Name: Adarsh Dhakar

Roll No.: 22CS01040

Repository Link: Github Link (https://github.com/adarshdhakar/reToDFA)

Deployed Link of React Simulation: React App (https://uppsskvp.netlify.app)

****The CPP and the web page code is attached in the submission. This is a pdf version of README.**

****More detailed README is added in the zip and uploaded on the Github.**

****React App has the simulation using Javascript, React, and few styling libraries, the code is in the Github repo.**

Regular Expression to DFA Converter

This C++ program provides a command-line tool to convert a given regular expression into an equivalent Deterministic Finite Automaton (DFA). The conversion process involves several key stages of compilation theory:

1. **Parsing and Validation**
2. **Infix to Postfix Conversion**
3. **NFA Construction**
4. **DFA Conversion**

Supported Operators and Alphabet

- **Alphabet**: The user defines the alphabet (e.g., a,b,c) at the start of the program. Any character not in the alphabet or the operator list is considered invalid.
- **Union (+)**: Represents the logical 'OR' operation. For example, a+b matches either 'a' or 'b'.
- **Concatenation (.)**: Represents sequencing. For example, a.b matches 'a' followed by 'b'.
- **Kleene Star (*)**: Represents zero or more occurrences of the preceding element. For example, a* matches ε (empty string), a, aa, etc.
- **Parentheses (())**: Used for grouping to control the order of operations.

How to Compile and Run (C++ version)

The program is written in standard C++ and can be compiled with g++.

1. **Save the Code**

   Save the provided C++ code into a file named regularExpressionToDFA.cpp.

2. **Compile the Code**

   Open your terminal or command prompt and run the following command.

   g++ regularExpressionToDFA.cpp -o re_to_dfa

3. **Run the Executable**

   Execute the compiled program from your terminal.

   ./re_to_dfa

The program will then prompt you to enter the grammar and the regular expression.

C++ program examples

Usage Example 1

```
This program converts a Regular Expression to a DFA.
The following characters are reserved operators: ( ) * + . E

Please enter a grammar (e.g., a,b): a,b,c
Please enter a regular expression (e.g., (a+b)*.a.b.b): a.b.c

Concatenation: a.b.c
Postfix Expression: ab.c.

--- NFA Construction ---
Start Node: 0
End Node: 5

Node no: 0 (Start)
  0 --a--> 1
Node no: 1
  1 --b--> 3
Node no: 3
  3 --c--> 5
Node no: 5 (Final)
  No outgoing edge.

--- DFA Transition Table ---
Start State: 0
Final States: { 3 }

State   |       a       |       b       |       c       |
--------|---------------|---------------|---------------|
 0      |       1       |       -       |       -       |
 1      |       -       |       2       |       -       |
 2      |       -       |       -       |       3       |
*3      |       -       |       -       |       -       |

(* denotes final state)
```

Usage Example 2

```
This program converts a Regular Expression to a DFA.
The following characters are reserved operators: ( ) * + . E

Please enter a grammar (e.g., a,b): a,b
Please enter a regular expression (e.g., (a+b)*.a.b.b): (a+b)*

Concatenation: (a+b)*
Postfix Expression: ab+*

--- NFA Construction ---
Start Node: 6
End Node: 7

Node no: 0
  0 --a--> 1
Node no: 1
  1 --E--> 5
Node no: 2
  2 --b--> 3
Node no: 3
  3 --E--> 5
Node no: 4
  4 --E--> 2
  4 --E--> 0
Node no: 5
  5 --E--> 4
  5 --E--> 7
Node no: 6 (Start)
  6 --E--> 4
  6 --E--> 7
Node no: 7 (Final)
  No outgoing edge.

--- DFA Transition Table ---
Start State: 0
Final States: { 1, 2, 0 }

State   |       a       |       b       |
--------|---------------|---------------|
*0      |       1       |       2       |
*1      |       1       |       2       |
*2      |       1       |       2       |

(* denotes final state)
```

How to Compile and Run (Web Version) -> React App (https://uppsskvp.netlify.app)

The project is designed to be run locally for development or deployed to any static web hosting service, and is the exact translation of the C++ version to JavaScript.

1. **Prerequisites**
   - **Node.js**: Version 14.0 or later.
   - **npm**: Node Package Manager, which is included with Node.js.
2. **Local Setup**
   - Open a terminal or command prompt and clone the source code from the GitHub repository:
     - **git clone https://github.com/adarshdhakar/reToDFA.git**
   - Enter into the newly created project folder using:
     - **cd reToDFA**
   - Run the following command to install all the required libraries and packages:
     - **npm install**
   - Execute the command below to compile the project and start a local web server:
     - **npm run dev**
3. **Access the Application**:
   Open a web browser and navigate to the local address provided in the terminal, which is generally **http://localhost:5173**. The application will be live and ready for use.

Example 1 : (a.b.c)

Regular Expression to DFA Converter

Enter an alphabet and a regular expression to see the generated DFA.

Alphabet (comma-separated): a,b,c

Regular Expression: a.b.c

Generate DFA & Graph

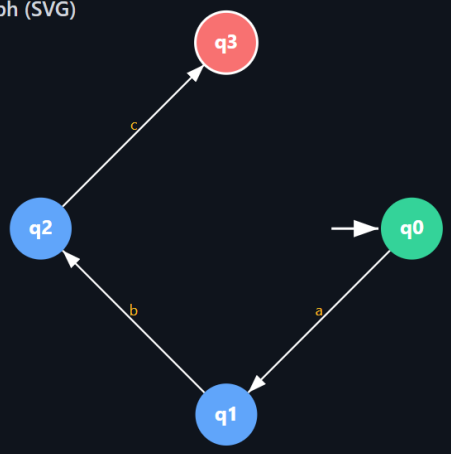
Processing Details

Processed RE: `a.b.c`

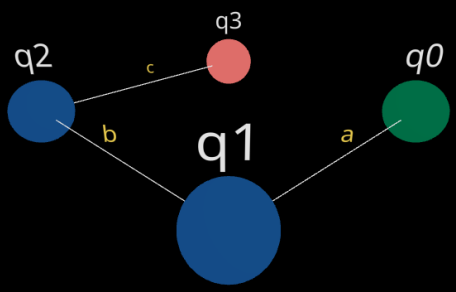
Postfix: `ab.c.`

DFA Graphs

2D Graph (SVG)



3D Graph



DFA Transition Table

Start State: q0

Final States: {q3}

State	Input 'a'	Input 'b'	Input 'c'
→ q0	q1	—	—
q1	—	q2	—
q2	—	—	q3
*q3	—	—	—

Example 2 : (a+b)*

Regular Expression to DFA Converter

Enter an alphabet and a regular expression to see the generated DFA.

Alphabet (comma-separated): a,b

Regular Expression: (a+b)*

Generate DFA & Graph

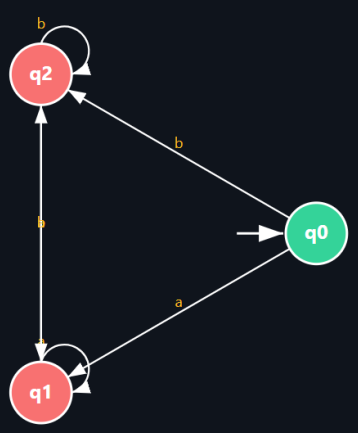
Processing Details

Processed RE: (a+b)*

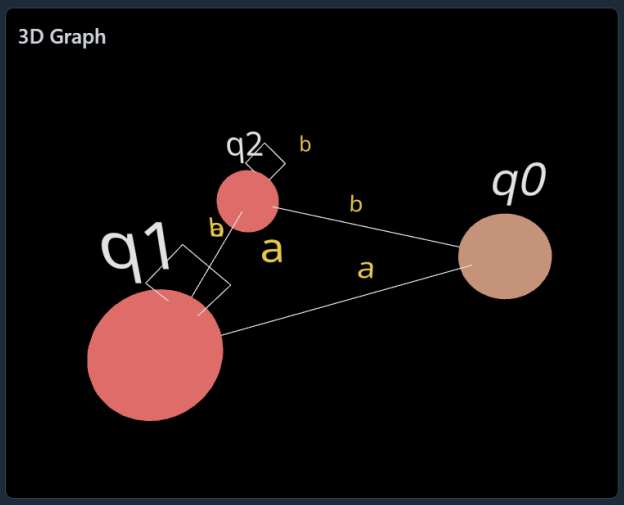
Postfix: ab+*

DFA Graphs

2D Graph (SVG)



3D Graph



DFA Transition Table

Start State: q0

Final States: {q0, q1, q2}

State	Input 'a'	Input 'b'
→ *q0	q1	q2
*q1	q1	q2
*q2	q1	q2