UCS 701 TOC

Finite Automata and Formal Language

Dr. Gourav Jain
AP, CSED,TIET

INTRODUCTION
TO
FINITE AUTOMATA

Automata theory (also known as Theory Of Computation) is a theoretical branch of Computer Science and Mathematics, **which mainly deals with the logic of computation with respect to simple machines**, referred to as automata.

Automata* enables scientists to **understand how machines compute the functions and solve problems.**

The main motivation behind developing Automata Theory was to develop methods to **describe and analyze the dynamic behavior of discrete systems**.

- What is Automata?
- Automata is an abstract machine for modeling computations.

- Why abstract machines?
- Used to model the essential parameters and ignore non essential parameter.

- What is computability?
- Computability is the ability to solve a problem in an effective manner.

Applications of Theory of computation

The theory of computation is applied in the following −

Traffic lights.
Lifts and elevators.
Cloud computing
Design of Digital Circuit
Compiler Construction
String Matching
String processing
Software Design
Other Applications

Symbol:

A symbol (often also called a character) is the smallest building block, which can be any alphabet, letter, or picture. i.e., a, b, c, 0, 1

Alphabets (Σ):

An alphabet is a finite and non empty set of symbols. Denoted by ∑

The example of alphabet is,

∑ ={a, b, c,……..z} The elements a, b, c, …….z are alphabets.

If ∑ ={0,1} Here 0 and 1 are alphabets.

Power of an Alphabet

- $\sum^k$ is the set of strings of length k, each of whose symbol is in $\sum$.

- $\sum^0 = \{\epsilon\}$ regardless of what symbol is in $\sum$. ϵ is only string whose length is zero.

- if $\sum = \{0,1\}$ then $\sum^1 = \{0,1\}, \sum^2 = \{00,01,10,11\}$

- $\sum^*$ = set of all string from $\sum$ including empty string (ϵ)
  $= \sum^0 \cup \sum^1 \cup \sum^2 \cup \sum^3 \cup \sum^4$ .....
- $\sum^+$ = set of all string from $\sum$ excluding empty string
  $= \sum^1 \cup \sum^2 \cup \sum^3 \cup \sum^4$ .....

$\sum^* = \sum^+ \cup \{\epsilon\}$

- If $\sum = \{0,1\}$ then
  $\sum^* = \{\epsilon , 0,1,00,01,10,11,000,001,010,011,100,101,.....\}$
  $\sum^+ = \{0,1,00,01,10,11,000,001,010,011,100,101,.....\}$

String:

A string is a **finite sequence of symbols** from some alphabet. A string is generally denoted as w and the length of a string is denoted as |w|.

For example, if ∑ = {a,b} then various strings that can be formed from ∑ are {ab, ba, aaa, bbbb, aba, bab, ….}. An infinite number of strings can be generated.

- Empty string is the string with zero occurrence of symbols, represented as ε.

Language : -

A language is a set of strings, chosen from some Σ* or we can say- '**A language is a subset of Σ* '.** A language that can be formed over ' Σ ' can be Finite or Infinite.

Example of Finite Language:

L1 = { set of string of 2 }

L1 = { xy, yx, xx, yy }

Example of Infinite Language:

L1 = { set of all strings starts with 'b' }

L1 = { babb, baa, ba, bbb, baab, ... }

Language

- Language is the set of string chosen from some $\sum^*$.
- L is language over $\sum$. Then $L \subseteq \sum^*$
- The language is defined using an input set. The input set is typically denoted by letter $\sum$.
- For example: $\sum^*$ ={Ɛ,0,00,000,0000,....}. Here the language L is defined as 'Any number of Zeros'.
- ϕ denotes empty language over any alphabet.
- {ϵ} is the language having only empty string over any alphabet
- $\phi \neq \epsilon$. No string ≠ one string
- $\sum$ = {a,b,c...z, -} set of strings= {ab,ba,ab ab} L = {a,b,ab,ba...}

Examples

- The set of all strings over {a,b,c} that have “ac” as substring can be
  - Input alphabet ∑ = {a,b,c}
  - Set of String = {ac, aac, aca, bac, acb, cac, acc, aaca, aacb, aacc, baca, bacb, back, caca, cacb, cacc, aaac, abac, acac, baac, bbac, bcac, caac, cbac, ccac, acaa, acab, acba, acbb, acbc, acca, accb, accc,…..}
  - Language L = {xacy | x,y ∈{a,b,c}∗} $\sum^*$

- The set of all strings over {0,1} that start with 0’s can be
  - Input alphabet ∑ = {0,1}
  - Set of String = {0, 00, 01, 001, 010, 000, 011, 0000, 0111, 0101, 0010……}
  - Language L = {0x | x ∈{0,1}*}

Examples

- The set of all strings over some alphabet Σ which start with 0 and end with 1
  - Input alphabet ∑ = {0,1}
  - String = {01,001,011,0001,0011,0101,0110,……}
  - Language L = {0x1| x∈{0,1}∗}

- The set of all strings over some alphabet Σ with equal number of a's and b's
  - Input alphabet ∑ = {a,b}
  - String = {ϵ ,ab,ba,aabb,abab,baba,bbaa,abba,baab……}
  - Language L = {x∈Σ∗ | $|x|_a = |x|_b$}

Examples

- The set of all strings over some alphabet Σ with any number of a's
  - Input alphabet ∑ = {a}
  - String = {∈, a,aa,aaa,aaaa,aaaaa,……}
  - Language L = {a^n| n∈ W} or {x | x ∈Σ∗}

- The set of all strings over some alphabet Σ with even number of a's
  - Input alphabet ∑ = {a}
  - String = {∈,aa,aaaa,aaaaaa,……}
  - Language L = {xx| x∈Σ∗ or {a^{2n}|n ∈ W}

Examples

- The set of all strings over some alphabet Σ in which any number of a's then followed by any number of b's
  - Input alphabet ∑ = {a,b}
  - String = {∈ , a, b,ab, aab, abb, aabb, aaab, abbb, aabbb.....}
  - Language L = $\{a^n b^m \mid n, m \geq 0\}$

- The set of all strings over some alphabet Σ in which any number of a's then followed by equal number of b's
  - Input alphabet ∑ = {a,b}
  - String = {∈,ab, aabb, aaabbb, aaaabbbb.....}
  - Language L = $\{a^n b^n \mid n \geq 0\}$

Examples

- The set of all strings over some alphabet Σ in which n number of a's followed by m number of b's where m = n+1 or b's is one more than a's
  - Input alphabet ∑ = {a,b}
  - String = {b, abb, aabbb, aaabbbb, aaaabbbbb,.....}
  - Language L = $\{a^n b^m \mid n \geq 0 \; and \; m = n + 1\}$ or $\{a^n b^{n+1} \mid n \geq 0\}$

- The set of all strings over some alphabet Σ in which any number of a's then followed by double number of b's
  - Input alphabet ∑ = {a,b}
  - String = {∈,abb, aabbbb, aaabbbbbb, aaaabbbbbbbb.....}
  - Language L = $\{a^n b^{2n} \mid n \geq 0\}$

Exercise

Write input set, strings and language for the following

- The set of all strings with three consecutive 0's over {0,1}
- The set of strings in which any number of a's followed by equal number of b's followed by equal number of c's
- The set of strings in which any number of a's followed by equal number of b's followed by any number of c's
- The set of string which starts with 00 and ends with 11.

- The set of all strings with three consecutive 0's over {0,1}
  - Input alphabet ∑ = {0,1}
  - String = {000,1000,0001,10001,000000, 0001000,00011,11000….}
  - Language L = (0+1)* 000 (0+1)*

- The set of strings in which any number of a's followed by equal number of b's followed by equal number of c's
  - Input alphabet ∑ ={a,b,c}
  - String = {∈, abc, aabbcc, aaabbbccc,……..}
  - Language L = $\{a^n b^n c^n \mid n \geq 0\}$

- The set of strings in which any number of a's followed by equal number of b's followed by any number of c's
  - Input alphabet ∑ = {a,b,c}
  - String = {∈, c, ab, abc, abcc, aabbc , aabbcc.....}
  - Language L = {$a^n b^n c^m$ | $n, m \geq 0$}

- The set of string which starts with 00 and ends with 11.
  - Input alphabet ∑ = {0,1}
  - String = {0011, 00011,00111,000011,000111,001011,001111...}
  - Language L = {00x11 | x ∈Σ∗}

Finite State Machine / Finite Automata

- The finite state machine represents a mathematical model that used to recognize patterns
- It takes string as input, move to next state according to the current input symbol and current state and generate accept or reject as output
- At the end of string if we are at the final state that means string is valid or accepted otherwise invalid or rejected.
- The input is processed by various states.
  - initial state or start state
  - intermediate states
  - final or accept state
- The finite state system is very good design tool for the programs such as TEXT EDITORS and LEXICAL ANALYZERS.

Definition of Finite Automata

- A finite automata is a collection of 5-tuple(Q,∑,δ,q0,F) Where ,
  - Q is finite set of states, which is non empty.
  - ∑ is input alphabet, indicates input set.
  - δ is transition function or mapping function. We can determine the next state using this function.
  - q0 is an initial state and q0 ϵ Q
  - F is set of final states. F ϵ Q

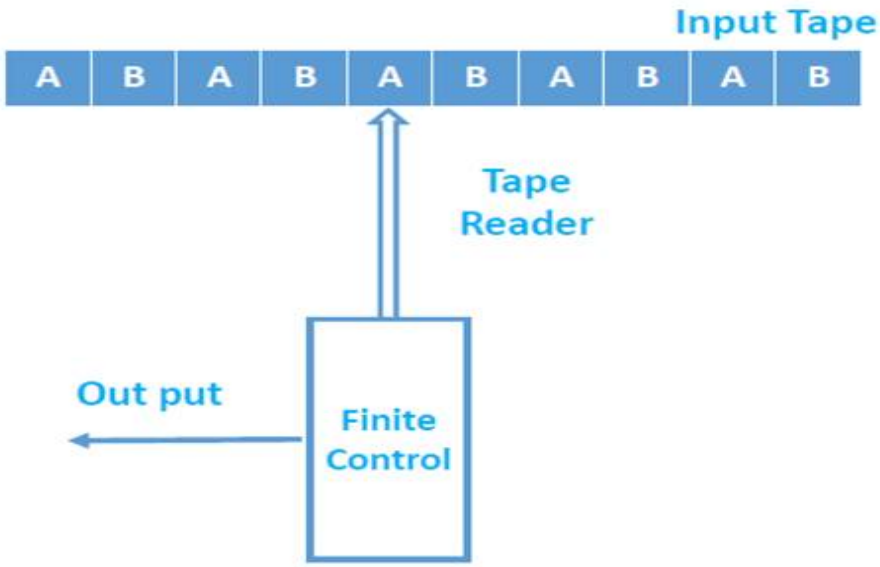
Finite Automata Model

- The finite automata can be represented as

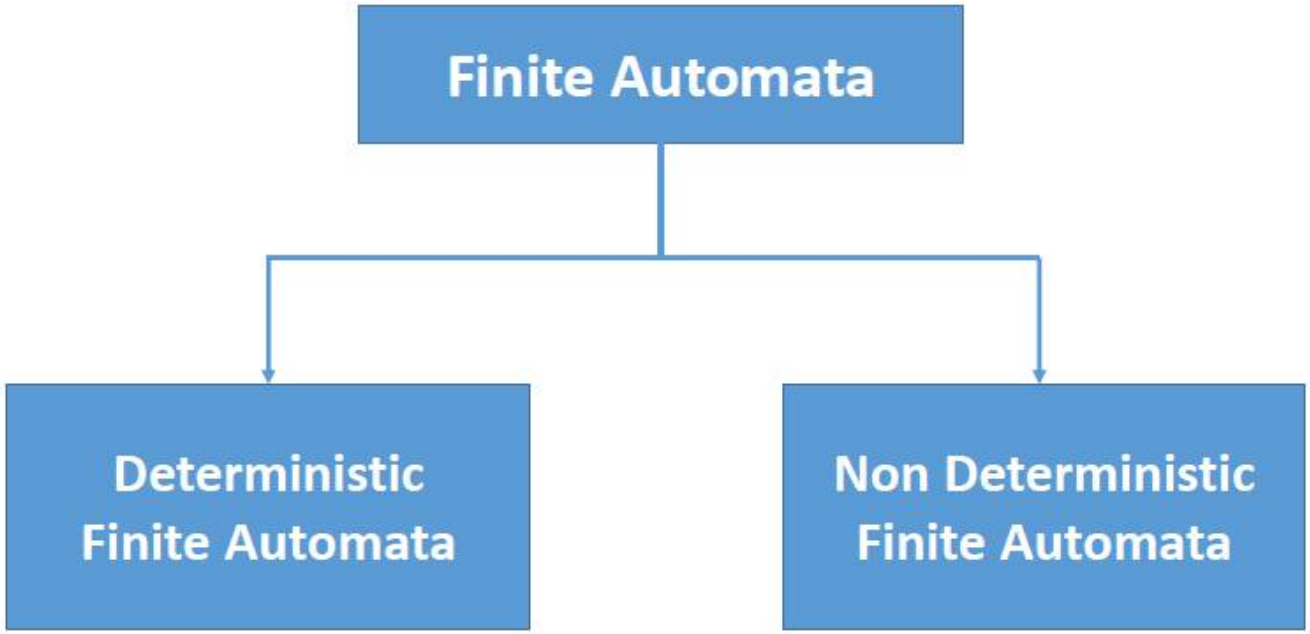
❖ Input Tape – It is a linear tape having some number of cells. Each input symbol is placed in each cell

❖ Finite Control – Finite control decides the next state on receiving particular input from input tape

❖Tape Reader – It reads the cell one by one from left to right, at a time only one input is read



Types of Automata



Deterministic Finite Automata (DFA)

The Finite Automata is called Deterministic Finite Automata if there is only one path for a specific input from current state to next state. It can be represented as follows:

- A machine M = (Q,∑,δ,q0,F) Where ,
  - Q is finite set of states, which is non empty.
  - ∑ is input alphabet, indicates input set.
  - δ is transition function or mapping function. We can determine the next state using this function.

    $\delta : Q \times \sum \rightarrow Q$
  - q0 is an initial state and is in Q
  - F is set of final states.

Non-Deterministic Finite Automata (NFA)

The Finite Automata is called Non Deterministic Finite Automata if there are more than one path for a specific input from current state to next state.

- A machine M = (Q,∑,δ,q0,F) Where ,
  - Q is finite set of states, which is non empty.
  - ∑ is input alphabet, indicates input set.
  - δ is transition function or mapping function. We can determine the next state using this function.

    $\delta : Q \times \sum \rightarrow 2^{Q}$
  - q0 is an initial state and is in Q
  - F is set of final states.

Q = {q0,q1,q2} 2^{Q} = {q0,q1,q2,{q0,q1},{q0,q2},{q1,q2},{q0,q1,q2}}

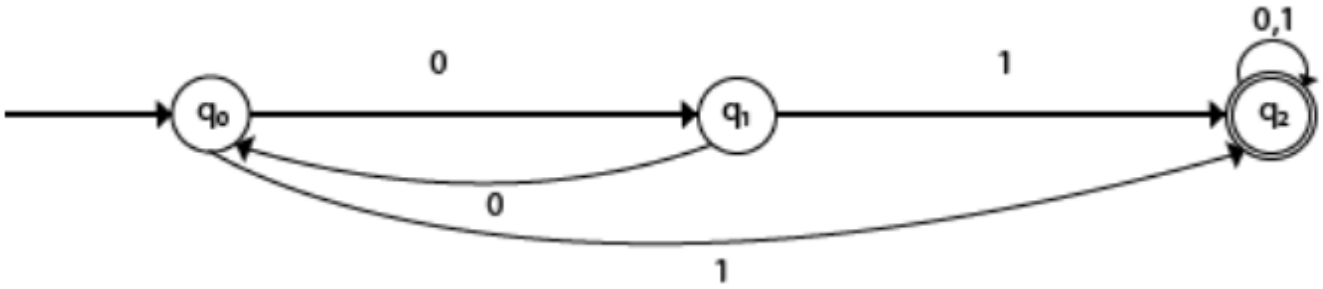
Representation of Machine

- Transition Diagram or State Diagram: A DFA is represented by digraphs called **state diagram** / transition diagram
  - For each state in Q, there is a node.
  - For each state q in Q, input symbol a in ∑. Let δ(q,a) = p then there is a arc from node q to node p labelled by a.
  - The initial state is denoted by an empty single incoming arc.
  - Nodes corresponding to final state denoted by double circle

δ(q0,0) = q1

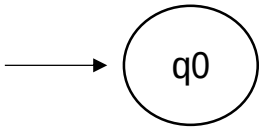
δ(q0,1) = q2

δ(q1,0) = q0

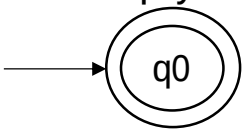


Notation for constructing transition diagram

- Initial state is indicated by single arrow

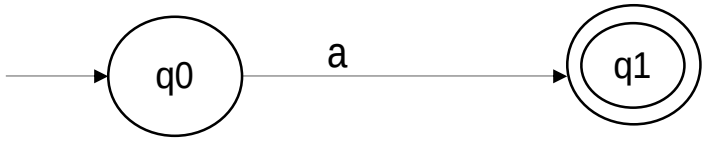


When machine accept null or empty string



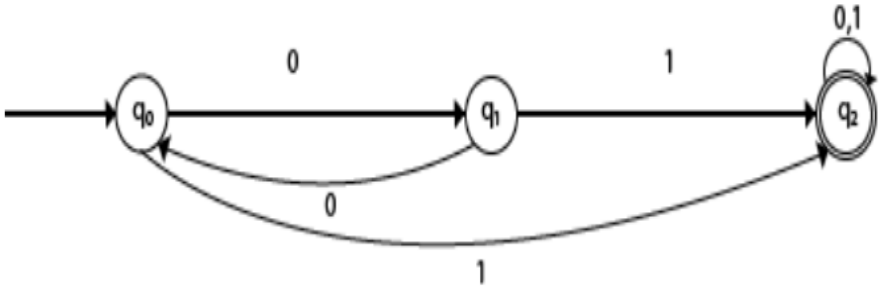
- Final state is indicated inside double circle

- Transition (q0,a) = q1 where q0 is initial state and q1 is final state



Representation of Transition Function

- **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.
  - Entry for the row corresponding to state q and column corresponding to input a is the state of δ(q,a).
  - The start state is denoted by an arrow with no source.
  - The accept state is denoted by a star or single circle

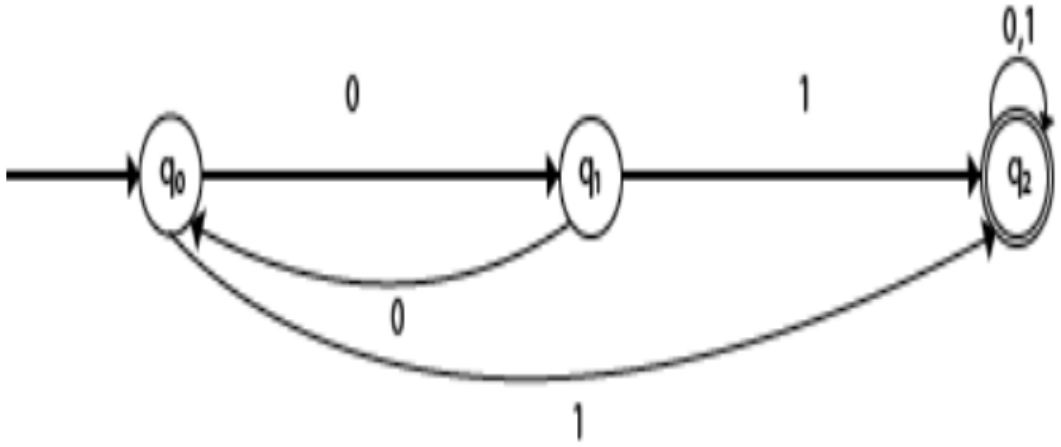


Present State	Next state for Input 0	Next State of Input 1
→q0	q1	q2
q1	q0	q2
*q2	q2	q2

Representation of FA

M = (Q,∑,δ,q0,F) where

- Q = {q0,q1,q2}
- ∑ = {0,1}
- q0 is initial state
- F = {q2}



- δ is define as function, diagram, table:

δ(q0,0) = q1
δ(q0,1) = q2
δ(q1,0) = q0
δ(q1,1) = q2
δ(q2,0) = q2
δ(q2,1) = q2

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

Processing of String using DFA

- To check the validity of any string for the DFA, always start with initial state.

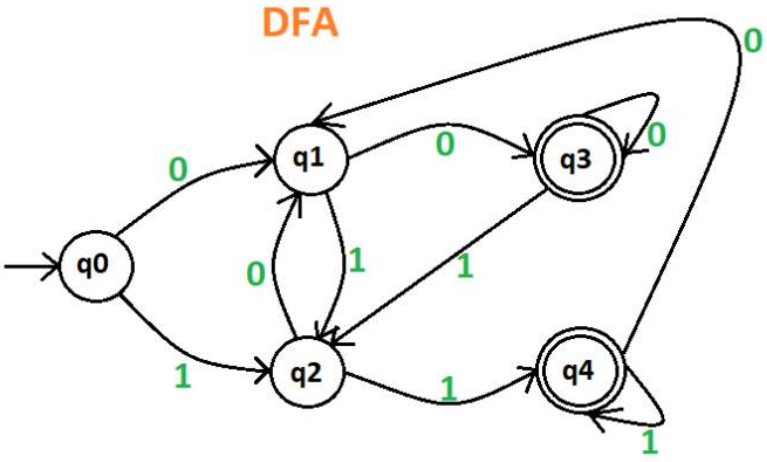
- Pass current alphabet of string to current state and check entry in transition table or transition diagram. Move to next state according to the entry in the transition. Then move with next input symbol of string with new state.

- At the end of string if we reach to the final state that means string is valid or accepted otherwise invalid or rejected.

Check the validity of the string 101100 for the given DFA.

For checking validity of string we have to start from initial state q0.

δ(q0,101100) -> δ(q2,01100) { δ(q0,1)=q2 }
-> δ(q1,1100) { δ(q2,0)=q1 }
-> δ(q2,100) { δ(q1,1)=q2 }
-> δ(q4,00) { δ(q2,1)=q4 }
-> δ(q1,0) { δ(q4,0)=q1 }
-> δ(q3,ϵ) { δ(q1,0)=q3 }



Now we are at the end of string and reach to q3 i.e final state.

At the end of string if we reach to final state that means given string 101100 is valid for the given DFA.

Check the validity of the string aababbba for the given DFA.

For checking validity of string we have to start from initial state q0.

δ(q0,aababbba) -> δ(q1,ababbba) { δ(q0,a)=q1 }
-> δ(q1,babbba) { δ(q1,a)=q1 }
-> δ(q2,abbba) { δ(q1,b)=q2 }
-> δ(q1,bbba) { δ(q2,a)=q1 }
-> δ(q2,bba) { δ(q1,b)=q2 }
-> δ(q3,ba) { δ(q2,b)=q3 }
-> δ(q0,a) { δ(q3,b)=q0 }
-> δ(q1,ϵ) { δ(q0,a)=q1 }

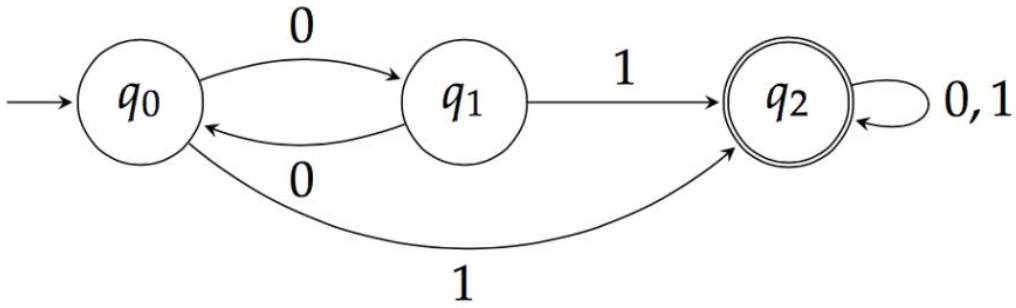
State/Input	a	b
→ q0	q1	q0
q1	q1	q2
q2	q1	q3
*q3	q1	q0

Now we are at the end of string and reach to q1 i.e. non final state.

At the end of string if we reach to non final state that means given string aababbba is not valid for the given DFA.

Exercise

- Check the validity of string 0010101 for given DFA

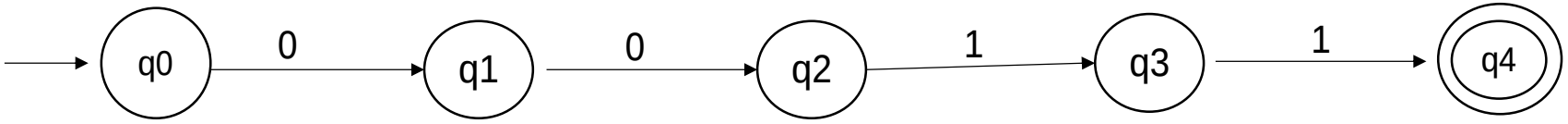


- Check the validity of the string abaabbab for the given DFA

State/Input	a	b
→ q0	q1	q2
q1	q4	q3
q2	q3	q4
q3	q3	q3
*q4	q4	q4

Construct FA which accept string 0011

- String = {0011} string= {11}

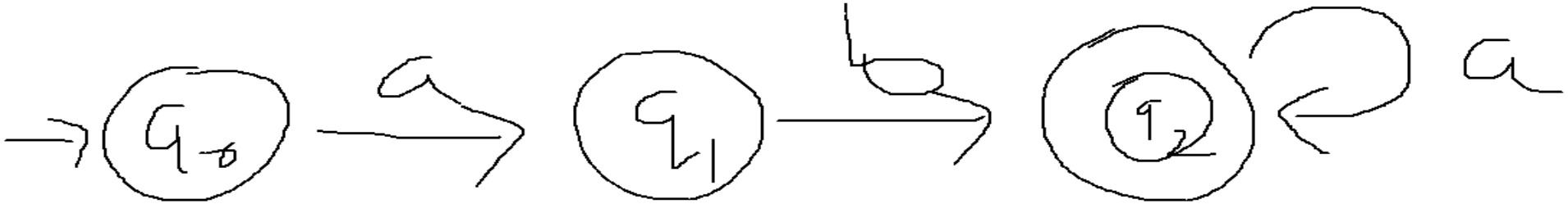


- DFA is define as M = (Q,∑,δ,q0,F) Where
- Q = {q0,q1,q2,q3,q4}
- ∑ = {0,1}
- q0 is initial state
- F = {q4}
- δ is define as table:

State/Input	0	1
→ q0	q1	ϕ
q1	q2	ϕ
q2	ϕ	q3
q3	ϕ	q4
*q4	ϕ	ϕ

Construct FA for the string ab$a^n \mid n \geq 0$

- String = {ab,aba,abaa,abaaa,abaaaa.....}

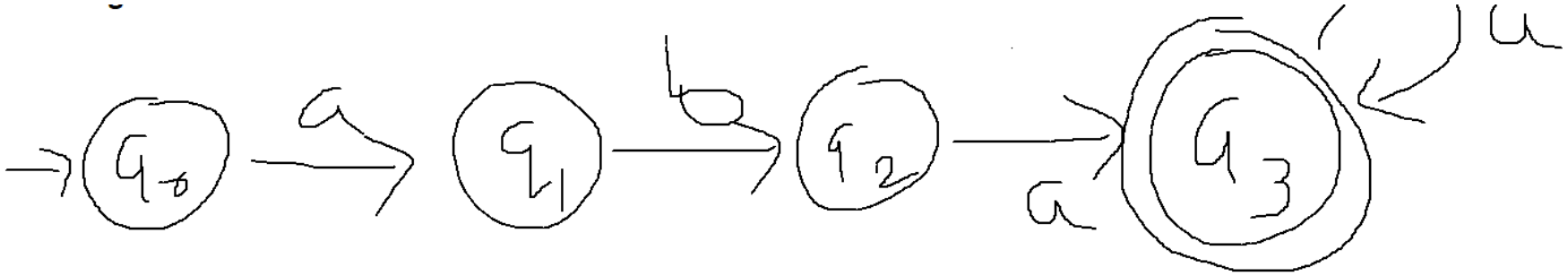


- DFA is define as M = (Q,∑,δ,q0,F) Where
- Q = {q0,q1,q2}
- ∑ = {a,b}
- Q0 initial state
- F = {q2}
- δ is define as table

	a	b
→ q0	q1	ϕ
q1	ϕ	q2
*q2	q2	ϕ

Construct FA for the string ab$a^n \mid n \geq 1$

- String = {aba,abaa,abaaa,abaaaa.....}



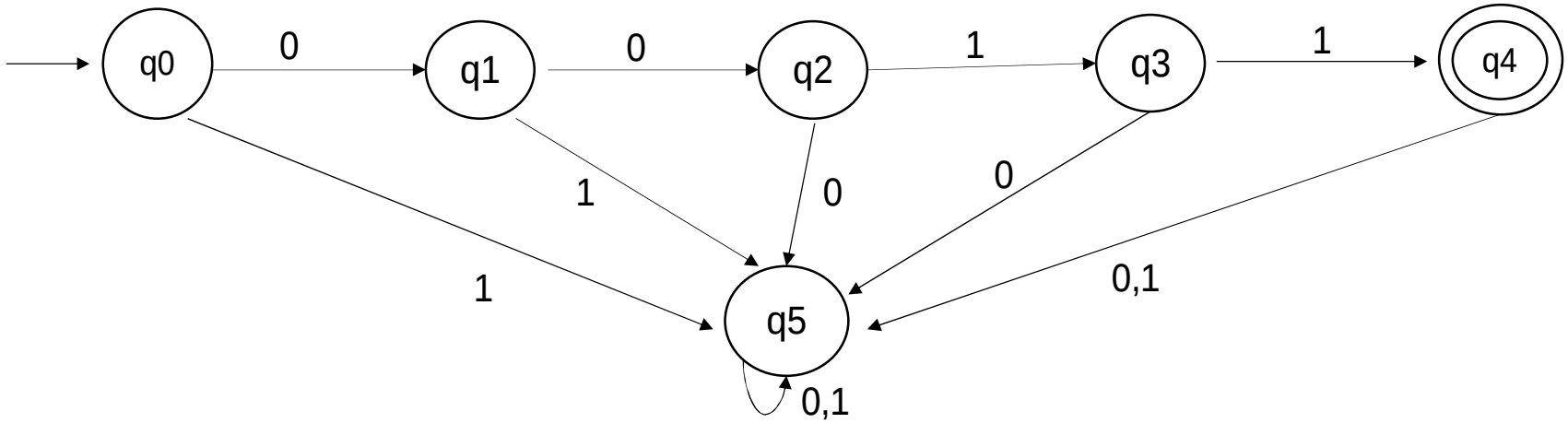
- DFA is define as M = (Q,∑,δ,q0,F) Where
- Q = {q0,q1,q2,q3}
- ∑ = {a,b}
- q0 initial state
- F = {q3}
- δ is define as table

	a	b
→ q0	q1	ϕ
q1	ϕ	q2
q2	q3	ϕ
*q3	q3	ϕ

Dummy or Dead State

- Having no meaning in the generation of the string.
- It is only designed to full the requirements of the finite state machines that every state has one transition for each alphabet.

Machine with dummy state (dead state)

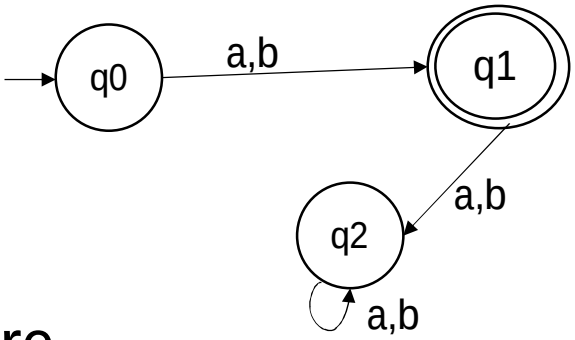


- DFA is define as M = (Q,∑,δ,q0,F) Where
  - Q = {q0,q1,q2,q3,q4,q5}
  - ∑ = {0,1}
  - q0 is initial state
  - F = {q4}
  - δ is define as table:

State/Input	0	1
→q0	q1	q5
q1	q2	q5
q2	q5	q3
q3	q5	q4
*q4	q5	q5
q5	q5	q5

Construct DFA which accept string either a or b

- String = {a,b}
- Language L = {a+b}

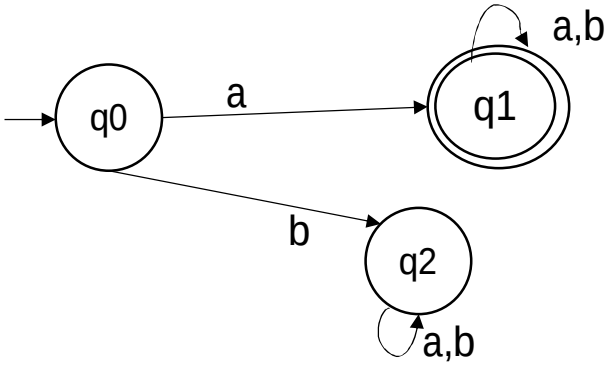


- DFA is define as M = (Q,∑,δ,q0,F) Where
  - Q = {q0,q1,q2}
  - ∑ = {a,b}
  - q0 is initial state
  - F = {q1}
  - δ is define as table:

State/Input	a	b
→ q0	q1	q1
*q1	q2	q2
q2	q2	q2

Construct DFA which accept string start with a over {a,b}

- String = {a,aa,ab,aaa,aab,aba,abb.....}
- Language L = {ax | x ϵ {a,b}*}
  or {a(a+b)*}
  or {ax | x ϵ ∑* }

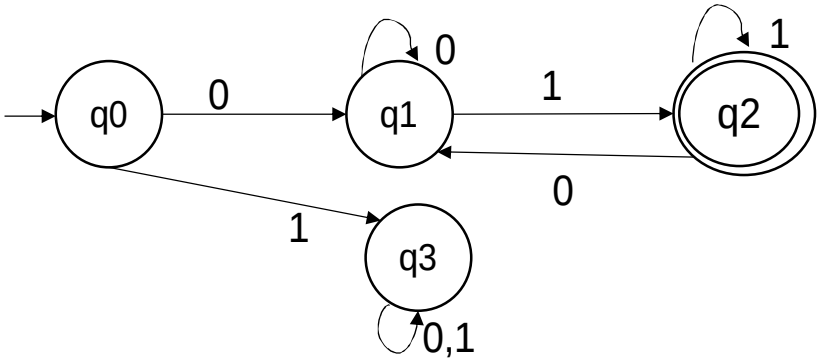


- DFA is define as M = (Q,∑,δ,q0,F) Where
  - Q = {q0,q1,q2}
  - ∑ = {a,b}
  - q0 is initial state
  - F = {q1}
  - δ is define as table:

State/Input	a	b
→ q0	q1	q2
*q1	q1	q1
q2	q2	q2

Construct DFA which accept string start with 0 and end with 1

- String = {01,001,011,0001,0011,0101,0111……}
- Language L = {0x1| x∈{0,1}∗}

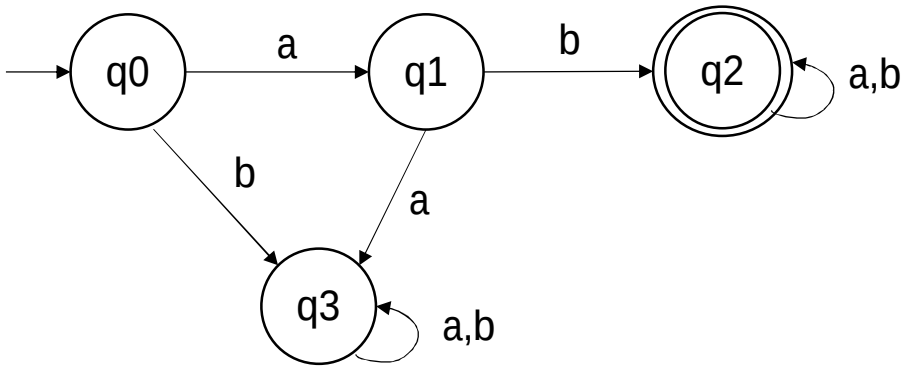


- DFA is define as M = (Q,∑,δ,q0,F) Where
  - Q = {q0,q1,q2,q3}
  - ∑ = {0,1}
  - q0 is initial state
  - F = {q2}
  - δ is define as table:

State/Input	0	1
→ q0	q1	q3
q1	q1	q2
*q2	q1	q2
q3	q3	q3

Construct DFA which accept string starting with string ab

- String = {ab,aba,abb,abaa,abab,abba,abbb……}
- Language L = {abx | x ϵ (a+b)*}

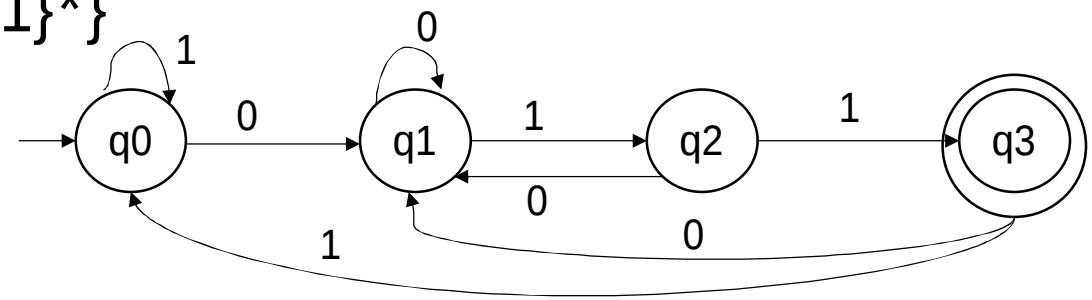


- DFA is define as M = (Q,∑,δ,q0,F) Where
  - Q = {q0,q1,q2,q3}
  - ∑ = {a,b}
  - q0 is initial state
  - F = {q2}
  - δ is define as table:

State/Input	a	b
→q0	q1	q3
q1	q3	q2
*q2	q2	q2
q3	q3	q3

Construct DFA which accept string end with 011

- String = {011,0011,1011,00011,01011,10011,11011.....}
- Language L = {x011 | x ϵ {0,1}*}

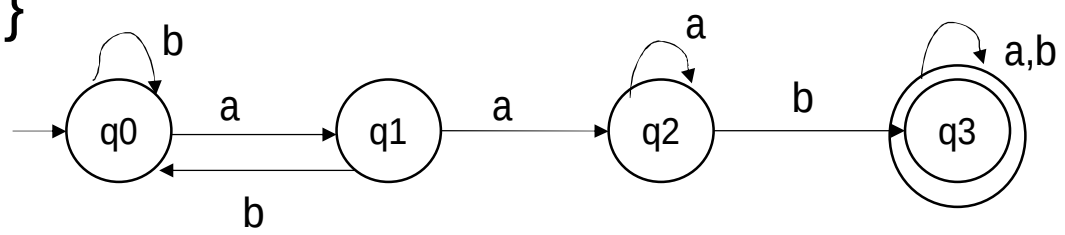


- DFA is define as M = (Q,∑,δ,q0,F) Where
  - Q = {q0,q1,q2,q3}
  - ∑ = {0,1}
  - q0 is initial state
  - F = {q3}
  - δ is define as table:

State/Input	0	1
→ q0	q1	Q0
q1	q1	Q2
q2	q1	Q3
*q3	q1	Q0

Construct DFA which accept string having substring aab

- String = {aab,aaab,baab,aaba,aabb,aaabb,ababb,baabb,bbabb,abbaa.....}
- Language L = {xabby | x,y ϵ {a,b}*}

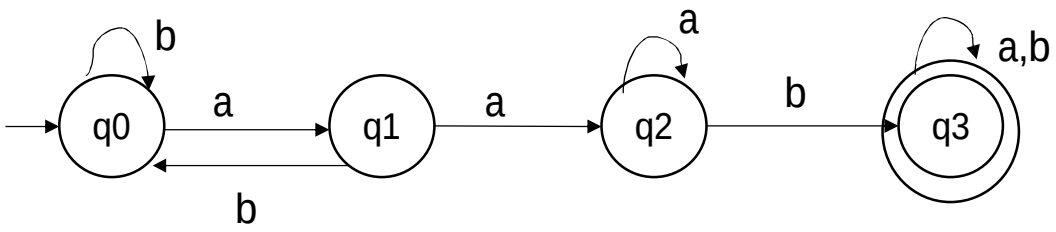


- DFA is define as M = (Q,∑,δ,q0,F) Where
  - Q = {q0,q1,q2,q3}
  - ∑ = {a,b}
  - q0 is initial state
  - F = {q3}
  - δ is define as table:

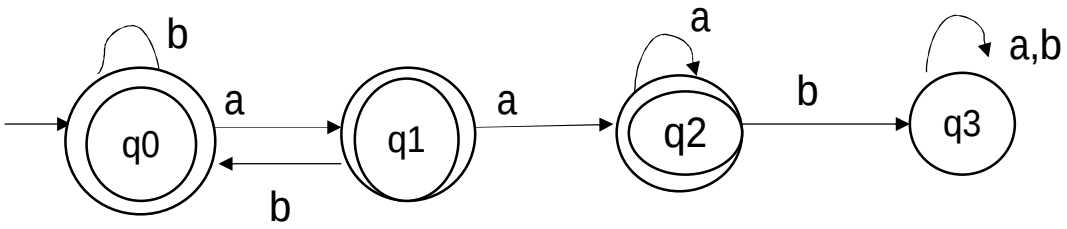
State/Input	a	b
→ q0	q1	q0
q1	q1	q0
q2	q2	Q3
*q3	q3	Q3

Construct DFA which accept string not having substring aab

- Trick :- first construct for having substring aab then reverse it by making final as non final and non final as final
- DFA having substring aab

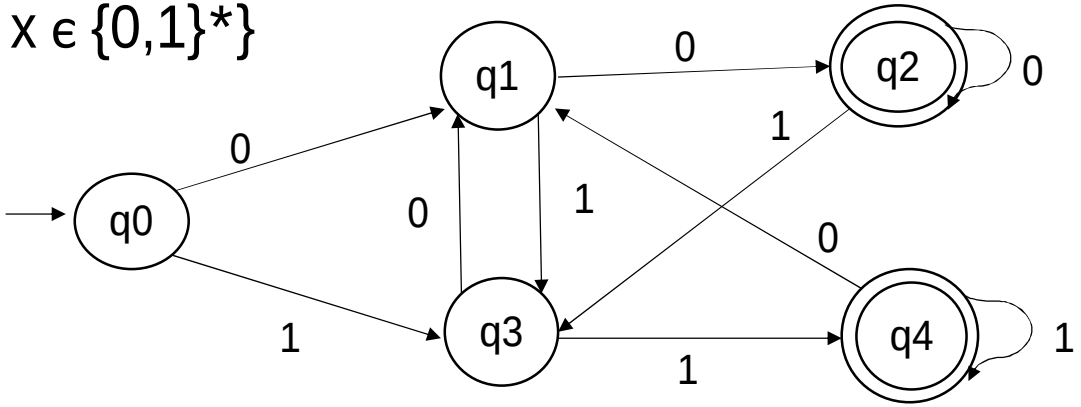


- Now Reverse above DFA by making q0, q1, q2 as final and q3 as non final



Construct DFA which accept string end with either 00 or 11

- String = {00,11,000,011,100,111,0000,0100,1000,1100,0011,0111,1111.....}
- Language L = {x(00+11) | x ϵ {0,1}*}

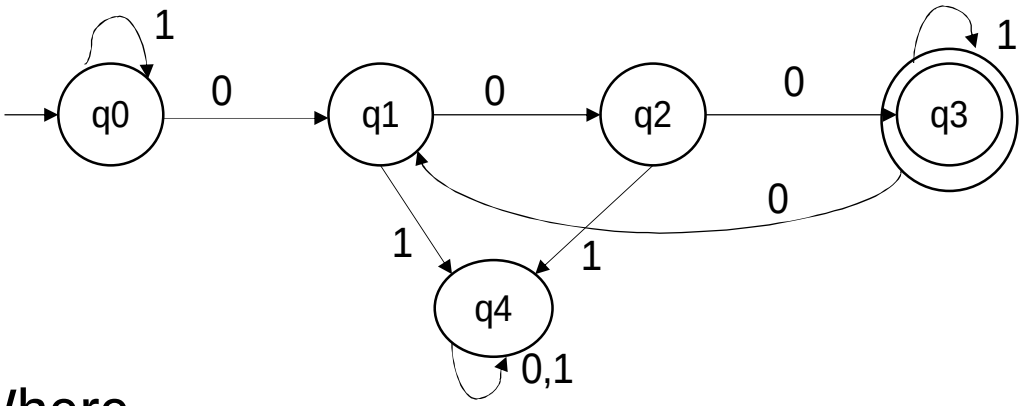


- DFA is define as M = (Q,∑,δ,q0,F) Where
  - Q = {q0,q1,q2,q3,q4}
  - ∑ = {0,1}
  - q0 is initial state
  - F = {q2,q4}
  - δ is define as table:

State/Input	0	1
→q0	q1	q3
q1	q2	q3
*q2	q2	q3
q3	q1	q4
*q4	q1	q4

Construct DFA which accept string have three consecutive 0's over {0,1}

- String = {000,1000,0001,10001,10001000,0001000,000000,.....}
- Language L = { (1+ϵ)*000(000+1)*}

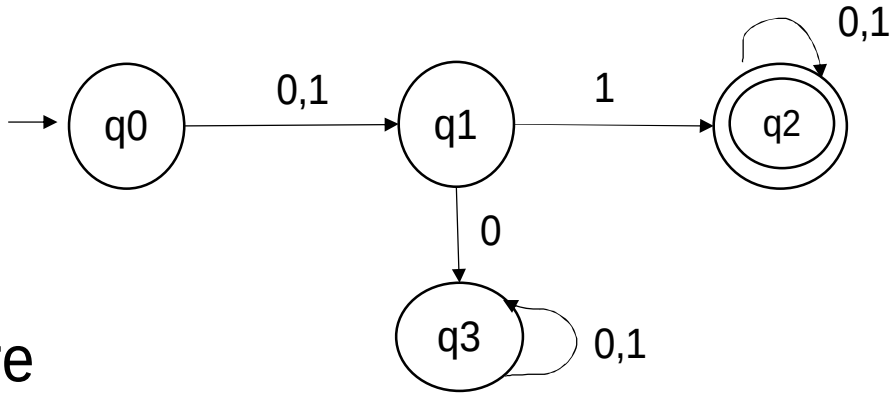


- DFA is define as M = (Q,∑,δ,q0,F) Where
  - Q = {q0,q1,q2,q3,q4}
  - ∑ = {0,1}
  - q0 is initial state
  - F = {q3}
  - δ is define as table:

State/Input	0	1
→ q0	q1	q0
q1	q2	q4
q2	q3	q4
*q3	q1	q3
q4	q4	q4

Construct DFA which accept string of {0,1} whose 2nd symbol from left is 1.

- String = {01,11,010,011,110,111,0100,0101,0110,0111,1100,1101.....}
- Language L = {(0+1)1(0+1)*}



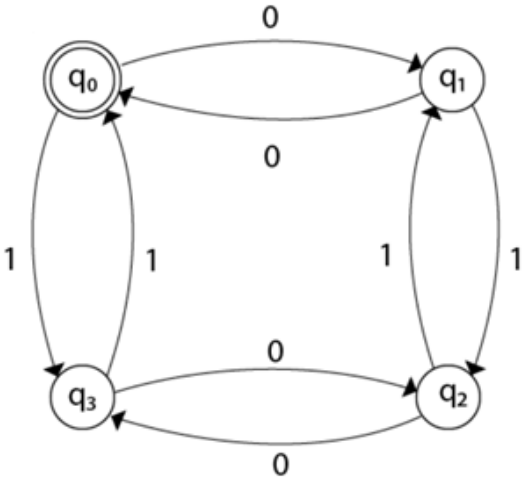
- DFA is define as M = (Q,∑,δ,q0,F) Where
  - Q = {q0,q1,q2,q3}
  - ∑ = {0,1}
  - q0 is initial state
  - F = {q2}
  - δ is define as table:

State/Input	0	1
→ q0	q1	q1
q1	q3	q2
*q2	q2	q2
q3	q3	q3

Construct DFA which accept string having even no of 0's and even no of 1's

- String = {ϵ,00,11,1010,0101,0110,1001,0000,1111,001100,010100.....}
- Language L = {x | x ϵ{0,1}* and

$$|x|_0 mod\ 2 = 0 \ \&\ |x|_1 mod\ 2 = 0\ \}$$



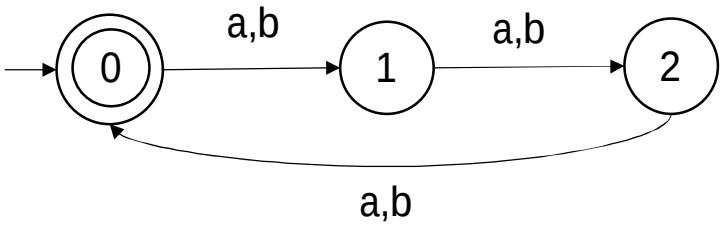
- ***q0 final**: state of even number of 0's and even number of 1's.*
  ***q1 final**: state of odd number of 0's and even number of 1's.*
  ***q2 final**: state of odd number of 0's and odd number of 1's.*
  ***q3 final**: state of even number of 0's and odd number of 1's.*

- DFA is define as M = (Q,∑,δ,q0,F) Where
  - Q = {q0,q1,q2,q3}
  - ∑ = {0,1}
  - q0 is initial state
  - F = {q0}
  - δ is define as table:

State/Input	0	1
→q0*	q1	q3
q1	q0	q2
q2	q3	q1
q3	q2	q0

Construct DFA which accept strings of {a,b} such that length of the string is divisible by 3

- String = {ϵ,aaa,aab,aba,abb,baa,bab,bba,bbb,aaaaaa, aaaaab……..}
- Language L = {((a+b)(a+b)(a+b))*}

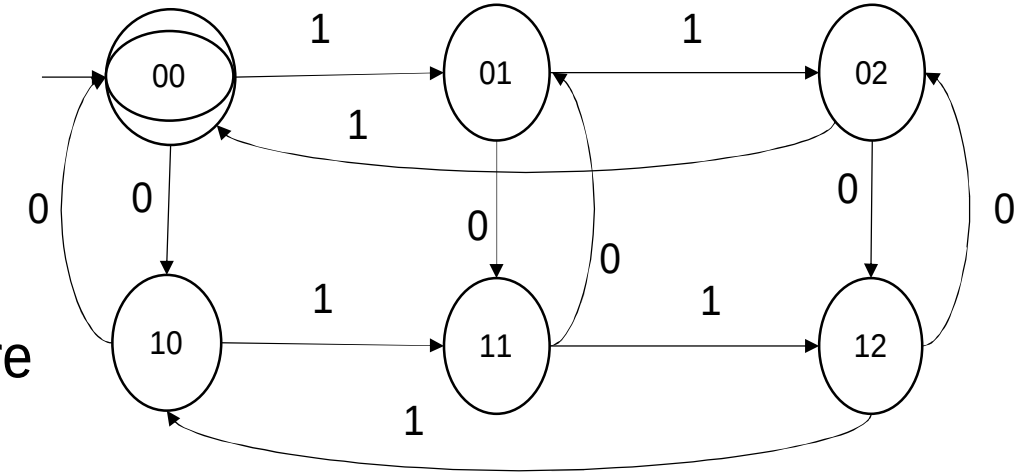


- DFA is define as M = (Q,∑,δ,q0,F) Where
  - Q = {q0,q1,q2}
  - ∑ = {a,b}
  - q0 is initial state
  - F = {0}
  - δ is define as table:

State/Input	a	b
→q0*	q1	q1
q1	q2	q2
q2	q0	q0

Construct DFA which accept strings such that number of 0's divisible by 2 and number of 1's divisible by 3.

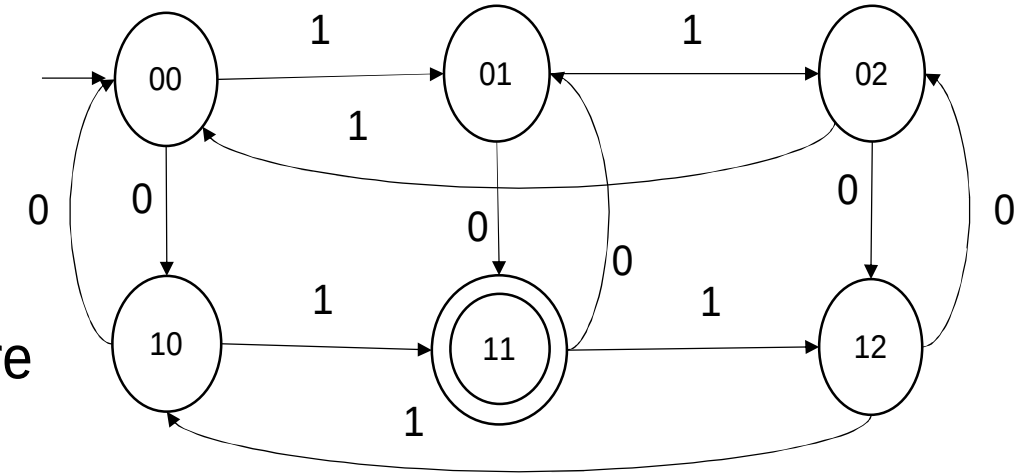
- String = {10101,11001,00111,11100,0000111,1001001111.....}
- Language L = {x | x ϵ{0,1}* and $|x|_0 mod\ 2 = 0$ & $|x|_1 mod\ 3 = 0$ }



- DFA is define as M = (Q,∑,δ,q0,F) Where
  - Q = {00,01,02,10,11,12}
  - ∑ = {0,1}
  - 00 is initial state
  - F = {00}
  - δ is define as table:

Construct DFA which accept strings x such that $|x|_0 mod\ 2 = 1$ and $|x|_1 mod\ 3 = 1$

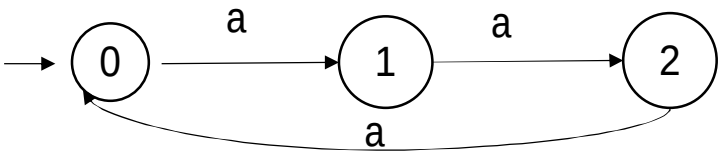
- String = {10101,11001,00111,11100,0000111,1001001111.....}
- Language L = {x | x ϵ{0,1}* and $|x|_0 mod\ 2 = 1$ & $|x|_1 mod\ 3 = 1$ }



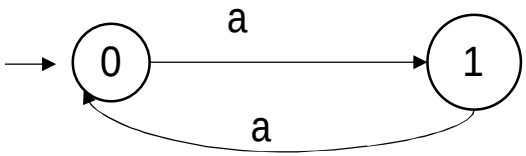
- DFA is define as M = (Q,∑,δ,q0,F) Where
  - Q = {00,01,02,10,11,12}
  - ∑ = {0,1}
  - 00 is initial state
  - F = {11}
  - δ is define as table:

Construct DFA which accept strings w satisfying the following condition: |w| mod 3 >= |w| mod 2 where w ϵ ∑ and ∑ = {a}

- First construct the machine M1 for |w| mod 3



- Then construct the machine M2 for |w| mod 2



- Machine M is the Combination of M1 and M2 i.e M1 x M2 and their states are Q1 x Q2

  Q = Q1 x Q2 = {0,1,2} x {0,1} = {(0,0),(0,1),(1,0),(1,1),(2,0),(2,1)}

- Transition of each states is calculated as

  start with initial state of M, 0 is the initial state of M1 and 0 of M2 so (0,0) is the initial state of M.

State	M δ ({p,q},a)	(M1 ,M2) δ1(p,a) , δ2(q,a)	a
→ (0,0)	δ ({0,0},a)	(δ1(0,a) , δ2(0,a))	(1,1)
(1,1)	δ ({1,1},a)	(δ1(1,a) , δ2(1,a))	(2,0)
(2,0)	δ ({2,0},a)	(δ1(2,a) , δ2(0,a))	(0,1)
(0,1)	δ ({0,1},a)	(δ1(0,a) , δ2(1,a))	(1,0)
(1,0)	δ ({1,0},a)	(δ1(1,a) , δ2(0,a))	(2,1)
(2,1)	δ ({2,1},a)	(δ1(2,a) , δ2(1,a))	(0,0)

- Condition is |w| mod 3 >= |w| mod 2 that means {0,1,2}>= {0,1} so in machine M states
  1. state (0,0) is final because 0 = 0
  2. state (1,1) is final because 1 = 1
  3. state (2,0) is final because 2 > 0
  4. state (1,0) is final because 1 > 0
  5. state (2,1) is final because 2 > 1

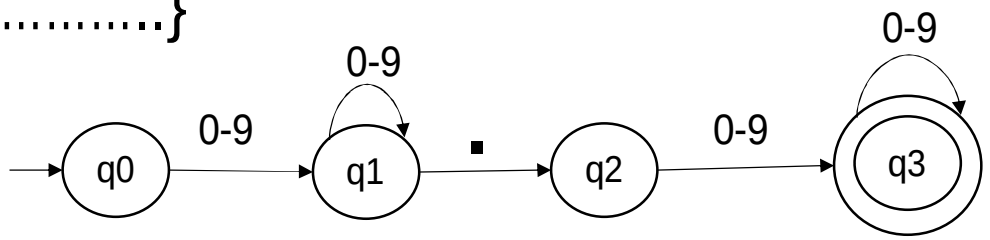
- DFA is define as M = (Q,∑,δ,q0,F) Where
  - Q = {00,01,10,11,20,21}
  - ∑ = {a}
  - 00 is initial state
  - F = {00,10,11,20,21}
  - δ is define as table:

State	a
→*(00)	(11)
*(11)	(20)
*(20)	(01)
(01)	(10)
*(10)	(21)
*(21)	(00)

- Construct DFA which accept strings w satisfying the following condition: |w| mod 3 ≠ |w| mod 2 where w ϵ ∑ and ∑ = {a}
- Construct DFA which accept strings w satisfying the following condition: |w| mod 3 >= |w| mod 2 where w ϵ ∑ and ∑ = {a,b}
- Construct DFA which accept strings w satisfying the following condition: |w| mod 3 ≠ |w| mod 2 where w ϵ ∑ and ∑ = {a.b}

Construct DFA which accept strings of decimal number

- String = {0.23,1.333,4.555333,.....}
- Language L = {x.x |xϵ{0-9}+}

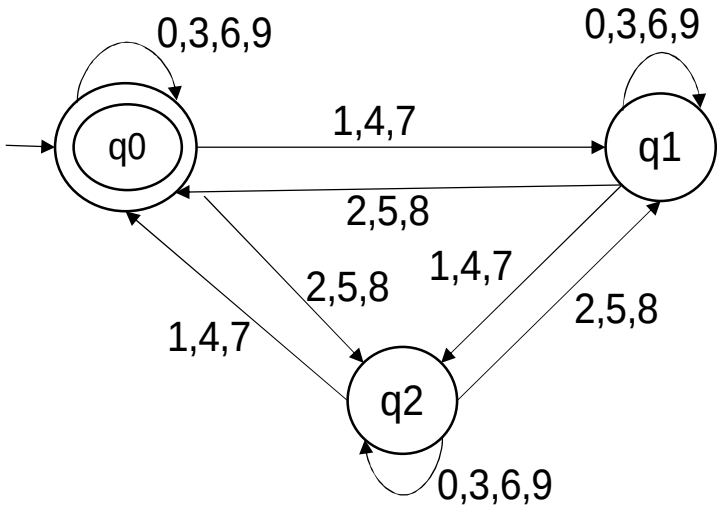


- DFA is define as M = (Q,∑,δ,q0,F) Where
  - Q = {q0,q1,q2,q3}
  - ∑ = {0-9}
  - q0 is initial state
  - F = {q3}
  - δ is define as table:

State/Input	0-9	.
→q0	q1	
q1	q1	q2
q2	q3	
*q3	q3	

Construct DFA which accept strings of integer divisible by 3

- String = {0,3,6,9,12,15,18,21,24,27,30,33,.....}
- Language L = {x | xϵ{0-9}* and x mod 3 = 0}

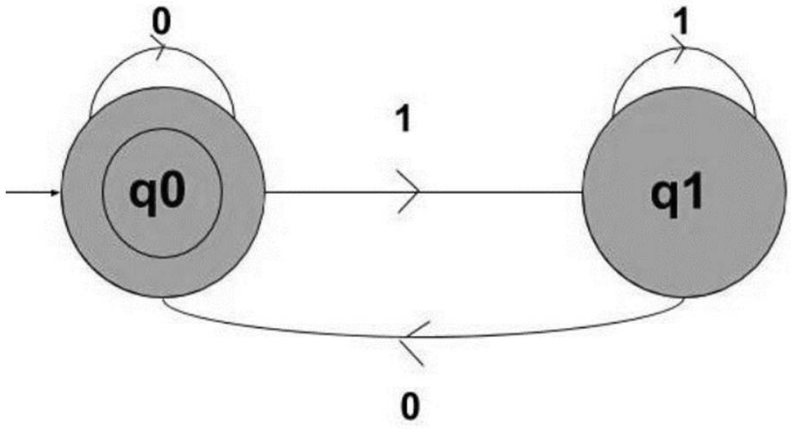


- DFA is define as M = (Q,∑,δ,q0,F) Where
  - Q = {q0,q1,q2}
  - ∑ = {0-9}
  - q0 is initial state
  - F = {q0}
  - δ is define as table:

State/Input	0,3,6,9	1,4,7	2,5,8
→q0*	q0	q1	q2
q1	q1	q2	q0
q2	q2	q0	q1

Construct DFA which accept strings binary divisible by 2

- String in integer = {0,2,4,6,8,10,12,14,16.....}
- String in binary = {00,10,100,110,1000,1100,1110,10000.....}

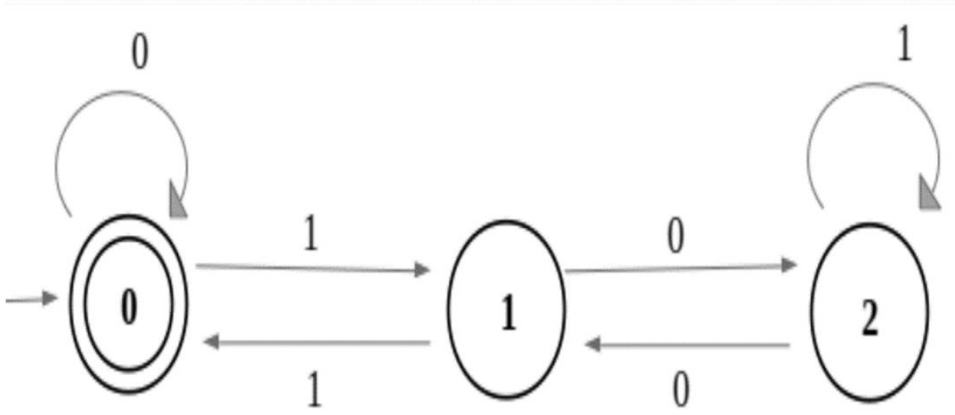


- DFA is define as M = (Q,∑,δ,q0,F) Where
  - Q = {q0,q1}
  - ∑ = {0,1}
  - q0 is initial state
  - F = {q0}
  - δ is define as table:

State/Input	0	1
→*q0	q0	q1
q1	q0	q1

Construct DFA which accept strings binary divisible by 3

- String in integer = {0,3,6,9,12,15,18,21,24,27,30,33,.....}
- String in binary = {00, 11,101,1001,1100,1111,10010,10101,}

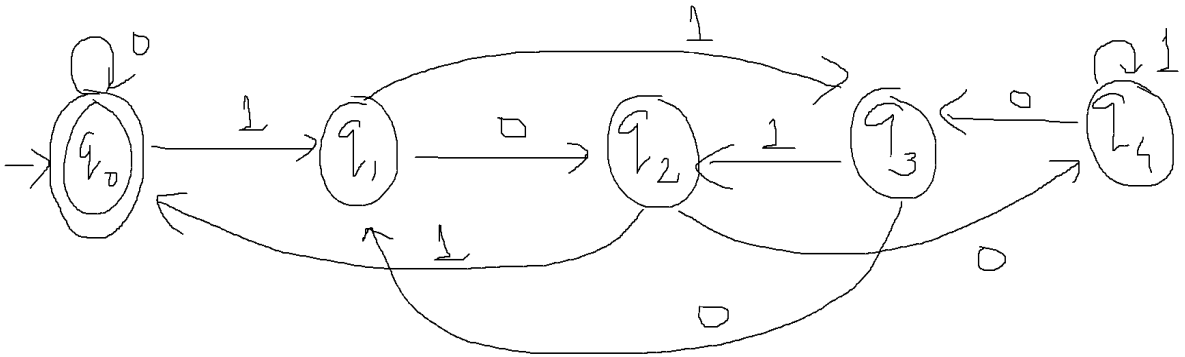


- DFA is define as M = (Q,∑,δ,q0,F) Where
  - Q = {0,1,2}
  - ∑ = {0,1}
  - 0 is initial state
  - F = {0}
  - δ is define as table:

State/Input	0	1
→ 0*	0	1
1	2	0
2	1	1

Construct DFA which accept strings binary divisible by 5

- String in integer = {0,5,10,15,20,25,30,35,.....}
- String in binary = {0,101,1010,1111,10100,11001,11110}



- DFA is define as M = (Q,∑,δ,q0,F) Where
  - Q = {0,1,2,3,4}
  - ∑ = {0,1}
  - 0 is initial state
  - F = {0}
  - δ is define as table:

State/Input	0	1
→ 0*	0	1
1	2	3
2	4	0
3	1	2
4	3	4