Research article Available online www.ijsrr.org ISSN: 2279–0543

International Journal of Scientific Research and Reviews

Algorithm Design and String Recognition for Suffix Strings Using Deterministic Finite Automata.

Singh Kamalpreet[*] and Goyal Gulshan[2]

[*]Scholar, Department of CSE, CCET (Degree Wing), Chandigarh (India)
[2]Assistant Professor, Department of CSE, CCET (Degree Wing), Chandigarh (India)

ABSTRACT:

Theory of computation is one of the challenging and important area in field of Computer Science. The subject of automata Theory deals with variety of formal languages like Regular Languages, Context-free languages, Context-sensitive languages etc. Different formal languages have different accepting machines. For example, regular languages can be accepted by finite automata. An automaton can be used to process strings of given language. A finite automaton can be deterministic or non-deterministic in nature. Very few researchers have focused on algorithmic approach for design and implementation of Deterministic Finite Automata (DFA). Present paper aims at proposing and implementing an algorithm for DFA construction using an approach in which array of size corresponding to the states of DFA is used to store the remaining input required to complete the suffix. Algorithm to find out DFA ending with suffix will take the suffix as input from user along with the possible input characters. In the scope of present paper only regular languages based on suffix strings are considered. This algorithm will have a transition table as output. At the end, a string will be taken as input to check whether the DFA accepts it or rejects it.

KEYWORDS: Alphabet, Regular languages, DFA, NFA, String Processing.

***Corresponding author:**

Kamalpreet Singh

Department of Computer Science and Engineering,

Chandigarh College of Engineering and Technology,

Chandigarh-160019, INDIA

E-mail: kamalaujla4vr@gmail.com, Mob. No.- 7508757424

INTRODUCTION

Languages are known as medium of communications. A Language can either be Natural language or Formal language. Natural Languages are the ones which are used by humans for communication for e.g. English, Hindi etc. Formal Languages are defined over set of alphabets in accordance of some rules known as grammar. Formal Languages are further classified in form of hierarchy: Regular Languages, Context-free Languages, Context-sensitive Languages and Recursively enumerable language.

Each language has an accepting automaton or recognizer. Finite Automata is used to check the acceptance of Regular Languages. Finite Automata is a machine consisting of set of states whose controls moves in response to external inputs. Finite Automata is further divided into two categories: deterministic finite automata (DFA) in which for every input corresponding to every state there exist exactly one transition, and Non- Deterministic finite automata (NFA) in which for every input corresponding to every state may exist 0, 1 or more than one transitions. DFA cannot have null transitions. DFA are also used for string recognition i.e. to check whether a string is accepted by given DFA or not.

A DFA can be represented using:

a. Transition diagram
b. Transition table

In Transition diagram, every transition in depicted by an arrow with input symbol written over it and every state is depicted by a circle with state name written inside it. Initial states are connected with an arrow. Final states are double circled. In transition table, each row will depict a state in DFA and each column will depict input symbol. Cell[i][j] will store the state where transition at state i will take place when input j is provided.

An example of DFA ending with suffix ‘abb’ is shown in Figure 1(transition diagram for DFA) and Table1 (transition table for DFA).

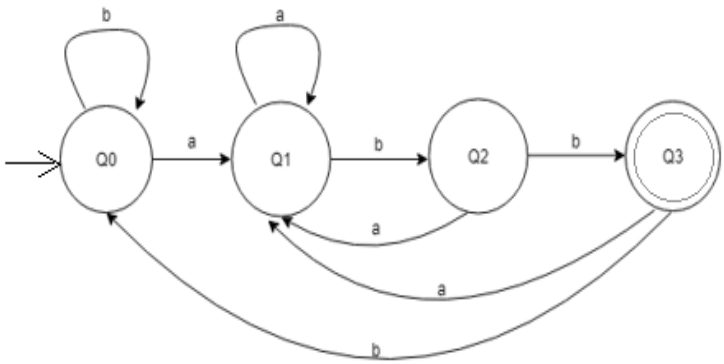

Figure 1: Transition Diagram for DFA Ending with Suffix ‘abb’

Table 1: Transition Table for DFA Ending with Suffix 'abb'

States	A	b
→q0	q1	q0
q1	q1	q2
q2	q1	q3
q3*	q1	q0

Once DFA is complete, String processing can be done by traversing string character by character and carrying out transition on DFA by taking those characters as input on the states of DFA. If the final state by the end of whole string is traversed is not the Final/Accepting state of DFA then the string is not accepted by the DFA.

Some of the major types of DFA are: DFA ending with suffix, DFA starting with prefix, DFA containing a substring, DFA containing exactly, at least or at most number of occurrences of symbol. Present paper aims at the implementation of an algorithm for construction of DFA ending with suffix.

RELATED WORK

Theory of computation has been a challenging research field since past few decades. It' s initial research begins in 1936 when Alan Turing came up with idea of first 'infinite' model of computation, known as Turing machine which is a type of finite automaton with infinite storage since finite automata doesn't have storage. Idea of finite automata came forward in 1943, when two neurophysiologists Warren McCulloch and Walter Pitts tried to model the human thought process, whether in the brain or in a computer [1]. In Turing machine, there's an infinite tape while in finite automata, only finite number of states are there. This is one of the reason finite automata is designed for lower level of languages.

In mid 1950s, two computer scientists, G.H. Mealy and E.F. Moore, came up with much powerful machines in separate papers in which output was also considered. In earlier model of Finite automata, it was a set of transitions and states with no output. Machines proposed by G.H Mealy and E.F. Moore, are named as Mealy and Moore machines. In Mealy machines determines its output for the transitions in DFA. An output is assigned to every transition in DFA. In Moore machines, the output is assigned to every state in DFA. Every DFA and NFA have 5-tuple definition which contains the information about states, transition function, alphabet set, initial and final states, discussed in [2]. There is lot of work and research going on applications of Deterministic Finite automata. Application of DFA in field of compiler design is discussed in [3]. Automata theory has wide range of applications in bio informatics, deep packet inspection etc as discussed in [4.] In networking, DFA is used for logic behind Intrusion Detection System [5]. Other DFA applications

include protocol analysis [6], text parsing, video game character behaviour, security analysis, CPU control units, natural language processing, and speech recognition. Additionally, many mechanical devices are frequently designed and implemented using DFAs, such as elevators, vending machines, and traffic-sensitive traffic lights. Along with these, there are few more applications of DFA like token recognition, Mealy and Moore machines (DFA with output) and Regular expression matching [7], data compression, optical character recognition etc. Speech recognition is carried out using finite-state transducer [8]. An algorithm to generate construction of DFA problems automatically is discussed in [9]. Though DFA, does not covers machines for all types like 'Turing machine' does, it covers all the automaton related to simple regular languages.

PROBLEM FORMULATION

Construction of DFA is an important part of automata theory and has a wide range of applications as discussed in previous section. Although research is going on automata theory for past decades, new learners still face difficulty in designing DFA as per the following reasons:

1. There exists no well-defined algorithm using which beginners can easily understand how to construct DFA ending with suffix.
2. In software like JFLAP, user need to add transitions for the DFA manually in order to have transition table for the given DFA.

In the upcoming section, an algorithm is proposed to construct a DFA ending with suffix along with its flowchart. The algorithm will take the suffix string and the alphabets as input and will give the transition table corresponding to the DFA as output. The algorithm is not programming language specific but general in nature, therefore can be used to construct DFA in any programming language.

METHODOLOGY.

In scope of present paper, algorithm for construction of DFA ending with suffix is discussed. In this, once provided with the alphabet used and the suffix, a transition table or diagram of DFA should be constructed in such a way that every string when traversed with its character given as input to current state ends with the given suffix is accepted by the DFA. If the suffix of the input string is not same as the given suffix, it should not be finished at final state but any other non-final state. For this construction of DFA ending with suffix, one approach is discussed below:

Algorithm:

Input: Input for the algorithm will be number of input alphabets *n*, set of alphabets in array *c*, string to be used as suffix for ending condition *str*.

Output: Output for the following algorithm will be transition table of the DFA.

BEGIN:

1. Input the number of characters n, character set c, and string to be used as suffix str.
2. Number of states in the DFA will be equal to the length of suffix str + 1. In case, if length of suffix is L, then the number of states in L+1.
3. Create an array 'state' of size L+1 and at each array cell at index 'i' store the substrings of str from index i to L.
4. Create a new variable 'current' to store the current processed input. Initialize the current variable to NULL. Set the initial state j = 0
5. For every state j and initial value of current, concatenate value of current by each character in character set. Repeat this step until the character set is fully covered.
6. Once the new value variable 'current' find the suffix of current of maximum length and concatenate it with every value of array 'state'. If this concatenation of maximum length suffix of current + state[k] is equal to the value of string to be used as suffix (str), then set the next state of transition for given character as 'k'.
7. For next step, set variable current = current + str[j].
8. Increment the value of j by 1 and repeat the step 5 until value of j is less than L+1.

END.

Flow- chart:

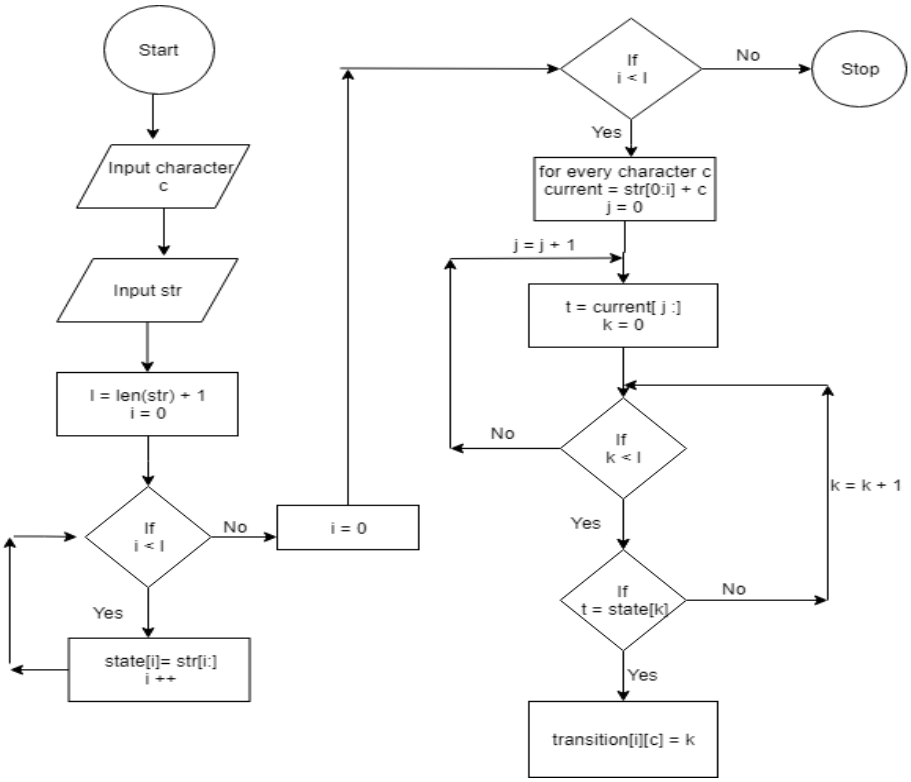

Figure 2: Flow Chart for Implementation of DFA with Suffix.

RESULTS and DISCUSSIONS

Construction of DFA is very important concept as DFA is recognizer of Regular languages. Different approaches are there for construction of DFA, but it is hard for new learners to learn those approach and use it for practical implementation. In present paper, one of the algorithms for construction of DFA ending with suffix is implemented using approach in which remaining input at a state 'i' is stored in array at index 'i'. Using this algorithm, a user can create a program to create DFA for case ending with a suffix. Though algorithm is language independent, the implementation considered for this paper is done in Python.

For the given algorithm, input required is the number of input characters, input character set and string to be used as suffix. Output for the algorithm is Transition table corresponding to the DFA.

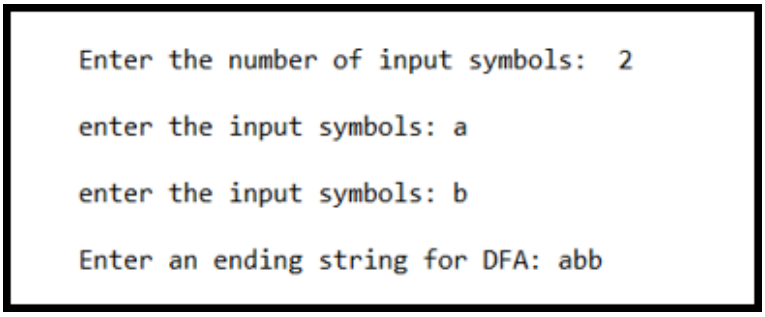

Figure 3. Input for the Given Algorithm

The output for the code corresponding to given algorithm will look like Figure 4. Once transition table for the DFA is created, next thing is to check whether a given string is accepted by the constructed DFA or not. Next input corresponding to DFA of a case where suffix(w) = " abb", will be a string 's'. There will be two possible results: either the string will be accepted if it has the suffix = w or else it will be rejected.

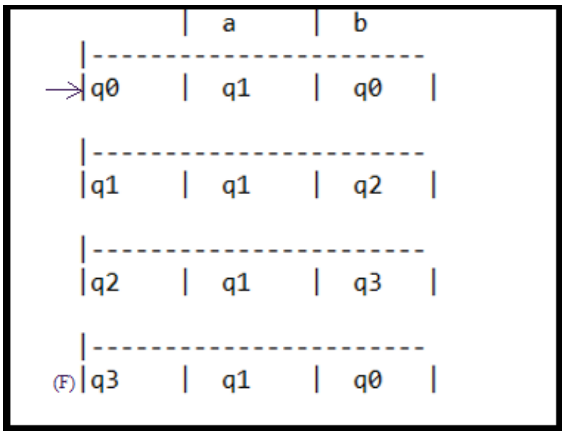

Figure 4. Transition Table Output Corresponding to the Input to Algorithm

For string processing, whole string will be traversed character by character starting from the initial state. For the DFA with suffix = 'abb' and string which we want to traverse is s = 'babb'. String processing corresponding to this is given in Figure 5.

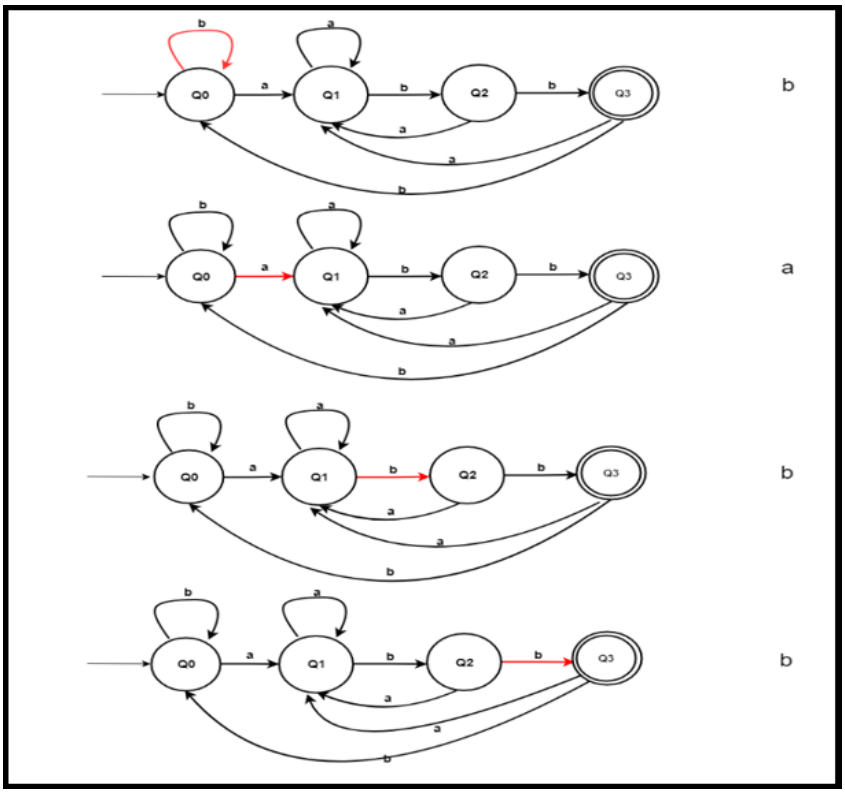

Figure 5. String Processing on the DFA with Suffix 'abb'

When the string input for processing contains suffix = 'abb' for which DFA is defined, the output will be "String accepted" as shown in Figure 6. When string input for processing doesn't contain suffix for which DFA is constructed, output will be "String Rejected" as shown in Figure 7.

```
Enter the string you want to check:  bbaabb
String accepted
```

Figure 6. Acceptance of String for DFA with Suffix 'abb'

```
Enter the string you want to check:  babbab
String rejected
```

Figure 7. Rejection of String for DFA with Suffix 'abb'

CONCLUSION AND FUTURE SCOPE

Automata theory is the branch of computer science which deals with computations. Design of automata is very important part of automata theory since it is used as recognizer for regular languages. In scope of present paper, an algorithm is proposed and implemented for the construction

of DFA ending with suffix since there exist no well-defined algorithm which can help new learners to implement the DFA using a program. This algorithm takes suffix and the alphabet set as input and gives transition table corresponding to the DFA as output. Along with this, discussions have also been made about the string processing, once transition table for DFA is obtained.

In future, time complexity of the proposed algorithm can be improved. Also, algorithm covers only specific case of DFA ending with suffix. A more generalized algorithm can be constructed which may cover other cases of DFA constructions as well.

REFERENCES

1. McCulloch, W.S., & Pitts, W.. *"A logical calculus of the ideas immanent in nervous activity"*. The bulletin of mathematical biophysics. 1943; 5(4): 115-133.
2. Hopcroft, J. E.. Introduction to Automata Theory, Languages and Computation. 3rd Edition. Pearson Education; India; 2008: 37-55.
3. Ullman, J. D. Applications of language Theory to Compiler Design, Proceedings of the spring joint computer conference (Spring). May 16-18, 1972: 235-242(ACM).
4. Aprarna, Goyal G., *"Application Review of Automata Theory"*, International Journal of Scientific Research in Computer Science, Engineering and Information Technology; 3(1) :947-955.
5. Rathod P.M., Marathe N., Vidhate A. V.,*"A survey on Finite Automata based pattern matching techniques for network intrusion Detection System (NIDS)"*. In proceedings of IEEE International Conference on Advances in Electronics Computer and Communications. Bengaluru, India, 2014: 1-5.
6. Graham, R. D., & Johnson, P. C. *"Finite state machine parsing for internet protocols: Faster than you think.*" IEEE Security and Privacy Workshops.2014 : 185-190.
7. Liu, A. X., Torng, E., Liu, A. X., & Torng, E. *"Overlay automata and algorithms for fast and scalable regular expression matching."* IEEE/ACM Transactions on Networking. 2016: 2400-2415.
8. Mohri, M., Pereira, F., & Riley, M. *"Speech recognition with weighted finite-state transducers"*. In Springer Handbook of Speech Processing, 2008: 559-584.
9. Shenoy V., Aparabhu U., Sripradha K., Kumar V., *"Generating DFA construction Problems Automatically"*. IJCTT. 2013; 4(3): 32-37.