CS410 - Automata Theory and Formal Languages
-
Project 1 Design Report

Introduction

This project aims to create a program that converts Nondeterministic Finite Automaton to Deterministic Finite Automaton. The program will get NFA as an input with the help of a pre-designed .txt file and it will print equivalent DFA with the same design.

Problem Definition

Since all NFA's have an equivalent DFA, it is possible to convert every NFA input to DFA. In this project the epsilon transition will be ignored. **Basically, a function which checks every transition and converts clusters of states to single states and also defines the new transitions for new states is the thing we really need.**

Tools

JAVA : Programming language for implementation
Eclipse: Integrated development environment for JAVA

Usage of The program

To use the java application of NFA to DFA Converter follow these steps:

- Compile all files in the directory by following command:
  javac *.java
- Run the program with giving your path to input file as an argument:
  java Selami_Karakas_S018705 <Path to Input File>
  Example: java Selami_Karakas_S018705 /home/selami/CS410/NFA1.txt
  Note: Please be sure that your path does not include any whitespaces.

Method

To create the program following structures and functions needed:

- A structure that describes finite automata.
- A structure that describes states with its transitions.
- A structure that describes symbols of the alphabet.
- A function that takes inputs and returns corresponding nondeterministic finite automaton object.
- A function that takes a nondeterministic finite automaton and returns the deterministic equivalent of it. (The steps of this function are explained in the next chapter.)
- A function that takes an automaton and prints its design.

Implementation Of Automaton Structure

To create the structure of Automaton those attributes will be described:

- An ArrayList for states.
- A State for start state.

- An ArrayList for final states.
- An ArrayList for Symbols of the alphabet.

Implementation Of State Structure

To create the structure of Automaton those attributes will be described:

- A String for the name of the state.
- A HashMap<Symbol, ArrayList<State>> structure for transitions.

Implementation Steps Of Conversion Function

In this section the conversion function is explained step by step with an example. The input example is at the end of this report.

This function returns a DFA and accepts an NFA as a parameter.

- Function initialize an empty array list for states.
- First function takes the start state of NFA and its possible transitions and defines them for new DFA.
  Ex: A is the start state and its possible transitions are A_0_A and A_ 1_ [A,B].
- Now, in a while loop, the function will check every transition and create new states for DFA. The important thing here is that the function will take state sets as single states.
  Ex: The loop will start with A's transactions. A goes A with 0 so we can add A as a new state but A is already added so we will skip. A goes [A,B] with 1 so we can add [A,B] as a new state. Loop will continue with the next state which is [A,B]. Now the function should define the new state [A,B]'s transactions. To define the function will combine A and B's transactions. The while loop will end when there is no state left to check.
- Final states array will be initialized.
- Function will construct a new DFA with new states, final state or states and previous alphabet.
- Function will return DFA.

Input Output Format

Here is the format mentioned in a few parts. There are examples with explanations.

```
ALPHABET
\\ Each line has a symbol from alphabet
0
1
STATES
\\ Each line has a state
A
B
C
START
A \\ Only one start state will be accepted.
```

FINAL
\\Final state or states can be described here
C
TRANSITIONS
\\ All possible transitions between states
A 0 A
A 1 A
A 1 B
B 1 C
END