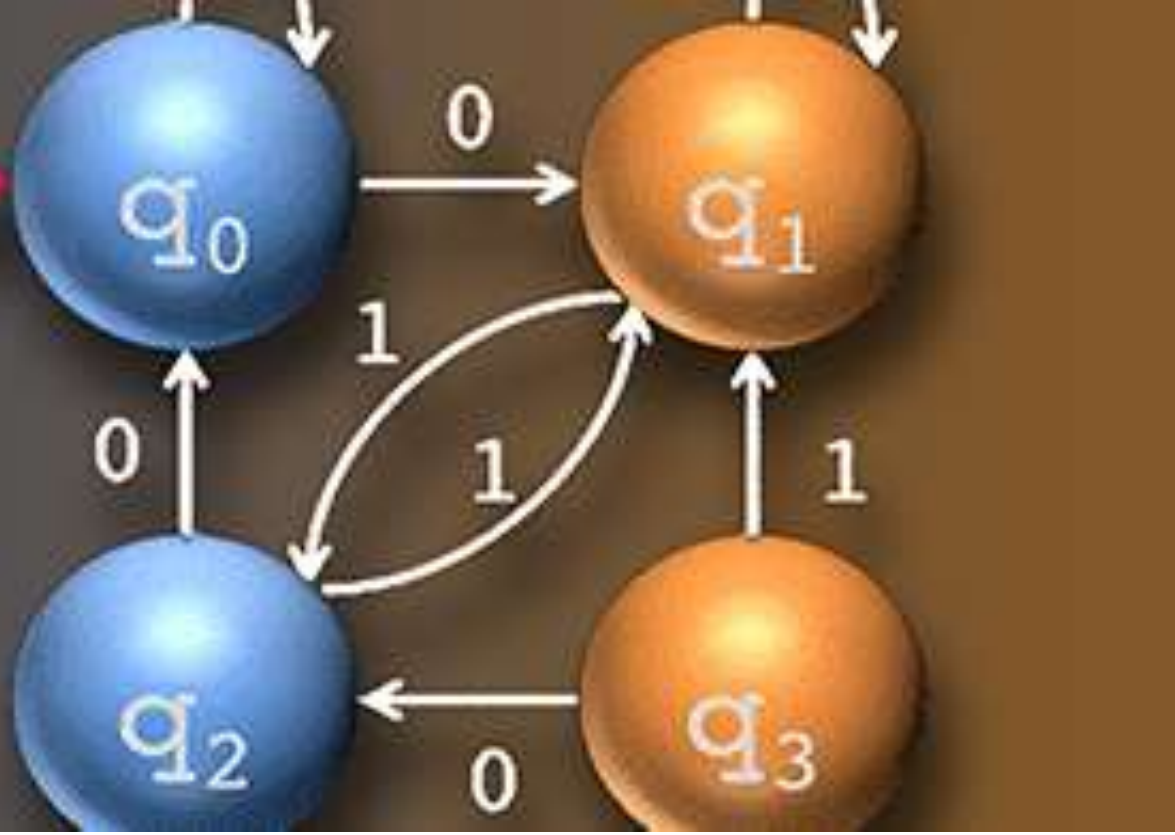
CSE 305

Theory of COMPUTATION



Lecture 14

Finite-State Automata (1)

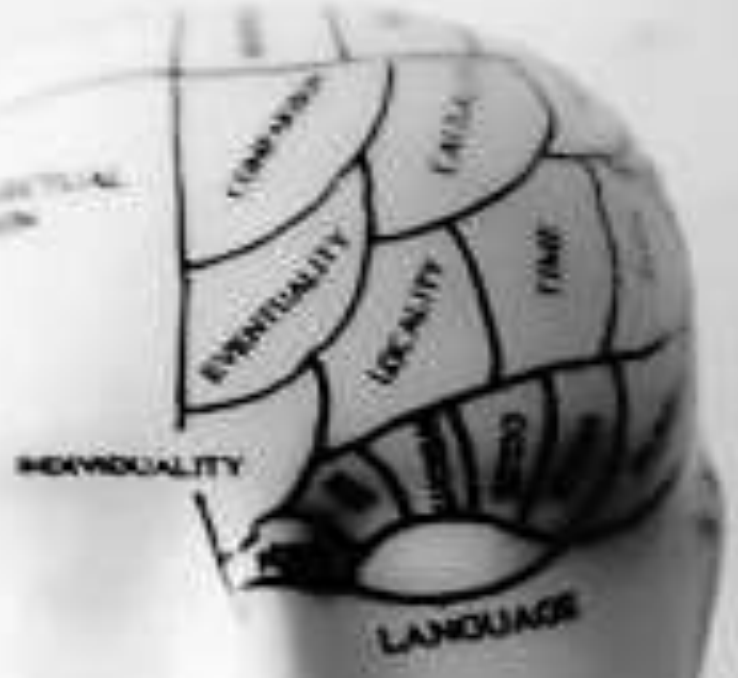
Md. Mijanur Rahman, Prof. Dr.

Dept. of Computer Science and Engineering, Jatiya Kabi Kazi Nazrul Islam University, Bangladesh.

www.mijanrahman.com

Contents

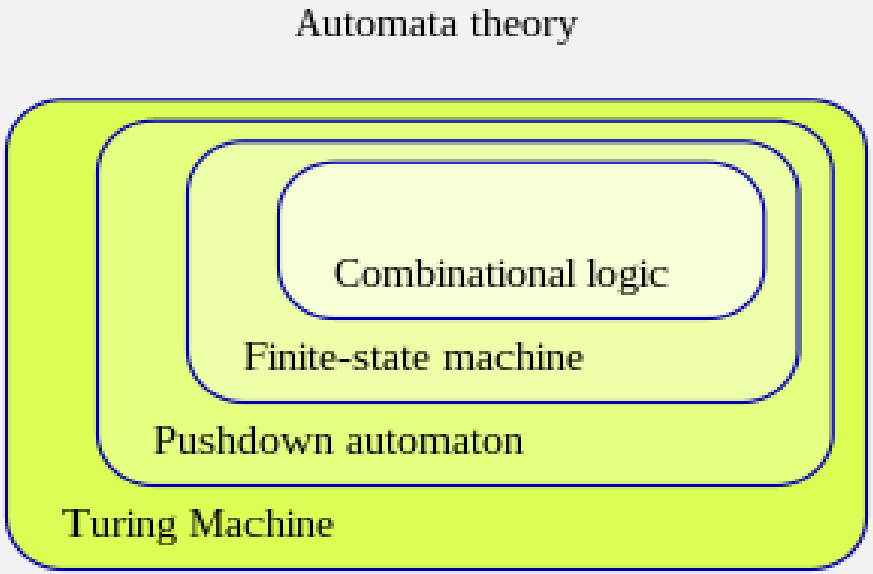
Finite State Automata



- Finite State Machine
- Types of Finite Automata
- Transition Function, Diagram and Table
- DFA, NFA, e-NFA with Definitions and Examples
- Extended Transition Function
- The Equivalence of DFA's and NFA's
- The Equivalence of NFA's with and without e-moves
- Conversion of e-NFA into NFA (without e)
- Two-way FA
- FA with Output: Moore machine, Mealy machine, Equivalence
- Applications of FA

Finite-State Machine

- A **finite-state machine** (**FSM**) or **finite-state automaton** (**FSA**, plural: *automata*), **finite automaton**, or simply a **state machine**, is a mathematical model of computation.
- **It is an abstract machine that can be in exactly one of a finite number of *states* at any given time. The FSM can change from one state to another in response to some inputs; the change from one state to another is called a *transition*.**
- An FSM is defined by **a list of its states, its initial state, and the inputs** that trigger each transition.



FSM: Examples

- The behavior of state machines can be observed in many devices in modern society that perform a predetermined sequence of actions depending on a sequence of events with which they are presented.
- Simple examples are **vending machines**, which dispense products when the proper combination of coins is deposited, **elevators**, whose sequence of stops is determined by the floors requested by riders, **traffic lights**, which change sequence when cars are waiting, and combination locks, which require the input of a sequence of numbers in the proper order.
- The finite-state machine has less computational power than some other models of computation such as the Turing machine.
- The computational power distinction means there are computational tasks that a Turing machine can do but an FSM cannot. This is because an FSM's memory is limited by the number of states it has.

FSM: Features

- **Major features of 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.

FSM: Elements or Tuples

- **Finite Automata(FA) is the simplest machine to recognize patterns. 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. Basically, it is an abstract model of a digital computer.
- The following figure shows some essential features of general automation.
    1. Input
    2. Output
    3. States of automata
    4. State relation
    5. Output relation

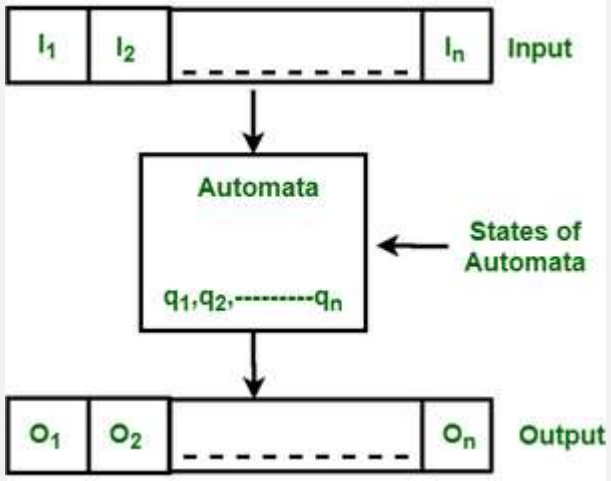


Figure: *Features of Finite Automata*

FSM: Definition of FA

- **Formal Definition of FA:**
- A finite automaton is a collection of 5-tuple (Q, ∑, δ, q0, F), where:
  Q : Finite set of states.
  Σ : set of Input Symbols.
  q : Initial state.
  F : set of Final States.
  δ : Transition Function.

- Thus, Formal specification of machine is { Q, Σ, q, F, δ }

FSM: FA Model

- **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.

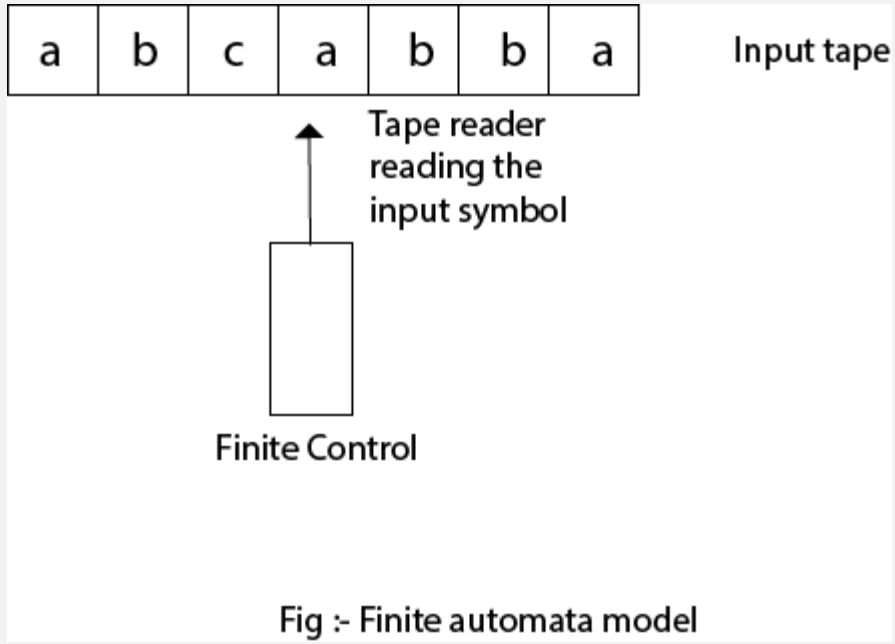


Fig :- Finite automata model

FSM: Types of FA

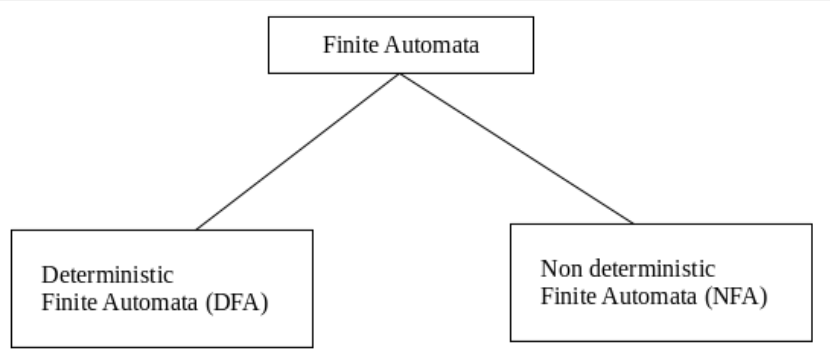
- **Finite-state automata are of two types: deterministic finite automata (DFA) and non-deterministic finite automata (NFA).** A deterministic finite automata can be constructed equivalent to any non-deterministic one.

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 accept the 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.



FSM: Types of FA

- **Some important points about DFA and NFA:**
  - Every DFA is NFA, but NFA is not DFA.
  - There can be multiple final states in both NFA and DFA.
  - DFA is used in Lexical Analysis in Compiler.
  - NFA is more of a theoretical concept.

FA: Transition Diagram

- A transition diagram or state transition diagram is a directed graph which can be constructed as follows:
  - 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 $\delta(q, a) = p$.
  - 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:

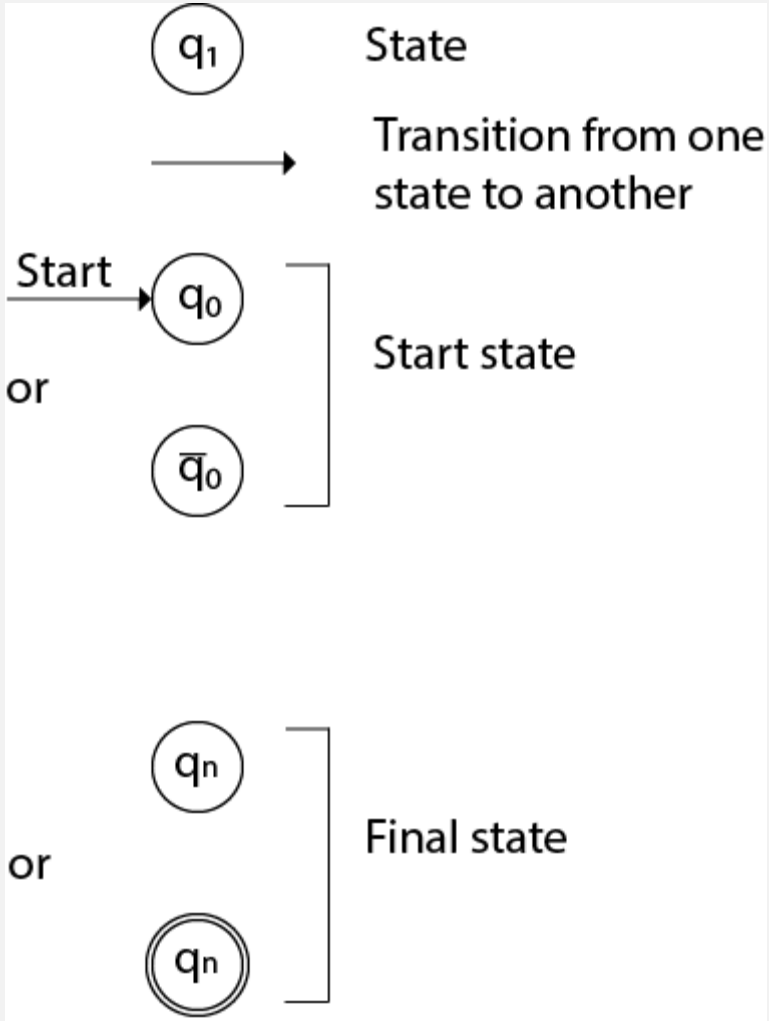


Fig:- Notations

FA: Transition Diagram

- **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 $\delta(p, a)$. 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 $r \in F$ 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 $r \notin F$.

FA: Transition Diagram

- **Example 1:**

DFA with $\sum = \{0, 1\}$ accepts all strings starting with 1.

Solution:

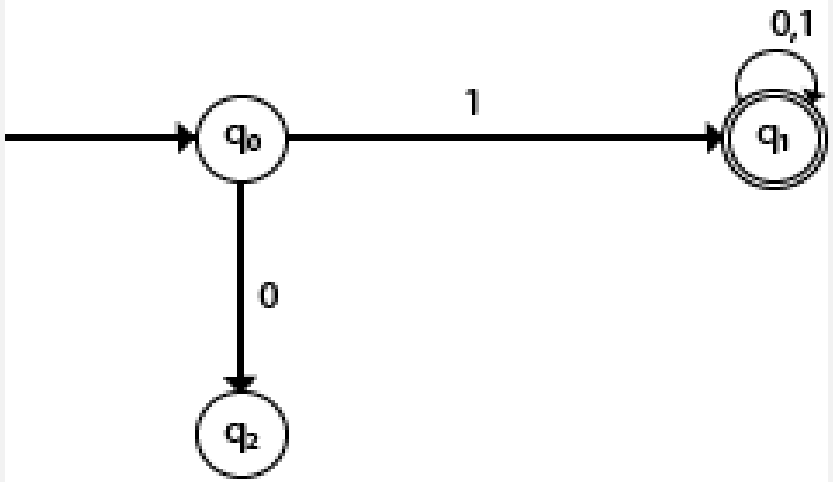


Fig: Transition diagram

- The finite automata can be represented using a transition graph. In the above diagram, the machine initially is in start state q_0 then on receiving input 1 the machine changes its state to q_1.
- From q_0 on receiving 0, the machine changes its state to q_2, which is the dead state.
- From q_1 on receiving input 0, 1 the machine changes its state to q_1, which is the final state.
- The possible input strings that can be generated are 10, 11, 110, 101, 111....., that means all string starts with 1.

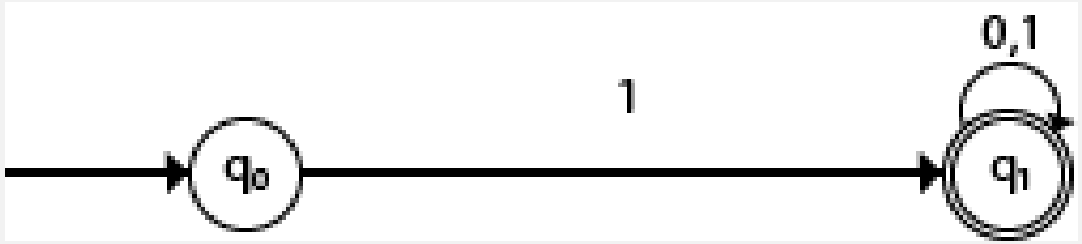
FA: Transition Diagram

- **Example 2:**

  NFA with $\sum = \{0, 1\}$ accepts all strings starting with 1.

Solution:

Transition Diagram:



- The NFA can be represented using a transition graph. In the above diagram, the machine initially is in start state q0 then on receiving input 1 the 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.

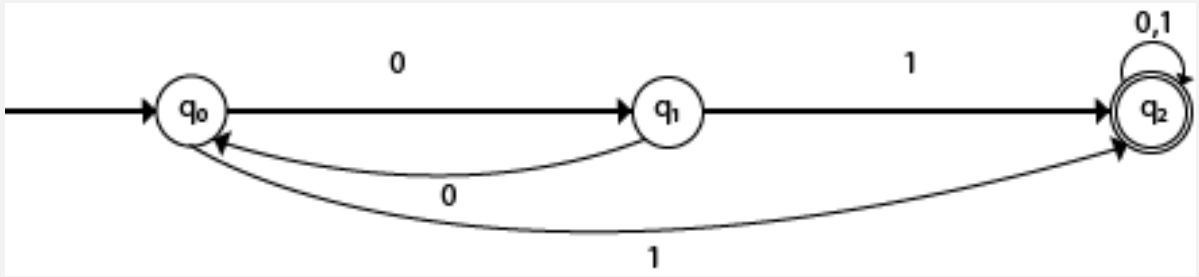
FA: Transition Table

- 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 by a star.

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

FA: Transition Table

- **Example 3: Consider the following DFA and find the transition table.**

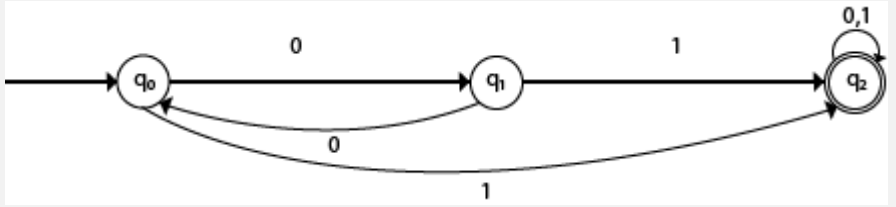


Solution:

- 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

FA: Transition Table



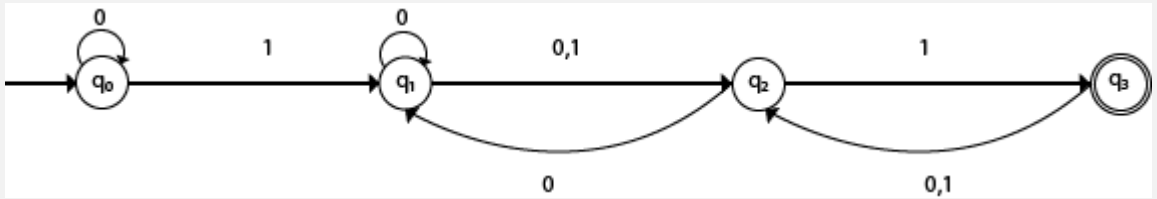
- **Example 3:**

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.

FA: Transition Table

- **Example 4: Consider the following NFA and find the transition table.**

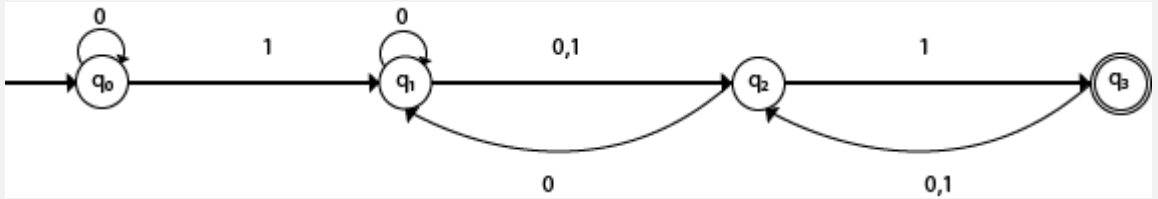


Solution:

- 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

FA: Transition Table



- **Example 4:**

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

- **Explanation:**
  - 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 q0 and on input 1 the next state will be q1.
  - In the second row, when the current state is q1, on input 0 the next state will be either q1 or q2, 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 q1, and on 1 input the next state will be q3.
  - In the fourth row, when the current state is q3 on input 0, the next state will be q2, and on 1 input the next state will be q2.

?

THE END