UNIT-2. REGULAR EXPRESSION AND LANGUAGES

Topics to be covered :

Regular Expression and Lnguages : *Regular Expression, Finite Automata and Regualar Expressions, Properties of Regualar Languages (RL) : Proving Languages not to be regular. Equivalence and minimization of Automata(DFA). Applications of Regular Expressions*

Upon completion you will be able to

- **Explain the concepts of Regular expression and its Applications.**
- **Explain connection between Regualar expression and Finite Automata**
- ***Demonstrate the equivalence of*** **DFA, NFA and *Ɛ-NFA*s.**
- **Prove Cetain Languages are not Regular using Pumping Lemma**

Regualar Expressions and Languages

1. Introduction to Regular expression
   - 1.1. Formal Defination for Regualar Expression.
   - 1.2. Language associated with Regular Expression
   - 1.3. Building Regular Expression for the Languages
2. Finite Automata and Regular expressions
   - 2.1. Building Ɛ-NFA from Regular Expression
     - 2.1.1. Examples on building Ɛ-NFA from Regular Expressions (Thomson's Scheme)
   - 2.2. Building Regular Expressions from DFA and NFA (State Elimination Method).
3. Minimization Of Automata(DFA)
   - Algorithm and Examples
4. Properties of Regular Languages
   - Pumping Lemma
   - Proving Languages not be Regular
   - Examples
5. Applications Of Regular Expressions

1. Introduction to Regular expression

- We know that DFA, NFA and Ɛ-NFA are three mechanism to describe the Regular Languages. These descriptions are machine like descriptions and are not a declarative way to describe the the Regular Languages.

- Regular expression is another mechanism to describe the Regular Languages which is algebraic like descriptions and is more declarative way to describe the the Regular Languages than DFA.

- Informally, Regular expression is a notation to describe the Regular Languages that involves a combination of strings of symbols from some alphabet ∑, paranthesis and operators such as +(UNION), • (Concatenation) and * (Star Closure).
  - Example-1. Simplest case **L = {a},** the regular expression is → **a**
  - Example-2. **L = {a, b}**, the regular expression is → **a+b** or **(a+b)**
  - Example-3**. L = {Ɛ, a, aa,aaa,aaaa,..........}** , the regular expression is → **a***

1.1. Formal Defination of Regular Expression :

Formally, Regualar Expression can be defined as follows :

Let ∑ be a given **alphabet**. Then

Rule -1. Ø, Ɛ and **a € ∑** are all **Regular expressions**. These are all called **primitive Regular Expressions.**

Rule - 2. if **R1** and **R2** are Two **Regular Expressions**, so are **R1 + R2, R1 • R2, R1* and (R1).**

Rule - 3. A string is Regular Expression if and only if it can derived from the **primitive regular expressions (mentioned in Rule - 1)**, by finite number of application of the **Rules mentioned In Rule - 2.**

Example -1. ∑ = { a b } be a given **alphabet**

The string - (a+b) is a **Regular Expression,** since it is constructed by applications of the above mentioned Rules, i.e Here R1=a,R2=,b are **primitive regular expressions and Rule -2 is** is applied **2 number of times (R1+R2 - Union and (R1) - Paranthesized Regular Expression) to get a string - (a+b)** .

Example -2. ∑ = { a,b,......z} be a given **alphabet,**

The string - sachin is a **Regular Expression,** since it is constructed by applications of the above mentioned Rules, i.e Here R1=s,R2=a,R3=c,R4=h,R5=i and R6=n are **primitive regular expressions and Operator •** is applied **5 number of times (s•a•c•h•i•n) to get a string sachin (In between alphabets • is implied) .**

Priority of the Operators :

+ → Union Operator → Least Precedence
• → Concatenation Operator → next Least Precedence
*** → Star closure Operator → Highest Precedence**

Some examples of Regular expressions and their assicated language :

→

Regular expressions	Meaning
(a+b)*	Set of strings of a’s and b’s of any length including the NULL string.
(a+b)*abb	Set of strings of a’s and b’s ending with the string abb
ab(a+b)*	Set of strings of a’s and b’s starting with the string ab.
(a+b)*aa(a+b)*	Set of strings of a’s and b’s having a sub string aa.
a*b*c*	Set of string consisting of any number of a’s(may be empty string also) followed by any number of b’s(may include empty string) followed by any number of c’s(may include empty string).
$a^+b^+c^+$	Set of string consisting of at least one ‘a’ followed by string consisting of at least one ‘b’ followed by string consisting of at least one ‘c’.
aa*bb*cc*	Set of string consisting of at least one ‘a’ followed by string consisting of at least one ‘b’ followed by string consisting of at least one ‘c’.
(a+b)* (a + bb)	Set of strings of a’s and b’s ending with either *a*or *bb*
(aa)*(bb)*b	Set of strings consisting of even number of a’s followed by odd number of b’s
(0+1)*000	Set of strings of 0’s and 1’s ending with three consecutive zeros(or ending with 000)
(11)*	Set consisting of even number of 1’s

1.2. Language associated with Regular Expression :

If **R** is a Regular Expression then **L(R)** denote the **Language(Regular)** associated with the Regular Expression **R.** The **Language L(R)** denoted by Regular Expression **R** is defined by the following rules.

1. **Ø** is a Regular Expression denoting the empty language i.e **L(R) = Ø** .
2. **Ɛ** is a Regular Expression denoting the language containing empty string - **Ɛ** i.e **L(R) = { Ɛ }**
3. For every **a € ∑, a** is a Regular Expression denoting the language containing **a.** i.e **L(R) = { a }.**

If **R1** and **R2** are Regular Expressions then

4. **L(R1 + R2)** = **L(R1)** Ụ **L(R2)** → **i.e Union of langaugues associated with R1** and **R2**
5. **L(R1 • R2)** = **L(R1) • L(R2)** → **i.e Concatenation of langauges associated with R1** and **R2.**
6. **L((R1))** = **L(R1)** → **i.e It is just a langauge associated with R1**
7. **L(R1*) = (L(R1))*** → **i.e It is just a STAR closure of a langauge associated with R1**

Example -1. $\sum = \{a, b\}$ be a given **alphabet** then
Demonstate the Language associated with Regular expression
R = (a*• (a+b))

Answer : L(R) = L((a*• (a+b)))
= L(a*) • L(a+b)
= (L(a))* • (L(a) Ụ L(b))
= {Ɛ, a, aa, aaa,.....} • {a, b}
= {a, aa, aaa, aaaa,......., b, bb, bbb, bbbb,.........}

Example -2. $\sum = \{a, b\}$ be a given **alphabet** then
Demonstate the Language associated with Regular expression
R = (a + b)* • (a + bb)

Answer : L(R) = L ((a + b)*•(a + bb))
= L((a+b)*) • (L(a) Ụ L(bb))
= (L(a+b))* • { a, bb }
= { Ɛ, a, aa,... b, bb,...., aab, aaab...,.....} • {a, bb}
= { a, aa,aaa,... ba, bba,.., aaba, aaaba,...., bb, abb,aabb,......}

Exercise Problems : Demonstate the Language associated with Regular expression : **1. R = (aa)* (bb)* b. 2. R=(a+b)* aa(a+b)***

1.3. Building Regular Expression for the Langauges :

Write Regular Expressions for the following Language descriptions on the Alphabet ∑ = { a, b }

SET-1

1. The set of all strings that begin with bba
2. Strings of a’s and b’s with Length two.
3. Strings of a’s and b’s with length <= 2
4. Strings of a’s and b’s having even length.
5. Strings of a’s and b’s having odd length.
6. Strings of a’s and b’s having at least three consecutive (aaa) a’s.
7. Strings of a’s and b’s starting ‘a’ and ending with ‘b’
8. Strings of a’s and b’s having no consecutive a’s
9. Strings of a’s and b’s having length either even or multiple of Three.
10. Strings of a’s and b’s having with alternate a’s and b’s

Write Regular Expressions for the following Language descriptions :

SET-2

i. L = { $a^n b^m$ | n>=4, m<=3 }
ii. L = { w | such that |w| mod 3=0, w € (a+b)*}
iii. L = { vuv | such that u,v € (a+b)* and |v| =2 }
iv. L = { $a^{2n} b^{2m}$ | n>=0, m>=0 }
v. L = { $a^{2n} b^{2m+1}$ | n>=0, m>=0 }
vi. L = { $a^n b^m$ | m + n is even}
vii. L = { $a^n b^m$ | m>=1, n>=1 and nm >=3}
viii. L = { w | such that N_a(w) mod 3=0, w € (a+b)*}
ix. L = { |w| mod 3 >= |w| mod 2, w € (a+b)* }
x. L = { |w| mod 3 <= |w| mod 2, w € (a+b)* }
xi. L = { |w| mod 5 <> 0, w € (a+b)* }

2. Finite Automata and Regular expressions

- We know that If a Language L is Regular then there exist a DFA, NFA Ɛ-NFA and Regular Expression. Conversly, the language denoted or described by DFA, NFA Ɛ-NFA and Regular Expression is always a Regular Language.

- The Figure 3.1 shows the equivalence of all these four Representations of Regular Languages.

- Further, For a Regular language L, If one representation exists then it is possible to get other representation without affecting the language description.

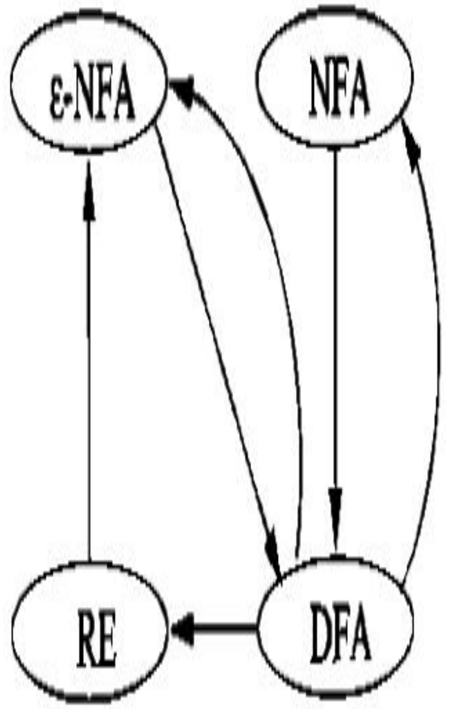


Figure 3.1: Plan for showing the equivalence of four different notations for regular languages

2.1. Building Ɛ-NFA from Regular Expression

- As mentioned in the equivalance of four diffirent notaions namely, DFA, NFA, Ɛ -NFA and Regular Expression, Let us discuss that for every language L that is L(R) for some Regular Expression R, there exist a L(E) for Some Ɛ -NFA E.
- The process is to have structural induction on the Regular Expression. In otherwords we start by constructing Ɛ -NFA/NFA for primitive Regular expressions namely, **Ø, Ɛ and a € ∑**. Later we combine these automata into larger automata that accepts the Union, Concatenation and star closure of the language accpeted by smaller automata.
- Statement of theorem associated with **Ɛ-NFA from Regular Expression is as follows :**
    - **Theorem - Every Language L defined by a Regular Expression is also defined by Finite Automata Ɛ-NFA. Consequently, L(R) is regular.**

Proof : We begin with automata that accepts the languages for the primitive Regular Expressions namely **Ɛ Ø**, and **a € ∑, .** The Primitive diagrams are shown in figure 3.16. i.e

Figure 3.16 (a) → **Ɛ,**

Figure 3.16 (b) → **Ø** and

Figure 3.16 (c) → **a € ∑**

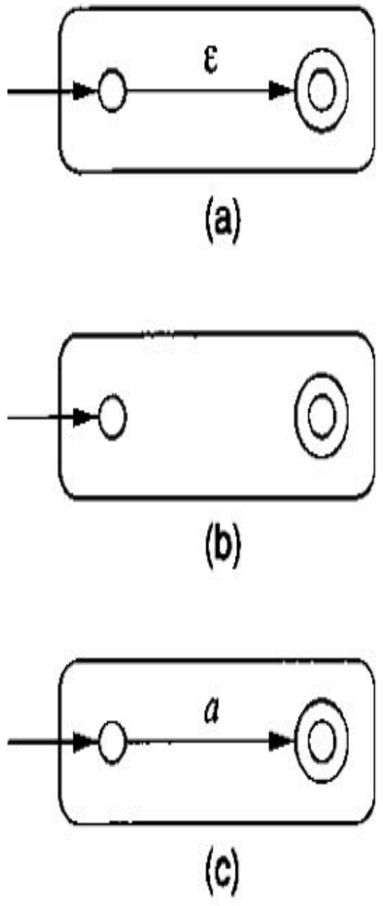


Figure 3.16: The basis of the construction of an automaton from a regular expression

- Assume that we have M(R) and M(S) are Two automata accepting the languages denoted by the regular expression R and S.
- With M(R) and M(S) represented in this way, we then construct automata for the Regular expressions R + S, R • S and R*.
- The Constructions are shown in the figure 3.17. i.e.

  Figure 3.17 (a) → R + S,

  Figure 3.17 (b) → R • S

  Figure 3.17 (c) → R*
- As indicated in the drawings, the initial and final states of the constituents machines lose their status and replaced by new initial and new final state.
- By stringing together several such steps, we can build Automata for arbitary complex regular Expression

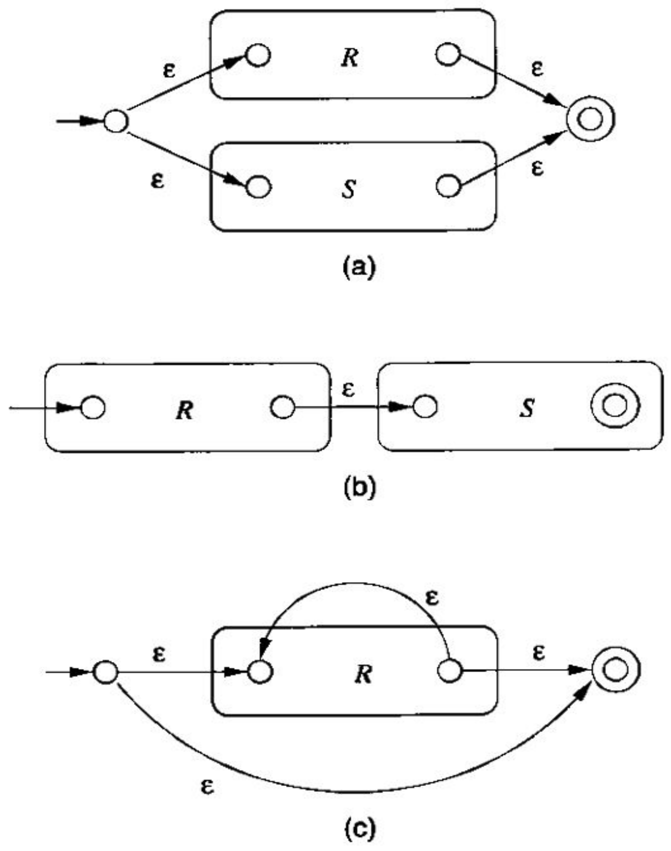


Figure 3.17: The inductive step in the regular-expression-to-ϵ-NFA construction

It is clear from the interpretetion of the graphs that this construction works for arbitary any Regular Expression R denoting the Language L(R).

2.1.1. Examples on building Ɛ-NFA from Regular Expressions

Example -1. Convert the Regular Expression :

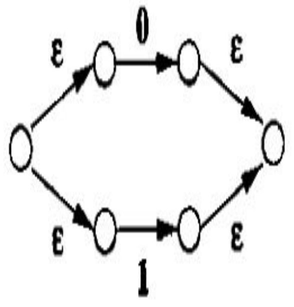
(0 + 1)* 1(0 + 1) to an Ɛ-NFA

Answer :

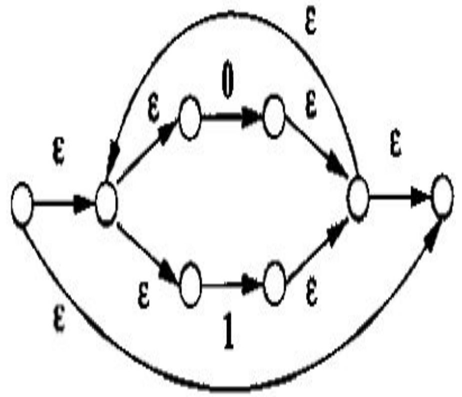
Step -1. We need to construct the Automata For **0 + 1** using primitive Automata Diagrams for input 0 and 1, we combine using the UNION. The same is Shown in **Figure -3.18 (a).**

Step - 2. In the second step we construct for **(0 + 1)*** by using STAR CLOSURE. The result is shown in **Figure -3.18 (b)**.

Step - 3. we construct for **(0 + 1)* 1** by using primpitive digram for Input 1 and Concatenation.



(a)



(b)

2.1.1. Examples on building Ɛ-NFA from Regular Expressions

Example -1. Convert the Regular Expression :

(0 + 1)* 1(0 + 1) to an Ɛ-NFA

Step - 4. Finally we string all the Automata of **Figure -3.18 (a) and Figure -3.18 (b) to get Automata for** (0 + 1)* 1 (0 + 1). The Resulting the final Automata as shown in **Figure -3.18 (c)**

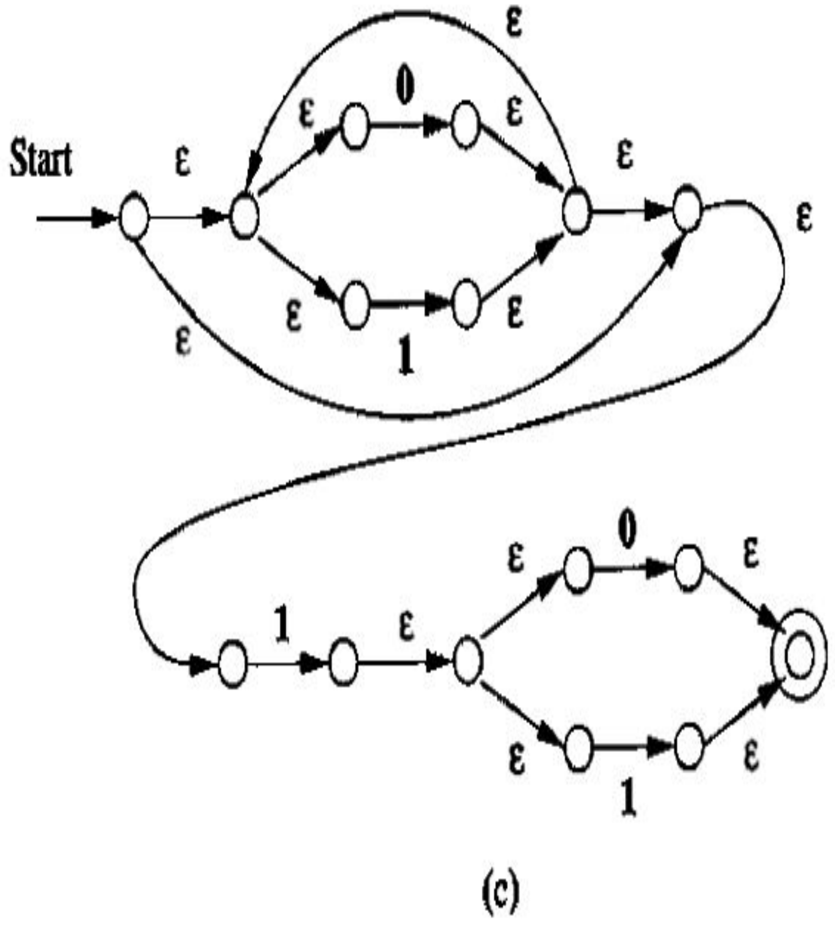


Figure 3.18: Automata constructed for Example 3.8

Exercise problems to Convert the Regular Expression to an Ɛ-NFA.

1. **(01 + 1)* → ∑ = {0, 1}**
2. **011(0 + 1)* → ∑ = {0, 1}**
3. **(a + bb)* (ba* + Ɛ) → ∑ = {a, b}**
4. **(ab*aa + bba* ab) → ∑ = {a, b}**

2.2. Building Regular Expressions from DFA and NFA by (State Elimination Method).

- Kleen’s theorem can be applied for NFA or even ε-NFA. However the construction of the regular expression is expensive i.e for an n-state automaton we have to construct about n^3 expressions.
- Also the length of the expression can grow by a factor of 4 on an average. If there is no simplification of the expressions then it could reach on the order of 4^n symbols
- To address this issue and improve the performance State elimination method is used.
- Here we Eliminate the intermediate states between start state and Final state of the automaton and replaces the edges with regular expressions that includes the behavior of the eliminated states until Eventually we get down to the situation with just a start and final node,

- Starting with the start state considering the intermediate states one at a time and then moving tewords accepting states, apply the state elimination process to produce an equivalent automaton with regular expression labels on the edges.
- In general If a **state s** is to be eliminated then we have to consider each **pair of successor and predecessor (q, p)**.
- The two situations are shown in Figure -1 and Figure -2.

Figure -1

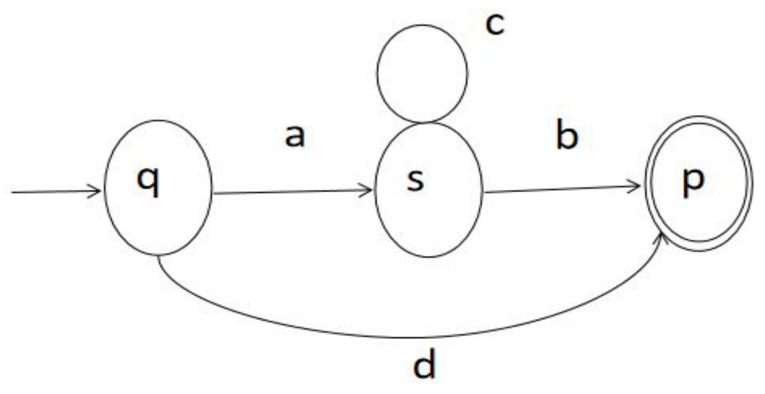
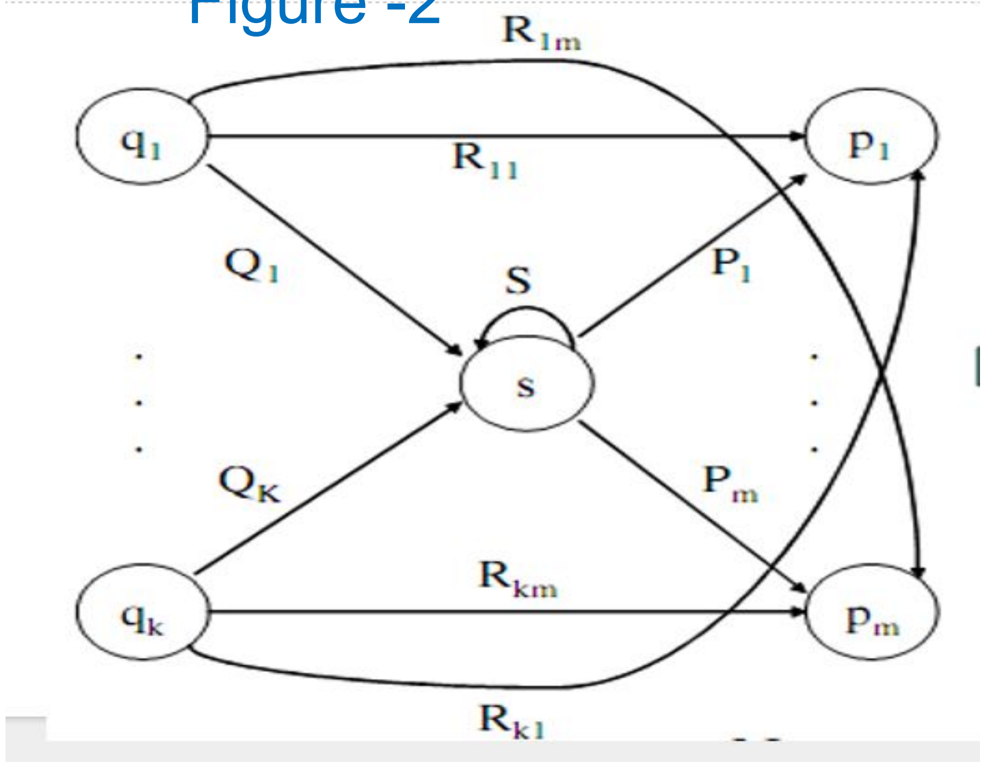


Figure -2



Principle : As Indicated in the Drawings Of **Figure -1 and Figure-2** . the followings steps can followed in the process of state elimination. The result of drawings are shown in the **Figure -3 and Figure-4**

- Let us Consider an intermediate **state s that** is to be eliminated.
- Let **state p** be its successor and **state q** be its predecessor.
- It is observed that all the strings of the form '**ac*b**' take the automata from **state q** to **state p** that pass through **state s** and also, the string **'d' that does not pass through state s** . So, We can now remove **state s** and attach an edge labeled '**ac*b +d**' from **state q to state p** directly.
- During this process the transitions are labeled by regular expression . The resulting diagrams are called as Generalized Transition Diagrams.

Figure - 1

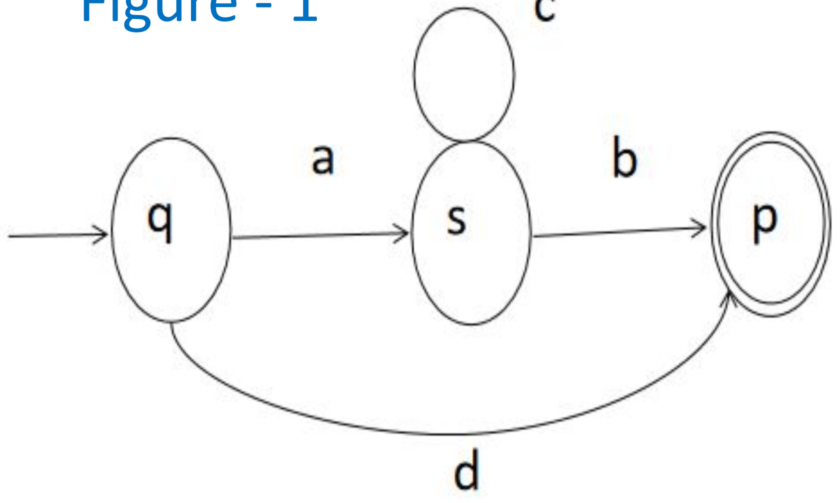
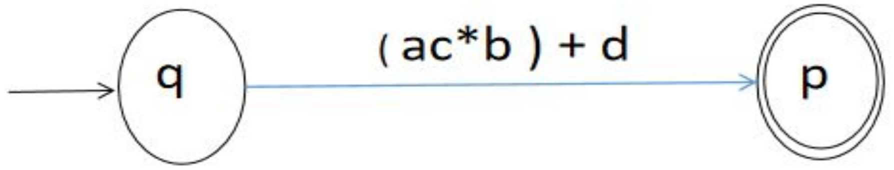
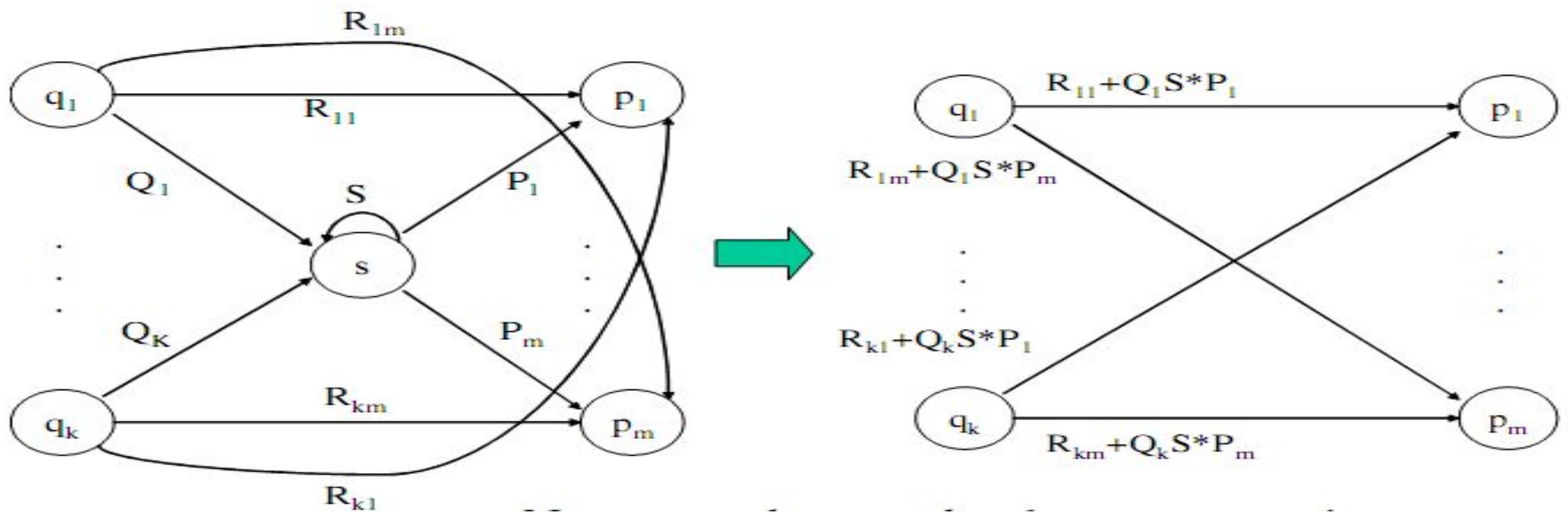


Figure - 3

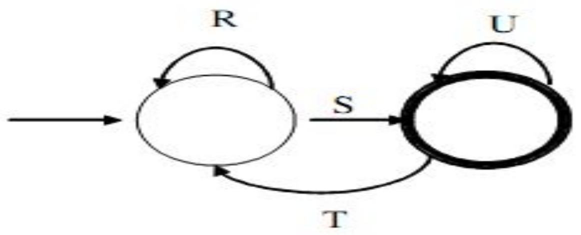


- Consider the Figures - below, which shows a generic **state s** about to be eliminated.The labels on all edges are **regular expressions.**
- To remove **state s**, we must make labels from **each qi to p1 up to pm** that include the paths we could have made through **state s**. The result of elimination is shown in the figure below.



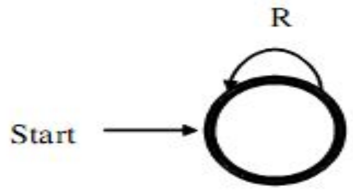
- By this process we **eliminate all intermediate states** and the result will be a **one or two state automaton** with a **start state** and **accepting state**.
- The configuration at the end is one of the following patterns

  I. **No intermediate states between start and final state (generic** Two **state automaton.)**



The regular expression in this case is (R+SU*T)*SU*

  II. **No intermediate state with Genaric One state Automata (start state and final states are same).**

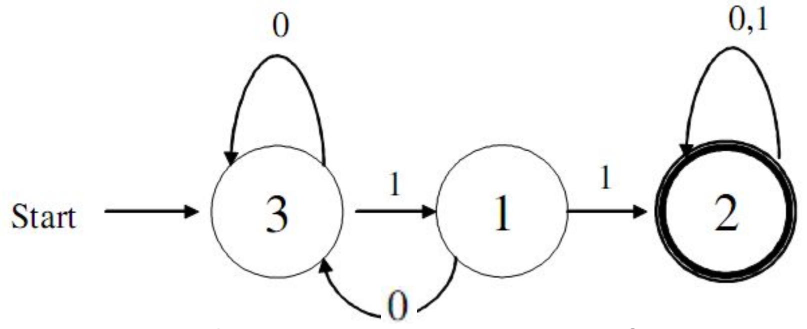


The regular expression in this case is - R*

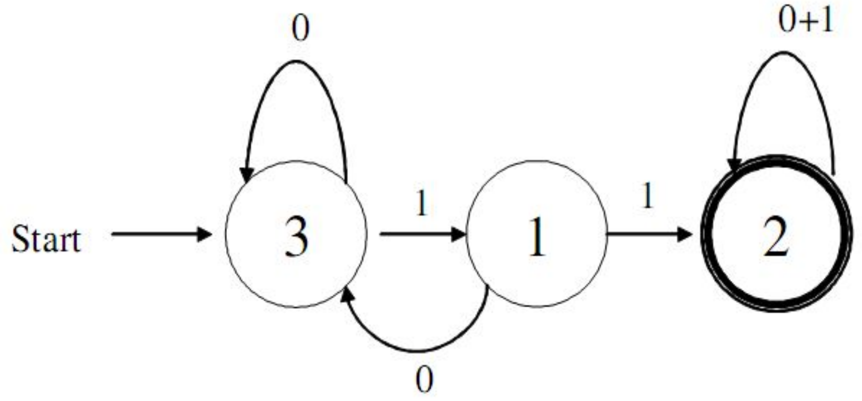
- If there are **n accepting states**, we must repeat the above steps for **each accepting states** to **get n different regular expressions, R1, R2, … Rn.**
- For each repeat we **turn any other accepting state to non-accepting.**
- The **desired regular expression** for the automaton is then the **union of each of the n regular expressions: R1 U R2… U RN**

DFA->RE Example

- Convert the following to a RE:

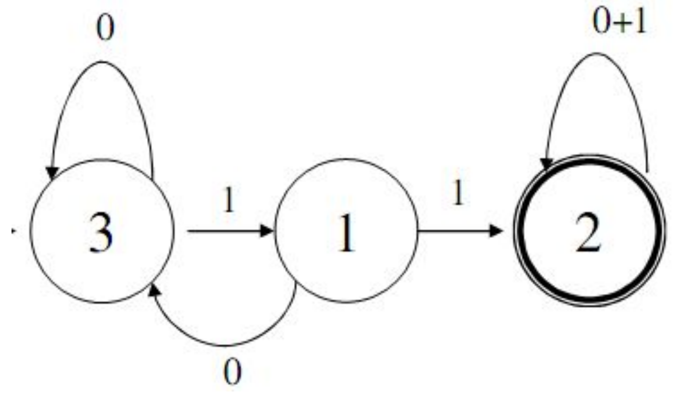


- First convert the edges to RE's:

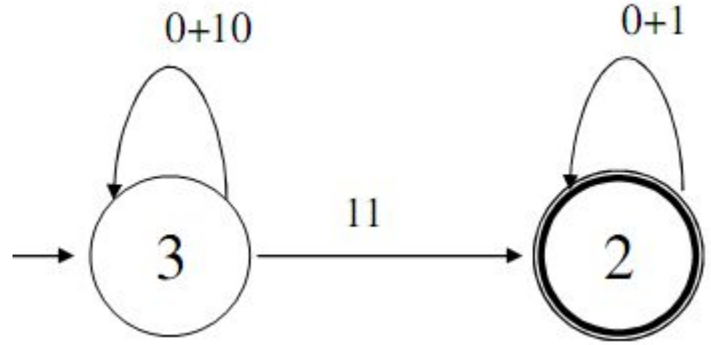


DFA -> RE Example (2)

- Eliminate State 1:



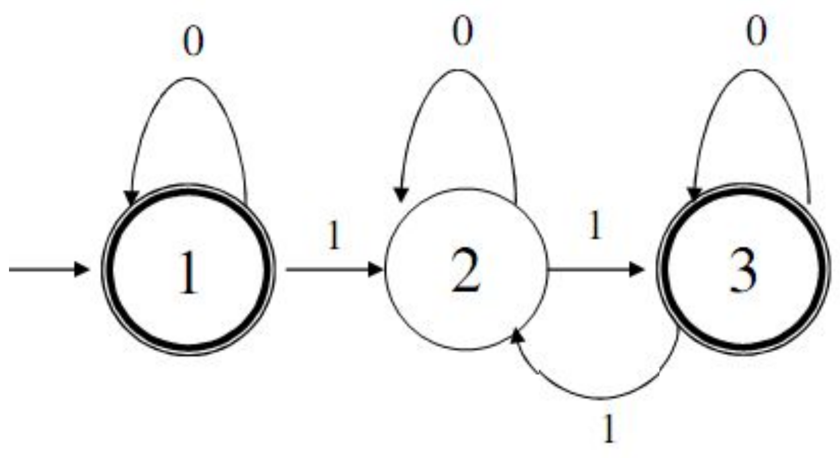
- Note edge from 3->3



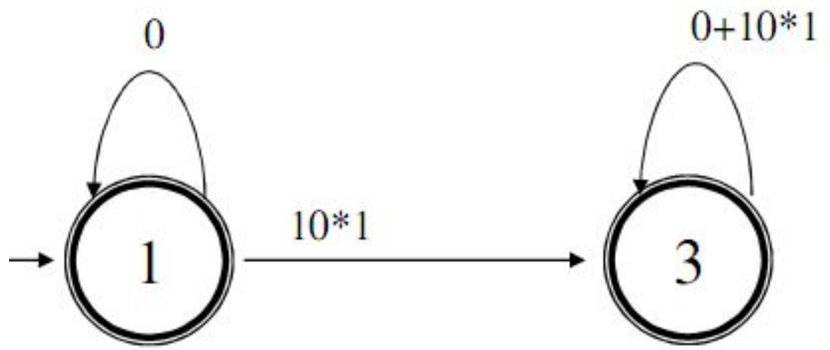
- Answer: (0+10)*11(0+1)*

Second Example

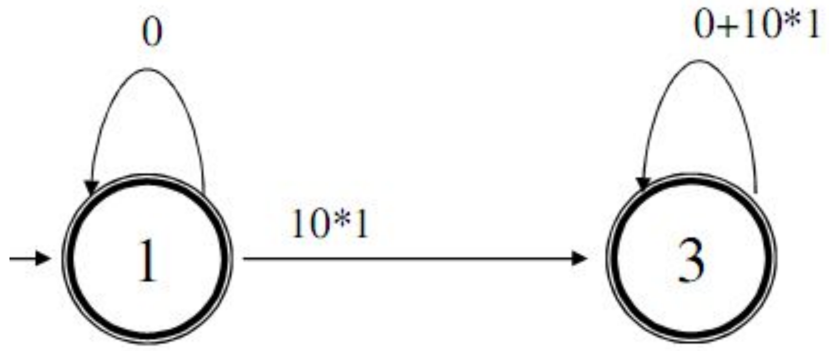
- Automata that accepts even number of 1's



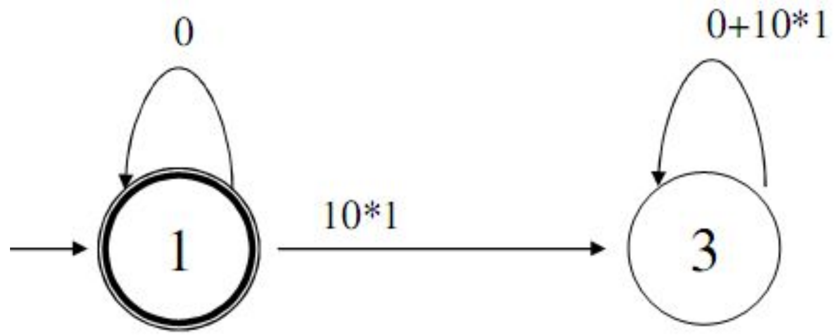
- Eliminate state 2:



Second Example (2)

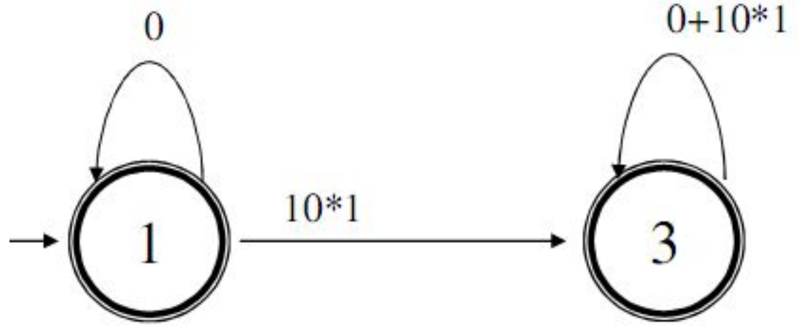


- Two accepting states, turn off state 3 first

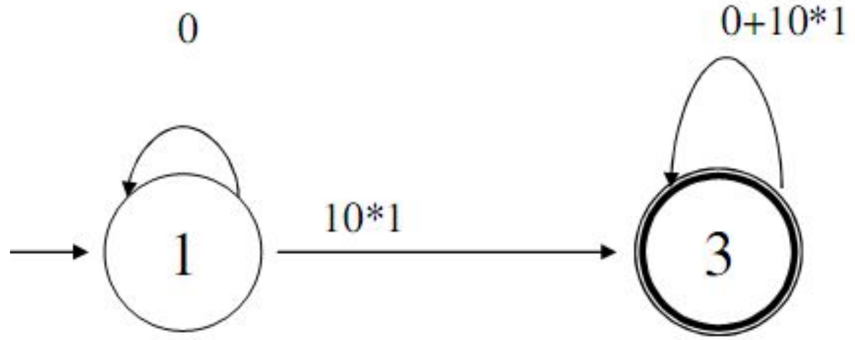


- This is just 0*; can ignore going to state 3 since we would "die"

Second Example (3)



- Turn off state 1 second:



- This is just 0*10*1(0|10*1)*
- Combine from previous slide to get 0* + 0*10*1(0+10*1)*

3. Minimization Of Automata(DFA) Using Table Filling Algorithm

- Let p and q are two states in DFA. Our goal is to understand when p and q ($p \neq q$) can be replaced by a single state.
- What are Distinguishable and Indistinguishable States ?.
- Table filling Algorithm
  - Procedure Mark( )
  - Minimize_states( )
  - Examples

What are **Indistinguishable** pair and **Distinguishable** pair of states ?

- Two **states p** and **q** of a DFA are called **Indistinguishable** if

  δ*(p, w) Є F implies **δ*(q, w) Є F**

  and

  δ*(p, w) ∉ F implies **δ*(q, w) ∉F** for all w Є ∑*.

- If on the other hand, there exists some string w Є ∑* such that

  δ*(p, w) Є F implies **δ*(q, w)** ∉ **F**

  or **vice versa**, then the **state p and q** are said to be a **distinguishable** by a string w.

Procedure **Mark( )**

```
Begin
    • Remove all inaccessible states. This can be done by enumerating all simple
      paths of the graph of the DFA starting at the initial state. Any state not part of
      such a path is inaccessible.
    • Consider all pairs of states (p, q).
          If $p Є F$ and $q ∉ F$ or vice versa Then
              mark the pair (p, q) as distinguishable.
    • Repeat the following step until no previously unmarked pairs are marked.
          ▪ For all pairs (p, q) and all $a Є ∑$, Compute
                $\delta(p, a) = p_a$ and $\delta(q, a) = q_a$.
          ▪ If the pair $(p_a, q_a)$ is marked as distinguishable Then
              mark the pair (p, q) as distinguishable.
End.
```

Note : Marking of **distinguishable** and **indistinguishable** states are shown in the Lower tringular matrix.

```
Procedure Minimize_states()
Begin
    . Using procedure MARK() obtain the table that gives pairs of distinguishable and
Indistinguishable states which are indicated by * and blank respectively

•  For each indistinguishable state set say qi,qj….qk   create new state labeled ij…k. for
   minimized DFA
•  For each transition rule of old DFA of the formδ (qr,a)=qp
        Begin
•          Find the sets to which qr and qp belongs
• .            If qr ∈ to (qi, qj.....qk) And qp ∈ to (ql.qm…..qn) then
                   add the following transition rule to minimized DFA
           δ(ij…k,a) = lm….n
          End
•  The initial state q0 for minimized DFA is the state whose label        includes 0
•  The Final states for minimized DFA is the set of all the states whose label Contains i such that qi
   ∈ F ( set of final sates of old DFA)
End
```

Examples on Minimization of DFA Using Table Filling Algorithm:

Example -1 :

Use Table Filling Algorithm to Minimize the states of DFA shown in the **FIGURE-1**. Also, Draw the Table showing the pair of Distingwishable and Indistingwishable States and Minimized DFA.

Answer :

Step-1 : Find the states which are inaccessible. It is clear from the FIGURE-1, that the **states q2 and q4 are not accessible from Start state** - **q0**. Hence **these states (q2,q4)** along with **connecting edges** are eliminated.

FIGURE-2 shows the DFA after eliminating the states q2 and q4

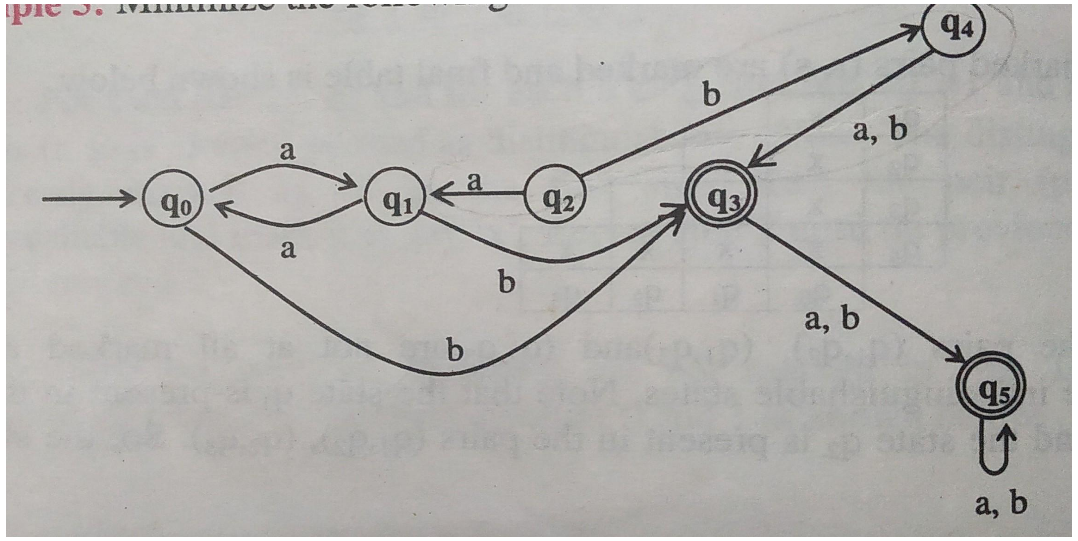


FIGURE - 1

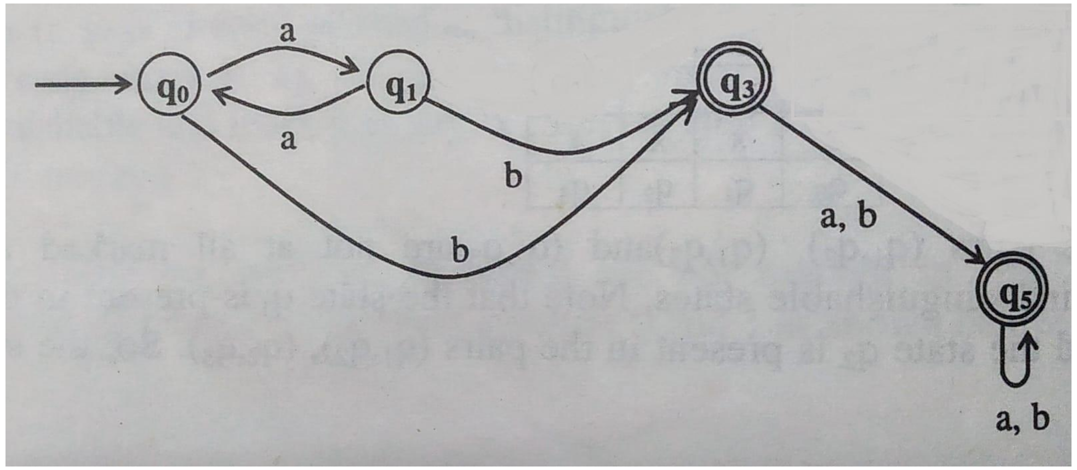


FIGURE - 2

Examples on Minimization of DFA Using Table Filling Algorithm:

Example -1 :

Step-2 : Draw the Table (Two dimentional Lower Tringular Matrix) to consider the pairs of states(p,q) for marking them as Distingwishable and Indistingwishable. **Figure-3** Shows the Table.

Step -3 : Use Mark() procedure to find the pair of Distingwishable and Indistingwishable.

- Initially, the pairs **(q0, q3), (q0, q5), (q1,q3) and (q1, q5)** are Marked as Distingwishable as one of state in the pair is a final state. **Figure-4**. Shows the Table

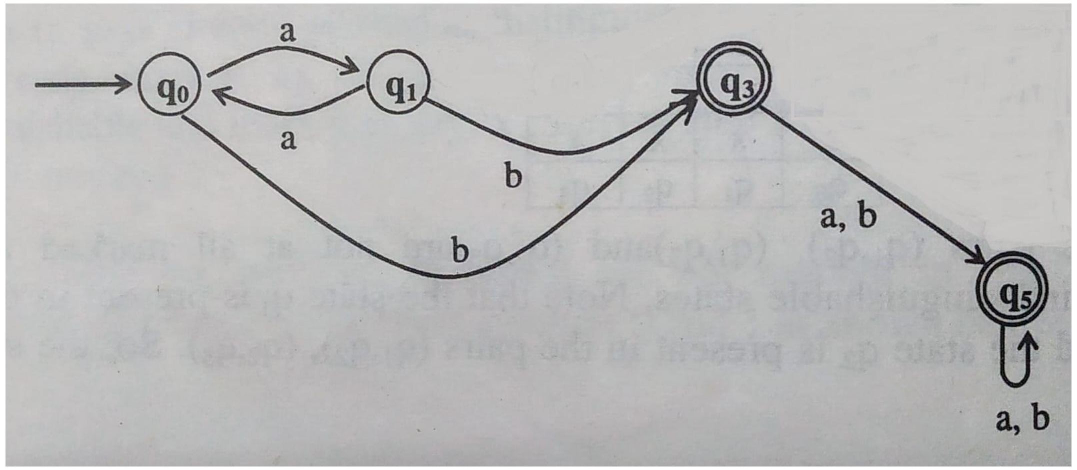


FIGURE - 2

q1			
q3			
q5			
	q0	q1	q3

←**FIGURE - 3**

q1			
q3	D	D	
q5	D	D	
	q0	q1	q3

←**FIGURE - 4**

Examples on Minimization of DFA Using Table Filling Algorithm:

Example -1 :

- Then Unmarked pairs namely,(q0,q1) and (q3, q5) are considered for marking. This is done as follows and Fiqure-5 Shows final marked Table :

δ	a	b	Computaions
(q0, q1)	(q1,q0)	(q3, q3)	δ(q0, a)= q1 and δ(q1, a)= q0 →**Pair is (q0,q1) Not Marked** δ(q0, b)= q3 and δ(q1, b)= q3 →**Pair is (q3,q3) Not a Pair**
(q3, q5)	(q5, q5)	(q5, q5)	δ(q3, a)= q5 and δ(q5, a)= q5 →**Pair is (q5,q5) Not a Pair** δ(q3, a)= q5 and δ(q5, a)= q5 →**Pair is (q5,q5) Not a Pair**

←FIGURE - 5

q1			
q3	**D**	**D**	
q5	**D**	**D**	
	q0	q1	q3

Example -1 :

Step-4.

- Find the minimized DFA . It is clear from the table shown in the **Figure -5** that the pairs **(q0, q1)** and **(q3, q5)** are not marked and hence they are Indistingwishable States.
- These can be combined as one state. Finaly minimized DFA has only two states - **(q0,q1)** and **(q3, q5)**.
- Its **trasitions - δ** is obtained by consulting **Original DFA.**
- Transition Table is shown in **Figure -6**

δ	a	b
(q0, q1)	(q0,q1)	(q3, q5)
(q3, q5)	(q3,q5)	(q3, q5)

←**FIGURE - 6**

q1			
q3	D	D	
q5	D	D	
	q0	q1	q3

←**FIGURE - 5**

←FIGURE - 4

Step -5. Start state of the minimized DFA is the group which has start state of the Original DFA.

Step -6. Final state of the Minimized DFA is the group which has Final state of the Original DFA. **Figure -7** shows the Final Minimized DFA

δ	a	b
→(q0, q1)	(q0,q1)	(q3, q5)
*(q3, q5)	(q3,q5)	(q3, q5)

←**FIGURE - 7**

Example -2

Step-1 : Find the states which are inaccessible. It is clear from the **Figure-1**, that the all the states are accessible and no State gets eliminated. and DFA remains the same.

Step-2 : Draw the Table (Two dimentional Lower Tringular Matrix) to consider the pairs of states(p,q) for marking them as Distingwishable and Indistingwishable. **Figure-2** Shows the Table.

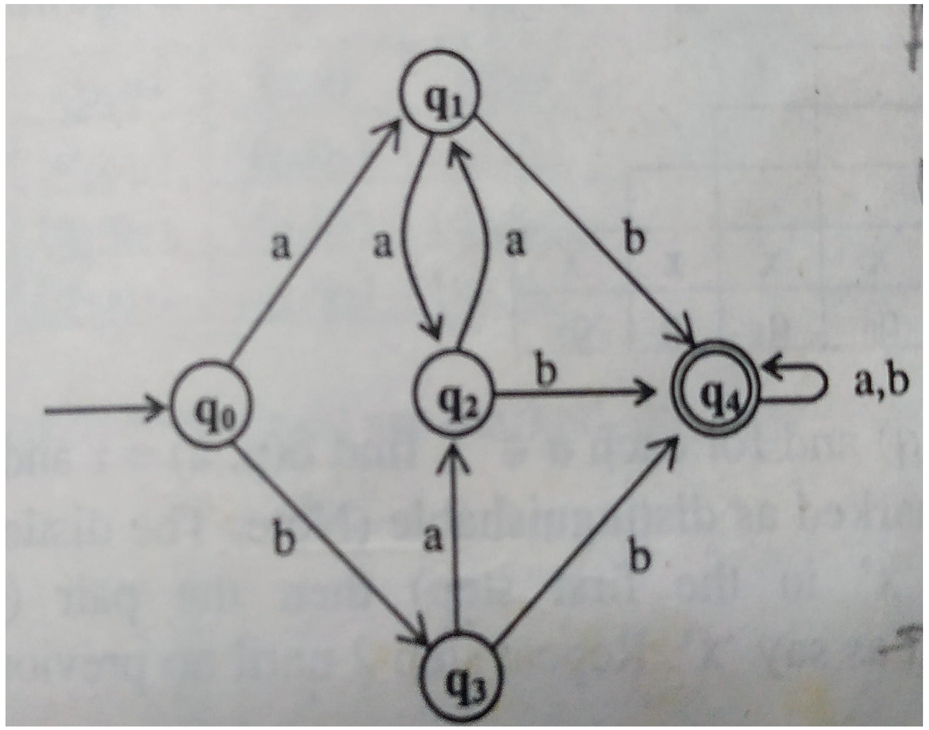


Figure -1

q1				
q2				
q3				
q4				
	q0	q1	q2	q3

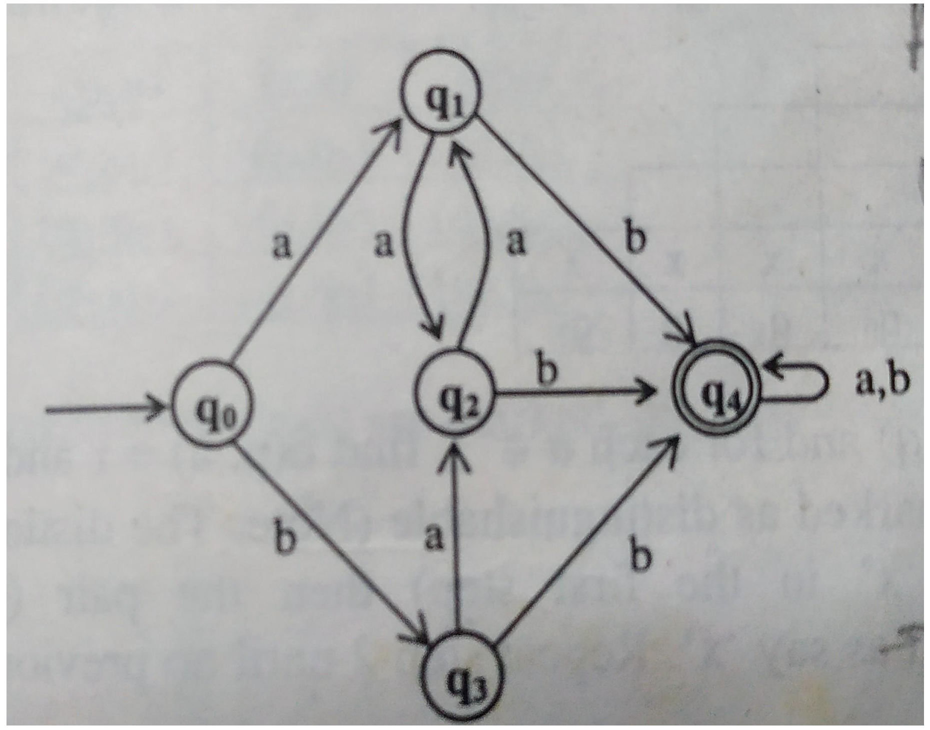
←**Figure -2**

Step -3 : Use Mark() procedure to find the pair of Distingwishable and Indistingwishable.

- Initially, the pairs **(q0, q4), (q1, q4), (q2,q4) and (q3, q4)** are Marked as Distingwishable as one of state in the pair is a final state. **Figure-3.** Shows the Table
- Then Unmarked pairs namely,**(q0,q1), (q0, q2),(q0, q3),(q1, q2), (q1, q3) and (q2, q3)** are considered for marking. This is done as follows and **Fiqure-4** Shows final marked Table :

q1				
q2				
q3				
q4	**D**	**D**	**D**	**D**
	q0	q1	q2	q3

←FIGURE - 3



δ	a	b	Computaions
(q0, q1)	(q1,q2)	(q3, q4)	δ(q0, a)= q1 and δ(q1, a)= q2 **→Pair is (q1,q2) is Not Marked** δ(q0, b)= q3 and δ(q1, b)= q4 **→Pair is (q3,q4) is Marked so (q0, q1) is Marked**
(q0, q2)	(q1, q1)	(q3, q4)	δ(q0, a)= q1 and δ(q2, a)= q1 **→Pair is (q1,q1) is Not a Pair** δ(q0, b)= q3 and δ(q2, b)= q4 **→Pair is (q3,q4) is Marked so (q0, q2) is Marked**
(q0, q3)	(q1, q2)	(q3, q4)	δ(q0, a)= q1 and δ(q3, a)= q2 **→Pair is (q1,q2) Not Marked** δ(q0, b)= q3 and δ(q3, b)= q4 **→Pair is (q3,q4) is Marked so (q0, q3) is Marked**
(q1, q2)	(q1, q2)	(q4, q4)	δ(q1, a)= q2 and δ(q2, a)= q1 **→Pair is (q1,q2) is Not Marked** δ(q1, b)= q4 and δ(q2, b)= q4 **→Pair is (q4,q4) is Not a Pair**
(q1, q3)	(q2, q2)	(q4, q4)	δ(q1, a)= q2 and δ(q3, a)= q2 **→Pair is (q2,q2) is Not pair** δ(q1, b)= q4 and δ(q3, b)= q4 **→Pair is (q4,q4) is Not a Pair**
(q2, q3)	(q1, q2)	(q4, q4)	δ(q2, a)= q1 and δ(q3, a)= q2 **→Pair is (q1,q2) is Not Marked** δ(q2, b)= q4 and δ(q3, b)= q4 **→Pair is (q4,q4) is Not a Pair**

←FIGURE - 4

q1	**D**			
q2	**D**			
q3	**D**			
q4	**D**	**D**	**D**	**D**
	q0	q1	q2	q3

Step-4.

- Find the minimized DFA . It is clear from the table shown in the **Figure -4** that the pairs **(q1, q2)** , **(q1, q3) and (q2, q3)** are not marked and hence they are all Indistingwishable States.
- These can be combined as one state. Further, by applying equivalnce property all q1, q2, q3 are indistingwishable and hence all three can be combined - (q1,q2,q3), as One state
  Finaly minimized DFA has only three states - **(q0), (q1, q2, q3)** and **(q4)**.
- Its **trasitions - δ** is obtained by consulting **Original DFA.**
- Transition Table is shown in **Figure -5**

q1	D			
q2	D			
q3	D			
q4	D	D	D	D
	q0	q1	q2	q3

←Figure - 4

δ	a	b
(q0)	(q1, q2, q3)	(q1, q2, q3)
(q1, q2, q3)	(q1, q2, q3)	(q4)
(q4)	(q4)	(q4)

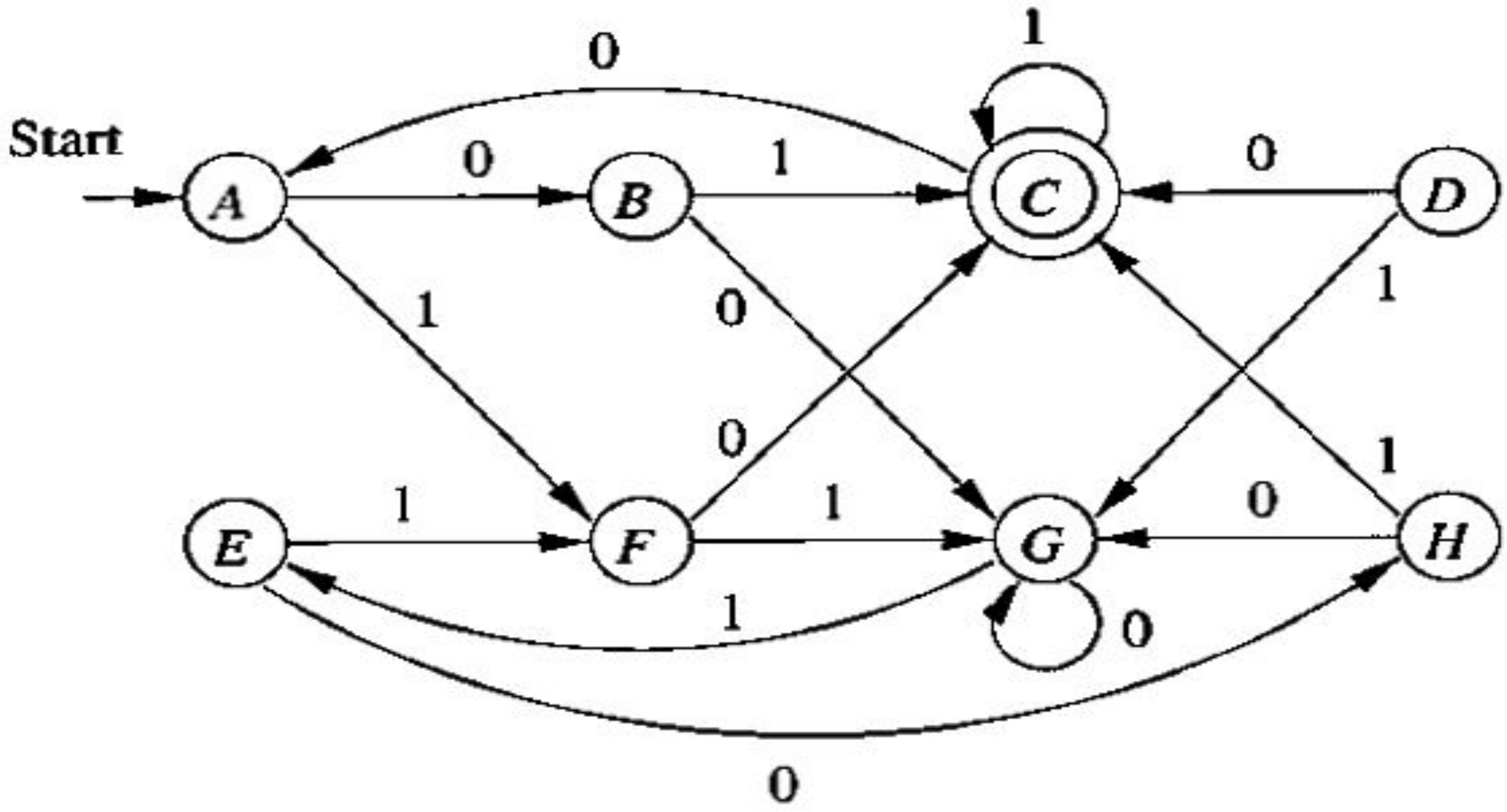
Figure - 5

Step -5. Start state of the minimized DFA is the group which has start state of the Original DFA.

Step -6. Final state of the Minimized DFA is the group which has Final state of the Original DFA. **Figure -6** shows the Final Minimized DFA

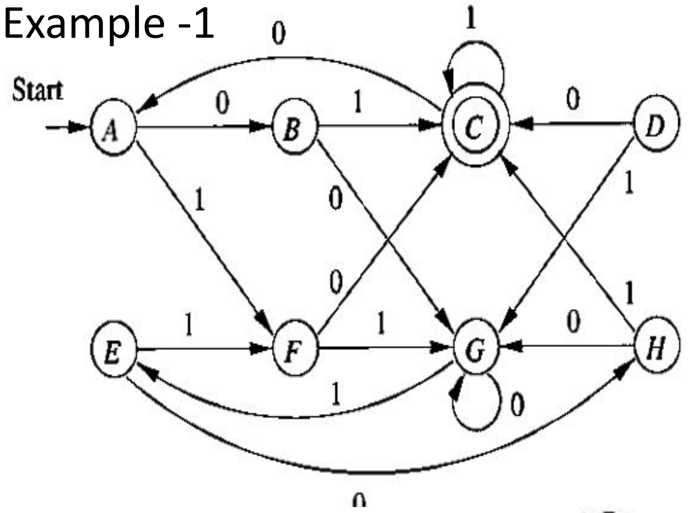
δ	a	b
→(q0)	(q1, q2, q3)	(q1, q2, q3)
(q1, q2, q3)	(q1, q2, q3)	(q4)
*(q4)	(q4)	(q4)

← **Figure - 6**


Start
A
B
C
D
E
F
G
H
0
1
0
1
0
1
0
1
0
1
1
0
1
1
0
0

Exercise Problems on minimization of DFA

Example -1



Example -2

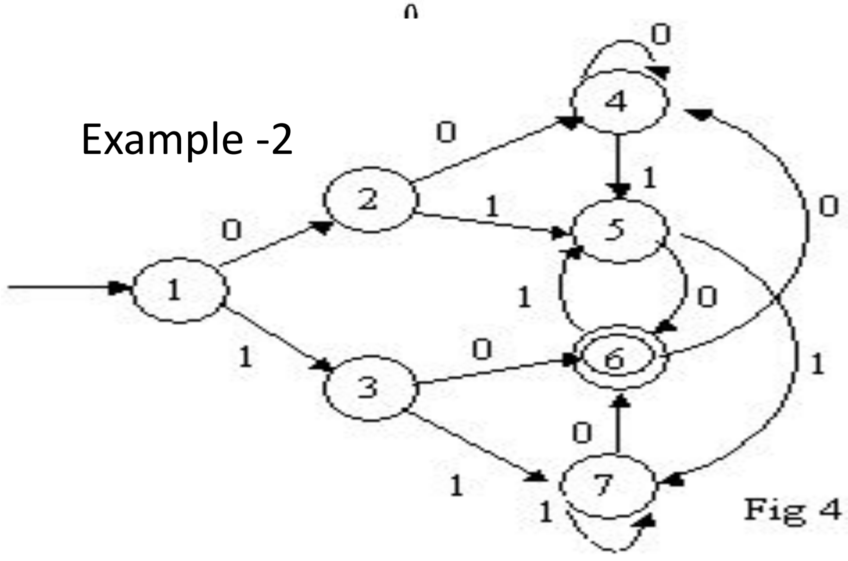


Fig 4

Example -3

	0	1
→ A	B	A
B	A	C
C	D	B
*D	D	A
E	D	F
F	G	E
G	F	G
H	G	D

Example -4

	0	1
→ A	B	E
B	C	F
*C	D	H
D	E	H
E	F	I
*F	G	B
G	H	B
H	I	C
*I	A	E

END of UNIT-1.