REVISION HISTORY

Date	Version	Description	Author
11/11/2023	1.0	Software Architecture and design description is added.	Ezgi Nur Alışan
12/11/2023	1.1	Error Handling, Testing is written.	Ezgi Nur Alışan
13/11/2023	1.2	The code structure part is added. Necessary regulations are made.	Ezgi Nur Alışan

TABLE OF CONTENTS

1 Introduction

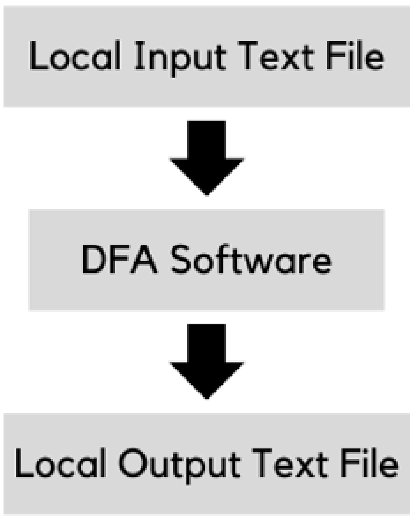
This document describes the design of the Deterministic Finite Automaton (DFA) software system. The purpose of this program is to simulate the DFA for determining whether a given string is accepted or rejected and giving the visited states.

1.1 References

1.1.1 Project References

#	Document Identifier	Document Title
[SDD]	DFA-SDD-2.1	DFA Software Development Design

2 Software Architecture overview



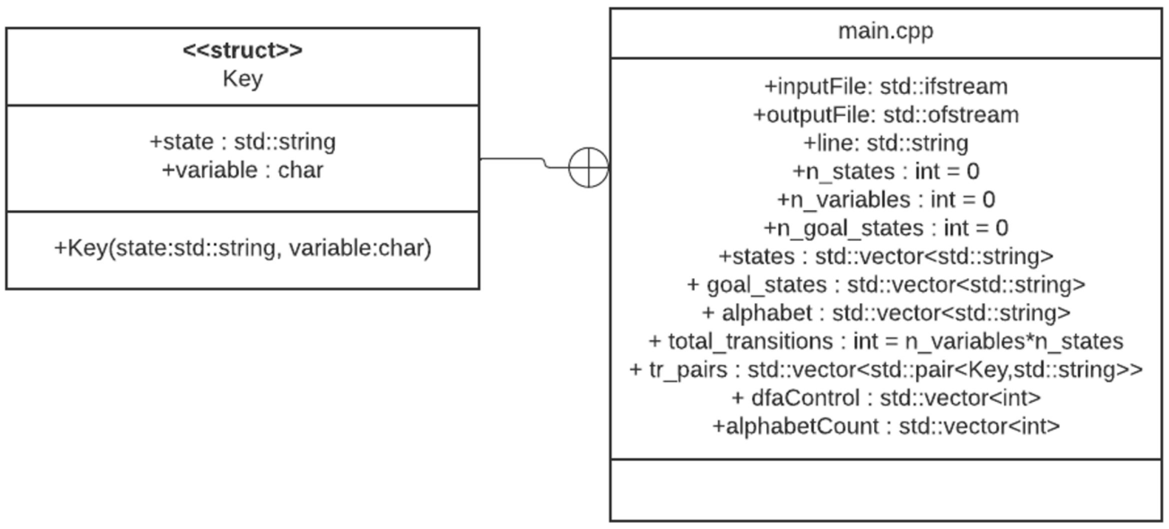
3 Software design description

3.1 C++ Software

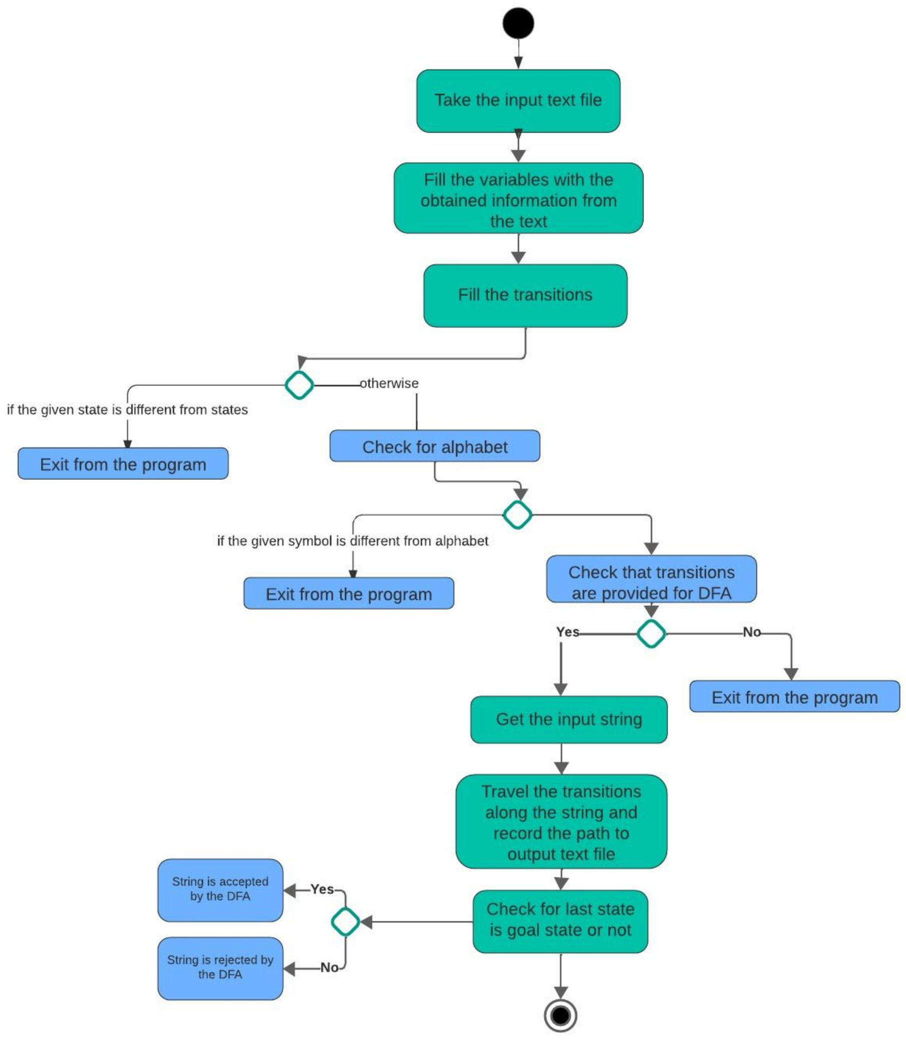
3.1.1 Component interfaces

This software, which is written in C++ language, takes an input file from the local and it reads the inputs from the text file. With the input data, it creates necessary components of DFA which are states, alphabet, final states, and transitions. Then, following the algorithm, it creates an output and writes the results to output.txt file.

3.1.2 Component design description


<<struct>>
Key
+state : std::string
+variable : char
+Key(state:std::string, variable:char)
main.cpp
+inputFile: std::ifstream
+outputFile: std::ofstream
+line: std::string
+n_states : int = 0
+n_variables : int = 0
+n_goal_states : int = 0
+states : std::vector<std::string>
+ goal_states : std::vector<std::string>
+ alphabet : std::vector<std::string>
+ total_transitions : int = n_variables*n_states
+ tr_pairs : std::vector<std::pair<Key,std::string>>
+ dfaControl : std::vector<int>
+alphabetCount : std::vector<int>

3.1.3 Workflows and algorithms


Take the input text file
Fill the variables with the obtained information from the text
Fill the transitions
otherwise
if the given state is different from states
Exit from the program
Check for alphabet
if the given symbol is different from alphabet
Exit from the program
Check that transitions are provided for DFA
Yes
No
Exit from the program
Get the input string
Travel the transitions along the string and record the path to output text file
Check for last state is goal state or not
Yes
No
String is accepted by the DFA
String is rejected by the DFA

3.1.4 Software requirements mapping

SRS-DFA-001.0 is being handled by this component.

4 Error Handling

- Error 1: A symbol that is not in the alphabet may have been given for the symbol in the transition.
  Solution: It writes this error to console clearly and exits from the program. It does not continue until the correct symbol is read from the file.
- Error 2: A state that is not in the states may have been given for the state in the transition.
  Solution: It writes this error to console clearly and exits from the program. It does not continue until the correct transition is read from the file.
- Error 3: The transitions may not provide a DFA.
  Solution: It writes this error to console clearly and exits from the program. It does not continue until the transitions provide a DFA.
- Error 4: There cannot be more than one identical symbol transition from the same state.
  Solution: It writes this error to console clearly and exits from the program. It does not continue until the transitions provide a DFA.
- Error 5: Input string may have different symbols which are not included in alphabet.
  Solution: It writes this error to console clearly and exits from the program. It does not continue until the input string is provided correctly.

5 Testing

For the correctness of my implementation, I applied 4 different test cases, and each case has 2 different examples. One of them from each case is provided below.

Example DFA:

- Q = {q1, q2}
- ∑ = {a, b}
- Start state is the first state of transitions, q1 in this case.
- The accept state is q2.
- Transitions:
  - q1 a q1
  - q1 b q2
  - q2 a q2
  - q2 b q1

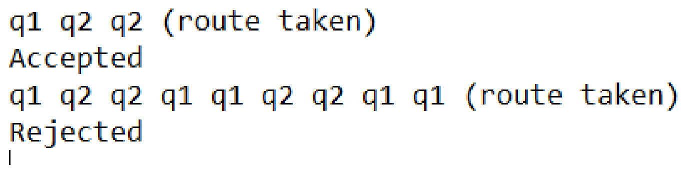
Positive Test Case: The string is chosen as “aaaaabbbaa” which should be accepted by this DFA. The result of the DFA software is:

```
q1 q1 q1 q1 q1 q2 q1 q2 q2 q2 (route taken)
Accepted
```

Negative Test Case: The string is chosen as “aaaaabb” which should be rejected by this DFA. The result of the DFA software is:

```
q1 q1 q1 q1 q1 q2 q1 (route taken)
Rejected
```

Demo Test Case: The first string “aba” should be accepted by the DFA. Second string “ababababa” should be rejected by the DFA.
The result of the DFA software is:



```
q1 q2 q2 (route taken)
Accepted
q1 q2 q2 q1 q1 q2 q2 q1 q1 (route taken)
Rejected
```

6 Code Structure

In the implementation of code, some variables are defined, and some structures are used such as if conditions, for loops, while loop, pair and struct element.

- Necessary variables are defined, since a DFA must carry the properties, such as having states, alphabet, transitions, final states, and start state.
- A structural element named Key is used for storing the state, which has a transition going from this state to another state, and its transition. Using the std::make_pair function from the C++ library, Key and destination state is paired and added to the transitions vector as a pair.
- For loops are used for reading the information from the input text file such as states, alphabet, goal states, transitions. Those are pushed to separate vectors. To control the given machine is DFA or not, for loops are used.
- While loop is used to take strings to be detected by the machine. Since text file can include lots of strings to be detected, while loop is decided to use.
- If conditions are used to find out whether there is text on the next line. Furthermore, these conditions are used to verify whether the provided element in the transition corresponds to a valid state and whether the given alphabet character is indeed part of the alphabet and verify whether the transitions conclude a DFA. Lastly, it is checked whether the last state reached is among the final states.

7 COTS Identification

There is not any external software used in this code.