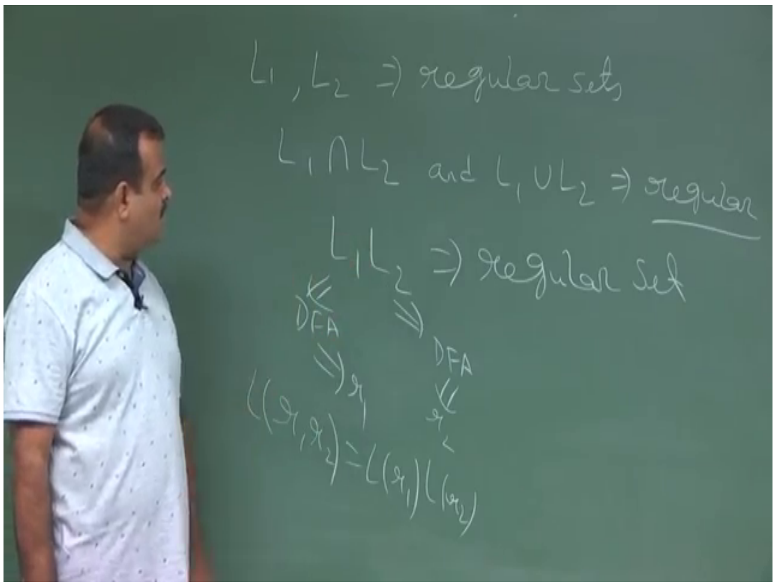
Introduction to Automata, Languages and Computation
Prof. Sourav Mukhopadhyay
Department of Mathematics
Indian Institute of Technology, Kharagpur

Lecture – 25
Closure Properties of Regular Set (Contd.)

So we are talking about the properties of closure properties of regular set or the regular language. So, we have seen the regular set is closure under union, intersection, concatenation also.

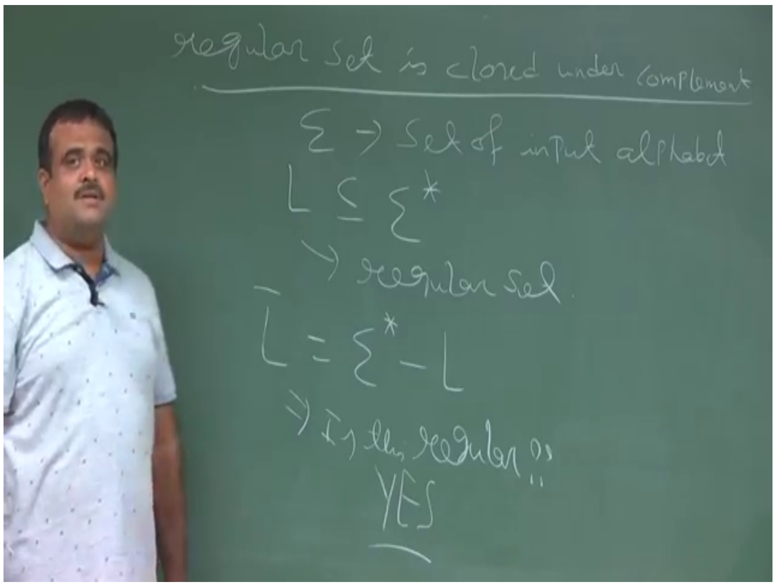
(Refer Slide Time: 00:35)


$L_1, L_2 \Rightarrow$ regular sets
$L_1 \cap L_2$ and $L_1 \cup L_2 \Rightarrow$ regular
$L_1 L_2 \Rightarrow$ regular set
DFA
DFA
r_1
r_2
$L(r_1 r_2) = L(r_1) L(r_2)$

If you have two regular sets L 1 and L 2 regular sets or regular language. In the last class we have seen L 1 intersection L 2 and L 1 union L 2 are both regular ok. In fact, last class we have constructed a DFA which is accepting this L 1 intersection L 2 and L 1 union L 2. Even L 1 concatenation and L 2 is also regular why because it is coming from a regular expression because since this is a regular so, we have a DFA for this, we have another DFA for this.

So, once we have a DFA we have a r 1 regular expression we have r 2 in r 1, r 2 is also a regular expression which in corresponding to L of r 1, L of r 2 and so, r 1 r 2 will corresponding to this language. So, r L l concatenate with L 2 is also regular. So, even the close star, star is also regular which is immediately coming from the regular expression. Now today we will discuss some more properties some more closure properties like complement. We have given a regular set is the complement is also regular.

(Refer Slide Time: 02:15)



regular set is closed under complement

$\Sigma \rightarrow$ set of input alphabet

$L \subseteq \Sigma^*$

$\rightarrow$ regular set

$\bar{L} = \Sigma^* - L$

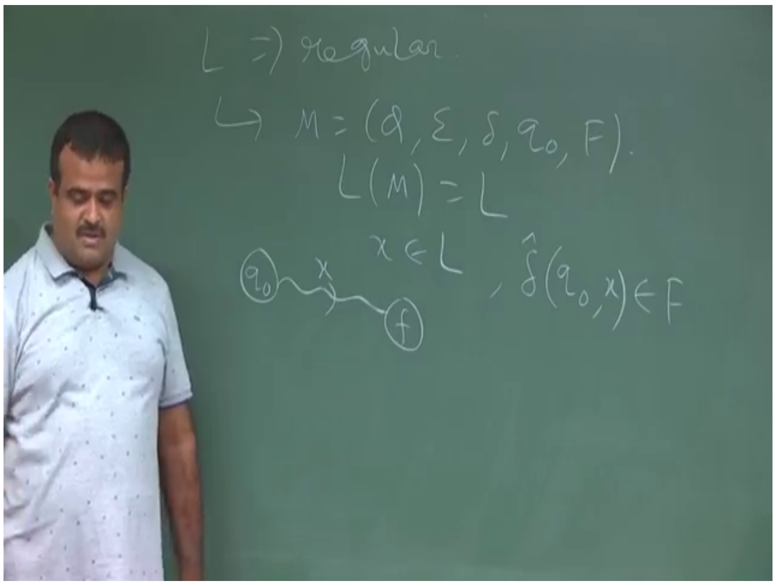
$\rightarrow$ Is this regular??

YES

So, regular set is closed under compliment. So that means what, that we have given a this sigma set of input alphabet finite set a input alphabet and suppose you have given a language which is the subset of any string consists of the alphabet give me coming from sigma.

Now, suppose this is regular we have given a regular set or regular language L. Now the question is whether the compliment, compliment means L c or L prime whichever notation will use this is nothing, but the set of all string which are not in L. The question is, is this regular? Is compliment is regular and the answer is yes we have to show that. We have to show the complement is also regular. So, how to show this? In fact, if we have given a regular expression for a language can we have a regular expression for the compliment?

(Refer Slide Time: 03:57)



$L \Rightarrow$ regular

$\hookrightarrow M = (Q, \Sigma, \delta, q_0, F)$.

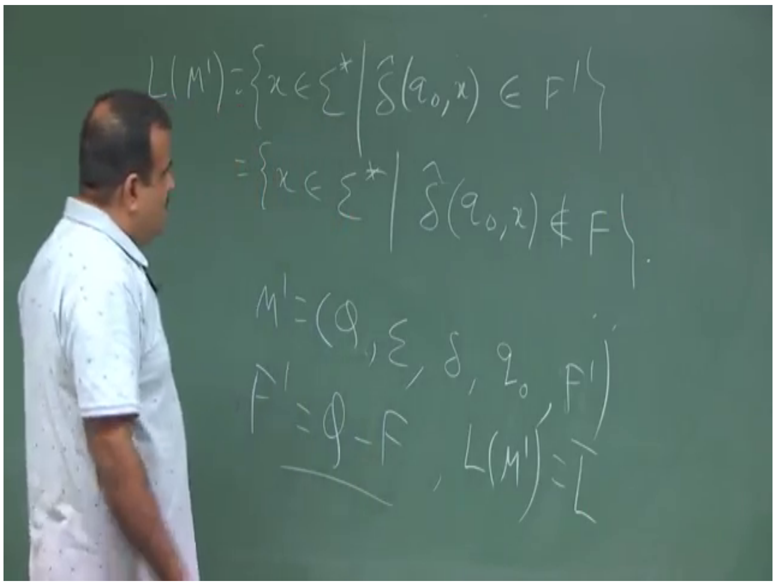
$L(M) = L$

$x \in L$, $\hat{\delta}(q_0, x) \in F$

$$\begin{tikzcd} q_0 \arrow[r, squiggly, "x"] & f \end{tikzcd}$$

So, first of all how to get the compliment I mean let L be a L is regular. This means there exist a DFA which is accepting L. So that means, all the string. So, L is the. So, this is the all the string if x belongs to L, then delta head of q 0, x is reaching a final state ok. So, that is the L. We start with q. So, with the x we should reach the F, one of the F there may be many final states. So, this is given now from here how we can construct a DFA which will accept the L dash or L c L complement.

(Refer Slide Time: 05:03)



$L(M') = \{x \in \Sigma^* \mid \hat{\delta}(q_0, x) \in F'\}$

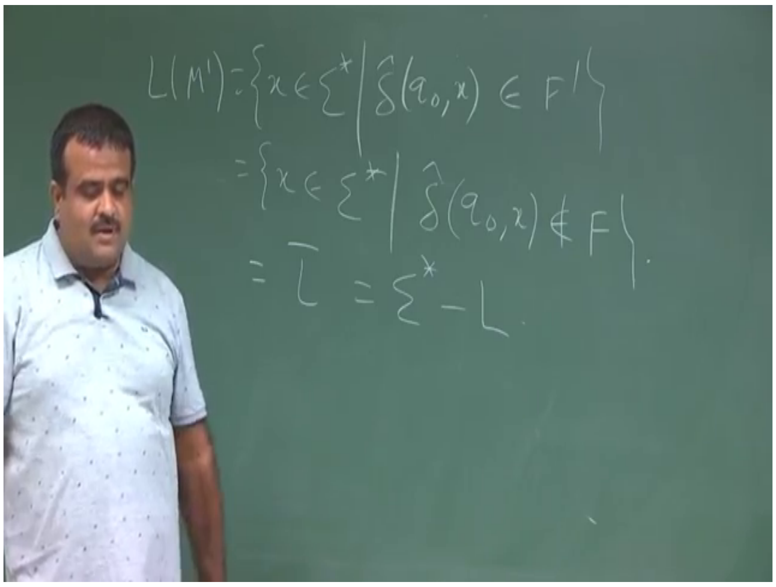
$= \{x \in \Sigma^* \mid \hat{\delta}(q_0, x) \notin F\}$.

$M' = (Q, \Sigma, \delta, q_0, F')$

$\underline{F' = Q - F}$, $L(M') = \bar{L}$

So, for that what we do we just take this we construct a M prime which is same as the states are same and input alphabet we can keep it same without any loss of generality, delta is also same, q 0 is same all the thing F we do the F complement. So, F is this is a prime. F prime is nothing but Q minus F ok. So, we will make all the other states as a final state other than this Q minus F that is the only change over here otherwise everything is same. In that case, we claim that if the language accepted by M is basically L prime or L c why? Because so, what are the language accepted by this? This is set of all string such that delta head of q 0 x must belongs to this must belongs to F prime. So, if a symbol is must belongs to L prime then it should not belongs to sorry should not belongs to F does not belongs to F that is all. So, this is the.

(Refer Slide Time: 06:55)

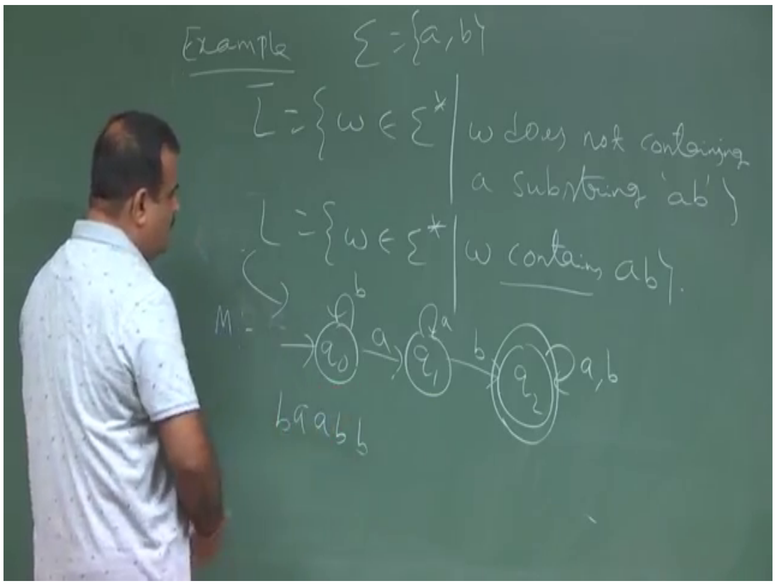


$$L(M') = \{x \in \Sigma^* \mid \hat{\delta}(q_0, x) \in F'\}$$
$$= \{x \in \Sigma^* \mid \hat{\delta}(q_0, x) \notin F\}.$$
$$= \bar{L} = \Sigma^* - L.$$

So, this is basically L prime set of all string which are not accepting by L. So, that will be accept by L prime. So, this is nothing but ok.

So, we just given a DFA we just make all the other states other than the existing final state we just make the those states as a final states. So, that is the that will accept this and this should be DFA, it is not for NFA, this is not possible because we can take an example for epsilon NFA, given epsilon NFA if we do this then it is not possible. So, we can just take a quick example.

(Refer Slide Time: 07:41)

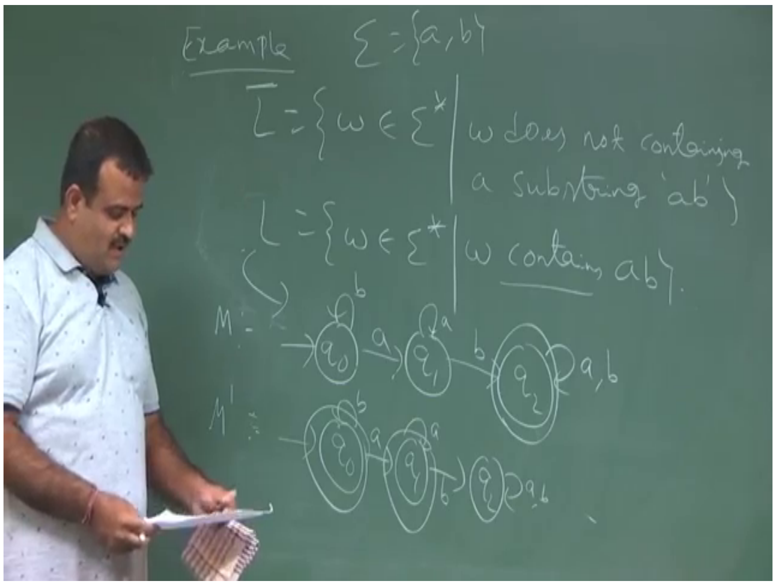

Example
Σ = {a, b}
L̄ = {w ∈ Σ* | w does not containing a substring 'ab'}
L = {w ∈ Σ* | w contains ab}.
M :
b
a
q0
a
q1
b
q2
a, b
baabb

So, if we have a language w, suppose it is a b the w does not the string w does not contain does not containing containing a substring ab.

Now, what is the complement? Complement is before the complement let us draw the. So, what is the complement? Complement is w belongs to this w does contain w contains a b. Now first of all we have to show this is regular. So, for that we need to get a DFA. So, we can easily have a DFA, this is the starting state say q 0, q 1 and say q 2. So, with a b now with a we are here, with b we go there and with a b. So, it does not this is the final state it does not contain the string a b, because if we are a b then with a we go here does not continue a substring a b ok. So, this is the thing and what is the complement of this? So, this is the DFA no this is the DFA containing this. So, I make this as a sorry this is our M.

So, we have a DFA a we have a regular language which is containing a b because if we take if we just take a b. So, with a we go here with b we go there. So, a b b also accepting a a a b b b a b b all the string is accepting now complete and is this one. So, now, from here we need to have a DFA which is accepting this language; that means, the language which is having no sub string containing a b. So, this the our method is we just make the take all the states other than this final state as a final state.

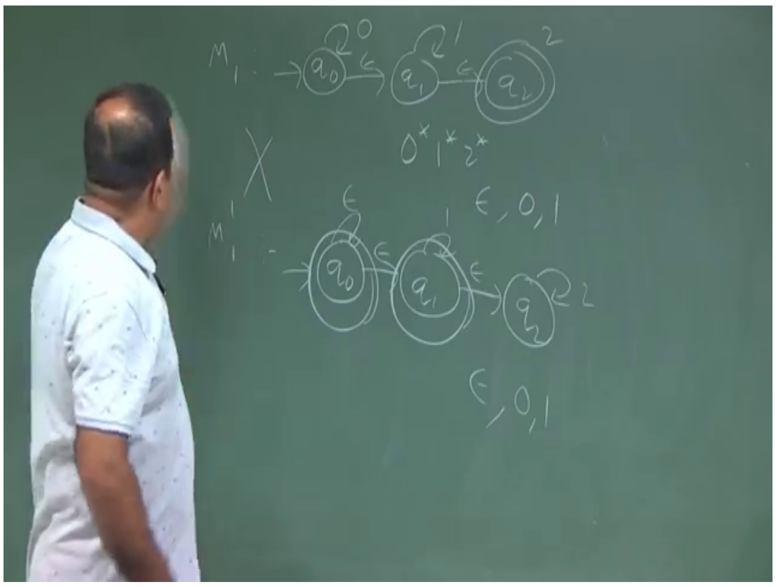
(Refer Slide Time: 10:41)


Example $\Sigma = \{a, b\}$
$\bar{L} = \{ w \in \Sigma^* \mid w$ does not containing a substring 'ab' $\}$
$L = \{ w \in \Sigma^* \mid w$ contains $ab \}$.
M :
M' :

So, what is M prime? So, rules are same only thing q 1 q 2. So, this is b this is a, transition rules are same. This is a this is b this is a b only thing we make this two as a final state we make this 2 as a final state ok. Then this is accepting the string set of all string which are not containing the substring a b ok.

Now, this method will fail when we have epsilon NFA. This is with the DFA, but if a epsilon NFA we cannot just make the complement of the final states that will not work.

(Refer Slide Time: 11:37)

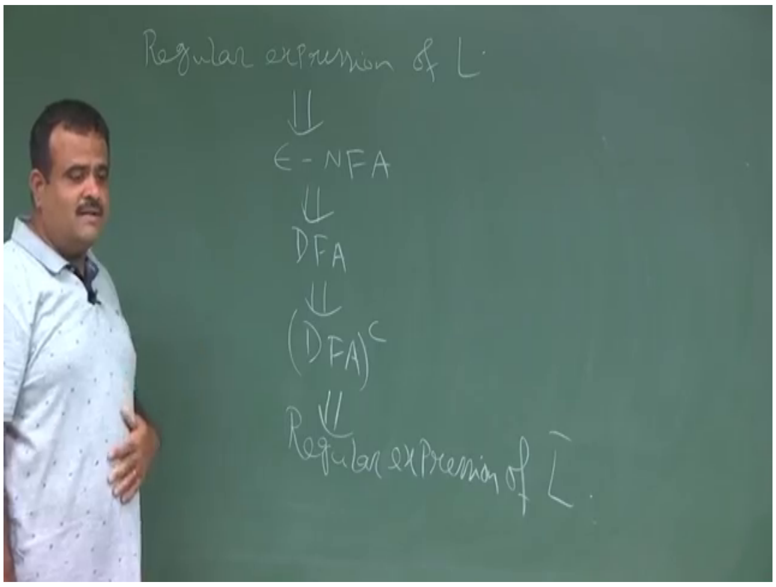


For example, if we have this epsilon NFA M 1, we having a q 0 this starting state, we have 0 q 1 with epsilon, we have 1 q 2 this is the final states 2. So, this is the epsilon NFA so, which is the language accepting this, this is 0 star 1 star 2 star. So, any number of 0 followed by any number 1 followed by any number of 2.

Now, this is also accepting epsilon, 0, 1. Now, if you apply the same rule on complement. So, M 1 star is nothing but, we are keeping the transition are same q 2. So, we are keeping the transition same epsilon 1 epsilon 2 only things this is starting only thing we are changing the final state as the compliment of the; here it was this to us not final you are making final we are keeping this not final. Ok now if you do that this is also accepting epsilon, this is also accepting 0, 1. But this is accepting by this. So, this is not correct. So, this way of doing is not correct while this automata is epsilon NFA. It is perfect if the automata is DFA then we can do the this complement things ok.

So, now we will talk about how to get the complement for the regular expression.

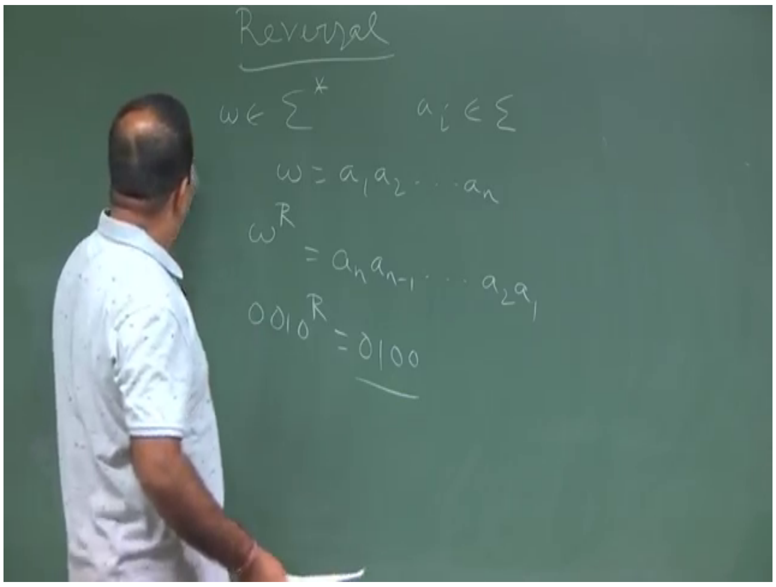
(Refer Slide Time: 13:17)



So, suppose you have a, we have given a regular expression for of L of a language. Now how to give the; how to get the regular expression for L complement? So, how to do that? Since we have a regular expression from here we can have a epsilon NFA we can construct. So, from here we can have a NFA or directly DFA substrate construction and then from DFA we can have a complement DFA, we complement DFA means we just do the changes of the final states final state we make the other state as a final state that is we can refer as form, then that will give us the regular expression for then we can have a regular expression for the new DFA regular expression of L prime that is all.

So, given a regular expression of [laugher], we can follow the step to get the regular. This is that this complement DFA means, we just change the a final state by taking the final state other than the final state of the original DFA. Now we will take discuss another property which is called reversal of a string reversal.

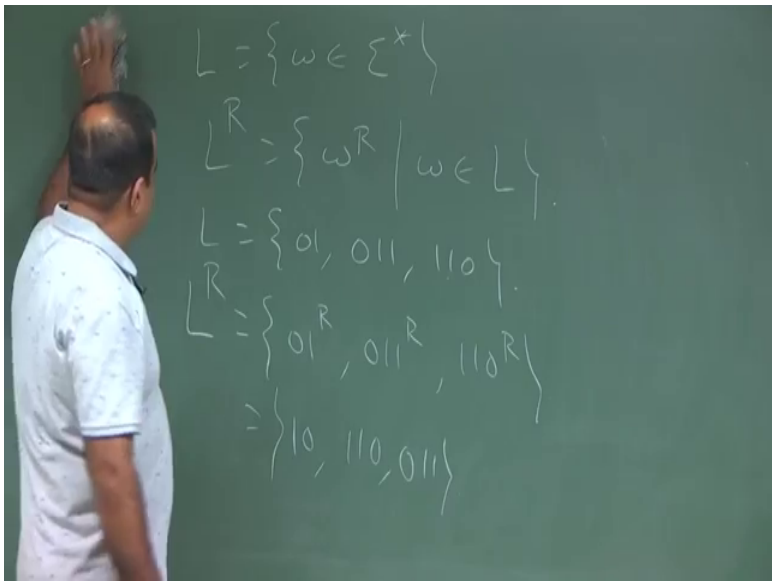
(Refer Slide Time: 14:41)


Reversal
$w \in \Sigma^*$ $a_i \in \Sigma$
$w = a_1 a_2 \cdots a_n$
$w^R = a_n a_{n-1} \cdots a_2 a_1$
$0010^R = 0100$

So, sigma is the set of alphabets. So, suppose w belongs to sigma star, now if w is say a 1, a 2, a a n where a i are coming from sigma, then the reversal of w is denoted by we just reverse this. So, a n will come first a n minus 1, a 2, a 1 will just reverse it that is called reversal of the string.

So, for example, if we have 0, if i sigma is 0 1 if you have 0 0 1 0, R means we just reverse this. So, 0 will come first then 1 then 0 0. This is the symbol we are using. This is the reversal of a string. Now reversal of a language means, the reversal of each string in that language.

(Refer Slide Time: 15:49)



$L = \{ w \in \Sigma^* \}$

$L^R = \{ w^R \mid w \in L \}.$

$L = \{ 01, 011, 110 \}.$

$L^R = \{ 01^R, 011^R, 110^R \}$

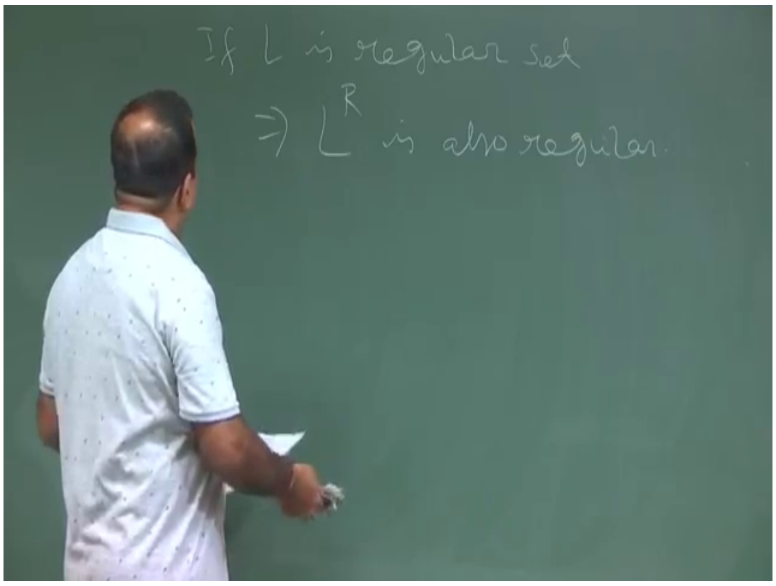
$= \{ 10, 110, 011 \}$

So, suppose we have given a language w sorry L. Now L R we just devoted by w R that is all.

So, w R set of all w R that w is belongs to L this connection ok. So, we just take the string and we reverse it that collection is L R. So, for example, if our L is a 0 1, 0 1 1, 1 1 0 ok. Now what is the L R? Reverse of each of this reverse of this, reverse of this, reverse of this. Now how to get the reverse? We will just take 0 1, 1 0, so it is 1 1 0, this is 0 1 1 this is the meaning of L R.

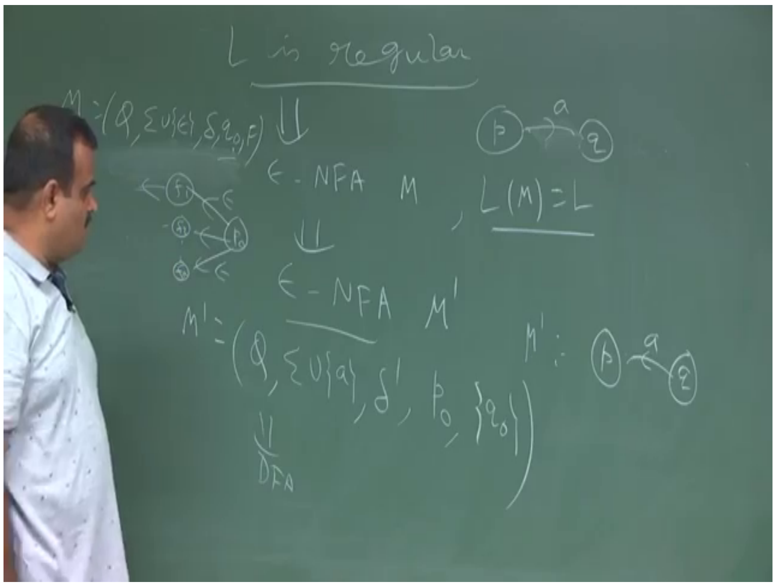
Now, the question is if L is regular then whether L R is regular or not; that means, whether the regular set is closed under the reversal the answer is yes. So, you have to prove that. So, you have to.

(Refer Slide Time: 17:15)



So, if L is regular set, this implies L R is also regular. So, how to show this? So, we can show this by either by regular expression, but we kept to formally show this, but informally we can just justify this like this suppose L is regular. So, L is regular means what?

(Refer Slide Time: 18:03)



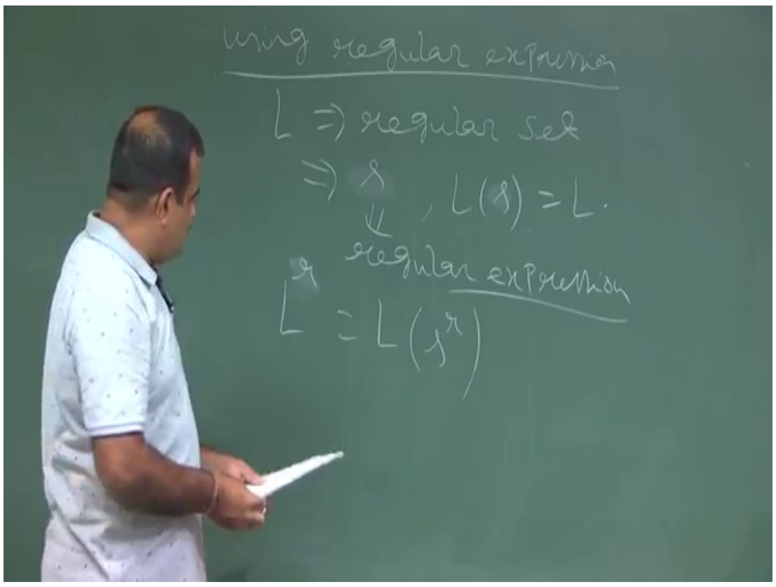
Suppose L is a regular either regular expression or regular set. So, this implies, we have a epsilon NFA or DFA, DFA can also feed as epsilon NFA.

So, we can construct a epsilon NFA M or M star. We can construct epsilon NFA M which is accepting this L. Now from here we can construct a epsilon NFA the epsilon NFA, M star by just reversing the arc. So, if there is a in M if there is a path like this p to q. So, this in M star will just make these arcs reverse like we will make it this way. So, we reverse the arc. So, if this is a, a could be epsilon also. So, in M it will be like this it will be q to q to p. So, we will just making the reverse of arc and then.

So, for this one. So, if this is say M is say Q this is a epsilon NFA Q sigma with epsilon sorry Q sigma with epsilon then delta, q 0, F and we are assuming it as accepting only 1 ok. So, this is the and then what is the M M prime? M prime is we are putting Q same sigma is also ok, delta is changing by reversing the arc and starting state we will need to have a new starting state p 0 and what is the final state? Final state will contain the q 0 the starting state of this that is all and then p 0.

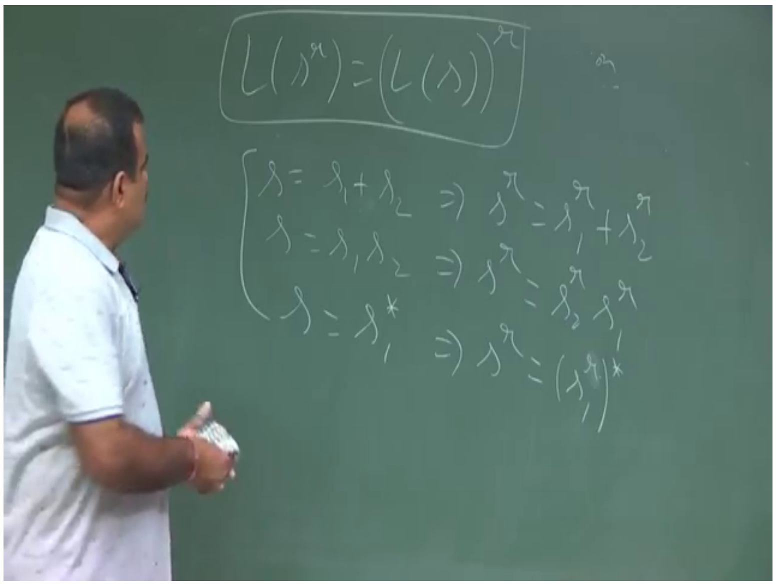
So, what we need to have? So, if there are say in this case if there are many final state f 1 f 2 like this f k then, from p 0 we have a epsilon move over here and we are reversing the arc. So, that way the only the reverse language will be accepted by this M prime ok, but this is the construction. And then from this M prime we can have a regular x we can convert into DFA and from DFA this is the regular expression we can have. But we can formally prove this by with the help of induction and with the help of regular expression.

(Refer Slide Time: 21:15)


using regular expression
$L \Rightarrow$ regular set
$\Rightarrow r$, $L(r) = L$.
$\Downarrow$ regular expression
$L^r = L(r^r)$

So, this is using the then this is the formal prove using the regular expression. The theorem is, if L is a regular language then this implies we have a r r we have a r such that L of r is equal to L this r is the regular expression because if it is a regular language then we have a DFA and from the DFA we can construct a regular expression ok.

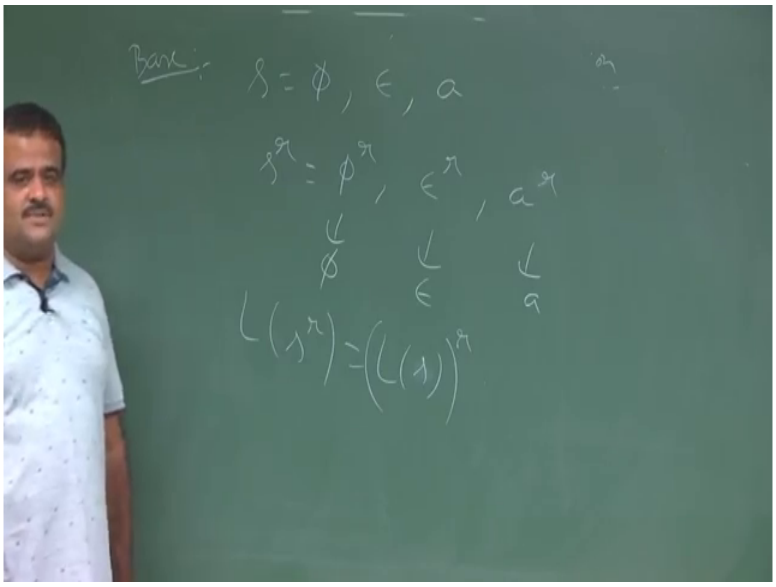
Then we want to show that, this L r is also regular the reversal. So, we can show that L r is nothing, but L of for this I must put s because I here use over here as a sorry this we can put s. So, this r L r is nothing, but s to the power r ok. So, this we have to. (Refer Slide Time: 22:41)


$\boxed{L(s^r) = (L(s))^r}$
$s = s_1 + s_2 \Rightarrow s^r = s_1^r + s_2^r$
$s = s_1 s_2 \Rightarrow s^r = s_2^r s_1^r$
$s = s_1^* \Rightarrow s^r = (s_1^r)^*$

So, that means, we can show that if we can show that s to the power r if we can show for a regular expression s to the power r is equal to then we are done. This will prove by help of induction and for that we need to have this result on the operator using the reversal what are the result? If s 1 and s 2 be two regular expression, then a if s is s 1 plus s 2 this imply s to the power r is nothing, but s 1 to the power r plus s 2 to the power r this is trivial because we can just take some example, this is the union this will be union and this is the reverse it.

So, this is trivial and this is true for concatenation also. This is s 2 to the s 2 r s 1 r this also you can just verify this is also trivial and if s is s 1 star then s to the power r is basically s 1 to the power r star small r ok. So, these are the result we will use for proving this. So, again we have two to prove this by induction, you have to show this is result is true for the base case this is we will prove this on the number of operators. So, suppose we assume that if the number of operated is 0 the base case. So, base case means we have only three possibilities.

(Refer Slide Time: 24:35)

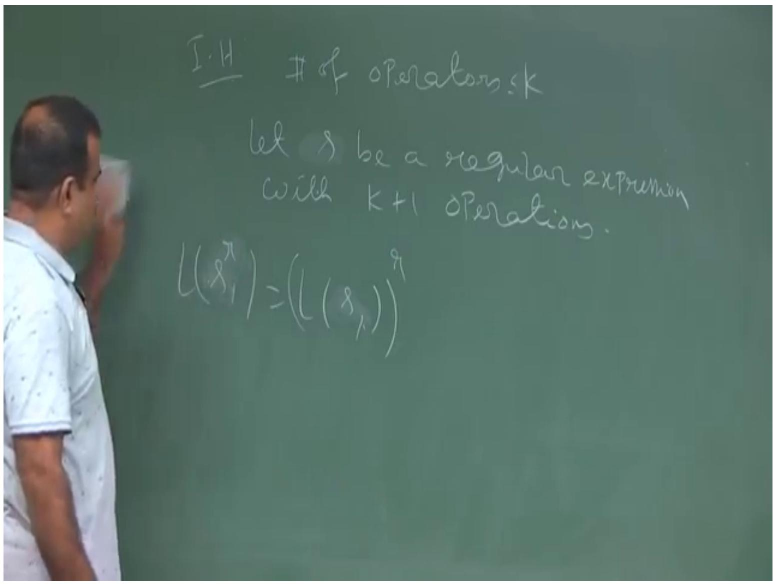


S is phi, epsilon, a where a is this is the base case here the number of operator is 0. Then in this case this is the base case with the 0 operator then in this case what is. So, what is s r? s r is this to the power r to the three cases a to the power r.

So, these are basically what these are basically same as phi, this is basically same as epsilon, this is this is a singleton set. So that means, L of s to the power r is same as L r to the power sorry L s to the power r because these are all our singleton sets if we reverse it, it will get the same set. So, this result is true for base case the number of operator is 0.

Now, if we assume that the result is true for up to more operator like up to k.

(Refer Slide Time: 25:45)



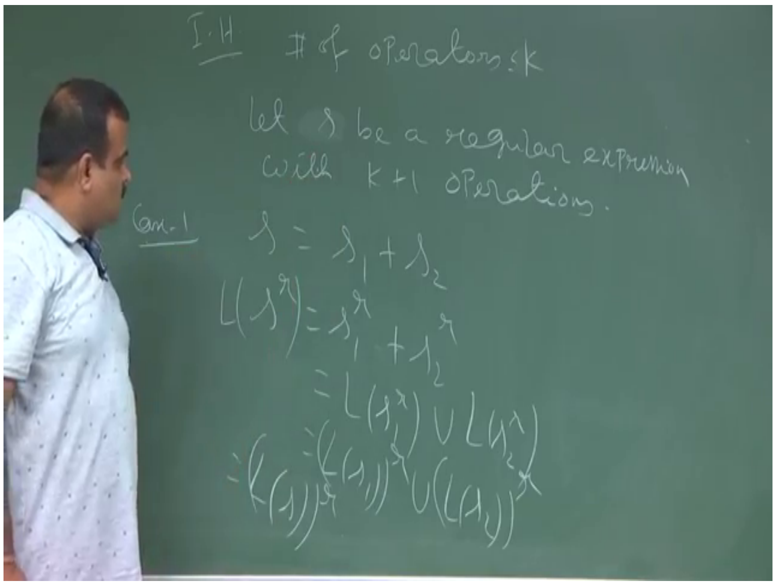
I.H # of operators ≤ k

let s be a regular expression with k+1 operations.

$L(s_1^r) = (L(s_1))^r$

Suppose this is the induction assumption supposes result is true for up to k number of operator is k or less than equal to k. Now we take a regular expression let r be a regular expression with k plus 1 operators use ok. So, we assume the result is true for this. So, if we have an operator like r 1 which is having in the r 1 if there are k number of operator then we have we can write that L of this is sorry r 1 not this is s 1 yes. So, s 1 to the power r is s 1 to the power r this is the assumption ok.

(Refer Slide Time: 26:57)



I.H # of operators ≤ k

let s be a regular expression with k+1 operations.

Case-1

$s = s_1 + s_2$

$L(s^r) = s_1^r + s_2^r$

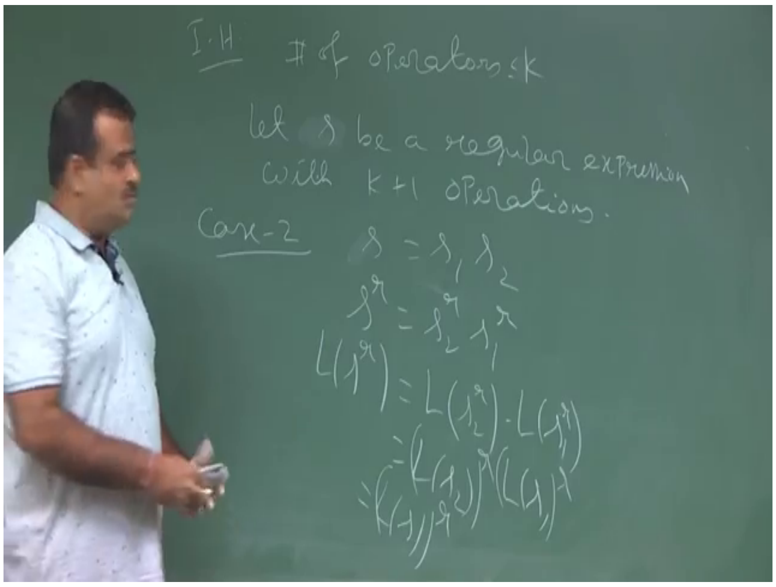
$= L(s_1^r) \cup L(s_2^r)$

$= (L(s_1))^r \cup (L(s_2))^r$

Now, if it is a k plus 1 operator then rk s can be written as either one of this form s 1 plus s 2 or s 1 s 2 concatenation or s 1 star and in that case all of these s 1 s 2 having less number of operator because we have used 1 operator already. So, they will be having the operator less than equal to k. So, by the assumption of induction, induction hypothesis we can say this result is true for both of these ok. So, this is case 1 is this. Then what is this? This we know so,

So, now, we take the r. So, this is s 1 r plus s 2 r this we know. Now this is basically L of this is nothing, but L of s 1 r union of L of s 2 r. Now by induction hypothesis we assume the result is true for up to k and there are at most k many operator as their. So, these can be written as s 1 to the power r union s 2 to the power r. So, that is all then this will be written as r of s to the power r. So, the result is true for k plus 1 operator and similarly, we can prove it for case 2, case 2 will be the concatenation.

(Refer Slide Time: 28:37)



I.H # of operators $\leq k$

let s be a regular expression with $k+1$ operations.

Case-2 $s = s_1 s_2$

$s^r = s_2^r s_1^r$

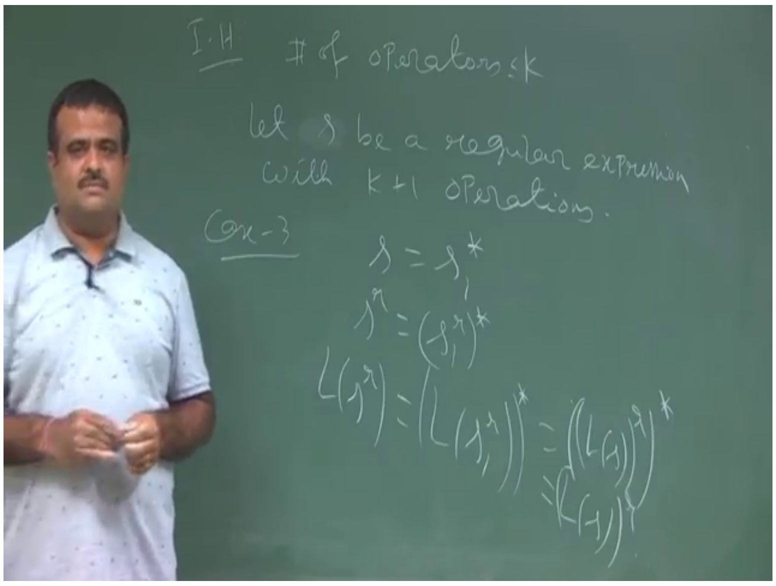
$L(s^r) = L(s_2^r) \cdot L(s_1^r)$

$= (L(s_2))^r (L(s_1))^r$

$= (L(s))^r$

Case 2 will be concatenation the s is s 1 s 2 in that case. So, s this r is basically s 2 r s 1 r. Now if you take this language, this is basically L of s 2 r concatenate with L of s 1 r. Now we assume this induction is true for this. So, this is basically L of s 2 r to the power r then L of s 1 to the power r. So, this is nothing but, L of s to the power r this is the case 2 and case 3 is the star.

(Refer Slide Time: 29:33)


I.H # of operators ≤ k
let s be a regular expression with k+1 operations.
Case-3
s = s1*
s^r = (s1^r)*
L(s^r) = (L(s1^r))* = ((L(s1))^r)* = (L(s1))^r

So, if this is sorry if s is s 1 star. So, you have used one operator. So, this s 1 must be having less than at most k operator because this is k plus 1 operator. So, if you take this, this is nothing, but s 1 r to the power star this we can easily prove. Now L of this is nothing but, L of s 1 r to the power star. Now we by induction hypothesis this is nothing but L of s to the power r to the power star.

So, this is nothing, but L of s to the power r. So, the result is true for this number of operator is k plus 1. So, by the induction method we can say the result is true for all. So, this is the formal proof of this closure under the reversal. So, if we have given a language, then if you convert if you revert all of the string of the language then that the new language is also be a regular language.

Thank you.