DFA (DETERMINISTIC FINITE AUTOMATA)

- DFA refers to deterministic finite automata. Deterministic refers to the uniqueness of the computation. The finite automata are called deterministic finite automata if the machine is read an input string one symbol at a time.
- In DFA, there is only one path for specific input from the current state to the next state.
- DFA does not accept the null move, i.e., the DFA cannot change state without any input character.
- DFA can contain multiple final states. It is used in Lexical Analysis in Compiler.

In the following diagram, we can see that from state q0 for input a, there is only one path which is going to q1. Similarly, from q0, there is only one path for input b going to q2.

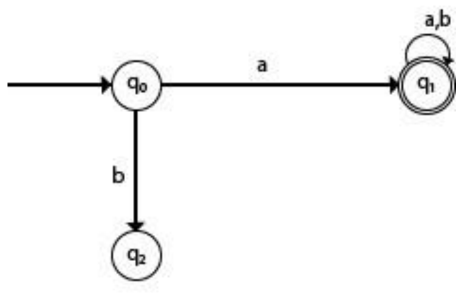


Fig:- DFA

Formal Definition of DFA

A DFA is a collection of 5-tuples same as we described in the definition of FA.

1. Q: finite set of states
2. ∑: finite set of the input symbol
3. q0: initial state
4. F: **final** state
5. δ: Transition function

Transition function can be defined as:

1. $\delta: Q \times \sum \rightarrow Q$

Graphical Representation of DFA

A DFA can be represented by digraphs called state diagram. In which:

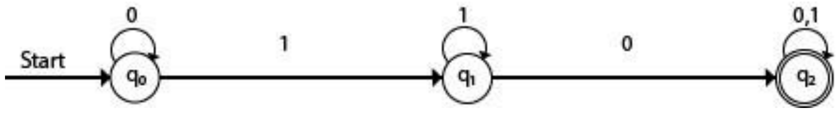
1. The state is represented by vertices.
2. The arc labeled with an input character show the transitions.
3. The initial state is marked with an arrow.
4. The final state is denoted by a double circle.

Example :

1. Q = {q0, q1, q2}
2. ∑ = {0, 1}
3. q0 = {q0}
4. F = {q2}

Solution:

Transition Diagram:



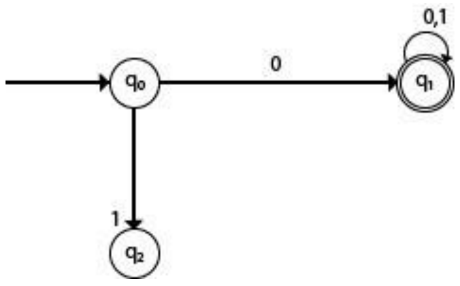
Transition Table:

Present State	Next state for Input 0	Next State of Input 1
→q0	q0	q1
q1	q2	q1
*q2	q2	q2

Example :

DFA with ∑ = {0, 1} accepts all starting with 0.

Solution:



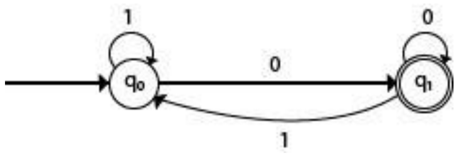
Explanation:

- In the above diagram, we can see that on given 0 as input to DFA in state q0 the DFA changes state to q1 and always go to final state q1 on starting input 0. It can accept 00, 01, 000, 001 .....etc. It cant accept any s tring which starts with 1, because it will never go to final state on a string starting with 1.

Example :

DFA with ∑ = {0, 1} accepts all ending with 0.

Solution:



Explanation:

In the above diagram, we can see that on given 0 as input to DFA in state q0, the DFA changes state to q1. It can accept any string which ends with 0 like 00, 10, 110, 100....etc. It cant accept any string which ends with 1, because it will never go to the final state q1 on 1 input, so the string ending with 1, will not be accepted or will be reject d.
e

Examples of DFA

Example :

Design a FA with ∑ = {0, 1} accepts those string which starts with 1 and ends with 0.

Solution:

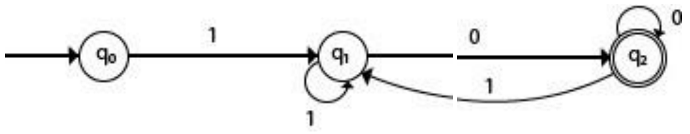
The FA will have a start state q0 fro m which only the edge with input 1 will go to the next state.

In state q1, if we read 1, we will be in state q1, but if we read 0 at state q1, we will reach to state q2 which is the final state. In state q , if we read either 0 or 1, we will go to q2 state or q1 state respectively. Note that if the input ends with 0, it will be in the final state.

Example :

Design a FA with ∑ = {0, 1} accepts the only input 101.

Solution:



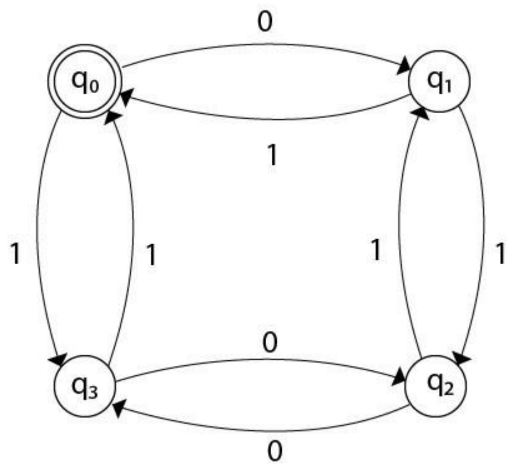
In the given solution, we can see that only input 101 will be accepted. Hence, for input 101, there is no other path shown for other input.

Example :

Design FA with ∑ = {0, 1} accepts even number of 0's and even number of 1's.

Solution:

This FA will consider four different stages for input 0 and input 1. The stages could be:



Here q0 is a start state and the final state also. Note carefully that a symmetry of 0's and 1's is maintained. We can associate meanings to each state as:

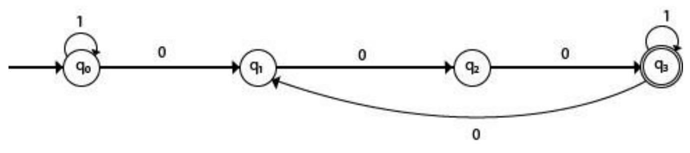
q0: state of even number of 0's and even number of 1's.q1: state of odd number of 0's and even number of 1's.q2: state of odd number of 0's and odd number of 1's.q3: state of even number of 0's and odd number of 1's.

Example 4:

Design FA with ∑ = {0, 1} accepts the set of all strings with three consecutive 0's.

Solution:

The strings that will be generated for this particular languages are 000, 0001, 1000, 10001, ...in which 0 always appears in a clump of 3. The transition graph is as follows:



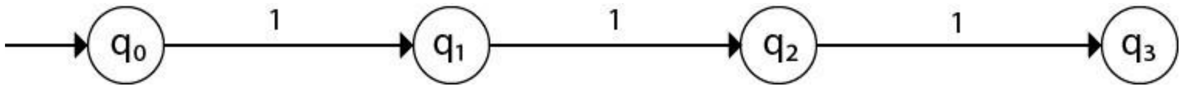
Note that the sequence of triple zeros is maintained to reach the final state.

Example 5:

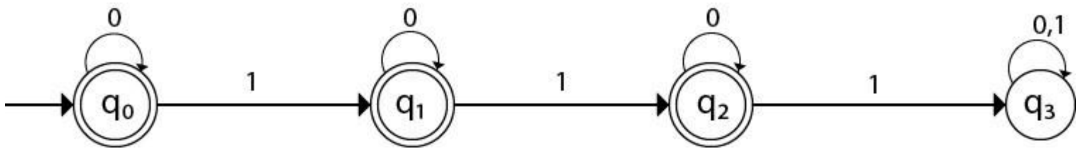
Design a DFA L(M) = {w | w ε {0, 1}*} and W is a string that does not contain consecutive 1's.

Solution:

When three consecutive 1's occur the DFA will be:



Here two consecutive 1's or single 1 is acceptable, hence



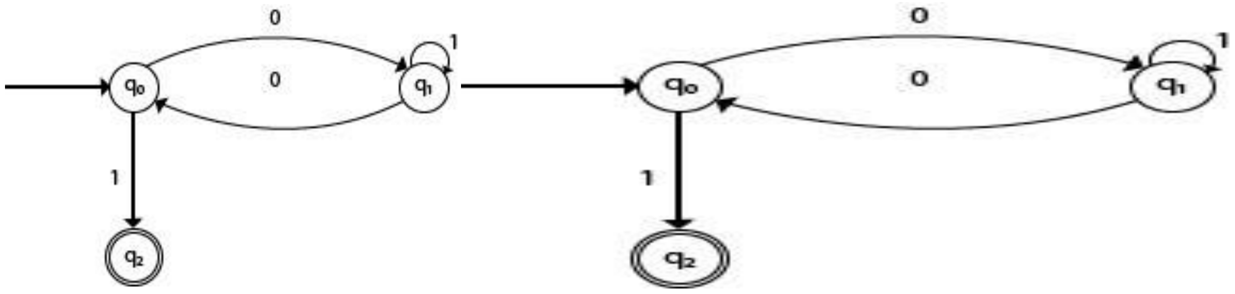
The stages q0, q1, q2 are the final states. The DFA will generate the strings that do not contain consecutive 1's like 10, 110, 101, etc.

Example 6:

Design a FA with ∑ = {0, 1} accepts the strings with an even number of 0's followed by single 1.

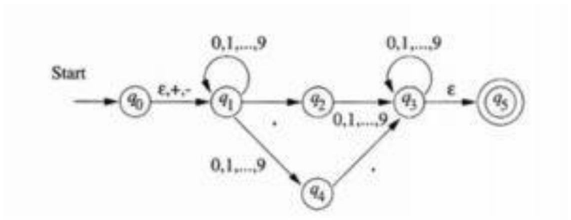
Solution:

The DFA can be shown by a transition diagram as:



FINITE AUTOMATA WITH ε –TRANSITIONS

An informal treatment of €-NFA's, using transition diagrams with f allowed as a label. In the examples to follow, think of the automaton as accepting those sequences of labels along paths from the start state to an accepting state. However, each e along a path is "invisible" j i.e., it contributes nothing to the string along the path.



[Source: "J.E.Hopcroft, R.Motwani and J.D Ullman, Introduction to Automata Theory, Languages and Computations, Second Edition, Pearson Education, 2003]

In Fig. is an €-NFAthat Accepts decimal numbers consisting of:2. A string of digits,
1. An optional + or – sign,
3. A decimal point, and
4. Another string of digits. Either this string of digits, or the string (2) can be empty, but at least one of the two strings of digits must be nonempty.

Nondeterministic Finite Automata with ε transitions (ε-NFA)

• A Non-Deterministic Finite Automata with ε transitions is a 5-tuple (Q, Σ, qo, δ, F)

where – Q is a finite set (of states)

Σ is a finite alphabet of symbols

qo ∈ Q is the start state

F ⊆ Q is the set of accepting states

δ is a function from Q x (Σ ∪ {ε}) to 2Q (transition function)

Transition function – δ is a function from

Q x (Σ ∪ {ε}) to 2Q

δ (q, a) = subset of Q (possibly empty)

In our example

• δ (q1, 0) = {q1, q4}

• δ (q1, .) = {q1}

- δ (q1, +) = ∅
- δ (q0, ε) = {q1}

Transition function on a string

It is a function from Q x Σ* to 2Q – (q, x) = subset of Q (possibly empty)

It Set of all states that the machine can be in, upon following all possible paths on input x

Here we need to consider all paths that include the use of ε transitions

E CLOSURE

– Before defining the transition function on a string ((q,x)), it is useful to first define what is known as the ε closure.

– Given a set of states S, the ε closure will give the set of states reachable from each state in S using only ε transitions

ε closure: Recursive definition

– Let M = (Q, Σ, qo, δ, F) be a ε-NFA

– Let S be a subset of Q

– The ε closure, denotes ECLOSE(S) is defined:

- For each state p ∈ S, p ∈ ECLOSE(S)
- For any q ∈ ECLOSE(S), every element of δ(q, ε) ∈ ECLOSE(S)
- No other elements of Q are in ECLOSE(S)

ε-Closure

- ε-Closure : Algorithm

– Since we know that ECLOSE(S) is finite, we can convert the recursive definition to an algorithm.

– To find ECLOSE(S) where S is a subset of Q

– Let T = S – While (T does not change) do

- Add all elements of δ(q, ε) where q ∈ T – ECLOSE(S) = T

FINITE AUTOMATA

- Finite automata are used to recognize patterns.
- It takes the string of symbol as input and changes its state accordingly. When the desired symbol is found, then the transition occurs.
- At the time of transition, the automata can either move to the next state or stay in the same state.
- Finite automata have two states, **Accept state** or **Reject state**. When the input string is processed successfully, and the automata reached its final state, then it will accept.

Formal Definition of FA

A finite automaton is a collection of 5-tuple (Q, ∑, δ, q0, F), where:

1. Q: finite set of states
2. ∑: finite set of the input symbol
3. q0: initial state
4. F: **final** state
5. δ: Transition function

Finite Automata Model:

Finite automata can be represented by input tape and finite control.

Input tape: It is a linear tape having some number of cells. Each input symbol is placed in each cell.

Finite control: The finite control decides the next state on receiving particular input from input tape. The tape reader reads the cells one by one from left to right, and at a time only one input symbol is read.

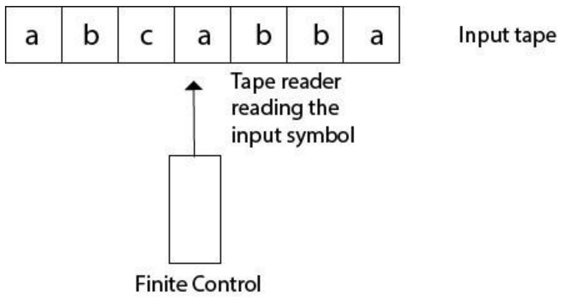


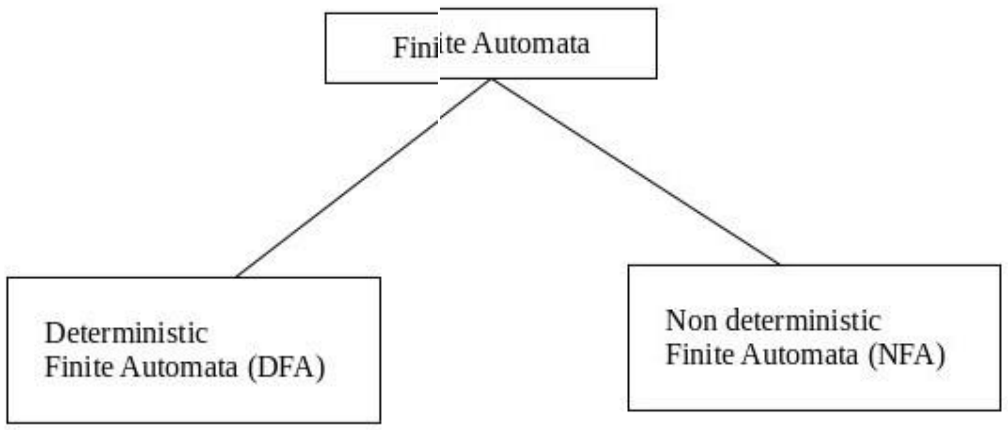
Fig :- Finite automata model

[Source: "J.E.Hopcroft, R.Motwani and J.D Ullman, Introduction to Automata Theory, Languages and Computations, Second Edition,Pearson Education, 2003]

Types of Automata:

There are two types of finite automata:

1. DFA(deterministic finite automata)
2. NFA(non-deterministic finite automata)


Finite Automata
Deterministic Finite Automata (DFA)
Non deterministic Finite Automata (NFA)

1. DFA

DFA refers to deterministic finite automata. Deterministic refers to the uniqueness of the computation.In the DFA, the machine goes to one state only for a particular input character. DFA does not acceptthe null move.

2. NFA

NFA stands for non-deterministic finite automata. It is used to transmit any number of states for a particular input. It can accept the null move.

Some important points about DFA and NFA:

1. Every DFA is NFA, but NFA is not DFA.
2. There can be multiple final states in both NFA and DFA.
3. DFA is used in Lexical Analysis in Compiler.
4. NFA is more of a theoretical concept.

Transition Diagram

A transition diagram or state transition diagram is a directed graph which can follows: be constructed as

- There is a node for each state in Q, which is represented by the circle.

- There is a directed edge from node q to node p labeled a if δ(q, a) = p.
- In the start state, there is an arrow with no source.
- Accepting states or final states are indicating by a double circle.

Some Notations that are used in the transition diagram:

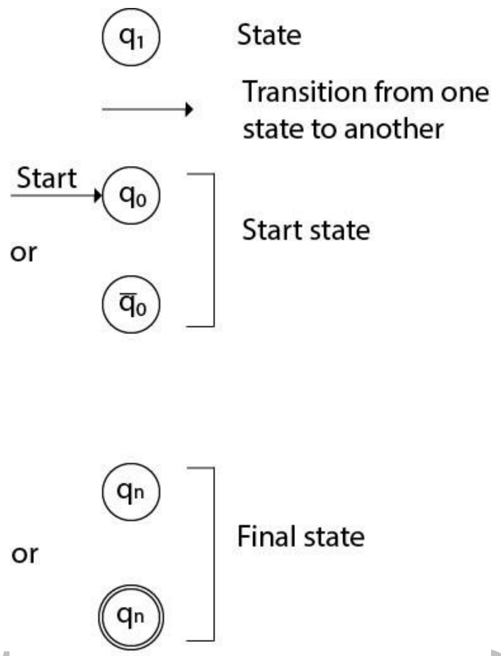


Fig:- Notations

There is a description of how a DFA operates:

1. In DFA, the input to the automata can be any string. Now, put a pointer to the start state q and read the input string w from left to right and move the pointer according to the transition function, δ. We can read one symbol at a time. If the next symbol of string w is a and the pointer is on state p, move the pointer to δ(p, a). When the end of the input string w is encountered, then the pointer is on some state F.

2. The string w is said to be accepted by the DFA if r ∈ F that means the input string w is processed successfully and the automata reached its final state. The string is said to be rejected by DFA if r ∉ F.

Example :

DFA with ∑ = {0, 1} accepts all strings starting with 1.

Solution:

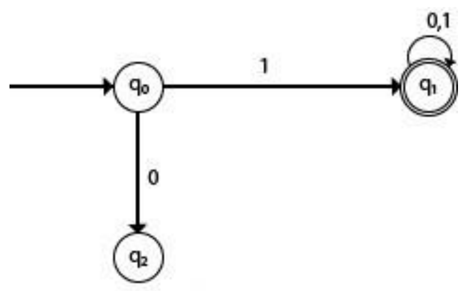


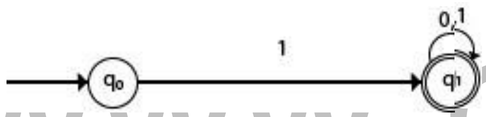
Fig: Transition diagram

The finite automata can be represented using a transition graph. In the above dia ram, the machine g
initially is in start state q0 then on r receiving 0, the machine changes eceiving input 1 the machine changes its state to q1. From q0 on state to q2, which is the dead state. From q1
its on receiving input 0,
1 the machine changes its state to q generated are 10, 11, 110, 1011, which is the final state. The possible input s trings that can be
111... ..., that means all string starts with 1.

Example :

NFA with ∑ = {0, 1} accepts all strin gs starting with 1.

Solution:



The NFA can be represented using a transition graph. In the above diagram, the ma hine initially is in start state q0 then on receiving input 1 cthe machine changes its state to q1. From q1 on receiving input 0, 1 the machine changes its state to q1. The possible input string that can be generated is 10, 11, 110,101, 111 , that means all string starts with 1.

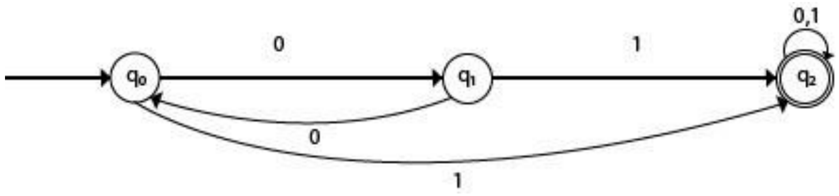
Transition Table

The transition table is basically a tabular representation of the transition function. It takes two arguments (a state and a symbol) and returns a state (the "next state").

A transition table is represented by the following things:

- Columns correspond to input symbols.
- Rows correspond to states.
- Entries correspond to the next state.
- The start state is denoted by an arrow with no source.
- The accept state is denoted b a star.

Example 1:



Solution:

Transition table of given DFA is as follows:

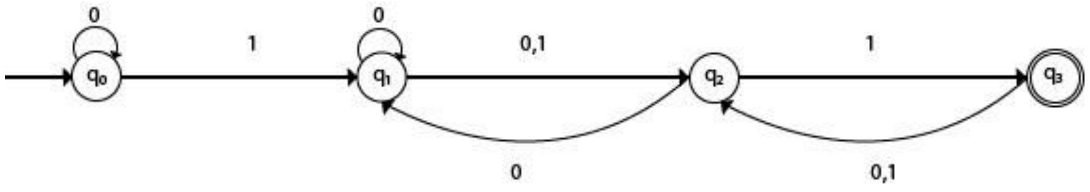
Present State	Next state for Input 0	Next State of Input 1
→q0	q1	q2
q1	q0	q2
*q2	q2	q2

Explanation:

- In the above table, the first column indicates all the current states. Under column 0 and 1, the next states are shown.
- The first row of the transition table can be read as, when the current state is q0, on input 0 the next state will be q1 and on input 1 the next state will be q2.
- In the second row, when the current state is q1, on input 0, the next state will be q0, and on 1 input the next state will be q2.
- In the third row, when the current state is q2 on input 0, the next state will be q2, and on 1 input the next state will be q2.
- The arrow marked to q0 indicates that it is a start state and circle marked to q2 indicates that it

  is a final state.

Example 2:



Solution:

Transition table of given NFA is as follows:

Present State	Next state for Input 0	Next State of Input 1
→q0	q0	q1
q1	q1, q2	q2
q2	q1	q3
*q3	q2	q2

INTRODUCTION TO FORMAL PROOF & ADDITIONAL FORMS OF PROOF

The formal proof can use

1.Inductive proof

2.Deductive proof

Inductive proof

It is a recursive kind of proof which consist of sequence of parameterized ststements that use the statement ,itself with lower values of its parameter

Deductive proof

It consist of sequence of statements given with logical reasoning ,in order to prove the first statement is called hypothesis.

Various forms of Proof

The additional forms of proof can be explained with the help of examples

1)Proof about sets

2)Proof by contracdiction

3)Proof by counter example

Proof about sets

The set is a collection of elements or items .By giving proof about the set we try to prove certain properties of the sets .

For example :

If there are two expressions A& B and we want to prove that both expressions A&B are equivalent

Let

PUQ = QUP

(A) (B)

Then prove A=B ,we need to prove

PUQ=QUP

That means an element x is in A if and only if it is in B

Proving LHS

1.x is in PUQ

2.x is in P or x is in Q

3.x is in Q or X is in P

4.x is in QUP

Like that we can prove RHS

1.x is in Q or X is in P

2.x is in QUP

3.x is in PUQ

4.x is in P or x is in Q

Hence

PUQ=QUP ,Thus A=B is true as element x is in B if and only if x is in A

Proof by contradiction

In this type of proof ,for the statement of the form is A & B ,we start with statement A is not true and thus by assuming false A.We try to get the conclusion of statement B .

When it becomes impossible to reach statement B ,we contradict ourself and accept that A is true

Example :

Prove that PUQ=QUP

Proof

Initially we assume that PUQ=QUP is not true.

Now consider that x is in Q or x is in PUQ

But this also implies that x is in QUP according to definition of union

Hence the assumption which are made initailly is false

Thus

PUQ=QUP is proved

Proof by counter Example

In order to prove certain statements ,we need to see all possible conditions in which that statement remains true .There are some situations in which that statement cannot be true

Example

A mod B = B mod A

Proof

Consider A=2 and B=3

2 mod 3 is not equal to 3 mod 2

NFA (NON-DETERMINISTIC FINITE AUTOMATA)

- NFA stands for non-deterministic finite automata. It is easy to construct an NFA than DFA for a given regular language.
- The finite automata are called NFA when there exist many paths for specific input from the current state to the next state.
- Every NFA is not DFA, but each NFA can be translated into DFA.
- NFA is defined in the same way as DFA but with the following two exceptions, it contains multiple next states, and it contains ε transition.

In the following image, we can see that from state q0 for input a, there are two next states q1 and q2,similarly, from q0 for input b, the next states are q0 and q1. Thus it is not fixed or determined that with a particular input where to go next. Hence this FA is called non-deterministic finite automata.

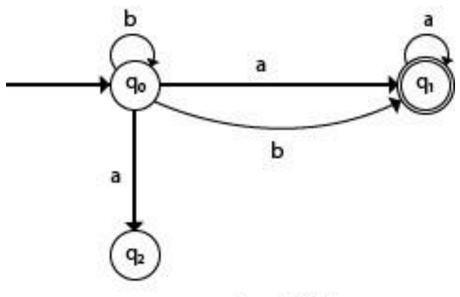


Fig:- NDFA

Formal definition of NFA:

NFA also has five states same as DFA, but with different transition function, as shown follows:

$\delta: Q \times \sum \rightarrow 2^Q$

where,

1. Q: finite set of states
2. ∑: finite set of the input symbol
3. q0: initial state
4. F: **final** state
5. δ: Transition function

Graphical Representation of an NFA

An NFA can be represented by digraphs called state diagram. In which:

1. The state is represented by vertices.
2. The arc labeled with an input character show the transitions.
3. The initial state is marked with an arrow.

4. The final state is denoted by the double circle.

Example 1:

1. Q = {q0, q1, q2}
2. ∑ = {0, 1}
3. q0 = {q0}
4. F = {q2}

Solution:

Transition diagram:

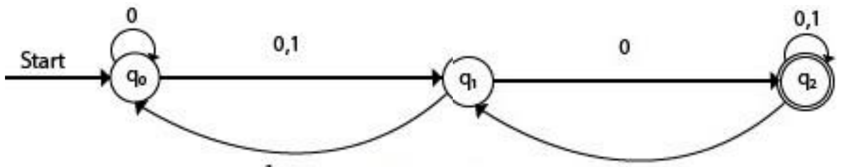


Fig: NFA

Transition Table:

Present State	**Next state for Input 0**	**Next State of Input 1**
→q0	q0, q1	q1
q1	q2	q0
*q2	q2	q1, q2

In the above diagram, we can see that when the current state is q0, on input 0, the next state will be q0 or q1, and on 1 input the next state will be q1. When the current state is q1, on input 0 the next state will be q2 and on 1 input, the next state will be q0. When the current state is q2, on 0 input the next state is q2, and on 1 input the next st a te will be q1 or q2.

Example :

NFA with ∑ = {0, 1} accepts all strings with 01.

Solution:

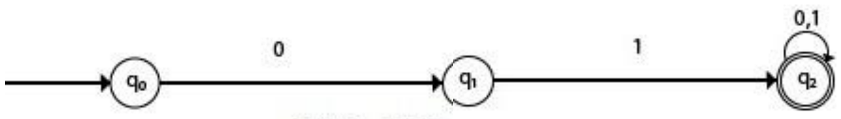


Fig: NFA

Transition Table:

Present State	Next state for Input 0	Next State of Input 1
→q0	q1	ε
q1	ε	q2
*q2	q2	q2

Example :

NFA with ∑ = {0, 1} and accept all string of length atleast 2.

Solution:

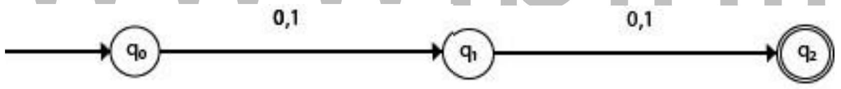


Fig: NFA

Transition Table:

Present State	Next state for Input 0	Next State of Input 1
→q0	q1	q1
q1	q2	q2
*q2	ε	ε

Examples of NFA

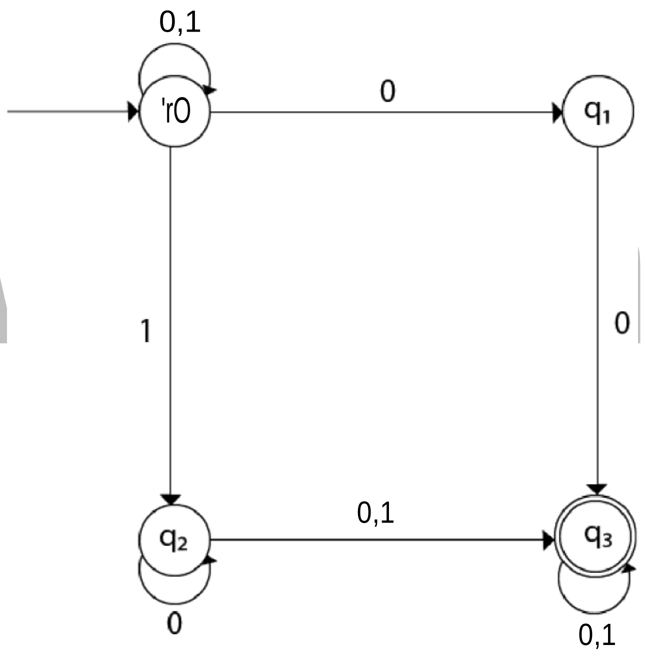
Example :

Design a NFA for the transition table as given below:

Present State	**0**	**1**
→q0	q0, q1	q0, q2
q1	q3	ε
q2	q2, q3	q3
→q3	q3	q3

Solution:

The transition diagram can be drawn by using the mapping function as given in the table.



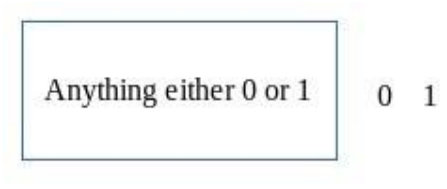
Here,

1. δ(q0, 0) = {q0, q1}
2. δ(q0, 1) = {q0, q2}
3. Then, δ(q1, 0) = {q3}
4. Then, δ(q2, 0) = {q2, q3}
5. δ(q2, 1) = {q3}
6. Then, δ(q3, 0) = {q3}
7. δ(q3, 1) = {q3}

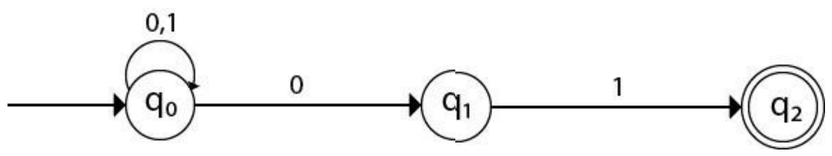
Example 2:

Design an NFA with ∑ = {0, 1} accepts all string ending with 01.

Solution:



Hence, NFA would be:

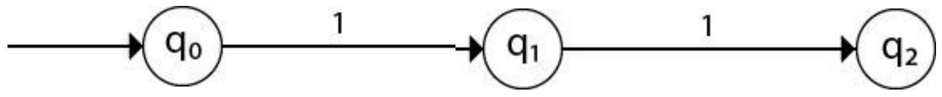


Example 3:

Design an NFA with ∑ = {0, 1} in which double '1' is followed by double '0'.

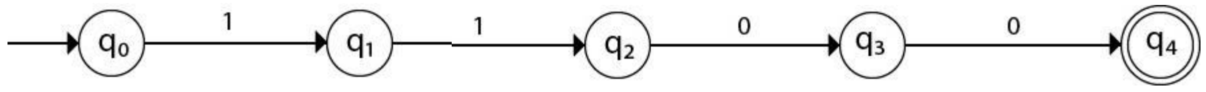
Solution:

The FA with double 1 is as follows:



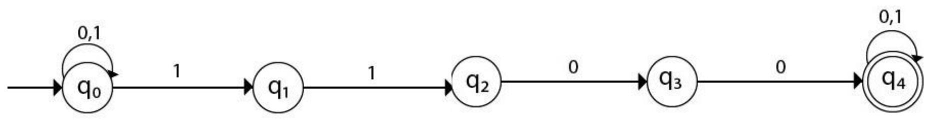
It should be immediately followed b y double 0.

Then,



Now before double 1, there can be string of 0 and 1. a ny string of 0 and 1. Similarly, after double 0, there can be any

Hence the NFA becomes:



Now considering the string 0110001 1

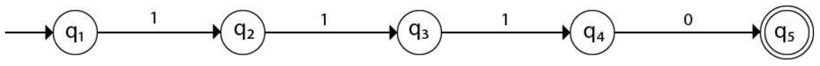
1. q0 → q1 → q2 → q3 → q4 → q4 → q4 → q4

Example 4:

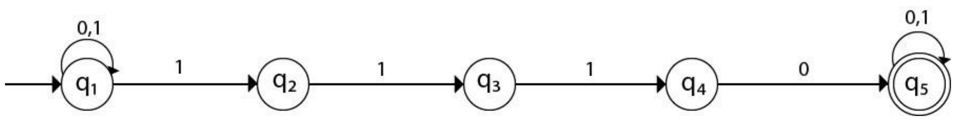
Design an NFA in which all the string contain a substring 1110.

Solution:

The language consists of all the string containing substring 1010. The partial transition diagram can be:



Now as 1010 could be the substring. Hence we will add the inputs 0's and 1's so that the substring 1010 of the language can be maintained. Hence the NFA becomes:



Transition table for the above transition diagram can be given below:

Present State	**0**	**1**
→q1	q1	q1, q2
q2		q3
q3		q4
q4	q5	
*q5	q5	q5

Consider a string 111010,

1. δ(q1, 111010) = δ(q1, 1100)
2. = δ(q1, 100)
3. = δ(q2, 00)

Got stuck! As there is no path from q2 for input symbol 0. We can process string 111010 in another way.

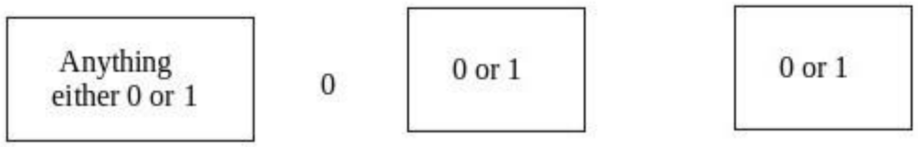
1. δ(q1, 111010) = δ(q2, 1100)
2. = δ(q3, 100)
3. = δ(q4, 00)
4. = δ(q5, 0)
5. = δ(q5, ε)

As state q5 is the accept state. We get the complete scanned, and we reached to the fi nal state.

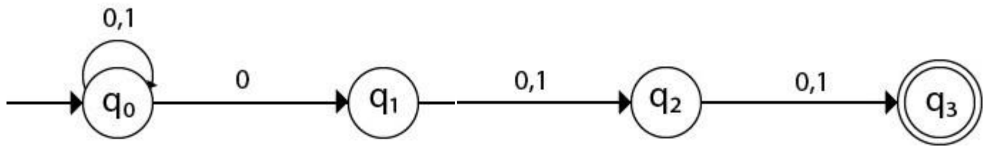
Example 5:

Design an NFA with ∑ = {0, 1} ac always 0. cepts all string in which the third symbol fro m the right end is

Solution:



Thus we get the third symbol from the right end as '0' always. The NFA can be:



The above image is an NFA because in state q0 with input 0, we can either go to state q0 or q1.