Artificial Intelligence for Big Data Systems

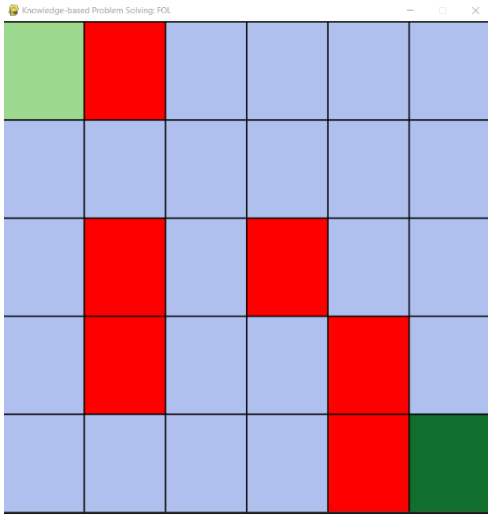
Assignment 2: Knowledge-Based Problem Solving

(Logical Theory Report)

Stefan Velev, 0MI3400521

Introduction

We are given the problem of finding a path between the starting node (0,0) and the end node (4,5) of a maze. The permitted moves are four: Left, Right, Up and Down. The red nodes contain obstacles and no movements are possible through them. The boundaries are walls which are blocking the movements.



We have to represent the maze problem using First-Order Logic (FOL) and find the path from the starting node to the end node of the maze using knowledge base containing that representation of the knowledge and applying logical inference.

Task 1: Specify the facts for modelling the maze using FOL in the format of ground formulas

The only facts that I state explicitly in the solution describe the direct connection between cells. Each fact specifies a direct path between two cells which can be modelled with the predicate: DirectPath(X, Y) where X and Y are two neighbouring cells. With that representation we can notice that there is no need for the boundaries to be described with separate facts since the direct path facts are enough to tell where the logical agent can go to and where it would be impossible.

```
DirectPath(c00, c10)
DirectPath(c10, c11)
DirectPath(c10, c20)
DirectPath(c20, c30)
DirectPath(c30, c40)
DirectPath(c40, c41)
DirectPath(c41, c42)
DirectPath(c42, c43)
DirectPath(c42, c32)
DirectPath(c43, c33)
DirectPath(c33, c32)
```

```
DirectPath(c32, c22)
DirectPath(c22, c12)
DirectPath(c11, c12)
DirectPath(c12, c02)
DirectPath(c12, c13)
DirectPath(c02, c03)
DirectPath(c13, c03)
DirectPath(c13, c14)
DirectPath(c03, c04)
DirectPath(c04, c05)
DirectPath(c04, c14)
DirectPath(c14, c15)
DirectPath(c14, c24)
DirectPath(c24, c25)
DirectPath(c05, c15)
DirectPath(c15, c25)
DirectPath(c25, c35)
DirectPath(c35, c45)
```

Task 2: Formulate heuristics using FOL in the format of rules with universally quantified variables

In the previous task I specified all the possible neighbouring paths. However, they are bidirectional. That means that if we can go from cell $\{0, 0\}$ to cell $\{1, 0\}$ we must be able to go from cell $\{1, 0\}$ to cell $\{0, 0\}$ as well. That must be applied for all the direct paths. With the help of the first-order logic, we do not have to write these facts specifically. We can use some heuristics that will help us with clearer solution. They can be expressed as rules involving predicates. For the problem above, we can formulate the following rules for movement:

A cell Y is reachable from cell X if there is a direct path between X and Y:

$$\forall X\ \forall Y\ (\text{DirectPath}(X, Y) \Rightarrow \text{Move}(X, Y))$$

A cell Y is reachable from cell X if there is a direct path between Y and X:

$$\forall X\ \forall Y\ (\text{DirectPath}(Y, X) \Rightarrow \text{Move}(X, Y))$$

Task 3: Choose suitable method for logical inference and suitable inference rules for automation of the inference in the logical theory of FOL

For the solution of the problem, I use backward chaining which is entirely based on representing the goal in a query. The recursive pathfinding logic is proposed with the following rules:

Base case – a path exists between two cells if there is a direct connection between them:

$$\forall S\ \forall F\ (\text{Move}(S, F) \Rightarrow \text{Path}(S, F))$$

Recursive case – if S connects to an intermediate cell Z (not the final destination) and Z is not visited (visited nodes are represented in the Visit list), extend the path:

$$\forall S\ \forall F\ \forall Z\ \forall \text{Visit}\ (\text{Move}(S, Z) \land Z \neq F \land \neg\text{Member}(Z, \text{Visit}) \land \text{Path}(Z, F)^{[1]} \Rightarrow \text{Path}(S, F))^{[2]}$$

Task 4: Formulate the success criteria for solving the problem as a goal statement and the starting position in FOL

[1] From the second iteration on we add S and all other explored nodes to the Visit list.
[2] This rule is not shown entirely. Full details are presented and can be seen in the program report.

The goal is to find a path from the starting node (c00) to the end node (c45). This can be represented in the language of first-order logic as follows:

We will find every Solution such that Solution is a valid path from the starting node (c00) to the end node (c45).

$$\forall Solution\ (Path(c00, c45) \Rightarrow FinalPath(Solution))^3$$

We use the following query to find a solution:

$$\text{FinalPath(Solution)}$$

Task 5: Demonstrate logical inference in the logical theory using the chosen inference method and rules of inference to find the solution

We use backward chaining algorithm for the query $\text{FinalPath(Solution)}$. With the above rule, we search for a path between c00 (starting node) and c45 (end node). As there is not a direct path between these cells, we turn to the recursive case where we look for Z such that Move(S, Z) is already present. From the first fact we know that there is a direct path between c00 and c10. Cell c10 is not visited and it is not the end node. Therefore, our next step is to calculate Path(c10, c45). The arguments that we continue with are similar. There is not a direct path between c10 and c45 (end node) so we search for another intermediate node and so on. Alternative inference rule is forward chaining where we start with known facts and apply rules iteratively until the goal is reached when we terminate. This approach is data-driven whereas the backward chaining algorithm is goal-driven.

[3] This rule is not shown in full detail as Solution is dependent on the result of the Path predicate. Full details are presented and can be seen in the program report.