AUTOMAT Machine Project

Machine Project Report - AUTOMAT

August 22, 2018

Manzano, Joshua
Student
2401 Taft Ave, Malate,
Manila, 1004 Metro Manila
+639178949025
josh_manzano@dlsu.edu.ph

Mendoza, Emir
Student
2401 Taft Ave, Malate,
Manila, 1004 Metro Manila
+639173962844
emir_mendoza@dlsu.edu.ph

Valencia, Josh
Student
2401 Taft Ave, Malate,
Manila, 1004 Metro Manila
+639064726584
josh_cezar_valencia@dlsu.edu.ph

ABSTRACT

The purpose of this paper is to produce and analyze a deterministic finite automaton devised to show the possible ways to solve the problem. A program is designed to simulate the problem.

KEYWORDS

Automata, State, Transition

1 INTRODUCTION

In this paper, a puzzle is given which can be analyzed and solved through the use of a finite state machine. The puzzle goes accordingly:

"The Earth is slowly dying and a scientist wants to help build a new life on Mars. However, he only has a spaceship that was so tiny it could only carry him and two other items. He wants to transport five things that he thought were essential to start a new life on Mars: two humans, one lion, one cow and one bag of grain. Whenever the scientist is not around, either human would kill the lion out of fear or eat the cow out of hunger, the lion will eat the cow, and the cow will eat the grain. Only the scientist knows how to fly the spaceship. How will the scientist transport all the five items to Mars? "

The task is to produce a Deterministic Finite Automaton (DFA) that would represent each possible state that the rocket ship can be in and to create a simulation wherein a player can provide the inputs for which item they would like to transport using the rocketship.

2 PROBLEM

The problem is determining how the puzzle and every possible move the player can make can be represented through a DFA. It's important to determine what can be considered a state, what can be considered transitions and what the final state would look like.

Aside from providing the DFA and the simulation, the group was tasked to also provide the minimum number of moves that a player can make to reach the destination or the accepting state of the DFA and show the possible paths that use said minimum number of moves using an algorithm that is left to the group to implement. a

3 IMPLEMENTATION

The program accepts the input of two letters selected from the choices (H, H, L, C, G). Which corresponds to Human, Human, Lion, Cow and Grain accordingly. The inputs are required by the transition class. The transition class also requires its destination. The transitions that are allowed with respect to the devised DFA are then added to each state. The final state is reached when the scientist manages to transport all 5 items to Mars.

The Automaton class contains all the possible states, including the initial and final state. It is called by the Interface class and SolutionFinder class whenever simulation of the automaton is needed.

The Deterministic Finite Automata that was constructed is defined as follows:

$M = (Q, \Sigma, q_0, F, \delta)$ where

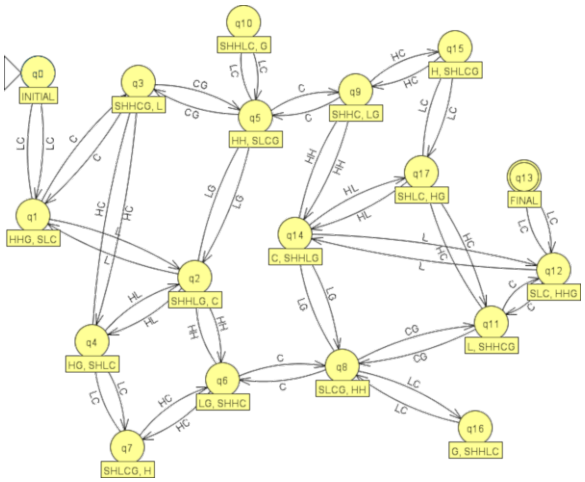
$Q = \{q0, q1, q2, q3, q4, q5, q6, q7, q8, q9, q10, q11, q12, q13, q14, q15, q16, q17\}$

$\Sigma = \{H, L, C, G, HH, HL, HC, HG, LC, LG, CG\}$

q_0 = {q0}

F = {q17}

State Diagram:



The State class contains a state number (Integer), a state label (String), and an ArrayList of Transitions.

The Transition class contains an input (String), and a destination (State).

The SolutionFinder class makes use of the Automaton class to apply a recursive algorithm to find all possible paths which lead to the final state.

The rest of the classes are used to implement the Graphical User Interface.

For simulating the game, validation of the input is done through checking all of the possible transitions of the current state. If a transition with that input is found, then the current state is changed to the destination provided by the transition. If no transition is found, then the input is assumed to be invalid and the game is restarted since the automaton dies. This continues until the current state equates to the final state.

4 SOLUTION

The solution utilizes a recursive algorithm wherein it iterates through all possible paths going to the final state. The algorithm starts with the initial state, and branches out all possible transitions from there. If the current branch reaches the final state, then it is added to the solutions. If the current branch is a dead-end, then it simply stops.

The implementation is similar to how a Non-deterministic Finite Automata works, in which all possible choices are explored. The key difference is that the algorithm keeps track of the previously visited states of the current branch. If one of the transitions contains a destination that was already visited before, then it is ignored. This prevents infinite-loops and redundant solutions.

The shortest solutions are then found by sorting all of the solutions and picking the ones where the minimum amount of transitions are used.

5 CONCLUSION

The minimum amount of transitions needed to finish the game is seven, and there are four possible paths which use this amount of transitions.

6 REFERENCES

[1] https://en.wikipedia.org/wiki/Deterministic_finite_automaton