CS410 Project 1 - Phase 1
Design Report

İsmail Can Yağmur

October 2022

Introduction

This reports discuss the implementation of NFA to DFA converter written in C++17 with g++ compiler. Finite Automatons written in a specific format in text files are read and processed in design logic.

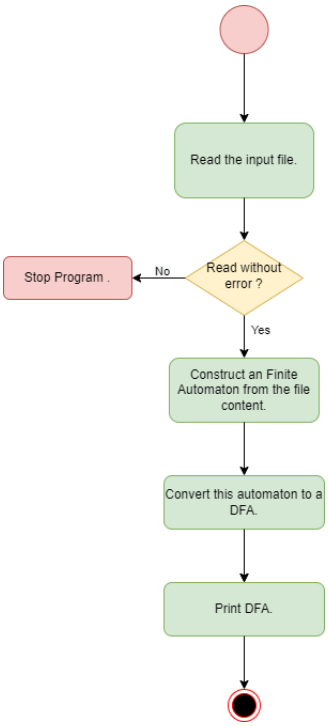
Objectives

The implementation follows below objectives.

1. Read the input file.
2. Construct an Finite Automaton from the file content.
3. Convert this automaton to a DFA.
4. Print the DFA to console in text file format.

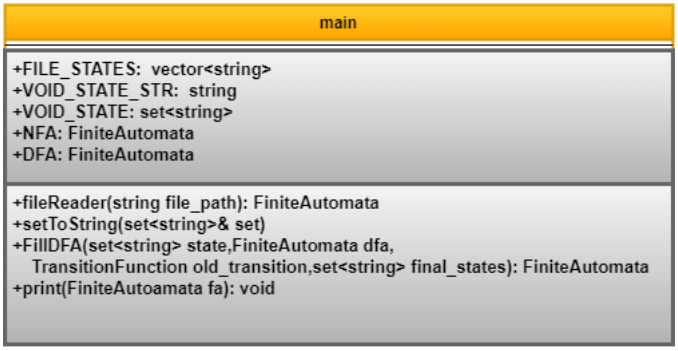
Software Architecture Overview

Activity Diagram



Finite Automaton definition text files are read thanks to **FileReader** method. Each state in the text file representing which definition member of the automaton is going to be given captured thanks to vector of strings containing file states called as **FILE_STATES**. While reading the content of the file, **FiniteAutomata** structure is used to encapsulate the definition members of automaton. This structure gives the blueprint of the the desired automaton whether it is NFA or DFA. These definition members are alphabet, states, start state, final state and transition function. After creating an NFA from the text file, **FillDFA** method is used for conversion. It follows the recursive procedure of creating DFA from NFA. After the conversion, the content of the DFA is displayed on the console in given file format thanks to **print** method.

Class Diagram



This program consists of 1 class and 1 struct. Struct named FiniteAutomata encapsulates the definition members of the finite automatons. Also, it has a boolean variable named isNFA to differentiate the finite automatons. ALPHABET,STATES,START and FINAL members of this struct are unordered sets of strings. TRANSITIONS member variable is unordered map of string and unordered map of string and set of string. In main class, using the FiniteAutomata struct, NFA and DFA variables are initiated. Also, VOID_STATE_STR and VOID_STATE variables are defined to denote the void states in converted DFA.

Recursive Conversion Process

While converting the NFA to DFA, recursive programming is used. The pseudocode of the algorithm is as follows. Initially, given NFA's alphabet and start state assigned to constructed DFA's corresponding fields. After that, recursion is triggered with NFA's initial state.

Algorithm 1 Given NFA is converted to DFA using recursion programming.

```
1: function convertToDFA(state,dfa,nfa_transitions,final_states)
2:     if state is not in dfa.States then
3:         INSERT state to dfa.States
4:         if state is SINK state then
5:             for each alphabet member do
6:                 Add transition to sink state
7:             end for
8:         else
9:             for each alphabet member do
10:                for each alphabet member do
11:                    Iterate over state to get unit states
12:                    if Any unit state is final state then
13:                        Assign main state as final state
14:                    end if
15:                    if Transition exist then
16:                        Add transition to accumulator
17:                    end if
18:                end for
19:                if Accumulated transition set is not empty then
20:                    Add transition to DFA
21:                else
22:                    Add sink transition to DFA
23:                end if
24:                if New transition is not in DFA states then
25:                    if accumulated transition set is not empty then
26:                        recursively call convertToDFA with new transition.
27:                    else
28:                        recursively call convertToDFA with sink state.
29:                    end if
30:                end if
31:            end for
32:        end if
33:    end if
34: end function
```

Results

Here, I tested my algorithm with given text files, NFA1.txt and NFA2.txt. The program takes the path of the text file as an argument and prints the original NFA and converted DFA in the text file format to the console.

NFA	DFA
[INFO]Reading the file : NFA1.txt <<<<<<< NFA >>>>>>> ALPHABET 0 1 STATES A B C START A FINAL C TRANSITIONS A 0 A A 1 A A 1 B B 1 C END	<<<<<<< DFA >>>>>>> ALPHABET 0 1 STATES A AB ABC START A FINAL ABC TRANSITIONS A 0 A A 1 AB AB 0 A AB 1 ABC ABC 0 A ABC 1 ABC END
[INFO]Reading the file : NFA2.txt <<<<<<< NFA >>>>>>> ALPHABET 0 1 STATES A B C START A FINAL C TRANSITIONS A 0 A A 1 B A 1 C B 0 B B 0 C C 0 A C 0 B C 1 B END	<<<<<<< DFA >>>>>>> ALPHABET 0 1 STATES A BC SINK ABC B START A FINAL BC ABC TRANSITIONS A 0 A A 1 BC BC 0 ABC BC 1 B SINK 0 SINK SINK 1 SINK ABC 0 ABC ABC 1 BC B 0 BC B 1 SINK END

Table 1: Table of results