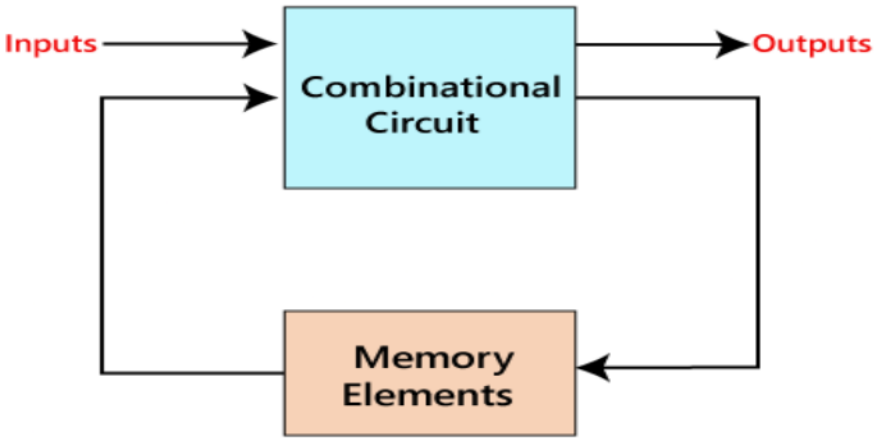
Unit 3: Finite state Automata, Grammars and Languages

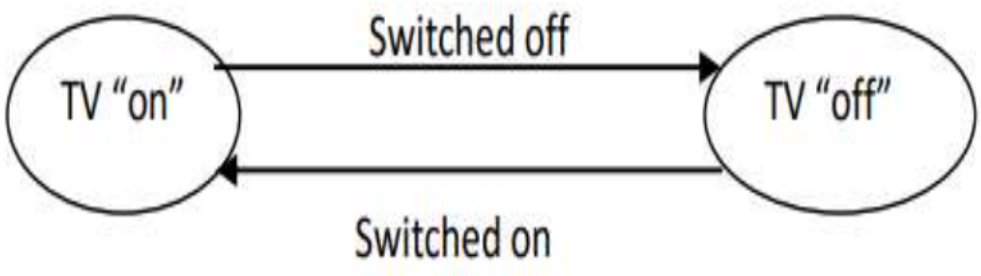
Sequential Circuit:

The sequential circuit is a special type of circuit that has a series of inputs and outputs. The outputs of the sequential circuits depend on both the combination of present inputs and previous outputs. The previous output is treated as the present state. So, the sequential circuit contains the combinational circuit and its memory storage elements.



Finite state machine

- Finite state machine is used to recognize patterns.
- Finite automata machine takes the string of symbol as input and changes its state accordingly. In the input, when a desired symbol is found then the transition occurs.
- While transition, the automata can either move to the next state or stay in the same state.
- FA has two states: accept state or reject state. When the input string is successfully processed and the automata reached its final state then it will accept.



A finite automata consists of following:

Q: finite set of states

∑: finite set of input symbol

q_0: initial state

F: final state

δ: Transition function

Transition function can be define as

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

DFA

DFA stands for Deterministic Finite Automata. Deterministic refers to the uniqueness of the computation. In DFA, the input character goes to one state only. DFA doesn't accept the null move that means the DFA cannot change state without any input character.

A deterministic finite automaton is defined by a quintuple (5-tuple) as (Q, Σ, δ, q0, F).

Where,

Q = Finite set of states,

Σ = Finite set of input symbols,

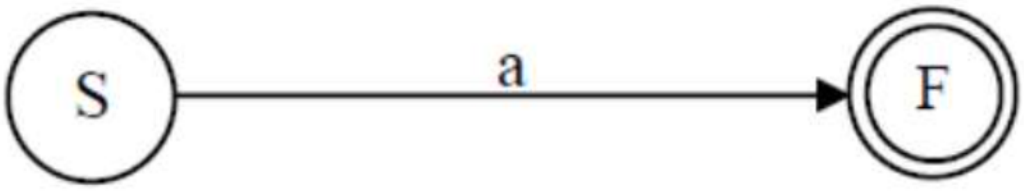
δ = A transition function that maps Q × Σ -> Q

q0 = A start state; q0 ∈ Q

F = Set of final states; F ⊆ Q.

A transition function δ that takes as arguments a state and an input symbol and returns a state. In our diagram, δ is represented by arcs between states and the labels on the arcs.

For example



If s is a state and a is an input symbol then δ(p,a) is that state q such that there are arcs labled ‘a’ from p to q.

General Notations of DFA

There are two preferred notations for describing this class of automata;

- Transition Table
- Transition Diagram

Transition Table:

Transition table is a conventional, tabular representation of the transition function δ that takes the arguments from Q × Σ & returns a value which is one of the states of the automation. The row of the table corresponds to the states while column corresponds to the input symbol. The starting state in the table is represented by -> followed by the state i.e. ->q, for q being start state, whereas final state as *q, for q being final state.

Transition Diagram:

A transition diagram of a DFA is a graphical representation where; (or is a graph) - For each state in Q, there is a node represented by circle, - For each state q in Q and each input a in Σ, if δ (q, a) = p then there is an arc from node q to p labeled a in the transition diagram. If more than one input symbol cause the transition from state q to p then arc from q to p is labeled by a list of those symbols. - The start state is labeled by an arrow written with “start” on the node. - The final or accepting state is marked by double circle.

For example:

Consider a DFA;

Q = {q0, q1, q2, q3}

Σ = {0, 1}

q0 = q0

F = {q0}

δ = Q × Σ -> Q

Then the transition table for above DFA is as follows:

δ	0	1
* -> q_0	q2	q1
q1	q3	q0
q2	q0	q3
q3	q1	q2

Table: Transition Tabel

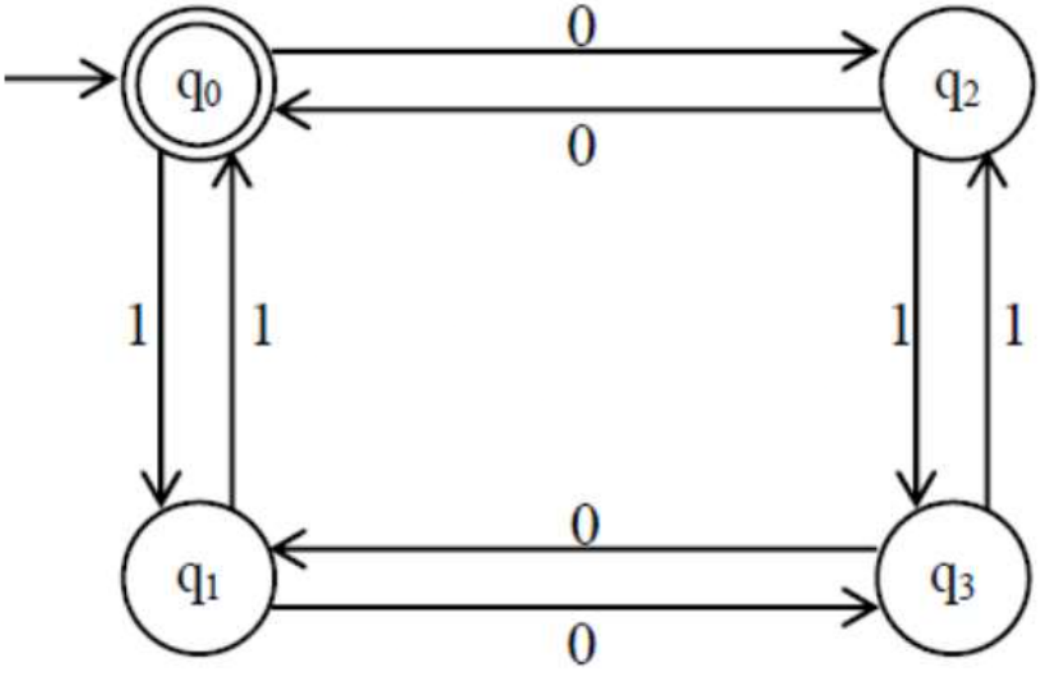


Fig: Transition Diagram

Question:

1. Design a FA with ∑ = {0, 1} accepts those string which starts with 1 and ends with 0.
2. Construct a DFA, that accepts all the strings over Σ = {a, b} that do not end with ba.
3. Design a FA with ∑ = {0, 1} accepts the only input 101.
4. Design FA with ∑ = {0, 1} accepts the set of all strings with three consecutive 0's.
5. DFA over {0, 1} accepting {1, 01}.
6. DFA over {a, b} that accepts the strings ending with abb.

[Refer Class notes for design and more DFA design]

How a DFA process strings?

The first thing we need to understand about a DFA is how DFA decides whether or not to "accept" a sequence of input symbols. The "language" of the DFA is the set of all symbols that the DFA accepts. Suppose a1, a2, …… an is a sequence of input symbols. We start out with the DFA in its start state, q0. We consult the transition function δ also for this purpose. Say δ (q0, a1) = q1 to find the state that the DFA enters after processing the first input symbol a1. We then process the next input symbol a2, by evaluating δ (q1, a2); suppose this state be q2. We continue in this manner, finding states q3, q4, …, qn. such that δ (qi-1, ai) = qi for each i. if qn is a member of F, then input a1, a2, --- an is accepted & if not then it is rejected.

NFA (Non-Deterministic finite automata)

- The finite automata are called NFA when there exist many paths for specific input from the current state to the next state.

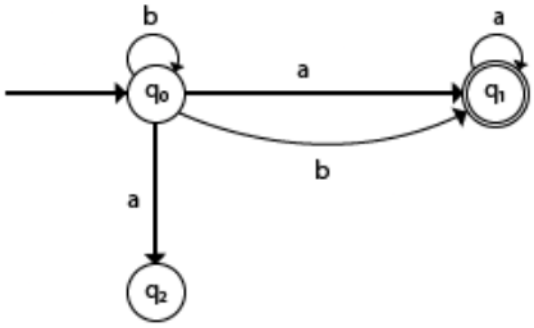


Fig:- NDFA

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

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

where,

Q: finite set of states

∑: finite set of the input symbol

q0: initial state

F: final state

δ: Transition function

Examples of NFA

1. Design an NFA with ∑ = {a, b} accepts all string starting with ab.
2. Design an NFA with ∑ = {a, b} accepts all string starting with bab.
3. Design an NFA with ∑ = {a, b} accepts all string ending with ba.
4. Design an NFA with ∑ = {a, b} accepts all string ending with bab.
5. Design an NFA with ∑ = {a, b} accepts all string must contain a substring aa.
6. Design an NFA with ∑ = {0, 1} accepts all string ending with 01.

Conversion from NFA to DFA

An NFA can have zero, one or more than one move from a given state on a given input symbol. An NFA can also have NULL moves (moves without input symbol). On the other hand, DFA has one and only one move from a given state on a given input symbol.

Conversion from NFA to DFA

Suppose there is an NFA N <Q, ∑, q0, δ, F> which recognizes a language L. Then the DFA D <Q', ∑, q0, δ', F'> can be constructed for language L as:

Step 1: Initially Q' = ϕ.

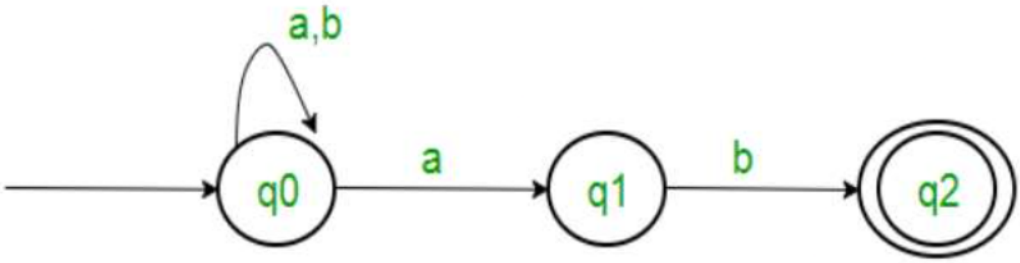
Step 2: Add q0 to Q'.

Step 3: For each state in Q', find the possible set of states for each input symbol using transition function of NFA. If this set of states is not in Q', add it to Q'.

Step 4: Final state of DFA will be all states with contain F (final states of NFA)

Example

Consider the following NFA



Following are the various parameters for NFA.

Q = { q0, q1, q2 }

∑ = (a, b)

F = { q2 }

δ (Transition Function of NFA)

State	a	b
q0	q0,q1	q0
q1		q2
q2		

Step 1: Q' = ϕ

Step 2: Q' = {q0}

Step 3: For each state in Q', find the states for each input symbol.

Currently, state in Q' is q0, find moves from q0 on input symbol a and b using transition function of NFA and update the transition table of DFA.

δ' (Transition Function of DFA)

State	a	b
q0	{q0,q1}	q0

Now { q0, q1 } will be considered as a single state. As its entry is not in Q', add it to Q'.

So Q' = { q0, { q0, q1 } }

Now, moves from state { q0, q1 } on different input symbols are not present in transition table of DFA, we will calculate it like:

δ' ({ q0, q1 }, a) = δ (q0, a) ∪ δ (q1, a) = { q0, q1 }

δ' ({ q0, q1 }, b) = δ (q0, b) ∪ δ (q1, b) = { q0, q2 }

Now we will update the transition table of DFA.

δ' (Transition Function of DFA)

State	a	B
q0	{q0,q1}	q0
{q0,q1}	{q0,q1}	{q0,q2}

Now { q0, q2 } will be considered as a single state. As its entry is not in Q', add it to Q'.

So Q' = { q0, { q0, q1 }, { q0, q2 } }

Now, moves from state {q0, q2} on different input symbols are not present in transition table of DFA, we will calculate it like:

δ' ({ q0, q2 }, a) = δ (q0, a) ∪ δ (q2, a) = { q0, q1 }

δ' ({ q0, q2 }, b) = δ (q0, b) ∪ δ (q2, b) = { q0 }

Now we will update the transition table of DFA.

δ' (Transition Function of DFA)

State	a	B
q0	{q0,q1}	q0
{q0,q1}	{q0,q1}	{q0,q2}
{q0,q2}	{q0,q1}	q0

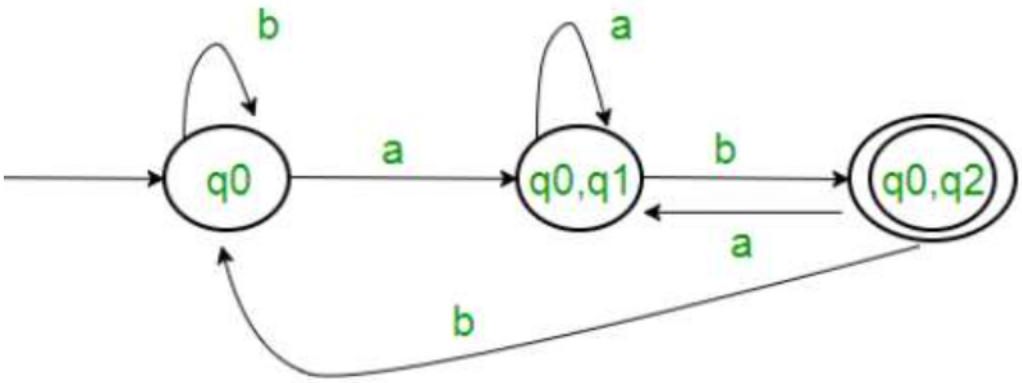
As there is no new state generated, we are done with the conversion. Final state of DFA will be state which has q2 as its component i.e., {q0, q2}

Following are the various parameters for DFA.

Q' = { q0, { q0, q1 }, { q0, q2 } }

∑ = (a, b)

F = { { q0, q2 } } and transition function δ' as shown above. The final DFA for above NFA has been shown in following figure



Question

Convert the following Non-Deterministic Finite Automata (NFA) to Deterministic Finite Automata (DFA).