LECTURE 04

Finite State Machine

Course Code : CSE 3103
Course Title : Theory of Computation

Md. Sozib Hossain
Lecturer, CSE, RUET
sozib.hossain@cse.ruet.ac.bd

Finite State Machine (FAM)

- 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 has **finite number of states.**
- It accepts **Regular language.**
- Finite State Machine is classified into two types:
  1. Deterministic Finite State Machine or Deterministic Finite Automata (DFA)
  2. Non-deterministic Finite State Machine or Non-deterministic Finite Automata (NFA)
- Finite State Machine Consist of five tuples:

  $(Q, \Sigma, \delta, Q_{0,}\ Q_f)$

  Q = set_of_all_states, Σ = set_of_all_input_symbol_or_alphabet,

  δ = transition_function, Q_0 = initial_state, Q_f = set_of_final_states

DFA vs NFA

DFA	**NFA**
For each symbolic representation of the alphabet, there is only one state transition in DFA.	No need to specify how does the NFA react according to some symbol.
DFA cannot use Empty String transition.	NFA can use Empty String transition.
DFA can be understood as one machine.	NFA can be understood as multiple little machines computing at the same time.
In DFA, the next possible state is distinctly set.	In NFA, each pair of state and input symbol can have many possible next states.
DFA is more difficult to construct.	NFA is easier to construct.
DFA rejects the string in case it terminates in a state that is different from the accepting state.	NFA rejects the string in the event of all branches dying or refusing the string.

DFA vs NFA (Cont…)

DFA	**NFA**
Time needed for executing an input string is less.	Time needed for executing an input string is more.
All DFA are NFA.	Not all NFA are DFA.
DFA requires more space.	NFA requires less space then DFA.
Dead configuration is not allowed.	Dead configuration is allowed.
Backtracking is allowed in DFA.	Backtracking is not always possible in NFA.
Conversion of Regular expression to DFA is difficult.	Conversion of Regular expression to NFA is simpler compared to DFA.
Epsilon move is not allowed in DFA	Epsilon move is allowed in NFA
DFA allows only one move for single input alphabet.	There can be choice (more than one move) for single input alphabet.

Transition Function

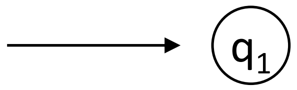
The transition function defines the movement of an automaton from one state to another by treating the current state and current input symbol as an ordered pair. For each pair of "current state" and "current input symbol" (the function input), the transition function produces as output the next state in the automaton.

There are two ways to represent transition function:

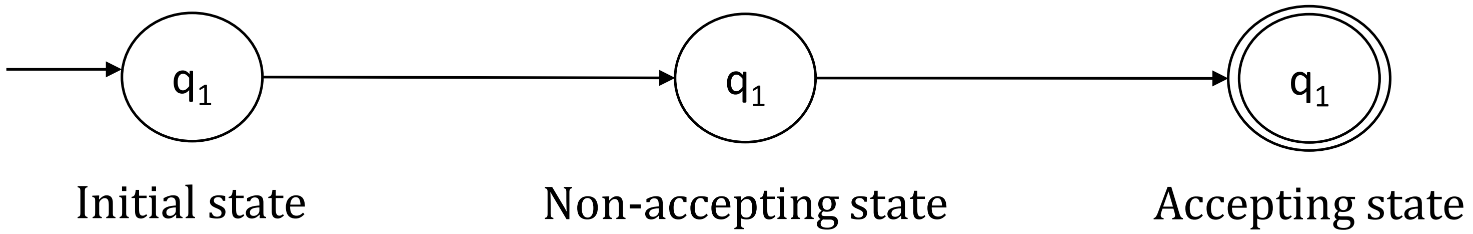
1. Transition diagram: use graph (Nodes & edges)
2. Transition Table: use table (Rows & columns)

States

- Each state represent the status of the FSM.
- **In transition diagram**, each vertices/node of a graph represent a state that is denoted by Circle. Each state has a name inside the circle.

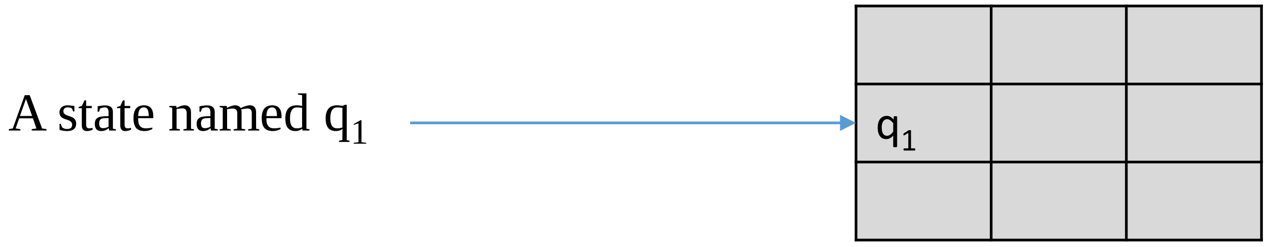
A state named q_1 

- There are there types of state:
  1. Initial State: a null arrow point it.
  2. Accepting State: double circle
  3. Non-accepting State: if not accepting state.

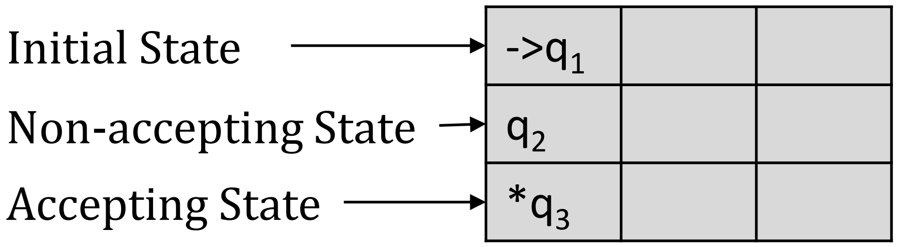


States(Cont…)

- Each state represent the status of the FSM.
- **In transition table**, each row of the table represent one state. Each state has a name written in the first cell of the corresponding state.

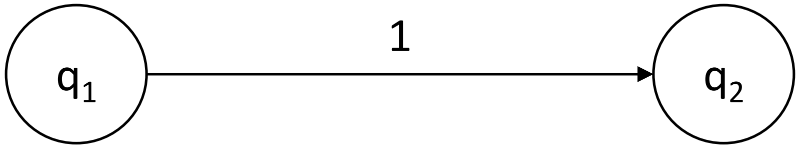


- Since there are there types of state:
  1. Initial State: a null arrow before the state name.
  2. Accepting State: ateric before the state name.
  3. Non-accepting State: if not accepting state.



Transition

- **In transition diagram**, each arc/edge of the graph represent movement of FSM from one state to another based on the current input symbol. The input symbol is labeled with the arc/edge.
- For example, A machine changes it's state from q_1 to q_2 when it encounters a input '1' at q_1.


$$\begin{tikzcd} q_1 \arrow[r, "1"] & q_2 \end{tikzcd}$$

Transition(cont…)

- **In transition table**, each column of the table represent a input symbol labeled at the top cell and all cell below it represent the next state when a state encounters the corresponding input.
- For example, A machine change it's state from q_1 to q_2 when it encounters a input '1'.

	1	
q_1	q_2	