# Validating Input String Through DFA

Guide: Prof. Pritee Bailke, Group Members: Prafull Sonawane, Srinath Divate, Sumit Tambe, Avadhoot Sutar, Priyansh Wankhade.

**Department of Artificial intelligence and Data Science**
**Vishwakarma Institute of Technology, Pune, 411037, Maharashtra, India**

*Abstract* — DFA refers to deterministic finite automata. Deterministic refers to the uniqueness of the computation. The finite automata are deterministic FA, if the machine reads an input string one symbol at a time. In DFA, there is only one path input from the current state to the next state. This concept we are using in our terminal-based program. We will pass a string to the program, then we have to choose whether to validate it with machine 1 and machine 2. Also, we have done email validation, so if we put an email id as a string, it will check for its validation, if it fits under the rules for validating email, then it will be accepted.

***keywords - DFA, string, Email validation, machines***

## I. INTRODUCTION

A DFA is a finite-state machine that accepts or rejects a given string of symbols, by running through a state sequence uniquely determined by the string. A DFA has a start state (denoted graphically by an arrow coming in from nowhere) where computations begin, and a set of accept states which help define when a computation is successful. DFA can model software that decides whether or not online user input such as email addresses are syntactically valid. In this paper we are mentioned 3 machines which are having 2 difference language and and 1 email validating which checks email is syntactically correct or not. For other 2 machines we are using some other language to check its validation.

## II. WHAT IS DFA?

A deterministic finite automaton *M* is a 5-tuple, $(Q, \Sigma, \delta, q_0, F)$, consisting of

- a finite set of states $Q$
- a finite set of input symbols called the alphabet $\Sigma$
- a transition function $\delta : Q \times \Sigma \rightarrow Q$
- an initial or start state $q_0 \in Q$
- a set of accept states

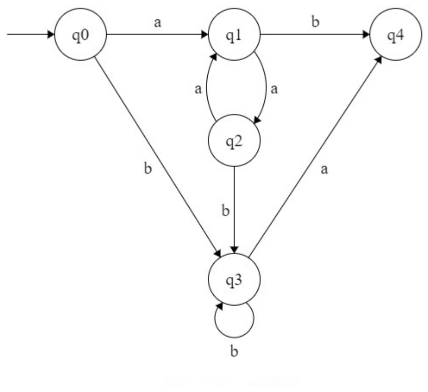
$F \subseteq Q$

## III. EXAMPLE

### 1. Machine 1

**L = {$a^n b^m$ | n % 2 = 0, m > 0}**

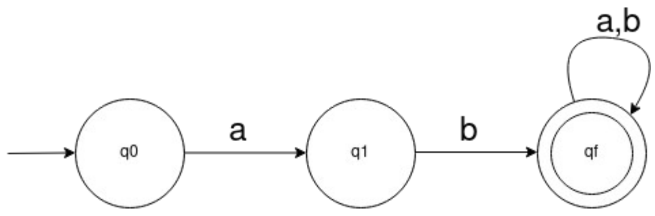
- in this machine for moving to state 1 char will be 'a' but if there is only 'b' then string is in final state
- there is a dead state which is state 4
- machine1 is valid for such example {aab, aaaabb, b, aabbb…}



### 2. Machine 2

**L = {ab.$a^n b^m$ | n >= 0, m >= 0}**

- in this machine for moving to state1 char will be only 'a' otherwise it will goes to dead state.
- Then at state2 string char much need 'b' to move to final state so there at least string size would be 2, which are consist of firstly 'a' and then 'b'
- there is a dead state
- after 'a' and 'b' there can be any char 'a' or 'b'
- examples: ab, abb, aba, abbbab

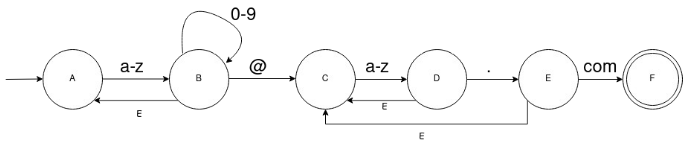


### 3. Machine 3

**email Validator**

**The rules for validating email IDs, and some valid and invalid examples are mentioned here:**

1. Email addresses must start with a letter symbol. And any number of letters or digits \can be appended, and only a single dot (.) is allowed but other symbols and white spaces are not allowed.
2. The name field of the address must end with either a letter or digit.
3. If an underscore or dote is used then before it, letters or digits must be used for a valid name.
4. Dot can be used only once



## IV. MERITS AND DEMERITS OF DFA

1. Merits
   - For each symbolic representation of the alphabet, there is only one state transition in DFA.

- DFA can be understood as one machine.
- Backtracking is allowed in DFA.
- Time needed for executing an input string is less.

2. Demerits
    - DFA cannot use Empty String transition.
    - DFA is more difficult to construct.
    - DFA rejects the string in case it terminates in a state that is different from the accepting state.
    - Conversion of Regular expression to DFA is difficult.

## V. CONCLUSION

non-deterministic automata can provide more compact representations of languages. For example, it is well-known that there are NFA whose minimal equivalent DFA are exponentially larger. Use in theory. Using non-determinism can simplify proofs, see e.g. converting regular expressions into finite automata.