy(Idx, Dy). True iff $[0, 0, -1, -1][\text{Idx}] = \text{Dy}$. Used for choosing one horizontal direction to move to.
- element(I, L, V). True iff $\text{L}[\text{I}] = \text{V}$.

4. Successor State Axioms

Only two predicates have successor state axioms: $me(X, Y, S)$, $rock(X, Y, S)$. Because they are the only states that might be affected by any of the possible actions.

$$\begin{aligned} robot(X, Y, result(A, S)) \equiv \\ dx(A, Dx) \wedge dy(A, Dy) \wedge robot(X + Dx, Y + Dy, S) \\ \wedge (empty(X, Y, S) \vee \\ rock(X, Y, S) \wedge \neg rock(X, Y, result(A, S) \vee \\ teleport(X, Y) \wedge goal(S)) \end{aligned}$$

The robot will be in position (X, Y) in situation $result(A, S)$ iff it existed in position $(X + dx[A], Y + dy[A])$ in S and it satisfies any of the following conditions:

- The current position (X, Y) was empty in situation S
- A rock existed in position (X, Y) in S, however it no longer exists in the same position in $result(A, S)$. This axiom is determined by the successor state axiom of the rock.
- The teleport is in position (X, Y) and the situation S has all rocks correctly placed on pressure pads.

Where $dx = [1, -1, 0, 0], dy = [0, 0, 1, -1]$.
Note that we assume that the robot must change its position in every action. That is it cannot run into an immovable object or attempt to jump out of the grid.

$$\begin{aligned} rock(X, Y, result(A, S)) \equiv \\ dx(A, Dx) \wedge dy(A, dy) \wedge [(rock(X, Y, S) \wedge (\neg robot(X + Dx, Y + Dy, S) \vee \\ \neg inside\_grid(X + Dx, Y + Dy) \vee \\ rock(X - Dx, Y - Dy) \vee \\ \neg inside\_grid(X - Dx, Y - Dy) \\ \vee \\ (robot(X + 2 * Dx, Y + 2 * Dy, S) \wedge \\ rock(X + Dx, Y + Dy, S) \wedge empty(X, Y, S))] \end{aligned}$$

The first part of the disjunction describes the successor state axiom if a rock did not change its position. It is the case if there is no robot in position $(X + dx[A], Y + dy[A])$, the rock lies on the border or the next cell contained an immovable object.

The second part describes the axiom if the rock was in a different position than the current one. In this case, all the following conjuncts must be satisfied in situation S:

- The robot was in position $(X + 2 * dx[A], Y + 2 * dy[A])$.
- There was a rock in position $(X + dx[A], Y + dy[A])$.
- The cell in position (X, Y) was empty.

5. Plan Generation Query

The plan generation takes place if the following two conjuncts are satisfied:

- The robot is in the same position as the teleport: $robot(X, Y) \wedge teleport(X, Y)$
- All rocks lie on pressure pads: $\forall X, Y : pad(X, Y) \implies rock(X, Y)$.

6. Experimentation

The map for the grid visualization is as follows:

M	Robot Position
*	Rock
#	Obstacle
_	Pressure Pad
|	Teleport

6.1. Example 1

```
M*_
...
.|#
```

The facts written initially to the knowledge base are as follows:

```
height(3).
width(3).
robot(1, 1, s0).

rock(1, 2, s0).

pad(1, 3).

obstacle(3, 3).

teleport(3, 2).
```

The generated plan is the following:

```
result(2, result(2, result(4, s0)))
```

Referring back to the actions syntax in Section 3.1, the above maps to:

```
Right -> Down -> Down -> Down
```

6.2. Example 2

```
M....
.....
..#..
...*.
.....
..._|
```

The facts written initially to the knowledge base are as follows:

```
height(6).
width(5).
robot(1, 1, s0).

rock(4, 4, s0).

pad(6, 4, s0).

obstacle(3, 3).

teleport(6, 5).
```

The generated plan is the following:

```
result(2, result(4, result(2, result(2, result(2, result
    (4, result(4, result(4, result(2, s0)))))))))
```

The above plan maps to:

```
Down -> Right -> Right -> Right -> Down -> Down -> Down
    -> Right -> Down
```