A General Approach to DFA Construction

N.Murugesan[1], O.V.Shanmuga Sundaram [2]
[1](Dept of Mathematics, Government Arts College (Autonomous), murugesanmathsgac@gmail.com)
[2](Dept of Mathematics, Sri Shakthi Institute of Engineering & Technology, Coimbatore, India, ovs3662@gmail.com)

Abstract— *In this paper, various types of automata and its accepted languages are illustrated. 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.*

Keywords— regular expressions, binary strings, and construction of NFA and DFA.

1. INTRODUCTION

In the theoretical foundations of computer science, automata theory is the study of abstract machines and the problems which they are able to solve. These abstract machines are called automata. A discrete automaton is a mathematical model for a finite state machine (FSM). An FSM is a machine that takes a symbol as input and transitions, from one state to another according to a transition function (which can be expressed as a transition table). This transition function tells the automaton which state to go to next given a current state and a current symbol. Turing Machine is one of the vital portions of Automata Theory; it's the father of all computers. That is Automata Theory a set of mathematical calculations and formulas describing the automation or process of that machine. Automata Theory does not deal with real automatons such as robots, but deals with simulated object in a computer. The automaton reads each character when it passes through states. The combined characters make up a string when the automata stops at an accept state.

Basically two models are discussing. Of which, one model called finite automaton is used in text processing, compilers, and hardware design. Another model, called the context-free grammar, is used in programming languages and artificial intelligence. Automata theory is an excellent place to begin the study of the theory of computation. The theories of computability and complexity require a precise definition of a computer. Automata theory allows practice with formal definitions of computation as it introduces concepts relevant to other non theoretical areas of computer science.

2. FINITE STATE AUTOMATA

2.1 ALPHABETS AND LANGUAGES

It is convenient to consider a set of finite length strings over some fixed finite alphabet to discuss finite state automata and regular expressions. An alphabet is any finite set. The arbitrary finite alphabet is denoted by the Greek letter C. The elements of $\sum$ are called the symbols and usually denote them by a, b, c, …, or 0, 1. Any finite length sequence w of symbols over the alphabet $\sum$ is called a string over $\sum$. The number of symbols in w is the length of the string and is denoted by $|w|$. A string of length 0 over any alphabet $\sum$ is called null string and denoted by symbol ε (epsilon). The set of all strings (including ε) over an alphabet $\sum$ is denoted as $\sum^*$. If $\sum$ is non-empty, then $\sum^*$ is an infinite set of finite length strings. Any subset of $\sum^*$ is called a language over $\sum$.

2.2 Finite State Automata

Let $\sum$ be a finite alphabet. An automaton over $\sum$ is a 5–tuple $A=(Q,\sum,I,\delta,F)$ where Q is a set of states, I is a subset of Q whose elements are called the initial states, F is a subset of Q whose elements are called final states, δ is called transition function defined from the Cartesian product $Q\times\sum$ to Q. The triplet (q,a,q'), where $\delta(q,a)=q'$ is called an edge, and the sequence $(q_i, a_i, q_{i+1}), i=1,2,...,n$ of consecutive edges is called a path. Then the word $w=a_1a_2....a_n$ is called the label of the path. A word w is said to be accepted by the automaton A if there is a path with label w such that $q_1\in I$ and $q_{n+1}\in F$. The set of all words accepted by an automaton A is called the language recognized by A. It is denoted by $L(A)$.

2.2.1 Definition

i. An Automaton A is called trim if for all $q\in Q$ there is at least one path through q beginning at an initial state and ending at a final state.

ii. An automaton A is called deterministic (DFA) if there is only one initial state and if for all $(q,a)\in Q\times\sum$, there is at most one state q' such that $\delta(q,a)=q'$, otherwise the automaton is called non-deterministic (NFA).

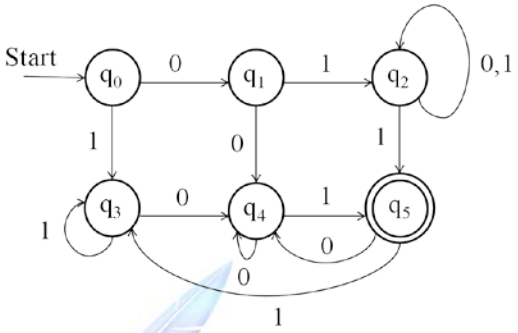
iii. An automaton is called standard or non-returning if there is only one initial state and there is no edge having the initial state on tail.

An automaton is also represented as a directed graph where each node represents a state and there is an arc labeled by an input symbol between the nodes. The initial states are denoted with the word start and final states are denoted by concentric circles.

For example, the various types of automata and the language they recognize are given below.

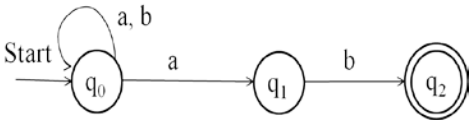
2.2.2 *Examples*

i. The automaton recognizing the language of 0's and 1's that begin with 01 or ends with 01, both is given below.

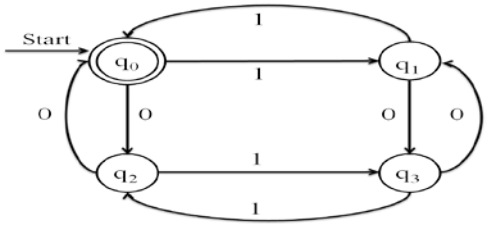


The above automaton is a trim, as well as DFA and standard.

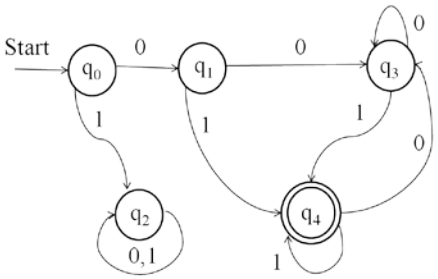
ii. The following automaton is an NFA recognizing the language of a's and b's that end with ab.



iii. The following is a DFA recognizing the language $L = \begin{cases} w/w \ \ has\, both\, even\ number\, of \\ 0's\, and\, even\ number\, of\ 1's \end{cases}$ over the alphabet $\sum = \{0,1\}$. This is not a standard automaton.



iv. The following is a DFA, but not trim which recognizing the language of 0's and 1's and that begins with 0, and end with 01.



2.3 CONSTRUCTION OF FINITE STATE AUTOMATA

The construction of FSA in general and the DFA in particular so as to accept the given language exactly is an interesting feature in the area of automata theory [3]. Naturally, the construction of NFA is quite simple, as it is assumed that NFA have the capability to guess something about input given, whereas a clear procedure is very much required to construct DFA so that the given DFA accepts only the strings for which it has been constructed, and rejects for all other strings. By rejection, it is meant that the process will not lead to final state when such strings are given as input.

As an example, let us construct a DFA that recognize the language of all strings with three consecutive zeros not necessarily at the end over the alphabet $\sum = \{0,1\}$. It can be seen that the strings over the given $\sum$ based on the given specifications can be classified as

i. The string processed so far has three consecutive 0's
ii. The string processed so far has two consecutive 0's
iii. The string processed so far has only one 0
iv. The string processed so far has no 0's

Thus, we need four states corresponding to these four cases. Among these, the state corresponding to the case of strings that is far away from the required form is assumed as the start state and the state representing the strings of the given language is assumed as the final state. In this example, we designate the states representing case iv and case i as start state q_0 and the final state q_3 respectively. Since, case ii is nearer to case i and case iii is nearer to case iv, they are to be represented as the states q_2 and q_1 respectively. Initially, let the DFA is at q_0 that is it has processed the strings having no 0's. Therefore, if the next input to be processed is 0, then we get a string from case iv to case iii, i.e., the DFA is at q_0 is to move from q_0 to q_1 on input 0. But, if 1 is the input, then again we have the same string of no 0's. i.e., the string of case iv.

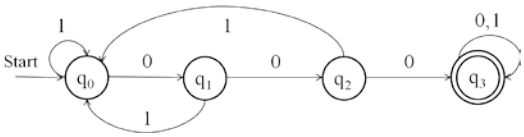
Therefore the DFA should remain at q_0 itself for the input 0. Thus, we define the transition function δ which directs the moves of the DFA at q_0 as $\delta(q_0,0)=q_1$ and $\delta(q_0,1)=q_0$. Next, let us assume that the DFA is at q_1, i.e., the DFA is at q_1 after processing the string that ends with a single 0. Therefore, if again 0 is the input then we get a string of case ii or if 1 is the input, we get a string of case iv. Thus, the DFA should move from q_1 to q_2 if 0 is the input or it should move from q_1 to q_0 if 1 is the input. Thus, the δ at q_1 is defined as $\delta(q_1,0)=q_2$ and $\delta(q_1,1)=q_0$. To define δ at q_2, we assume that the DFA is at q_2 after processing the string that ends with 00. If the next input is 0, we get the required string having three consecutive 0's but, if 1 is the input, we are at having a string of case iv, i.e., the string that ends with 1. Thus, we define δ at q_2 as $\delta(q_2,0)=q_3$ and $\delta(q_2,1)=q_0$. Finally, assume that the DFA is at q_3. It means that the DFA has already encountered the string consisting three consecutive 0's. Therefore, it will accept all the strings irrespective of the next inputs. Thus, we define δ at q_3 as $\delta(q_3,0)=q_3$ and $\delta(q_3,1)=q_3$. Thus, the DFA accepting the given languages is $D=(Q,\Sigma,\delta,q_0,F)$ where $Q=\{q_0,q_1,q_2,q_3\}$; $\Sigma=\{0,1\}$, q_0 is the start state; $F=\{q_3\}$ and is defined as

$$\delta(q_0,0)=q_1;\ \delta(q_0,1)=q_0;\ \delta(q_1,0)=q_2;\ \delta(q_1,1)=q_0$$
$$\delta(q_2,0)=q_3;\ \delta(q_2,1)=q_0;\ \delta(q_3,0)=q_3;\ \delta(q_3,1)=q_3.$$

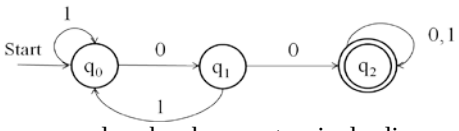
The transition table is given by

	0	1
→ q_0	q_1	q_0
q_1	q_2	q_0
q_2	q_3	q_0
q_3*	q_3	q_3

The transition diagram is given below.



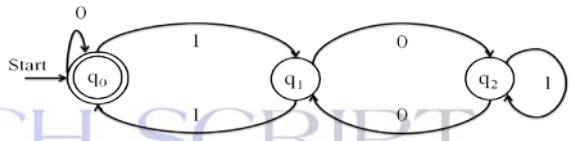
Similarly, one can easily construct the DFA that accepts the language of strings of 0's and 1's that has 00 as a substring as in the fig.



The above procedure has been extensively discussed in [3]. But this procedure will not suit for all types of languages. For example, consider the language of binary strings represent multiples of three. Here the strings are $0, 11, 110, 1001, 1100, 1111, 10010, \ldots$. It is hard to classify the strings as done in the previous example. Hence the strings are grouped, and the states are associated in the following way.

if the number represented by the string scanned so far is	then the DFA will be in state
0 mod 3	q_0
1 mod 3	q_1
2 mod 3	q_2

The transition diagram of the DFA is given below



Dexter C.Kozen [1] has proved by induction that the above DFA accepts exactly the language of binary strings represent multiples of three.

An approach was proposed by N.Murugesan and B.Samyukthavarthini [4] in 2013. This procedure is more helpful when the given set of strings do not possess any generic nature. This approach provides a systematic way to design a DFA from the easily constructed NFA's. The following are the steps involved in this approach.

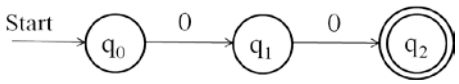
Step 1

Identify the set B_1 of basis strings of minimum length of the given language L.

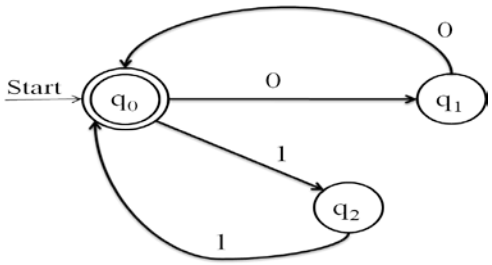
For example, if L_1 is the language of strings of 0's and 1's that consist at least two 0's, then, the set B_1 of basis strings of minimum length is $\{00\}$ in case of the language L_2 of strings of 0's and 1's that consist even number of 0's and even number of 1's, the set B_1 of basis strings is $\{\varepsilon, 00, 11\}$, where ε denotes the empty string.

Step 2

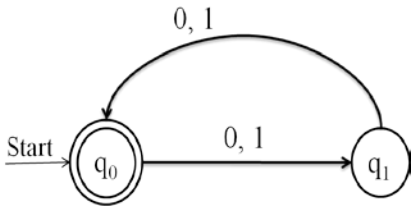
Design an NFA that accepts the strings of the set B_1. For the language L_1, the FA accepting B_1 is



and for the language L_2, the NFA accepting B_1 is



It can also be designed NFA, accepting B_1 of L_2 as



that too accepts ε, 00 and 11, but it also accepts the strings 01 and 10 that are not in L_2. This is the point to be kept in mind at every juncture of this type construction.

Step 3

If the NFA constructed in step 2 is a DFA, then this is the DFA accepting L.

The above two NFA constructed for L_1 and L_2 are not DFA. In the first case, the NFA not at all considering the input 1 at any state and in the second case only q_0 is the complete state in the sense that the NFA moves to q_1 and q_2 for the inputs 0 and 1 respectively at q_0. The moves not at all defined at q_1 for the input 0 at q_2.

Step 4

If the NFA is not a complete NFA, then generate the next set B_2 of strings from the set B_1 by incorporating the minimum number of input symbols to each and every element of B_1.

In the case of L_1, the set $B_2 = \{000, 001, 010, 100\}$ and for L_2, $B_2 = \{0000,\ 0011,\ 0101, 0110, 1001, 1010, 1100,\ 1111\}$.

Step 5

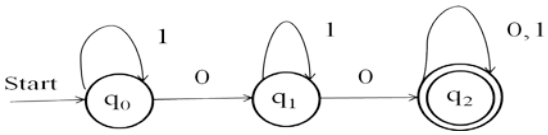
Modify the NFA constructed in step 2 that accepts the set B_2 together with B_1. It may accept some other strings of L but that should not lead to accept the strings not from the language L as discussed in step 2.

Step 6

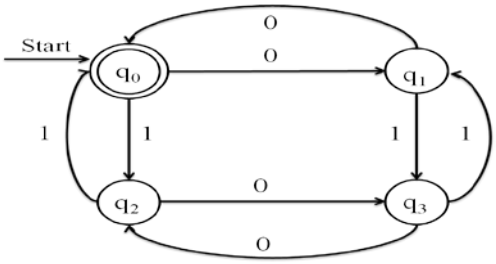
If the modified NFA is a DFA, then it is the DFA accepting the language L. Otherwise the above steps to be repeated from step 4 and to generate the set B's again and again until a complete DFA, is obtained.

It is quite surprising to see that a naturally defined language in one way corresponds so closely to naturally define a theoretical machine in another way. Is this merely a coincidence? Whenever the NFA becomes the DFA, then it accepts the entire language L not merely the set $\bigcup_{i=1}^{n} B_i$.

The modified NFA accepting the set B_2 together with B_1 of L_1 is



and the same for L_2 is



Note that both NFA become DFA and see that they accept the language L_1 and L_2 respectively. Now let us consider another example which is little harder than the above two examples.

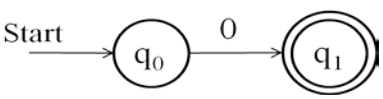
Let $L = \begin{cases} x \in (0,1)^* \ / \ x \text{ represents a multiple three} \\ \text{in binary allowing leading 0's} \end{cases}$

* indicates that the L contains empty string also.

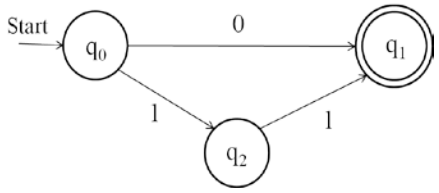
The following is the list of binary strings that represents multiples of 3 and their decimal equivalent.

Binary	Decimal equivalent	Binary	Decimal equivalent
0	11	1100	12
11	3	1111	15
110	6	10010	18
1001	9	10101	21

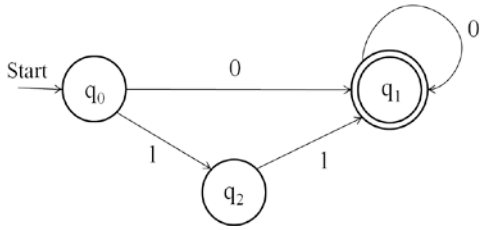
Let $B_1 = \{0\}$. The NFA accepting B_1 is



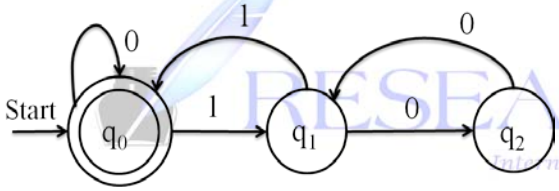
Let $B_2 = \{11\}$. The NFA accepting $B_1 \cup B_2$ is



Let $B_3 = \{110\}$. The NFA accepting $B_1 \cup B_2 \cup B_3$ is

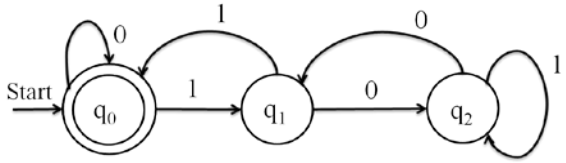


This NFA accepts also the strings $11000(=24)$, $110000(=48)$ and so on…., that are also multiples of 3. Let $B_4 = \{1001, 111\}$. In order to make the NFA to accept 1001, the only possibility is that we have to form a loop at q_2 for the input 0. If we do it, then it will also accept the string $101(=5)$ which is not a multiple of 3. Therefore a complete reorganization of states is necessary. The modified structure may be drawn as follows.

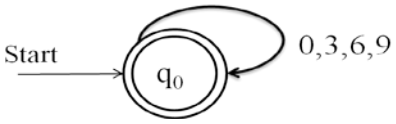


Now consider $B_5 = \{10010, 10101\}$, the NFA already accepting 10010, and if we form a loop at q_2 for the input 1, then it will accept 10101 and also 101101 (=45), 1011101(=93), … and so on. These all are again multiples of 3.

The important thing is that it has become a DFA and so we can easily verify that it accepts exactly the given language L.

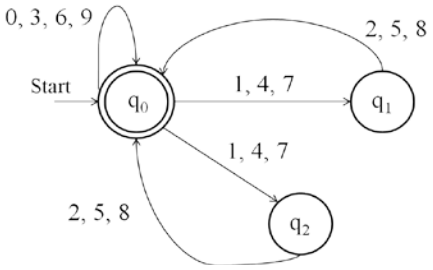


As another example, let us consider the language L that recognizes the multiples of 3 in the decimal system. Initially, let $B_1 = \{0, 3, 6, 9\}$ and by intuition we realize that the repeated symbols are also belongs to L. i.e., 000…, 333….666….999… are multiples of 3. Thus the NFA accepting B_1 is



Let $B_2 = \{12, 15, 18, 21, 24, 27, 30, 33, 36, 39, 42, 45, 48, 51, 54, 57, 60, 63, 66, 69, 72, 75, 78, 81, 84, 87, 90, 93, 96, 99\}$

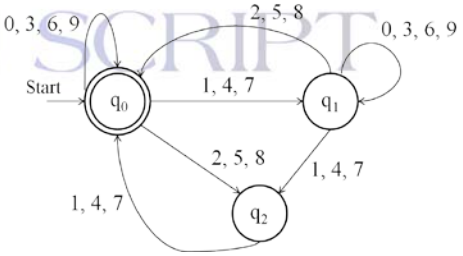
The FA accepting $B_1 \cup B_2$ is



The above NFA also recognize the numbers that end with 0, 3, 6, 9 from third digit onwards.

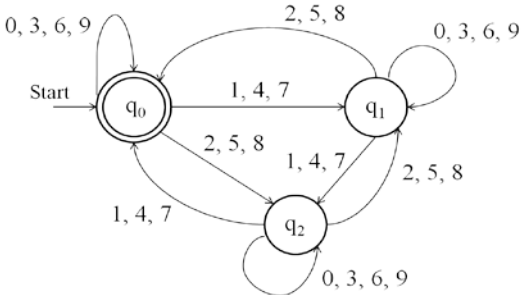
If $B_3 = \{102, 105, 108, 111, 114, 117, 120, 123, 126, 129, 132, 135, 138, 141, 144, 147, 150, 153, 156, 159, 162, 165, 168, 171, 174, 177, 180, 183, 186, 189, 192, 195, 198\}$

then the NFA accepting $B_1 \cup B_2 \cup B_3$ is



Let $B_4 = \{201, 204, 207, 210, 213, 216, 219, 222, 225, 228, 231, 234, 237, 240, 243, 246, 249, 252, 255, 258, 261, 264, 267, 270, 273, 276, 279, 282, 285, 288, 291, 294, 297, 300\}$

The NFA accepting $B_1 \cup B_2 \cup B_3 \cup B_4$ is



The NFA becomes a DFA, and it accepts all the numbers of multiples of 3. For example, let us check this for the number 37068.

Input	Current state	Input	Current state
-	q_0	0	q_1
3	q_0	6	q_1
7	q_1	8	q_0

Since q_0 is the final state, 37098 is a multiple of 3.

3 CONCLUSION

In this paper, the languages of binary strings representing DFA have been discussed. From the NFA construction, An approach proposed by N.Murugesan et al., is more helpful when the given set of strings do not possess any generic nature.

REFERENCES

[1]. Dexter C. Kozen, *Automata and Computability*, Springer-Verlag, New York, Inc., 1977.
[2]. Hopcroft J.E and Ullman J.D, , *Introduction to Automata Theory, Languages and Computation*, Addison – Wesley, 1979.
[3]. Murugesan N, *Principles of Automata theory and Computation*, 2004, Sahithi Publications.
[4]. Murugesan N, and Samyukthavarthini B, *A Study on Various types of Automata*, M.Phil., Dissertation, Bharathiar University, 2013.
[5]. Yu S, Regular Languages, in: A.Salaomaa, eds., Handbook of Formal Languages, Vol. I, Springer-Verlag, Berlin, 1997, 41 – 110.