Artificial Intelligence

Project 2 (The Matrix) – Report

Team (7)

Team Members:

1. Ziad Amr 43-5548 (T-17)
2. Mohamed Ashraf 43-10871 (T-17)
3. Muhammad Ehab 43-7667 (T-17)
4. Hesham Abdulhamid 43-1684 (T-17)

- *A discussion of the syntax and semantics of the **action terms** and **predicate symbols** you employ.*

Initial State

- The initial state is represented by the following prolog predicate:

```
neo(X, Y, C, Hostages, s0).
```

This predicate uses the values provided in the knowledge base to assign the initial values of the Neo's position, the maximum capacity and the positions of the hostages.

Actions Applicability

- The following predicate is used to decide if a move is applicable or not:

```
move_applicability(A, X, Y).
```

Based on the action ***(A)*** and Neo's current position ***(X, Y)***, the predicate verifies if this action is applicable or not. This predicate is used in the predicate representing the successor state axiom to make sure that the movement that will result in the next situation is valid.

- The following predicate is used to decide if a carry action is applicable or not:

carry_applicability(X, Y, C, Hostages).

Based on Neo's current position ***(X, Y)***, the remaining capacity ***(C)*** and the positions of the remaining hostages to be carried ***(Hostages)***, this predicate verifies if the carry action is applicable or not. This predicate is used in the predicate representing the successor state axiom to make sure that the carry action that will result in the next situation is valid.

- The following predicate is used to decide if a drop action is applicable or not:

drop_applicability(X, Y, C).

Based on Neo's current position ***(X, Y)*** and the remaining capacity ***(C)***, this predicate verifies if the drop action is applicable or not. This predicate is used in the predicate representing the successor state axiom to make sure that the carry action that will result in the next situation is valid.

Goal Test

- The goal of Neo is represented in the following predicate:

```
goal_test(S).
```

This predicate decides whether situation ***(S)*** is the situation needed by Neo to reach his goal. The predicate check if Neo is in the telephone booth's location and if all hostages have been carried and dropped at the telephone booth. If so, Neo has finally reached his goal.

IDS

- Since prolog uses DFS to implement backward chaining, the program may run forever. The following predicate prevents that from happening:

```
ids(S, D).
```

This predicate implements the Iterative Deepening Search to find a goal situation ***(S)*** using a maximum search depth ***(D)*** using the predefined ***call_with_depth_limit*** predicate. If it fails to find a goal situation, the maximum depth is incremented, and the predicate is called again with the incremented depth to perform the search again till a goal situation is reached.

- *A discussion of your implementation of the **successor-state axioms**.*

In this project, we used only one successor state axiom which is presented as the following prolog predicate:

neo(X, Y, C, Hostages, result(A,S)).

This predicate represents the current state of our agent. It keeps track of Neo's current position ***(X, Y)***, the remaining capacity ***(C)*** and the position of the remaining hostages to be carried ***(Hostages)***. This predicate results in a new situation when applying action ***(A)*** on the previous situation ***(S)***. Neo's action in each situation that leads him to a new situation is one of (*right*, *left*, *up*, *down*, *carry*, *drop*). The prolog predicate, using inference mechanism tries to reach the goal by moving on the grid, carrying all the hostages and dropping them in the telephone booth.

- *A description of the predicate **goal(S)** used to query the KB to generate the plan.*

The following predicate is used to start the search of a goal situation:

```
goal(S).
```

This predicate first checks if situation ***(S)*** is a variable or not. If it is a variable, so a goal situation is needed to be found and ids(S, D) predicate is called to do so. If it is not a variable, goal_test(S) predicate is called, and back chaining is performed to decide whether this situation is a valid one or not.

- *At least **two running examples** from your implementation.*

These are some running examples based on the same KB but using different queries than the ones provided in the project description.

1. Example 1:
    - goal(S).
    - *S = result(drop, result(up, result(carry, result(down, result(drop, result(right, result(up, result(carry, result(down, result(right, s0))))))))).*

2. Example 2:
    - goal(result(drop, result(up, result(carry, result(down, result(drop, result(right, result(up, result(carry, result(right, result(down, s0))))))))))).
    - **true.**

3. Example 3:
    - goal(result(drop, result(left, result(carry, result(down, result(drop, result(right, result(up, result(carry, result(right, result(down, s0))))))))))).
    - **false.**