Theory of Computation

BY

MR. VIPIN WANI

UNIT 1

FORMAL LANGUAGE THEORY AND FINITE AUTOMATA

Computation

- Computation is a general term for any type of information processing that can be represented as an algorithm precisely (mathematically).

Computation

- Computation is a general term for any type of information processing that can be represented as an algorithm precisely (mathematically).

Examples:

- Adding two numbers in our brains, on a piece of paper or using a calculator.

Computation

- Computation is a general term for any type of information processing that can be represented as an algorithm precisely. (mathematically)

Examples:

- Adding two numbers in our brains, on a piece of paper or using a calculator.
- Converting a decimal number to its binary presentation or vise versa.

Computation

- Computation is a general term for any type of information processing that can be represented as an algorithm precisely (mathematically).

Examples:

- Adding two numbers in our brains, on a piece of paper or using a calculator.
- Converting a decimal number to its binary presentation or vise versa.
- Finding the greatest common divisors of two numbers.

Why Theory in Computation ?

- Theory gives us
- Concepts
- Principles
- Steps
- Limitations
- Advantages
- etc

Theory of Computation

- A very fundamental and traditional branch of Theory of Computation seeks:
  - A more tangible definition for the intuitive notion of *algorithm* which results in a more concrete definition for computation.

Theory of Computation

- A very fundamental and traditional branch of Theory of Computation seeks:
  - A more tangible definition for the intuitive notion of *algorithm* which results in a more concrete definition for computation.
  - Finding the boundaries (limitations) of computation.

Algorithm

- A finite sequence of simple instructions that is guaranteed to halt in a finite amount of time.

Basic Concepts

- The Basic Concepts of TOC are
  1. Languages
  2. Grammar
  3. Automata (Automaton)

Languages

- A **language** is a structured system of communication used by humans, based on speech and gesture (spoken **language**), sign, or often writing.
- Language consist of

1. Symbols
2. Alphabets
3. Strings

Strings and Languages

- **Symbols:** are Building blocks of language.

  Ex. {a, b, c, d, e,…… 1, 2, 3, 4,……., A, B, C, D…….}

- **Alphabet**: Finite, nonempty set of symbols
  - Examples:
    - $\sum$ = {0, 1}: binary alphabet
    - $\sum$ = {*a*, *b*, *c*, …, *z*}: the set of all lower case letters
    - The set of all ASCII characters

- **String**: Finite sequence of symbols from an alphabet $\sum$ is represented by w
  - Examples:
    - 01101 where $\sum$ = {0, 1} =0,1,01,10, 11,
    - *abracadabra* where $\sum$ = {*a*, *b*, *c*, …, *z*}

Strings and Languages

- **Empty String**: The string with **zero** occurrences of symbols from ∑ and is denoted **e** or λ **or** ε

- **Length of String**: Number of symbols in the string
  - The length of a string *w* is usually written |*w*| w=1010
  - |1010| = 4
  - |**e**| = 0
  - $|uv| = |u| + |v|$

- **Reverse** : w^R
  - If w = abc, w^R = cba

Strings and Languages

- **Concatenation**: if ***x*** and ***y*** are strings, then ***xy*** is the string obtained by placing a copy of ***y*** immediately after a copy of ***x***
  - $x = a_1a_2 \ldots a_i$, $y = b_1b_2 \ldots b_j$
  - $xy = a_1a_2 \ldots a_ib_1b_2 \ldots b_j$
  - Example: $x = 01101$, $y = 110$, $xy = 01101110$
  - $xe = ex = x$

Strings and Languages

- **Substring**: any string of consecutive characters in some string *w*
  - If *w* = abc
  - e, a, b, c, bc, ab, abc are substrings of *w*
- **Prefix** and **suffix**:
  - if w = vu
  - v is a prefix of w
  - u is a suffix of w
  - Example 1:
    - If w = abc
    - a, ab , abc are prefixes of w
    - c, bc, abc are suffixes of w
    - Example 2:
- Unhappy in this “un” is prefix and in quickly “ly” is suffix

Strings and Languages

- Suppose: S is the string banana
  - **Prefix : ban, banana**
  - **Suffix : ana, banana**
  - **Substring : nan, ban, ana, banana**

* Closure/ Kleene Closure/ Power of an (∑) Alphabet

- **Power of an (∑) Alphabet**: Σ^k = the set of strings of length k with symbols from Σ
  - Example: $\Sigma = \{0, 1\}$
- Σ^0 =set of all string of length zero $\{\varepsilon\}=2^0$ =1
- Σ^1 = set of all string of length 1 = $\Sigma = \{0, 1\} = 2^1$
- Σ^2 = set of all string of length 2
  - $= \Sigma.\ \Sigma = \{0, 1\}.\ \{0, 1\} = \{00, 01, 10, 11\} = 2^2$
- Σ^3 = set of all string of length 3 = $\Sigma.\ \Sigma\ .\ \Sigma = 2^3$
  - $= \Sigma^2.\ \Sigma = \{00, 01, 10, 11\}\ .\ \{0, 1\} = \{000, 010, 100, 110, 001, 011, 101, 111\}$

Kleene Closure/ Power of an (∑) Alphabet

- The set of all possible strings over Σ is called Kleene Closure denoted by Σ^*
  - $\Sigma^* = \Sigma^{n=} = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \ldots$

- **Positive Closure**

  - $\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \ldots$
  - $\Sigma^* = \Sigma^+ \cup \{\varepsilon\}$ or $\Sigma^+ + \{\varepsilon\}$
  - $\Sigma^+ = \Sigma^* - \{\varepsilon\}$

Strings and Languages

- **Language**: set of strings chosen from some alphabet
- A language is a subset of Σ^*
  - Example of languages:
    - The set of valid Arabic words
    - If $\sum$=(a, b) then L1 is Language of all strings of length two.
      - L1={aa, bb, ab, ba}
    - If $\sum$=(a, b) then L2 is Language of all strings of length three.
      - L2={aaa, bbb, aba, aab, abb, bab, bba, baa }
    - The set of strings consisting of *n* **0**'s followed by *n* **1**'s if $\sum$=(0, 1)
      - {e, 01, 0011, 000111, ...}
    - The set of strings with equal number of **0**'s and **1**'s if $\sum$=(0, 1)
      - {e, 01, 10, 0011, 0101, 1010, 1001, 1100, ...}

Strings and Languages

- **Empty language**: $\varnothing = \{\ \}$
  - The language { } consisting of the empty string
  - Note: $\varnothing \neq \{e\}$
- **Languages can be finite or infinite**
  - L1 = {*a*, *aba*, *bba*}
  - L2 = $\{a^n \mid n > 0\}$={a, aa, aaa, aaaa,…….}
  - If ∑=(a, b) then L3 is Language of all strings starts with a.

    L3={a, aa, aaa, aaaa,…….ab, abb, aab,……….}

Strings and Languages

- Can concatenate languages
  - $L_1L_2 = \{xy \mid x \in L_1, y \in L_2\}$
  - L^n = L concatenated with itself *n* times
  - $L^0 = \{e\}$
  - $L^1 = L$
- Star-closure
  - $L^* = L^0 \cup L^1 \cup L^2 \cup \ \ldots$
  - $L^+ = L^* - L^0$

Strings and Languages

OPERATION	DEFINITION
union of L and M written $L \cup M$	$L \cup M = \{s \mid s \text{ is in } L \text{ or } s \text{ is in } M\}$
concatenation of L and M written LM	$LM = \{st \mid s \text{ is in } L \text{ and } t \text{ is in } M\}$
Kleene closure of L written as L*	$L^* = L^0 \cup L^1 \cup L^2 \cup \ldots$ L* denotes "zero or more concatenations of " L
positive closure of L written as L^+	$L^+ = L^1 \cup L^2 \cup \ldots$ L^+ denotes "one or more concatenations of " L $L^+ = LL^*$

Strings and Languages (Example)

- The language **L** consists of strings over **{a,b}** in which **each** string begins with an **a** should have an **even** length
  - aa, ab $\in$ L
  - aaaa, aaab, aaba ,aabb ,abaa, abab, abba, abbb $\in$ L
  - baa $\notin$ L
  - a $\notin$ L

Strings and Languages (Example)

- The language **L** consists of strings over **{a,b}** in which each occurring of **b** is **immediately preceded** by **a**

  - e $\in$ L
  - ab $\in$ L
  - abaabab $\in$ L

  - bb $\notin$ L
  - bab $\notin$ L
  - abb $\notin$ L

Strings and Languages (Example)

- Let X = {a,b,c} and Y = {abb, ba}. Then
  - XY = {aabb, babb, cabb, aba, bba, cba}
  - X^0 = {e}
  - X^1 = X = {a,b,c}
  - X^2 = XX = {aa,ab,ac,ba,bb,bc,ca,cb,cc}
  - X^3 = XXX =

    {aaa,aab,aac,aba,abb,abc,aca,acb,acc,baa,bab,bac,bba,bbb,

    bbc,bca,bcb,bcc,caa,cab,cac,cba,cbb,cbc,cca,ccb,ccc}

Strings and Languages (Example)

- The language L = {a,b} consists of the strings over {a,b} that contain the substring **bb**
  - bb ∈ L
  - abb ∈ L
  - bbb ∈ L
  - aabb ∈ L
  - bbaaa ∈ L
  - bbabba ∈ L
  - abab ∉ L
  - bab ∉ L
  - b ∉ L

Strings and Languages (Example)

- Let **L** be the language that consists of all strings that begin with **aa** or end with **bb**
  - $L_1 = \{aa\}\{a,b\}^*$
  - $L_2 = \{a,b\}^*\{bb\}$
  - $L = L_1 \cup L_2 = \{aa\}\{a,b\}^* \cup \{a,b\}^*\{bb\}$
  - $bb \in L$
  - $abb \in L$
  - $bbb \in L$
  - $aabb \in L$
  - $bbaaa \notin L$
  - $bbabba \notin L$
  - $abab \notin L$
  - $bab \notin L$
  - $ba \notin L$

Finite Representations of Languages

- **Languages may be infinite sets of strings. We need a finite notation for them. There are at least four ways to do this:**

1. Language generators. The language can be represented as a mathematical sequence w1, w2, w3,……. such that the language is equal to the {w1, w2, w3………}. Given an integer i, the generator will produce the string wi.
2. Language acceptors. The language can be represented as a mathematical predicate, a membership tester. Given a string, this will tell if the string is in the language.
3. Mathematical descriptions, like { $a^n b^n$: n>=0 }
4. Explicit listings, like {0, 1, 01, 10}.

Finite Representations of Languages

- Explicit listings work only for finite languages.
- Math descriptions are very general, but it may be hard to know if a string is in the language.
- Language acceptors have a hard time answering some questions, such as whether the language is empty.
- Language generators have a hard time testing if a string is in the language.
- There are uncountably many languages over a nonempty set ∑ but only countable many representations in a finite set of symbols. Therefore most languages will never have a finite representation.

Regular Expressions for Representations of Languages

Regular Expressions: Regular expressions are one way to represent languages. They are analogous to arithmetic expressions for representing quantities. This notation will turn out to be useful for describing programming languages and also for text searching applications.

- There are rules of inference for constructing regular expressions over an alphabet .

  1. If a Є ∑ then a itself is a regular expression over ∑.
  2. φ is a regular expression over ∑ .
  3. If E and F are regular expressions over ∑ then so is (EF).
  4. If E and F are regular expressions over ∑ then so is (E U F).
  5. If E is a regular expression over ∑ then so is (E).
  6. Parentheses can often be omitted.

- .

Regular Expressions for Representations of Languages

- Example: Suppose Σ = {0, 1}.

- Then 0 is a regular expression over {0, 1} by 1.
- So (0) is a regular expression over {0, 1} by 5.
- Also, 1 is a regular expression over {0, 1} by 1.
- So 1.(0) is a regular expression over {0, 1} by 3.
- Also (1) is a regular expression over {0, 1} by 5.
- So 0.(1) is a regular expression over {0, 1} by 3.
- Thus 1(0)U0(1) is a regular expression over {0, 1} by 4.

Regular Expressions for Representations of Languages

- Language Represented by a Regular Expression
- If E is a regular expression then let L(E) be the language it represents.
- We have the following rules:

- If a Є ∑ then L(a) = {a}
- L(φ) = φ;
- L(EF) = L(E) . L(F)
- L(E U F) = L(E) U L(F)
- L(E*) = L(E)*

Grammar for Representations of Languages

- A grammar for a natural language tells us whether a particular sentence is well-formed or not.
- It also tells us weather the sentence /string is a part of language or not.
- Formal grammar:

$$G = (V, T, S, P)$$

V: finite set of variables

T: finite set of terminal symbols

$S \in V$: start variable

P: finite set of productions

Grammar for Representations of Languages

- G ({S, A}, {a,b}, S, P)
  where P

  S -> Ab,

  A -> aAb,

  A -> L.

Automata

- An automaton also known as finite automata (FA) or Finite state machine (FSM) is an abstract model of a digital computer.
- A **finite automaton** (FA) is a simple idealized machine used to recognize patterns within input taken from some character set (or alphabet) C.
- The job of an FA is to accept or reject an input depending on whether the pattern defined by the FA occurs in the input.

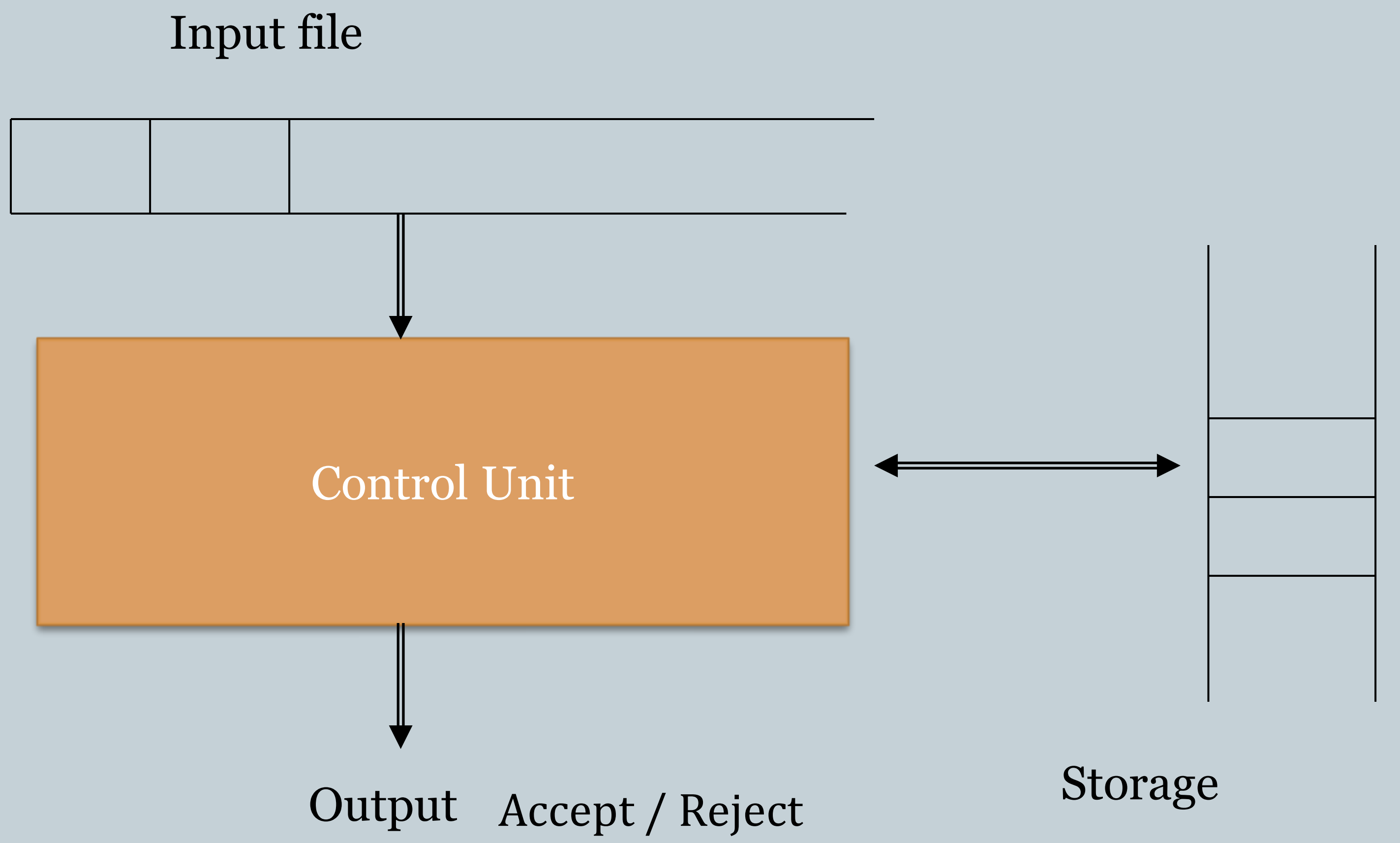
  Includes:
  - Input file:
    - Read as a string
    - One symbol at a time, left to right, EOF.
  - Can produce output
  - Has temporary storage
    - Unlimited, one character, may change
  - Control unit
    - Finite number of internal states, may change

Automata

- A **finite automaton** (FA) is used for solving some common types of algorithms such as.

1. Design of digital circuit
2. String Matching
3. Communication protocols for information exchange
4. Lexical analyzer of a compiler

Automata



Automata

- Input file: is divided into squares.
- Input is a string over a given alphabet.
- Each input square holds a symbol.
- The symbols are read from left to right, one at a time.
- The end of the input string can be detected.
- Storage: consists of an unlimited number of cells.
- Each cell can hold a symbol from an alphabet (which can be different from the input alphabet).
- The contents of the storage cells can be read and changed.

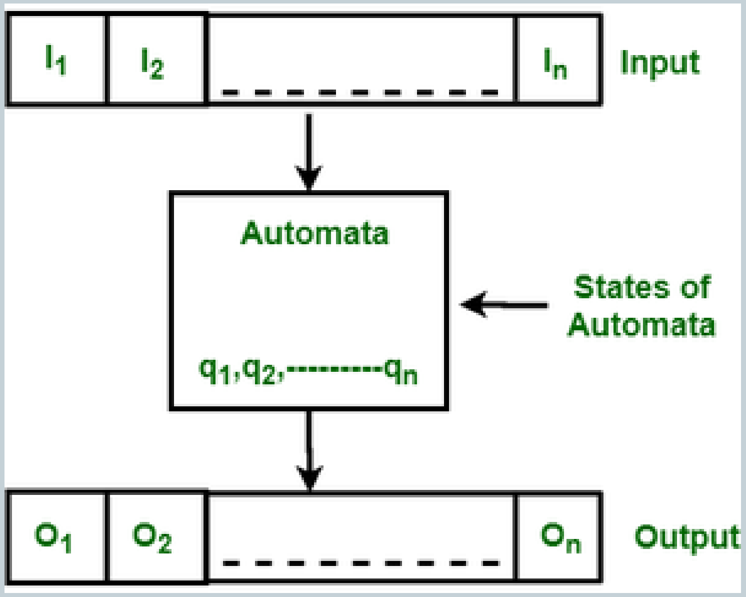
Automata

- Control unit: has a finite number of internal states.
- Can be in any one of the internal states.
- Can change state in some defined manner.

Automata

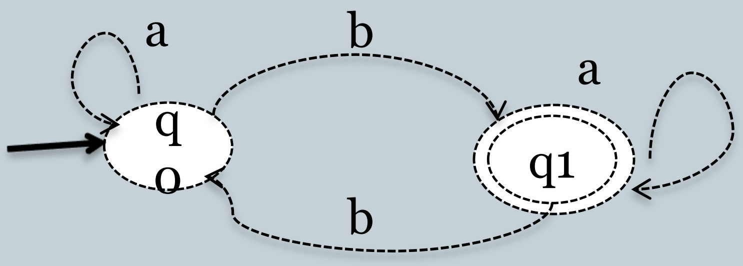
Finite Automata(FA) is the simplest machine to recognize patterns. The finite automata or finite state machine is an abstract machine which have five elements or tuple. It has a set of states and rules for moving from one state to another but it depends upon the applied input symbol. Basically it is an abstract model of digital computer. Following figure shows some essential features of a general automation.

Formal specification of machine is
{ Q, Σ, q, F, δ }.
Q : Finite set of states.
Σ : set of Input Symbols.
q : Initial state.
F : set of Final States.
δ : Transition Function.



Automata

To understand consider the example. Consider the two states q0, q1. q0 is a Initial state while q1 is final state and if the input b is applied then its state is changes from q0 to q1 or q1 to q0. for input a it remains in the same state.

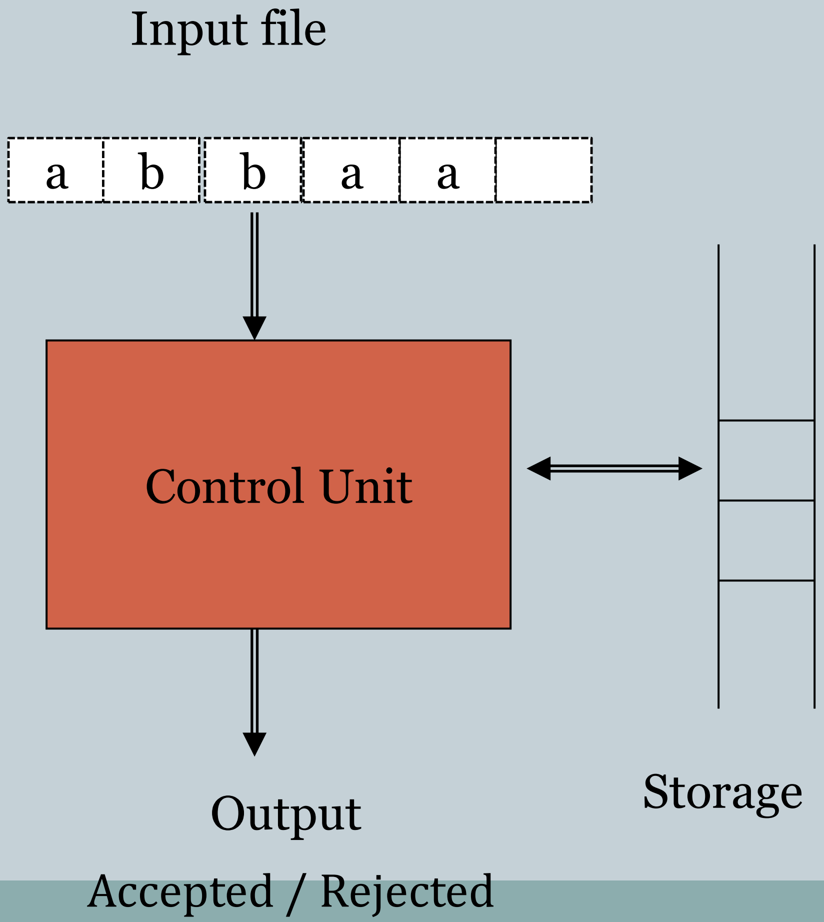


Q0b→Q1

Q0a→Q0

Q1b→Q0

Q1a→q1



Basic Machine Vs Finite Automata

- Machine is a system in which energy, material or information can be transformed without any human interaction.
- Ex. Photo printing machine, metal cutting machine etc.
- But it is very difficult to incorporate decision making power in machine using mechanical control system.
- FA is mathematical model for actual physical process and it can be viewed as a several restricted model of computer.
- A **finite automaton** (FA) is used for solving some common types of algorithms such as.

1. Design of digital circuit
2. String Matching
3. Communication protocols for information exchange
4. Lexical analyzer of a compiler

Limitations of Finite Automata

- It cannot be used for computations.
- It cannot modify its input.
- It cannot be used for context free language.
- It cannot be used for recursive languages
- It has a finite number of states and hence It cannot recognize string of the form a^nb^n

Types of Finite Automata

- Deterministic Finite Automata
- Non Deterministic Finite Automata

Deterministic Finite Automata (DFA)

- In DFA, for each input symbol, one can determine the single state to which the machine will move. Hence, it is called **Deterministic Automaton**.
- As it has a finite number of states, the machine is called **Deterministic Finite Machine** or **Deterministic Finite Automaton.**

- A DFA can be represented by a 5-tuple (Q, ∑, δ, q_0, F) where –

1. **Q** is a finite set of states.
2. **∑** is a finite set of symbols called the alphabet.
3. **δ** is the transition function where $\delta: Q \times \Sigma \rightarrow Q$
4. **q_0** is the initial state from where any input is processed ($q_0 \in Q$).
5. **F** is a set of final state/states of Q ($F \subseteq Q$).

Deterministic Finite Automata (DFA)

- A DFA is represented by digraphs called **state diagram.**

1. The vertices represent the states.
2. The arcs labeled with an input alphabet show the transitions.
3. The initial state is denoted by an empty single incoming arc.
4. The final state is indicated by double circles

- Example: Construct FA (state Diagram) for given M
- Let a deterministic finite automaton be →
- $M=(Q, \sum, \delta, q_0, F)$ where

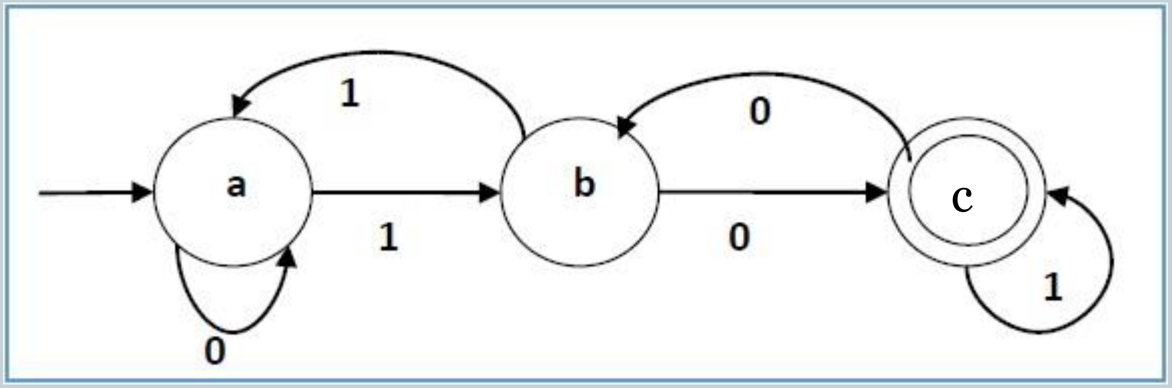
- $Q = \{a, b, c\}$,
- $\sum = \{0, 1\}$,
- $q_0 = \{a\}$,
- $F = \{c\}$, and

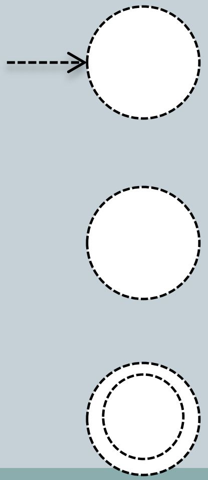
Deterministic Finite Automata (DFA)

- Transition function δ or state transition table can be shown as following table –

Present State	Next State for Input 0	Next State for Input 1
a	a	b
b	c	a
c	b	c

State Transaction diagram :





Example on Counting of Symbols (DFA)

Ex 1. Give DFA accepting the following language over the alphabet {0,1}

- 1. Number of 1's is multiple of 3.
- 2. Number of 1's not multiple of 3.

Solution:

1. Number of 1's is multiple of 3: number of 1's seen so far by machine can be written as

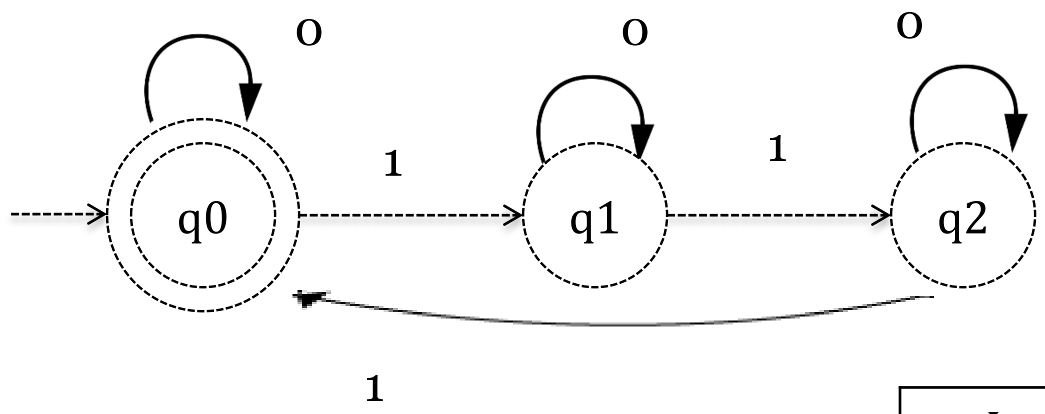
i. 3n ii. 3(n+1) iii. 3(n+2)

Corresponding to three states mentioned above there will be 3 states.

- State q0- no of 1's so far is 3n
- State q1- no of 1's so far is 3n+1
- State q2- no of 1's so far is 3n+2
- Possible transactions that can cause are
- q0 to q1 if machine is in q0
- q1 to q2 if machine is in q1
- q2 to q0 if machine is in q2
- An input of 0 will cause transaction to same state.

Example on Counting of Symbols (DFA)

Solution: 1. Number of 1's is multiple of 3.



Input Symbols	0	1
→q0*	q0	q1
q1	q1	q2
q2	q2	q0

q0 is final state . Machine will be in final state if number of 1's read in Multiple of 3n.
Thus L= { w ∈ {0,1} | no of 1's in multiple of 3. }

Example on Counting of Symbols (DFA)

Solution: 1. Number of 1's is multiple of 3.

- M=(Q, ∑, δ, q_0, F) where

Q = {q0, q1, q2},
∑ = {0, 1},
q_0 = {q0},
F = {q0}, and

Input Symbols	0	1
→ q0*	q0	q1
q1	q1	q2
q2	q2	q0

q0 is final state . Machine will be in final state if number of 1's read in Multiple of 3n.
Thus L= { w ∈ {0,1} | no of 1's in multiple of 3. }

Example on Counting of Symbols (DFA)

- Solution: Number of 1's not multiple of 3.

1. Let L be the language having Number of 1's is multiple of 3 so:

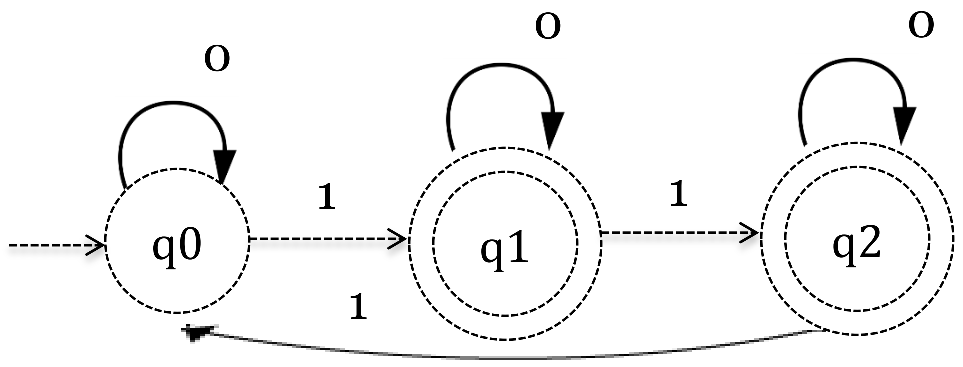
   Thus L= { w ∈ {0,1} | no of 1's in w multiple of 3. }

So Complement of L will be given by

L'= { w ∈ {0,1} | no of 1's in w not multiple of 3. }

So DFA for L' can be obtained by following modifications in L:

1. Every Final state in L becomes non accepting / non final state in L'.
2. Every non Final state in L becomes accepting / final state in L'.



Input Symbols	0	1
→ q0	q0	q1
Q1*	q1	q2
Q2*	q2	q0

q1, q2 are final states. Machine will be in final state if number of 1's read not in Multiple of 3n. Thus L= { w ∈ {0,1} | no of 1's in multiple of 3. }

Example on Substring (DFA)

Ex 1. Draw DFA for the following language over the alphabet {a,b}

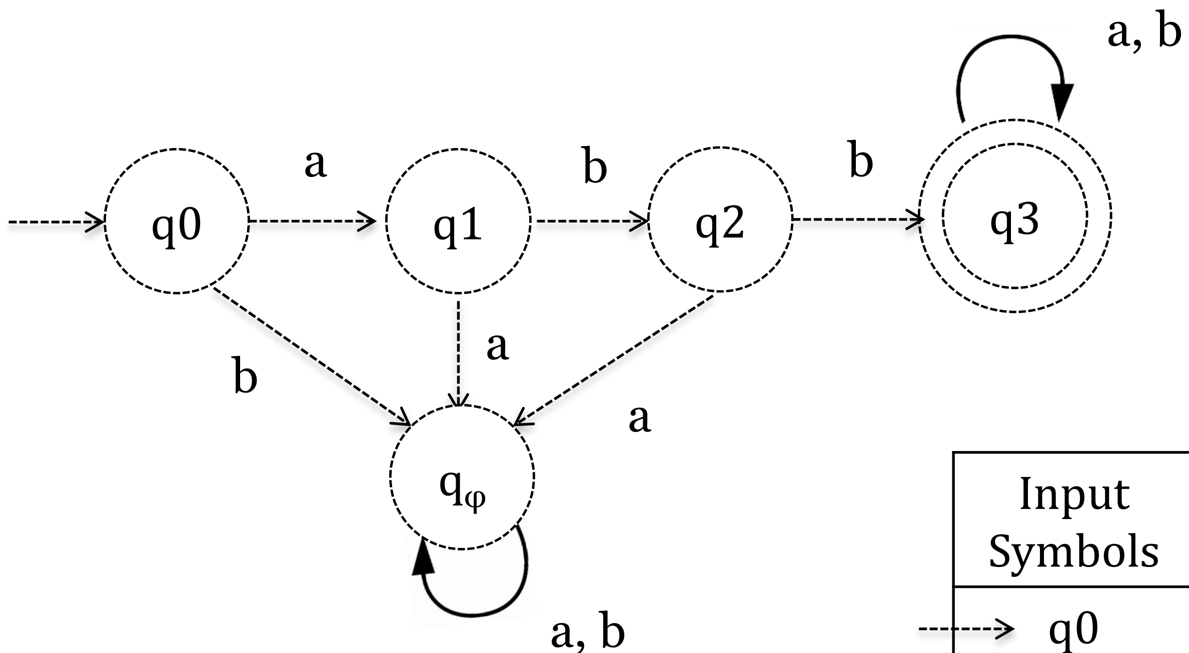
- 1. All strings starting with abb.
- 2. All strings with abb as substring.
- 3. All strings ending in abb.

Solution:

- 1. All strings starting with abb.

- First input b will take the machine to failure State.
- First two input as “aa” will take the machine to failure State.
- First three input as “aba” will take the machine to failure State.
- First three input as “abb” will take the machine to final State.
- So consider the states q_0, q_1, q_2 and q_3.
- Where q0 is initial state and q3 will be the final state.
- Failure state will be represented by q_φ
- SO DFA can be constructed as

Example on Substring (DFA)



State Transition Diagram

Input Symbols	a	b
→ q0	q1	q_φ
q1	q_φ	q2
q2	q_φ	q3
q3*	q3	q3
q_φ	q_φ	q_φ

State Transition table

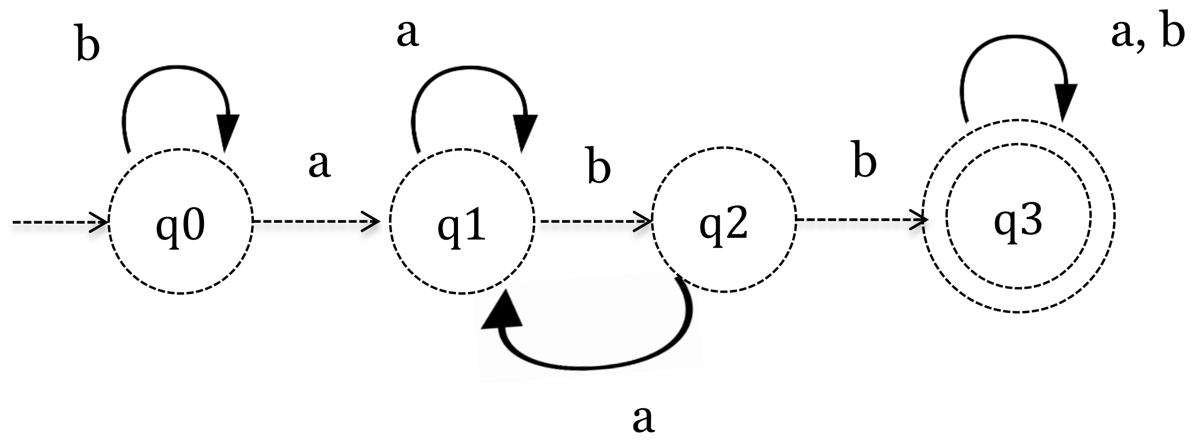
Example on Substring (DFA)

Solution:

➢ 2. All strings with abb as substring.

- q0 is the initial state First input a will take the machine to q1 State.
- For state q1 preceding character is a and required ab to complete abb, so let us take b input to reach next State q2.
- For state q2 preceding characters are “ab” and required b to complete abb, so let us take b input to reach next State q3.
- A input b to state q0 will bring it to same state.
- A input a to state q1 will bring it to same state.
- A input a to state q2 will bring it to q1 state.
- Q3 is the final state.
- So let us draw State Transition Diagram and State Transition table.

Example on Substring (DFA)



State Transition Diagram

Input Symbols	a	b
$\rightarrow$ q0	q_1	q_0
q1	q_1	q_2
q2	q_1	q_3
q3*	q_3	q_3

State Transition table

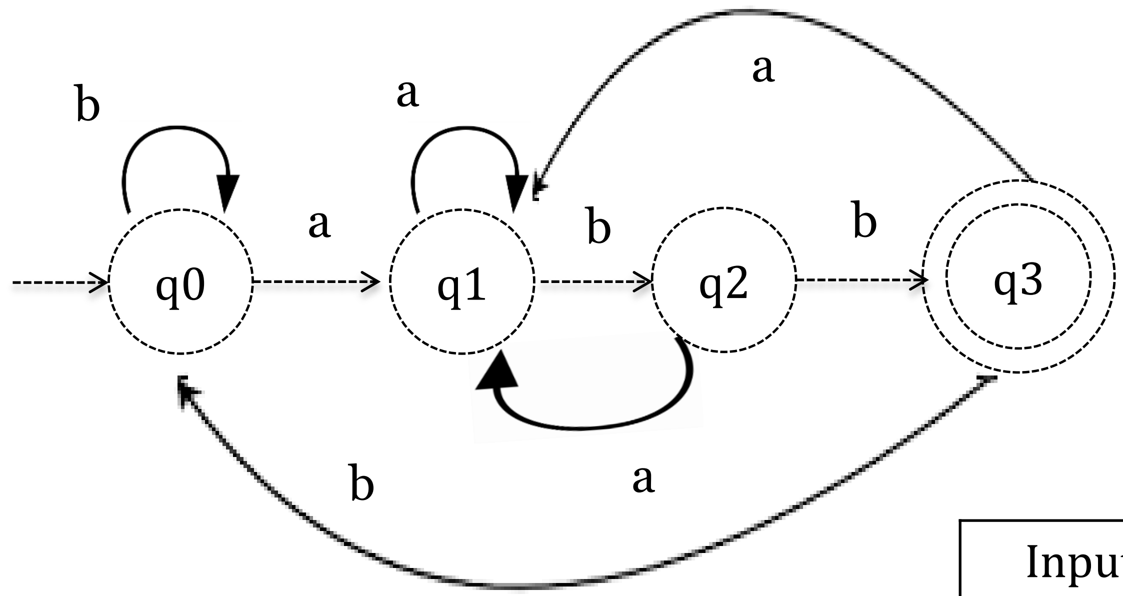
Example on Substring (DFA)

Solution:

➢ 3. All strings ending in abb.

- q0 is the initial state First input a will take the machine to q1 State.
- For state q1 preceding character is a and required ab to complete abb, so let us take b input to reach next State q2.
- For state q2 preceding characters are "ab" and required b to complete abb, so let us take b input to reach next State q3.
- A input b to state q0 will bring it to same state.
- A input a to state q1 will bring it to same state.
- A input a to state q2 will bring it to q1 state.
- A input a to state q3 will bring it to q1 state.
- A input b to state q3 will bring it to q0 state.
- Q3 is the final state.
- So let us draw State Transition Diagram and State Transition table.

Example on Substring (DFA)



State Transition Diagram

Input Symbols	a	b
→ q0	q_1	q_o
q1	q_1	q_2
q2	q_1	q_3
q3*	q_1	q_0

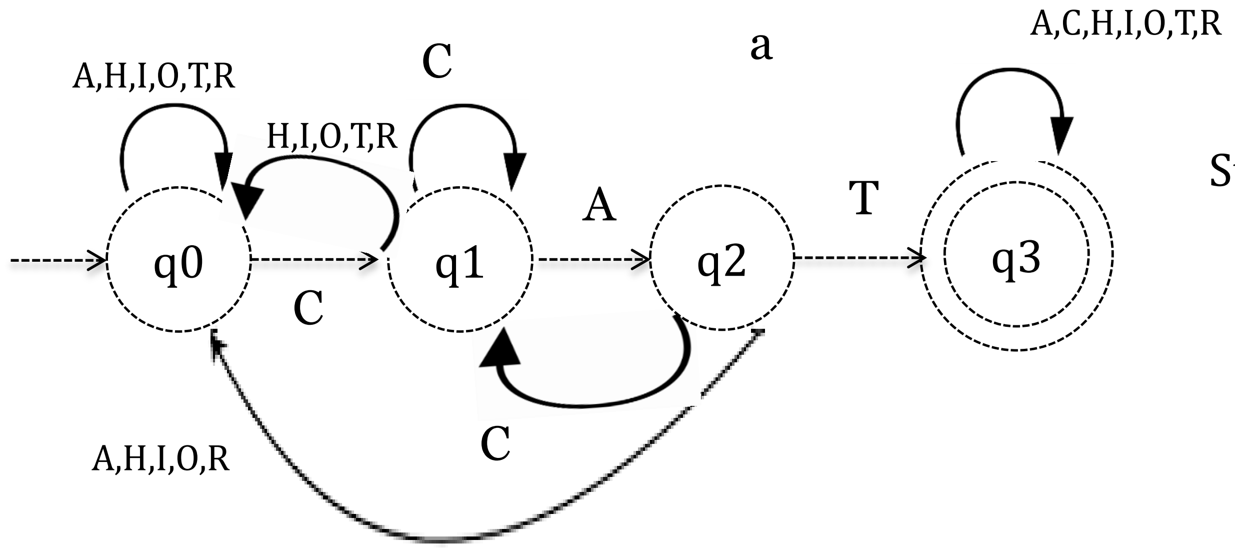
State Transition table

Example on Substring (DFA)

Ex. 2: Design DFA that reads strings made up of letters in the word "CHARIOT" and recognize these strings that contain the word CAT as a substring.

- q0 is the initial state First input C will take the machine to q1 State.
- For state q1 preceding character is C and required AT to complete CAT, so let us take A input to reach next State q2.
- For state q2 preceding characters are "CA" and required T to complete CAT, so let us take T input to reach next State q3.
- Any input (A,H,I,O,T,R) other than C to state q0 will bring it to same state.
- An input C to state q1 will bring it to same state.
- An input C to state q2 will bring it to q1 state and Any input (H,I,O,T,R) other than C & T will bring it to q0 state .
- Any input to state q3 will bring it to same state.
- Q3 is the final state.
- So let us draw State Transition Diagram and State Transition table.

Example on Substring (DFA)



State Transition Diagram

State Transition table

Input Symbols	C	H	A	R	I	O	T
→ q0	q_1	q_o	q_o	q_o	q_o	q_o	q_o
q1	q_1	q_o	q_2	q_o	q_o	q_o	q_o
q2	q_1	q_o	q_o	q_o	q_o	q_0	q_3
q3*	q_3	q_3	q_3	q_3	q_3	q_3	q_3

Example on Divisibility(DFA)

Ex. 1: Design DFA Which can accept a binary number divisible by 3.

Solution:

- A binary number is divisible by 3 if the reminder is 0. So we must derive the machine to determine the final reminder.
- The binary number system having only two symbols {0, 1}.
- The possible reminders could be
- 0→ Associated state q0
- 1→ Associated state q1
- 2→ Associated state q2
- Now let us try to evaluate reminder for binary numbers.
- Consider the following table to calculate reminder.

Example on Divisibility(DFA)

Ex. 1: Design DFA Which can accept a binary number divisible by 3.

Solution:

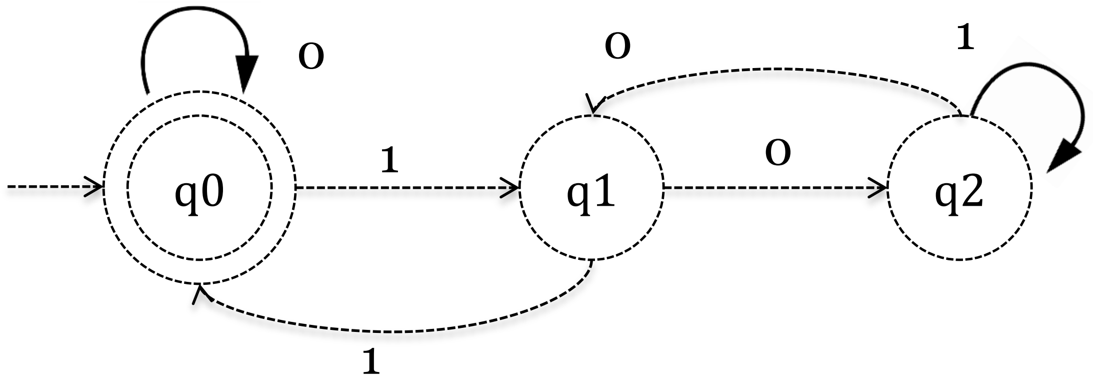
Input	Binary Number	Division of Binary Number By 3	Reminder	Input ends in state
0	00	00 % 3	0	q0
1	01	01 % 3	1	q1
0	10	10 % 3	2	q2
1	11	11 % 3	0	q0
0	100	100 % 3	1	q1
1	101	101 % 3	2	q2
0	110	110 % 3	0	q0

Example on Divisibility(DFA)

Ex. 1: Design DFA Which can accept a binary number divisible by 3.

Solution:

- So now let us draw the DFA for the given problem.



Input Symbols	0	1
→ q0*	q0	q1
q1	q2	q0
q2	q1	q2

Example on Divisibility(DFA)

Ex. 2: Design DFA Which can accept a Decimal number divisible by 3.

Solution:

- A number is divisible by 3 if the reminder is 0. So we must derive the machine to determine the final reminder.
- A decimal Number system having 10 symbols.
- ∑ = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
- The possible reminders could be
-     0→ Associated state q0
-     1→ Associated state q1
-     2→ Associated state q2
- Now let us try to evaluate reminder for decimal numbers.

Example on Divisibility(DFA)

Ex. 2: Design DFA Which can accept a Decimal number divisible by 3.

Solution:

- As there are only three possible reminders so consider the transitions for only three inputs
  - 0 for (0, 3, 6, 9)
  - 1 for (1, 4, 7)
  - 2 for (2, 5, 8)
- Now let us draw the table to derive the reminders for decimal number.

Example on Divisibility(DFA)

Ex. 1: Design DFA Which can accept a decimal number divisible by 3.

Solution:

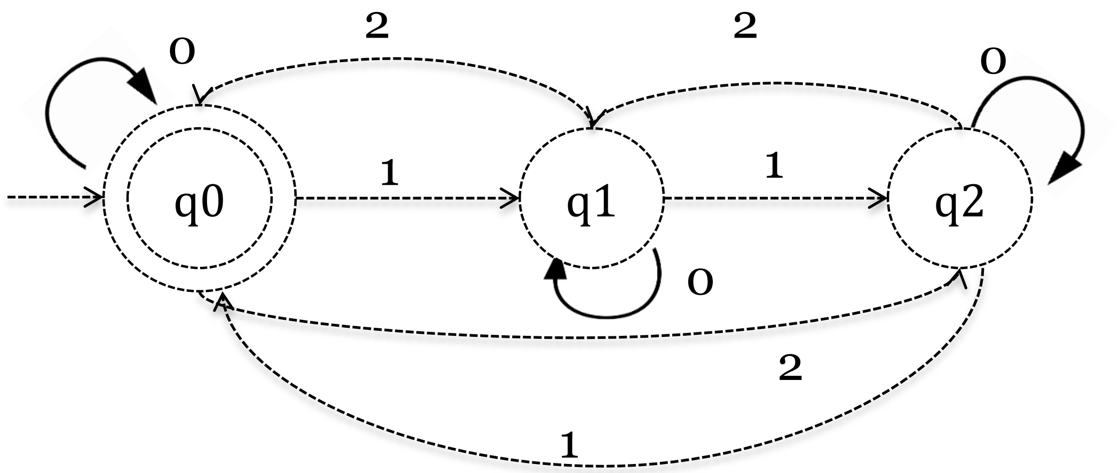
Input	Decimal Number	Division of Binary Number By 3	Reminder	Input ends in state
0	0	0 % 3	0	q0
1	1	1 % 3	1	q1
2	2	2 % 3	2	q2
0	3	3 % 3	0	q0
1	4	4 % 3	1	q1
2	5	5 % 3	2	q2
0	6	6 % 3	0	q0
1	7	7 % 3	1	q1
2	8	8 % 3	2	q2
0	9	9 % 3	0	q0
1	10	10 % 3	1	q1

Example on Divisibility(DFA)

Ex. 1: Design DFA Which can accept a decimal number divisible by 3.

Solution:

- So now let us draw the DFA for the given problem.



Input Symbols	0	1	2
→ q0*	q0	q1	q2
q1	q1	q2	q0
q2	q2	q0	q1

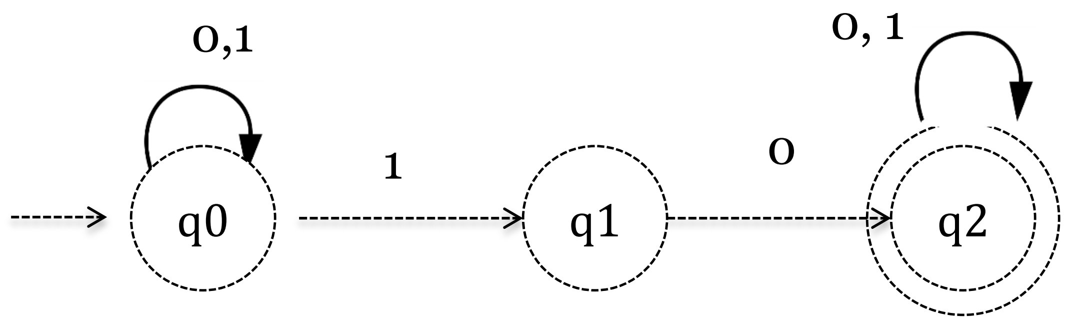
Non Deterministic Finite Automata (NFA)

- In NFA, for each input symbol, one can determine the multiple states to which the machine will move. Hence, it is called Non **Deterministic Automaton**.
- In NFA, FA can reside in multiple states at same time for single input symbol.
- As it has a finite number of states, the machine is called **Deterministic Finite Machine** or **Deterministic Finite Automaton.**

- A NFA can be represented by a 5-tuple M=(Q, ∑, δ, q_0, F) where –

1. **Q** is a finite set of states.
2. **∑** is a finite set of symbols called the alphabet.
3. **δ** is the transition function where $\delta: Q \times \sum \rightarrow Q$
4. **q_0** is the initial state from where any input is processed ($q_0 \in Q$).
5. **F** is a set of final state/states of Q ($F \subseteq Q$).

Example of NFA



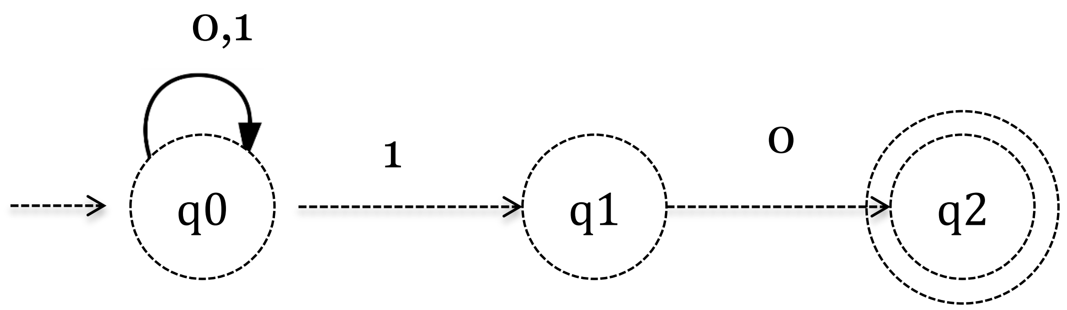
State Transition Diagram

Input Symbols	0	1
→q0	{q0}	{q0,q1}
q1	{q2}	φ
q2*	{q2}	{q2}

State Transition table

Processing of String by NFA

A string w ∈ ∑* is accepted by NFA M if δ* =(q0,w) containing a final state. Let us See how the String 011010 is processed by given NFA.



State Transition Diagram

Processing of String by NFA 011010

$$\begin{tikzcd}
q_0 \arrow[r, dashed, "0"] & q_0 \arrow[r, dashed, "1"] \arrow[d, dashed, "1"'] & q_0 \arrow[r, dashed, "1"] \arrow[d, dashed, "1"'] & q_0 \arrow[r, dashed, "0"] & q_0 \arrow[r, dashed, "1"] \arrow[d, dashed, "1"'] & q_0 \arrow[r, dashed, "0"] & q_0 \\
& q_1 \arrow[d, dashed, "1"'] & q_1 \arrow[d, dashed, "0"'] & & q_1 \arrow[d, dashed, "0"'] & & \\
& \varphi \arrow[d, dashed, "0"'] & q_2 \arrow[d, dashed, "1"'] & & q_2 & & \\
& \varphi \arrow[d, dashed, "1"'] & \varphi \arrow[d, dashed, "0"'] & & & & \\
& \varphi \arrow[d, dashed, "0"'] & \varphi & & & & \\
& \varphi & & & & &
\end{tikzcd}$$

$$\begin{tikzcd}
\{q_0\} \arrow[d, dashed, "0"'] \\
\{q_0\} \arrow[d, dashed, "1"'] \\
\{q_0, q_1\} \arrow[d, dashed, "1"'] \\
\{q_0, q_1, \varphi\} \arrow[d, dashed, "0"'] \\
\{q_0, q_2, \varphi\} \arrow[d, dashed, "1"'] \\
\{q_0, q_1, \varphi\} \arrow[d, dashed, "0"'] \\
\{q_0, q_2, \varphi\}
\end{tikzcd}$$

Non Deterministic Finite Automata (NFA)

- On input 011010 the automata can take any of the following four Path.

1. $q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_0 \xrightarrow{1} q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_0 \xrightarrow{0} q_0$

2. $q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_0 \xrightarrow{1} q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_1 \xrightarrow{0} q_2$

3. $q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_0 \xrightarrow{1} q_1 \xrightarrow{0} q_2 \xrightarrow{1} \varphi \xrightarrow{0} \varphi$

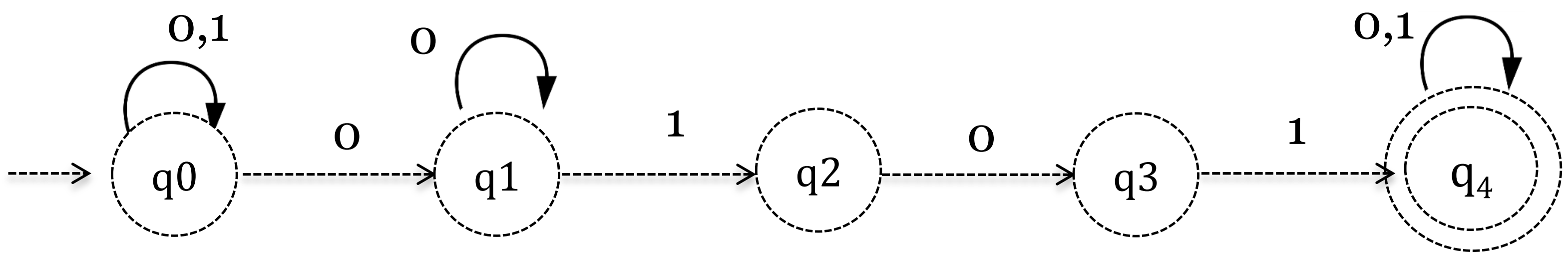
4. $q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_1 \xrightarrow{1} \varphi \xrightarrow{0} \varphi \xrightarrow{1} \varphi \xrightarrow{0} \varphi$

So from above we can conclude that the given string is accepted as one of the solution (2)

Is ending to final state. Also every string ending with 10 is accepted by NFA.

Examples on NFA

EX.1 Draw NFA to accept strings containing the substring 00101.



State Transition Diagram

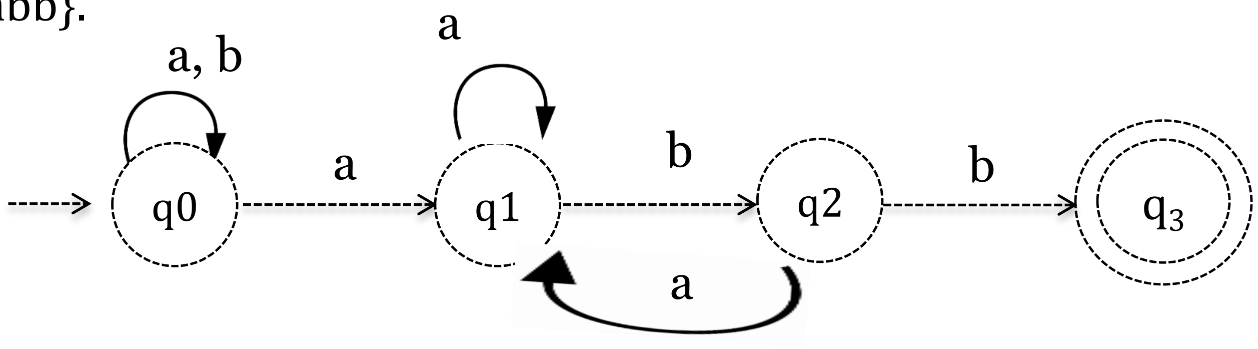
- To reach the final string from q0 to final state q4 a substring 0101 is required.
- Any string containing sub string 0101 is accepted by above NFA.

Input Symbols	0	1
→q0	{q0,q1}	{q0}
q1	{q1}	{q2}
q2	{q3}	φ
q3	φ	{q4}
q4*	{q4}	{q4}

State Transition Table

Examples on NFA

EX.2 Draw NFA that accept various string from Language L={x ∈ {a,b} | x ends with abb}.



State Transition Diagram

- To reach the final string from q0 to final state q4 a substring aab is required.
- Any string containing sub string aab is accepted by above NFA.

Input Symbols	a	b
→ q0	{q0,q1}	{q0}
q1	{q1}	{q2}
q2	q1	{q3}
q3*	φ	φ

State Transition Table

Examples on NFA

EX.3 Consider the NFA with states 1-5 and input alphabet {a, b} has following transaction table.

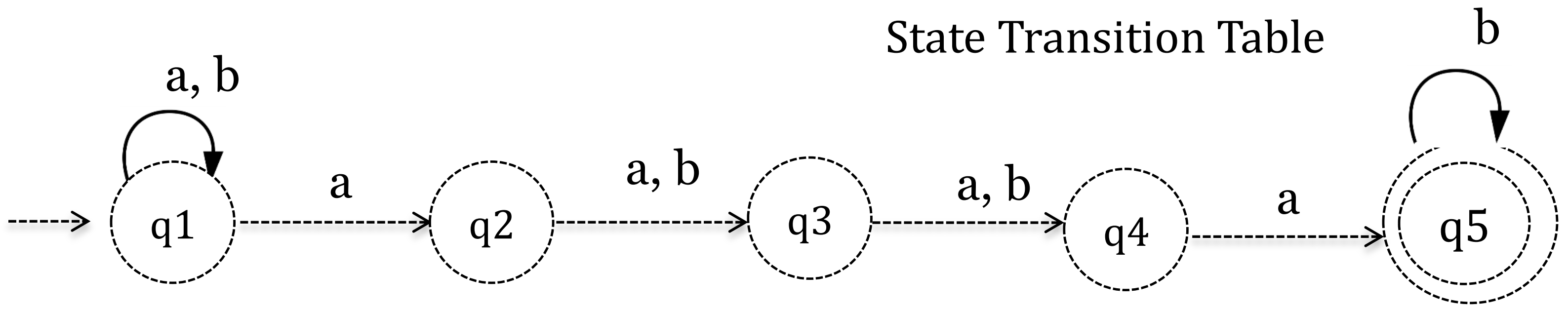
- Draw a Transaction Diagram
- Calculate δ*(1, ab)
- Calculate δ*(1, abaab)

Input Symbols	a	b
→ q1	{1,2}	{1}
q2	{3}	{3}
q3	{4}	{4}
q4	{5}	φ
q5	φ	{5}

State Transition Table

Solution:

1.



State Transition Diagram

Examples on NFA

Solution:

2.

$\delta^*(1, ab) = \delta((\delta(1, a), b)$

$= \delta((\delta(1, 2), b)$

$= \delta(1, b), \cup \delta(2, b) = \{1, 3\}$

3. $\delta^*(1, abaab) = \delta^*(\delta(1, a), baab) = \delta^*((1, 2), baab) = \delta^*(\delta(1,2),b),aab)$

$= \delta^*(1, 3), aab)$

$= \delta^*(1, 2, 4), ab)$

$= \delta(1, 2, 3, 5), b)$

$= \{1, 3, 4, 5\}$

NFA to DFA Conversion

- The conversion of NFA to equivalent DFA is based on Subset construction. If a NFA consists of three states, Q= {q0,q1,q2}
- The Machine could be in any of the following states:

1. φ
2. (q0), (q1), (q2)
3. (q0, q1), (q1, q2), (q0, q2)
4. (q0, q1, q2)

- The no of possible states are 2^Q
- The procedure of finding whether the given string is accepted by NFA includes following steps.

1. Finding all possible paths followed by machine for given string.
2. Finding the set of states reached final state from the starting state by applying the string.
3. If the set of states obtained in step 2 contain a final state the given string is accepted by NFA.

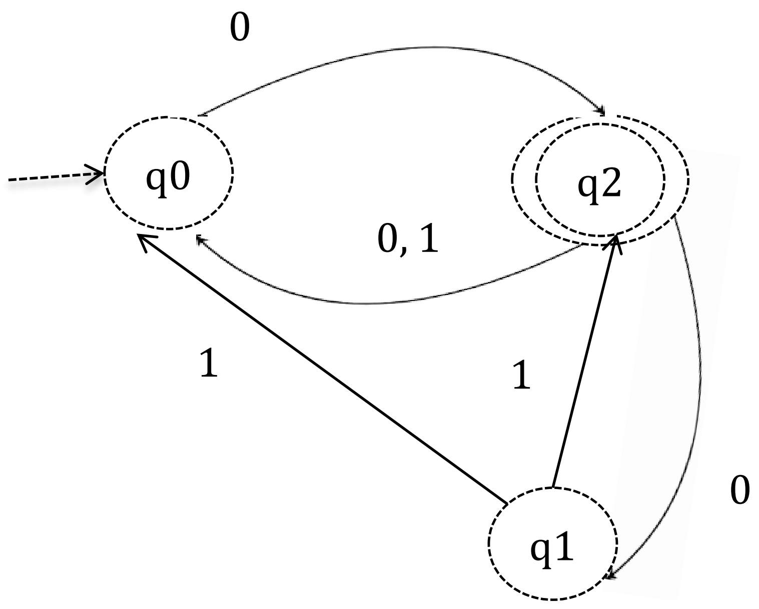
- This procedure can be applied to any string by constructing a successor table.

NFA to DFA Conversion

- Step 1: Initially Q' = ϕ
- Step 2: Add q0 of NFA to Q'. Then find the transitions from this start state.
- Step 3: In Q', find the possible set of states for each input symbol. If this set of states is not in Q', then add it to Q'.
- Step 4: In DFA, the final state will be all the states which contain F(final states of NFA)

NFA to DFA Conversion Example-1

Ex. 1: Construct the Equivalent DFA from given NFA.

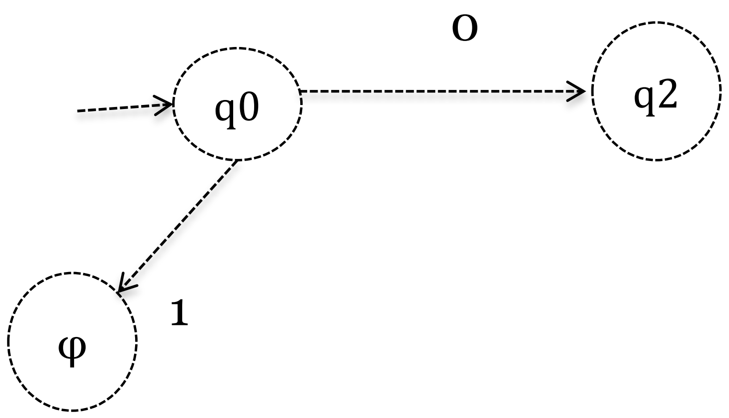


Input Symbols	0	1
q0	{q2}	φ
q1	φ	{q0, q2}
q2	{q0, q1}	{q0}

Solution: Construct the successor for given NFA.
Step by step as explained follow

NFA to DFA Conversion Example-1

Step 1: Starting State {q0} is the first sub state.
0 successor for q0 is q2 i.e δ: (q0, 0)→ q2
1 successor for q0 is qφ i.e δ: (q0, 1)→ qφ

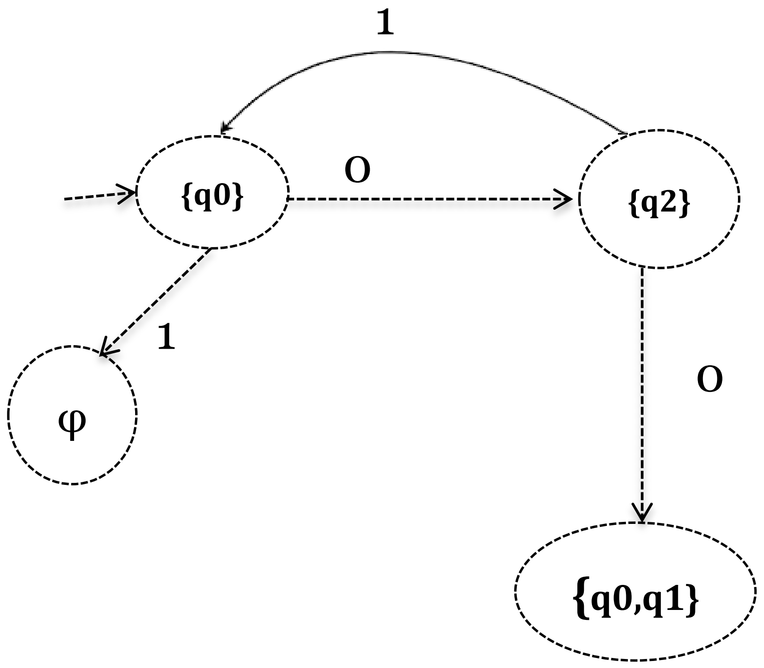


Input Symbols	0	1
{q0}	{q2}	φ

So in this step new substate q2 is generated so find the successor for q2

NFA to DFA Conversion Example-1

Step 2: Consider sub State {q2} is the recently generated sub state.
O successor for q2 is q2 i.e δ: (q0, 0)→ (q0, q1)
1 successor for q2 is qφ i.e δ: (q0, 1)→ qo

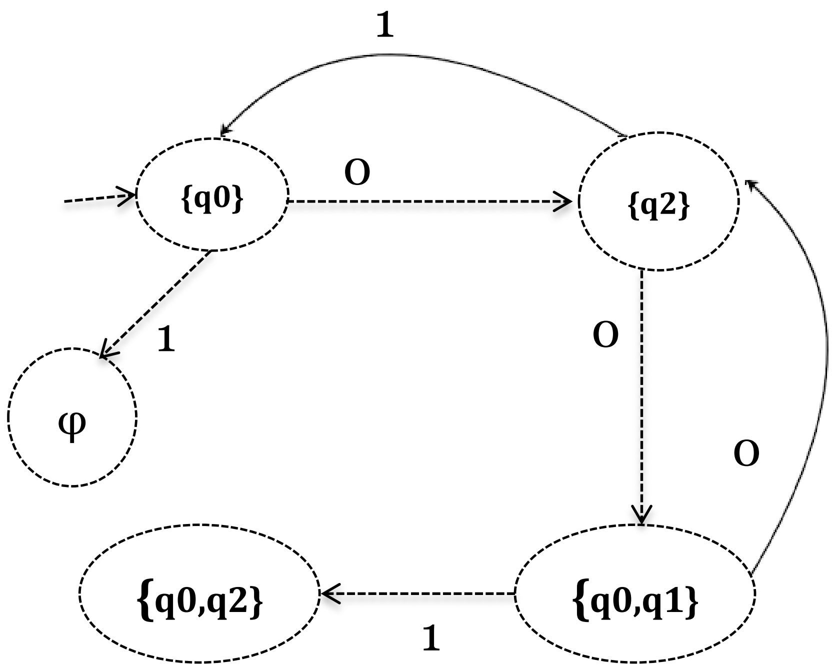


Input Symbols	0	1
{q0}	{q2}	φ
{q2}	{q0, q1}	{q0}

So in this step new sub state {q0, q1} is generated so find the successor for {q0, q1}

NFA to DFA Conversion Example-1

Step 3: Consider sub State {q0, q1} is the recently generated sub state.
O successor for {q0, q1} is q2 i.e δ: ((q0,q1), 0)→ (q2)
1 successor for {q0, q1} are q0, q2 i.e δ: ((q0,q1), 1)→ (q0, q2)

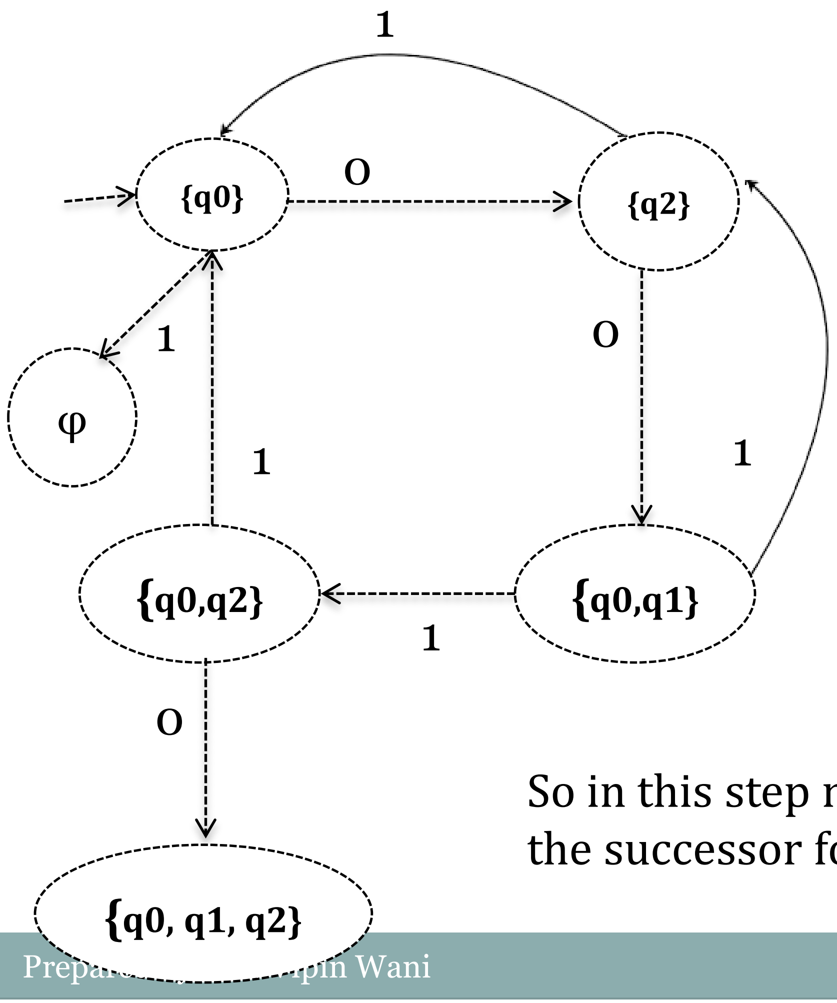


Input Symbols	0	1
{q0}	{q2}	φ
{q2}	{q0, q1}	{q0}
{q0, q1}	{q2}	{q0, q2}

So in this step new sub state {q0, q2} is generated so find the successor for {q0, q2}

NFA to DFA Conversion Example-1

Step 4: Consider sub State {q0, q2} is the recently generated sub state.
O successor for {q0, q2} are q0, q1, q2 i.e δ: ((q0,q2), 0)→ (q0, q1, q2)
1 successor for {q0, q2} is q0 i.e δ: ((q0,q2), 1)→ (q0)

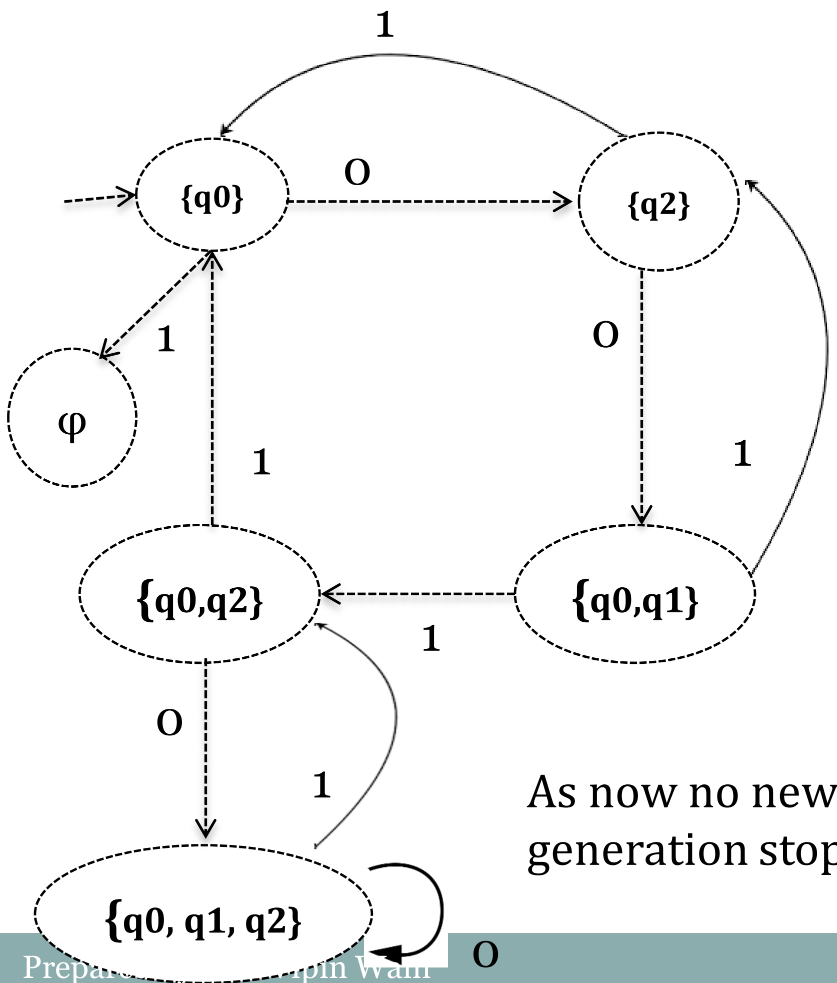


Input Symbols	0	1
{q0}	{q2}	φ
{q2}	{q0, q1}	{q0}
{q0, q1}	{q2}	{q0, q2}
{q0, q2}	{q0, q1, q2}	{q0}

So in this step new sub state {q0, q1, q2} is generated so find the successor for {q0, q1, q2}

NFA to DFA Conversion Example-1

Step 5: Consider sub State {q0, q1, q2} is the recently generated sub state.
O successor for {q0, q1, q2} are q0, q1, q2 i.e δ: ((q0, q1, q2), 0)→ (q0, q1, q2)
1 successor for {q0, q1, q2} are q0, q2 i.e δ: ((q0, q1, q2), 1)→ (q0, q2)

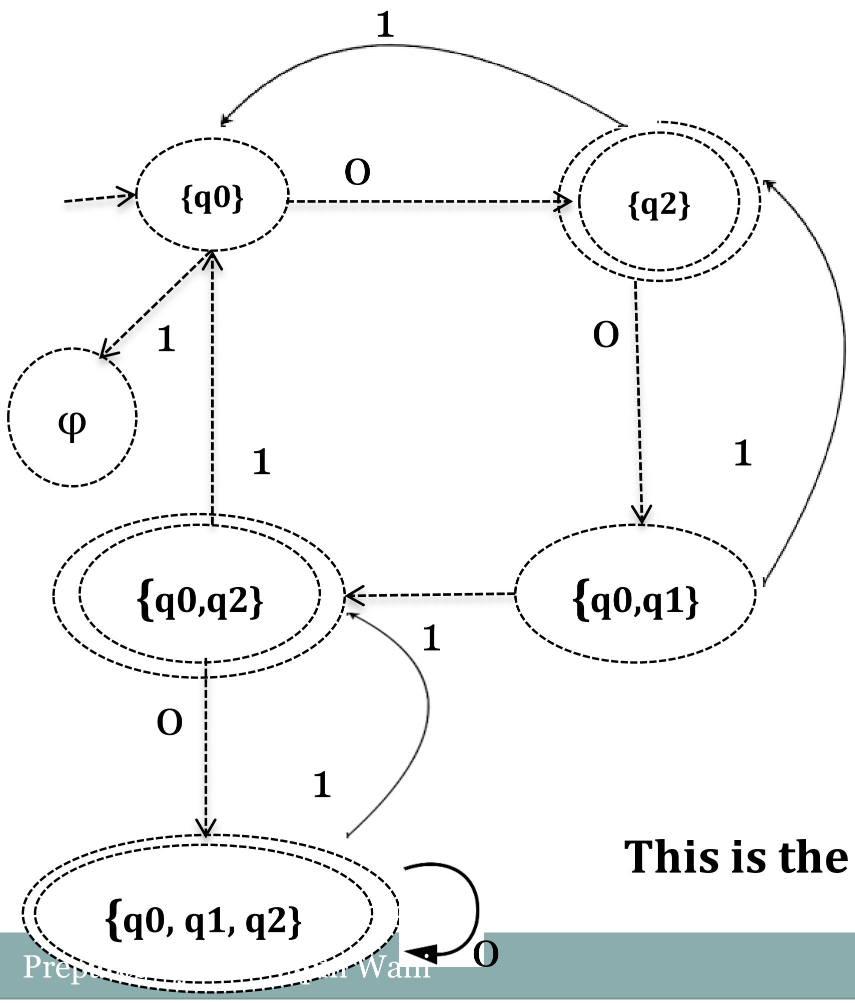


Input Symbols	0	1
{q0}	{q2}	φ
{q2}	{q0, q1}	{q0}
{q0, q1}	{q2}	{q0, q2}
{q0, q2}	{q0, q1, q2}	{q0}
{q0, q1, q2}	{q0, q1, q2}	{q0, q2}

As now no new sub state is generated so the process of sub set generation stops.

NFA to DFA Conversion Example-1

Step 6: As the q2 is the final state in NFA so every sub state containing q2 will be final state in DFA.



Input Symbols	0	1
→{q0}	{q2}	φ
{q2}*	{q0, q1}	{q0}
{q0, q1}	{q2}	{q0, q2}
{q0, q2}*	{q0, q1, q2}	{q0}
{q0, q1, q2}*	{q0, q1, q2}	{q0, q2}

This is the final DFA after representing Final states.

NFA to DFA Conversion Example-2

Ex. 2: Construct the Equivalent DFA from given NFA.

Input Symbols	0	1
p	{p, q}	{p}
q	{r}	{r}
r	{s}	φ
s*	{s}	{s}

Solution: Start the subset generation step by step.
The maximum number of possible sub states are $2^{Q} = 16$.

NFA to DFA Conversion Example-2

Input Symbols	0	1
{p}	{p, q}	{p}
{p, q}	{p, q, r}	{p, r}
{p, q, r}	{p, q, r, s}	{p, r}
{p, r}	{p, q, s}	{p}
{p, q, s}	{p, q, r, s}	{p, r, s}
{p, q, r, s}	{p, q, r, s}	{p, r, s}
{p, r, s}	{p, q, s}	{p, s}
{p, s}	{p, q, s}	{p, s}

A new subset {p, q} is generated

A new subset {p, q, r} & {p, r} are generated

A new subset {p, q, r, s} is generated

A new subset {p, q, s} is generated

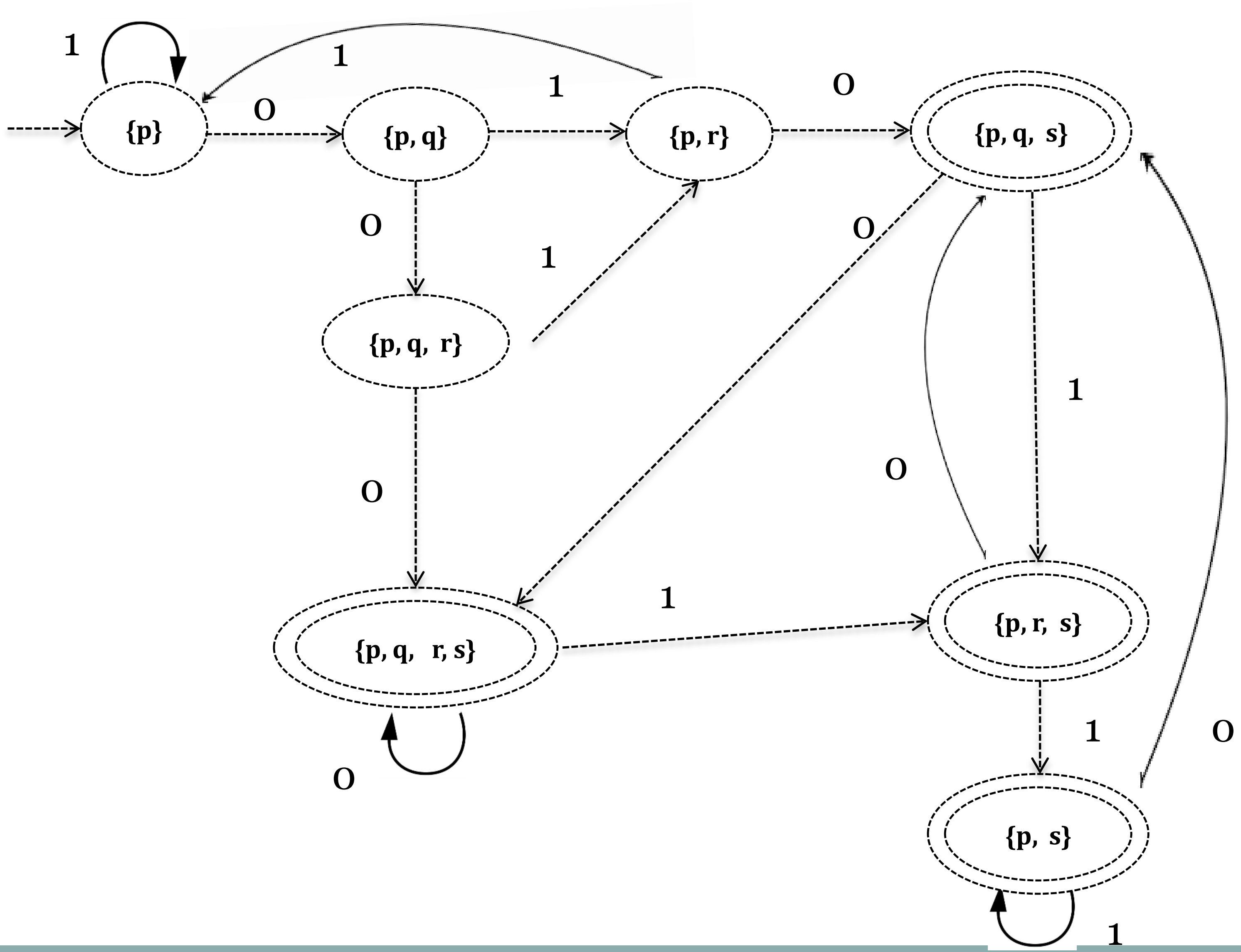
A new subset {p, r, s} is generated

A new subset {p, s} is generated

A new subset {p, q, s} is generated

No new sub state so stop.

NFA to DFA Conversion Examples-2



NFA to DFA Conversion Example-3

Ex. 3: Construct the Equivalent DFA from given NFA.

Input Symbols	0	1
--------> p	{p, q}	{p}
q	{r, s}	{t}
r	{p, r}	{t}
s*	φ	φ
t*	φ	φ

Solution: Start the subset generation step by step.
The maximum number of possible sub states are $2^{Q} = 32$.

NFA to DFA Conversion Example-3

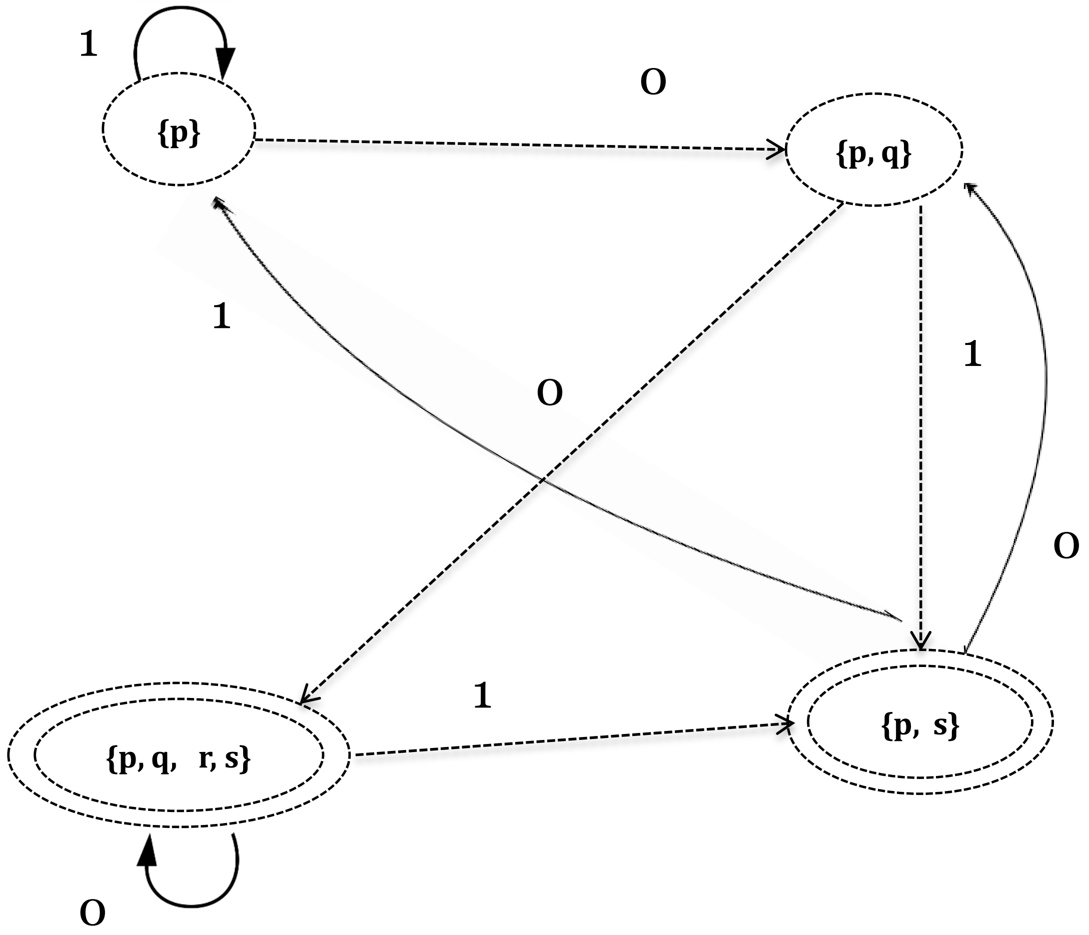
Input Symbols	0	1
{p}	{p, q}	{p}
{p, q}	{p, q, r, s}	{p, t}
{p, q, r, s}	{p, q, r, s}	{p, t}
{p, t}	{p, q}	{p}

A new subset {p, q} is generated

A new subset {p, q, r, s} & {p, t} are generated

No new sub state so stop.

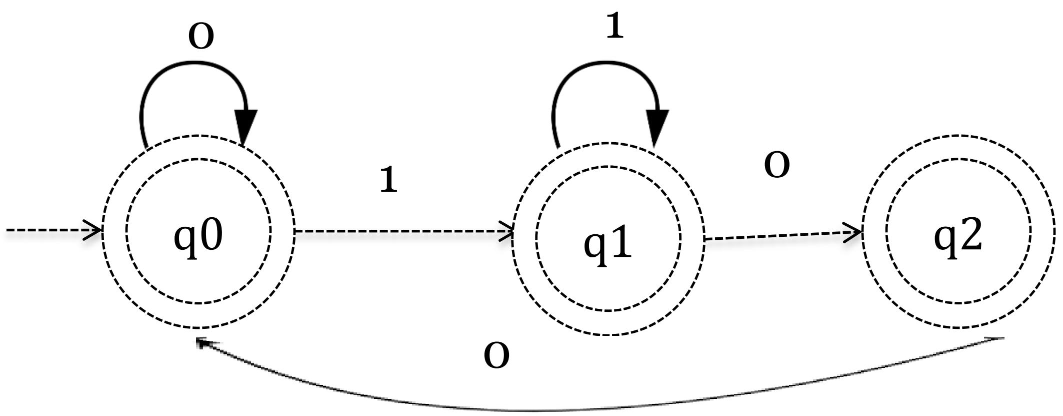
NFA to DFA Conversion Example-3



Construction of NFA & DFA Example-1

Ex. 1: Construct the NFA and then Equivalent DFA accepting strings over {0,1}
For accepting all possible strings of Zero and ones not containing 101 as sub string.

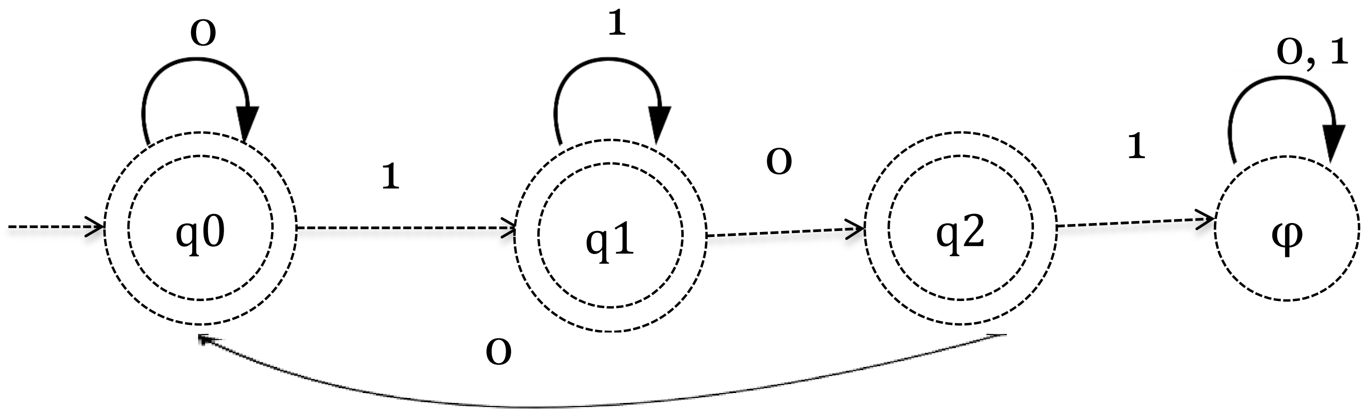
Solution: This problem is fully deterministic. A DFA with φ transitions not mentioned
Is Called as NFA



- State q0 is initial state which will cause machine to move in state q1 on input1.
- State q1 will cause machine to move in state q2 on input 0.
- State q2 will cause machine to move in state q1 on input 0 and on input 1 it will move
- to failure state φ.

Construction of NFA & DFA Example-1

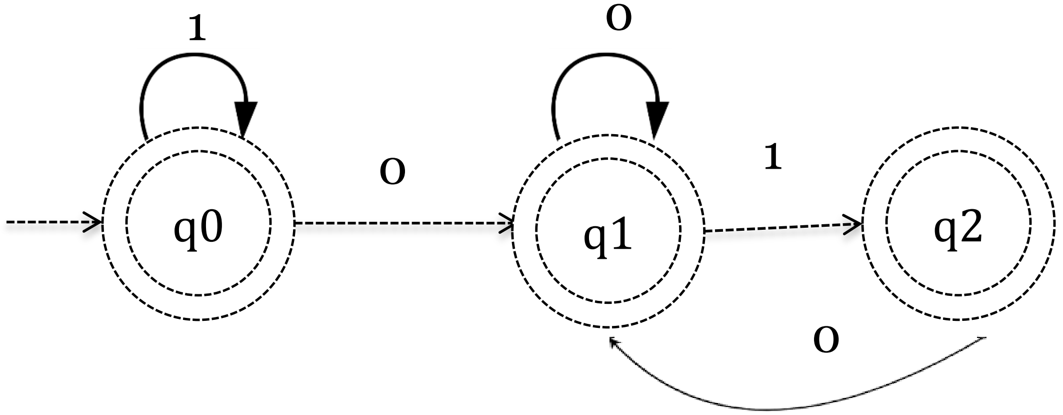
- NFA can be converted into DFA by introducing an explicate failure state.
- So the equivalent DFA can be constructed as follow



Construction of NFA & DFA Example-2

Ex. 1: Construct the NFA and then Equivalent DFA accepting strings over {0,1}
For accepting all possible strings of Zero and ones not containing 011 as sub string.

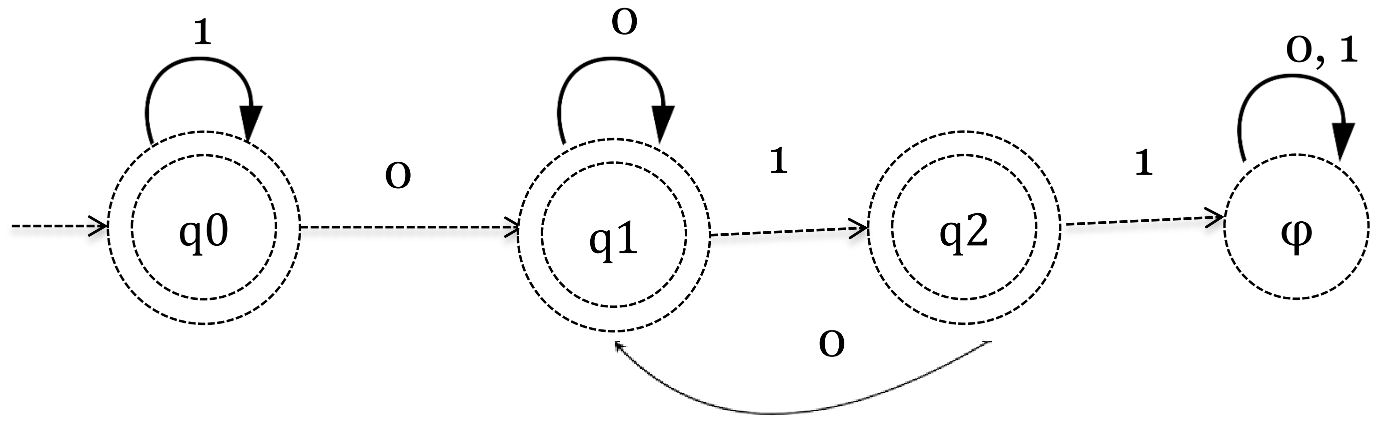
Solution: This problem is fully deterministic. A DFA with φ transitions not mentioned
Is Called as NFA



- State q0 is initial state which will cause machine to move in state q1 on input 0.
- State q1 will cause machine to move in state q2 on input 1.
- State q2 will cause machine to move in state q1 on input 0 and on input 1 it will move
- to failure state φ.

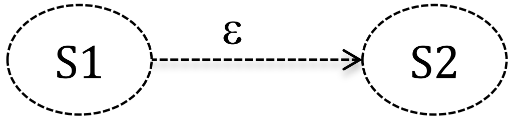
Construction of NFA & DFA Example-1

- NFA can be converted into DFA by introducing an explicate failure state.
- So the equivalent DFA can be constructed as follow



NFA with ε –Transitions

- ε stands for a null symbol. An ε transition allows transition on ε or no input the empty string . This implies that a Machine can make a transition without any input.
- When finding the string describe by path containing arc with label ε, the ε symbols are discarded.
- The use of ε transition may specify the transition graph by reducing numbers of labeled arcs.
- **Steps to remove e transition from NFA.**

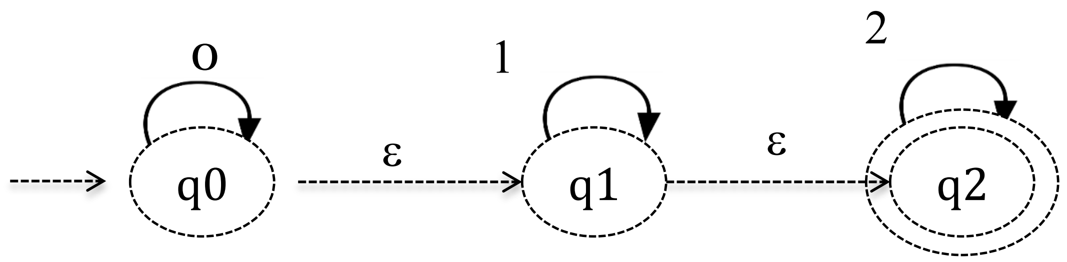


Consider the ε transition from state S1 to s2 then follow the following steps to remove transition .

1. Find all edges starting from S2.
2. Duplicate all edges to S1 without changing the edge label.
3. If s1 is initial state then mark s2 also as initial state.
4. If s2 is final state then mark s1 also as final state.
5. Remove the dead states.

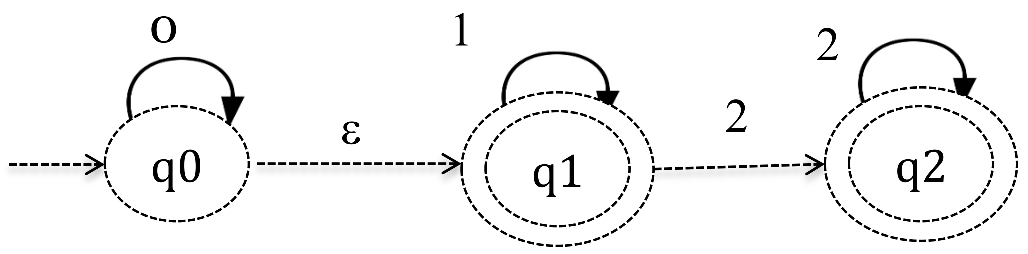
NFA with ε –Transitions Example

Ex. 1: Use direct methods to find an equivalent NFA without ε moves for the given NFA bellow.

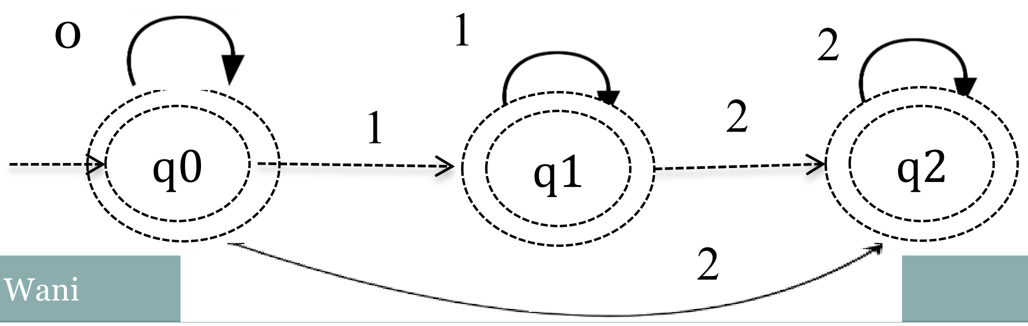


Solution:

Step 1: Transition from q2 duplicated on q1, and make a q1 as final state.

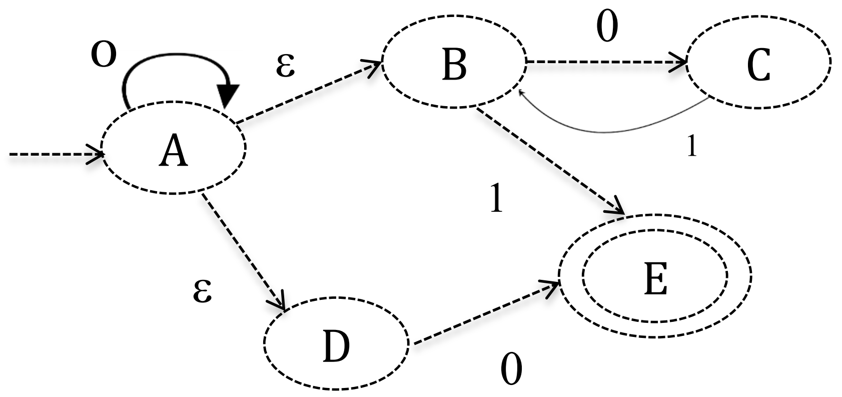


Step 2: Transition from q1 is duplicated on q0, and make a q0 as final state.

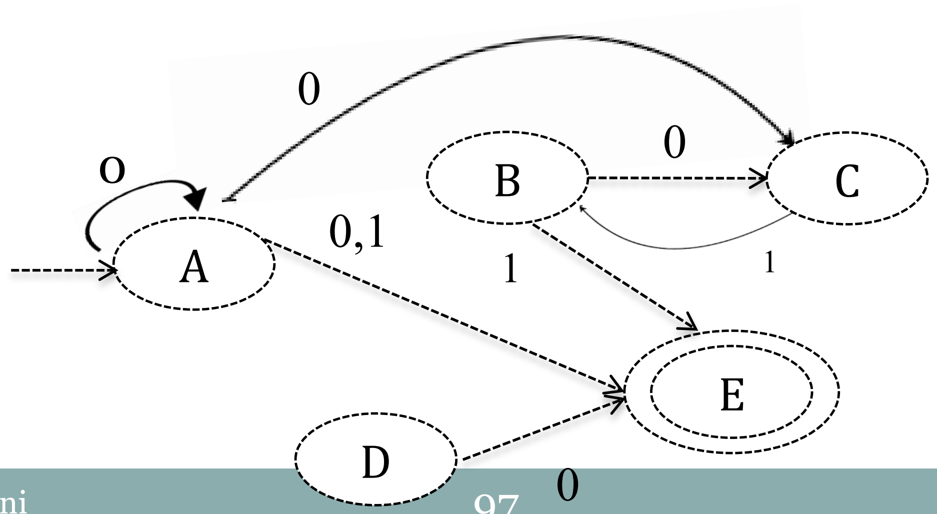


NFA with ε –Transitions Example-2

Ex. 2: Use direct methods to find an equivalent NFA without ε moves for the given NFA bellow.

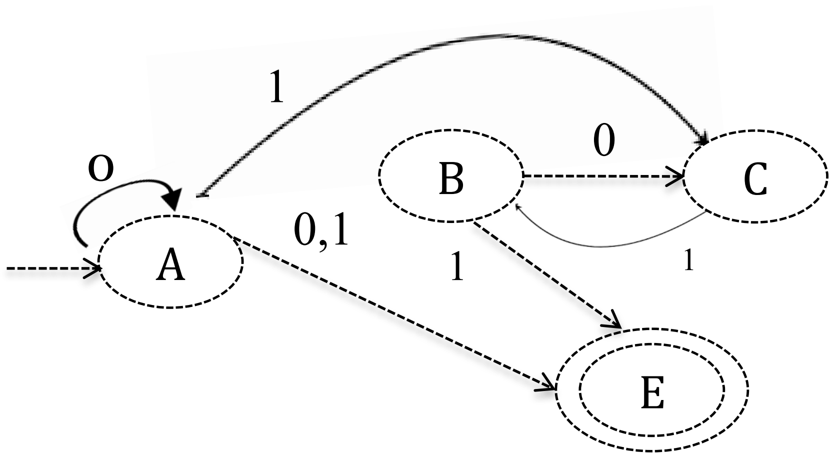


Outgoing transitions from B & D are duplicated on A as δ (A, ε) = B and δ (A, ε) = D.



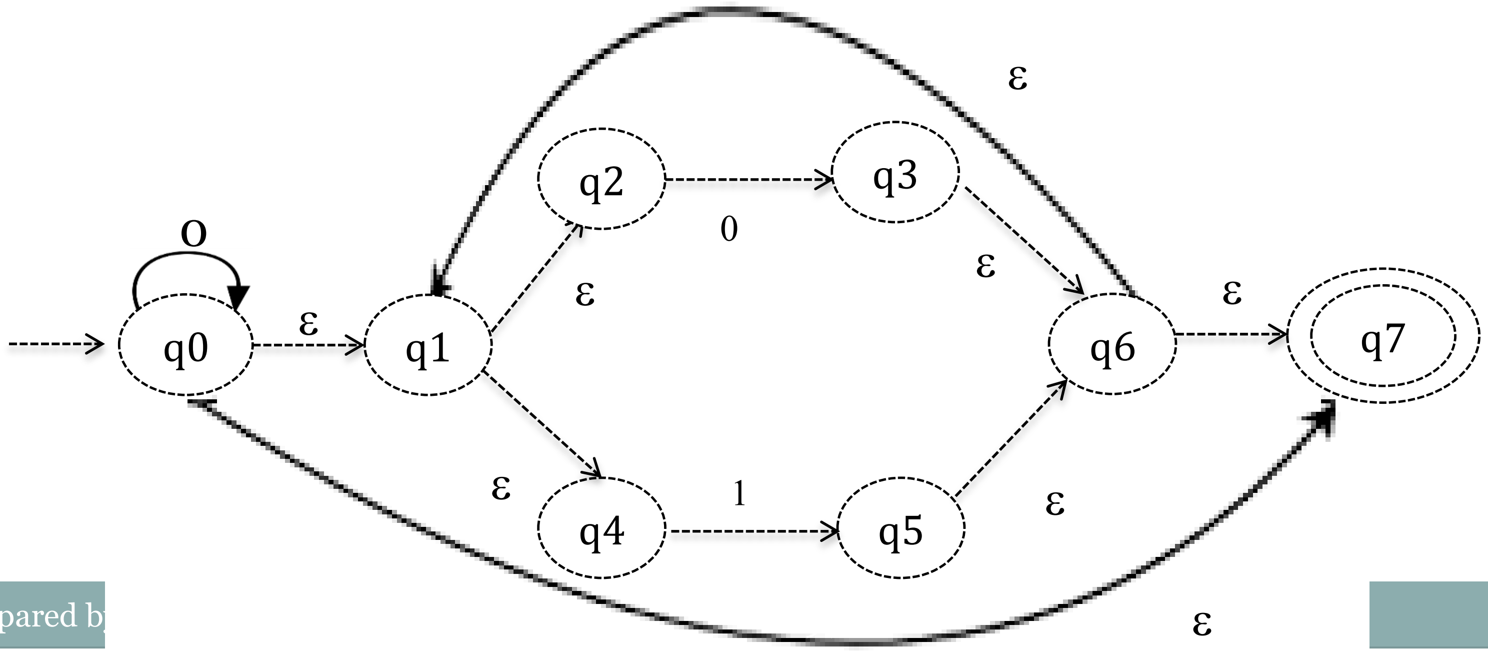
NFA with ε –Transitions Example-2

The state D can be deleted as it don’t have any incoming transitions.



ε –Closures

- Give formal definition of ε closures with suitable example. for a null symbol.
- ε Closures of state qi is the set of states including qi where qi can be reached by any numbers of ε moves of given non deterministic finite automata.
- ε Closure of qi
- ε Closure of qi, includes qi
- Set of states reachable from qi on ε moves.
- Set of states reachable from existing states in ε closure using ε moves and so on .



ε –Closures Example

- Find ε Closure for given NFA.
- ε Closures of state qi is the set of states including qi where qi can be reached by any numbers of ε moves of given non deterministic finite automata.
- so let us determine the ε Closures for each sub state.
- Step 1: ε Closures of q0= {q0, q1, q2, q4, q7}

  The ε moves from q0 are = {q0 to q1,
  q0 to q7,
  q1 to q2,
  q1 to q4. }
- Step 2: ε Closures of q1= {q1, q2, q4}

The ε moves from q1 are = { q1 to q2,
q1 to q4. }

- Step 3: ε Closures of q2= { q2 }
- There are no any ε moves from q2

ε –Closures Example

- Step 4: ε Closures of q3= {q3, q6, q7, q1, q2, q4}

  The ε moves from q3 are = {q3 to q6,
  q6 to q7,
  q6 to q1
  q1 to q2,
  q1 to q4. }

- Step 5: ε Closures of q4= { q4 }
- There are no any ε moves from q4
- Step 6: ε Closures of q5= {q5, q6, q7, q1, q2, q4}

The ε moves from q5 are = { q5 to q6,
q6 to q7,
q6 to q1,
q1 to q2,
q1 to q4}

ε –Closures Example

- Step 7: ε Closures of q6= { q6, q7, q1, q2, q4}

  The ε moves from q6 are = { q6 to q7,
  q6 to q1,
  q1 to q2,
  q1 to q4}

- Step 8: ε Closures of q7= { q7 }
- There are no any ε moves from q7

- So the ε Closures of all states can be

Summarized in table shown bellow.

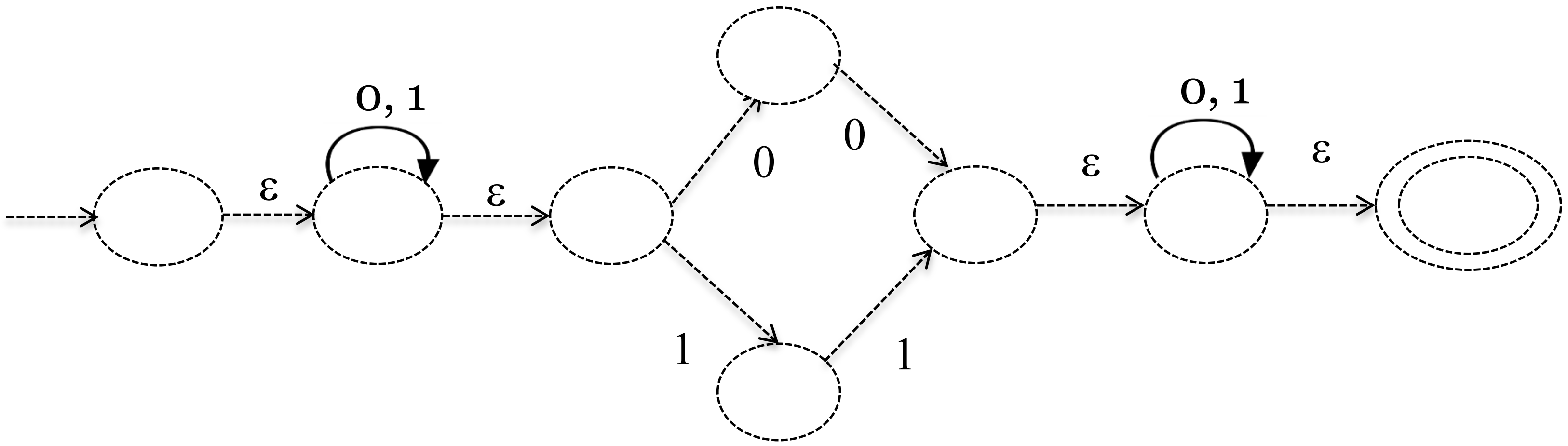
State	ε -Closures
q0	{q0, q1, q2, q4, q7}
q1	{q1, q2, q4}
q2	{q2}
q3	{q3, q6, q7, q1, q2, q4}
q4	{ q4 }
q5	{q5, q6, q7, q1, q2, q4}
q6	{ q6, q7, q1, q2, q4}
q7	{q7}

Construction of NFA & DFA Example

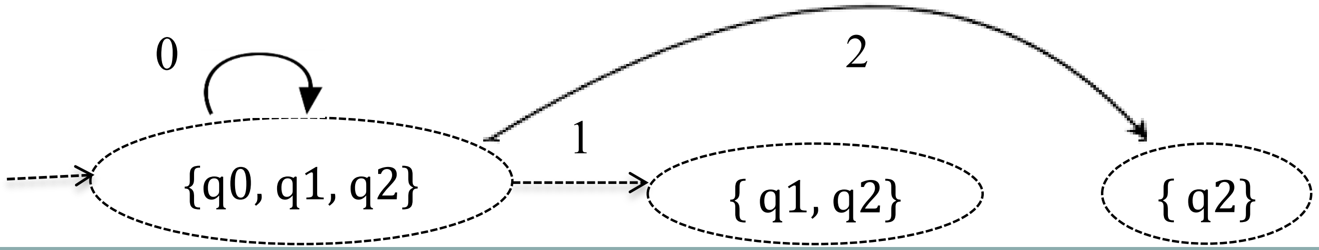
Construct NFA and then DFA with reduced states equivalent to

(0+1)* (00+11) (0+1)*

Solution: Construction of above mentioned NFA.

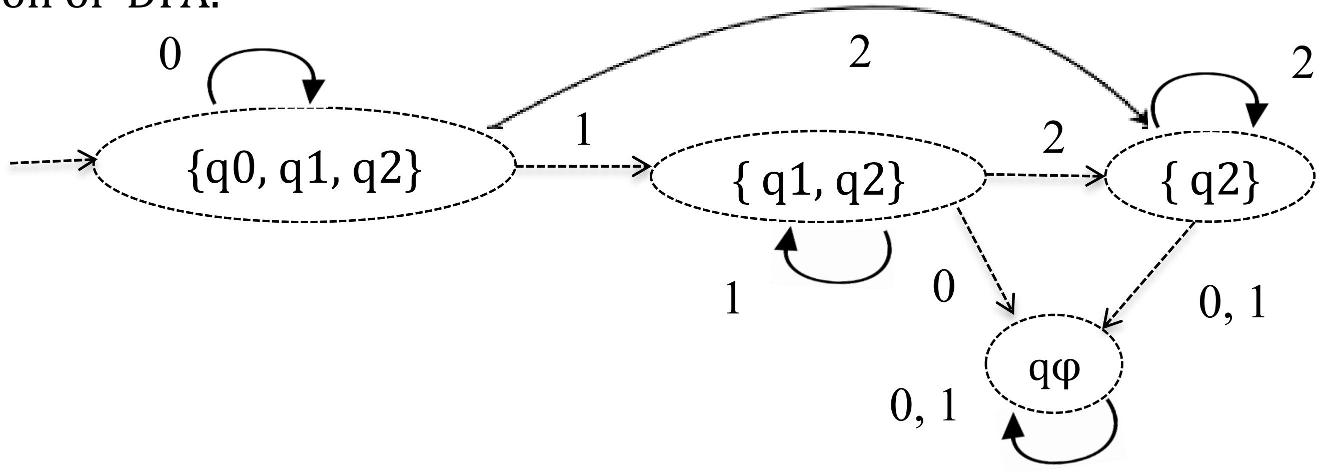


Now remove unnecessary ε moves .

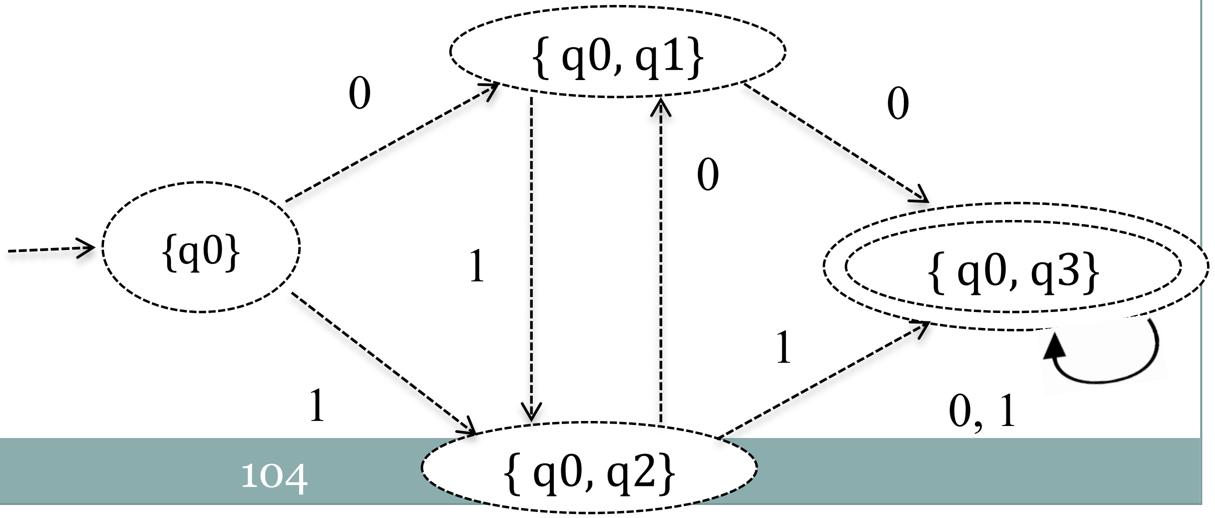


Construction of NFA & DFA Example

Solution: Construction of DFA.



The State {q0, q1, q3 } and {q0, q2, q3 } are not repeatable, as they are having common outgoing behavior.



Deference Between NFA & DFA

DFA	NFA
DFA is Deterministic Finite Automata.	NFA is Nondeterministic Finite Automata.
For each symbolic representation of the alphabet, there is only one state transition in DFA.	For each symbolic representation of the alphabet, there are multiple state transition in NFA.
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.
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.
It is easy to determine weather w ∈ L as transitions are deterministic.	It is difficult to determine weather w ∈ L as transitions are non deterministic.

Finite automata as Output Device (Moore / Mealy Machine)

- Finite automata we have discussed so far had limited functionalities.
- It was able to either accept or reject the string based on reachability of machine from start to final state.
- Finite automata can also be used as an output device.
  - Such machines don't have final states.
  - Such machines can generate output on every input.
  - The value of output is a function of current state and current input.
  - This machine can be characterized by two behaviors.

1. State Transition Function (STF): denoted by **δ**

   $$\delta: \Sigma \times Q \rightarrow Q$$

2. Output Function / Machine Function (MAF): denoted by **λ**

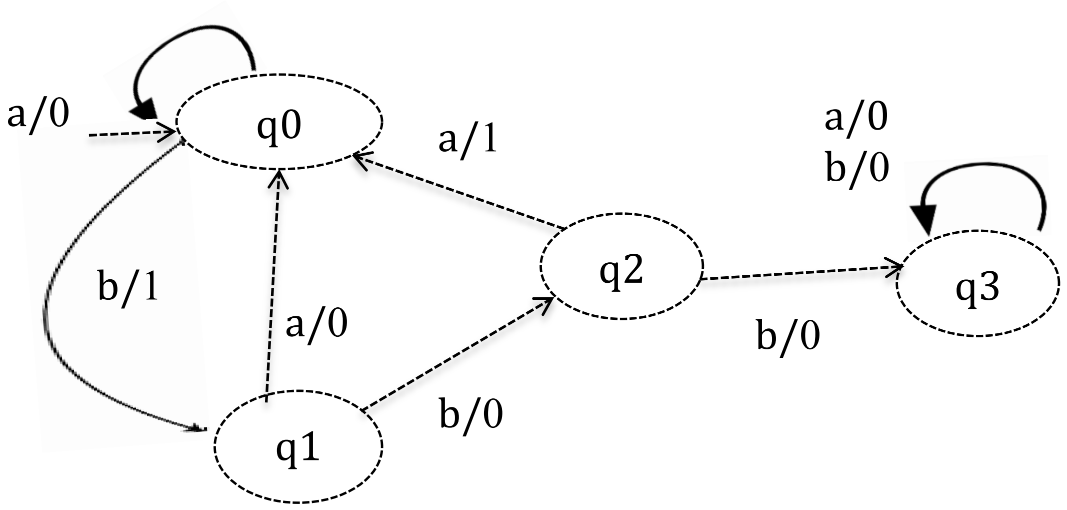
   $$\lambda: \Sigma \times Q \rightarrow O \,/\, \lambda: Q \rightarrow O$$

Formal Definition of Mealy Machine

➢ A Mealy machine is a finite-state machine whose output values are determined by both its current state and the current inputs.

➢A Mealy machine can be represented by a 6-tuple M=(Q, ∑, O, δ, λ, q_0) where –

1. **Q** is a finite set of states.
2. **∑** is a finite set of input symbols called the alphabet.
3. **O** is a finite set of output symbols.
4. **δ** is the transition function where δ: Q × ∑ → Q
5. **λ** is the output function where **λ** : Q × ∑ → O
6. **q_0** is the initial state from where any input is processed (q_0 ∈ Q).



State transition Diagram For Mealy machine

Mealy Machine

State table for both transition function (δ) and output function (λ) can be shown as

Input Symbols	a	b
q0	q0	q1
q1	q0	q2
q2	q0	q3
q3	q3	q3

State transition function
For Mealy machine

States	a	b
q0	0	0
q1	0	0
q2	1	0
q3	0	0

output function for
Mealy machine

Input Symbols	a	b
q0	q0/0	q1/0
q1	q0/0	q2/0
q2	q0/1	q3/0
q3	q3/0	q3/0

Here in q1 / 0
q1 is next state on transition.
0 is a output on applying input.

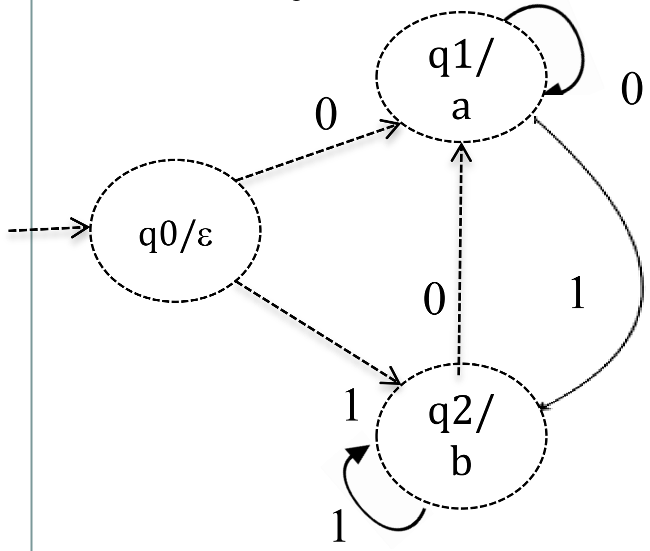
State transition and output function combined For Mealy machine

Formal Definition of Moore Machine

➢ A Moore machine is a finite-state machine whose output values are determined solely by its current state only.

➢A Mealy machine can be represented by a 5-tuple M=(Q, ∑, O, δ, λ, q_0) where –

1. **Q** is a finite set of states.
2. **∑** is a finite set of input symbols called the alphabet.
3. **O** is a finite set of output symbols.
4. **δ** is the transition function where δ: Q × ∑ → Q
5. **λ** is the output function where **λ** : Q → O
6. **q_0** is the initial state from where any input is processed (q_0 ∈ Q).



Input Symbols	0	1
q0	q1	q2
q1	q1	q2
q2	q1	q2

State transition function
For Moore machine

State	output
q0	ε
q1	a
q2	b

output function for
Moore machine

State transition Diagram

Moore Machine

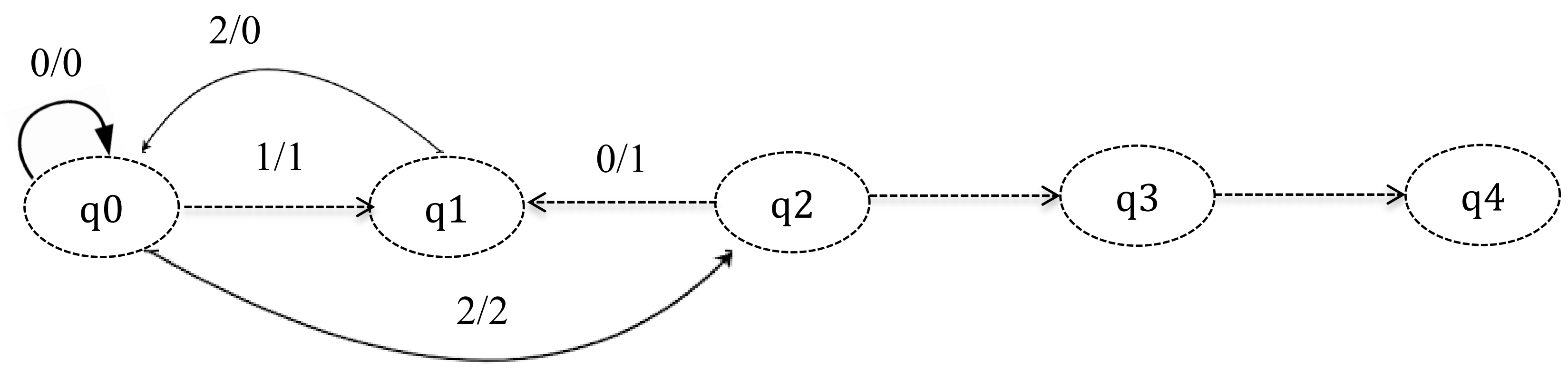
State table for both transition function (δ) and output function (λ) can be shown as

Input Symbols	0	1	output
q0	q1	q2	ε
q1	q1	q2	a
q2	q1	q2	b

- Output in Moore machine is associated with a state and not with transition.
- In more machine length of output string will be equivalent to length of input string plus one.
- i.e. if length of input string is 5 so length of output string will be 5+1=6.

Moore / Mealy Machine Examples

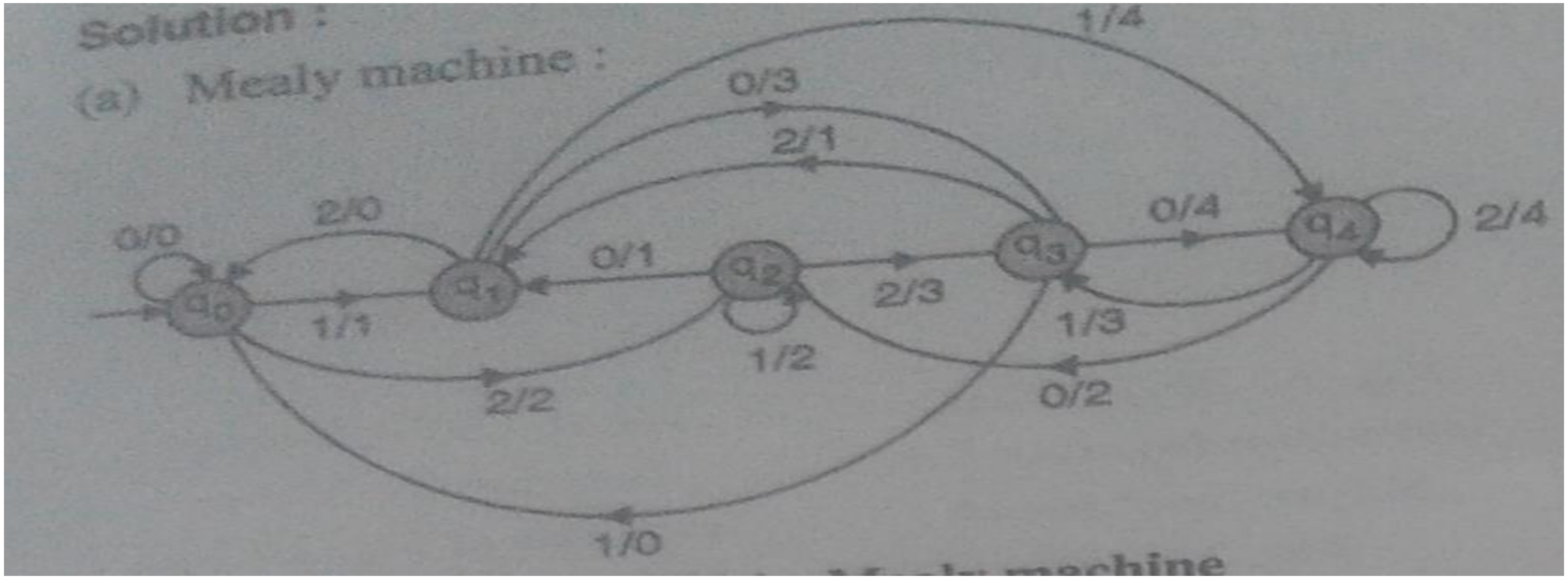
Construct moore & mealy machine for the input {0 ,1 ,2} print the residue module 5 Of the input treated as ternary (Base 3).



Output in Moore machine is associated with a state and not with transition.

Moore / Mealy Machine Example-1

Ex 1: Construct Moore & mealy machine for the input {0 ,1 ,2} print the residue module 5 Of the input treated as ternary (Base 3).

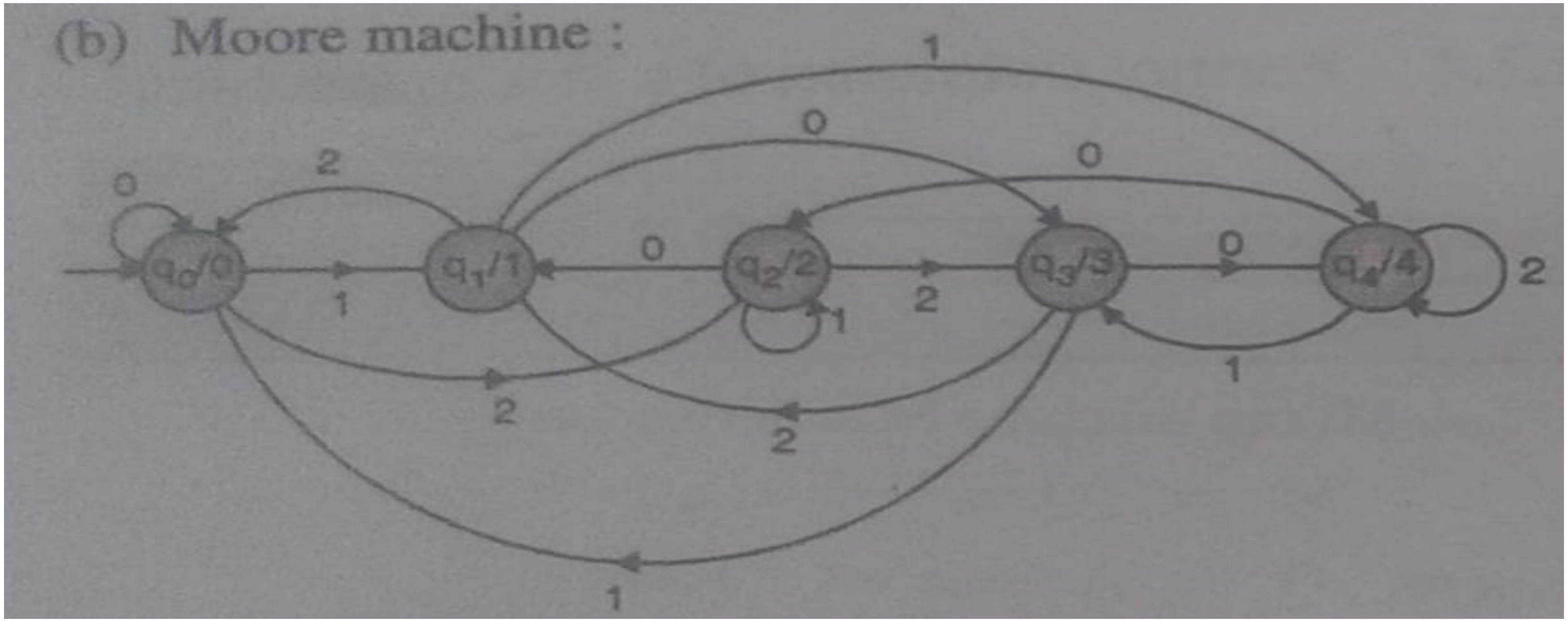


Meaning of various states is given bellow:
q0 is Running reminder is 0.
q1 is Running reminder is 1.
q2 is Running reminder is 2.
q3 is Running reminder is 3 =(10)3.
q4 is Running reminder is 4=(11)3.

Mealy Machine is shown in above diagram

Moore / Mealy Machine Examples

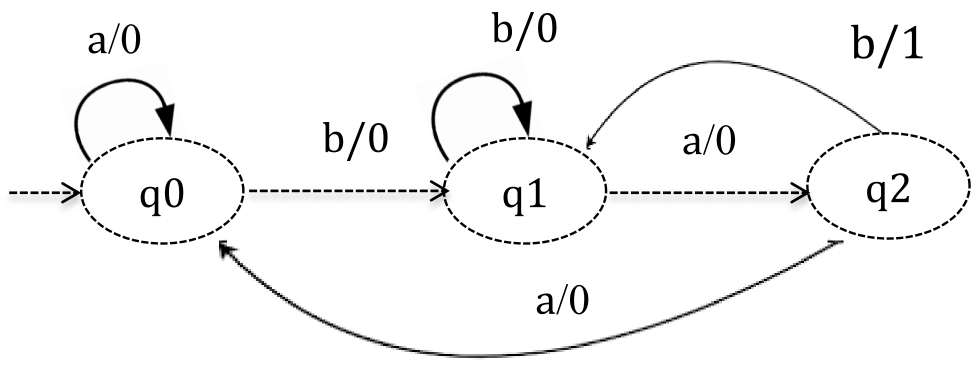
Moore Machine:



Moore Machine is shown in above diagram

Moore / Mealy Machine Examples-2

Ex 2: Construct mealy machine that gives output of 1 if the input string ends in "bab".



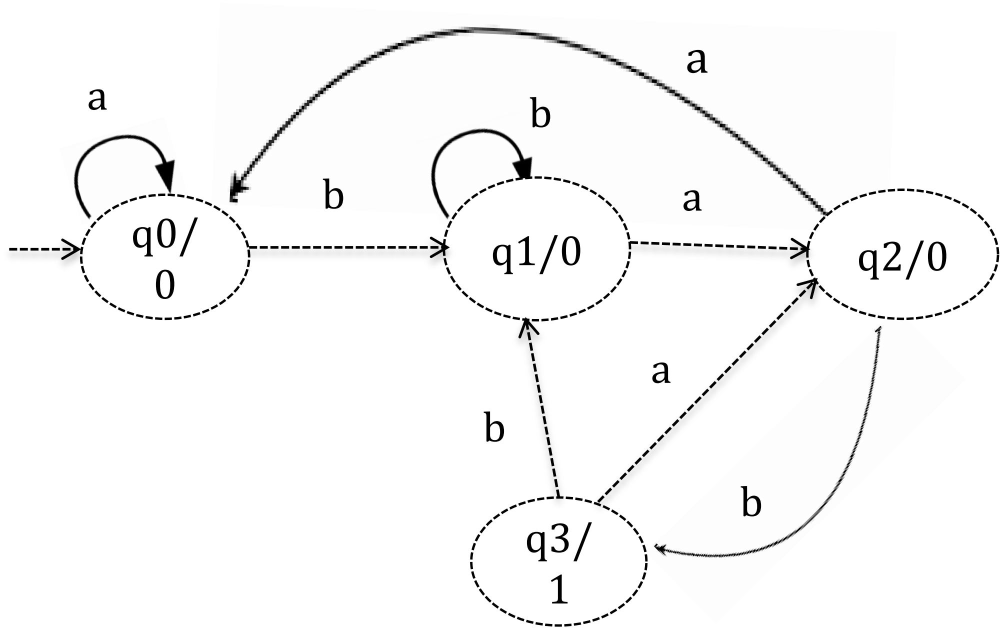
Meaning of various states.

- q0 Starting state.
- q1 Make the preceding character b.
- q2 generate the preceding character ba
- q2 to q1transition will generate the preceding character bab so output will be 1.

Input Symbols	a	b
q0	q0/0	q1/0
q1	q2/0	q1/0
q2	q0/0	q1/1

Moore / Mealy Machine Examples-3

Ex 3: Construct Moore machine that gives output of 1 if the input strings ends in "bab".



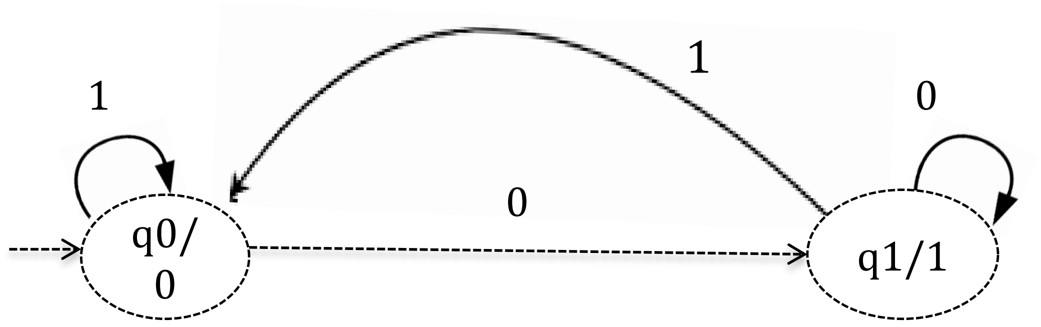
Meaning of various states.

- q0 Starting state.
- q1 Make the preceding character a.
- q2 generate the preceding character ba
- q2 to q3 transition will generate the preceding character bab so output will be 1.

Input Symbols	a	b	Output
q0	q0	q1	0
q1	q2	q1	0
q2	q0	q3	0
q3	q2	q1	1

Moore / Mealy Machine Examples-4

Ex 4: Construct Moore machine that gives output of 1's complement of binary number.



Meaning of various states.

- q0 Starting state.
- q0 moves the machine to q1 on input 0 & output 1. O's complement is 1.
- q1 moves the machine to q0 on input 1 & output 0. 1's complement is 0.

Input Symbols	0	1	Output
q0	q1	q0	0
q1	q1	q0	1

Moore / Mealy Machine Examples-5

Ex 5: Construct Mealy machine that gives output of 2’s complement of binary number.

$$\begin{tikzcd}
\arrow[r, dashed] & q0 \arrow[loop above, "0/0"] \arrow[rr, dashed, "1/1"] & & q1 \arrow[loop above, "\begin{matrix}0/1\\1/0\end{matrix}"]
\end{tikzcd}$$

Meaning of various states.

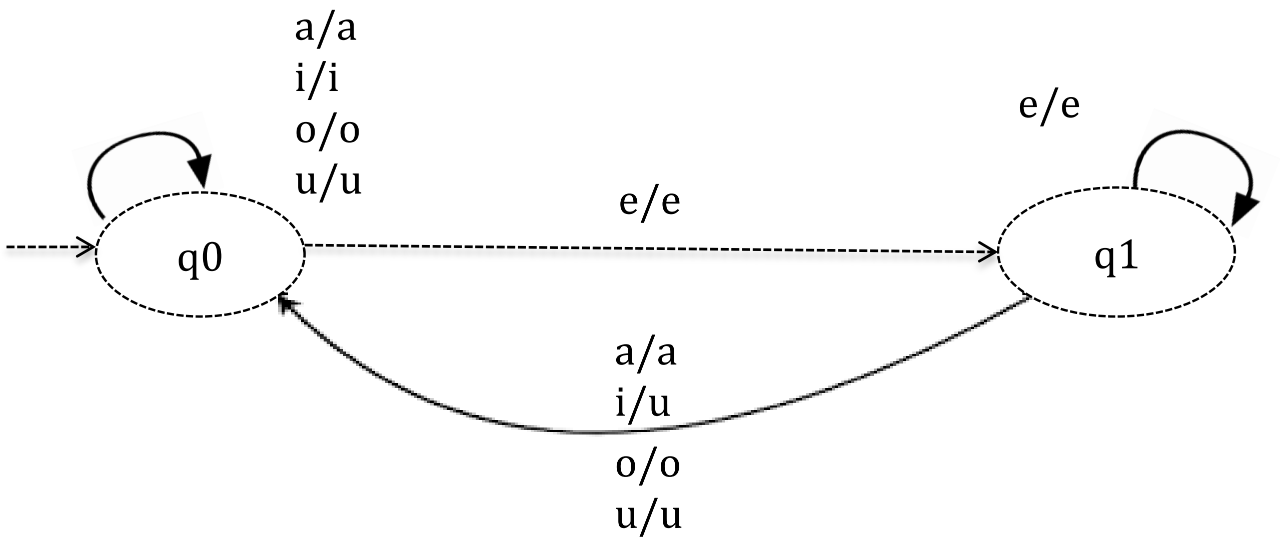
- 2’s Complement of binary number can be found by not changing the bits from right side till 1st ‘1’ and then complementing remaining bits.
- Ex. Consider Binary number 010110 1000

Input Symbols	0	1
q0	q0 / 0	q0/ 0
q1	q1 /1	q1 /1

$\underbrace{010110}_{\text{Complement Every bit}} \quad \underbrace{1000}_{\text{No Change}} \quad = 1010011000$

Moore / Mealy Machine Examples-6

Ex 6: Design Mealy machine that will read the sentence made up of vowels of English language it will output in same sequences except in cases where i follows e it will be converted to u.



Meaning of various states.

- Machine must convert every occurrence of ei into eu. In other cases output will be same as input.
- State q1 implies that thee preceding character is e.
- An Input I in state q1 is converted to u.

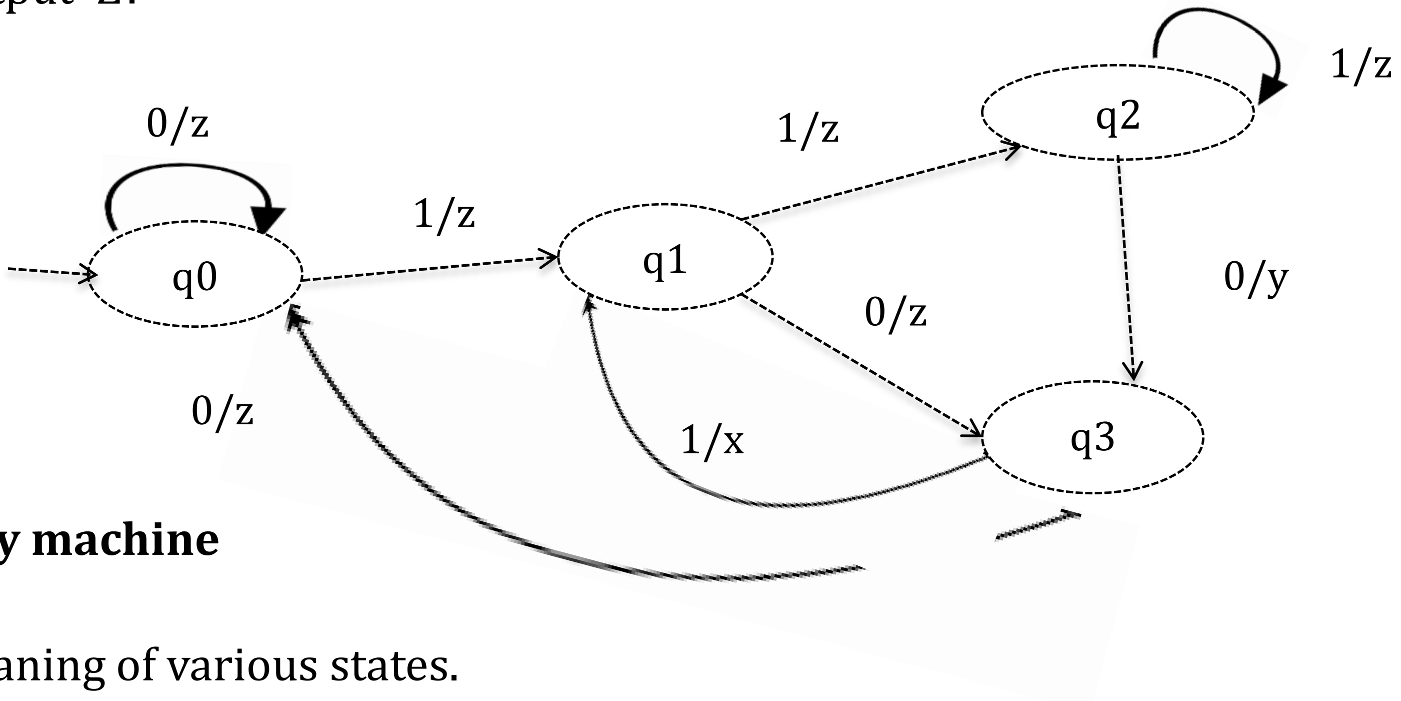
Moore / Mealy Machine Examples-6

State Transition & Output Function can be shown as bellow.

Input Symbols	Next State/ Output				
	a	e	i	o	u
q0	q0	q1	q0	q0	q0
	a	e	i	o	u
q1	q0	q1	q0	q0	q0
	a	e	u	o	u

Moore / Mealy Machine Example-7

Ex 7: Design Mealy and Moore machine for the following processes. For input from (0+1)* if input end in 101 output 'X' and if input end in 110 output 'Y' , otherwise output 'Z'.

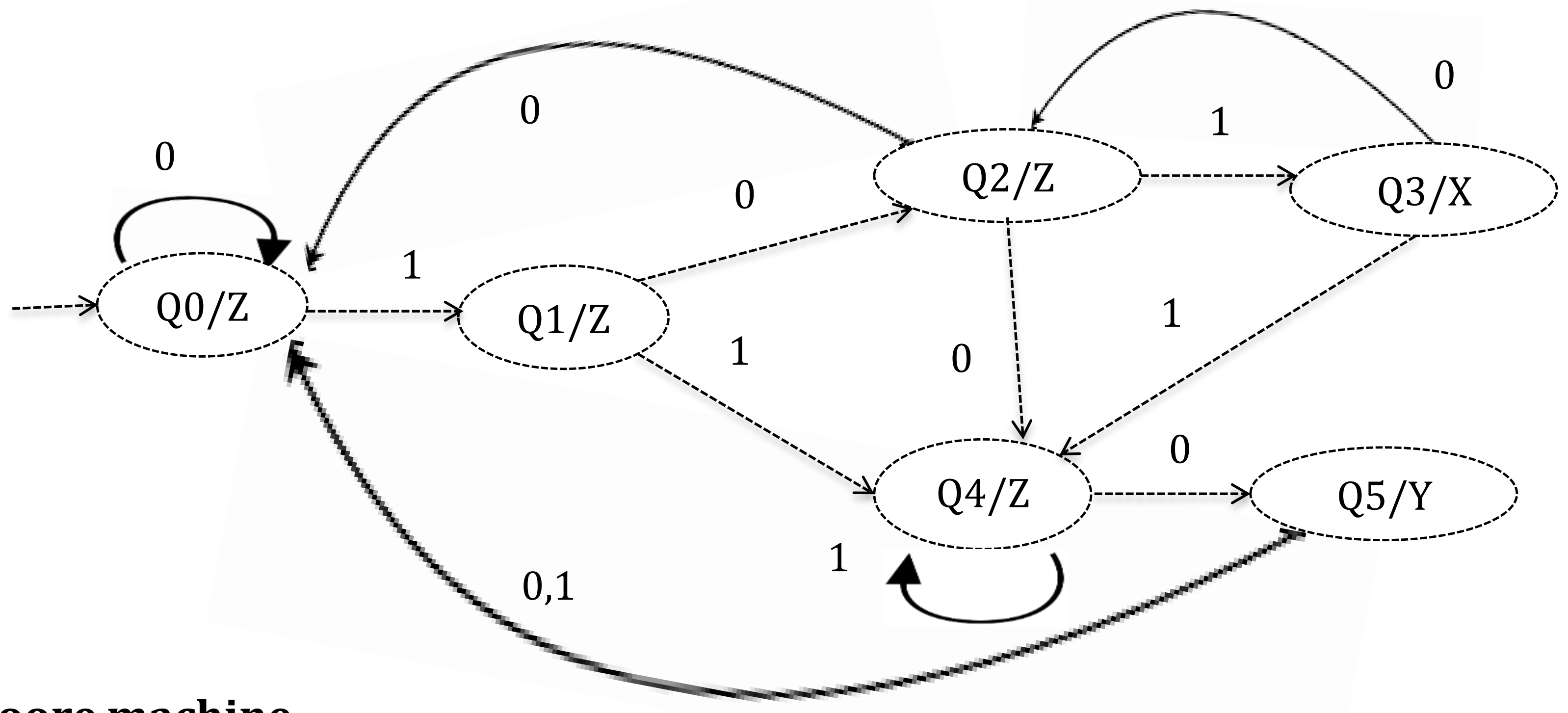


Mealy machine

Meaning of various states.

- Machine must produce output X on input 101 so q0 to q1, q1 to q3 and q3 to q1 producing string 101 which will produce the output X.
- Machine must produce output Y on input 110 so q0 to q1, q1 to q2 and q2 to q3 producing string 110 which will produce the output Y.
- All other transitions will produce output Z.

Moore / Mealy Machine Example-7



Moore machine

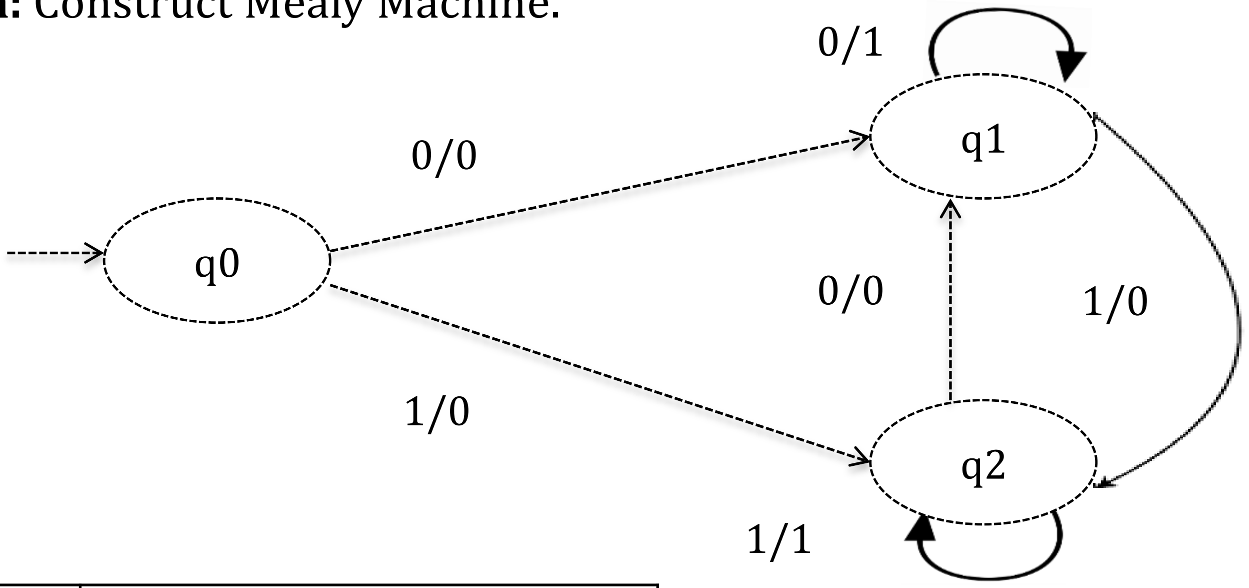
Meaning of various states.

- Machine must produce output X on input 101 so q0 to q1, q1 to q2 and q2 to q3 producing string 101 which will produce the output X.
- Machine must produce output Y on input 110 so q0 to q1, q1 to q4 and q4 to q5 producing string 110 which will produce the output Y.
- All other transitions will produce output Z.

Conversion of Mealy Machine to Moore Machine Example-1

Construct a mealy machine that accept string ending in 00 and 11. Convert the same to moore machine.

Solution: Construct Mealy Machine.



Input Symbols	Next State /Output	
	0	1
q0	q1 /0	q2/ 0
q1	q1 /1	q2/ 0
q2	q1 /0	q2/ 1

Required Mealy Machine, State transition and output function as shown here.

Conversion of Mealy Machine to Moore Machine Example-1

Meaning of various states.
q0- is a initial state.
q1- has preceding input 0 from q0 next input 0 will complete the sequence 00.
q2- has preceding input 1 from q0 next input 1 will complete the sequence 11.

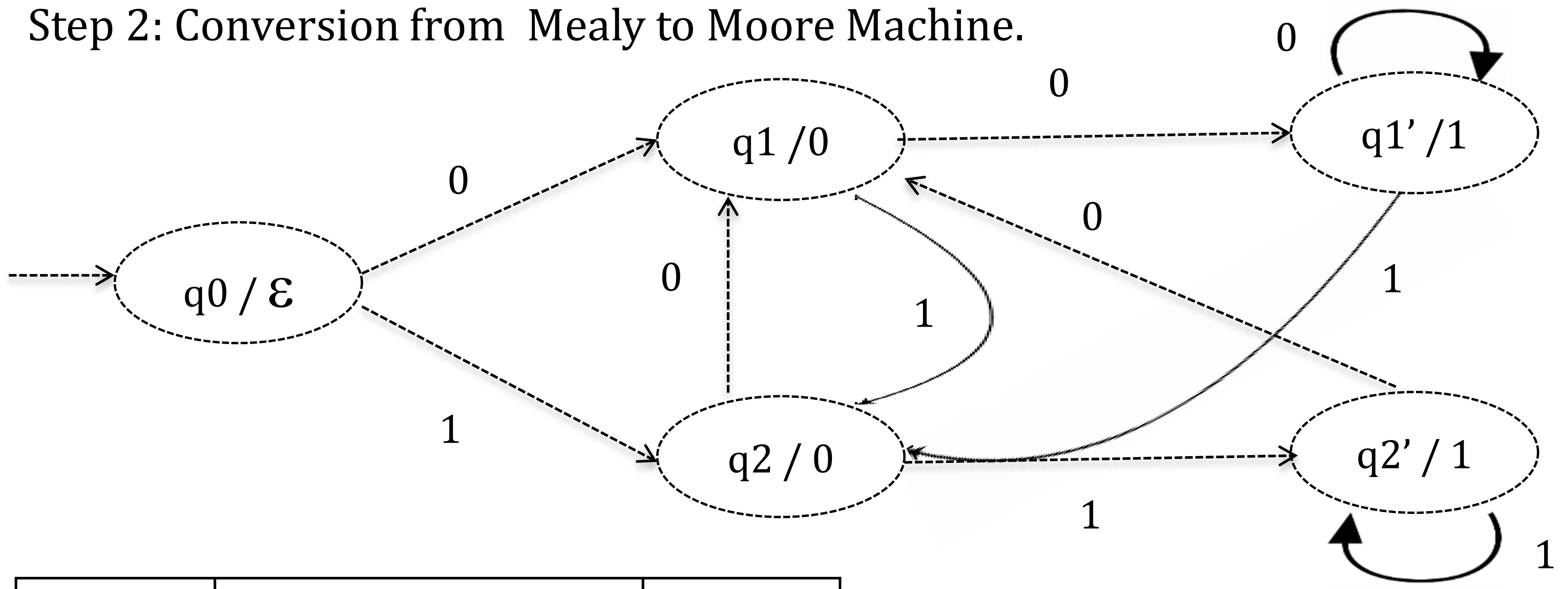
Conversion from Mealy to Moore Machine:
Step 1: Splitting of the states

- There is no incoming line associated with q0 so output associated with q0 will be ε.
- There are three incoming lines to state q1
  - q0 to q1 on input 0 and output 0.
  - q1 to q1 on input 0 and output 1.
  - q2 to q1 on input 0 and output 0.
- Since there are two different output we will split the state q1 in two sub states namely q1 for output 0 and q1' for output 1.
- Similarly split q2 into q2 for output 0 and q2' for output 1.

- Step 2: Draw equivalent Moore machine after splitting.

Conversion of Mealy Machine to Moore Machine Example-1

Step 2: Conversion from Mealy to Moore Machine.



Input Symbols	Next State		Output
	0	1	
q0	q1	q2	ε
q1	q1'	q2	0
q1'	q1'	q2	1
q2	q1	q2'	0
q2'	q1	q2'	1

- Required More Machine, State transition and output function as shown here.
- If mealy machine having m no of states and n output symbols then possible no of states in resultant Moore machine will be mn.

Conversion of Mealy Machine to Moore Machine Example-2

Ex 2: Convert the given mealy machine to equivalent more machine.

Input Symbols	Next State /Output	
	0	1
→ q1	q1 /1	q2/ 0
q2	q4 /1	q4/ 1
q3	q2 /1	q3/ 1
q4	q3 /0	q1 /1

Solution:

Step 1: Splitting of the states: Split the states if the incoming arcs are having different output.

Split the state q1 into q1 with output 0 and q1' with output 1.

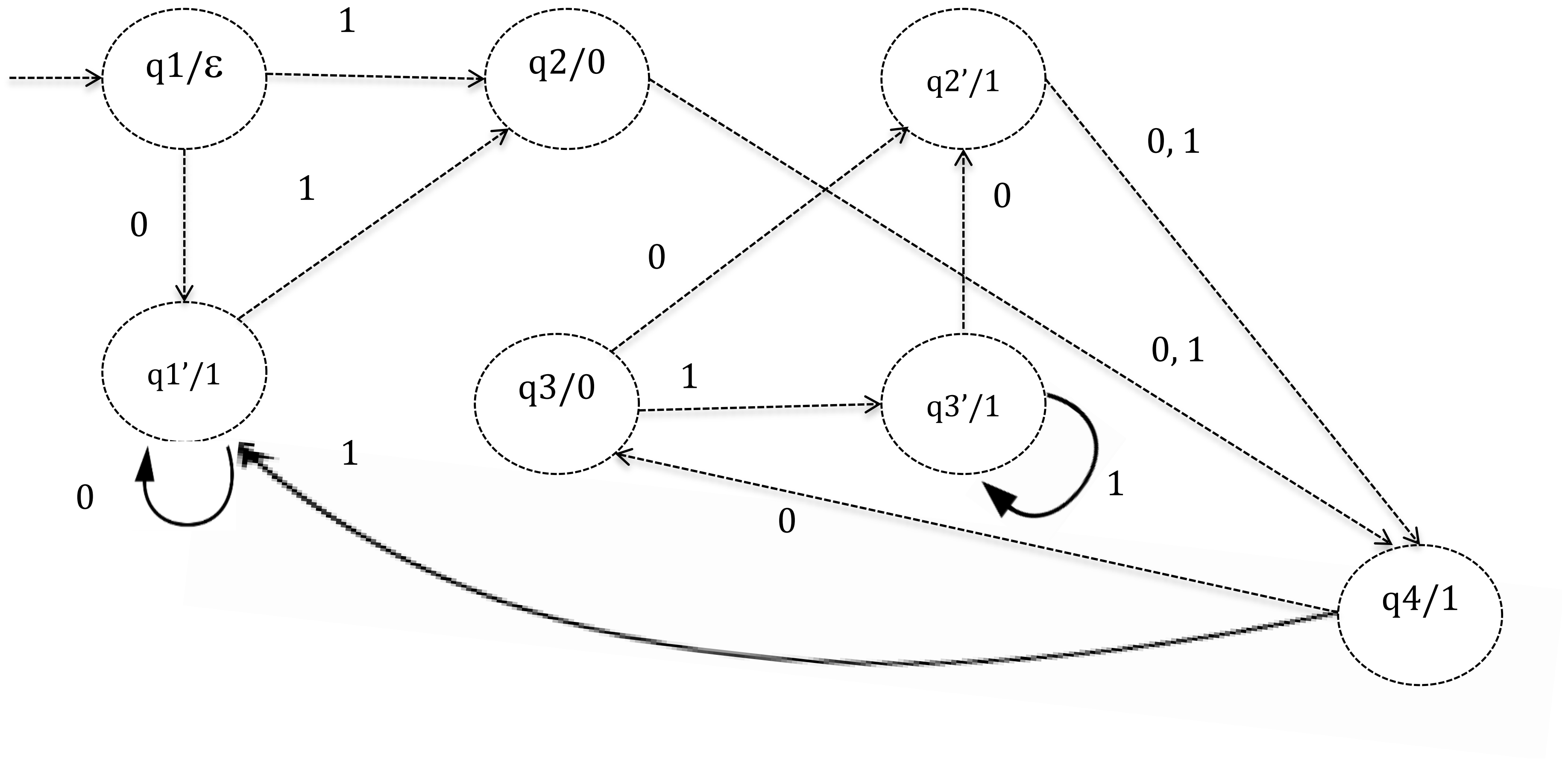
Split the state q2 into q2 with output 0 and q2' with output 1.

Split the state q3 into q3 with output 0 and q3' with output 1.

As all incoming edge to q4 are producing same output so no need to split it will always produce the output 1.

Conversion of Mealy Machine to Moore Machine Example-2

Step 2: Convert to equivalent Moore machine.



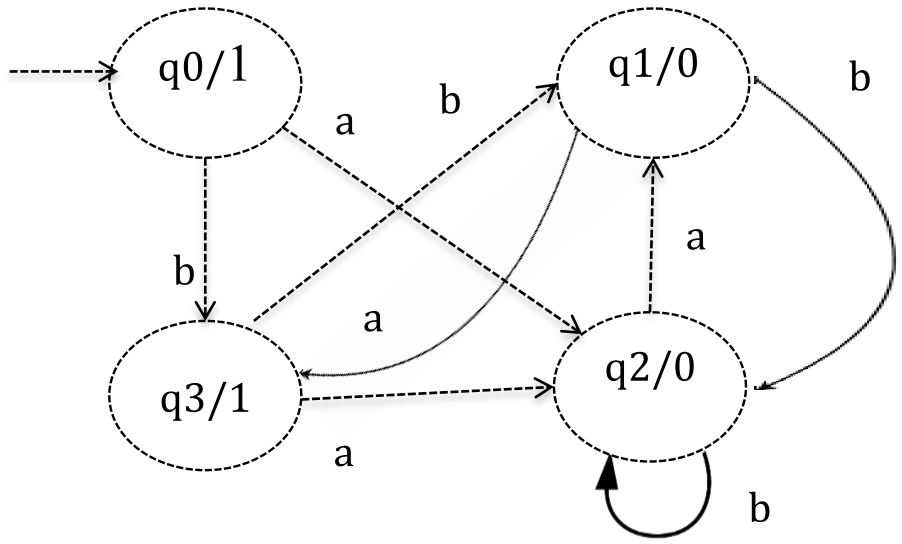
Conversion of Mealy Machine to Moore Machine Example-2

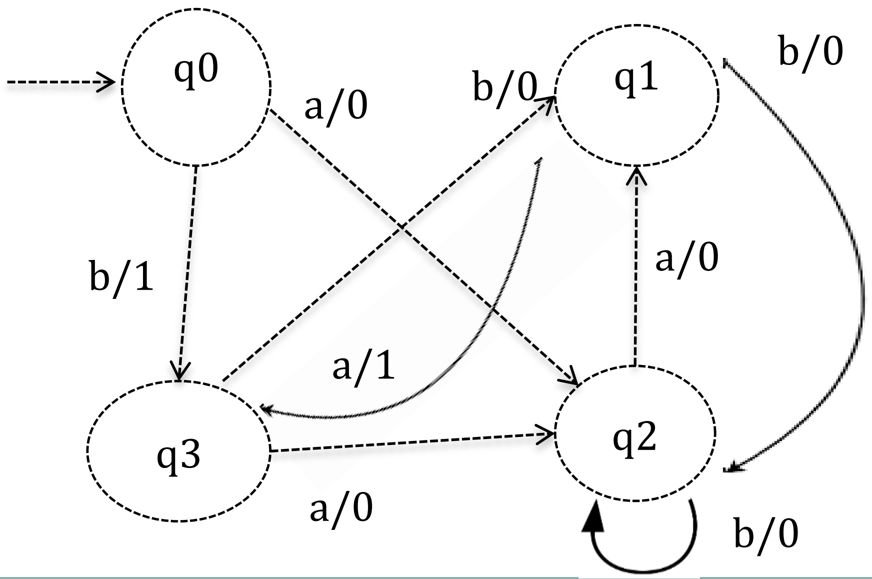
Step 2: Construct state transition and output function for Moore machine.

Input Symbols	Next State		Output
	0	1	
q1	q1’	q2	ε
q1’	q1’	q2	1
q2	q4	q4	0
q2’	q4	q4	1
q3	q2’	q3’	0
q3’	q2’	q3’	1
q4	q3	q1’	1

Conversion of Moore Machine to Mealy Machine Example-1

Ex1: Convert the given Moore machine in to mealy machine





Solution: Construction of trivial mealy machine by moving output associated with a state to transitions entering that state, we get.

Conversion of Moore Machine to Mealy Machine Example-1

Construct state transition and output function for mealy machine.

Input Symbols	Next State /Output	
	0	1
q0	q2 /0	q3/ 1
q1	q3/ 1	q2 /0
q2	q1 /0	q2 /0
q3	q2 /0	q1 /0

Step 2: Minimization of mealy machine

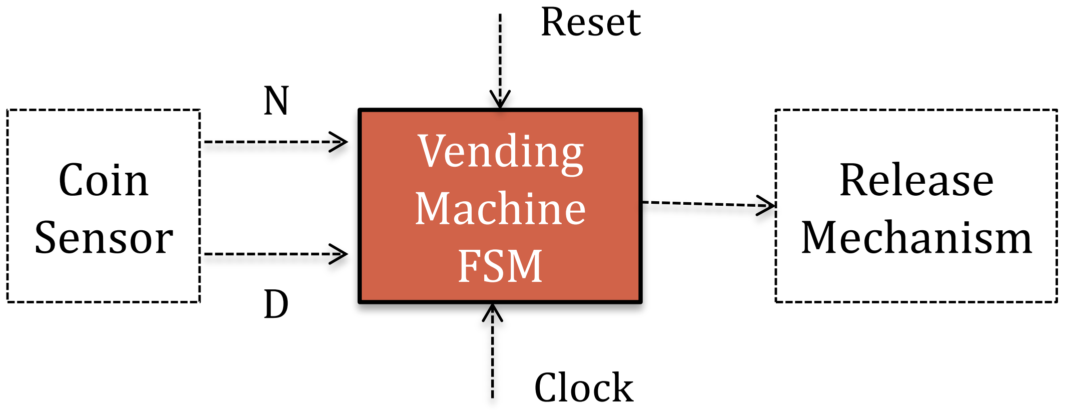
- 1 equivalent participations p1 based on output

  P1= (q0), (q1), (q2, q3)
- 2 equivalent participations p2 based on transition:
  - q2 is mapped with block (q1) on input 0.
  - q3 is mapped with block (q2, q3) on input 0.

  P2= (q0), (q1), (q2) , (q3)

FSM for Vending Machine

- A typical vending machine look like as following.



- A vending machine can be designed using FSM model with auto billing. It reduces the hardware. The typical no of steps included in FSM vending machines are:
  1. Selection of product by User.
  2. waiting for Insertion of Coin.
  3. Product delivery and service.
- Entire things can be modeled using mealy machine.
- Machine can count the value of inserted coin and take the decision regarding product release.
- FSM can also be used for spelling checking.

Mealy Machine Vs Moore Machine.

The following table highlights the points that differentiate a Mealy Machine from a Moore Machine.

Mealy Machine	Moore Machine
Output depends both upon the present state and the present input	Output depends only upon the present state.
Generally, it has fewer states than Moore Machine.	Generally, it has more states than Mealy Machine.
The value of the output function is a function of the transitions and the changes, when the input logic on the present state is done.	The value of the output function is a function of the current state and the changes at the clock edges, whenever state changes occur.
Mealy machines react faster to inputs. They generally react in the same clock cycle.	In Moore machines, more logic is required to decode the outputs resulting in more circuit delays. They generally react one clock cycle later.

Thank You