AUTOMATA FORMAL LANGUAGES AND LOGIC

Lecture Notes on Deterministic Finite Acceptor/Automata(DFA)

Prepared by :
Prof. Sangeeta V I
Assistant Professor

Department of Computer Science & Engineering
PES UNIVERSITY
(Established under Karnataka Act No.16 of 2013)
100-ft Ring Road, BSK III Stage, Bangalore – 560 085

Table of Contents

Examples Solved

8	The language of strings over Σ ={a,b} where every string must contain “aa” as the substring.	**17**
9	The language of strings over Σ ={a,b} of the form $a^n\ b^m$ |n,m>=1.	**18**
10	The language of strings over Σ={a,b} of the form $a^n\ b^m$ |n,m>=0.	**19**
11	The language of strings over Σ={a,b} where, n_a(w)mod 2=0	**21**
12	The language of strings over Σ={a,b} where, n_a(w)mod 2=0 and n_b(w)mod 2=0	**22**
13	The language of strings over Σ={a,b} where, n_a(w)mod 3=0 and n_b(w)mod 2=0	**23**
14	The language of strings over Σ={a,b} where, n_a(w)mod 3=0 and n_b(w)mod 3=0	**24**
15	The language of strings over Σ={a,b} where, n_a(w)mod 3 = **2** and n_b(w)mod 3 = **1**	**25**
16	DFA for binary number divisible by 2.	**26**
17	DFA for binary number divisible by 3.	**28**

1.Formal Definition of a DFA

A DFA can be represented by a 5-tuple $(Q, \sum, \delta, q_0 F)$ where –

- Q: a finite set of internal states
- Σ: a set of symbols called the input alphabet
- δ: a transition function from Q X Σ to Q
- q_0: the initial state
- F: a subset of Q representing the final states

In a DFA, for a particular input character, the machine goes to one state only. A transition function is defined on every state for every input symbol.

2.Language Accepted or Recognized by a DFA :

The language accepted or recognized by a DFA M is the set of all strings accepted by M , and is denoted by

$L(M)=\{w \epsilon \Sigma | M \text{ accepts } w\}$

3.Constructing a DFA

We can construct automata for every string in the finite language.So,every finite language has a finite acceptor .

3.1Graphical Representation of a DFA

A DFA is represented by digraphs called state diagrams.

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

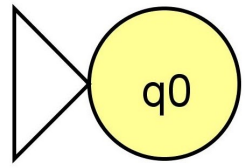
Length of a string : The number of symbols in a string w is called its length, denoted by |w|.

Example : | 011 | = 4, |11| = 2, | b | = 1

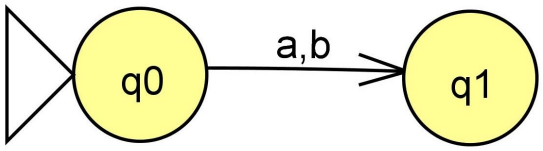
Example 1:

Assume Σ={a,b}
Consider language L1={strings of length 2 over Σ }={w | |w|=2}
Minimum strings:{ab,ba,aa,bb}
L1={ab,ba,aa,bb} The language is finite and we can construct a DFA.
We will construct the automata for the minimum string.

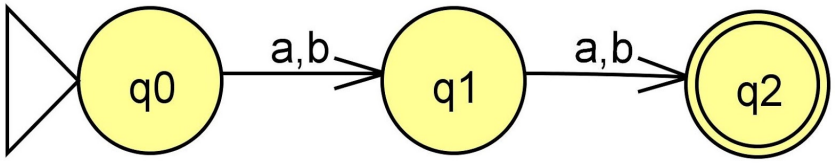
- Start state q0 with an arrow.



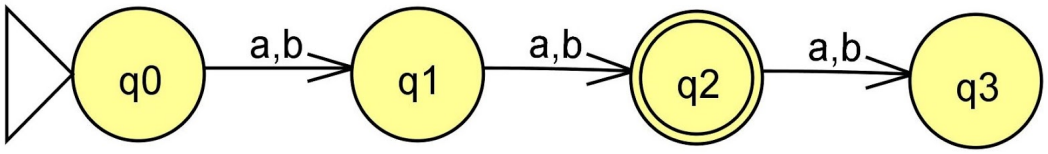
- The first symbol could be either a or b, we accept it and move to the next state q1.



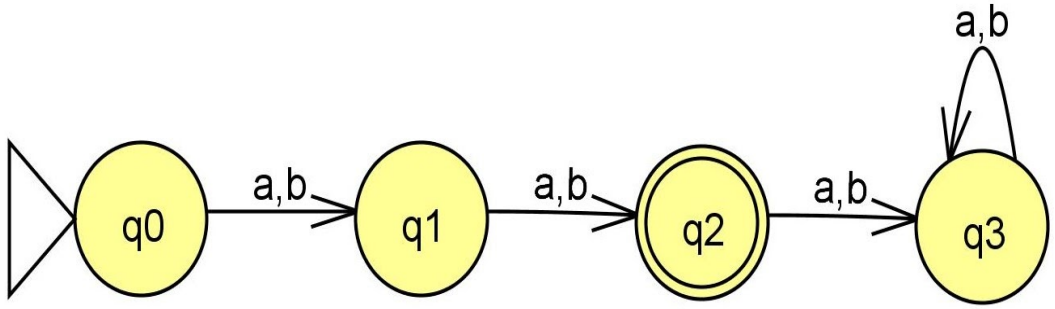
- The second symbol could be either a or b,we accept it and move to the next state q2.



- When we reach q2 we have covered all possible combinations ab,ba,aa,bb so we can make q2 as the final state.
- If we encounter any strings of length greater than 2 ,we reject such strings!

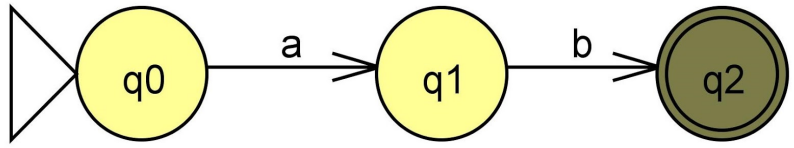


- On state q2 when we get any symbol we reject it and send it to q3 , a non accepting state/non final state/reject state/trap state/dead state.



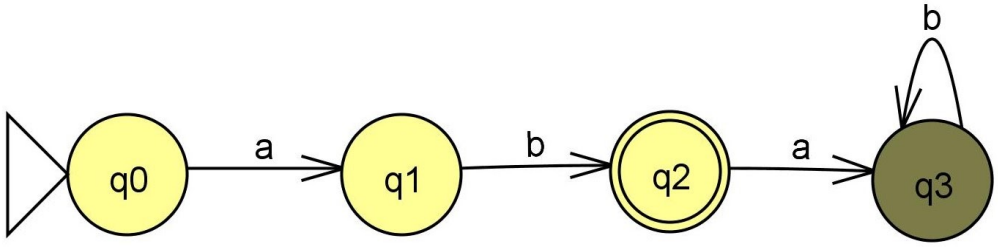
- On state q3 when we get any symbol they remain in q3.
- Any string which lands in a non-accepting state is rejected .

❖ Tracing the string ab.



String ab is accepted as it lands in state q2.

❖ Tracing the string abab



abab is rejected as it lands in state q3 which is a trap state/dead state.

Example 2:

Assume Σ={a,b}
Consider language L2={strings of length <=2 }={w | |w|<=2}

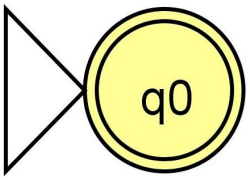
We will enumerate the elements in the language.

- Minimum Strings:
  - String of length 0=λ(empty string)
  - Strings of length 1={a,b}
  - Strings of length 2={ab,ba,aa,bb}

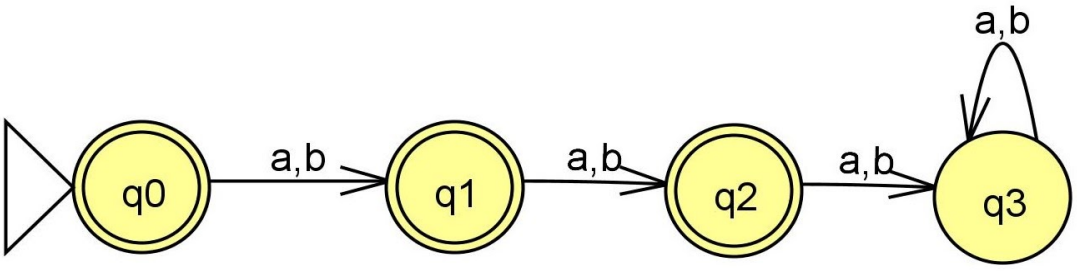
L2={λ,a,b,ab,ba,aa,bb}

We will construct the automata for the minimum string.

- Whenever λ (empty string) is a part of the language ,the start state itself will be the final state.



- On state q0, if we see a or b ,we accept it as a and b belong to the language and move to state q1.
- On state q1, if we see a or b we accept it as strings ab,ba,aa,bb are accepted and move to state q2.
- On state q2, if we see a or b (strings of length >2) we move to q3 which is dead or a trap state.



Example 3:

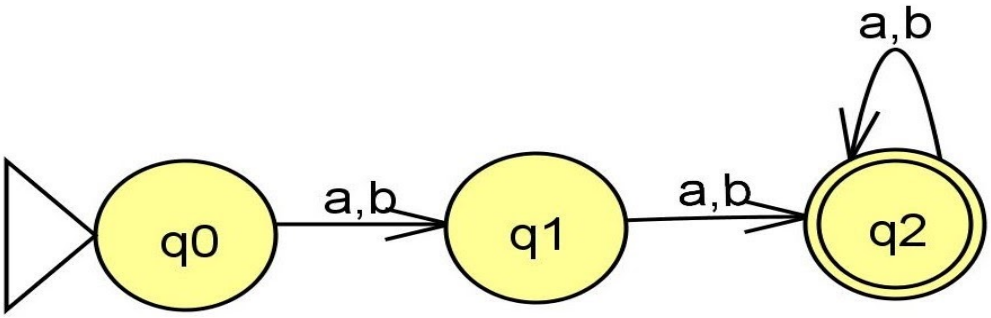
Assume Σ={a,b}
Consider language L3={strings of length >=2 }={w | |w|>=2}

We will enumerate the elements in the language.

- Minimum Strings:
  - String of length 2={ab,ba,aa,bb}
  - Strings of length 3={aaa,aab,aba,abb,.....}
  - Strings of length 4={aaaa,bbbb,.....} and so on.

L3={ab,ba,aa,bb,aaa,aab,aba,abb,.....,aaaa,bbbb,.....}
We will first construct the automata for the minimum string .

- q0 is the start state .
- On q1, which is a non accepting state,if we see a or b we move to state q2.
- q2 is accepting state as we have encountered strings of length 2 .
- On q2 ,we have a jolly ride /self loop on a or b as we can accept strings of length >2.
  We accept all strings of length >=2 and reject strings of length <2.



Example 4:

Assume Σ={a,b}
Consider language L4={strings which start with a }

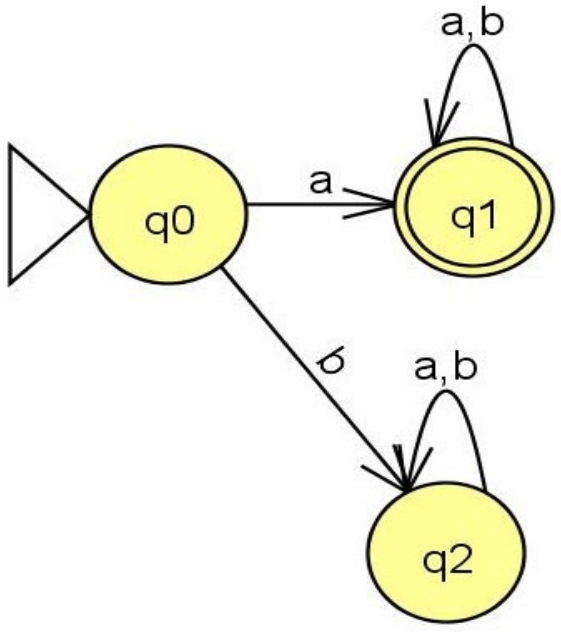
We will enumerate the elements in the language.

- Minimum Strings:
  - String of length 1:{a}
  - Strings of length 2:{ab,ba,aa,bb}
  - Strings of length 3:{aaa,abb,aba,aab} =a (a/b) (a/b)
  - String of length 4 and so on .

L4={a,ab,ba,bb,aaa,abb,aba,aab,.....}

We will first construct the automata for the minimum string .

- q0 is the start state .
- On q0, if we encounter a we move to q1 ,if we encounter b we reject it(reject strings starting with b).
- On q1, we will have a jolly ride/self loop on a,b.(accept string of length >1).
- All the strings starting with b are rejected.q2 is a trap/dead state.
- On q2, if we encounter a or b we will have a jolly eide/self loop .



3.2 Representation of DFA with Transition Table:

It is basically a tabular representation of the transition function that takes two arguments (a state and a symbol) and returns a value ("the next state").

- Rows correspond to states,
- Columns correspond to input symbols,
- Entries correspond to next states
- The start state is marked with an arrow
- The accept states are marked with a star (*).

Transition table for Example 4:

	a	b
→q0	q1	q2
q1	q1	q1
✱q2	q2	q2

In the transition table for a DFA ,we must specify the transition of each state and every input symbol from the alphabet exactly once.
Every state on every input symbol will go only to one state.

3.3 Representation of DFA with Transition Function δ:

The transition function describes how the machine changes its internal state.
It is a function

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

from (state,input symbol)pait to state.If the machine is in state q and the present input symbol is a then the machine changes its internal state to $\delta(q,a)$ and moves to the next input symbol.

The basic transition function δ gives the new state of the machine after a single symbol/letter is read. The extended function (that we will first denote by δ^*) gives the new state after an arbitrary string of symbols is read. In other words, $\delta^*$is a function.

- δ^*: $Q \times \Sigma^* \rightarrow Q$. For every state q and word w the value of $\delta^*(q,w)$ is the state that the DFA reaches if in state q it reads the input word w.

The language accepted or recognized by a DFA M is the set of all strings accepted by M , and is denoted by

$L(M)=\{w \in \Sigma^* \mid \delta^*(q0,w) \in F\}$

Transition function for Example 4:

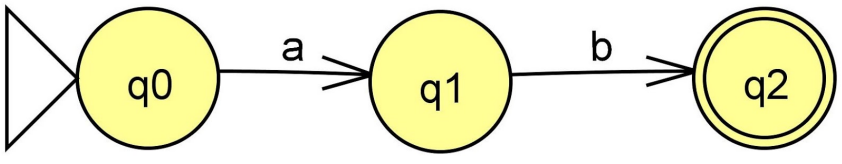
- $\delta(q0,a)=q1$
- $\delta(q0,b)$-q2
- $\delta(q1,a)=q1$
- $\delta(q1,b)=q1$
- $\delta(q2,a)=q2$
- $\delta(q2,b=)q2$

Example 5:

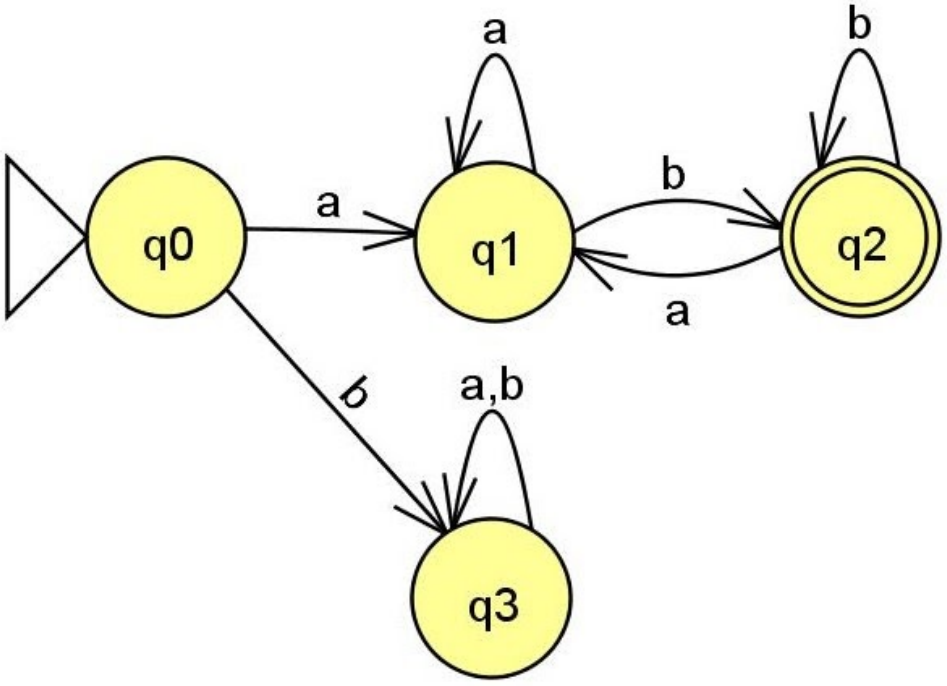
L5={w|awb,wϵ{a,b}*}
The language defines strings which start with a and end in b.
We will enumerate the elements in the language.

- Minimum String:ab

L5={ab,a.(followed by any number of a's or b's). b}
We will first construct the automata for the minimum string ab.



- DFA for the minimum string ab.



- On q0,if we see b re will reject and send to satte q3, it would violate the condition.
- On q1, we can have a jolly ride/self loop on a.
- On q2, we can have a jolly ride/self loop on b.
- On q2,if we see the symbol a we move to state q1 so that we do not miss on string like ababa,abaaab etc

Example 6:

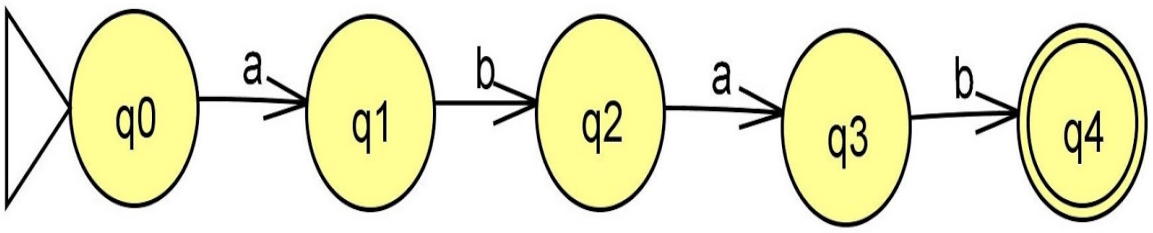
L6={w|abwab,w**ϵ**{a,b}*}

We will enumerate the elements in the language.

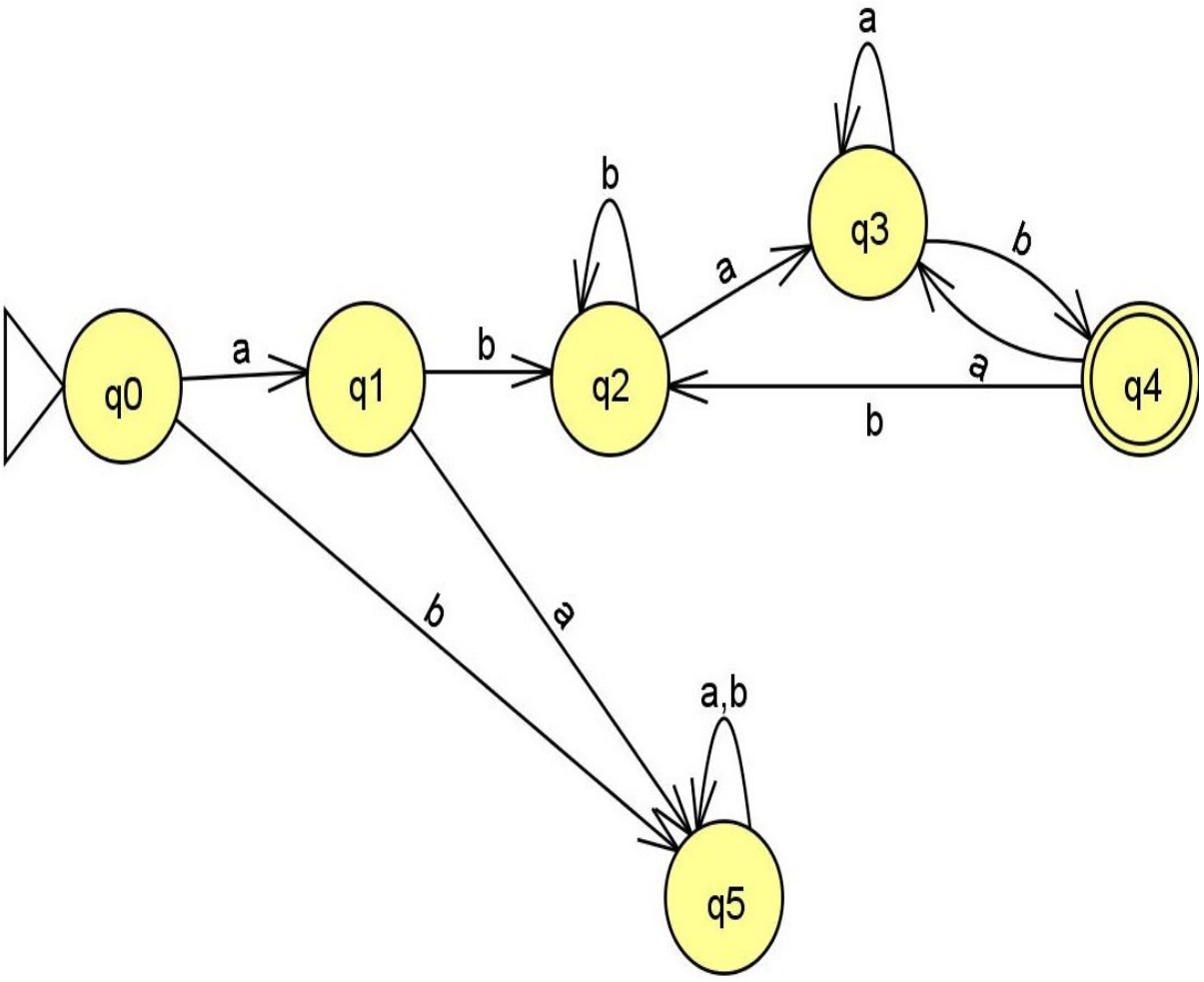
- Minimum String:abab

L6={abab,ab (any number of a's or b's) ab}

We will first construct the automata for the minimum string abab



- DFA for the minimum string abab.



- On state q0,if we see ‘b’ we move to state q5(dead/trap state) as we reject strings starting with b.
- On q1, if we see ‘a’ we move to dead/trap state as we accept strings starting with ab.
- On q2 we can have a self loop on ‘b’ .
- On q3,we can have a jolly ride/self loop on ‘a’.
- On state q4,if we see the symbol ‘a’ we move back to state q3 ,so that we accept strings of the form :ababab
- On state q4 ,if we see the symbol ‘b’ we move back to state q2 ,so that we accept strings of the form:ababbab.

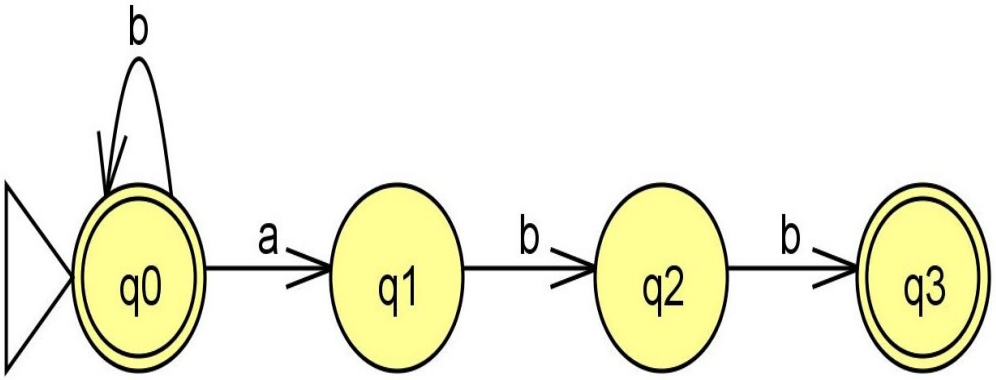
Example 7:

L7={Every a followed by bb}
We will enumerate the elements in the language.

L7={λ,b,bb……,abb,b….babb….abb…...}

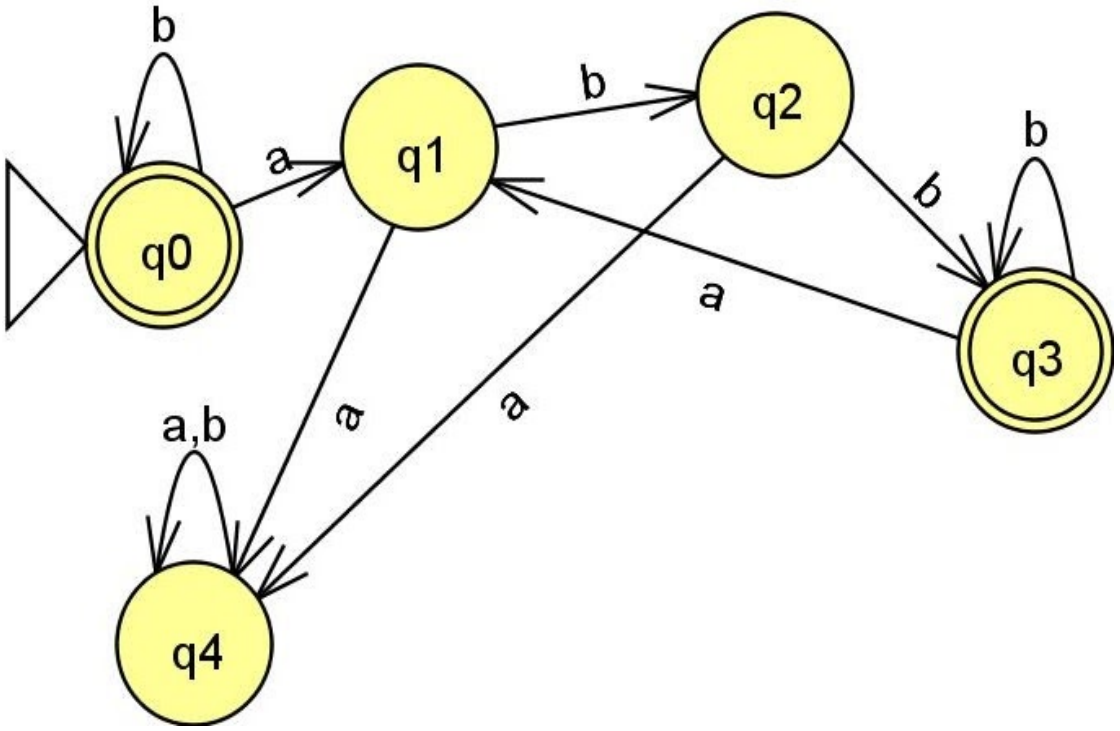
We will first construct DFA for the minimum string.

- Since λ is a per of the language,start state itself is the final state.



- On q0,if we see a ‘b’ ,we can have a jolly ride on ‘b’ ,since there is no restriction on the number of b’s.
- On q0,if we see an ‘a’ it has to be followed by ‘bb’

.



- On state q1 if we see ‘a’,we move t o dead/trap state.
- On state q2,if we see ‘a’,we move to dead/trap state.
- On state q3,if we see ‘a’ we move back to q1,so we take care of strings like abbabb
- On state q3,since there is no restriction on ‘b’ ,we have a jolly ride/self loop on ‘b’.

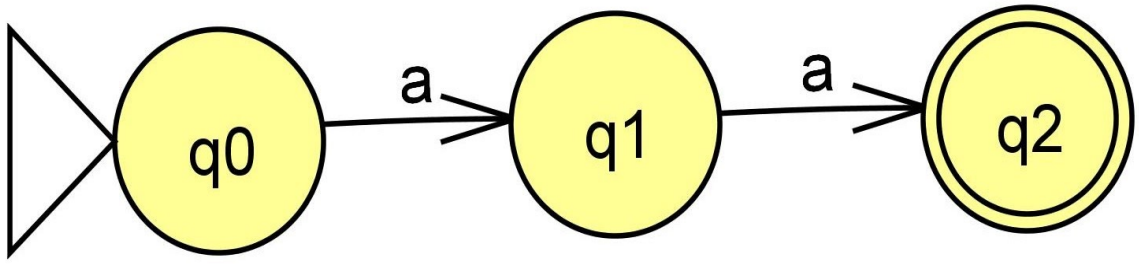
Example 8:

L8={Every string must contain “aa” as the substring}

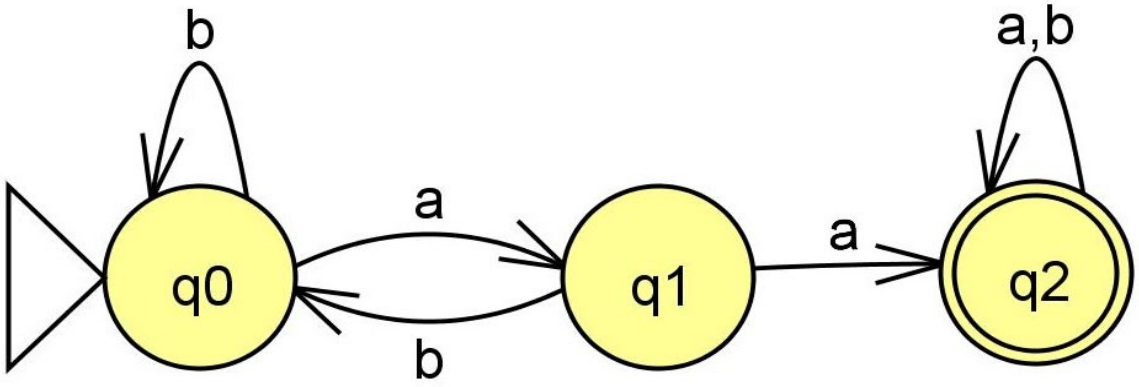
We will enumerate the elements in the language.

L8={aa, (any number of a’s or b’s) aa (any number of a’s or b’s) }

We will first construct DFA for the minimum string.



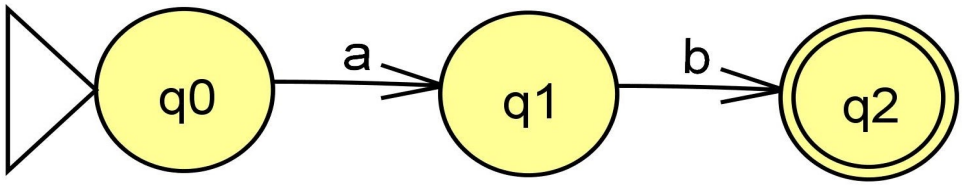
- DFA for the minimum string aa.



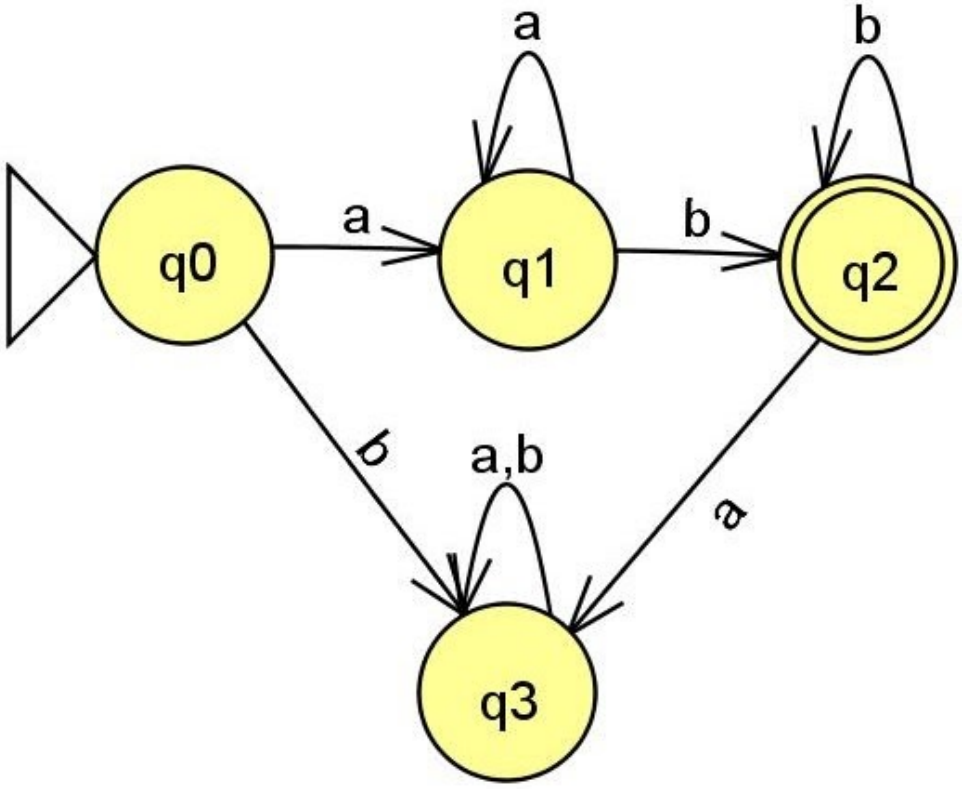
- On state q0,if we see ‘b’ we can have a jolly ride/self loop .
- On q1,if we see ‘b’ we will go back to q0 ,so that we can take care of strings like abaab,abbabaabb etc
- On q2,we will have seen “aa” ,now can see any number of a’s or b’s so we can have a jolly ride/self loop on q2.

Example 9:

L9={ a^n b^m |n,m>=1}
We will enumerate the elements in the language.
L9={ab,aab,aaaabbbbbbbbb,..... }
Minimum String: ab
We will first construct DFA for the minimum string.



- DFA for the minimum string ab.



- On state q0 ,if we see symbol 'b' we move to dead/trap state as the language is a's followed by b's.
- On q1 ,we can have a jolly ride/self loop on 'a'.
- On q2,we see symbol 'a' we will move to dead/trap state as the language is a's followed by b's,and we cannot have a's after b's.

Example 10:

L10={ a^n b^m |n,m>=0}

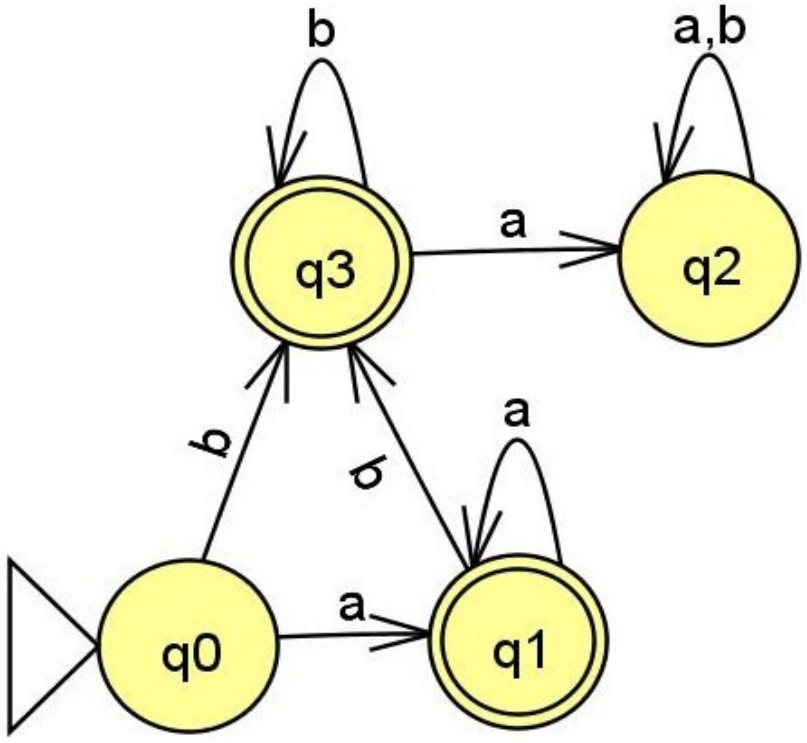
We will enumerate the elements in the language.

n (number of a's)	m (number of b's)	
0	> 0	any number of b's
> 0	0	any number of a's
0	0	λ (empty string-no a's ,no b's)
> 0	> 0	some a's followed by some b's

L10={λ , any number of a's , any number of b's , some a's followed by some b's }
Minimum String: ab

We will first construct DFA for the minimum string.

Since λ is a part of the language,start state itself is a final state.



- On q0,if we see input symbol b we move to q3 ,as the string can have only b's.

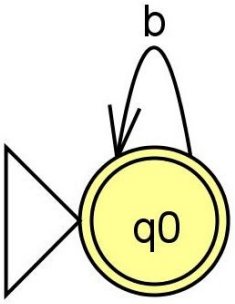
- We reach q3 from q0 on symbol 'b'.If we see the symbol 'a' in q3 we move it to the dead/trap state as it would not belong to the language.

- On q1,we see the symbol 'b' and we move to state q3.

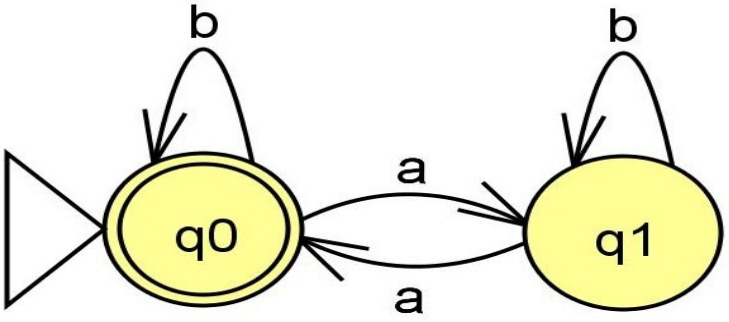
Example 11:

L11={n_a(w)mod 2=0}
We will enumerate the elements in the language.
Note that there is no restriction on the number of b's.
L11={ $\boldsymbol{\lambda}$,any number of b's,even number of a's and any number of b's in between}

We will first construct DFA for the minimum string.
Since $\boldsymbol{\lambda}$ is a part of the language,start state itself is a final state.



- Since,there is no restriction on the number of b's so we can have a jolly ride/self loop on b in q0.

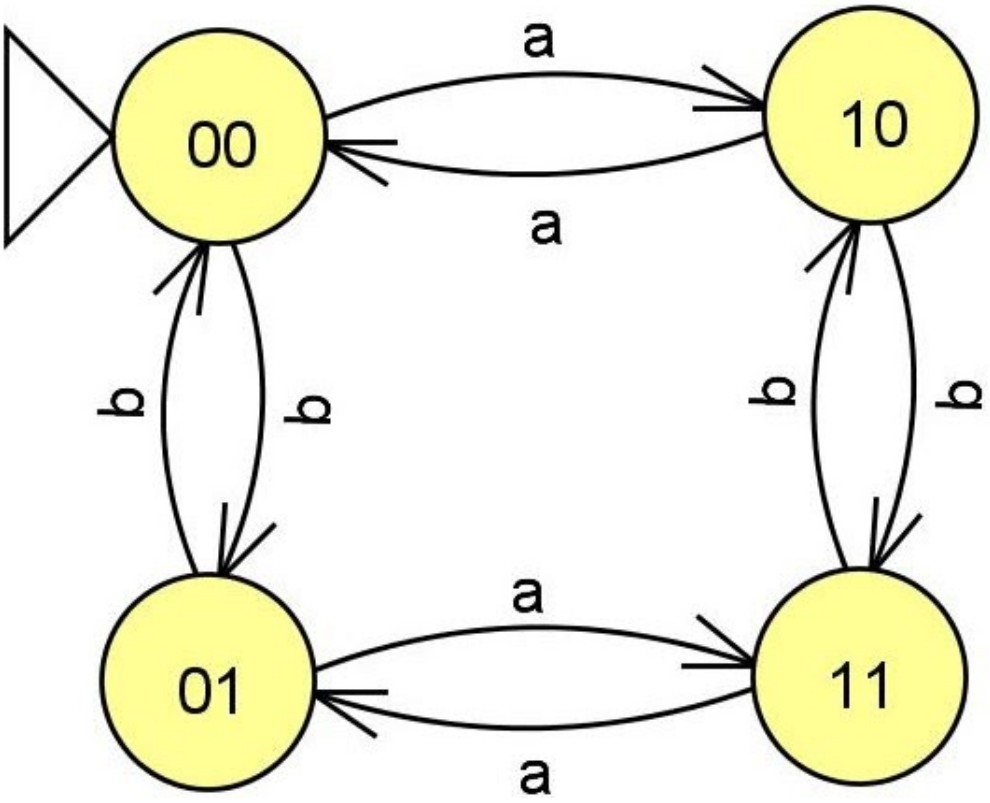


- On state q0 ,if we see the symbol 'a' we move to state q1,we will not accept a single 'a' so we will not make q1 as an accepting state.On q1,if we see the symbol 'a' we move back to q0. So only when we see 2 a's we reach the final state.

- The DFA accepts only even numbers of a's.

Example 12:

L12={n_a(w)mod 2=0 and n_b(w)mod 2=0 }



- 1 indicates odd
- 0 indicates even
- 00 indicates even a's and even b's
- 10 indicates odd a's and even b's
- 01 indicates even a's and odd b's
- 11 indicates odd a's and odd b's

Number of states in a machine for modulo questions:

- Single symbol
  n_a(w)mod n=0
  The number of states in the DFA is 'n' i.e the number of remainders for mod n.

- Two symbols
  If the language is n_a(w)mod n=0 and n_b(w)mod n=0 then the number of states in the DFA is m✖n.

Example 13:

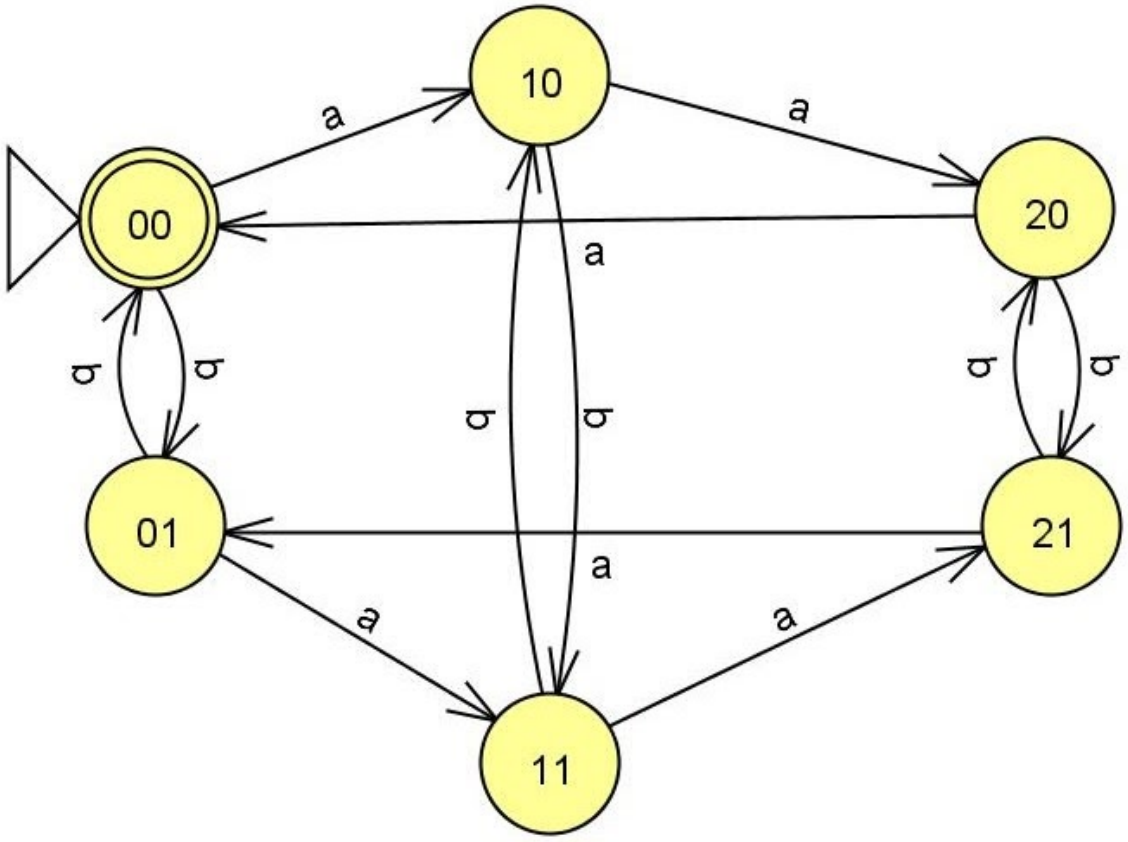
L13={n_a(w)mod 3=0 and n_b(w)mod 2=0 }

The remainder on mod 3 is 0,1,2.
So in DFA for n_a(w)mod 3=0 there will be 3 states .

The remainders on mod 2 is 0,1
So in DFA for n_b(w)mod 2=0 there will be 2 states

Therefore,the DFA for the language L13 will have 3✖2=6 states.

- In the DFA we have all the a's horizontally and all the b's vertically.



- The states are named based on the remainder on the number of a's and b's.

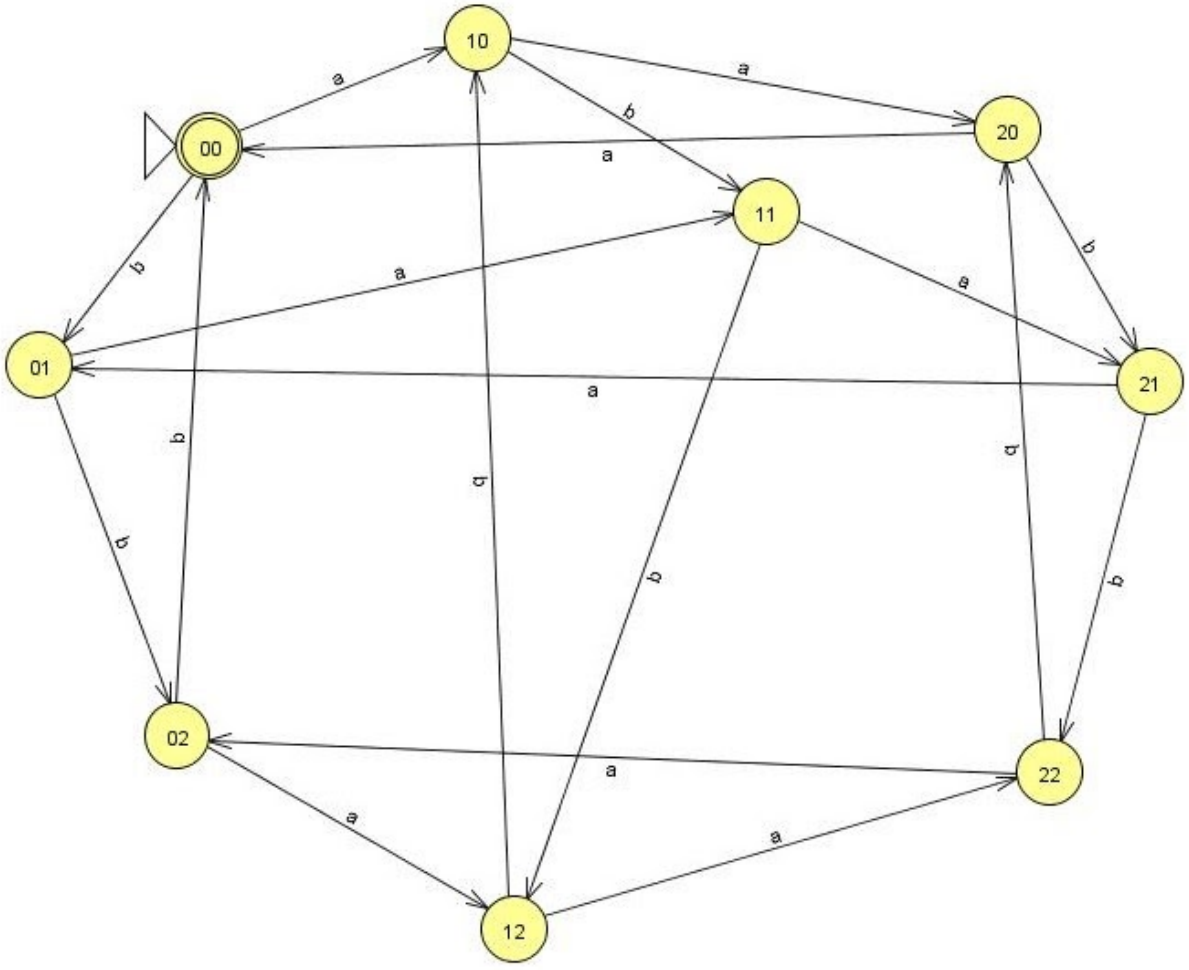
Example 14:

L14={n_a(w)mod 3=0 and n_b(w)mod 3=0 }

The remainder on mod 3 is 0,1,2.
So in DFA for n_a(w)mod 3=0 there will be 3 states .

Also in DFA for n_a(w)mod 3=0 there will be 3 states .

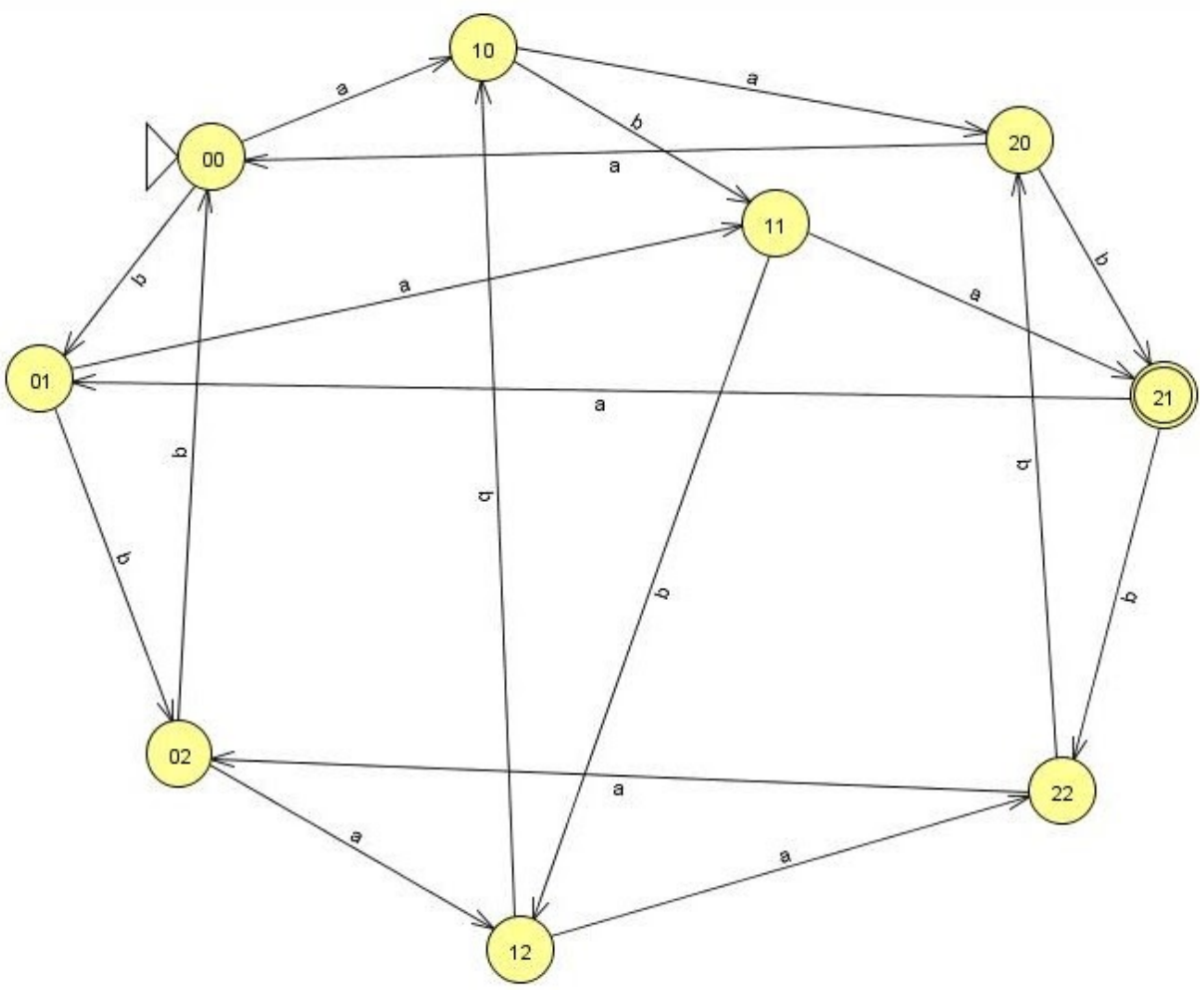
Therefore,the DFA for the language L14 will have 3✖3=9 states..



- The states are named based on the remainder on the number of a's and b's.

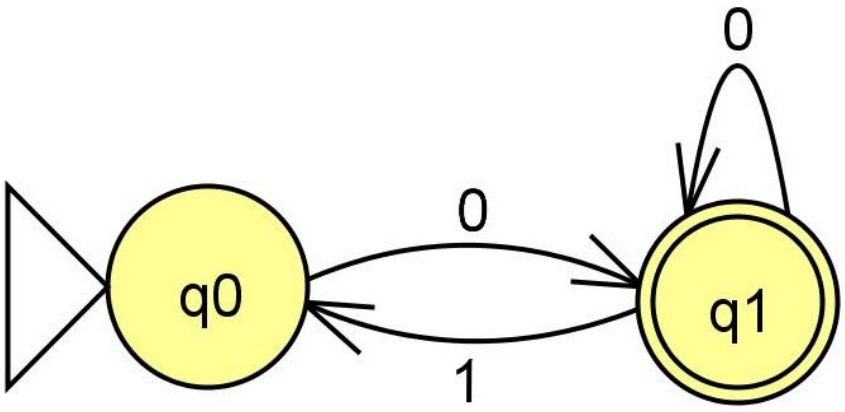
Example 15:

L15={n_a(w)mod 3 = **2** and n_b(w)mod 3 = **1** }
We can refer to the same DFA constructed for the language L14.
The DFA for L15 would be the same as DFA for L14,but there will be change in the final state.
The final state for L15 will be the state numbered **21**.



Example 16:

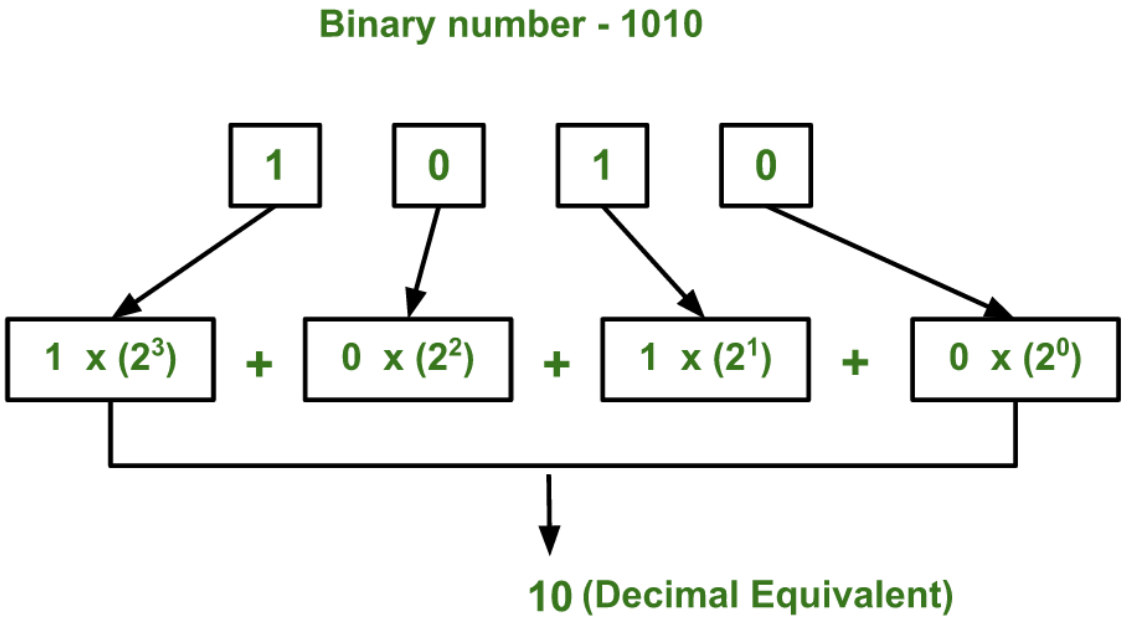
Constructing a DFA for binary number divisible by 2 or w mod 2 =0
In the binary system,any number divisible by 2 would end in 0.



Constructing a DFA for binary number divisible by 3 or w mod 3=0
The remainder on mod 3 is 0,1,2.
So in DFA for w mod 3=0 there will be 3 states .

Methods to convert binary to decimal :

One method:



Another method:

Example: 101

.

101

2*1=2 Multiply base i.e 2 with the most significant bit

2+0=2 Add the previous result to the next bit

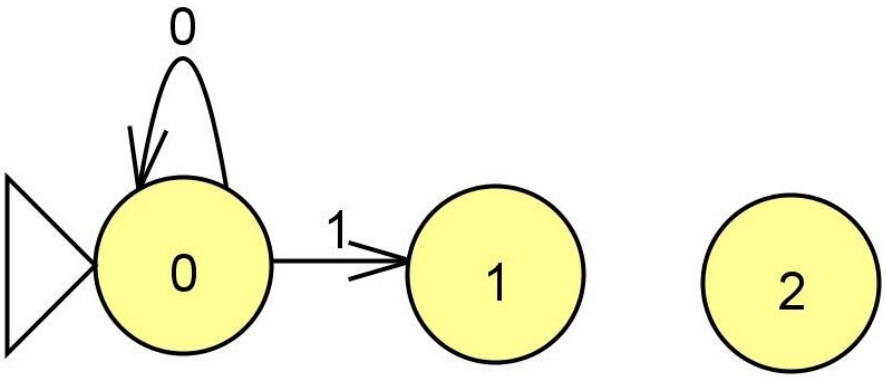
2*2=4 Multiply the previous result with base i.e 2

4+1=5 Add the next bit to the precious result.

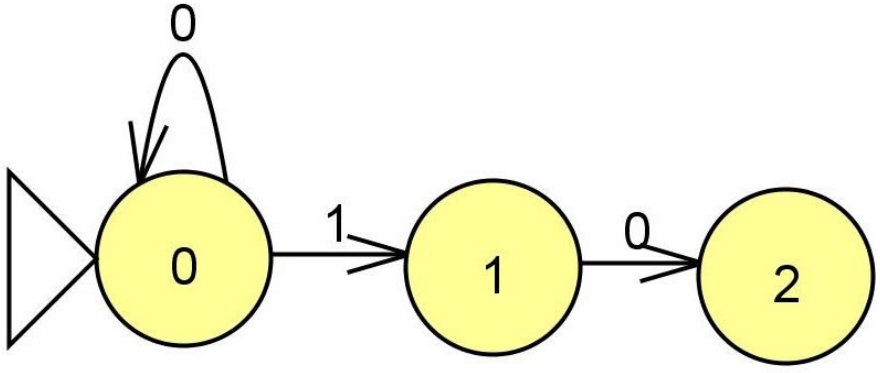
Example 17:

Constructing a DFA for binary number divisible by 3 or w mod 3 =0
The remainder on mod 3 is 0,1,2.

Let us construct DFA ,



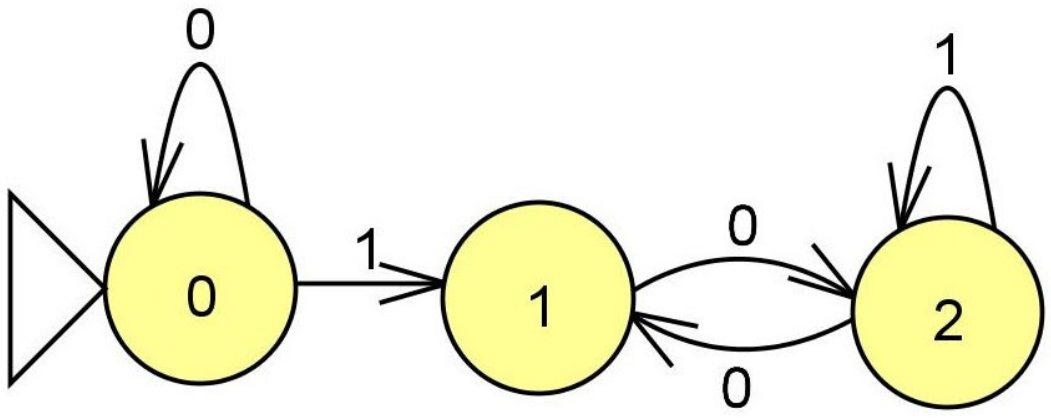
- The remainders on mod 3 0,1,2 are the states in DFA.
- If x is the input number to the DFA, x mod 3= either 0,1,2,it ends up in the state depending on what the remainder is. .
- 0 mod 3 = 0,hence a self loop on state 0.
- 1 mod 3 = 1 so we move to state 1.



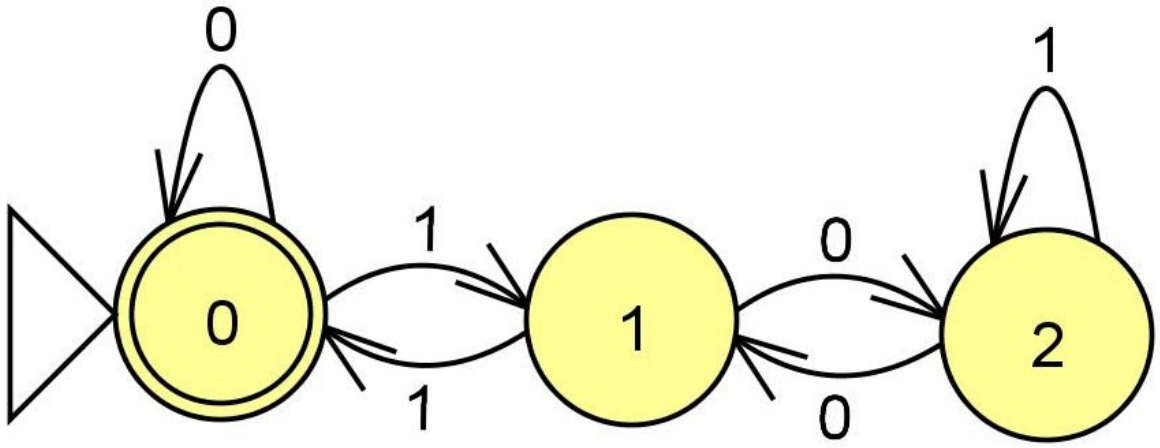
- 2 mod 3 =2
- The binary equivalent of 2 is 10 .

In state 2 if we get a '1' ,what do we do ?
For example ,if the input is 101, the decimal equivalent is 5.
Now, 5mod3=2.
So ,in state 2 if we see a '1' it will remain in state2.

In state 2 if we see a ‘0’,what do we do?
For example ,if the input is 100,the decimal equivalent is 4.
Now,4 mod 3=1
So ,in state 2 if we see symbol ’1’ we move to state 1.



In state 0 and state 2 we know what happens on 0,1 .
In state 1 we have to figure out what happens if we see symbol 1.
For example,if the input is 11.
The decimal equivalent of 11 is 3.
Now, 3 mod 3 = 0.
So in state 1 if we see symbol ‘1’ we move to state 0.



- This is the DFA for w mod 3=0