Journal of Harbin Engineering University
ISSN: 1006-7043

Vol 44 No. 10
October 2023

DFA that Accepts the Set of All Strings of 0's and 1's, such that Each Block of 5 Digits has atleast 2 0's

[1]Venkatesan, G., [2]Vijayalakshmi V

[1]Department of Civil Engineering, Saveetha Engineering College, Tamil Nadu, India.

[2]Department of Computer Science Engineering, Saveetha Engineering College,Tamil Nadu, India.

Abstract

Automata theory (also known as Theory of Computation) is a theoretical branch of Computer Science and Mathematics, which mainly deals with the logic of computation with respect to simple machines, referred to as automata. Automata enables scientists to understand how machines compute the functions and solve problems. The main motivation behind developing Automata Theory was to develop methods to describe and analyze the dynamic behavior of discrete systems. Finite automata are employed in pattern-matching and recognition tasks. They can be used to search for specific patterns or sequences of characters within a given input. Applications include text processing, string matching, and searching algorithms.It has been clearly known that there has been difficulty in understanding the concepts of Finite Automaton. Many people face difficulty in understanding the construction of a DFA. This article analyses about the different papers that has been published related to construction of a finite automata. It explains about the construction of a DFA with examples and a solution has been given to construct a DFA that accepts the set of all strings of 0's and 1's, such that each block of 5 digits has atleast 2 0's.

Keywords: FiniteAutomata,Deterministic Finite Automata, Transition Table, Transition Diagram, Regular Languages.

1.INTRODUCTION

Finite Automata(FA) is the simplest machine to recognize patterns.It is used to characterize a Regular Language. Also it is used to analyze and recognize Natural language Expressions. The finite automata or finite state machine is an abstract machine that has five elements or tuples. It has a set of states and rules for moving from one state to another but it depends upon the applied input symbol. Based on the states and the set of rules the input string can be either accepted or rejected. Basically, it is an abstract model of a digital computer which reads an input string and changes its internal state depending on the current input symbol. Every automaton defines a language i.e. set of strings it accepts.

The following figure shows some essential features of general automation.

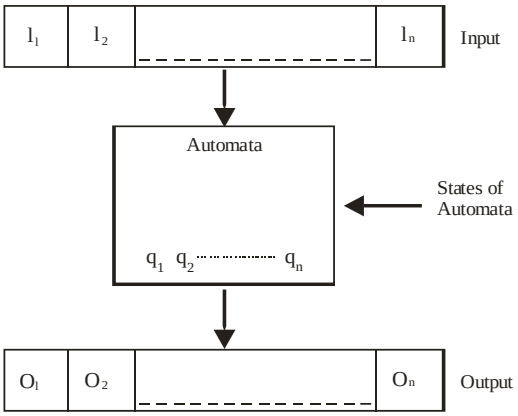


Figure: Features of Finite Automata

The above figure shows the following features of automata:

1. Input
2. Output
3. States of automata
4. State relation
5. Output relation

FA is characterized into two types:

1) Deterministic Finite Automata (DFA)

DFA consists of 5 tuples {Q, Σ, q, F, δ}.

Q : set of all states.

Σ : set of input symbols. (Symbols which machine takes as input)

q : Initial state. (Starting state of a machine)

F : set of final state.

δ : Transition Function, defined as δ : Q X Σ --> Q.

Generally a DFA is a machine where, for a particular input character, the machine goes to one state only. A transition function is defined on every state for every input symbol. Also in DFA null (or ε) move is not allowed, i.e.,The DFA cannot change its state without any input character.

For example, let us construct a DFA which accept a language of all strings ending with'a'.

First let us write the formal specification for the given DFA.The input symbols can be a,b. We can consider the starting state to be q0.There will be an accepting state which can be q1.Q denotes the number of states that can be in the DFA.so the formal specification can be as follows.

Σ = {a,b}, q = {q_0}, F={q_1}, Q = {q_0, q_1}

Now let us try to write the language L, which accepts set of all the possible acceptable strings in order to construct an accurate state transition diagram.so we can write the language as follows
L = {a, aa, aaa, aaaa, aaaaa, ba, bba, bbbaa, aba, abba, aaba, abaa}
The strings specified in L denotes the set of strings that can be accepted by the DFA.It accepts set of all stings ending in 'aa'.

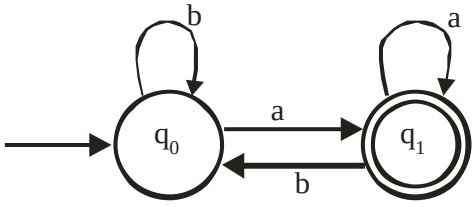


Fig 1. State Transition Diagram for DFA which accepts strings ending in 'a'

Strings not accepted are, ab, bb, aab, abbb, etc.

State transition table for above automaton,

↓State \ Symbol→	A	B
q_0	q_1	q_0
q_1*	q_1	q_0

We have to know that **there can be many possible DFAs for a pattern**. A DFA with a minimum number of states is generally preferred.

2) Nondeterministic Finite Automata(NFA): NFA is similar to DFA except following additional features:

1. Null (or ε) transition is allowed i.e., it can move forward without reading symbols.

2. Ability to transmit to any number of states for a particular input.

However, these above features don't add any power to NFA. If we compare both in terms of power, both are equivalent.

Due to the above additional features, NFA has a different transition function, the rest is the same as DFA.

δ: Transition Function

δ: Q X (Σ U ε) --> 2 ^ Q.

Now let us construct an NFA for the same language above

The transition diagram can be drawn as follows

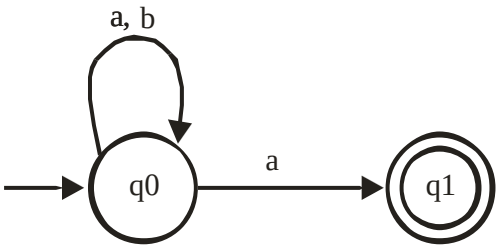


Fig 2. State Transition Diagram for NFA which accepts strings ending in 'a'

State Transition Table is as follows

↓State \ Symbol→	A	b
q_0	$\{q_0,q_1\}$	q_0
q_1*	∅	∅

One important thing to note is, **in NFA, if any path for an input string leads to a final state, then the input string is accepted**. For example, in the above NFA, there are multiple paths for the input string ending in 'a'.

Some Important Points:

- **Justification:**

Since all the tuples in DFA and NFA are the same except for one of the tuples, which is Transition Function (δ)

In case of DFA

δ : Q X Σ --> Q

In case of NFA

δ : Q X Σ --> 2^Q

Now if you observe you'll find out Q X Σ –> Q is part of Q X Σ –> 2^Q.

On the RHS side, Q is the subset of 2^Q which indicates Q is contained in 2^Q or Q is a part of 2^Q, however, the reverse isn't true. So mathematically, we can conclude that **every DFA is NFA but not vice-versa**. Yet there is a way to convert an NFA to DFA.

1. Both NFA and DFA have the same power and each NFA can be translated into a DFA.
2. There can be multiple final states in both DFA and NFA.
3. NFA is more of a theoretical concept.
4. DFA is used in Lexical Analysis in Compiler.
5. If the number of states in the NFA is N then, its DFA can have maximum 2^N number of states.

2.Basic Terminologies of Theory of Computation

Here are the basic terminologies, which are important and frequently used in the Theory of Computation.

a.Symbol:

A symbol (often also called a **character**) is the smallest building block, which can be any alphabet, letter, or picture.

Eg:a,b,c,0,1,…….

b.Alphabets (Σ):

Alphabets are a set of symbols, which are always ***finite***.

∑={0,1} is an alphabet of binary digits

∑={0,1,……..9} is an alphabet of decimal digits

∑={a,b,c}

∑={A,B,C,…..Z}

c.String:

A string is a ***finite*** sequence of symbols from some alphabet. A string is generally denoted as ***w*** and the length of a string is denoted as ***|w|***.

Empty string is the string with

zero occurrence of symbols,

represented as ε.

Number of Strings (of length 2)

that can be generated over the alphabet {a, b}:

- -

aa

a b

b a

b b

Length of String |w| = 2

Number of Strings = 4

For alphabet {a, b} with length **n**, number of strings can be generated = $\mathbf{2^n}$.

Note: *If the number of symbols in the alphabet Σ is represented by |Σ|, then a number of strings of length n, possible over Σ is* $\boldsymbol{|\Sigma|^n}$.

d.Closure Representation in TOC

$\mathbf{L^+}$: It is a ***Positive Closure*** that represents a set of all strings except Null or ε-strings.

$\mathbf{L^*}$: It is "**Kleene Closure**", that represents the occurrence of certain alphabets for given language alphabets from zero to the infinite number of times. In which ε-string is also included.

From the above two statements, it can be concluded that:

$\mathbf{L^* = \varepsilon L^+}$

Example:

(a) Regular expression for language accepting all combination of g's over Σ={g}:

$$\mathbf{R = g^*}$$

R={ε,g,gg,ggg,gggg,ggggg,...}

(b) Regular Expression for language accepting all combination of g's over Σ={g} :

$\mathbf{R = g^+}$

R={g,gg,ggg,gggg,ggggg,gggggg,...}

Note: Σ* is a set of all possible strings(often power set(need not be unique here or we can say multiset) of string) So this implies that ***language*** *is a subset of* Σ*.This is also called a ***"Kleene Star"***.

Kleene Star is also called a ***"Kleene Operator"*** or ***"Kleene Closure"***. Engineers and IT professionals make use of Kleene Star to achieve all set of strings which is to be included from a given set of characters or symbols. It is one kind of Unary operator. In Kleene Star methodology all individual elements of a given string must be present but additional elements or combinations of these alphabets can be included to any extent.

Example:

Input String: "GFG".

Σ* = { ε,"GFG","GGFG","GGFG","GFGGGGGGGG","GGGGGGGGFFFFFFFFGGGGGGGG",...}

Kleene Star is an infinite set but if we provide any grammar rules then it can work as a finite set.

Here we can include ε string also in given Kleene star representation.

e.Language:

A language is a *set of strings*, chosen from some Σ* or we can say- **'**A language is a subset of Σ* **'**. A language that can be formed over ' Σ ' can be **Finite** or **Infinite**.

Example of Finite Language:

L1 = { set of string of 2 }

L1 = { xy, yx, xx, yy }

Example of Infinite Language:

L1 = { set of all strings starts with 'b' }

L1 = { babb, baa, ba, bbb, baab, ... }

3.Literature Review

Constructing a FiniteAutomatafinds more importancenow a dayssinceitsapplicationis in manyareas.

[1]Ravirajsinh Chauhan and Chandan Trivedi(2019) have introduced a new approach in order to construct MDFA directly from infinite regular language which is free from equal states and unreachable states. [2]Syed Asif Ali(2013) has designed a minimal state deterministic finite automaton for Sign language pattern using the concept regular expression. [3]M. P. Rajakumar, J. Ramya and R. Sonia(2020) has developed a novel algorithm for constructing DFA directly from regular expressions (REs) for pattern recognition problems are proposed. This algorithm can be extended to accept more input variables with minor modifications. [4]Sanjaybhargava and G. N. Purohit(2011) has developed a method for constructing a minimal deterministic finite automaton (DFA) from a regular expression. It is based on a set of graph grammar rules for combining many graphs (DFA) to obtain another desired graph (DFA). The graph grammar rules are presented in the form of a parsing algorithm that converts a regular expression R into a minimal deterministic finite automaton M such that the language accepted by DFA M is same as the language described by regular expression R.

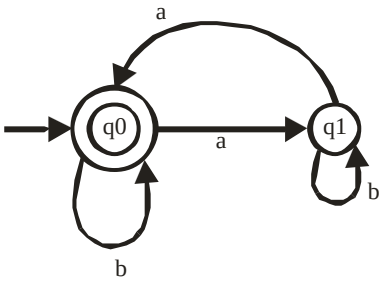
[5]R. Harsh Gujar, Shubham Deokate, Karan Gawali, Yash Chungade and Vivek Ingle(2022). has discussed about how pattern matching is done. The given input will be searched for the pattern. This proposed model will give the information about the content of the given input string. [6] Abhishek bhardwaj, Achint Chaudhary, Shankar Z. Thawkar, Jhalak and Vijay S.Katta(2016) have constructed optimized deterministic finite automata to accept palindrome string from the string length one to five. [7]N.Murugesan and O.V.Shanmuga Sundaram(2015) has illustrated various types of automata and its accepted languages. The exact language of binary strings of multiples of three representing the DFA is constructed. Based on proposed approach, a systematic way to design a DFA is provided from constructed NFA's.[8]Rashandeepsingh and Dr. Gulshan goya(2022) have proposed an algorithm for designing of deterministic finite automata (DFA) for a regular language with a given prefix. The proposed method further aims to simplify the lexical analysis process of compiler design. Singh [9]Kamalpreet and Goyal Gulshan(2019) proposed 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. [10]S. Ramesh babu, A. Sanyal and G. Venkatesh(1996) has presented a proof of correctness of an algorithm for directly constructing a deterministic finite automaton (DFA) from a regular expression. They do this in a functional framework by introducing a structure called dot annotated regular expression (dare). A dare acts as an implicit representation of a state in a DFA. State transitions in a DFA correspond to dot movements in a dare. They investigate and identify certain algebraic properties of dares which are then used to prove the correctness of the algorithm. The proof is algebraic and presented in the same framework as that of the algorithm itself.[11]K.Senthil Kumar and D.Malathi(2015) has described a novel method which finds Deterministic Finite Automata directly from a given regular grammar without going through NFA .They have extended the idea of GOTO and CLOSURE functions in LR parsing model to Regular grammars and use the same to find the states of the required Deterministic Finite Automata. Also they have proposed a new algorithm which minimizes the number of states of the obtained Deterministic Finite Automata. [12]A.Senthil,Dr. V.P. Shukla, Dr. Sangappa and R.Biradar(2013) have proposed a fundamental model of computation in terms of non- deterministic finite automata (NFA) for the Susceptible-Infectives-Recovered (SIR) model. Through this model they have tried to prove there could be certain languages which are epidemic regular since it

could be compared with the normal regular languages for which we can have NFA or regular grammar. [13]H.Karamath Ali, and Dr. D. I. George Amalarethinam(2013) have proposed a method for automatic construction of an FSM that can be used to recognize user activities in real time. The proposed method gives promising results when tested with publicly available smart home datasets. [14]P.Sri Ram Chandra, K.S.Sravan and M.S.Chakravarthy(2019) have proposed an outline of Finite Automata that accepts the class of IPV4 Addresses.In this paper, a Deterministic finite automata to accept the set of all stringsof0'sand 1's such that each block of 5 digits has atleast two 0's is being constructed.

4.Construction of DFA for various Languages

We will see some popular regular expressions and how we can convert them to finite automata.

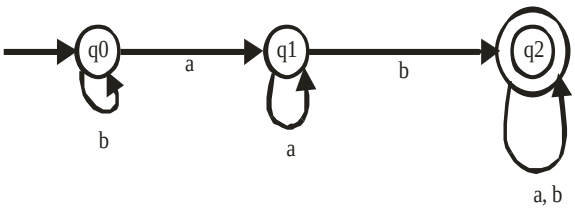
- **Even number of a's :** The regular expression for even number of a's is **(b|ab*ab*)***. We can construct a finite automata as follows



The above automata will accept all strings which have even number of a's. For zero a's, it will be in q0 which is final state. For one 'a', it will go from q0 to q1 and the string will not be accepted. For two a's at any positions, it will go from q0 to q1 for 1st 'a' and q1 to q0 for second 'a'. So, it will accept all strings with even number of a's.

If we want to construct a DFA to accept odd number of a's, then we can change q1 as accepting state in the above DFA.

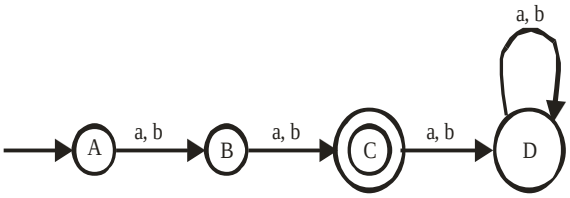
- **DFA to accept String with 'ab' as substring :** The regular expression for strings with 'ab' as substring is **(a|b)*ab(a|b)***. We can construct finite automata as shown below



The above automata will accept all string which have 'ab' as substring. The automata will remain in initial state q0 for b's. It will move to q1 after reading 'a' and remain in same state for all 'a' afterwards. Then it will move to q2 if 'b' is read. That means, the string has read 'ab' as substring if it reaches q2.

Construction of a DFA for the set of string over {a, b} such that length of the string |w|=2 i.e, length of the string is exactly 2

L = {aa, ab, ba, bb}



Here, State A represent set of all string of length zero (0), state B represent set of all string of length one (1), state C represent set of all string of length two (2). State C is the final state and D is the dead state it is so because after getting any alphabet as input it will not go into final state ever.

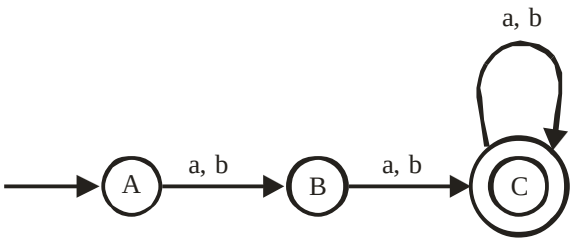
Number of states: n+2

Where n is |w|=n

The above automata will accept all the strings having the length of the string exactly 2. When the length of the string is 1, then it will go from state A to B. When the length of the string is 2, then it will go from state B to C and when the length of the string is greater than 2, then it will go from state C to D (Dead state) and after it from state D TO D itself.

2.Construction of a DFA for the set of string over {a, b} such that length of the string |w|>=2 i.e, length of the string should be at least 2.

L = {aa, ab, ba, bb, aaa, aab, aba, abb.....}



Here, State A represent set of all string of length zero (0), state B represent set of all string of length one (1), and state C represent set of all string of length two (2).

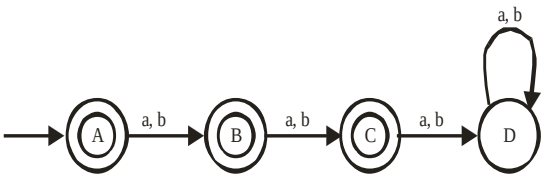
Number of states: n+1

Where n is |w|>=n

The above automata will accept all the strings having the length of the string at least 2. When the length of the string is 1, then it will go from state A to B. When the length of the string is 2, then it will go from state B to C and lastly when the length of the string is greater than 2, then it will go from state C to C itself.

3.Construction of a DFA for the set of string over {a, b} such that length of the string |w|<=2 i.e, length of the string is atmost 2.

L = {?, aa, ab, ba, bb}



Here, State A represent set of all string of length zero (0), state B represent set of all string of length one (1), state C represent set of all string of length two (2), state A, B, C is the final state and D is the dead state it is so because after getting any alphabet as input it will not go into final state ever.

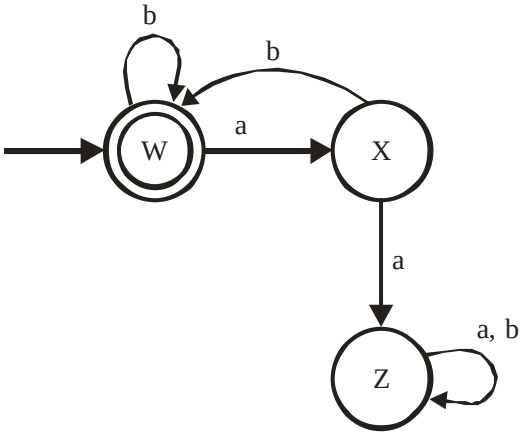
Number of states: n+2

Where n is |w|<=n

The above automata will accept all the strings having the length of the string at most 2. When the length of the string is 1, then it will go from state A to B. When the length of the string is 2, then it will go from state B to C and lastly when the length of the string is greater than 2, then it will go from state C to D (Dead state).

4.Construction of a minimal DFA accepting set of strings over {a, b} in which every 'a' is followed by a 'b'.

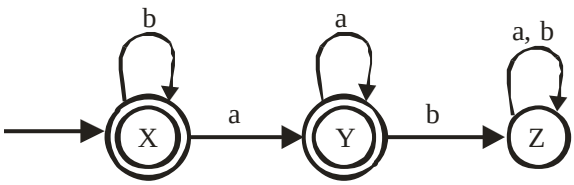
L1 = {ε, ab, abab, abbbb, abababab, ...}



In the above DFA, state 'W' is the initial and final state too which on getting 'b' as the input it remains in the state of itself and on getting 'a' as the input it transit to a normal state 'X' which on getting 'b' as the input it transit to the final state 'W'. The state 'X' on getting 'a' as input it transit to the dead state 'Z'. The state 'Z' is called dead state because on getting any input it can not transit to the final state ever.

5.Construction of a minimal DFA accepting set of strings over {a, b} in which every 'a' is never be followed by 'b'

L1 = {ε, a, aa, aaaa, b, bba, bbbbba.....}

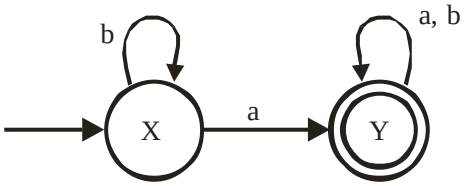


In the above DFA, state 'X' is the initial and final state which on getting 'b' as the input it remains in the state of itself and on getting 'a' as input it transit to the final state 'Y' which on getting 'a' as the input it remains in the state of itself and on getting 'b' as input transit to the dead state 'Z'.

The state 'Z' is called dead state this is because it can not ever go to any of the final states.

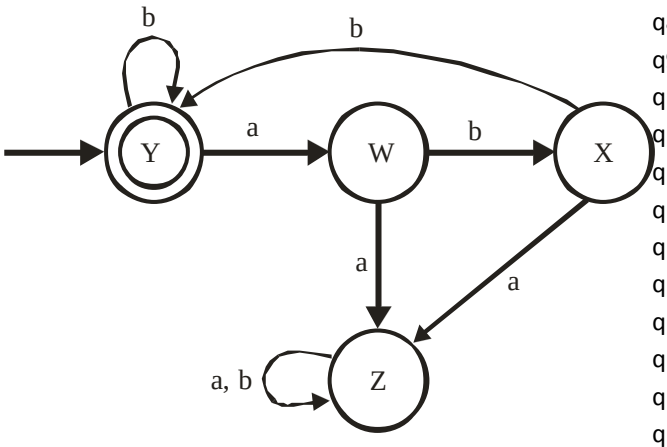
6.Construction of a minimal DFA accepting set of string over {a, b} where each string containing 'a' as the substring.

L1 = {a, aa, ab, ba, ...}



7.Construction of a minimal DFA accepting set of strings over {a, b} in which every 'a' is followed by a 'bb'.

L1 = {ε, abb, abbabb, bbbbabb, abbabbabbabbabbbb, ...}



In the above DFA, state 'Y' is the initial and final state too which on getting 'b' as the input it remains in the state of itself and on getting 'a' as the input it transit to a normal state 'W' which on getting 'a' as the input it transit to the dead state 'Z' and getting 'b' as the input it transit to the normal state 'X' which on getting 'b' as input it transit to the final state 'Y' and on getting 'a' as the input it transit to the same dead state 'Z'. The state 'Z' is called dead state because on getting any input it can not transit to any of the final states ever.

Deterministic finite automata to accept the set of all stringsof0'sand 1's such that each block of 5 digits has atleast two 0's.

Explanation:

Assuming that this DFA accepts only inputs where the length of the string is a multiple of 5 ,because the presence of two 0's can be confirmed for each 5 digits.If the input is not having number of digits which is not a multiple of five, then the input will not be accepted.

We will first decide the number of states for the given problem.

q- Start State
q0- first input symbol is a 0
q1-- first input symbol is a 1
q2-the processed string so far is 00
q3-the processed string so far is 01
q4-the processed string so far is 10
q5-the processed string so far is 11
q6-the processed string so far is 000
q7-the processed string so far is 001
q8-the processed string so far is 010
q9-the processed string so far is 011
q10-the processed string so far is 100
q11-the processed string so far is 101
q12-the processed string so far is 110
q13-the processed string so far is 111
q14-the processed string so far is 0000
q15-the processed string so far is 0001
q16-the processed string so far is 0010
q17-the processed string so far is 0011
q18-the processed string so far is 0100
q19-the processed string so far is 0101
q20-the processed string so far is 0110
q21-the processed string so far is 0111
q22-the processed string so far is 1000
q23-the processed string so far is 1001
q24-the processed string so far is 1010
q25-the processed string so far is 1011
q26-the processed string so far is 1100
q27-the processed string so far is 1101
q28-the processed string so far is 1110
q29-the processed string so far is 1111
q30-the processed string so far is 00000
q31-the processed string so far is 00001
q32-the processed string so far is 00010
q33-the processed string so far is 00011
q34-the processed string so far is 00100
q35-the processed string so far is 00101
q36-the processed string so far is 00110

q37-the processed string so far is 00111
q38-the processed string so far is 01000
q39-the processed string so far is 01001
q40-the processed string so far is 01010
q41-the processed string so far is 01011
q42-the processed string so far is 01100
q43-the processed string so far is 01101
q44-the processed string so far is 01110
q45-the processed string so far is 01111
q46-the processed string so far is 10000
q47-the processed string so far is 10001
q48-the processed string so far is 10010
q49-the processed string so far is 10011
q50-the processed string so far is 10100
q51-the processed string so far is 10101
q52-the processed string so far is 10110
q53-the processed string so far is 10111
q54-the processed string so far is 11000
q55-the processed string so far is 11001
q56-the processed string so far is 11010
q57-the processed string so far is 11011
q58-the processed string so far is 11100
q59-the processed string so far is 11101
q60-the processed string so far is 11110
q61-the processed string so far is 11111

We can write the transition table as follows

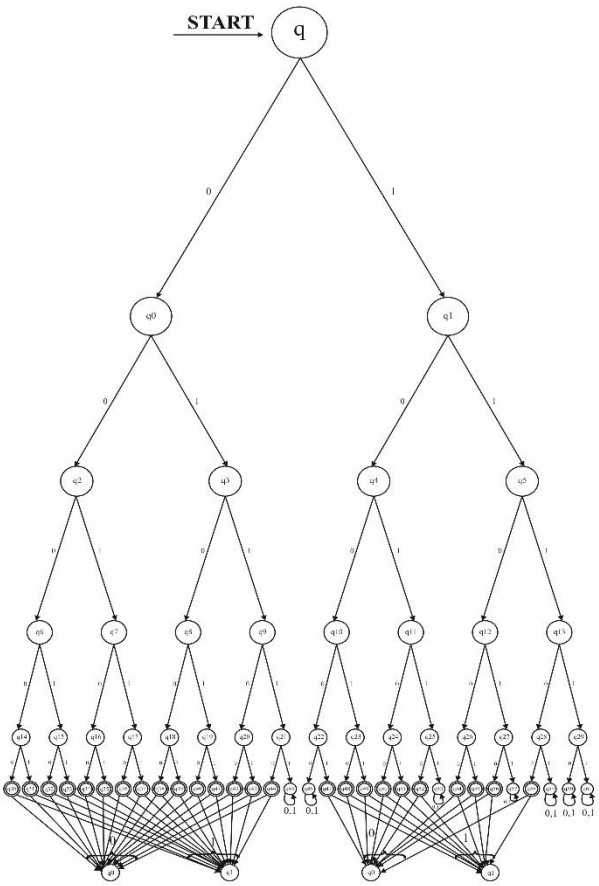
	0	1
→q	q0	q1
q0	q2	q3
q1	q4	q5
q2	q6	q7
q3	q8	q9
q4	q10	q11
q5	q12	q13
q6	q14	q15
q7	q16	q17
q8	q18	q19
q9	q20	q21
q10	q22	q23
q11	q24	q25
q12	q26	q27
q13	q28	q29
q14	q30	q31
q15	q32	q33
q16	q34	q35
q17	q36	q37

	0	1
*q31	q0	q1
*q32	q0	q1
*q33	q0	q1
*q34	q0	q1
*q35	q0	q1
*q36	q0	q1
*q37	q0	q1
*q38	q0	q1
*q39	q0	q1
*q40	q0	q1
*q41	q0	q1
*q42	q0	q1
*q43	q0	q1
*q44	q0	q1
q45	q45	q45
q46	q46	q46
*q47	q0	q1
*q48	q0	q1
*q49	q0	q1

q18	q38	q39	*q50	q0	q1
q19	q40	q41	*q51	q0	q1
q20	q42	q43	*q52	q0	q1
q21	q44	q45	q53	q53	q53
q22	q46	q47	*q54	q0	q1
q23	q48	q49	*q55	q0	q1
q24	q50	q51	*q56	q0	q1
q25	q52	q53	q57	q57	q57
q26	q54	q55	*q58	q0	q1
q27	q56	q57	q59	q59	q59
q28	q58	q59	q60	q60	q60
q29	q60	q61	q61	q61	q61
*q30	q0	q1			

*-Denotes the accepting states

STATE DIAGRAM:



5.Background Study

Applications of Finite Automata

Finite automata have several practical applications across various fields. Here are some notable applications:

- **Lexical Analysis:** Finite automata are extensively used in compiler design for lexical analysis, which involves tokenizing and scanning the source code of a programming language. Lexical analyzers employ finite automata to recognize and classify different lexical units such as keywords, identifiers, operators, and literals.
- **Pattern Recognition:** Finite automata are employed in pattern-matching and recognition tasks. They can be used to search for specific patterns or sequences of characters within a given input. Applications include text processing, string matching, and searching algorithms.
- **Network Protocol Analysis:** Finite automata are used in network protocol analysis and packet filtering. They can be employed to define rules and match patterns in network traffic, enabling functionalities like intrusion detection, firewalls, and network monitoring.
- **Digital Circuit Design:** Finite automata can model the behavior of digital circuits and aid in their design and testing. Sequential circuits, such as counters and state machines, can be represented and analyzed using finite automata.
- **Natural Language Processing:** Finite automata are used in natural language processing for tasks such as text tokenization, morphological analysis, and part-of-speech tagging. They help in parsing and understanding the structure of natural language sentences.
- **Vending Machines:** Finite automata are an appropriate model for designing and implementing the control logic of vending machines. They can manage the states and transitions required to process user inputs and dispense the appropriate products.
- **Regular Expression Processing:** Finite automata are closely related to regular expressions, a powerful tool for specifying patterns in strings. Regular expression engines often utilize finite automata to efficiently match and process regular expressions.
- **DNA Sequence Analysis:** Finite automata find applications in bioinformatics for analyzing DNA sequences. They can be used to identify specific patterns or motifs within DNA sequences, aiding in gene identification, sequence alignment, and genetic research.

6.Limitations and Future Scope

Since the number of states in the DFA is more, it may take more time to process the input strings. To overcome this problem we can use any algorithm to minimize the dfa and reduce the number of states and increase speed processing.

7.Conclusion

Finite automata, also known as finite state machines, are mathematical models used to describe systems with a finite number of states and transitions based on inputs. They have practical applications in areas such as lexical analysis, pattern recognition, network protocol analysis, digital circuit design, natural language processing, and more. Finite automata provide a formal framework for understanding the behavior of systems and are valuable tools for modeling, analyzing, and solving problems in various fields.

8.Summary

Thus this paper has analysed the different papers in the construction of a finite automata.Many finite automatas for different languages have been created and finally a DFA that accepts the set of all strings of 0's and 1's, such that each block of 5 digits has atleast 2 0's has been created.From this survey it can be concluded that we can construct a Finite Automata for any language.We can just create the Finite Automata just by writing the Regular expression which denotes the language for

the problem statement and then proceeding in the construction of the DFA.

REFERENCE

[1] Ravirajsinhchauhan, ChandanTrivedi"International Journal of New Innovations in Engineering and Technology June 2019""A Novel Approach for construction of Minimal Deterministic Finite Automata - RC Algorithm"

[2] Syed AsifAli"International Journal of Recent Technology and Engineering"IJRTE ISSN: 2277-3878 (Online), Volume-2 Issue-1, March 2013" " Design of Minimal States Deterministic Finite Pattern Using the Concept of Regular Expression"

[3] M.P.Rajakumar, J.Ramya, R.Sonia, B.Umamaheswari"AdvancesinMathematics: Scientific Journal 9 (2020), no.9, 6971–6980 ISSN: 1857-8365 (printed); 1857-8438 (electronic) https://doi.org/10.37418/amsj.9.9.46 ""Algorithm for constructing deterministic finite automata directly from regular expressions for pattern recognition related problems "

[4] Sanjay bhargava, G. N. Purohit"InternationalJournalof Computer Applications (0975 – 8887) Volume 15– No.4, February 2011"" Construction of a Minimal Deterministic Finite Automaton from a Regular Expression"

[5] Harsh gujar,ShubhamDeokate,KaranGawali,YashChungade,Vivek Ingle(APRIL 2, 2022)"A Review Paper On Finite Automata Application in String Identification"

[6] Abhishekbhardwaj, Achintchaudhary, Shankar z. Thawkar, Jhalak, Vijay S. Katta"International Journal of Computer Science and Information Technologies, Vol. 7 (3) , 2016, 1440-1443""A Novel Approach to Design Optimized Deterministic Finite Automata to Accept the Palindrome using Three Input Characters"

[7] N.MurugesanAndO.V.ShanmugaSundaram"IJRCS - International Journal of Research in Computer Science Volume: 02 Issue: 04 2015"" A General Approach to DFA Construction"

[8] Rashandeep Singh, Dr.Gulshangoyal,"DOI: 10.37398/JSR.2022.660203"" Algorithm Design for Deterministic Finite Automata for a Given Regular Language with Prefix Strings "

[9] Singh kamalpreet and Goyalgulshan"IJSRR, 8(2) April. – June., 2019"" Algorithm Design and String Recognition for Suffix Strings Using Deterministic Finite Automata"

[10] S. Ramesh babu, A. Sanyal, G.Venkatesh"International Journal of Computer Mathematics (Received 27 February 1996)"" Proof of correctness of a direct construction of dfa from regular expression"

[11] K.Senthilkumar, D.Malathi"International Journal of Scientific & Engineering Research, Volume 6, Issue 3, March-2015 106 ISSN 2229-5518"" A Novel Method To Construct Deterministic Finite Automata From AGiven Regular Grammar"

[12] A.Senthil,Dr.V.P.Shukla,Dr.SangappaR.Biradar "International Journal of Engineering Research & Technology (IJERT) Vol. 2 Issue 9, September - 2013 IJERT ISSN: 2278-0181"" Finite Automata for SIR Epidemic Model"

[13] H.KaramathAli,Dr.D.I.GeorgeAmalarethinam" IOSR Journal of Engineering (IOSRJEN) e-ISSN: 2250-3021, p-ISSN: 2278-8719 Vol. 3, Issue 12 (December. 2013), ||V3|| PP 40-45"" Automatic Construction of Finite Automata for Online Recognition of User Activities in Smart Environments"

[14] P.Sri Ram Chandra ,K.S.Sravan , M.S.Chakravarthy"I.J. Mathematical Sciences and Computing, 2019, 1, 65-79 Published Online January 2019 in MECS (http://www.mecs-press.net) DOI: 10.5815/ijmsc.2019.01.06"" A New Approach to the Design of a Finite Automaton that accepts Class of IPV4 Addresses"