Theory of Computation
Chapter 1 : Regular Language

GATE Computer Science Lectures

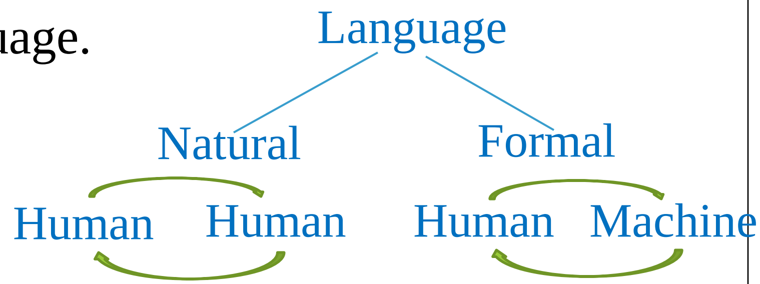
By

Monalisa Pradhan

- **Section 6: Theory of Computation($\cong$ 10mark)**
  Regular expressions and finite automata. Context-free grammars and push-down automata. Regular and context free languages, pumping lemma. Turing machines and undecidability.

- Chapter 1:Regular Language (RL,FA,RE ,Pumping lemma)
- Chapter 2: Context free Language (RG,CFG,CFL &PDA)
- Chapter 3: Recursive enumerable Language (CSL, LBA ,RS,RES,TM)
- Chapter 4: Undecidability

- **Theory of Computation:-**
- The study of mathematical representation of computing system and there capability.
- **Formal Language:-**
- The language which have proper alphabet, grammar & a model to recognize .
- In the formal language we need to give importance only for the formation of a string rather than meaning of string.
- Formal language are more accurate than natural language.
- It implement Artificial Intelligence.
- Natural: English,Hindi,Oriya,Marathi
- Machine: C,C++,Java, Python ..



- **Grammar:-**
- The collection of rules which are used to generate the string is called as grammar.
- Grammar is a generating device.
- **Automata:-**
- It is a mathematical representation of formal language.
- Automata is a recognizing device.

Chomsky Hierarchy:

Types	Language	Grammar	Automata
Type 0	Recursive Enumerable Language	Recursive Enumerable Grammar	Turing Machine
Type 1	Context sensitive language	Context sensitive Grammar	Linear bounded Automata
Type 2	Context free Language	Context free Grammar	Push Down Automata
Type 3	Regular Language	Regular Grammar	Finite Automata

- The number of language accepted by automata called accepting ,Expressive , recognizing power.

TM > LBA > PDA > FA

- E(FA)=1(RL)
- E(PDA)=2(RL,CFL)
- E(LBA)=3(RL,CFL,CSL)
- E(TM)=4(RL,CFL,CSL,REL)

Regular Language=>FA → DFA, NFA

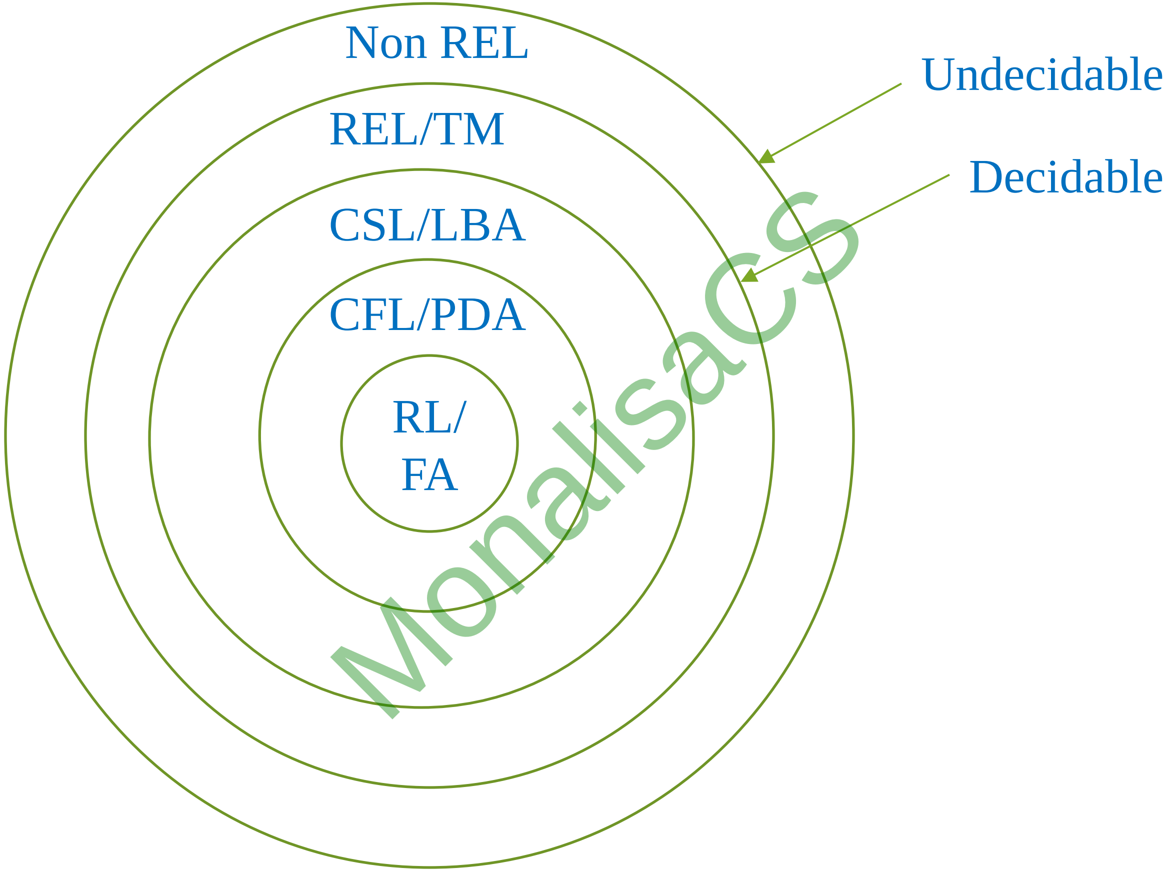
- A problem can be deterministic or nondeterministic .
- E(DFA)=E(NFA)
- DFA is more efficient than NFA.
- NFA design is easier than DFA.

- Every DFA is NFA and every NFA can be converted into DFA.
- $E(DPDA) \neq E(NPDA)$
- $DPDA \subset NPDA$
- DPDA is more efficient than NPDA.
- NPDA is more powerful than DPDA.
- All DPDA is NPDA but no algorithm exists to convert NPDA to DPDA.
- Restricted TM=LBA
- Properties Undecidable.
- $\in$ cann't accepted by LBA,It can be accepted by TM.
- E(DTM)=E(NTM)
- DTM is more efficient than NTM.
- Every DTM is NTM but every NTM can be converted to DTM.
- TM is a language generator or enumerator.
- FA=Static or limited amount memory.
- PDA=FA+1 stack.
- TM=FA +Tape =PDA+1 Stack
- =FA+2 stack=FA+3 stack=…….= FA+n stack

Context Free Language=>PDA → DPDA, NPDA

Recursive Enumerable Language=>TM → DTM, NTM

Relationship between formal languages



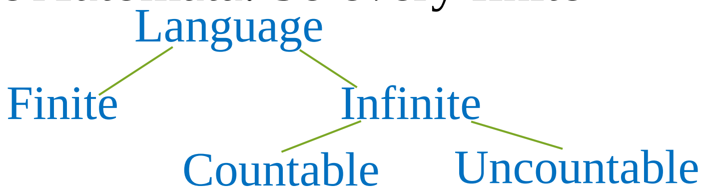
- **Theory of Computation**
- **Alphabets (Σ):**The nonempty finite set of symbol is called as an alphabet.
- Ex: Σ ={a,b,c…..z,0,1…..}, Σ ={0,1} or{a,b}
- **Strings(w):**The sequence of symbol from Σ is called as string .
- Ex: Σ ={0,1} ,w_1=10, w_2=010, w_2=1100,….
- **Length of string |w|**: The number of symbol in w called length of string.
- **Empty String** (∈):A string of length 0 or without symbol called empty string.
- w= ∈ , |w| = 0.
- w . ∈=w= ∈ . w
- **Substring:** Let u,w be the string over alphabet Σ.Then u is said to be substring of w if u is obtained from w.$|u| \leq |w|$.
- ∈ is substring of every w.
- Every string is substring to itself.
- Ex: w=TOC, substring={∈,T,O,C,TO,OC,TOC}
- Trivial/Improper substring:
- Substring w itself & ∈. Ex: {∈, TOC}
- Non trivial substring:
- Any string other than trivial(w, ∈) {T,O,C,TO,OC}

- W=GATE
- Substring of length=0 {∈} 1
- Substring of length=1 {G,A,T,E} 4
- Substring of length=2 {GA,AT,TE} 3
- Substring of length=3 {GAT,ATE} 2
- Substring of length=4 {GATE} 1
- If w is any string with distinct symbol and |w|=n then total number of substring
- $\sum_{i=1}^{n} i + 1 = \frac{n(n+1)}{2} + 1$
- Number of trivial substring=2
- Number of non trivial substring = $\frac{n(n+1)}{2} - 1$.
- **Prefix**: The sequence of symbol from starting is called as prefix. Ex:{∈,G,GA,GAT,GATE}
- **Suffix**: The sequence of symbol from ending is called as suffix.
- Ex:{∈,GATE,ATE,TE,E}
- Number of prefix=number of suffix=n+1.
- Trivial substring is common for both prefix as well as suffix.
- Every prefix or suffix is substring but every substring need not be prefix or suffix.

- **Power of Alphabet:**
- If Σ is any alphabet then Σ^k is set of all strings of length k.
- $\Sigma = \{0,1\}$
- $\Sigma^0 = \{\in\}$
- $\Sigma^1 = \{0,1\}$
- $\Sigma^2 = \{00,01,10,11\}$
- $\Sigma^3 = \{000,001,010,011,100,101,110,111\}$
- ………..
- Σ^+ (+ve closer)= $\Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \Sigma^4$……..
- Ex : $\Sigma^+ = \{0,1,00,01,10,11, 000,001,010,011,100,101,110,111,0000\ldots\}$
- Σ^* (Kleene closer)=$\Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \Sigma^4$……..= $\in \cup \Sigma^+$
- Ex: $\Sigma^* = \{\in, 0,1,00,01,10,11, 000,001,010,011,100,101,110\ldots\}$
- **Language** :Collection of string from Σ is called language.
- **Representation of language**:language can be represented in 3 way.
- 1.Enumuration /Listing all string .Ex:{00,01,10,11}.
- 2.Set builder form/Infinite form.Ex:$L=\{(01)^n \mid n \geq 1\}$.
- 3.Statement Ex:{all string of 0's and 1's ending with 0}
- Σ^* is called universal language.

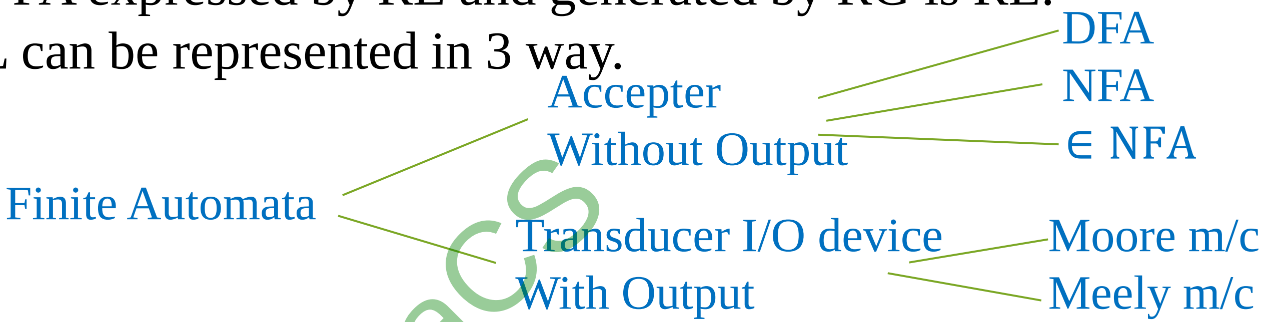
- If L is any language over Σ then $L \subseteq \Sigma^*$.
- **Empty Language (ϕ)**: The language which doesn't contain any string even ∈ is called as empty language.
- $L=\phi \Rightarrow |L|=|\phi|=0$
- |w|= Number of symbols,|L|=Number of strings.
- **Non Empty Language:** The language which contain at least one string even ∈ is called as non empty language.
- $L= \in \Rightarrow |L|=1$
- **Finite Language**: The language which contain finite number of string but length of every string is finite called finite language.
- Ex: $L_1=\{00,10\}, L_2=\{0^n 1^n \mid 1 \leq n \leq 100\}$
- **Infinite Language**: The language which contains infinite number of string but length of every string is finite called infinite language.
- Ex: $L_1 =\{a^n b^n \mid n \geq 1\}$, L_2={All string of a's & b's starting with a}
- Empty language ϕ is always finite language.
- $L.\phi= \phi$, $L . \in= L$, $\phi . \in= \phi$, $\phi \neq \in$.

- Every finite language can be accepted by some finite Automata. So every finite language is Regular language.
- Finite language=RL⊆ CFL ⊆CSL ⊆REL.



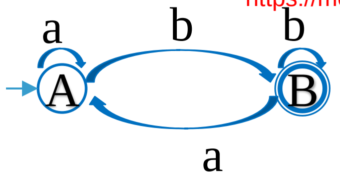
- **Countable Set** is a set having cardinality same as **N** the set of natural numbers . A countable set is the one which is listable. Cardinality of a countable set can be a finite number.
- **Uncountable Sets:**A set such that its elements cannot be listed.
- Every subset of countable set is countable.
- Power set of countable set is uncountable.
- Σ = Alphabet
- Σ^* =set of all strings, Countable.
- 2^{Σ^*}=Set of all language, Uncountable.
- Set of all RL,CFL,CSL,REL countable.
- REL =Decidable =Countable
- Non REL=Undecidable=Uncountable

- **Regular Language:**
- If a language accepted by FA expressed by RE and generated by RG is RL.
- Representation of RL: RL can be represented in 3 way.
- 1.Finite Automata
- 2.Regular Expression
- 3.Regular Grammar



- **DFA:** The mathematical representation of RL is called as Finite Automata.
- A DFA can be represented by a 5-tuple (Q, Σ, δ, q_0, F) where
- ***Q*** *is a finite set of states.*
- *Σ is a finite set of symbols called the alphabet.*
- ***δ*** *is the transition function where* $\delta: Q \times \Sigma \rightarrow Q$
- $\boldsymbol{q_0}$ *is the initial state from where any input is processed* ($q_0 \in Q$).
- ***F*** *is a set of final state/states of Q* ($F \subseteq Q$).
- In DFA, for each input symbol, one can determine the state to which the machine will move.
- Hence, it is called Deterministic Automaton.

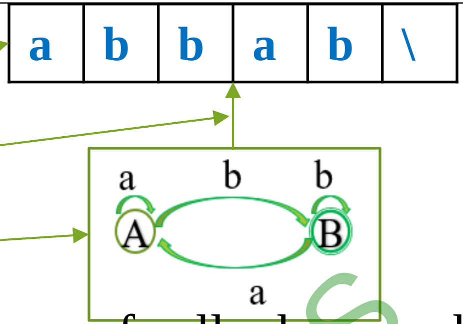
Ex: $\Sigma = \{a, b\}$, L=accepts all strings ending with b.



- L={b,ab,bb,aab,abb,bab,bbb,………}
- Q={A,B} , $\Sigma = \{a, b\}$, q_0 =A , F=B
- δ(A,a)=A, δ(A,b)=B, δ(B,a)=A, δ(B,b)=B
- By default FA is DFA.
- DFA response both valid as well as Invalid string.
- DFA is complete if a transition function is defined for each and every input symbol at each and every state.
- Number of transition at a state=$|\Sigma|$=number of input symbol
- Every string has only one transition path i.e. a string can be process only one way.
- Sequence of transition is called as transition path.
- **Instantaneous description (ID)**: ID describes the movement of FA.
- δ(Q(current state), Σ(current I/P symbol))=Q(new state)
- FA can be represented in two way
- 1.Transition diagram
- 2.Transition Table

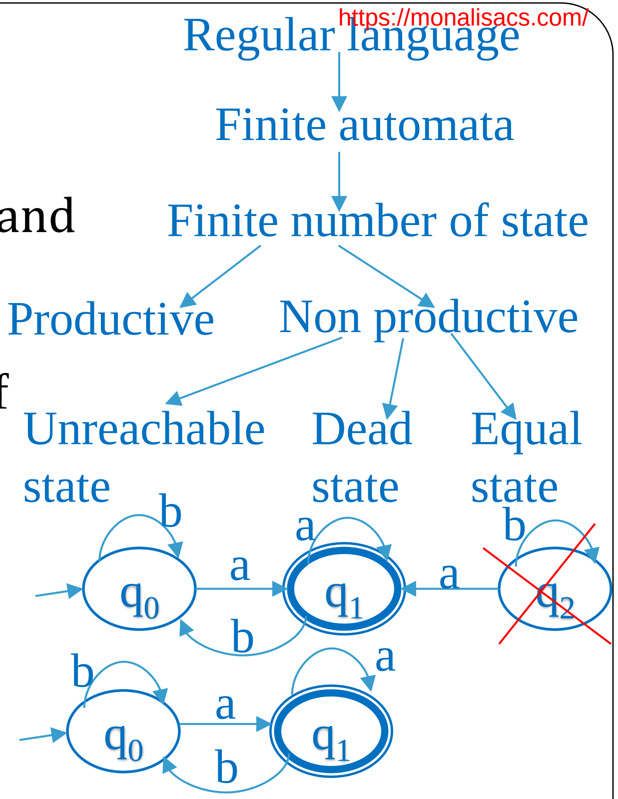
	a	b
A	A	B
B	A	B

- **Block diagram of FA**
- It consist of 3 parts
- 1.Input Tape
- 2.Tape header
- 3.Finite control unit

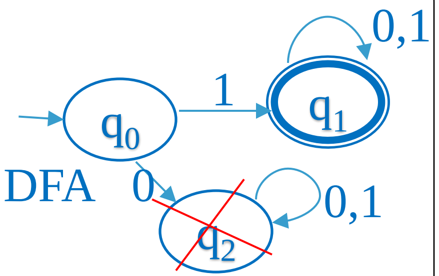


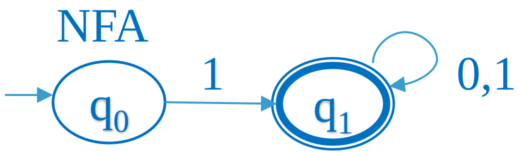
- **Input Tape:** It is divided into finite no of cell where each cell can hold only one input symbol
- At any point time the input tape contain only one input string.
- **Tape header**: It scan input symbol from input tape starting from leftmost end.
- Header can scan only one symbol at any point of time.
- After reading input symbol the header moves to exactly one cell ahead toward right.
- The header movement is left to right unidirectional.
- **FCU**:it just like CPU where it implement business logic.
- FCU take care of movement of FA inorder to process input string .
- **Drawback of FA :**
- 1.FA has static & limited amount of memory.
- 2.FA depends on state for everything.
- i.e the symbols are processed at the state but can't be store.

- 3.FA can remember the current processing symbol but can't remember previous symbol.
- **Acceptance of FA**:Let x is any string from alphabet Σ.
- If there exist a transition path which start at initial state and end with any one final state then string x is accepted by FA.
- FA accept ∈ if initial state is final state.L= ∈
- **Productive state**: The state which involves in the process of valid input string called productive state.
- **Non Productive state:** the state which not involve in the process of any valid input string is called non productive state.
- Or ,The state whose presence or absence will not affect the language called non productive sate.
- **Unreachable state:**The state that can't reach from initial state is called unreachable state.
- If we remove unreachable state from DFA then there won't be any change in language as well as structure of FA.
- **Ex:** q_2 is unreachable we can remove this.

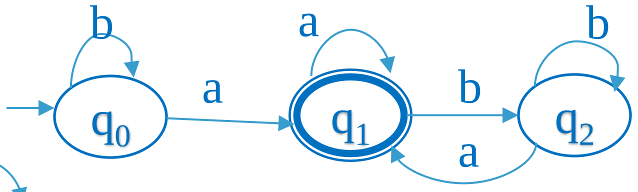


- **Dead state:** The non final state from which we can't reach final state & goes to itself for every input alphabet is called dead state.
- If we remove the dead state from DFA then there won't be any change in language but there will be change in structure.
- The DFA become NFA which is incomplete in structure.
- Ex: L=set of string start with 1, $\Sigma = \{0,1\}$
- q_2 is dead state. After removal DFA becomes NFA.

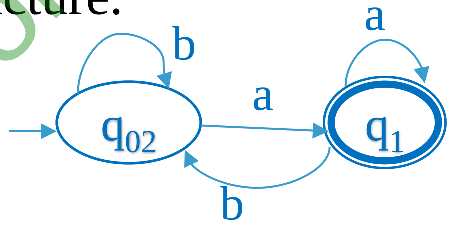




- **Equal state:** Two state p,q is said to be equal if $\delta(p,x) = \delta(q,x)\ \forall x \in \Sigma^*$
- If we remove equal state from DFA then there won't be any change in language and structure.

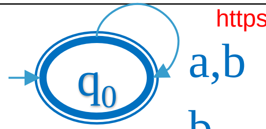


- Ex: q_0 and q_2 are equal state.
- $\delta(q_0,a)= q_1$(final state)$= \delta(q_2,a)$
- $\delta(q_0,b)=$ non final state$= \delta(q_2,b)$
- $\delta(q_0,aba)= q_1$(final state)$= \delta(q_2,aba)$
- We can merge q_0, q_2 there will be no change

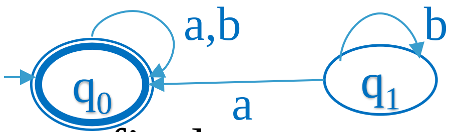


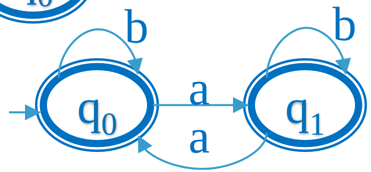
$\Sigma = \{a,b\}$

- The DFA where every state is final accept universal language Σ^*.

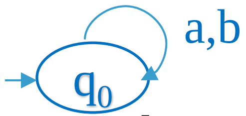


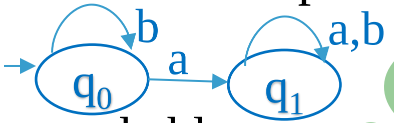
- The DFA where every non final state is unreachable accept universal language Σ^*.

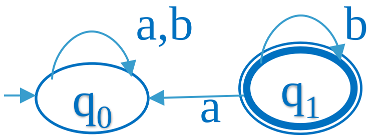




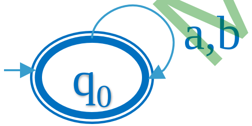
- The FA where every state is non final accept empty language Φ.





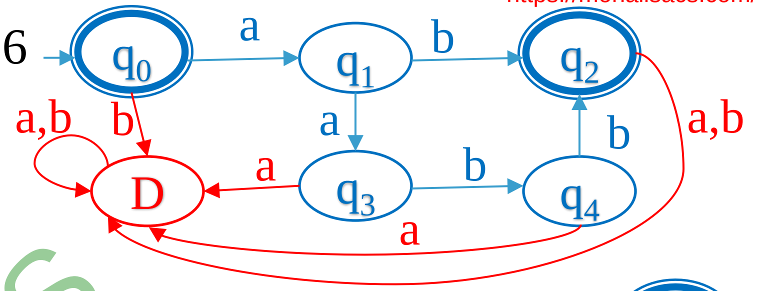


- The DFA where final state is unreachable accept empty language Φ.
- Every FA accept only one language but a language can be accepted by more than one DFA.DFA is not unique.
- Every Regular language accepted by only one minimal DFA so minimal DFA is unique.
- The number of state in minimal DFA to accept universal language Σ^* is one.

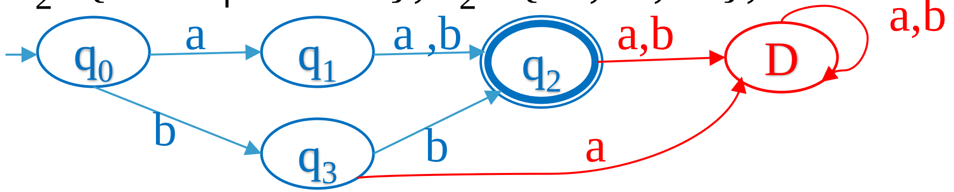


- Number of state in minimal DFA to accept empty language Φ is 2.

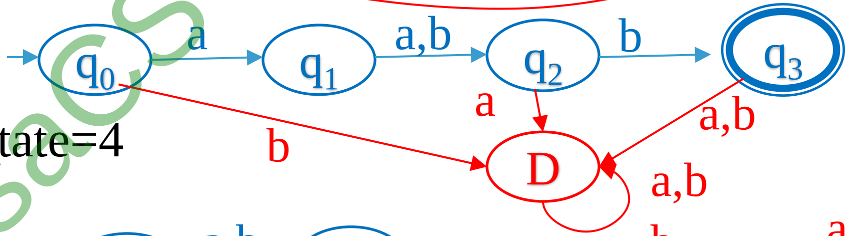
- **Construct DFA for finite language Σ = {a, b}:**
- $L_1=\{a^n b^n \mid 0\le n \le 2\}$, $L_1=\{\epsilon, ab, aabb\}$, Min state =6



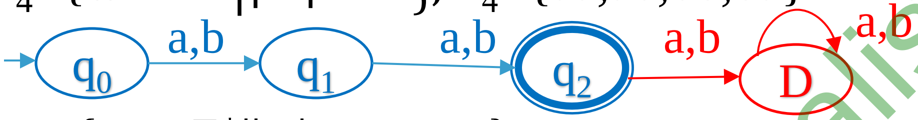
- $L_2=\{a^m b^n \mid m+n=2\}$, $L_2=\{aa,ab,bb\}$, min state=5



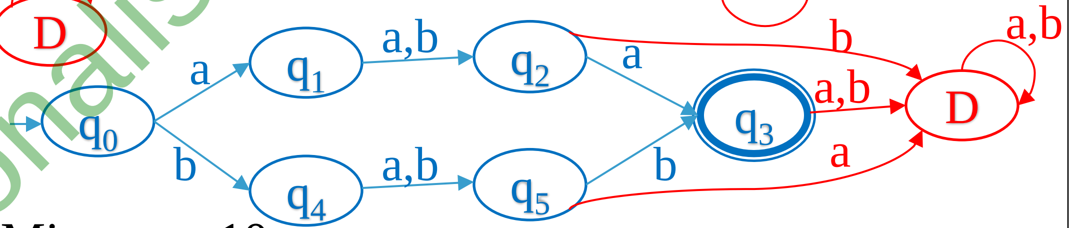
- $L_3=\{a^m b^n \mid m.n=2\}$, $L_3=\{aab,abb\}$, Min state=5



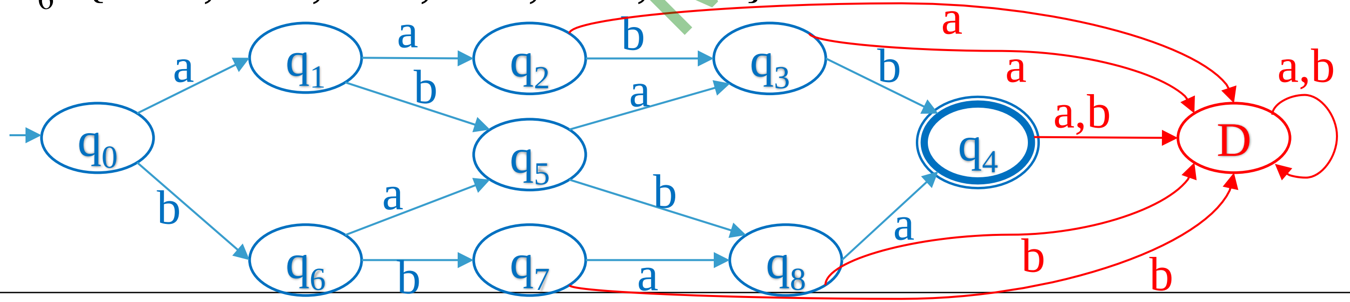
- $L_4=\{w\in \Sigma^* \mid |w| = 2\}$, $L_4=\{aa,ab,bb,ba\}$ Min state=4



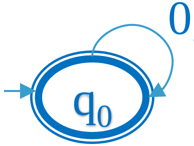
- $L_5=\{w\in \Sigma^* \mid |w|=3, w=w^r\}$
- $L_5=\{aaa,aba,bab,bbb\}$, Min state=7

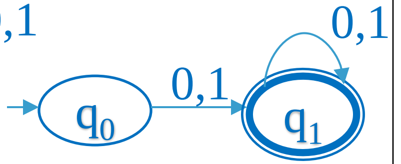


- $L_6=\{w\in \Sigma^* \mid |w| = 4, |w|_a = |w|_b\}$
- $L_6=\{aabb,abab,abba,bbaa,baba,baab\}$ Min state=10

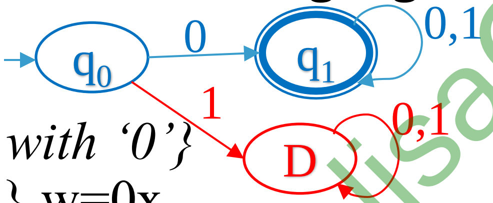


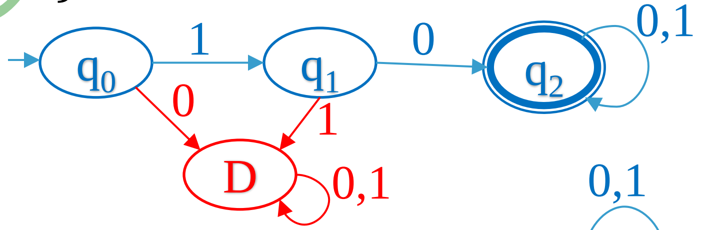
- Every finite language is accepted by some DFA=>Regular language.
- Every path from initial state to final state=valid string.
- Number of string accepted by DFA=Number of path from initial state to final state.
- DFA accept finite language if it is free from cycle & loop.
- DFA accept infinite language if it contain cycle & loop.



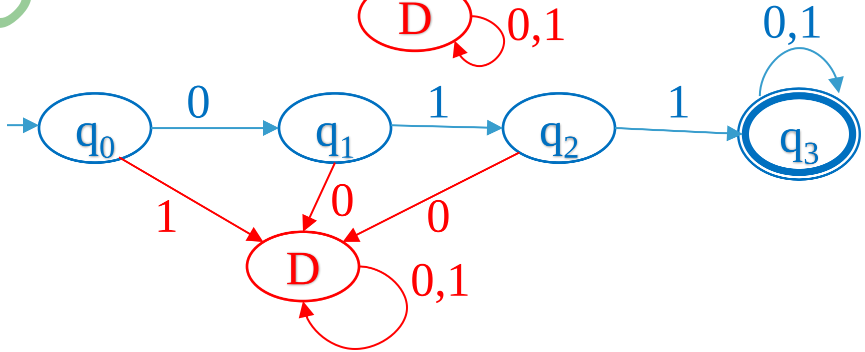


- **Construction of DFA for Infinite language $\Sigma = \{0, 1\}$:**
- $L_{7.1}=\Sigma^*$, Min. state=1
- $L_{7.2}=\Sigma^+$, Min. state=2
- *$L_{8.1}$={Every string start with '0'}*
- {0,00,01,000,001,0^+…..},w=0x



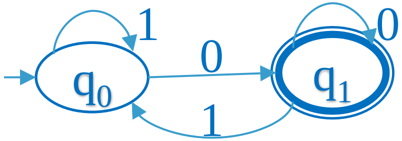


- *$L_{8.2}$={Every string start with '10'}*
- {10,100,101,1000,1001…….},w=10x
- *$L_{8.3}$={Every string start with '011'}*
- {011,0110,0111,01110011,…. },w=011x

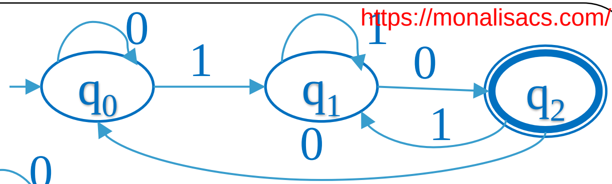


- If in L every string start with s or prefix s ,i.e w=sx & |s|=n
- Then number of state in Minimal DFA to accept L =n+2

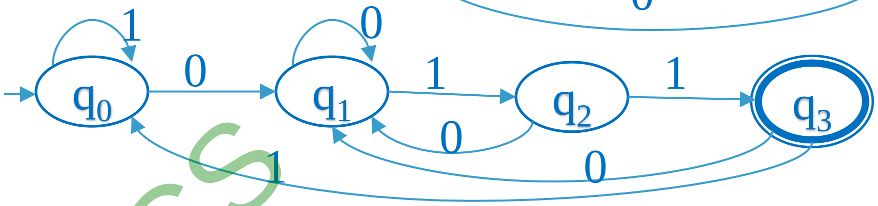
- $L_{9.1}$=*{Every string end with '0'}*
- {0,10,00,000,100, 0^+,…},w=x0



- $L_{9.2}$=*{Every string end with '10'}*
- {10,010,110,0010,1010,…},w=x10

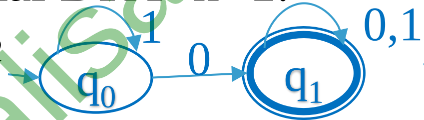


- $L_{9.3}$=*{Every string end with '011'}*
- {011,0011,1011,10011…},w=x011

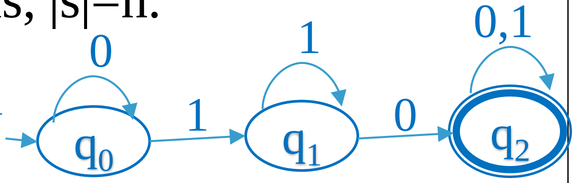


- If in L every string ends with substring s or suffix s i.e w=xs, |s|=n.
- Then number of state required in minimal DFA=n+1.

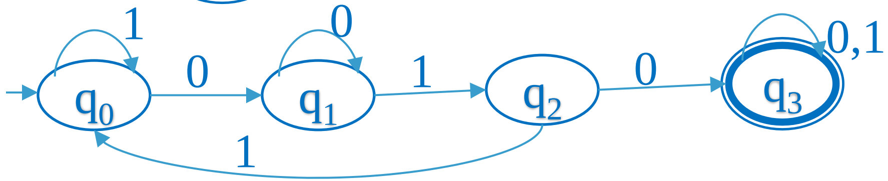
- $L_{10.1}$=*{Every string contain substring '0'}*
- {0,10,00,000,100,110, 0^+,…},w=x0x



- $L_{10.2}$=*{Every string contain substring '10'}*
- {10,100,010,0100,110,…},w=x10x

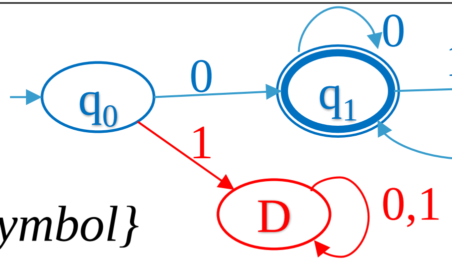


- $L_{10.3}$=*{Every string contain substring '010'}*
- {010,0100,0101,0100,…},w=x010x

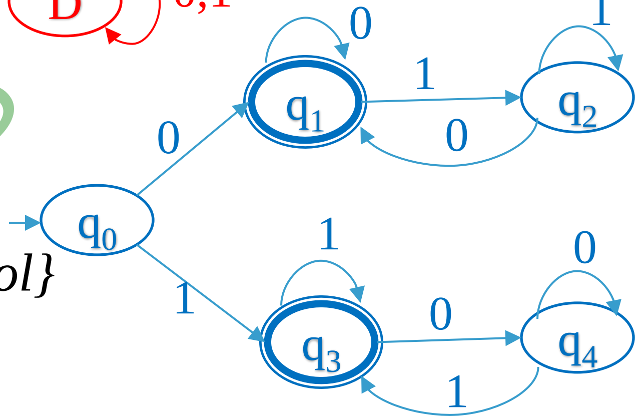


- If Language contain substring s, i.e w=xsx , |s|=n
- Then number of state required in Minimal DFA = n+1

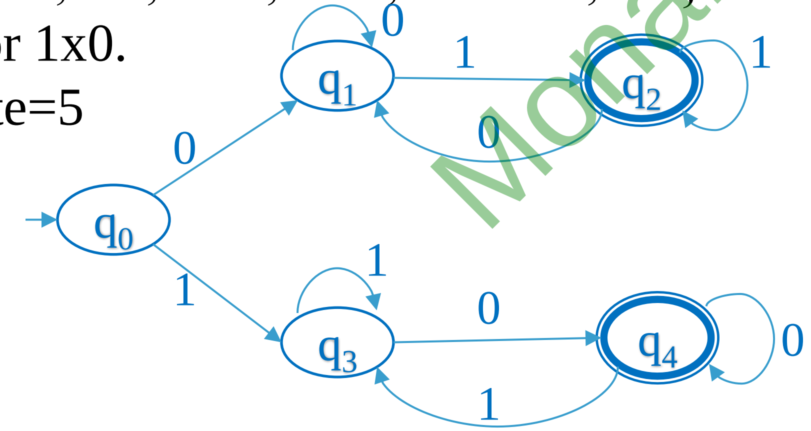
- *$L_{11.1}$={Every string start and end with '0'}*
- {0,00,010,0110,0100,0^+,…},w=0x0
- Min. state=4



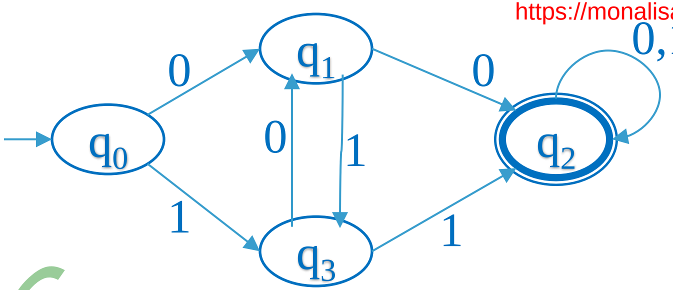
- *$L_{11.2}$={Every string start and end with same symbol}*
- {0,1,00,11,0^+, 1^+,010,101,0000,0110,10101…}
- w=0x0 or 1x1.
- Min. state=5



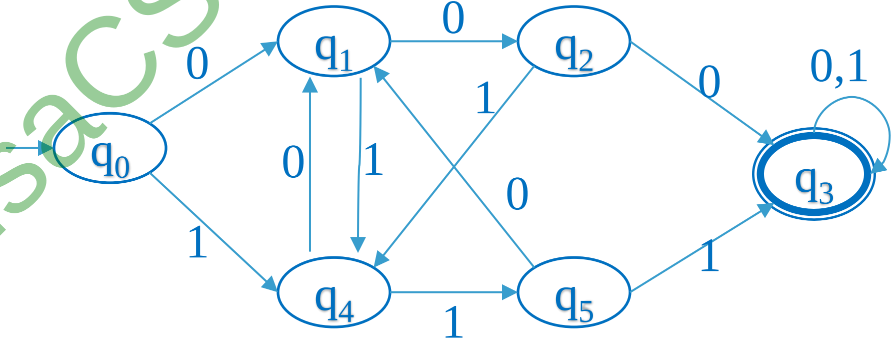
- *$L_{11.3}$={Every string start and end with different symbol}*
- {01,10,001,110,1000,0101,1001010,……}
- w=0x1 or 1x0.
- Min. state=5



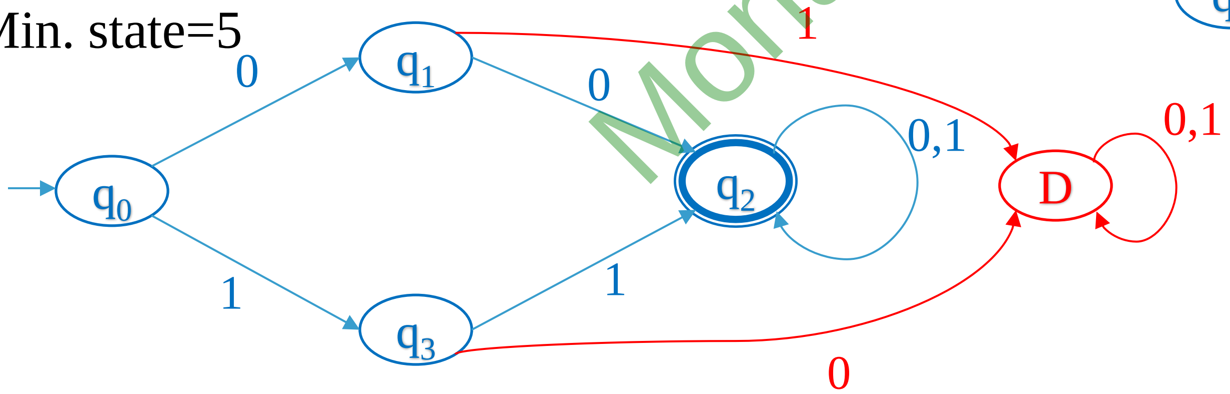
- *$L_{12.1}$={Every string contain substring DIBIT}*
- {00,11,011,100,0110,1001,…..}
- w=x00x or x11x
- Min. state=4



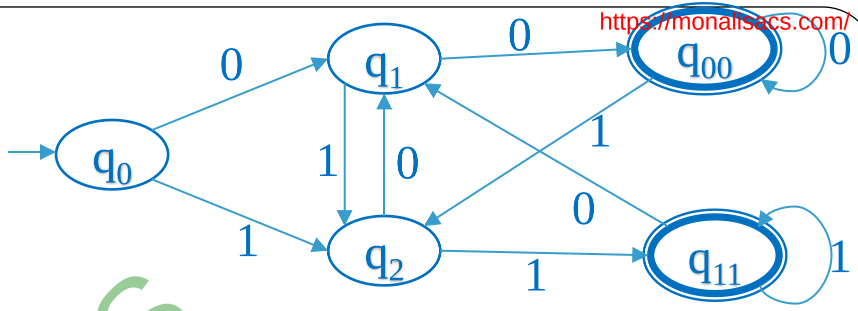
- *$L_{12.2}$={Every string contain substring TRIBIT}*
- {000,111,0111,1000,01110,10001,…..}
- w=x000x or x111x
- Min. state=6



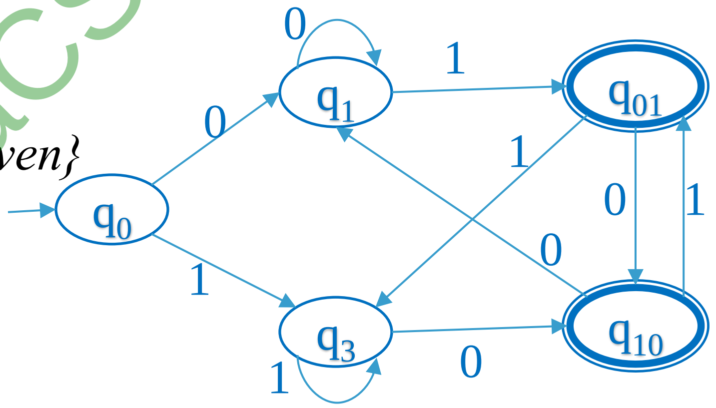
- *$L_{13.1}$={Every string start with '00' or '11'}*
- {00,11,001,1101,00110,11001,…..}
- w=00x or 11x
- Min. state=5



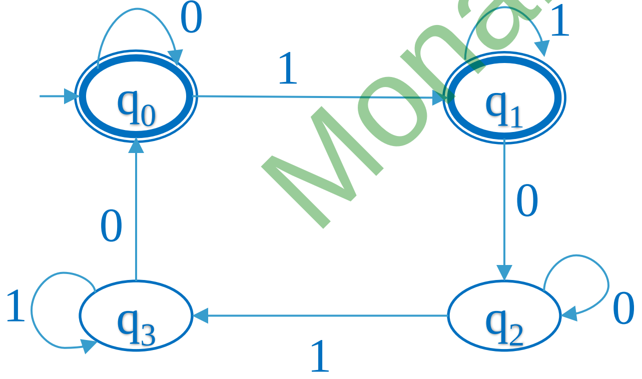
- *$L_{13.2}$={Every string end with '00' or '11'}*
- {00,11,100,011,1100,10011,000*,111*..}
- w=x00 or x11
- Min. state=5



- *$L_{13.3}$={Every string end with '01' or '10'}*
- {01,10,101,010,1101,0010,10110,01101….}
- w=x01 or x10
- Min. state=5

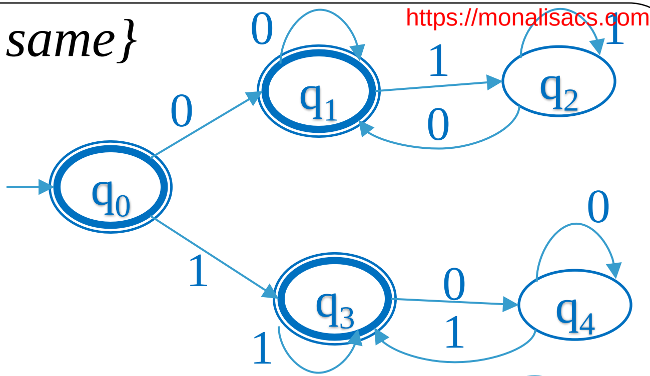


- *L_{14}={Number of occurrence of substring '10' is even}*
- {∈ ,0^* ,1^* ,01,11,1010,101110,1001011010,…..}
- Min. state=4

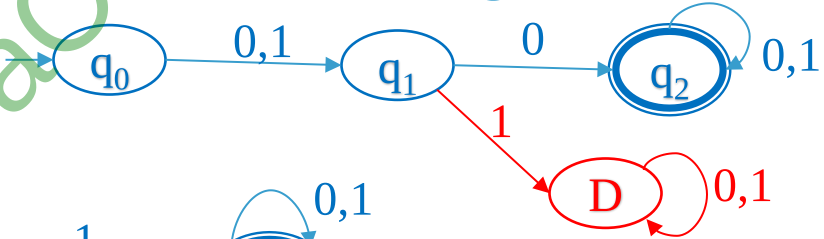


- *If L_{14} ={Number of occurrence of substring 10 is odd}*
- Then q_2 ,q_3 will be final state.

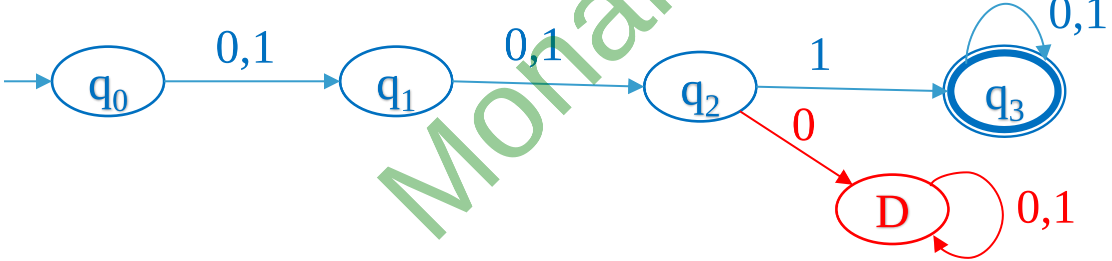
- *L_{15}={Number of occurrence of substring 10 and 01 is same}*
- {∈ ,0^* ,1^* ,101,010,1001,0110,110001,0100110,….}
- Min state=5



- *$L_{16.1}$={The 2^{nd} symbol from left end is 0}*
- {00,10,100,101,001,000,101110,….}
- w=_0x

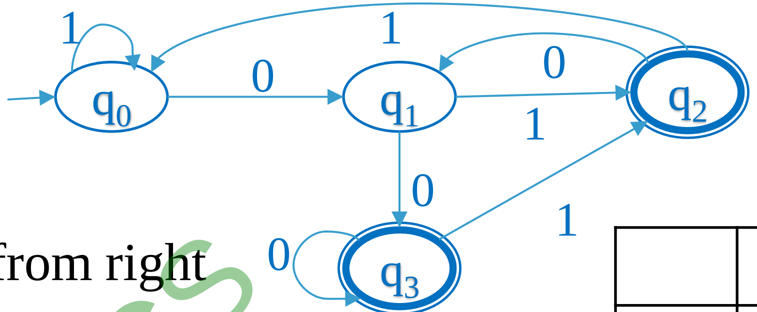


- *$L_{16.2}$={The 3^{rd} symbol from left end is 1}*
- {001,101,111,011,0010,1011,1011001,….}
- w=_ _1x



- The minimal DFA that accept n^{th} symbol from left end is fixed contain n+2 state.

- *$L_{17.1}$={The 2nd symbol from right end is '0'}*
- {00,01,100,101,001,01000,01001,….}
- w=x0_



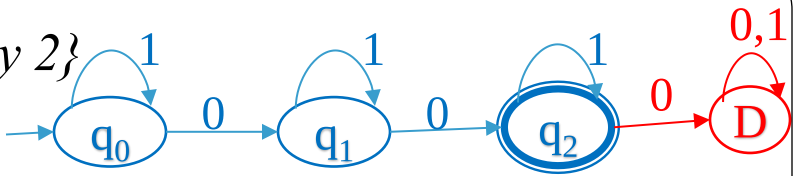
- Shortcut : fix(0)=odd, not fix= even
- The minimal DFA that accept nth symbol from right end is fixed contain 2^n state.
- Number of final state= 2^{n-1}

	0	1
→q_0	q_1	q_0
q_1	q_3	q_2
*q_2	q_1	q_0
*q_3	q_3	q_2

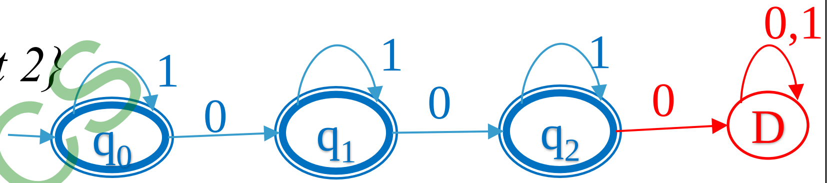
- *$L_{17.2}$={The 3rd symbol from right end is '1'}*
- {100,101,110,111,0101,10100,01101,….}
- w=x1_ _
- Number of state=2^3 =8,
- Number of final state= 2^2=4

	0	1
→q_0	q_0	q_1
q_1	q_2	q_3
q_2	q_4	q_5
q_3	q_6	q_7
*q_4	q_0	q_1
*q_5	q_2	q_3
*q_6	q_4	q_5
*q_7	q_6	q_7

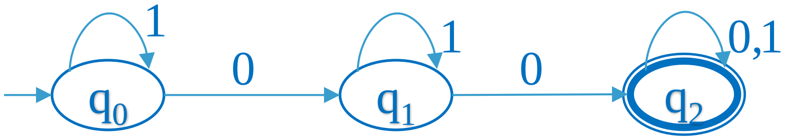
- $\Sigma = \{0,1\}$
- $L_{18.1}$=*{Number of '0' present in a string is exactly 2}*
- {00,010,100,001,1001,0110,1101011 ….}
- $|w|_0=2$



- $L_{18.2}$=*{Number of '0' present in a string is at most 2}*
- {∈ ,0,00,1*,101,11001,01011,110101, ….}
- $|w|_0 \leq 2$

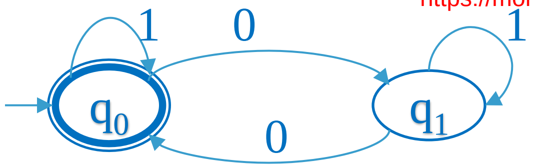


- If any alphabet present in a string is exactly or at most n then number of state in minimal DFA =n+2 .
- $L_{18.3}$=*{Number of '0' present in a string is at least 2}*
- {00,010,000,010010,110101,….}
- $|w|_0 \geq 2$

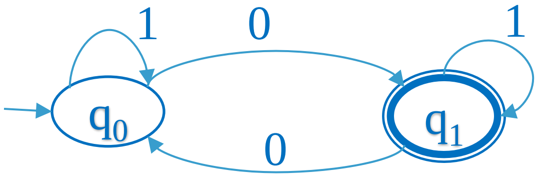


- Minimum state =3.
- If at least n then number of state=n+1

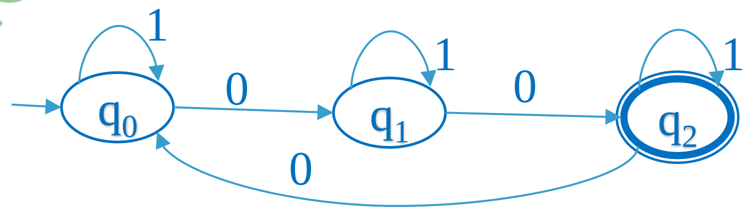
- $L_{18.4}$=*{Number of '0' present in a string is even}*
- {∈,1*,00,010,100,001,100101011,…….}
- $|w|_0$= even = 0,2,4,6,8,10,….



- $L_{18.5}$=*{Number of '0' present in a string is odd}*
- {0,01,101,110,10100,001011,…….}
- $|w|_0$= odd = 1,3,5,7,9,11….

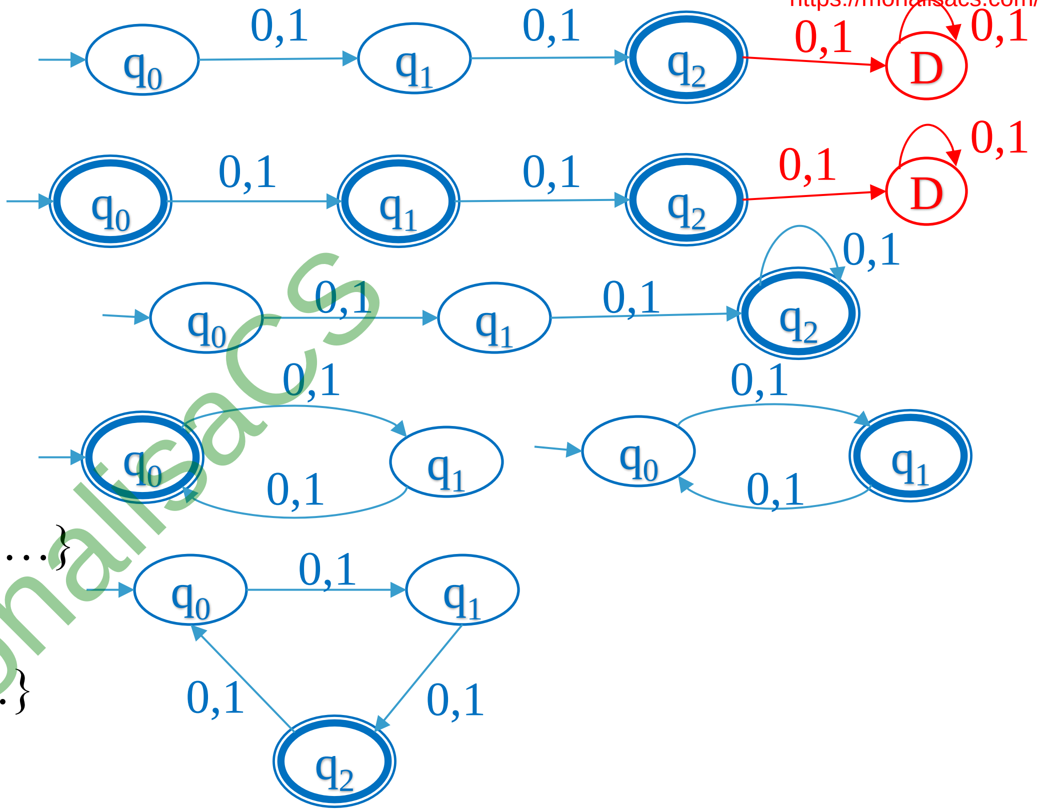


- $L_{18.6}$=*{Number of '0' in a string is 2 (mod 3)}*
- {00,1010,0011,1001,010110010,…….}
- $|w|_0$ mod 3 =2 [2,5,8,11….]

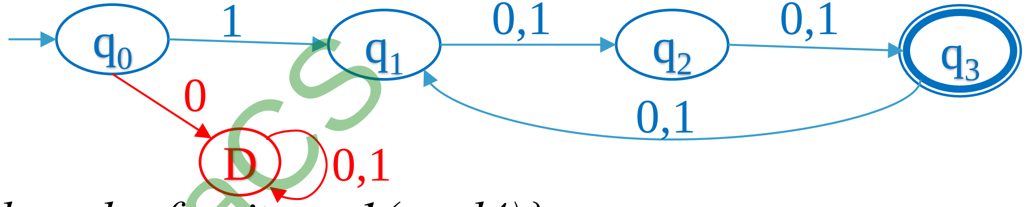


- If $|w|_0$ =r (mod n)
- Then number of state in minimal DFA=n

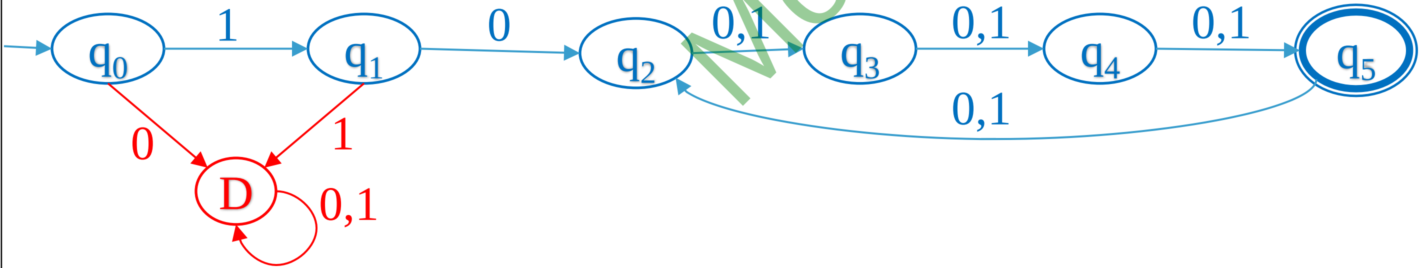
- $\Sigma = \{0,1\}$
- $L_{19.1}$=*{length of string is exactly 2}*
- {00,01,10,11}
- |w| = 2
- $L_{19.2}$=*{length of string is at most 2}*
- {∈,0,1,00,01,10,11}
- |w| ≤2
- $L_{19.3}$=*{length of string is at least 2}*
- {00,01,10,11,010,1100,10100,…}
- |w| ≥2
- $L_{19.4}$=*{length of string is even}*
- {∈,00,01,10,11,0000,0011,0110,1111,…}
- |w|=even [0,2,4,6,….]
- $L_{19.5}$=*{length of string is odd}*
- {0,1,000,010,011,110,01100,11010,…}
- |w|=odd [1,3,5,7,….]
- $L_{19.6}$=*{length of string is 2(mod 3)}*
- {00,01,10,11,00000,01101,11001,10001,10011011…}
- |w| mod 3 =2 [2,5,8,11….]
- Length exactly and at most n →n+2 state ,at least n→n+1 state
- |w| =r (mod n) →n state require in minimal DFA.



- $\Sigma = \{0,1\}$
- $L_{20.1}$ =*{Every string start with '1' & length of string is divisible by 3}*
- {100, 101,110, 111,100000,100101,110010100,110111001,…..}
- w=1x & |w|=0(mod 3) [3,6,9,12….]
- Number of state in minimal DFA
- = 2+3=5

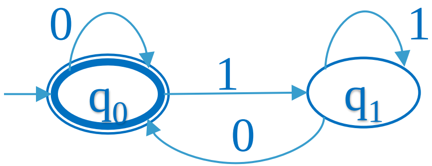


- $L_{20.2}$ =*{Every string start with '10' & length of string =1(mod4)}*
- {10000,10110,10111,10101,101101000,…….}
- w=10x & |w|=1(mod 4) [5,9,13,17,21….]

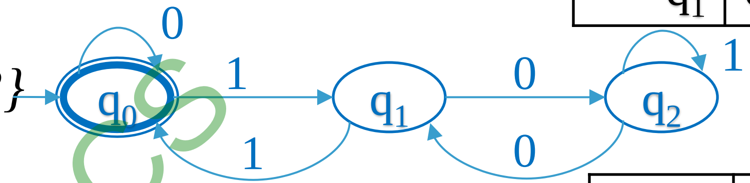


- Number of state in minimal DFA = 3+4=7

- $\Sigma = \{0,1\}$
- $L_{21.1}$ ={*Decimal equivalent is divisible by 2*}
- {∈,0*,10,100,110,1000,1010,1100,…….}
- $(w)_{10}$ mod 2=0 [0,2,4,6,8,10,…..]
- $L_{21.2}$ ={*Decimal equivalent is divisible by 3*}
- {∈,0*,11,110,1001,1100,1111,…….}
- $(w)_{10}$ mod 3=0 [0,3,6,9,12,15…..]
- $L_{21.3}$ ={*Decimal equivalent is divisible by 4*}
- {∈,0*,100,1000,1100,10000,…….}
- $(w)_{10}$ mod 4=0 [0,4,8,12,16…..]

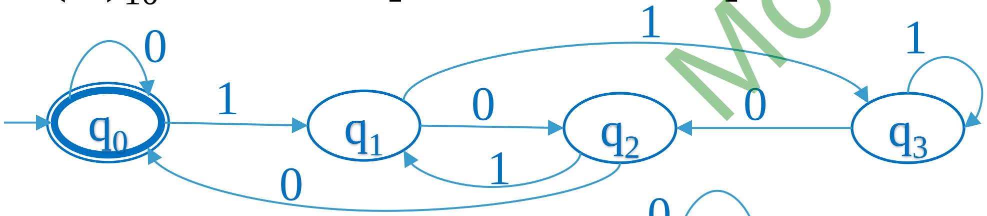


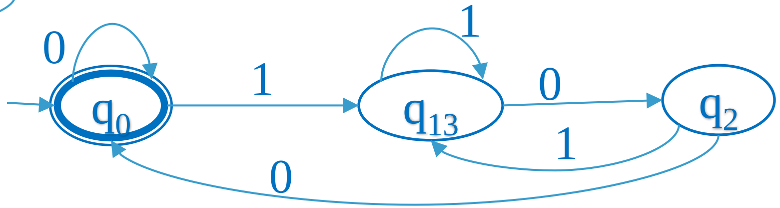
	0	1
→*q_0	q_0	q_1
q_1	q_0	q_1



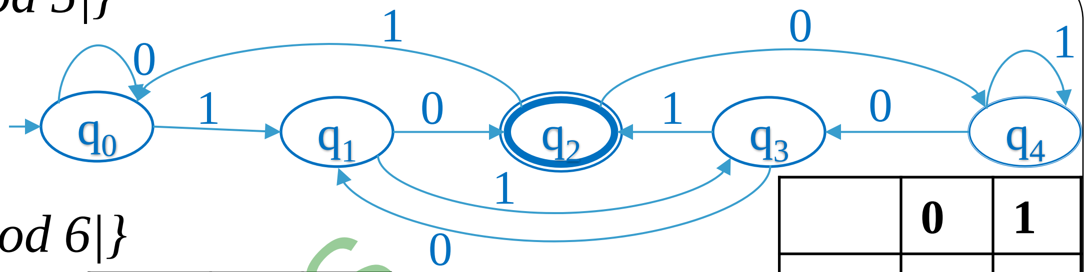
	0	1
→*q_0	q_0	q_1
q_1	q_2	q_0
q_2	q_1	q_2

	0	1
→*q_0	q_0	q_1
q_1	q_2	q_3
q_2	q_0	q_1
q_3	q_2	q_3





- $L_{21.4}$ ={*Decimal equivalent is 2 |mod 5|*}
- {10,111,1100,10001,10110,…….}
- $(w)_{10}$ mod 5=2 [2,7,12,17,22…..]

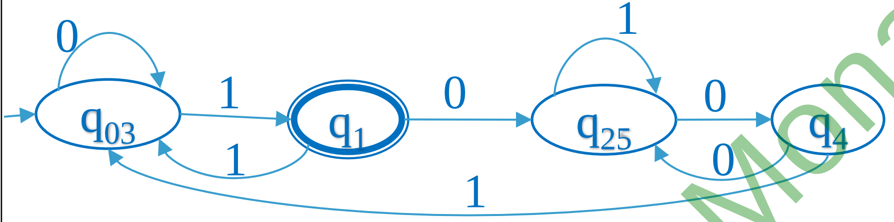


	0	1
→q_0	q_0	q_1
q_1	q_2	q_3
*q_2	q_4	q_0
q_3	q_1	q_2
q_4	q_3	q_4

- $L_{21.5}$ ={*Decimal equivalent is 1 |mod 6|*}
- {1,111,1101,10011,11001,…….}
- $(w)_{10}$ mod 6=1 [1,7,13,19,25…..]
- $q_0 = q_3$, $q_2 = q_5$ merge equal states

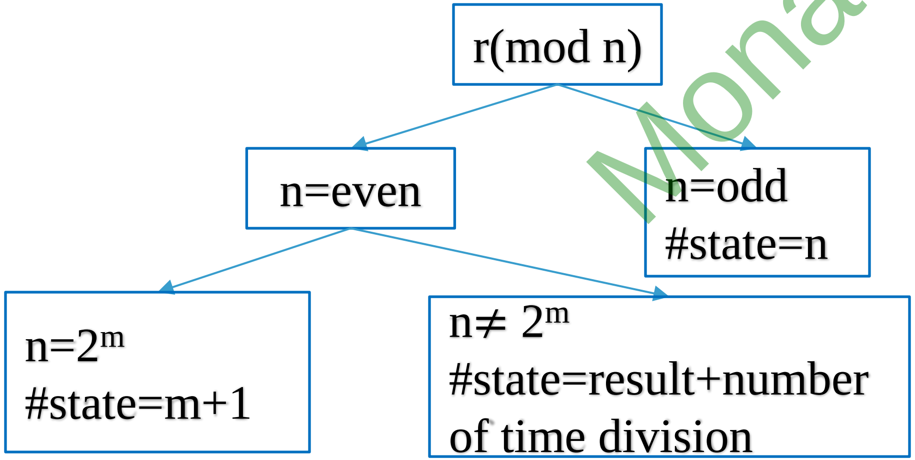
	0	1
→q_0	q_0	q_1
*q_1	q_2	q_3
q_2	q_4	q_5
q_3	q_0	q_1
q_4	q_2	q_3
q_5	q_4	q_5

	0	1
→q_{03}	q_{03}	q_1
*q_1	q_{25}	q_{03}
q_{25}	q_4	q_{25}
q_4	q_{25}	q_{03}

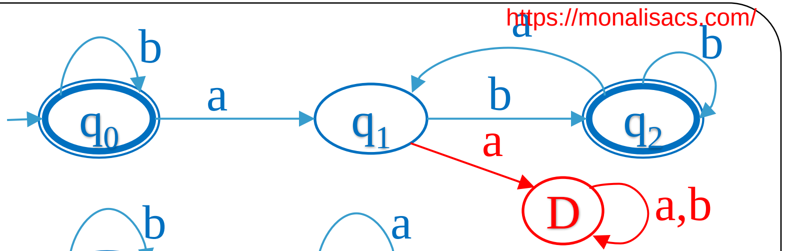


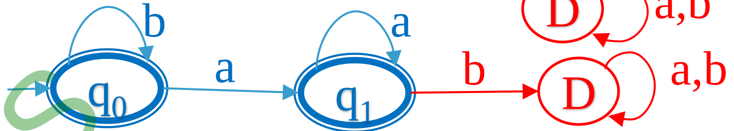
- If *L* =*Decimal equivalent is r |mod n|*}
- Then number of state require in minimal DFA=n [n=odd]
- Then number of state require in minimal DFA ≤n [n=even]

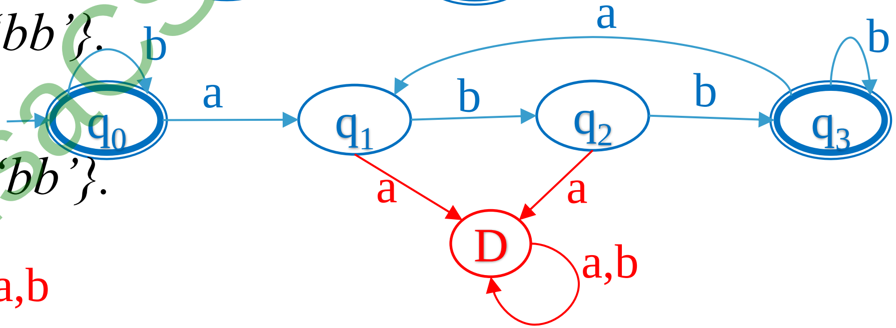
- Minimum number of state require when decimal number
- divisible by 8 => 2^3= 1+3=4 [result + number of time division to get odd result]
- 3 (mod 12) =>12/2=6/2=3+2=5 [3 result+ 2 time division to get 3]
- 5(mod 14) =>14/2=7+1=8 [7 result+1 time division to get 3]
- 10(mod 16) =>16=2^4=4+1=5 [1 result+4 time division to get 1]
- 13(mod 20) => 20/2=10/2=5+2=7 [5 result +2 time division]
- 10 (mod 32) =>32=2^5=5+1=6 [1 result+5 time division]
- 19(mod 56) =>56/2=28/2=14/2=7+3=10 [7 result + 3 time division]

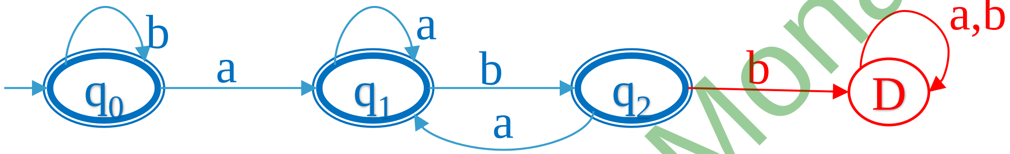


- $\Sigma = \{a, b\}$
- $L_{22.1}$ ={*Every 'a' should be followed by a 'b'*}.
- {∈, b*, ab , abab , abbbab,…….}
- $L_{22.2}$ ={*Every 'a' should not followed by a 'b'*}.
- {∈, b*, a*, ba , bbba , bbaaa,…..}
- $L_{22.3}$ ={*Every 'a' should be followed by a 'bb'*}.
- {∈, b*, abb , bbbabb, abbbabb,…..}
- $L_{22.4}$ ={*Every 'a' should not followed by a 'bb'*}.
- {∈, b*, a*,ab , bbbab, aababa,…..}



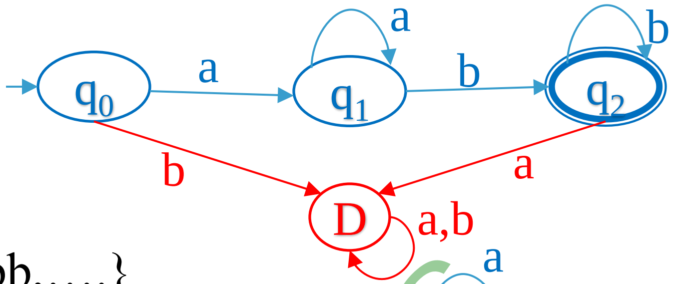


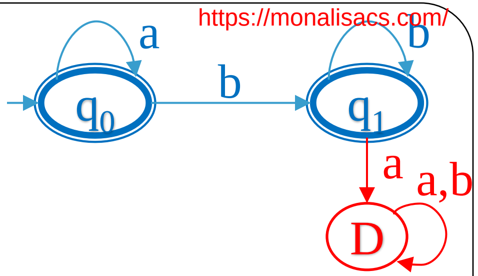


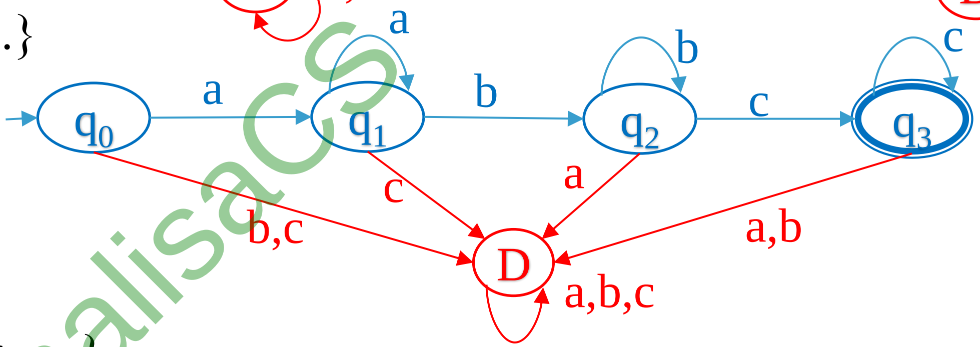


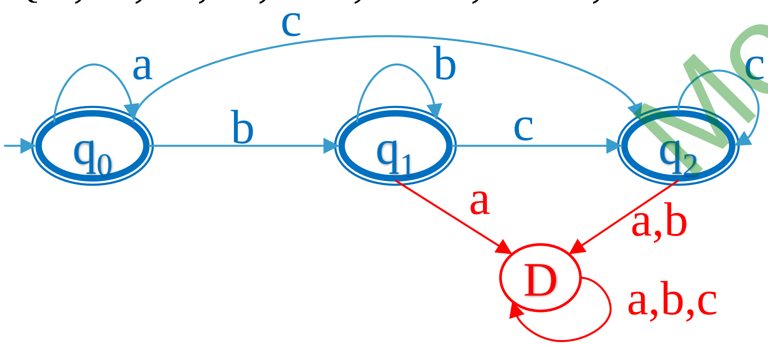
- If L={Every 'x' should be followed by a 'y'}.
- Then minimum number state =|x|+|y|+2
- If L={Every 'x' should not followed by a 'y'}.
- Then minimum number state =|x|+|y|+1

- $\Sigma = \{a, b\}$
- $L_{23.1} = \{a^n b^m \mid n,m \geq 1\}$
- {ab,aab,abb,aabbb,aaabb,….}
- $L_{23.2} = \{a^n b^m \mid n,m \geq 0\}$
- { ∈,a*,b*,ab,aab,abb,aabbb,aaabb,….}
- $\Sigma = \{a, b, c\}$
- $L_{23.3} = \{a^n b^m c^l \mid n,m,l \geq 1\}$
- {abc ,aabc ,abbc ,aabbbcc,….}
- $L_{23.4} = \{a^n b^m c^l \mid n,m,l \geq 0\}$
- {∈,a*,b*,c*,abc ,aabc ,abbc ,aabbbcc,….}

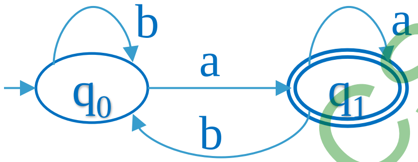


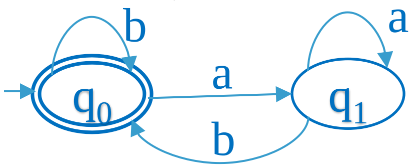


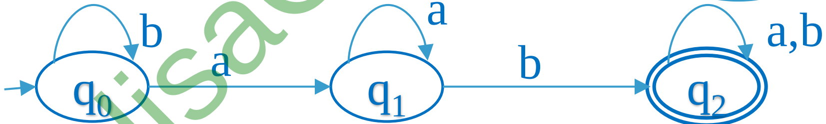


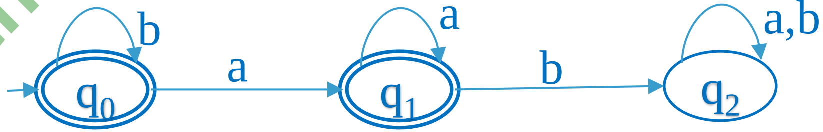


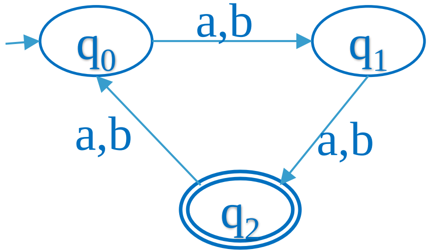
- **Complement** :If L is a regular language then $\bar{L}$ or $L' = \Sigma^* - L$ is also regular language.
- The DFA for L' can be obtained by interchanging final & nonfinal state.
- Number of state in minimal DFA of $L = L'$.
- If L have n state and k final state then L' have n state but n-k final state.
- $\Sigma = \{a, b\}$
- L_1:{*Strings not end with a*}
- {∈ ,b,b*,ab,aab,aabab,baab,….}
- L_2 :{*Don't contain substring ab*}
- {∈ ,a,b,a*,b*,ba , bba,….}
- $L_3 : |w| \neq 2(mod3)$ [0,1,3,4,6,7,9,10..]
- {∈,a,b,aab,bbb,abab,aabbab,…..}

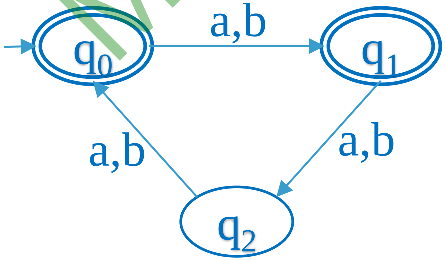






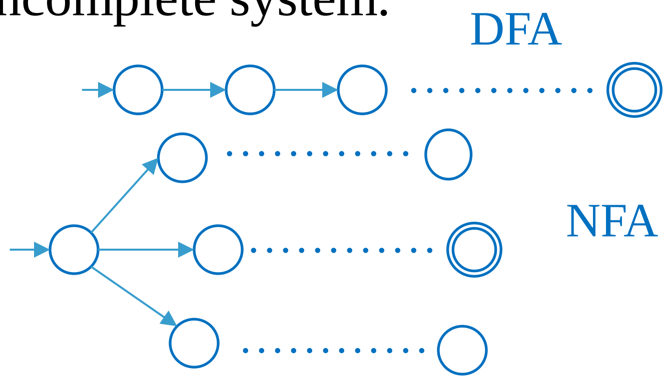




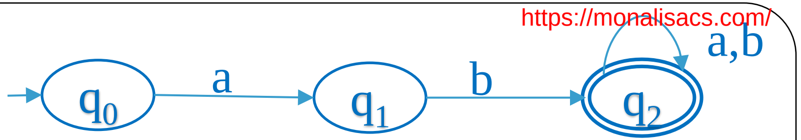


Non Deterministic Finite Automata (NFA)

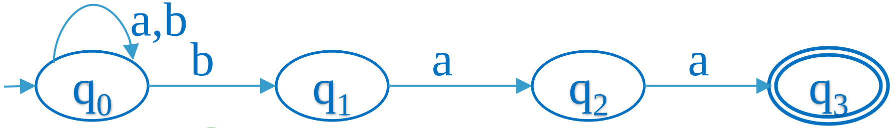
- If FA has 0 or more transaction for any input symbol from any state then its a NFA.
- NFA can move to more than one state after talking input symbol.
- NFA-Only about acceptancy but DFA-For both acceptancy & rejections.
- NFA take care of only valid input string no need to take care of invalid input string.
- No concept of Dead state & complement as it is a incomplete system.
- A NFA has 5-tuple (Q, Σ, δ, q_0, F)
- $\delta : Q \times \Sigma \rightarrow 2^Q$[power set of Q]
- For one language there can be more than one NFA.
- E(DFA)=E(NFA)
- DFA is more efficient and than NFA.
- Representation of RL by NFA is easier than DFA.
- Every DFA is NFA & every NFA can be converted to DFA.
- In NFA there can be multiple transaction path for input string.
- After taking input symbol NFA go to multiple path.
- If one path end with final state then it accept that string.



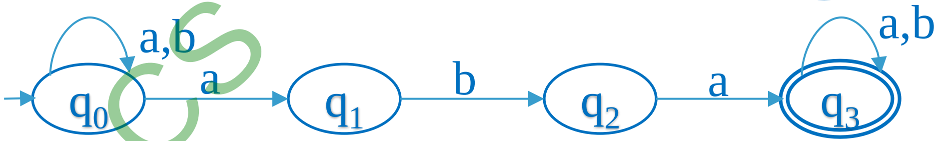
- Construct NFA for following language. $\Sigma = \{a, b\}$
- L_1 :{*Every string start with 'ab'*}.
- {ab,aba,abb,abaa,abba,abaaabbba,……}



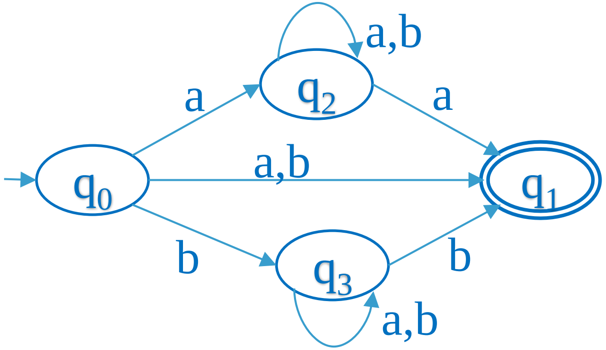
- L_2 :{*Every string end with 'baa'*}.
- {baa,abaa,ababaa,bbaabaa,……}



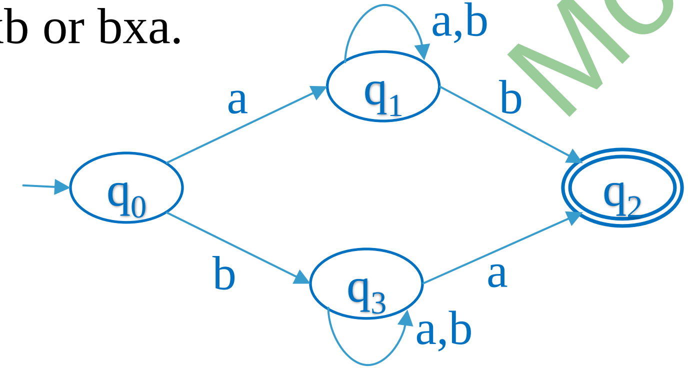
- L_3 :{*Every string contain substring 'aba'*}.
- {aba,aaba,bbabaa,aabaabb,……}



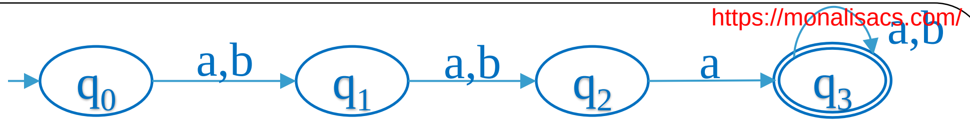
- L_4 :{*Every string start and end with same symbol*}.
- {a,b,aa,bb,aba,bab,abaaabba,baabab,……}
- w=axa or bxb.



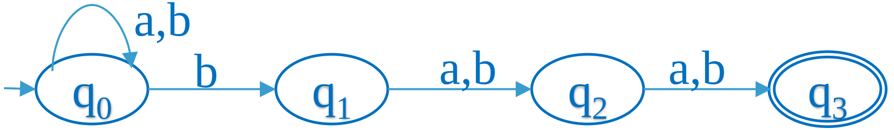
- L_5 :{*Every string start and end with different symbol*}.
- {ab,ba,aab,baa,abbaab,babba,……}
- w=axb or bxa.



- *L_6 :{The 3rd symbol from left end is a}.*
- {aaa,aba,bba,ababb,bbaaba,abaabb,……}
- w=_ _ ax



- *L_7 :{The 3rd symbol from right end is b}.*
- {baa,bbb,bab,babba,abaabab,……}
- w= xb_ _

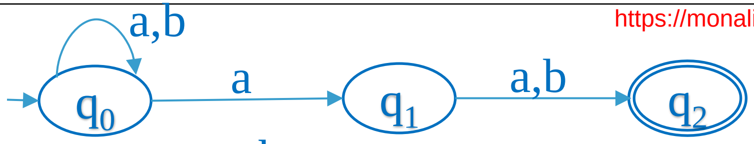


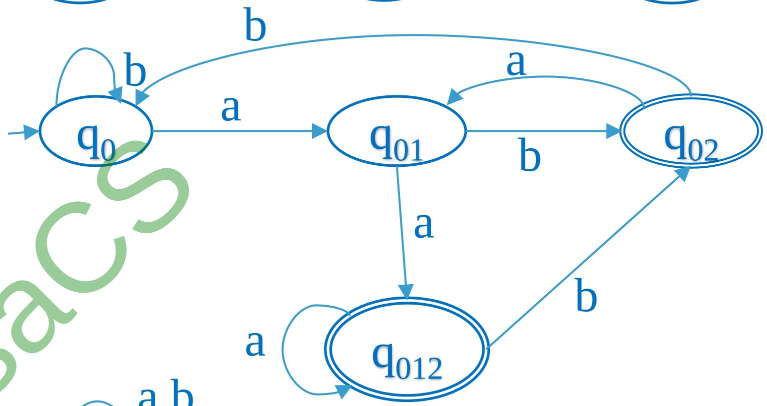
- **Conversion from NFA to DFA:**
- Subset Construction :Process of converting NFA toDFA.
- **Alg:**Let $M=(Q, \Sigma, \delta, q_0, F)$ NFA and $M'=(Q', \Sigma, \delta', q_0', F')$ equivalent DFA
- Start from initial state and continue for every new state.
- Initial state: $q_0 = q_0'$ no change in initial state.
- Final state F' : Every subset which contain final state in NFA is final state in DFA
- Construction of δ'
- $\delta'(q,x)= \delta(q,x)$, $\delta'((q_0,q_1),x)= \delta(q_0,x) \cup \delta(q_1,x)$
- $\delta'(q_0,q_1 \ldots\ldots q_n,x)=\bigcup_{i=0}^{n} \delta(q_i, x)$
- Number of state in DFA $n \leq 2^m$
- m=number of state in NFA, n=number of state in DFA.

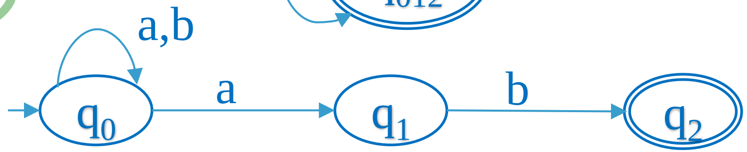
- Conversion NFA to DFA. $\Sigma = \{a, b\}$
- L_1={*The 2nd symbol from right end is 'a'*}
- {aa,ab,aab,abaa,abbaa,……}
- w=xa_

NFA		
	a	b
→q_0	q_0,q_1	q_0
q_1	q_2	q_2
*q_2	ϕ	ϕ

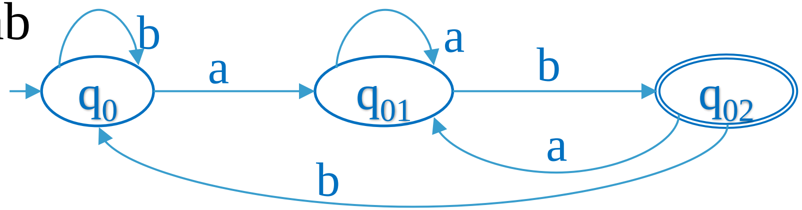
Equivalent DFA		
	a	b
→q_0	q_0,q_1	q_0
q_0,q_1	q_0,q_1,q_2	q_0,q_2
*q_0,q_1q_2	q_0,q_1,q_2	q_0,q_2
*q_0,q_2	q_0,q_1	q_0





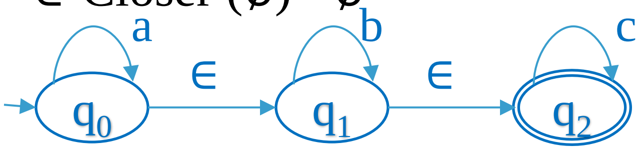


- L_2={*Every string ending with ab*}
- {ab,aab,bab,abab,bbab,aabab,……}
- w=xab

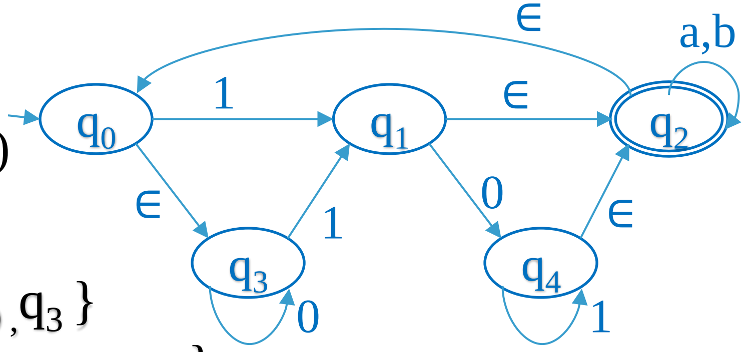


∈ −NFA

- NFA having transaction for empty string called ∈-NFA.
- It has 5-tuple $(Q, \Sigma, \delta, q_0, F)$
- $\delta : Q \times \Sigma \cup \{\in\} \rightarrow 2^Q$
- E(∈-NFA) =E(NFA) = E(DFA)
- **∈-Closer(∈*) :**If q is any state in ∈-NFA then set of all state that can be reach from q on ∈ is ∈-Closer (q) or the states which are at 0 distance from q called ∈-Closer (q) .
- δ(q, ∈)=q ,every state is at zero distance from itself.
- ∈-Closer (q) is non empty finite subset of Q.
- ∈-Closer $(q_0 \cup q_1 \ldots \ldots \cup q_n)$= $\in^*(q_0) \cup \in^*(q_1) \ldots . \in^*(q_n)$
- ∈-Closer (∅)= ∅

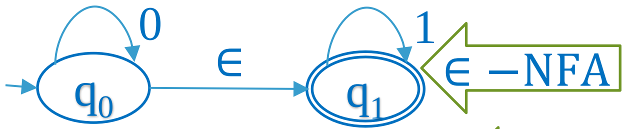


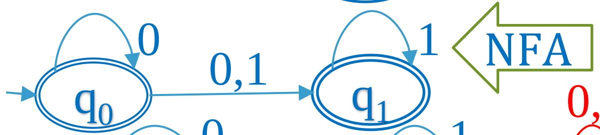
- ∈-Closer $(q_0)=\{q_0, q_1, q_2\}$
- ∈-Closer $(q_1)=\{q_1, q_2\}$
- ∈-Closer $(q_2)=\{q_2\}$



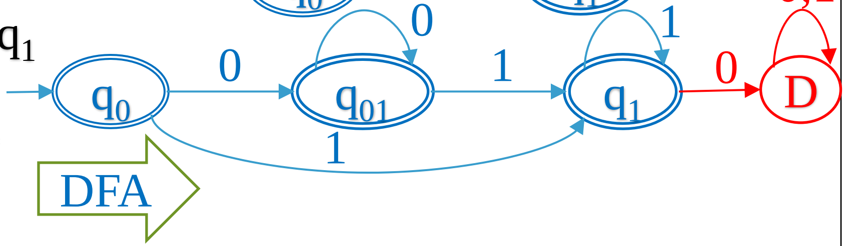
- ∈-Closer $(q_0)=\{q_0, q_3\}$
- ∈-Closer $(q_1)=\{q_0, q_1, q_2, q_3\}$
- ∈-Closer $(q_2)=\{q_0, q_2, q_3\}$
- ∈-Closer $(q_3)=\{q_3\}$
- ∈-Closer $(q_4)=\{q_0, q_2, q_3, q_4\}$

- **Conversion of ∈- NFA to NFA**
- No change in number of state.No change in initial state.
- There may be change in final state.
- Alg: Let $M=(Q, \Sigma, \delta, q_0, F)$ ∈-NFA and $M'=(Q', \Sigma, \delta', q_0', F')$ equivalent NFA
- Initial state: $q_0 = q_0'$ no change in initial state.
- Final state F' :Every state whose ∈-Closer contain the final state of ∈- NFA is a final state in NFA.
- Construction of δ': $\delta'(q,x)=$ ∈-Closer{δ(∈-Closer(q),x)}
- ∈-Closer $(q_0)=\{q_0, q_1\}$ ∈-Closer{δ(∈-Closer(q),x)}
- ∈-Closer $(q_1)=\{q_1\}$





	∈*	0	∈*
q_0	q_0	q_0	q_0, q_1
	q_1	φ	φ
	∈*	1	∈*
q_0	q_0	φ	φ
	q_1	q_1	q_1
	∈*	0	∈*
q_1	q_1	φ	φ
	∈*	1	∈*
q_1	q_1	q_1	q_1



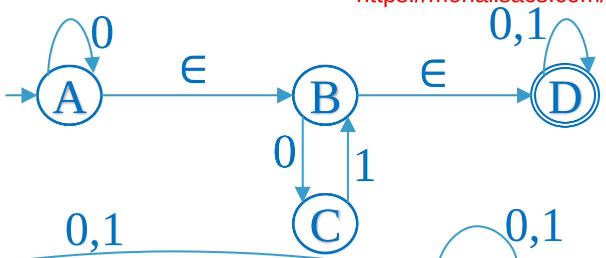
Equivalent NFA

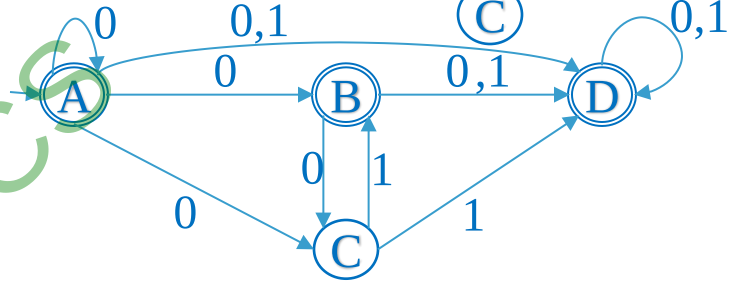
	0	1
$\rightarrow *q_0$	q_0, q_1	q_1
$*q_1$	φ	q_1

- ∈-Closer (A)={A,B,D}
- ∈-Closer (B)={B,D}
- ∈-Closer (C)={C}
- ∈-Closer (D)={D}

Equivalent NFA		
	0	1
→*A	A,B, C,D	D
*B	C,D	D
C	ф	B,D
*D	D	D

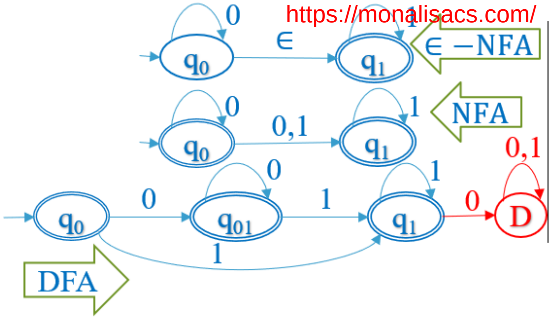
	∈*	0	∈*
A	A	A	A,B,D
	B	C	C
	D	D	D
	∈*	**1**	**∈***
A	A	ф	ф
	B	ф	ф
	D	D	D
	∈*	**0**	**∈***
B	B	C	C
	D	D	D
	∈*	**1**	**∈***
B	B	ф	ф
	D	D	D
	∈*	**0**	**∈***
C	C	ф	ф
	∈*	**1**	**∈***
C	C	B	B,D
	∈*	**0,1**	**∈***
D	D	D	D





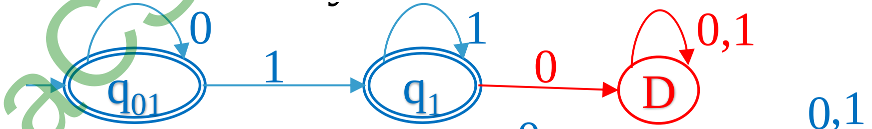
Conversion of ∈- NFA to DFA

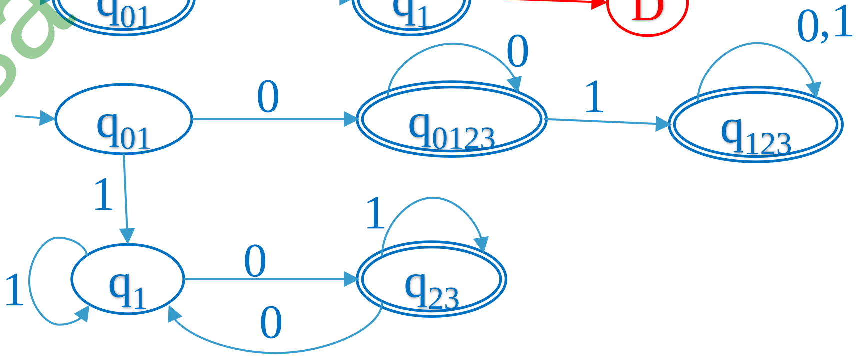
- Alg: Let $M=(Q, \Sigma, \delta, q_0, F)$ ∈-NFA and $M'=(Q', \Sigma, \delta', q'_0, F$
- Initial state: q'_0 = ∈-Closer(q_0)
- Final state F' :Every subset which contain F is final state in
- Construction of δ': $\delta'(q,x)$= ∈-Closer{$\delta(q,x)$}
- Start construction of δ' in initial state and continue for ever

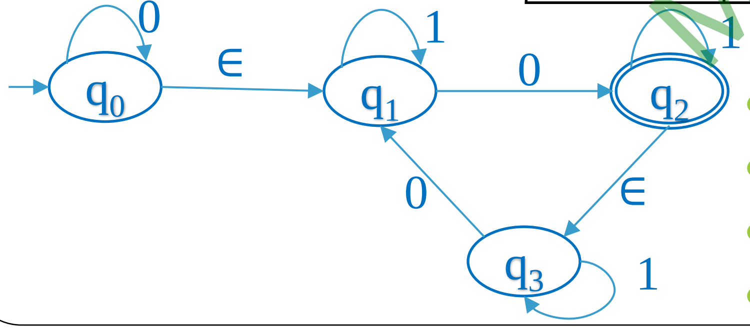


- ∈-Closer (q_0)={q_0 ,q_1}
- ∈-Closer (q_1)={q_1}

Equivalent DFA		
	0	1
→*q_0,q_1	q_0,q_1	q_1
*q_1	D	q_1
D	D	D



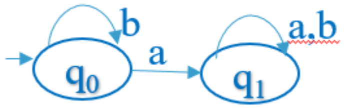




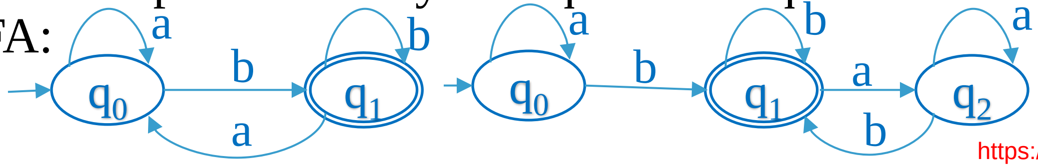
- ∈-Closer (q_0)={q_0 ,q_1}
- ∈-Closer (q_1)={q_1}
- ∈-Closer (q_2)={q_2 ,q_3}
- ∈-Closer (q_3)={q_3}

Decision Properties of FA

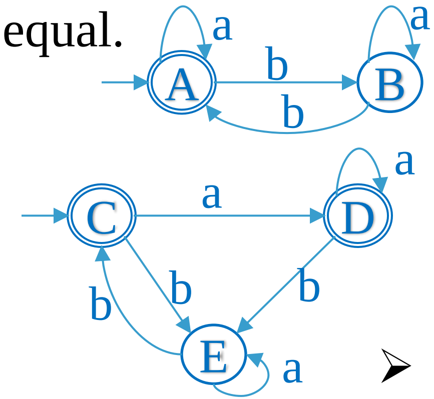
- 1.Emptyness 2.Finiteness 3.Equalness 4.Membership
- <u>Emptiness:</u>
- Remove all the unreachable state.
- If the resulting FA contain at least one final state then the FA accept non empty language.
- If it is free from final state then FA accept empty language. Example



- <u>Finiteness:</u>
- Remove all the unreachable state & Dead state
- If the resulting FA is free from cycle and loop then the FA accept finite language.
- If FA contain loop & cycle then FA accept infinite language.
- <u>Equalness:</u>
- Two FSM M_1 & M_2 are said to be equal if both of them accept same language.
- Equal machine no need to contain same number of state.
- Two FSM are isomorphic to each other .If both of them accept same language with same number of state.
- If two machines are isomorphic then they are equal. But equal need not be isomorphic.
- Example of Equal FA:
- $L(M_1)=L(M_2)=xb$



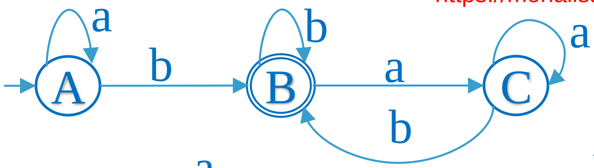
- <u>Membership:</u>
- After reading a string ‘x’ if it reaches at final state then ‘x’ is member of that FA.
- If it reaches non final state then ‘x’ is not member of that FA.
- <u>Comparison Algorithm to check Equalness :</u>
- Let M_1 & M_2 be two FSM.
- Construct transaction table that contain pair of states (x,y) s.t $x \in M_1$, $y \in M_2$.
- Start construction of table with the pair of initial state.
- While constructing table if we get any pair of the form (F,NF) or (NF,F) then stop construction and declare two machines are not equal.
- Continue construction of table for every new pair of the form (F,F) & (NF,NF) and stop construction whenever no such pair occur.
- If transaction table contain all the pair of the form (F,F) & (NF,NF) then two FSM are equal.

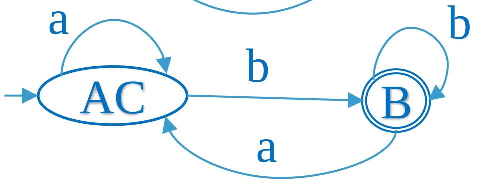


	a	b
A,C	A,D	B,E
A,D	A,D	B,E
B,E	B,E	A,C

➢ Both DFA are equal as all pairs are (F,F) or (NF,NF)

- **Minimization of FA:**
- Find unreachable state and remove them .
- Find the equal state and remove them.
- There are two algorithm for finding equal state.
  - 1.State equivalence method
  - 2.Table filling method(Myphill-Nerode Theorem)
- The FA which is free from unreachable and equal state is called minimal FA.
- **1.State equivalence method**
- Two state p and q are equivalent
- If $\delta(p,w) \in F \Rightarrow \delta(q,w) \in F$ and $\delta(p,w) \in NF \Rightarrow \delta(q,w) \in NF$
- All the states **Q** are divided in two partitions − **final states** and **non-final states** and are denoted by $\mathbf{P_0}$ or 0^{th} equivalent.
- If |w|=0 , 0^{th} equivalent
- If |w|=1 , 1^{th} equivalent
- If |w|=n , n^{th} equivalent
- Find equivalents till two equivalents are same.



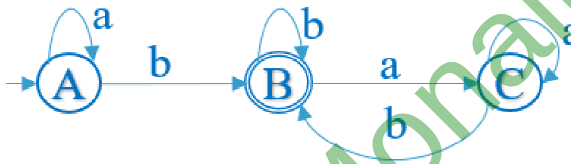


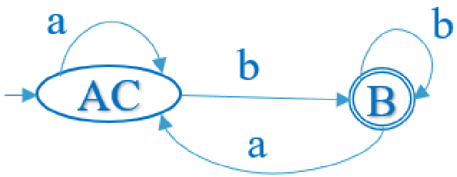
- $P_0 = \{(A,C),(B)\}$
- $P_1 = \{(A,C),(B)\}$

	a	b
→A	A	B
*B	C	B
C	C	B

- **2.Table filling method**(Myphill-Nerode Theorem)
- **Step 1** – Draw a table for all pairs of states (Q_i, Q_j) not necessarily connected directly[All are unmarked initially]
- **Step 2** – Consider every state pair (Q_i, Q_j) in the DFA where $Q_i \in F$ and $Q_j \in NF$ or vice versa and mark them.
- **Step 3** – Repeat this step until we cannot mark anymore states –
- If there is an unmarked pair (Q_i, Q_j), mark it if the pair {δ (Q_i, Σ), δ (Q_j, Σ)} is marked for some input alphabet.
- **Step 4** – Combine all the unmarked pair (Q_i, Q_j) and make them a single state in the reduced DFA.

	a	**b**
→A	A	B
*B	C	B
C	C	B



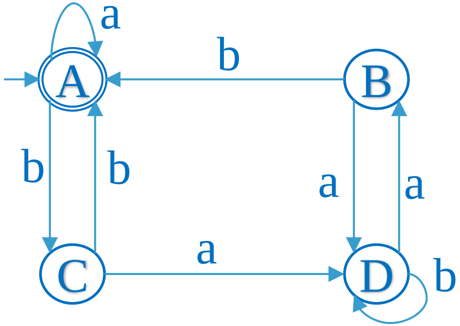


	A	B	C
A	AA	AB	AC
B	BA	BB	BC
C	CA	CB	CC

	A	B
B	X	
C	=	X

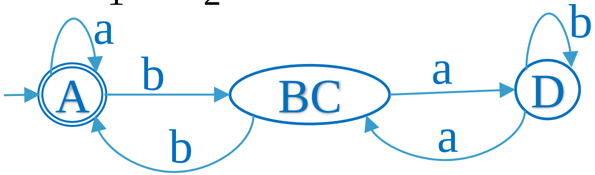
- Example-2

	A	B	C
B	X		
C	X	=	
D	X	X	X



	a	b
→*A	A	C
B	D	A
C	D	A
D	B	D

- $\Pi_0 = \{(A),(B,C,D)\}$
- $\Pi_1 = \{(A),(B,C),(D)\}$
- $\Pi_2 = \{(A),(B,C),(D)\}$
- $\Pi_1 = \Pi_2$

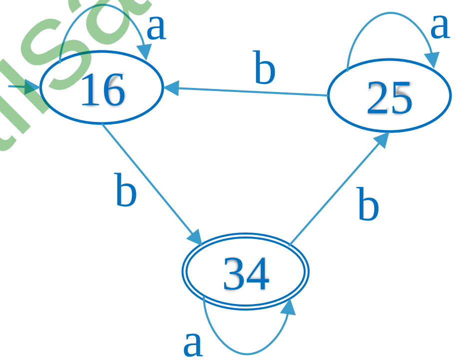


- Example-3

	a	b
→1	6	3
2	5	1
*3	4	2
*4	3	5
5	2	6
6	1	4

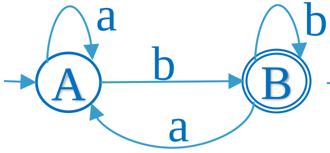
- $\Pi_0 = \{(1,2,5,6),(3,4)\}$
- $\Pi_1 = \{(1,6),(2,5),(3,4)\}$
- $\Pi_2 = \{(1,6),(2,5),(3,4)\}$

	1	2	3	4	5
2	X				
3	X	X			
4	X	X	=		
5	X	=	X	X	
6	=	X	X	X	X

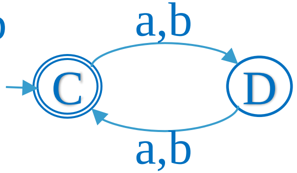


- **Compound Automata :(Divide & Conquer)**
- Compound FA is the resultant DFA formed after performing operation (∪, ∩, -,.) on given DFAs D1 and D2.
- D1 = (Q_1, ∑, δ, q_1, F_1) and D2 = (Q_2, ∑, δ, q_2, F_2)
- **Properties of Compound Finite Automata (FA):**
- Number of states in compound FA (D1XD2) is equal to m*n, where m is the number of states in D1 and n is the number of states D2.
- Initial state of compound FA is combination of initial states of D1 and D2.
- Final state of compound FA depends on the operation performed.
- In case of AND the subset which contain both final state will be final state.
- In case of OR the subset which contain at least one final state is final state
- A-B=$A \cap B'$.
- Ex-1:L={Every string end with b AND length of string is even}, Σ={a,b}

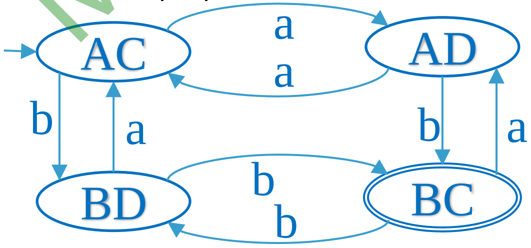
w=xb |w|=2 w=xb ∩|w|=2 #state=2*2=4



M_1

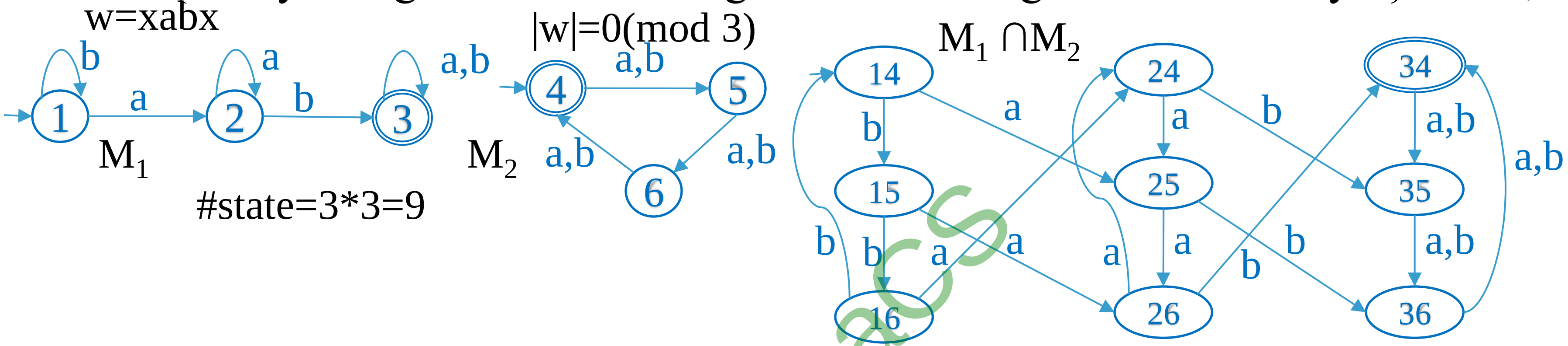


M_2

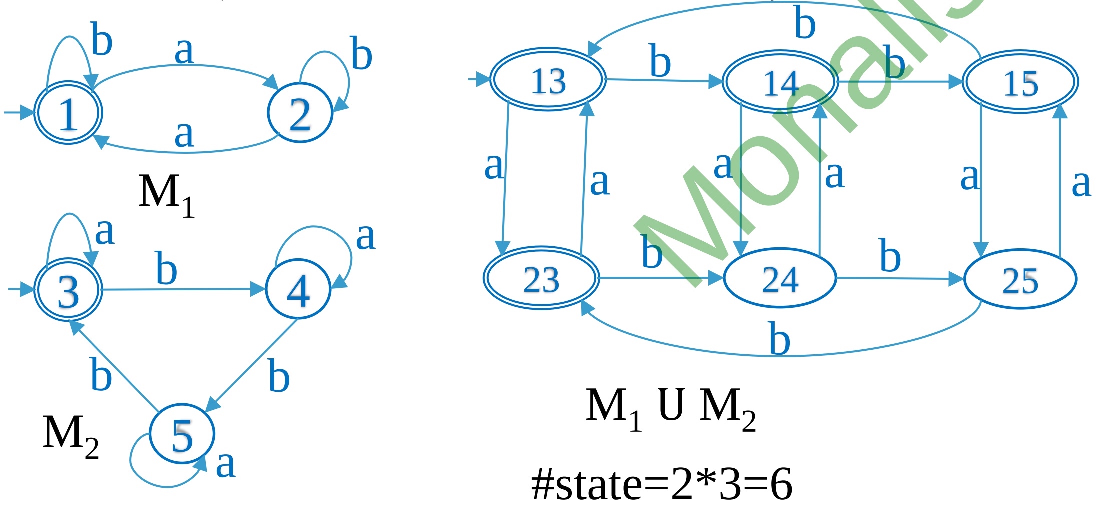


$M_1 \cap M_2$

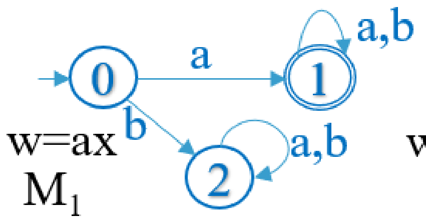
- Ex-2:L={Every string contain substring ‘ab’ AND length is divisible by 3}, Σ={a,b}

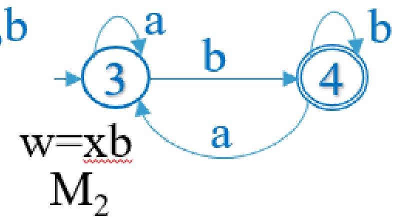


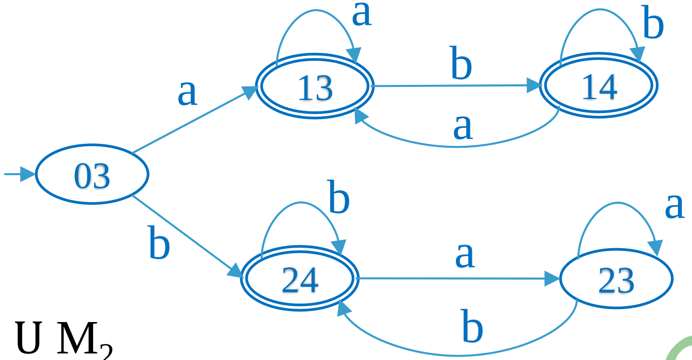
- Ex-3:L={Number of a’s divisible by 2 OR number of b’s divisible by 3} Σ={a,b}



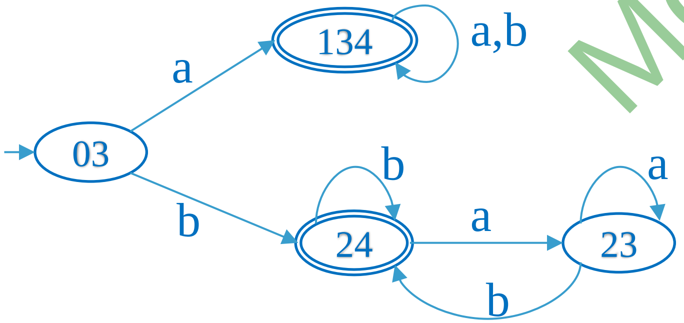
- Ex-4:L={Every string start with ‘a’ OR ends with ‘b’} Σ={a,b}



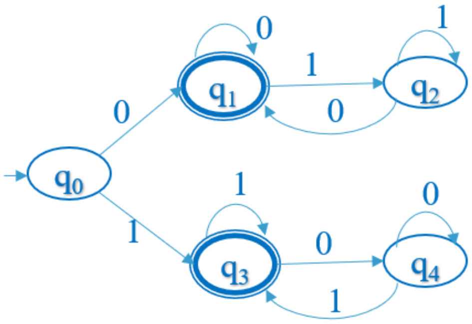




M_1 U M_2

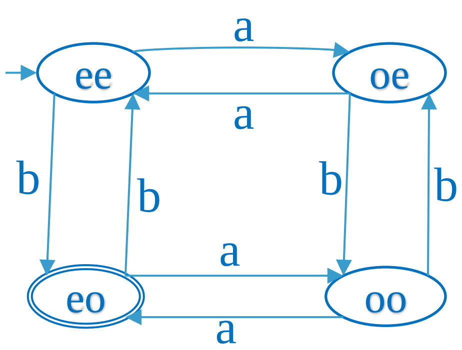


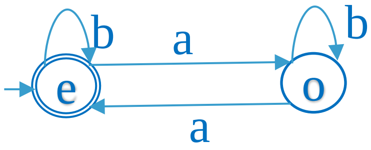
a,b,aa,ab,bb,ba,baab,aba,baa



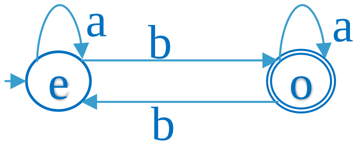
#state=4

- Ex-5:L={Number of a's is even AND number of b's is odd} $\Sigma=\{a,b\}$





M_1



M_2

$M_1 \cap M_2$ #state=2*2=4

Final State

- # a's even and #b's even=ee
- # a's odd and #b's even=oe
- # a's odd and #b's odd=oo
- # a's even or #b's even=ee,eo,oe
- # a's odd or #b's even=oe,oo,ee
- # a's even or #b's odd=eo,ee,oo

- Ex-6:L={Length of string is divisible 2 and divisible by 4} $\Sigma=\{a,b\}$
- $L_1=|w|=0 \pmod 2=\{0,2,4,6,8,..\}$ 2 state, $L_2=|w|=0 \pmod 4=\{0,4,8,12,..\}$ 4 state
- $L_1 \cup L_2=\{0,2,4,6,8,10,\ldots\ldots\}$=2 state, $L_1 \cap L_2=\{0,4,8,12,16,\ldots\ldots\}$=4 state

	Language	**# of state**
L_1	Length divisible by 2 OR divisible by 4	2
L_2	Length divisible by 2 AND divisible by 4	4
L_3	Length divisible by 9 OR divisible by 3	3
L_4	Length divisible by 4 AND divisible by 12	12
L_5	Length divisible by 16 OR divisible by 8	8

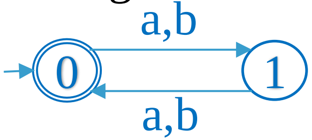
	Language	# of state
L_6	Length divisible by 3 OR divisible by 4	12
L_7	Length divisible by 5 AND divisible by 4	20
L_8	Length divisible by 6 OR divisible by 4	12
L_9	Length divisible by 8 AND divisible by 10	40
L_{10}	Length divisible by 12 OR divisible by 18	36

If GCD (A,B) =A or B	A∪ B, OR	GCD(A,B)
	A∩ B, AND	LCM(A,B)
If GCD(A,B) ≥1≠A or B	A∪ B, OR	LCM(A,B)
	A∩ B, AND	LCM(A,B)

- **Difference** :If L_1 ,L_2 two RL then L_1-L_2 also a regular language. $L_1-L_2 = L_1 \cap L_2' = M_1 \cap M_2'$
- Ex-7:L={Length of string is even but number of a is not even} Σ={a,b}

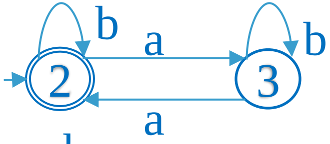
|w|=even

M_1

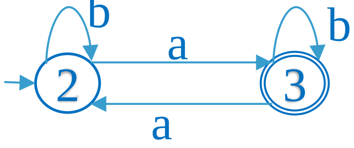


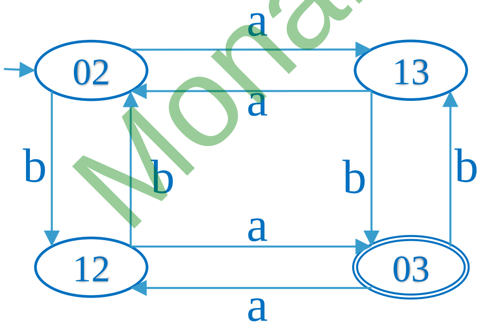
$|w|_a$=even

M_2



M_2'



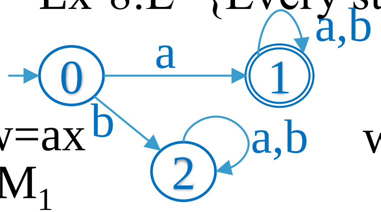


$M_1 \cap M_2'$

- **Concatenation** :If L_1 ,L_2 two RL then $L_1.L_2$ also a regular language.
- Ex-8:L={Every string start with ‘a’ ,ends with ‘b’} Σ={a,b}

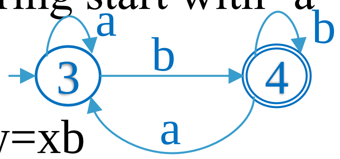
w=ax

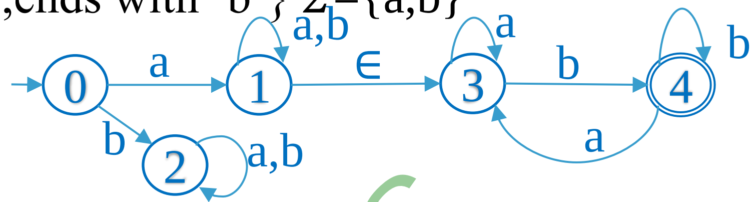
M_1

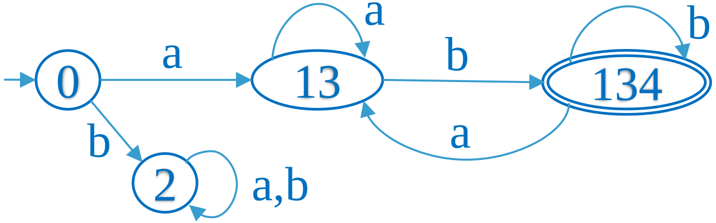


w=xb

M_2



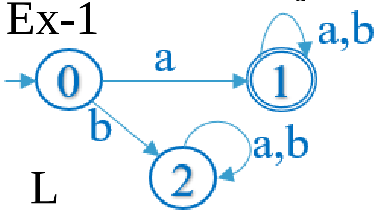




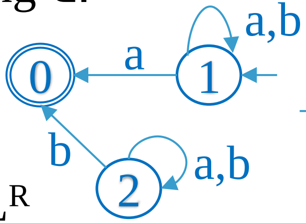
- For $M_1.M_2$ add a ∈ transaction from M_1final state to initial state of M_2 .
- Make M_1 final state to nonfinal state.
- Convert ∈-NFA to DFA.

- **Reverse** :If L is a RL then L^R also a regular language. Reverse of DFA may not be DFA.
- Process to get FA for L^R
- Step-1:Interchange initial & final state.
- Step-2:Change the direction of edge if more than one initial state occur then convert into one initial state by using ∈.

Ex-1

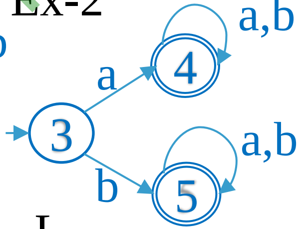


L

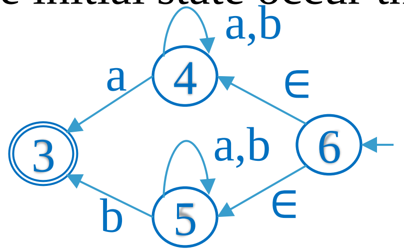


L^R

Ex-2

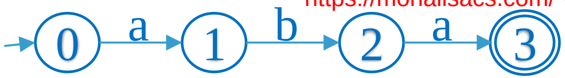


L



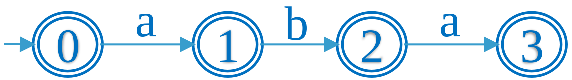
L^R

- **Prefix** :If L is a RL then Prefix(L) also a regular language. Ex-1

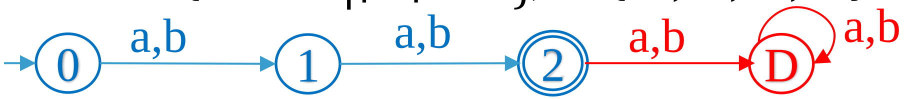


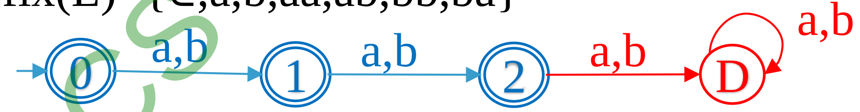
- <u>Process to get FA for Prefix(L):</u>
- In NFA convert all state into final state.
- In DFA convert all state into final state except dead state.

L={aba} Prefix(L)={∈,a,ab,aba}



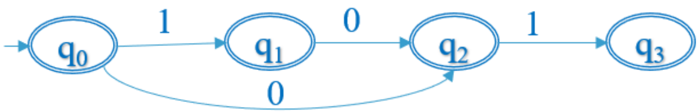
- Ex-2:$L=\{w\in \Sigma^* \mid |w| = 2\}$, L={aa,ab,bb,ba} Prefix(L)={∈,a,b,aa,ab,bb,ba}





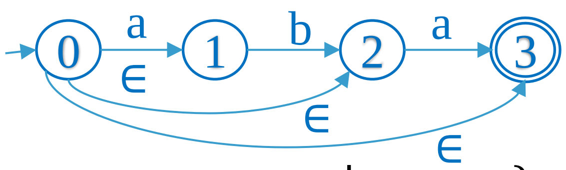
- Number of state in FA and Prefix(L) is same.
- **Substring:** If L is a RL then all Substring(L) also a regular language if L is a finite language.
- For infinite language Its substring may be RL or non RL.
- L=a*b* then substring of L can be (a^nb^n)which is not RL

- <u>Process to get FA for all Substring(L):</u>
- Design FA for prefix.
- Add transaction for substring which are not prefix.

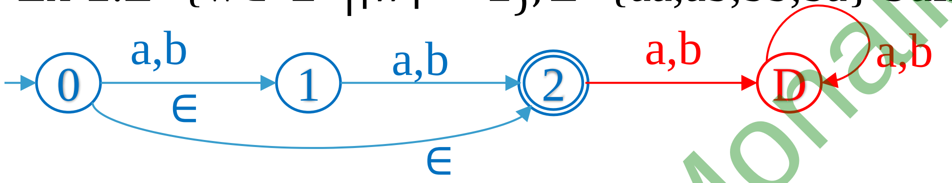


- Ex:L is set of all substrings of ‘w’, if w = ‘101’ ,
- Then L={ ϵ, 0, 1, 10, 01, 101}.

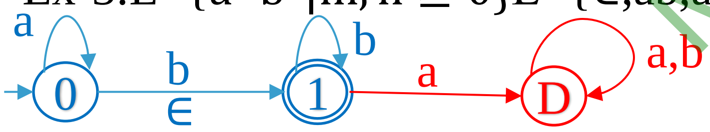
- **Suffix** :If L is a RL then Suffix(L) also a regular language.
- <u>Process to get FA for Suffix(L):</u>
- In NFA add ∈ transition from initial state to every state .
- In DFA add ∈ transition from initial state to every state except dead state.
- Ex-1: L={aba}, Suffix(L)={aba,ba,a, ∈}



- Ex-2:L={w∈ $\Sigma^* \,\big|\, |w| = 2$}, L={aa,ab,bb,ba} Suffix(L)={aa,ab,bb,ba,a,b, ∈}



- Ex-3:L={$a^m b^n | m, n \geq 0$}L={∈,ab,aab,abb,….} Suffix(L)={b,ab,abb,aab,b*,a*, ∈..}

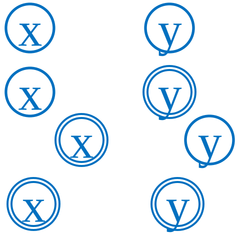


DFA

- $|\Sigma| = m$ (# Symbol) ,$|Q| = n$ (# State)
- Number of types = ${}^{n}C_0 + {}^{n}C_1 + {}^{n}C_2 \ldots\ldots + {}^{n}C_n = 2^n$ [From n state how many state are final]
- Number of DFA of each type = $n^{m.n}$
- Total number of DFA in each type= $2^n \times n^{m.n}$
- Q1:how many number of 2 state DFA with fixed initial state can be constructed over $\Sigma=\{0,1\}$?
- $Q=\{x,y\}$
- $2^4=16$

	0	**1**
x	x or y 2 way	x or y 2 way
y	x or y 2 way	x or y 2 way

0 Final state: ${}^{2}\mathbf{C_0}=1$	16 DFA
1 Final state: ${}^{2}\mathbf{C_1}=2$	16 DFA
	16 DFA
2 Final state: ${}^{2}\mathbf{C_1}=1$	16 DFA
Total # DFA	64 DFA



- $n=2, m=2$,#DFA= $2^2 \times 2^{2.2} = 4\times16 = 64$ DFA possible
- Q2:how many number of 3 state DFA with constant initial state can be constructed over $\Sigma=\{0,1\}$? $n=3, m=2$, #DFA= $2^3 \times 3^{2.3} = 8 \times 729 = 5832$ DFA possible.
- Q3:how many number of 5 state DFA with constant initial state can be constructed $|\Sigma|=2$ with at most 3 final state?
- $n=5, m=2$, #Types= ${}^{5}C_0 + {}^{5}C_1 + {}^{5}C_2 + {}^{5}C_3 = 1+5+10+10=26$
- #DFA= $26 \times 5^{2.5} = 26 \times 5^{10}$ DFA possible.

Regular Expression

- An expression which is constructed over using operator *,.,+ is called as RE
- A regular expression can be described as a sequence of pattern that defines a string.
- Regular expressions are used to match character combinations in strings
- It is used in compiler,interpreter,testing tool,searching,text editor,pattern recognition.
- Every RE generate only one RL but RL can have more than one RE.RE is not unique.
- The language generated by RE is accepted by some FA and is a RL.
- <u>Regular operator:</u>
- *= Kleene closer, . =Concatenation ,+ /U=Union
- If r_1 and r_2 two RE then $(r_1+r_2)^*=(r_1^*+r_2)^*=(r_1+r_2^*)^*=(r_1^*+r_2^*)^*=(r_1^*.r_2^*)^*$
- $r_1(r_2.r_1)^*=(r_1.r_2)^*r_1$ shifting rule.
- Two RE are equal iff $L(r_1)=L(r_2)$
- $r=\phi$, $r^*=\epsilon$, $r^+=\phi$
- $r=\epsilon$, $r^*=\epsilon$, $r^+=\epsilon$
- r^* is infinite language except for $r=\phi$ or ϵ.

- Simplify following RE
- 1.$(0^*+\in)^*=0^*$
- 2.$(0^*.0)^*=(0^+)^*= 0^*$
- 3.$(0^*1+1)^*=(0^*1)^*$
- 4.$(0^*1+10^*)^*=(0^*10^*)^*$
- 5.$(0^*+1^+0^*)^*+(10)^*= (\in.0^*+1^+0^*)^*+(10)^*=((\in+1^+)0^*)^*+(10)^*$
- $=(1^*0^*)^*+(10)^*=(1+0)^*+(10)^*=(1+0)^*$
- If $L_1=\phi$, $L_2= \in$, $L_3= a$
- $L_1^*= \in$, $L_2^*= \in$, $L_3^*=a^*$
- $(L_1.L_2)^*=(\phi.\in)^*= \phi^*= \in$
- $L_1 L_3^* + L_1^*=\phi.a^* + \in = \phi + \in = \in$
- $(L_2.L_3)^*. L_1= (\in.a)^*. \phi =a^*. \phi = \phi$
- $L_1^*.(L_2^*+L_3)^*= \phi^*.(\in+a)^*= \in.a^*=a^*$
- Q: Which of the following RE are identical
- 1)a^* 2)$(aa)^*$ 3)$(a+\in)a^*$ 4)$a(aa)^*$
- $(a+\in)a^*=a^++a^*=a^*$
- Ans: $a^* =(a+\in)a^*$

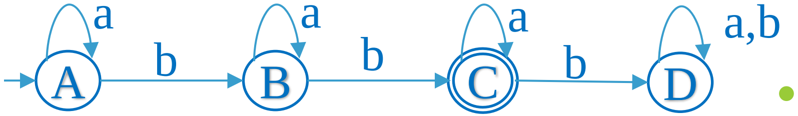
- **Construction of Regular Expression:**
- $\Sigma=\{0,1\}$, $\Sigma^*=(0+1)^*$, $\Sigma^+=(0+1)^+$
- $L_1=(01)^n|0\leq n\leq 2$ $r=(\in +01+0011)$
- $L_2=0^m1^n|m+n=2$ $r=(01+00+11)$
- L_3={Every string start with 10}, $r=10(0+1)^*$
- L_4={Every string end with 01}, $r=(0+1)^*01$
- L_5={Every string contain substring 101}, $r=(0+1)^*101(0+1)^*$
- L_6={Every string start and end with 0}, $r=0(0+1)^*0+0$
- L_7={Every string start and end with same symbol}, $r=0(0+1)^*0+1(0+1)^*1+0+1$
- L_8={Every string start and end with different symbol}, $r=0(0+1)^*1+1(0+1)^*0$
- L_9 ={The 3rd symbol from left end is 1}, $r=(0+1)^2 1(0+1)^*$
- L_{10}={The 4th symbol from right end is 0}, $r=(0+1)^*0(0+1)^3$
- L_{11}={Length of string is exactly 3}, $r=(0+1)^3$
- L_{12}={Length of string is at most 3}, $r=\in +(0+1)+(0+1)^2+(0+1)^3$ or $r=(\in+0+1)^3$
- L_{13}={Length of string is at least 3}, $r=(0+1)^3(0+1)^*$
- L_{14}={Length of string is divisible by 3}, $r=((0+1)^3)^*$
- L_{15}={Number of ‘0’ is exactly 2} , $r=1^*01^*01^*$

- L_{16}={Length of string is even}, $r=((0+1)^2)^*$
- L_{17}={Length of string is odd}, $r=(0+1)((0+1)^2)^*$
- L_{18}={Every string start with '0' and Length of string is even}, $r=0(0+1)((0+1)^2)^*$
- L_{19}={Every string start with '1' and Length of string is odd}, $r=1((0+1)^2)^*$
- L_{20}={Every string start with 0 & doesn't contain two consecutive 1}, $r=(0+01)^+$
- L_{21}={Every string doesn't contain two consecutive 0 or consecutive 1}, $r=(1+\in)(01)^*(0+\in)$ or $r=(0+\in)(10)^*(1+\in)$
- $L_{22}=\{0^n | n\geq 0\}$, $r=0^*$
- $L_{23}=\{0^n | n\geq 1\}$, $r=0^+$
- $L_{24}=\{0^m1^n | m\geq 0, n\geq 0\}$, $r=0^*1^*$
- $L_{25}=\{0^m1^n | m\geq 1, n\geq 1\}$, $r=0^+1^+$
- $L_{26}=\{0^m1^n | m\geq 1, n\geq 0\}$, $r=0^+1^*$
- $L_{27}=\{(01)^n | n\geq 0\}$, $r=(01)^*$
- $L_{28}=\{0^m1^n | m+n=\text{even}\}$, $r=(00)^*(11)^*+0(00)^*1(11)^*$
- $L_{29}=\{0^m1^n | m+n=\text{odd}\}$, $r=0(00)^*(11)^*+(00)^*(11)^*1$

- $L_{30}=\{wxw^r|w,x \in(0+1)^+\}$
- $wxw^r = 0x0 + 1x1$
- $r=0(0+1)^+0+ 1(0+1)^+1$
- $L=\{wxw^r|w,x \in(0+1)^+\}$ is RL
- $L=\{wxw^r|w \in(0+1)^+\}$ is NRL ,It's CFL.
- $L=\{xww^r|w,x \in(0+1)^+\}$ is NRL ,It's CFL.
- $L=\{ww^rx|w,x \in(0+1)^+\}$ is NRL ,It's CFL.
- $L=\{ww^r|w,\in(0+1)^+\}$ is NRL ,It's CFL.
- Q:Number of string of length ≤3 generate by RE (0+10)*0(1+0)*
- |w|=1 {0} 1
- |w|=2 {00,01} 2
- |w|=3 {001,000,100,010,011} 5
- Total # string |w| ≤3 is 1+2+5=8
- **Conversion of Finite Automata to Regular Expression**
- 1. Arden's Theorem
- 2.State elimination method

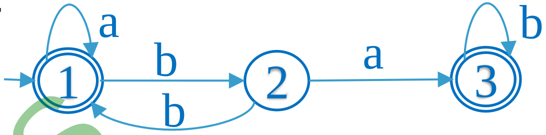
- **Arden's Theorem: (FA →RE)**
- Let P ,Q & R be 3 RE over Σ. P doesn't contain ∈
- Then the equation R=Q+RP has unique solution R=QP*.
- $R=Q+RP \Rightarrow R=QP^*, P \neq \in$
- If P contain ∈ then R has infinite many solution.
- In case of DFA make it free from non productive state then find RE from rest of state
- **Proof** –
- R = Q + (Q + RP)P [After putting the value R = Q + RP]
- $= Q + QP + RPP = Q + QP + QP^2 + QP^3 \ldots\ldots$
- $R = Q(\varepsilon + P + P^2 + P^3 + \ldots) = QP^*$ [As P* represents $(\varepsilon + P + P^2 + P^3 + \ldots)$]
- **Process:**
- **Step 1** – Create equations for all the states of the FA having n states with initial state q_1.
- $q_1 = q_1R_{11} + q_2R_{21} + \ldots + q_nR_{n1} + \in$
- $q_2 = q_1R_{12} + q_2R_{22} + \ldots + q_nR_{n2}$
- …………………………
- $q_n = q_1R_{1n} + q_2R_{2n} + \ldots + q_nR_{nn}$
- $\mathbf{R_{ij}}$ represents Incoming edges from $\mathbf{q_i}$ to $\mathbf{q_j}$, if no such edge exists, then $\mathbf{R_{ij} = \emptyset}$
- **Step 2** – Solve these equations to get the equation for the final state in terms of $\mathbf{R_{ij}}$

- Ex-1



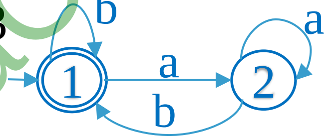
- A= ∈ +Aa
- = ∈a*
- = a*
- B =Ab+Ba
- =a*b+Ba
- =a*ba*
- C=Bb+Ca
- =a*ba*b+Ca
- =a*ba*ba*
- R=Q+RP ⇒R=QP*
- R.E.=R.E.(C)= a*ba*ba*

- Ex-2



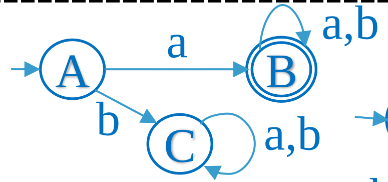
- R.E.=R.E.(1)+R.E.(3)= (a+bb)*+(a+bb)* bab*
- 1= ∈ +1a+2b
- = ∈ +1a+1bb
- = ∈ +1(a+bb)
- = ∈ (a+bb)*
- = (a+bb)*
- 2=1b
- 3=2a+3b
- =1ba+3b
- =(a+bb)* ba+3b
- =(a+bb)* bab*

- Ex-3



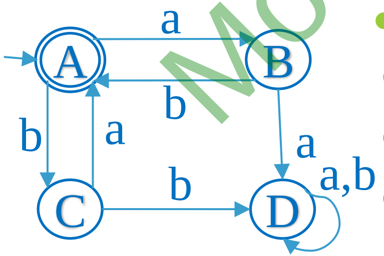
- 1= ∈ +1b+2b
- 1= ∈ +1b+1aa*b
- = ∈ +1(b+aa*b)
- = ∈ (b+aa*b)*
- = (b+a^+b)*=(a*b)*
- 2=1a+2a=1aa*
- R.E.=R.E.(1)=(a*b)*

- Ex-4



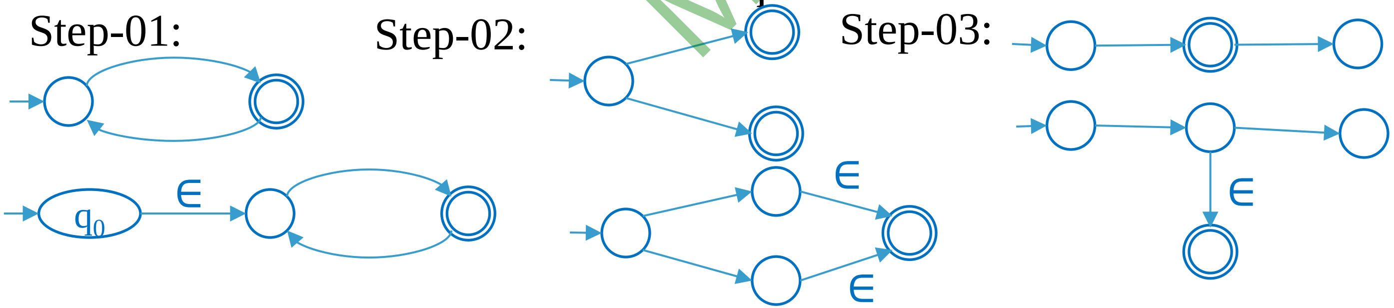
- A= ∈
- B=Aa+Ba+Bb
- =Aa+B(a+b)
- = ∈a+B(a+b)
- =a(a+b)*
- R.E.=R.E.(B)= a(a+b)*

- Ex-5



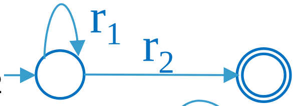
- B=Aa
- C=Ab
- A=Bb+Ca+ ∈
- =Aab+Aba+ ∈
- =A(ab+ba) + ∈
- = ∈(ab+ba)*
- =(ab+ba)*
- R.E.=R.E.(A) =(ab+ba)*

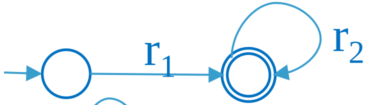
- Drawback :Finding Regular expression for every non final state.
- **State Elimination method: (FA →RE)**
- Step-01:If there exists any incoming edge to the initial state, then create a new initial state having no incoming edge to it.
- Step-02:If there exists multiple final states in the DFA, then convert all the final states into non-final states and create a new single final state.
- Step-03:If there exists any outgoing edge from the final state, then create a new final state having no outgoing edge from it.
- Step-04:Eliminate all the intermediate states one by one.
- These states may be eliminated in any order.
- In the end,Only an initial state going to the final state will be left.
- The cost of this transition is the required regular expression.
- In case of DFA make it free from non productive state then find RE from rest of state

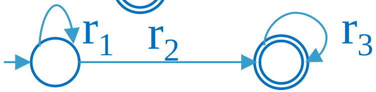


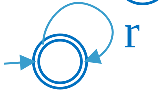
- Eliminate states until the transaction graph take anyone of fallowing form and obtain regular expression .

1)R.E.=r 

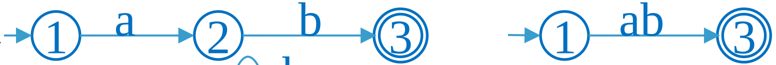
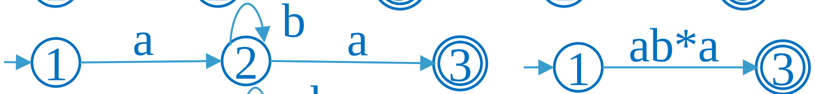
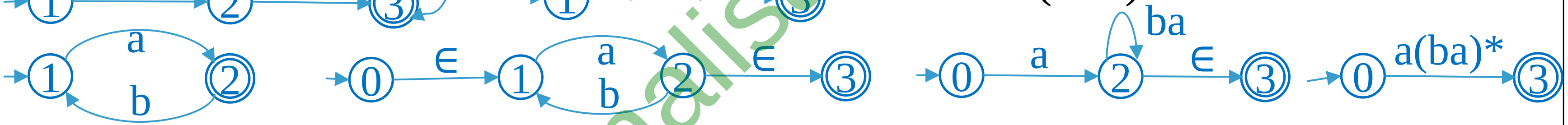
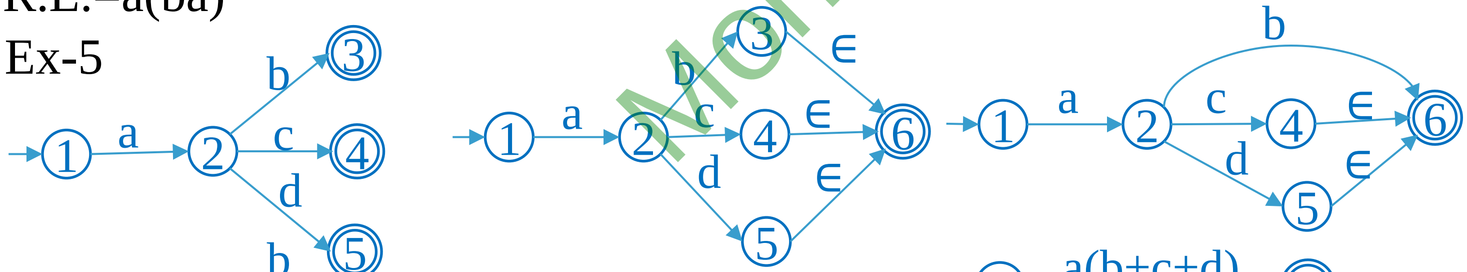
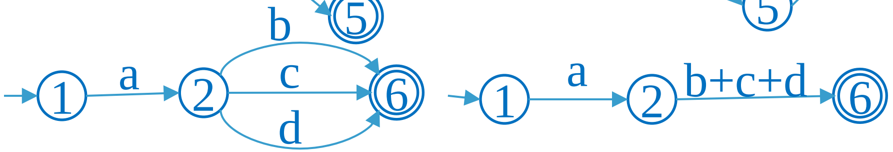
2)R.E.=r_1*r_2 

3)R.E.=r_1r_2* 

4)R.E.=r_1*r_2 r_3* 

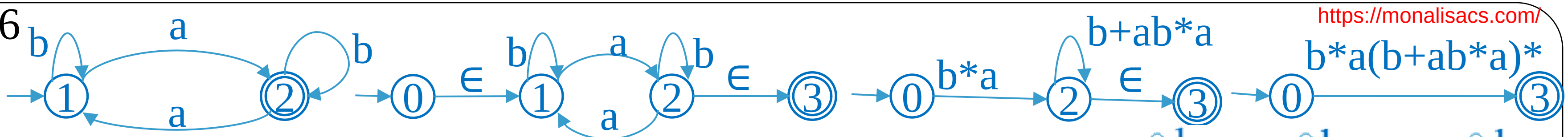
5)R.E.=r* 

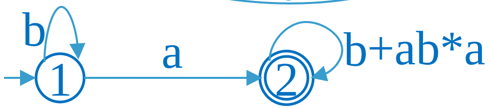
6)R.E.= (r_1+r_2)* 

- Ex-1 
R.E.= ab
- Ex-2 
R.E.= ab*a
- Ex-3 
R.E.= a(a+b)*ba*
- Ex-4 
R.E.=a(ba)*
- Ex-5 



R.E.= a(b+c+d)

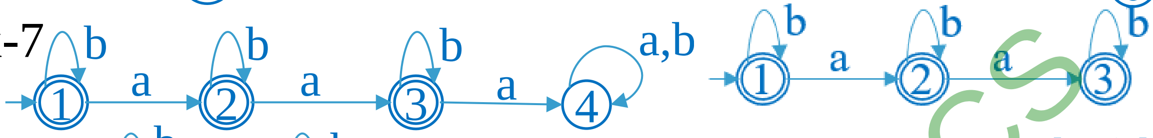
- Ex-6

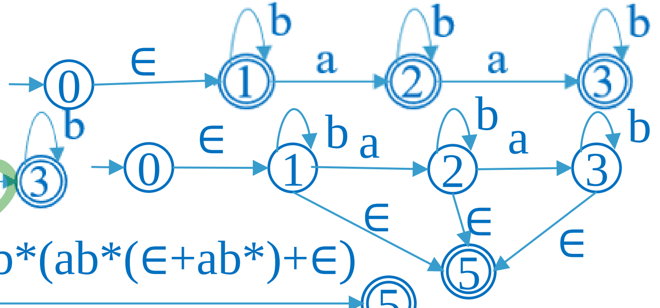


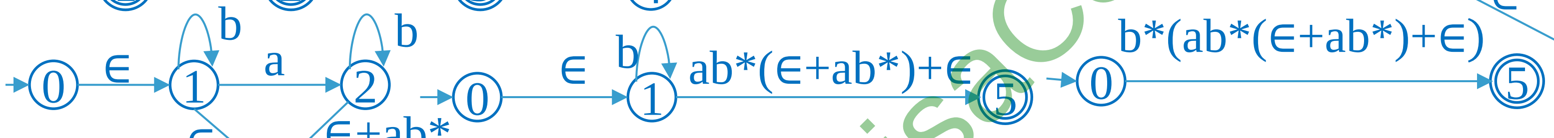


- R.E.=b*a(b+ab*a)*

- Ex-7







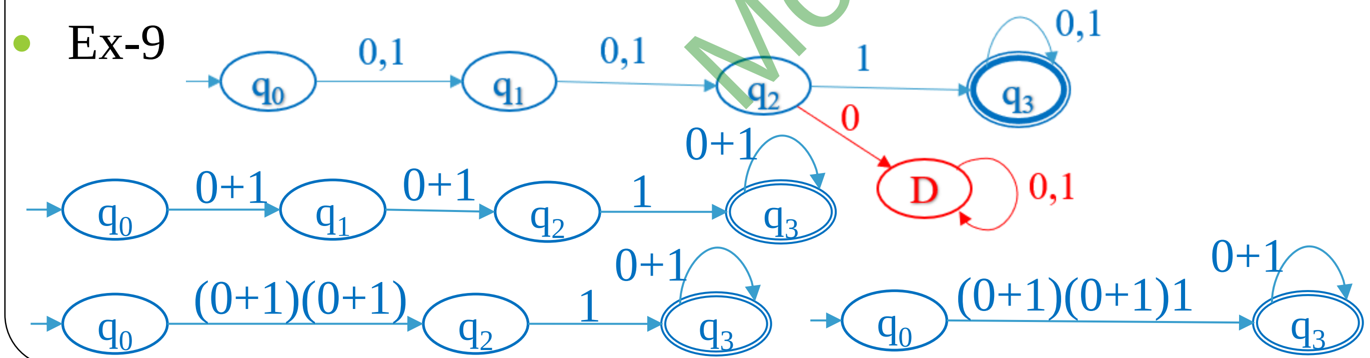
- R.E.=b*(ab*(∈+ab*)+∈)

- R.E.=1*01*0(0+1)*

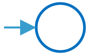
- Ex-8

- R.E.=(0+1)(0+1)1(0+1)*
- R.E.=$(0+1)^2 1(0+1)^*$

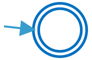
- Ex-9



- **Conversion of Regular Expression to Finite Automata**
- 1.Method of Synthesis
- 2.Method of Decomposition (State Creation Method)
- <u>1.Method of Synthesis</u>
- R.E.=(ϕ)

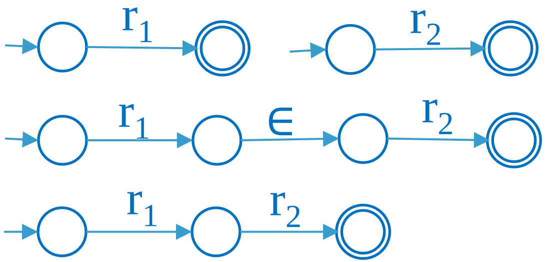


- R.E.=($\in$)

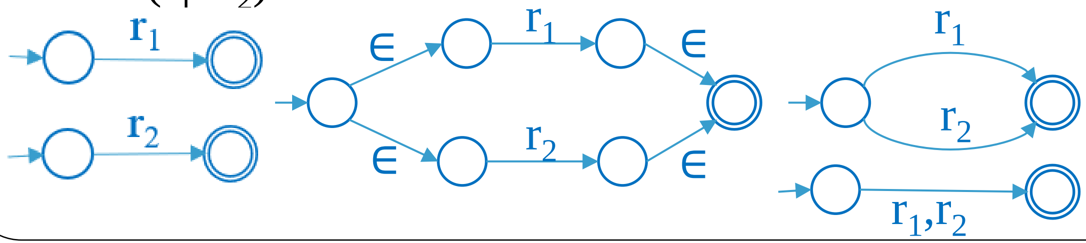


- R.E.=(r)

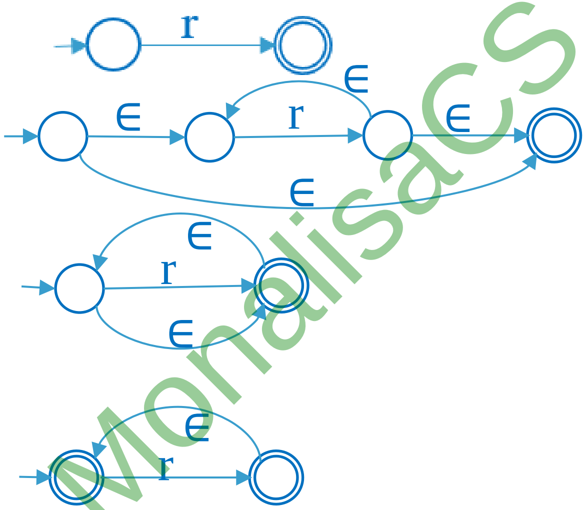
- R.E.=($r_1.r_2$) Concatenation



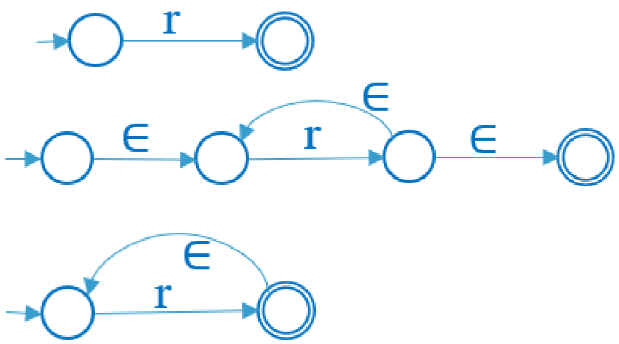
- R.E.=(r_1+r_2) Union



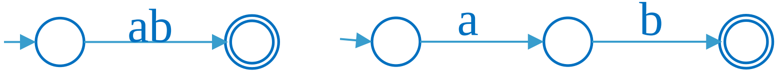
- R.E.=(r)* Kleene closure



- R.E.=$(r)^+$ positive closure



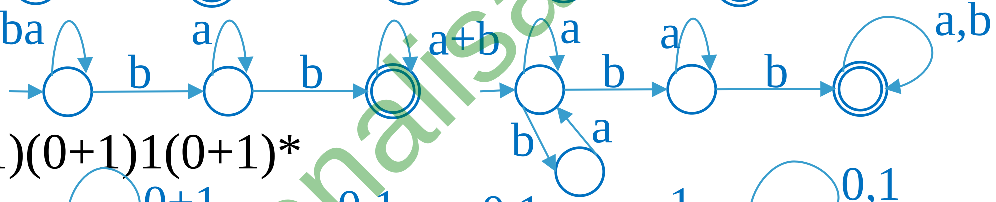
- <u>2.Method of Decomposition (State Creation Method)</u>
- Write RE r as an edge level for the NFA with two state.
- Spilt r into symbol and create the state.
- Continue the state creation process until RE is divided into symbols.
- r=ab



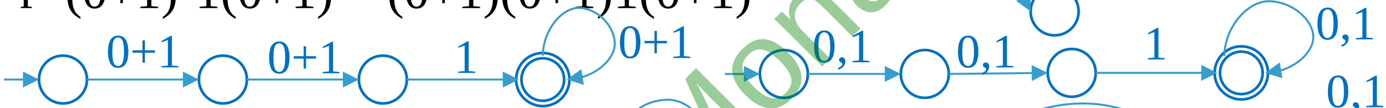
- r=ab*

- r=(ab)*ba*b

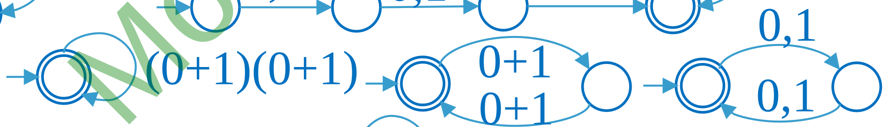
- r=(a+ba)*ba*b(a+b)*



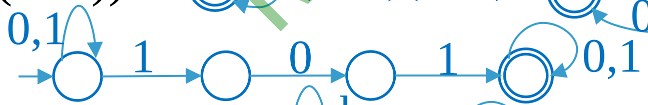
- r=(0+1)²1(0+1)*=(0+1)(0+1)1(0+1)*



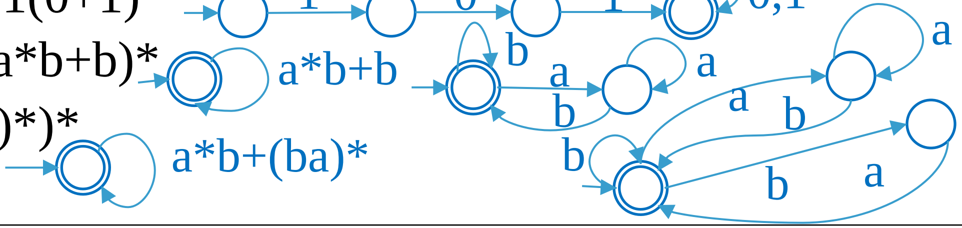
- r=((0+1)²)*=((0+1)(0+1))*



- r=(0+1)*101(0+1)*



- r=(a*b)*=(a*b+b)*
- r=(a*b+(ba)*)*



- **Algebraic properties of Regular Expression(*,.,+)**
- <u>1.Closer</u>: If r_1 & r_2 are RE then r_1*, $r_1.r_2$, r_1+r_2 are Regular. All are closer w.r.t. *,.,+
- <u>2.Associative</u>:+,. Satisfy $(r_1+r_2)+r_3 = r_1+(r_2+r_3)$, $(r_1.r_2).r_3 = r_1.(r_2.r_3)$
- <u>3.Identity:</u> Let x=identity element r+x=r , r.x=r ,ϕ identity w.r.t + & ∈ identity w.r.t .
- <u>4.Annihilator</u>: r.x=x , ϕ is annihilator w.r.t . ,No annihilator w.r.t +.
- <u>5.Idempotent</u>: r+r=r, satisfy w.r.t. + but not .
- <u>6.Commutative</u> : $r_1+r_2 = r_2+r_1$,$r_1.r_2 \neq r_2.r_1$,+ Satisfy not .
- 7.Distributive :$r_1.(r_2+r_3)=r_1.r_2 + r_1.r_3$, $(r_1+r_2).r_3 = r_1.r_3 + r_2.r_3$. Is distributive over + but + is

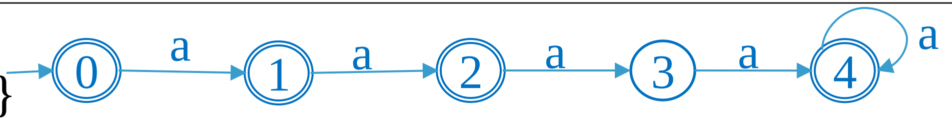
If r_1 and r_2 two RE then $(r_1+r_2)^* = (r_1^*+r_2)^* = (r_1+r_2^*)^* = (r_1^*+r_2^*)^* = (r_1^*.r_2^*)^*$

- Q1.Let A=(1*0+0)*,B=(1*0)* Which is true? i)A⊂B ii)B⊂A iii)A=B
- A=(1*0+0)*=((1*+ ∈)0)*=(1*0)*=B
- Ans: A=B
- Q2.Let A =(1*0*+0*1*)*,B=(1*+0)* Which is true? i)A⊂B ii)B⊂A iii)A=B
- A =(1*0*+0*1*)*=((1*0*)*+(0*1*)*)*
- ((1*+0*)*+(0*+1*)*)*
- =(1*+0*)*=(1*+0)*=B
- Ans: A=B

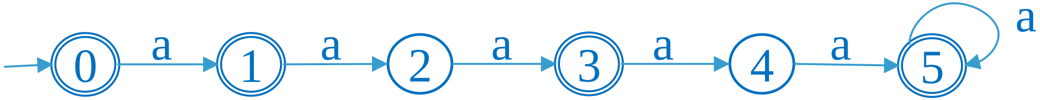
- Q3.Let A =(10*+1)*,B=(10)* Which is true? i)A⊂B ii)B⊂A iii)A=B
- A =(10*+1)*=(10*)* ,B=(10)* So B⊂A
- Q4.Let A =1(0+1)*,B=10*,C=10*0 Which is true?
- i)A⊂B ,B ⊂C ii)B⊂A ,B ⊂C iii)C ⊂A,C ⊂B iv)B ⊂A,A ⊂C
- A ={1,10,11,100,101,110,….},B={1,10,100,1000,…..},C={10,100,1000,…..}
- Ans: iii)C⊂A,C⊂B is true.
- Q5.Which of the following is true?
  a)$(r_1^*+r_2)^*=(r_1^*.r_2)^*$ b)$(r_1+r_2^*)^*=(r_1.r_2^*)^*$ c) $(r_1+r_2)^*=(r_1^*.r_2^*)^*$ d) $(r_1^*+r_2^*)^*=(r_1.r_2)^*$
- Ans: c) $(r_1+r_2)^*=(r_1^*.r_2^*)^*$
- Q6.Number of state in minimal DFA that accept language of RE =(0+1)(0+1)….(0+1) n time. a)n+1 b)n+2 c)2^n d)n
- Ans: b)n+2
- Q7.Which is correct a)(xx)*y=x(xy)* b)y(xy)*=(yx)*y c)x(xy)*=(xx)*y d)(yx)*=(xy)*
- Ans:b)y(xy)*=(yx)*y
- Q8.Which of following RE does not contain substring 100
- a)0*1*0* b)0*1*01* c)0*1*(0+1)* d)0*1*0*1
- Ans: b)0*1*01*

Single Alphabet DFA

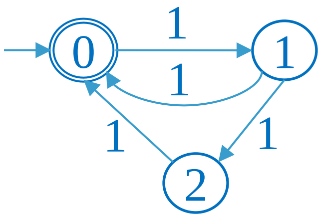
- $L_1=\{a^n|n\geq0,n\neq3\},\Sigma=\{a\}$



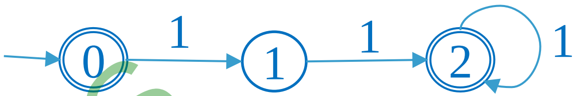
- $L_1=\{\in,a,aa,aaaa,aaaaa,\ldots.\}$

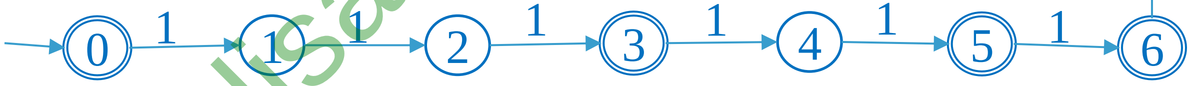


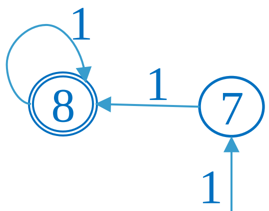
- # state =5
- $L_2=\{a^n|n\geq0,n\neq2,n\neq4\},\Sigma=\{a\}$



- $L_2=\{\in,a,aaa,aaaaa,aaaaaa,\ldots.\}$
- # state =6
- R.E.$(L_3)=(11+111)^*,\Sigma=\{1\}$

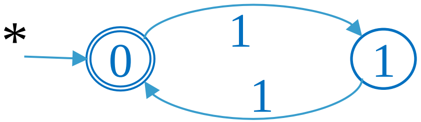






- $(11)^*$,#1=0,2,4,6,8,10,…
- $(111)^*$, #1=0,3,6,9,12,…
- $(11+111)^*$,#1=0,2,3,4,5,6,7,8,9,…
- # state =3
- R.E.$(L_4)=(111+11111)^*,\Sigma=\{1\}$
- $(111)^*$,#1=0,3,6,9,12,…
- $(11111)^*$, #1=0,5,10,15,20,…
- $(111+11111)^*$,#1=0,3,5,6,8,9,10,11,12,…
- # state =9
- R.E.$(L_5)=(11+1111)^*,\Sigma=\{1\}$

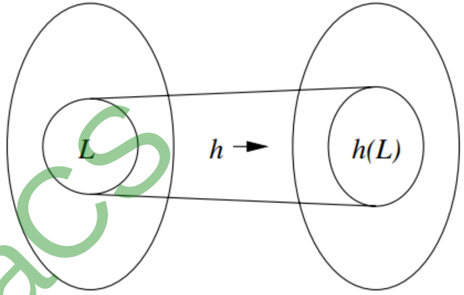
- $(11+1111)^*=(11)^*$



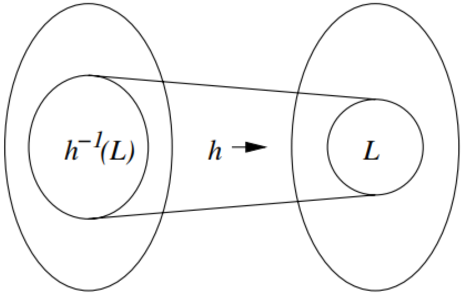
- # state =2

- **Closer property of Regular Language**
- Regular language satisfy closer property w.r.t. following operator.
- Unary: Complement ,Kleene Closer,+ve closer ,Reverse, Prefix,Suffix.
- Binary: Union, Intersection, Concatenation, Difference , Symmetric difference, Quotient operator , Homomorphism, Inverse Homomorphism ,
- Quotient operator: If L_1,L_2 be two RL then L_1/L_2 is also RL.
- $x.y \in L_1$ for some $y \in L_2$,$x \in L_1/L_2$, $L_2 \subset L_1$
- Ex: $\frac{1001}{01} = 10, \frac{1101}{11} = \emptyset$
- $\frac{0*1}{01}$ =0* $\frac{1}{01}, \frac{01}{01}, \frac{001}{01}, \frac{0001}{01}, \ldots\ldots$
- $\frac{10*}{101*}$ =∈ $\frac{1}{10}, \frac{10}{10}, \frac{100}{10}, \frac{10}{101}, \ldots\ldots\ldots$
- $\frac{10*1}{0*1}$ =10* $\frac{11}{1}, \frac{101}{1}, \frac{10001}{01}, \frac{10001}{0001}, \ldots$

- <u>Homomorphisms</u> : Let Σ & Δ be two alphabet then Homomorphisms is a mapping from $\Sigma \rightarrow \Delta$ s.t the symbol of Σ is replaced by single string of another alphabet Δ.
- Substitution of strings for symbols.
- Ex: Σ={a,b}, Δ={0,1} ,h(a)=0,h(b)=10
- L={ab,ba},h(L)={h(ab),h(ba)}={010,100}
- L=ab*,h(L)=h(ab*)=0(10)*
- L=(a*b)*,h(L)=h((a*b)*)=(0*10)*



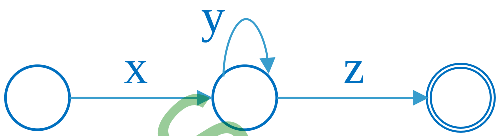
- <u>Inverse Homomorphisms</u> :Let h: $\Sigma \rightarrow \Delta$ is a Homomorphisms & L is a RL over alphabet Δ then $h^{-1}(L)=\{x|h(x) \in h(L)\}$ is also a RL over alphabet Σ.
- $h^{-1}(L)$ called Inverse Homomorphisms of h
- Ex: Σ={a,b}, Δ={0,1} ,h(a)=01,h(b)=0
- L={01,00,010,11,110} , $h^{-1}(L)$={a,bb,ab}
- L={0100,100,001,101}, $h^{-1}(L)$={abb,ba}
- L=0*1, $h^{-1}(L)$=b*a



- **General Properties of Regular Language:**
- Every Finite Language is regular.
- An Infinite language can be RL or NRL.
- Every NRL is infinite. A RL can be finite or infinite.
- Every Regular set is Countable.
- Every subset of RL need not be Regular.
- Ex $L_1=\{a^m b^n | m,n>0\}$ RL, $L_2=\{a^m b^n | m=n\}$ NRL $L_2 \subset L_1$
- Every finite subset of Regular & NRL is always Regular.
- Every subset of a NRL set need not be NRL may be RL or NRL.
- Union of finite collection regular set is always regular.
- Union of infinite collection of regular set need not be regular.
- Intersection of finite collection of RL is always regular.
- Intersection of infinite collection of regular set need not be regular.

Pumping Lemma

- If L is a RL(Infinite).There exists a pumping length n s.t for every string $w \in L, |w| \geq n$.
- We can break w into 3 strings ,w=xyz
- 1.$|y| \neq 0$
- 2.$|xy| \leq n$
- 3.$xy^k z \in L, \forall k \geq 0$

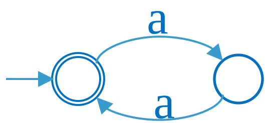


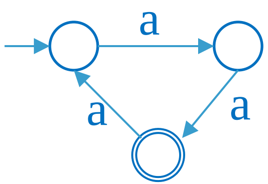
- Pumping Lemma is used to prove some of language is non regular.
- Every infinite RL satisfy pumping lemma property.
- NRL→ NRL
- It is technique to prove non regularity.
- The language which doesn’t satisfy pumping lemma property is non regular.
- <u>Process of Pumping Lemma:</u>
- 1.Assume that L is regular.
- 2.Select $w \in L$ s.t. $|w| \geq n$
- 3.Split w into 3 part $|xy| \leq |w|$ and $|y| \neq 0$
- 4.If there exist at least one value for k s.t. $xy^k z \notin L$
- Then L does not satisfy PL property
- Which is a contradict .Hence L is NRL.

- Ex 1:Prove that $L=\{a^n b^n \mid n \geq 0\}$ is a NRL
- Let $w= a^n b^n$,$x=a^{n-1}$,$y=a$,$z=b^n$
- $|y|=|a|\neq 0$
- $|xy| \leq |w|$
- Now check $xy^k z \in L, \forall k \geq 0$
- Let k=0 , $xy^0 z= a^{n-1}\, \varepsilon\, b^n = a^{n-1} b^n \notin L$
- Let k=1, $xy^1 z= a^{n-1}\, a\, b^n = a^n b^n \in L$
- Let k=2, $xy^2 z= a^{n-1}\, a^2\, b^n = a^{n+1} b^n \notin L$
- So L is NRL.
- Or w=aaabbb ,x=aaa, y=b, z=bb [x=aaa,y=bbb,z= ε][x= ε,y=aa,z=abbb]
- $|y|=|b|\neq 0$
- $|xy|=|aaab|=4 \leq |w|=6$
- Now check $xy^k z \in L, \forall k \geq 0$
- Let k=0 , $xy^0 z$=aaa ε bb=aaabb ∉ L
- Let k=1, $xy^1 z$=aaa b bb=aaabbb ∈ L
- Let k=2, $xy^2 z$=aaa bb bb=aaabbbb ∉ L
- So L is NRL.

- Ex 2: $L=\{ww^R|w \in \{0,1\}^*\}$
- $W=001\ 100$, $x=001$, $y=1$, $z=00$
- Now check $xy^kz \in L, \forall k \geq 0$
- Let $k=0$, $xy^0z=001\ \varepsilon\ 00=00100 \notin L$
- Let $k=1$, $xy^1z=001\ 1\ 00=001100 \in L$
- Let $k=2$, $xy^2z=001\ 11\ 00 = 0011100 \notin L$
- So L is NRL.
- Ex 3: $L=\{a^mb^n|m,n\geq 0\}$
- $w=aabbb$, $x=aa$, $y=b$, $z=bb$
- Now check $xy^kz \in L, \forall k \geq 0$
- Let $k=0$, $xy^0z=aa\ \varepsilon\ bb=aabb \in L$
- Let $k=1$, $xy^1z= aa\ b\ bb=aabbb \in L$
- Let $k=2$, $xy^2z=aa\ bb\ bb=aabbbb \in L$
- Let $k=3$, $xy^3z=aa\ bbb\ bb=aabbbbb \in L$
- L is a Regular Language.

- **Weak form of Pumping lemma**
- If L is any language defined over the alphabet Σ with only one symbol s.t. the length of string of L are in some AP then L is RL

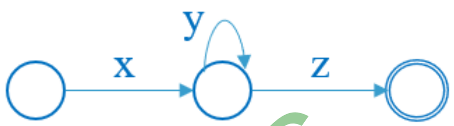




- $L_1=\{a^{2n}|n\geq 0\}$RL [0,2,4,6…....AP]
- $L_2=\{a^{3n+2}|n\geq 0\}$RL [2,5,8,11…..AP]
- $L_3=\{a^{2n-5}|n\geq 3\}$RL [1,3,5,7, …..AP]
- $L_4=\{a^{n^2}|n\geq 0\}$NRL [0,1,4,9,…..Not in AP]
- $L_5=\{a^{n^2+1}|n\geq 0\}$NRL [1,2,5,10,....Not in AP]
- $L_6=\{a^{2^n}|n\geq 0\}$NRL [1,2,4,8, ….Not in AP]
- $L_7=\{a^{n!}|n>0\}$NRL [1,2,6,24,…Not in AP]
- $L_8=\{a^p|p$ is a +ve prime$\}$NRL [1,3,5,7,11, ….Not in AP]
- Lets check L_1 by pumping Lemma.
- W=aaaa ,x=aa,y=a,z=a
- Now check $xy^kz\in L,\forall k\geq 0$
- Let k=0 , xy^0z=aa ∈ a=aaa ∉ L
- Let k=2, xy^2z= aa aa a=aaaaa ∉ L
- So L is NRL.

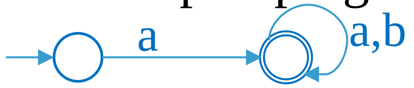
Pumping Lemma

- If L is a RL(Infinite).There exists a constant n s.t for every string $w \in L, |w| \geq n$.
- We can break w into 3 strings ,w=xyz
- 1.$|y| \neq 0$
- 2.$|xy| \leq n$
- 3.$xy^kz \in L, \forall k \geq 0$



➢ Pumping Length

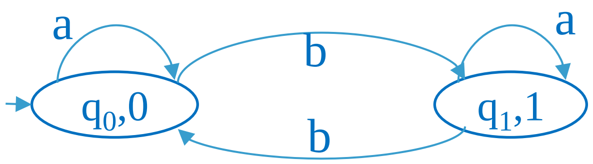
- Pumping length for a regular language makes sure that any string in that language with the length ≥pumping length has some repetition. n is the pumping length.
- L_1={Every string start with ‘a’}={a,aa,ab,aba,….}



- $|w| \geq 1$,Min w=a then x=∈,y=a,z= ∈ ,Pumping length=1
- L_2={Every string contain substring ‘aba’}={aba,aaba,bbabaab,..},Pumping length= 3
- L_3={The 3[rd] symbol from left end is ‘1’}={001,101,011,1011,….},Pumping length=3
- $L_4 = \{a^{2+3k} \mid k \geq 0\} = \{a^2, a^5, a^8, a^{11} \ldots\ldots\}$,Pumping Length=2
- $L_5 = \{ b^{10+12k} \mid k \geq 0\} = \{b^{10}, b^{22}, b^{34}, b^{46}, \ldots\ldots\}$,Pumping length=10
- $L_6 = \{a^m b^n \mid m \geq 2, n \geq 3\} = \{a^2b^3, a^5b^8, a^{20}b^{34}, \ldots\}$, Pumping length=5
- L_7={Every string ends with ‘abba’}={abba,baabba,abbabba,…….} Pumping length=4

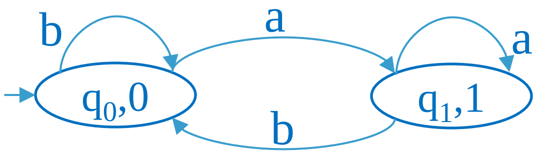
FA with Output

- Both Moore & Mealy M/C are special case of DFA.
- Moore & Mealy M/C are output producer rather than Language accepter So no need to define final state.
- **Moore Machine**
- FA where o/p is associated with state is called Moore M/C
- A Moore machine can be described by a 6 tuple (Q, Σ, Δ, δ, λ, q_0) where
- **Q** is a finite set of states.
- Σ(Upper Sigma) is a finite set of symbols called the input alphabet.
- Δ(Upper Delta) is a finite set of symbols called the output alphabet.
- **δ** (Lower Delta)is the input transition function where $\delta: Q \times \sum \rightarrow Q$
- λ (Lower Lambda) is the output transition function where $\lambda : Q \rightarrow \Delta$
- $\mathbf{q_0}$ is the initial state from where any input is processed.
- **Representation of Moore M/C**
- It can be represented in two way
- 1)Transition Diagram. 2)Transition Table.
- input=aba , output=0011, input= baab ,output=01110
- If |input|=n then |output|=n+1



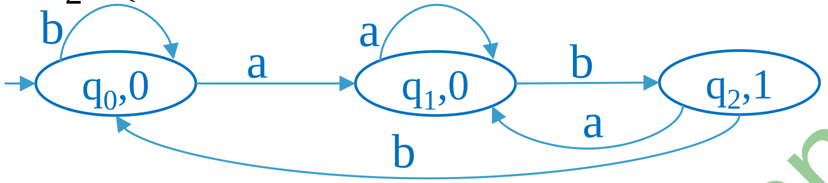
	a	b	λ
q_0	q_0	q_1	0
q_1	q_1	q_0	1

- Output depends on state.
- Moore machine respond for empty string ∈ adding extra output at initial state.
- $\lambda(\epsilon)=\lambda(q_0)$
- L_1={Count number of a's in the given input string}, Σ ={a,b}, Δ ={0,1}



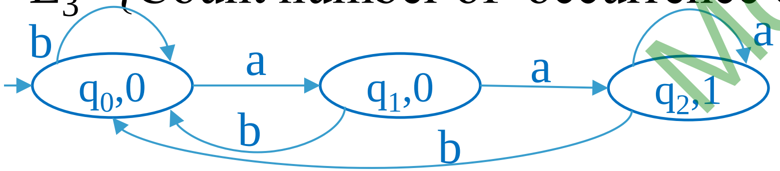
- λ (aba)=0101 ,# a=2
- λ (abaaba)=0101101 ,# a=4
- DFA= {end with 'a'}

- L_2={Count number of occurrence of substring 'ab'}, Σ ={a,b}, Δ ={0,1}



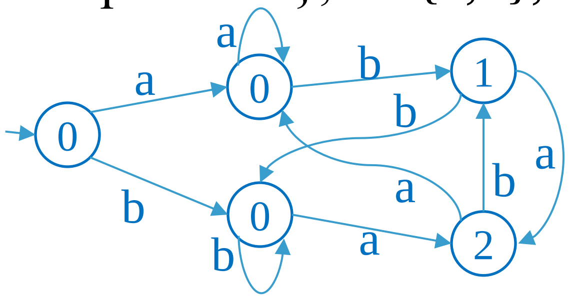
- λ (aababb)=0001010 ,# ab =2
- DFA= {end with 'ab'}

- L_3={Count number of occurrence of two consecutive 'a'}, Σ ={a,b}, Δ ={0,1}



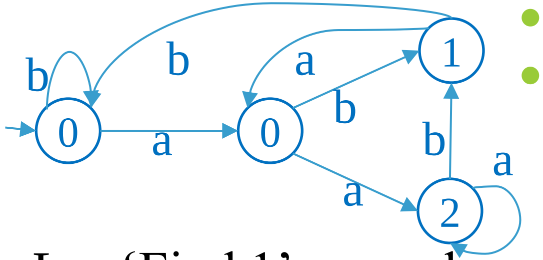
- λ (abaaaaba)=000011100 ,# aa =3
- DFA= {end with 'aa'}

- L_4={output is '1' if input is 'ab' ,2' if input is 'ba',otherwise output is '0'}, Σ ={a,b}, Δ ={0,1,2}

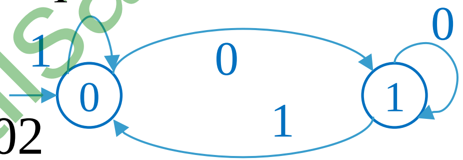


- λ (aab)=0001
- λ (babb)=00210
- λ (aaba)=00012

- L_5={output is '1' when input is 'ab' ,'2' if input is 'aa',otherwise output is '0'}, Σ ={a,b}, Δ ={0,1,2}

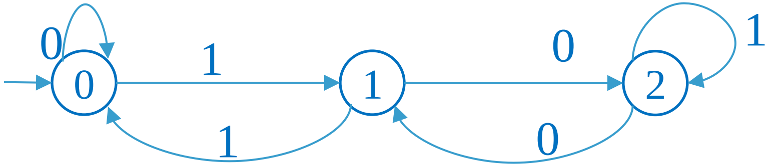


- λ (aab)=0021
- λ (babaa)=000102



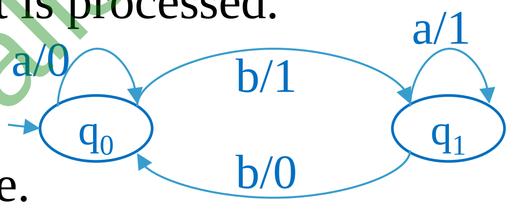
- λ (011)=0100
- λ (10100)=001011

- L_6={Find 1's complement of binary number}, Σ ={0,1}, Δ ={0,1}



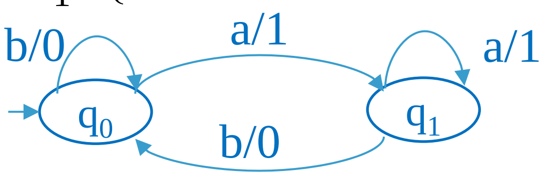
- L_7={Produce reminder when a binary number is divisible by 3},Σ ={0,1},Δ ={0,1,2}
- λ (011)=0010 , λ (101)=0122, λ (1100)=01000

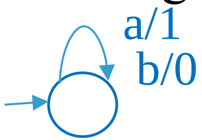
- **Mealy Machine**
- FA where o/p is associated with transition is called Mealy machine
- A Mealy machine can be described by a 6 tuple $(Q, \Sigma, \Delta, \delta, \lambda, q_0)$ where
- **Q** is a finite set of states.
- Σ(Upper Sigma) is a finite set of symbols called the input alphabet.
- Δ(Upper Delta) is a finite set of symbols called the output alphabet.
- **δ** (Lower Delta)is the input transition function where $\delta: Q \times \Sigma \rightarrow Q$
- λ (Lower Lambda) is the output transition function where $\lambda : Q \times \Sigma \rightarrow \Delta$
- **q_0** is the initial state from where any input is processed.
- **Representation of Mealy M/C**
- It can be represented in two way
- 1)Transition Diagram. 2)Transition Table.
- input=aba , output=011,
- input= baab ,output=1110
- |input|=|output|=n
- Output depends on state and input symbol
- Mealy Machine can't respond for empty string $\in, \lambda(\in) = \in$



	a	λ	b	λ
q_0	q_0	0	q_1	1
q_1	q_1	1	q_0	0

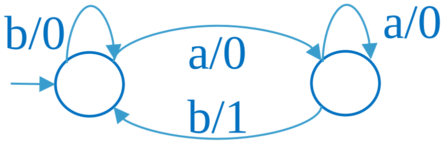
- L_1={Count number of a's in the given input string}, Σ ={a,b}, Δ ={0,1}





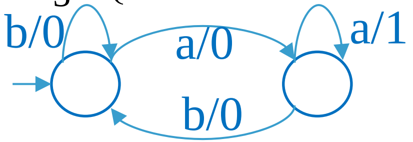
- a=1,b=0
- λ (aba)=101 ,# a=2
- λ (abaaba)=101101 ,# a=4

- Two state p,q are equal if both of them go to same state with same output sequence.
- Mealy Machine take less number of state than DFA and Moore Machine.
- L_2={Count number of occurrence of substring 'ab'}, Σ ={a,b}, Δ ={0,1}



- ab=1
- λ (aababb)=001010 ,# ab =2

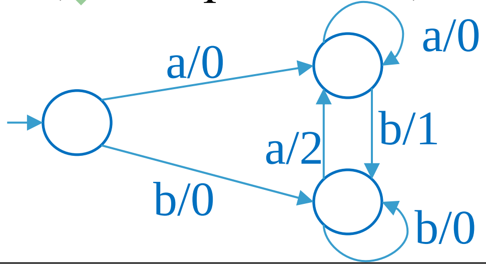
- L_3={Count number of occurrence of two consecutive 'a'}, Σ ={a,b}, Δ ={0,1}



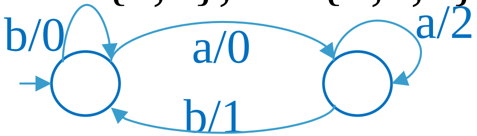
- aa=1
- λ (abaaaaba)=00011100 ,# aa =3

- L_4={output is '1' if input is 'ab','2' if input is 'ba',otherwise output is '0'},
- Σ ={a,b}, Δ ={0,1,2}
- ab=1
- ba=2

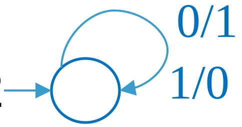
- λ (aab)=001
- λ (babb)=0210
- λ (aaba)=0012



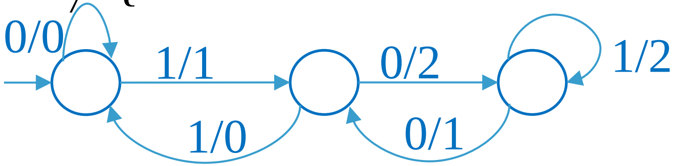
- L_5={output is ‘1’ when input is ‘ab’ ,’2’ if input is ‘aa’, otherwise output is ‘0’}, Σ ={a,b}, Δ ={0,1,2}



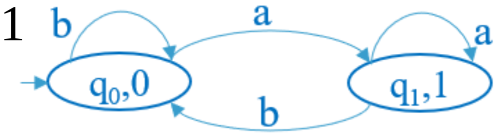
- ab=1
- aa=2
- λ (aab)=021
- λ (babaa)=00102

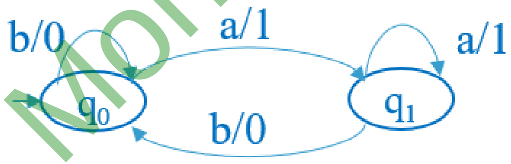


- λ (011)=100
- λ (1010)=0101
- 0→1,1→ 0
- L_6={Find 1’s complement of binary number}, Σ ={0,1}, Δ ={0,1}
- L_7={Produce reminder when a binary number is divisible by 3},Σ ={0,1},Δ ={0,1,2}



- λ (011)=010 , λ (101)=122, λ (1100)=1000
- **Conversion of Moore machine to Mealy machine**
- Remove the output from state and add with edge
- Ex-1





	a	b	λ
q_0	q_1	q_0	0
q_1	q_1	q_0	1

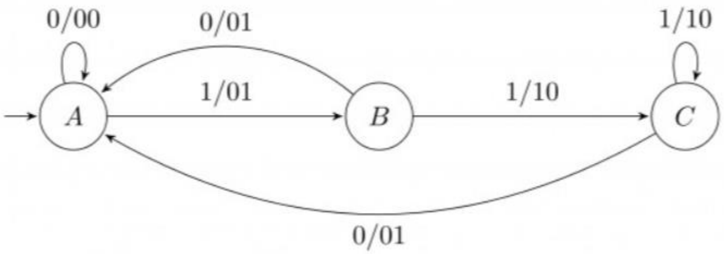
	a	λ	b	λ
q_0	q_1	1	q_0	0
q_1	q_1	1	q_0	0

- **Conversion of Mealy machine to Moore machine**
- Remove the output from edges and add with state.
- GATE-CS-2002,Q-5,1Mark
- The Finite state machine described by the following state diagram with A as starting state, where an arc label is x / y and x stands for 1-bit input and y stands for 2- bit output

	a	λ	**b**	λ
q_0	q_2	0	q_1	0
q_1	q_1	0	q_0	0
q_2	q_0	1	q_2	1

	a	**b**	λ
q_{00}	q_{20}	q_1	0
q_{01}	q_{20}	q_1	1
q_1	q_1	q_{00}	0
q_{20}	q_{01}	q_{21}	0
q_{21}	q_{01}	q_{21}	1

- **(A)** Outputs the sum of the present and the previous bits of the input.
- **(B)** Outputs 01 whenever the input sequence contains 11.
- **(C)** Outputs 00 whenever the input sequence contains 10.
- **(D)** None of these



- Let input string 110.
- Output 01 10 01
- previous input bit + present input bit = output
- Ans:**(A)** Outputs the sum of the present and the previous bits of the input.