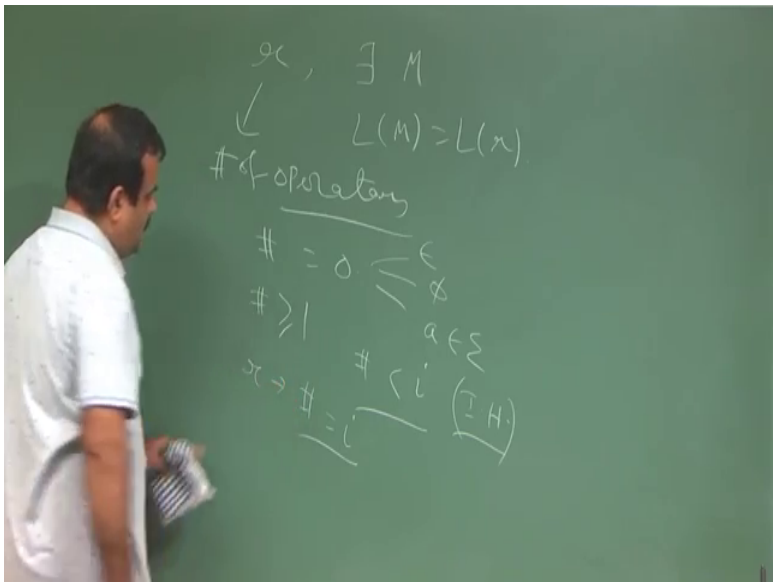
Introduction to Automata, Languages and Computation
Prof. Sourav Mukhopadhyay
Department of Mathematics
Indian Institute of Technology, Kharagpur

Lecture - 20
Equivalence of Epsilon-NFA and Regular Expression (Contd.)

So, we are proving the Equivalence between Regular Expression and the Epsilon NFA. So, this is the continuation of the from the last class.

(Refer Slide Time: 00:31)

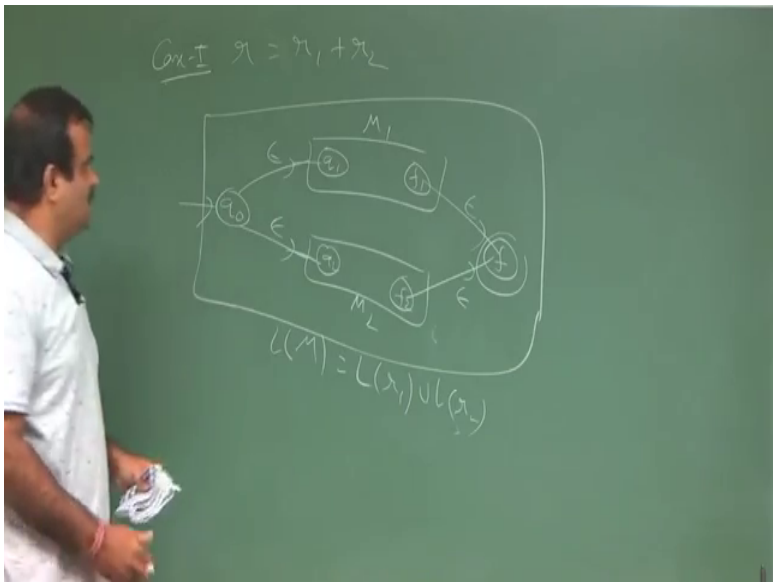


So, in the last class what we have seen? We have seen that I mean so, given a regular expression we are trying to show there exist a NFA epsilon NFA such that, so this we are proving by the help of mathematical induction on the number of operator operated on the regular expression.

Now, this is if the number of operator is 0, number operator is 0 that is the base case; that means, we have only 3 regular expression for this I mean this a is a input alphabet then we have the epsilon NFA for all these. So, base case we are we are through. So, for now if the number of operator is more than 1, more than 0 then we have a assumption that induction hypothesis is that if the number of operator is less than k less than i less than strictly less than, up to i we have a regular expression.

That is the induction assumption or induction hypothesis, then with the help of that we will prove that if the number of operator is i then we can construct a epsilon NFA which is accept which will accept the language of that r.

(Refer Slide Time: 02:11)

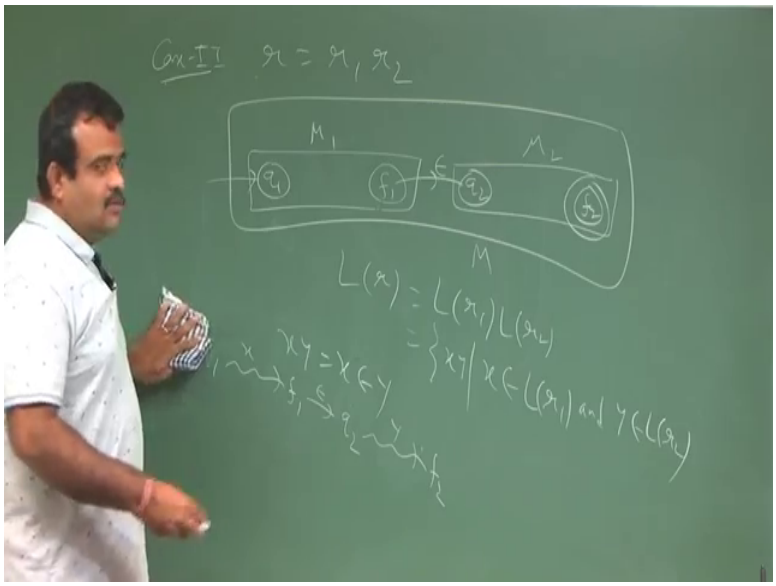


So, for that, if we assume r is a regular expression with the number of operator is i then r is, if r is greater than 0; greater than 0 then r is of the 3 form r is either some r 1 and r 2. So, this was the case I which we proved in the last class and for this is the case I. Now, r is i number of operators so that means, r 1 and r 2 must be there having less than i number of operator. So that means, in that case how; so, by the assumption what we have a regular we have epsilon NFA for r 1 and we have epsilon 1 for r 2 so that means, say that is M 1. So, that is q 1, f 1, q 2, f 2, this is M 1, this is M 2 this is by the induction assumption.

Then we construct a regular expression like this sorry we constructed a epsilon NFA like this, so q f so this we discuss in the last class. So, this is our M which is accepting a language. So, if this is the m, this is the final state. So, this is the M, then L of M is nothing but L of r 1 union L of r 2. So, this we proved in the last class.

Then the case II also we discussed case II is r 1 r 2.

(Refer Slide Time: 03:41)



Case II is r is r 1 concatenate r 2 in that case also we are using one operator so that means, this r 1 and r 2 will because r is having i number of operator. So, r 1 r 2 have must be having less than i number of operator. And in that case also we use the induction hypothesis that; there is a there are epsilon NFA for r 1, there is epsilon NFA for r 2, then we have we have seen that we can construct epsilon NFA for r. How? So, this we will discuss in the last class. So, the suppose this is q 1 and f 1 this is the M 1, epsilon NFA for r 1 and this was the final state for M 1.

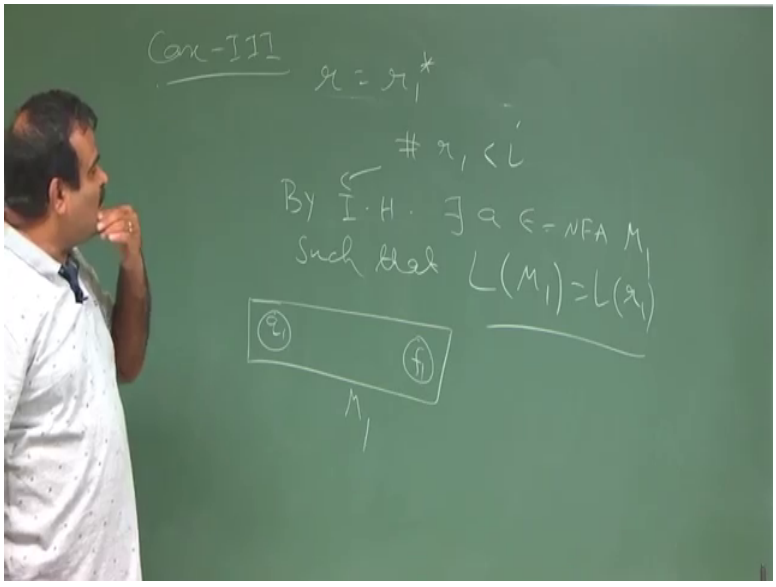
And here we have another important thing is that we assuming we are taking this epsilon NFA as only one final state. We could do for many final state, but to make this proof simple we are showing this result is true for, I mean our epsilon NFA having only one final state and there is no transition from the final state to any other state, ok. So, this is q 2 and f 2.

Now, how we construct in this is M 2. So, M 1 is accepting the L of r 1 M 2 is accepting L over r 2. Then how to construct the regular expression epsilon NFA for this? So, which accept this accepting means the concatenation this, so that means, this is x y such that x is coming from this and y is coming from this. So, this also we discussed in the last class. We take out this as a starting state and then we will add a epsilon over here and we make this is as a final state.

So, this is our M. Now, what is the language of M? Language of M will be consist of this type of x y, because x y can be treated as x epsilon y. So, we first take the x move from q 1, we take the x move we go to f 1 then form f 1 we take on the epsilon loop, we go to we go to q 2 then from with the help of y we go to f 2, f 2 is the final state. So, this is accepted. This change is accepted. So, this proof we have seen in the last class.

Now, today we will discuss the third case which is the star operation and then we will take one example on this, ok.

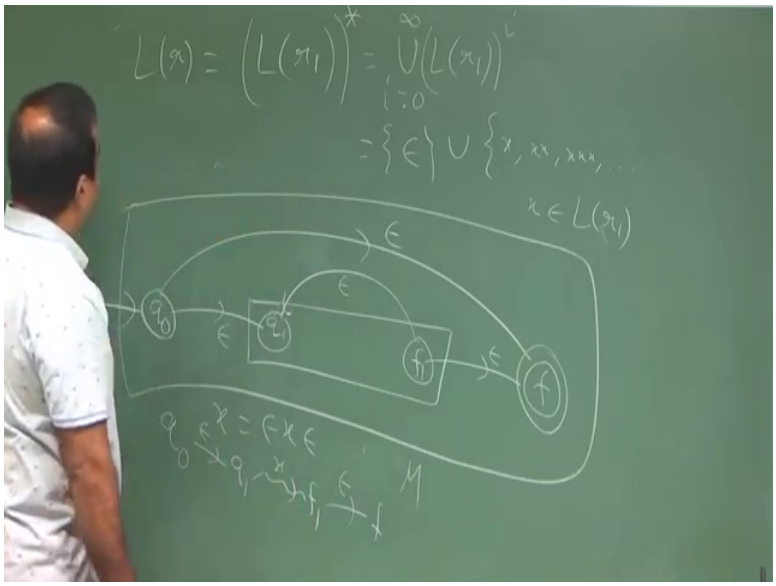
(Refer Slide Time: 06:34)



So, case III suppose r is r 1 star for some r 1, where r 1 is a regular expression which have being number of operator in r 1 is less than i, because you have already used one operator. So, it must be less than i I mean i minus 1. So, in that case we have a epsilon NFA for r 1, that is the assumption that is the induction hypothesis. So, we do have a regular expression i, we do have epsilon NFA for r 1 which is accepting this L of r 1 then using that we need to construct a regular expression for r.

So, this means by the induction hypothesis there exist a epsilon NFA M 1 such that this is by the assumption, and then we are going to prove that we can construct epsilon NFA for r, ok. So, suppose this is our epsilon NFA for q 1, f 1, this is our M 1. Now, we need to construct epsilon NFA which will accept this r 1 star. So, what do you mean? Which will accept the r 1 star I mean accept means the language wise so that means, we have to construct a epsilon NFA which will accept L of r 1 star.

(Refer Slide Time: 08:21)



So, the star is nothing but union of L of r 1 to the power i, i time concatenation. So, this is from 0 to. So, this is if we exclude epsilon this is all the term like say x, xx, xxx, like this, where x is coming from L of r 1 x is a stream coming from L of r 1 like this. I mean (Refer Time: 09:00).

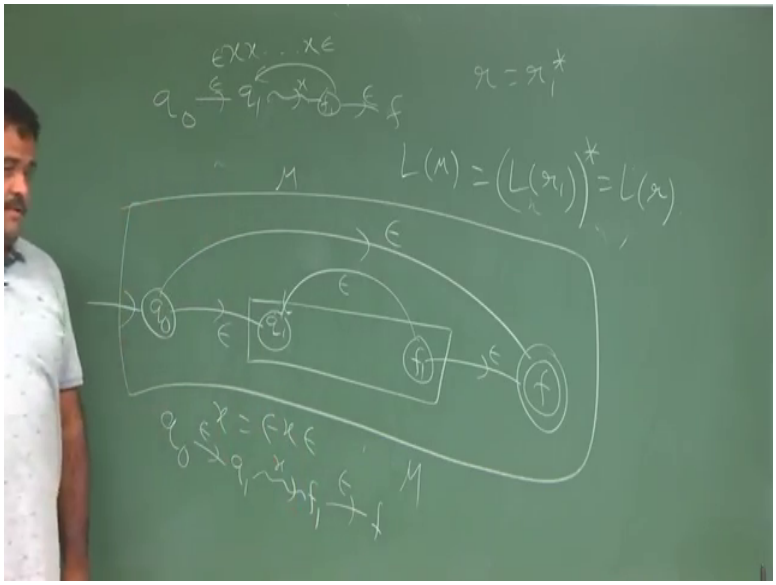
So that means, we need to accept the epsilon, we do not know whether epsilon is accepted in here or not. So, we need to accept the epsilon and then all the string like this, ok. So, how to construct it? So, let us try to construct the power M which will accept this L of r. So, we start with a new starting state and we have a new final state this is the final state f, ok.

So, in order to accept the epsilon NFA in order to accept the epsilon we take a move for q to this with epsilon then we take a move from here epsilon, then we have a move x there, then we go to epsilon. So, this will accept x. How? Because x can be written as epsilon x epsilon; so, we start with q 0, then we take a epsilon move we go to q 1, then we take x string we go to f 1 and then we take the epsilon over here when we go to f. This is the way this is accepting x.

Now, if you have to accept xx then? For that we need to have a epsilon over here from here to, we need to have a epsilon move over here. So, this is our m. So, this will accept due to this epsilon move, this will accept any time types of any number of x, x star. So, x y we go there then again come back here then we take the x move go to the final state.

xxx 3 times we off here. So, how many times we want? Then it is accepting all types of x star like xxxx all this way.

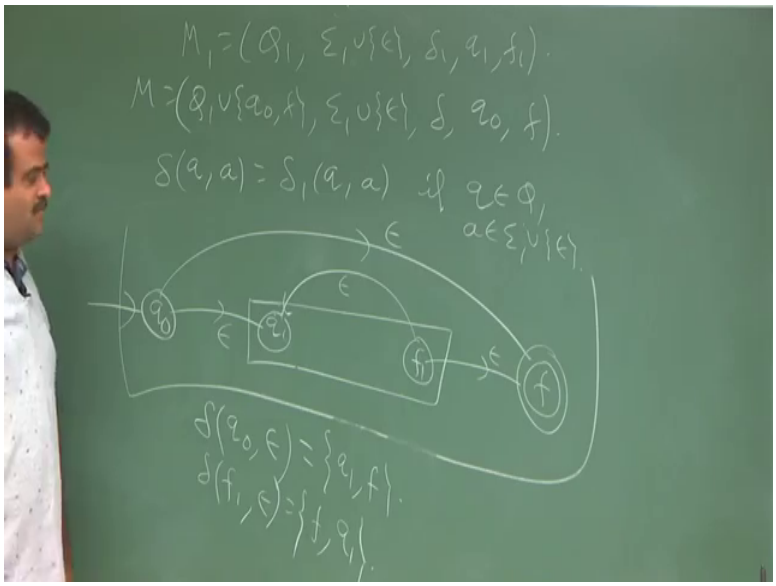
(Refer Slide Time: 11:32)



So, this is our M, this is our M which is accepting L of r ok, which is accepting L of r, because it can accept any string like x star, xx any number of x because we will start from q 0 we take epsilon because this consists of epsilon also because epsilon is the null string. We take epsilon move we go to q 1 then from q 1 we go to F 1 by x. Now, depending on the number of x we will keep on hopping there and then finally, we when the x is ended, we take it because they say epsilon over here go to f accepted.

So, that means, this is if this is M L of M is nothing, but which is nothing, but L of r. So, this is the epsilon NFA for the regular expression r which is of the form r 1 star, where r 1 is the regular expression which is having less number of I mean less than i number of operator. So, we have epsilon for this by the assumption induction hypothesis then this, ok.

(Refer Slide Time: 12:53)



So, formally what is M? What is M? So, if M 1 is Q 1 sigma 1 along with epsilon then we have a delta 1, a 1, f 1. Then what is our M? M is nothing but, so we have Q 1 but we need to add two more state then we have a sigma 1 is same, delta and we have a new starting state and new final state, ok.

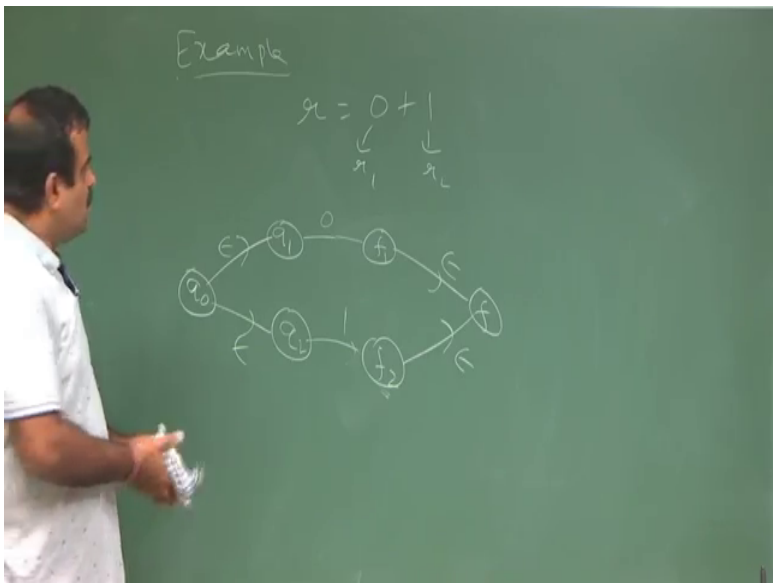
Now, what is the rule? So, rule is delta of we have added few epsilon move delta q 0 epsilon is nothing but q 1 f, and delta of f 1 epsilon is nothing but it can either go to the final state or it can come back to the starting state of M 1, ok. And what about and delta of; delta of q comma a is nothing, but delta 1 of q comma a if q is belongs to q is belongs to Q 1 and a is belongs to sigma 1 sigma including epsilon. This is this we have to add in the previous cases also, because we may have epsilon over here inside because this is the epsilon NFA. So, done so, this is accepting the language L of this, ok.

So, now, we take some example. So, this is the proof. So, that means, in the all the cases because you know all the cases you have we are getting a epsilon NFA which is accepting this.

(Refer Slide Time: 15:16)

So that means, what we have? We have suppose r is a regular expression with number of operator is i, and then by the assumption then if we have a regular expression with less than i operator then we have a epsilon NFA. So, then we have seen that we have epsilon NFA for r, and we have already proved the base case that means, we have already seen that we have epsilon NFA for i is equal to 0. So, by the mathematical induction we can say that this proof is for all r. So, we have a for given any r we have epsilon NFA which except that r, ok; so, will take some example now.

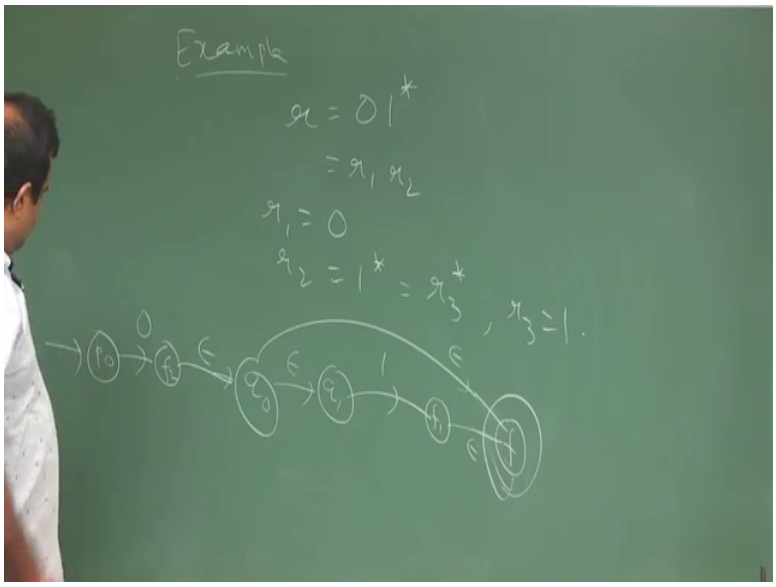
(Refer Slide Time: 16:14)



Let us take one example then will see, ok. So, we want to construct a, we want to construct a regular epsilon NFA for this, we want to construct automata which is accepting the language this. So, how to construct it? So, this is of the form this is how many operator has there 1 2 3, 3 operators are there. So, we have to break it into few parts like if we have only this. So, if we have only 0 plus 1 then we know the regular expression for this, because this is r 1 this is r 2 and we have q 1 say f 1 this is with 0 and we have q 2, f 2 this is with 1.

This two are NFA epsilon NFA corresponding to these two and then we have to construct a epsilon NFA for this. So, how to construct? We take this two are no more final state for this. So, this is f 1, this is f 2, then we take a q 0 and we take a f. So, we take a epsilon over here, epsilon over here, that is all. So, this is our new f, ok.

(Refer Slide Time: 18:06)



So, now if we have this one like 0 1 star; so, it is concatenation of r 1, r 2. So, r 1 is, say r 1 is 0, r 2 is 1 star. So, again this is concatenation of some r 3 star, where r 3 is 1. So, then how to get the regular expression for: how to get the regular expression for r 2? To get the regular expression for r 2 we need to get the regular expression for r 3. So, r 3 means we have a q 1, f 1 is the final state, this is 1 this is the starting state, this regular expression for r 3.

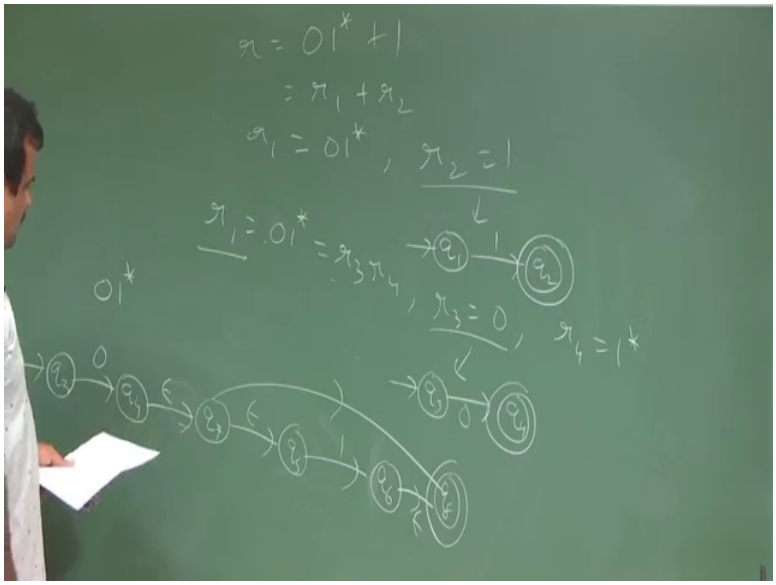
Now, how to get r 3 star, 1 star? So, for that we have to have say q 0, then we have f over here then we have a epsilon move over here, we have epsilon move over here, then we

have epsilon move over here then this is the regular expression and this is the NFA this the starting state. This is the epsilon NFA and this is the final state accepting one star, ok.

Now, we need to have a regular expression which is accepting 1. So, that is easy we have a p 0, 0 or p 1 f 2 we have a news, so this is the 1. Now, can we have a regular expression this is the product. So, this is we have to have here. So, this is or we can have with this. So, this we cannot make it final state, this is some f. Now, we have a p 1 and we have say f 2. So, this is our 0 is here sorry, 0 this is 0 one star. So, 0 is 1. So, anyway p 0 and this is f 2. So, we have a 0 over here, so we make this as a initial state of this. Then we take epsilon move over here and then we make these as a final state. So, this is accepting 0 1 star, ok.

Now, we want to have a regular, we have to have epsilon NFA which is accepting 0 1 star plus 1. So, that is the 0 1 star plus 1. So, we will do it step by step.

(Refer Slide Time: 21:11)

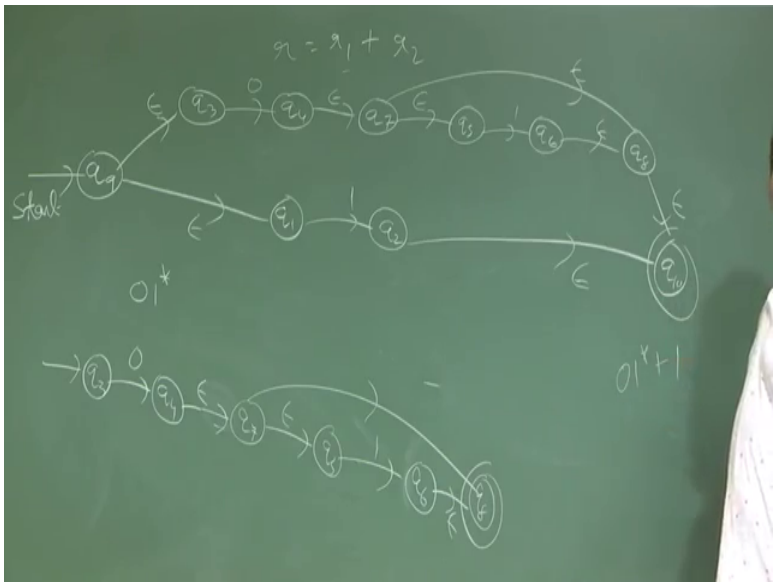


So, like this we all right this is r this is r 1 plus r 2, where r 1 is 0 1 star r 2 is 1. Now, for this we have a epsilon NFA like this q 1 this is the starting state this is one this is our kind of M 1 or M 2. So, now, r 1 is nothing but, r 1 is 0 1 star this is r 3 r 4 where r 3 is 0 r 4 is 1 star. Now, for 0 we can have a regular expression easily, so q 3 this is the starting state and you have a q 4 which is the final state with the 0 move. And then r 4 is 1 star which is nothing but r 5 star, where r 5 is 1.

Now, we can have regular a we can have epsilon NFA for this. So, we start with say q 5 and q 6 is the final state we will take one move, go on. Then we can construct a regular expression for r 2 sorry, we can use this board. So, we can construct a regular expression for r 2 using this.

So, this will be nothing but. So, this is q 6 we can have q 7 as a starting state and we have a epsilon move over here and we have a q 8. So, we take a epsilon move over here this is the final state you have epsilon move, this is 1 like this. So, this is our r 4. Now, once we have r 4 then we can have we know the r 3 then we can have r 1, r 2. So, this will be like this. So, then we can have a NFA for this. So, this will start with say q 3 and q 4. So, this is with 0, then we have epsilon move over here, this is the starting state and this is the final state this is basically 0 1 r star r 1, then with that we have to add this.

(Refer Slide Time: 24:24)



So, let us write this, this is r 1 plus r 2. So, r 1 we already got a regular expression for r 1, so which is q 3 I sorry epsilon NFA for r 1 q 3, q 4 we have a 0 move q 7, q 5, q 6, q 8 this was the final state earlier. So, let us write the move. This is epsilon q 7 to this, this is epsilon, this is 1, this is epsilon and we have a q 7 to q 8. We have a epsilon here. Now, this was for r 1.

Now, for we need to bring the r 2 over here. So, r 2 is nothing but q 1 q 2, so this is 1. Now, we need to have this joint. So, we take a q 9, this is the starting state, this is the starting state then we take a epsilon move, over here, and epsilon move over here. And

then we make a q 10 for the final state of this you take epsilon move over here we take a epsilon move over here, ok. So, this is our final NFA like this. So, we start with q 9 and we end with this is the final state q 10 and this is accepting the language 0 1 star plus 1, 0 1 star is this one, plus 1 is this one.

So, this is the way we construct the epsilon NFA for a regular experience. So, there is a equivalence between the we can have any example in the lecture note, but this is the way how we construct the, how we show the equivalency between regular expression and epsilon NFA.

So, in the next class what we do, we have we will see the equivalency between the DFA and the regular expression that means, given a DFA will try to construct a regular expression which is accepting the same language. So, then these are all equivalent; so, that we will discuss in the next class.

Thank you.