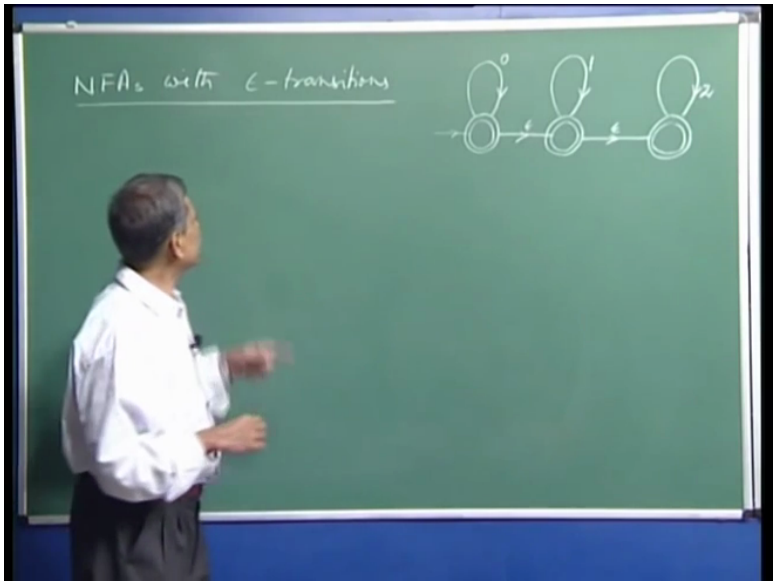
Theory of computation
Prof. Somenath Biswas
Department of Computer Science and Engineering
Indian Institute of Technology Kanpur
Lecture 10
NFA's with Epsilon transition

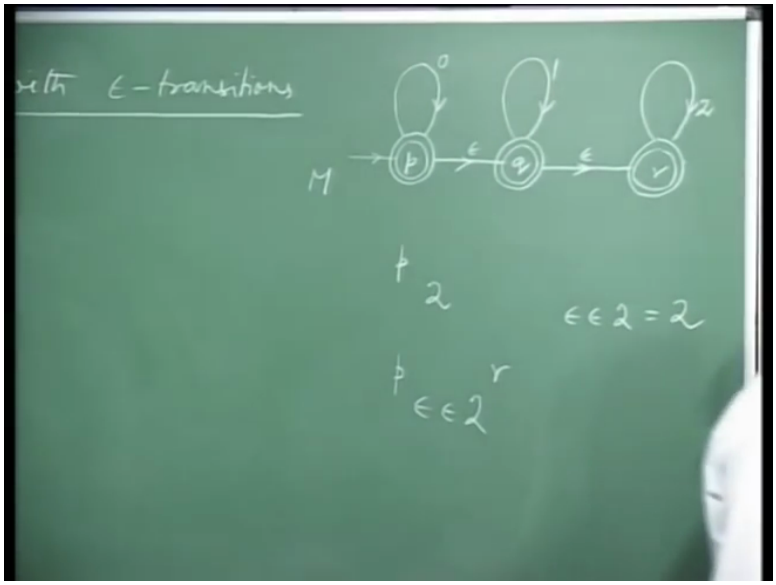
(Refer Slide Time: 0:33)



Today we will consider a generalisation of NFA's recall NFA's themselves wear a generalisation of DFA's and now we are considering a generalisation of NFA's and that generalisation is that such automata NFA's with Epsilon transitions will allow transitions not only on symbols as well as on the empty string Epsilon. Now here is a, this is an example of such an automaton.

Let us first informally understand how this Epsilon transitions work? First notice that there are 3 states in this machine M and all these 3 states are final states and yet as we will see it is not the case that this automaton accepts all strings. Now you see for example the automaton is in this state and from this state it can take an Epsilon transition to this state, what does it mean?

(Refer Slide Time: 1:50)

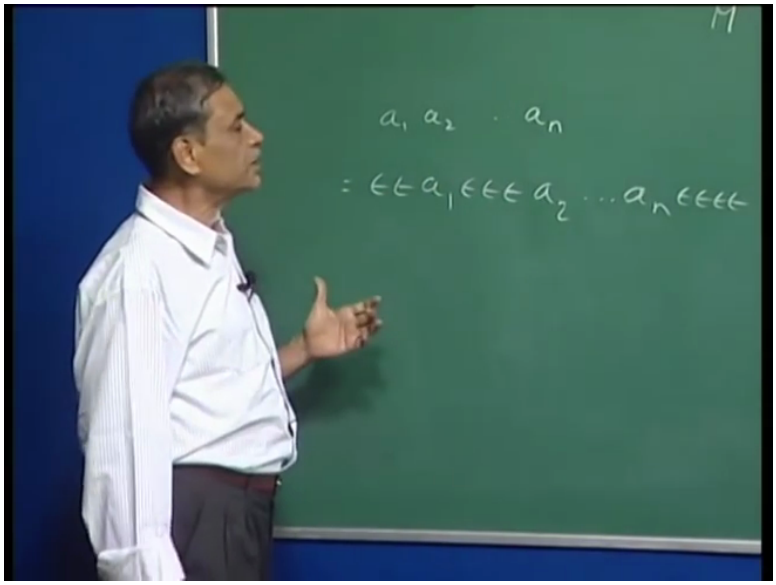


That imagine these let me name these states as P, q and r at some point of time the machine M is in state P and let us say 2K now, the next symbol is the symbol 2 because as you can see in this transition diagram that the alphabet that machine uses is 0 1 and 2. Now in this is the state sequence and in this state sequence at some point of time the machine is in state P and the symbol 2 comes.

Now what the machine can do? It has the choice of not using this symbol immediately what is on the input? But it can take a transition like this Epsilon and then it would be in q but then again from this q it can again take another Epsilon to go to the state r in other words we are thinking of this 2 as if there were 2 Epsilon's and followed by 2 but of course 2 Epsilon's followed by 2 is of course the string 2 itself.

And therefore when such transitions are allowed the see what the machine can do? It is in state P takes this Epsilon transition to go to q from q takes Epsilon transition to go to r and now it consumes the symbol 2 and therefore it can be in the state r just on the symbol 2 because it has the ability to make transitions on Epsilon.

(Refer Slide Time: 3:45)

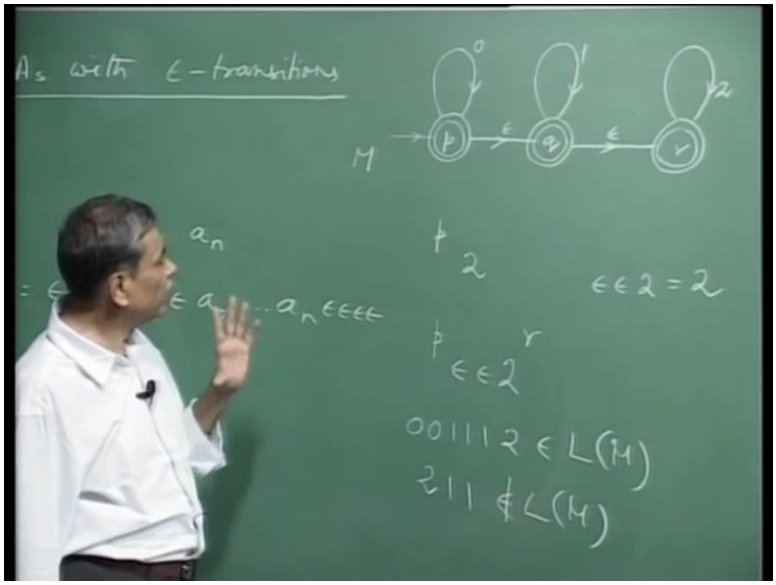


First of all we should understand that suppose I have an input string a1, a2, and I have a machine, some machine which does allow Epsilon transition and what do I think about these where do these Epsilon's lie? One way of thinking of this string is that it is of course same as infinitely many other strings which are obtained by adding 0 or more Epsilon's either in front of the entire string or between 2 symbols or at the end of this.

So for example this as you can see because Epsilon is the empty string this is surely same as Epsilon Epsilon a1, Epsilon Epsilon Epsilon a2, so on a n and maybe 4 Epsilon's. When I say that the machine some machine with Epsilon transition as on the input such a string and then I would like to figure out which all states the machine can go to. On scanning this string I need take care of all such strings not just the string a1, a2, a n but as I said padding 0 or more Epsilons in between 2 symbols, in front of the string as well as at the end of the string.

So it appears that now we have a hard task to figure out even which all states the machine can be given an input string because there are any number of Epsilon's that can be there all over the string. As we will see that this problem can be taken care of quite nicely and simply but before that maybe we should understand this particular example a little bit.

(Refer Slide Time: 6:30)



If we see this machine we can of course see that the string 001112 will be accepted by this machine, why? Because on the first to 0 of course the machine remains here in state P, second 0 machine remains here and now it takes an Epsilon transition it considers there as if there is an Epsilon here between this 0 and the 1 comes to state q and then remains in state q for the next 3 symbols which are all 1's which it can.

And now it takes an Epsilon transition to go to r and there is this 2 and so it comes back to 2 therefore what I am seeing is that it is possible for the machine to go from its initial state to one of its final states, right? Using the transition including of course Epsilon transition. So therefore this string is in the language of the machine M.

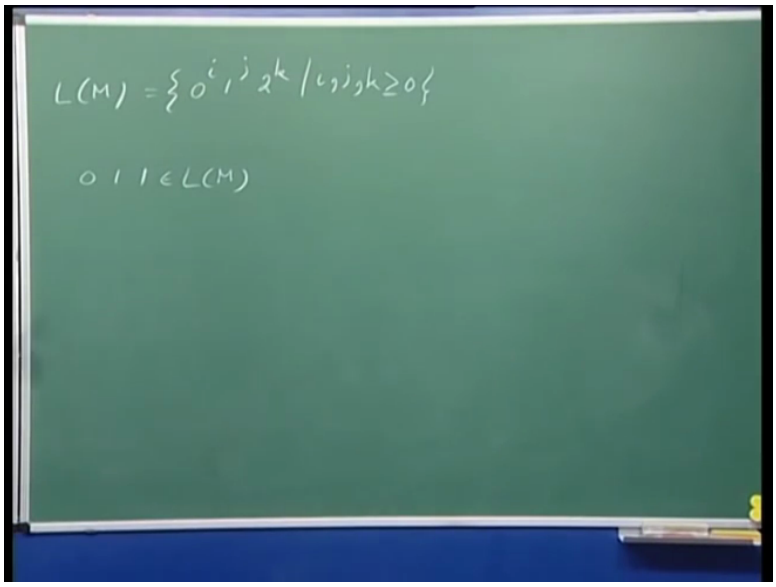
Remember that for an NFA what we said was a string is accepted by the NFA, if such a string can take the machine from its initial state to one of the final states this can, on the other hand if it had not seen or used the Epsilons the way I just told you it might have been in some other state, in fact it will be in some other state which is alright but having considered that an Epsilon here on which the machine makes transition.

Another Epsilon here the machine makes a transition on these 2 epsilons, the machine can go from its initial state to one of its final states, right? At the end of the string. On the other hand can this machine except the string let us say 2 1 1 that I claim that this string is not accepted by this machine because what happens, the first symbol itself is 2, right? To consume this symbol the only way the machine can do so, is to go all the way up to this state which of course it can do.

Right in the beginning it takes 2 Epsilon transitions; it considers that there are 2 epsilons in front. So they need right in the beginning taking those 2 epsilon transitions it comes to the state r and now it consumes this symbol 2, after that what can it do? Now 1 is the there are no Epsilon transitions possible from here. So no transition will be defined even if the machine consumes the first symbol 2.

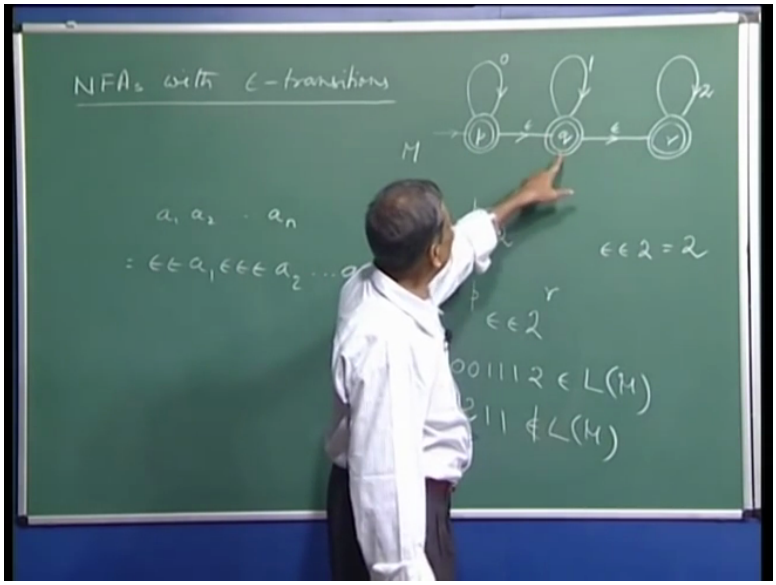
And we should be able to convince ourselves that there is no way how many epsilons you may put either in front or in between 2 symbols or at the end this particular string will not be accepted by the machine M.

(Refer Slide Time: 10:06)



Again let me describe the language accepted by this machine and we will definitely see the justification for that. LM is the language in which I claim the strings are of the form 0i 1j2k where the constraints on ijk is really nothing except that each one can be 0 and all that we have to ensure for a string to be in this language the way we have defined that you may have 0 or more number of zeros but all of them, they must win the beginning followed by 0 or more number of 1's followed by 0 or more number of 2's.

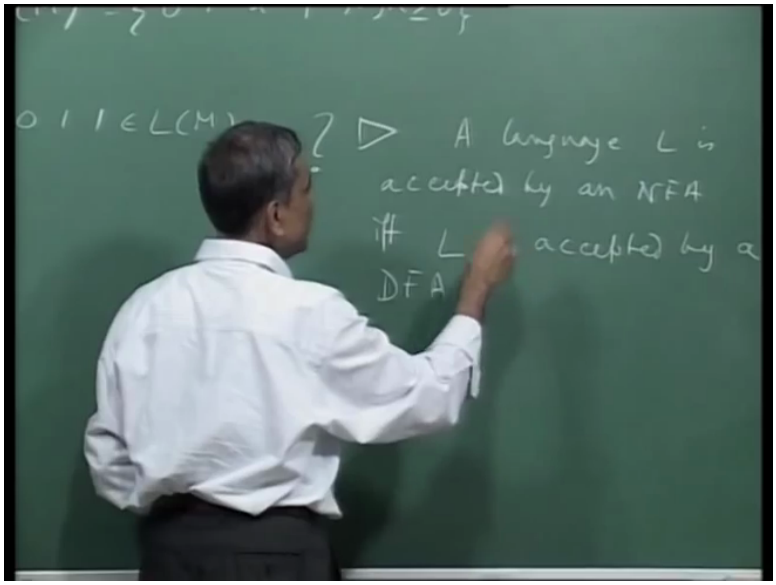
(Refer Slide Time: 11:29)



In fact you see for example let us just take 0 1 1, indeed first 0 it is here then it takes an Epsilon transition remains here for the next 2, talking of this 0 1 1 and it is easy to see that machine can go from the state the initial state to q on 0 1 1 and q is a final state therefore this will be in the language and so on and this manner can at least intuitively convince ourselves that that this is the language accepted by the machine M which has Epsilon transitions.

Now our question will be can NFA's with Epsilon transitions accept some language which is not regular the question is relevant, obviously because when we started with DFA our definition was regular languages are precisely those languages which can be accepted by DFA's then we generalize the notion of DFA to NFA's nondeterministic finite automata and that generalisation was in this form that we said from a state on a symbol in case of an NFA there can be 0 1 or more transitions.

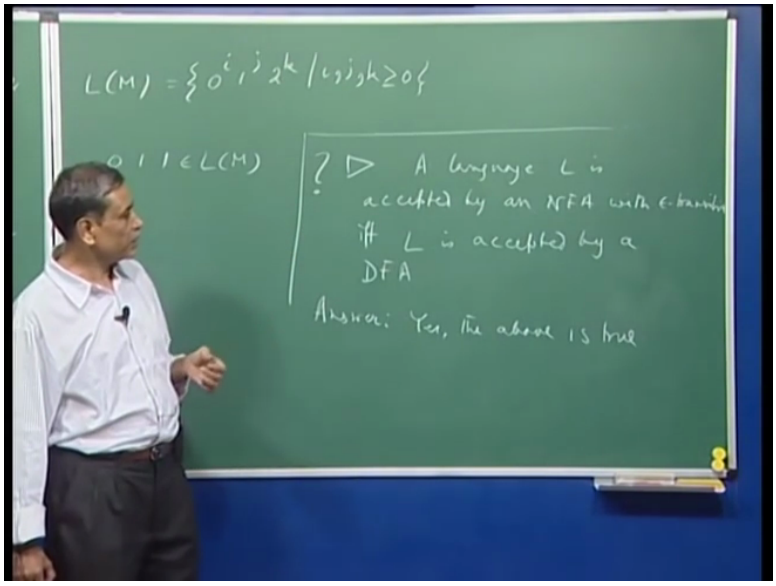
(Refer Slide Time: 13:11)



Unlike in case of DFA where there is precisely exactly one transition possible and yet with this generalisation one result that we had seen that a language L is accepted by an NFA if and only if L is accepted by a DFA and we proved this result by taking an NFA and finding an equivalent DFA by you know that that DFA had set of all subsets of states space of machine accepted I mean the NFA machine and we showed that that particular DFA will accept the same (()) (14:00).

So this result tells me that NFA will precisely capture will precisely accept the same class of languages which is the class of regular languages and now we have yet another generalisation we have added Epsilon transitions to NFA's. So the question is, is it true that a language L is accepted by an NFA?

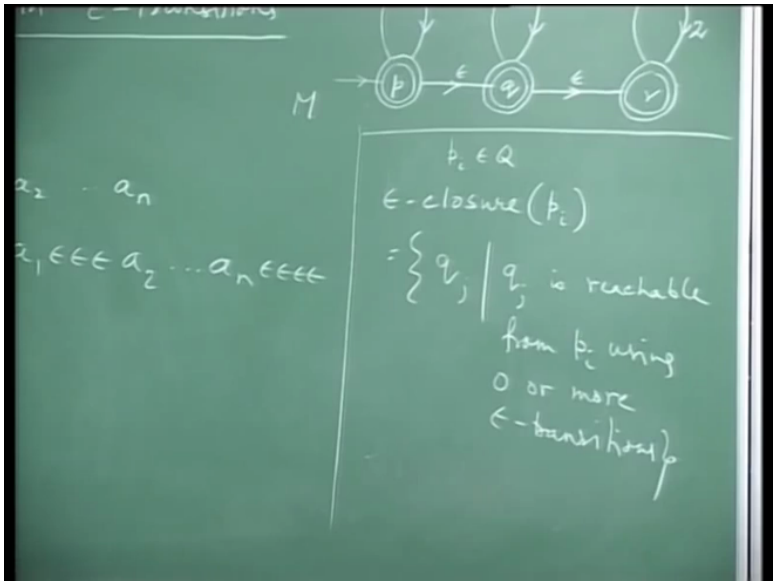
(Refer Slide Time: 14:57)



Now if I put this "with Epsilon transitions" if and only if L is accepted by DFA, is it true? Because if it is in that case of course that even giving this extra power of Epsilon transitions we do not go beyond the class of regular languages. On the other hand if it is the case that using Epsilon transitions, we can do more than what a DFA or an NFA can do then we have something interesting.

And in fact we will show that the answer is yes, the above is true and let us prove this fact that an NFA with Epsilon transitions and we will prove it in the same manner that is given an NFA with Epsilon transitions we will construct a DFA which will accept the same parameters and therefore if that construction we managed to show then of course the answer is yes that even NFA's with Epsilon transitions will accept languages which can be accepted by DFA.

(Refer Slide Time: 16:30)

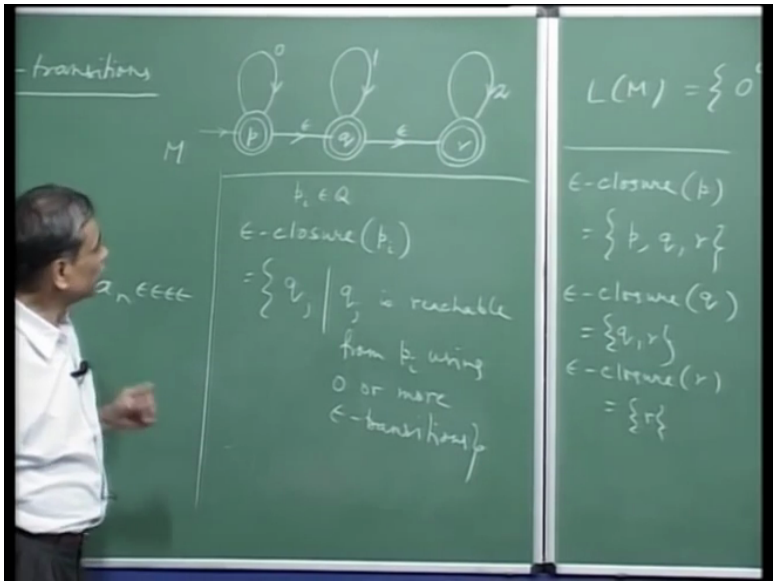


In order to consider this question we need to take care of this problem that I mentioned before that now since there are Epsilon transitions available to the machine the input cannot be consumed symbol by symbol we need to take care of possibility of Epsilon transitions between 2 symbols maybe at the end of the string as well as before the string. Now it might appear since there are many many in fact infinitely many strings which are possible in this manner by plugging in 0 or more epsilons all over that the problem is going to be something which is horrendously difficult.

But as it happens there is a very simple notion which will let me handle all this infinite strings from one simple input string to all possible strings with lots of epsilons all over in between symbols etc and that simple idea which will let me take care of all these situations is the notion of Epsilon closure.

Now this let me first define Epsilon closure for any state let us say Pi, informally it is the set of all states Pj, so you understand Pi is a element of the set of states of the machine and Epsilon closure of a state is the set of all qj such that qj is reachable from Pi using 0 or more epsilons transitions. So we can understand this definition more clearly if we just look at this example above.

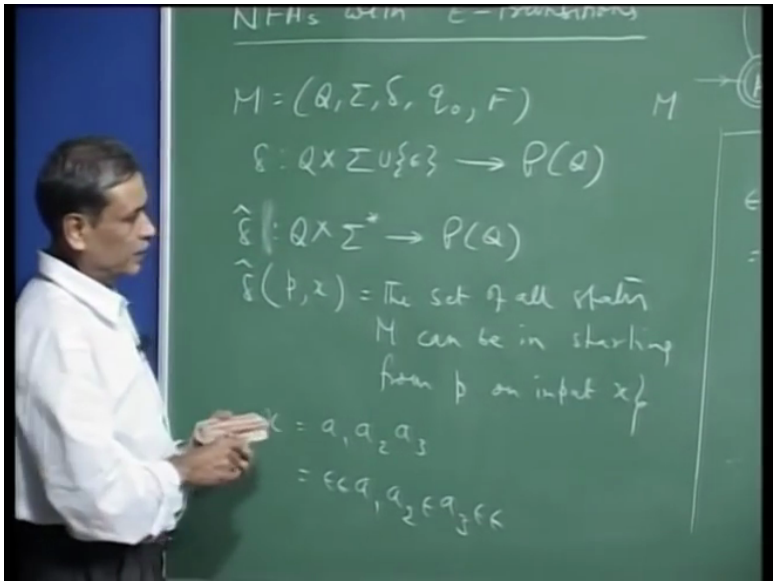
(Refer Slide Time: 19:28)



So what is the Epsilon closure of this state P? Remember Epsilon closure is going to give me the subset of states each of which can be reached from the state using only Epsilon transition 0, so things all the states which are reachable, so it is not difficult to see that Epsilon closure of this state P is P, q and r, why?

Because of course P you can reach from P using 0 epsilons if you are just there, q you can go using this one Epsilon and r also you can go from P using 2 epsilons. So let me do the rest other 2 states, what is the Epsilon closure of q? It is not difficult to see it is q set consisting of q and r and Epsilon closure of r is precisely that state itself, right? And given a transition diagram you can compute Epsilon closure of any state by simple reach ability algorithm many of which we know, alright.

(Refer Slide Time: 21:37)



Now how do we handle what is the set of states the machine can go to on an input string using any number of Epsilon transitions? So what I am trying to say is that suppose M is an NFA with Epsilon transitions Q, Sigma, Delta, q0, F this definition is like NFA where is the distinction between plain NFA's which do not have Epsilon transition, it is here that Delta is now a mapping from Q cross Sigma union Epsilon to power, right, why?

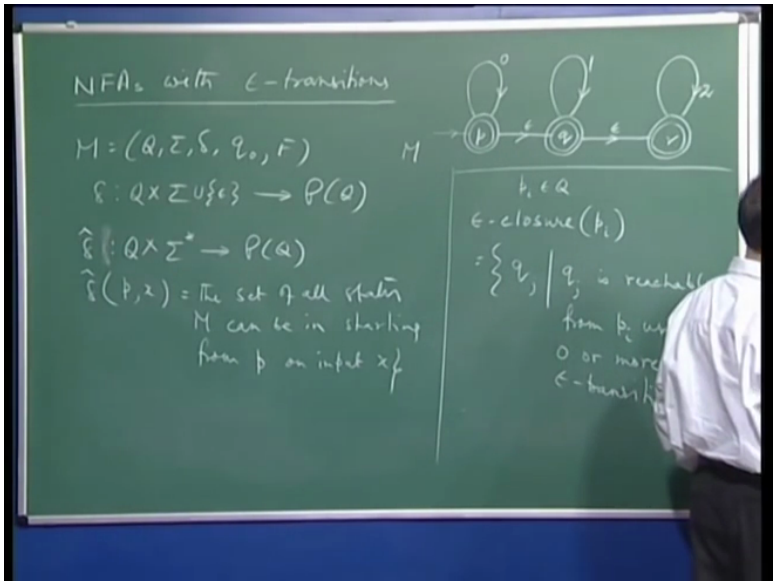
Because there are Epsilon transitions, so you are in a state of course you can consume an input symbol as well as you can use an Epsilon to go to some other state, number of states. Now if we managed to define Delta hat properly, what you want Delta hat to be? Delta hat should be as before that it should it should be a map for Q cross Sigma star to power set of Q and the idea would be the mapping we should define it in this manner such that Delta hat of let us say P and x is the set of all states M can be in starting from P on input x.

Now I am not putting epsilons explicitly in this because the string x as I know is same as a string, you know suppose x is just, suppose x is a1, a2, a3 but this is also same as Epsilon Epsilon a1, a2, Epsilon a3, Epsilon Epsilon, right? So Delta hat must take care of all these Epsilon transitions because by string x when I give x the machine can either take it as just plain simple sequence of symbols or of course it can take it in this manner that between symbols before or after the string there are epsilons as many as it wishes to have.

So all such strings which are actually seem as the string x, in fact what I am trying to say is even when you pad in all pad into this string all these epsilons, the string actually is same

because Epsilon is the empty string. So my Delta hat should be able to handle all this and how do I handle that?

(Refer Slide Time: 25:54)

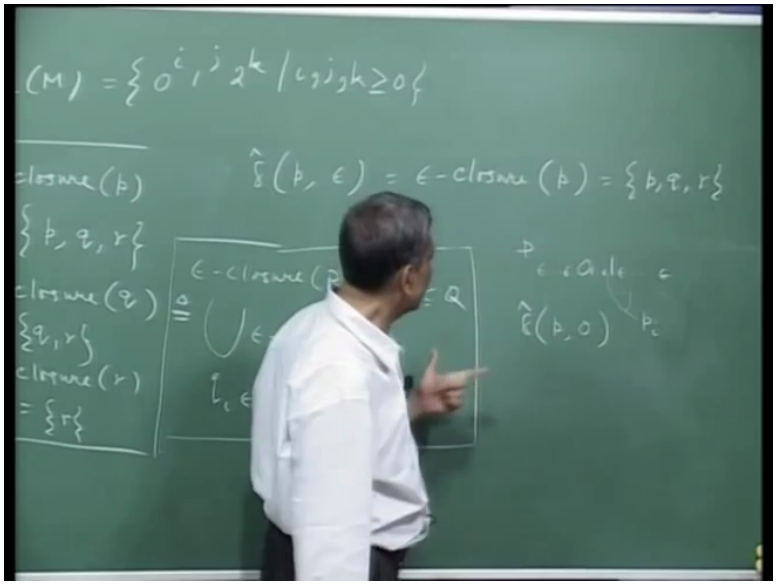


Okay, so before formally defining Delta hat let me work out this example and the situation will be quite clear. So let us see what is Delta hat where of Epsilon or this case? From the start state P Delta hat P Epsilon, so this is of course the empty string Epsilon and now I am asking, which all states the machine can be? This particular machine can be in the beginning without consuming any symbol from the input.

Now as we can see of course which can be in P as well as it can be in q because it just uses one Epsilon as well as it can be in r. So now do you see that if Delta hat of P Epsilon is the Epsilon closure of the start state of this machine P M and this is just simply this which is of course as we have seen here, it is P, q, r and now let us say the machine is in a after scanning some strings some portion of an input string the machine is in some of these, you know some of these states, right?

Now this notion of Epsilon closure of course can be extended to a subset of states rather than a single state where P is the subset of Q the set of all states of the machine and this is simply the union of Epsilon closure of you know some qi where qi is this set that is easy to see rather this is really a definition but it makes sense that is I can ask the same thing that the meaning of when I have a set of states rather than a single state what I am trying to find out through this definition is the set of all states the machine can be or the set of all states reachable using only Epsilon transitions from anyone of the states in this set of states P, okay.

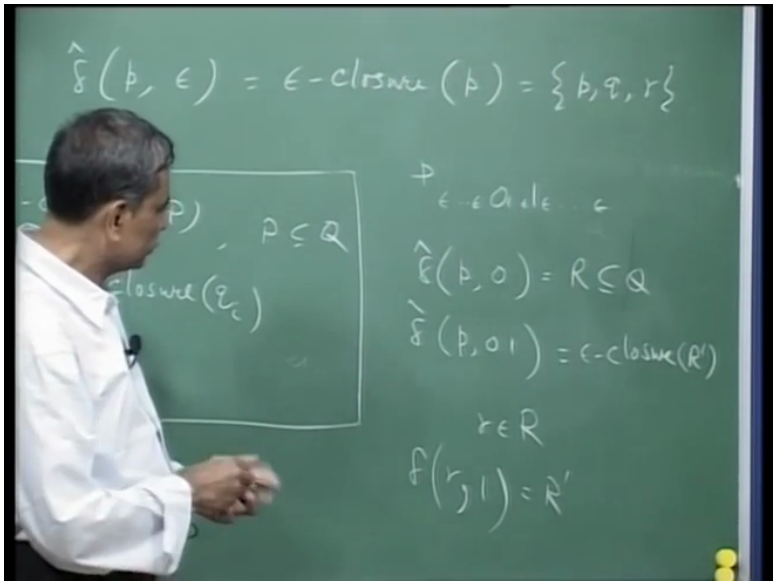
(Refer Slide Time: 29:29)



So this is clear and now let us say I just have this string 0 1, right? And I want to know which all states this particular machine can be on the string 0 and 1. Now let us assume inductively that I know Delta hat of P which is the initial state of this machine and for this that is the 0, that is I know which all states the machine can be starting from P on consuming one symbol and of course now any number of Epsilon.

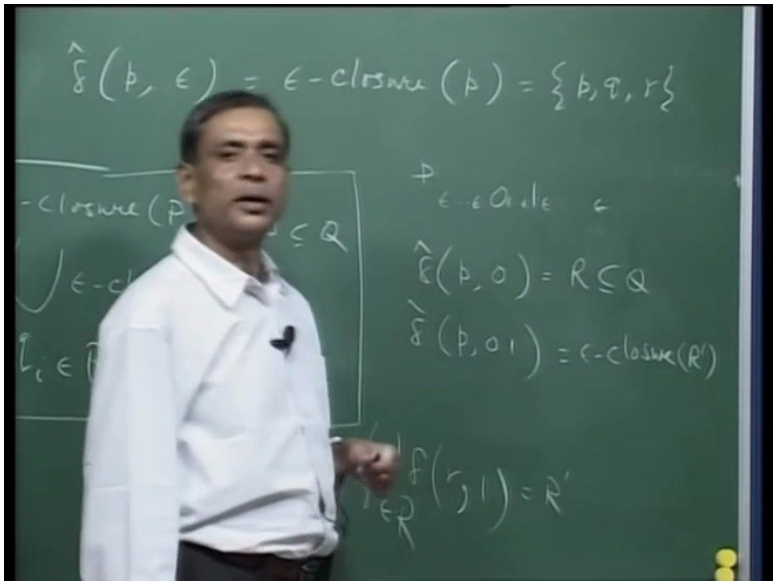
So you see this string can be thought in terms of there are any number of epsilons here, any number of epsilons here and any number of epsilons here. So suppose using after 0 if you use some Epsilon transitions when you are in some state let us say Pi starting from P, view see that by definition Pi will be a member of this set of states, right? Whether you use 0 Epsilon or one Epsilon or any number of Epsilons all those things must be taken care of by my definition of Delta hat, if it is done correctly.

(Refer Slide Time: 31:03)



So in that case if I know that Delta hat P0 is some R which is the subset of Q, now what is going to be Delta hat of P01? So you see what I need to take care of is that this 0 with some epsilons before, some epsilons after all the states that one could reach that is already there in this capital R. So I take any state let us say r from capital R and then take Delta of r and now this 1 has come and I will get a number of states let us say R dash but now I need to look at the effect of having epsilons, so then all I need to do is to take Epsilon closure of what R dash, is this clear?

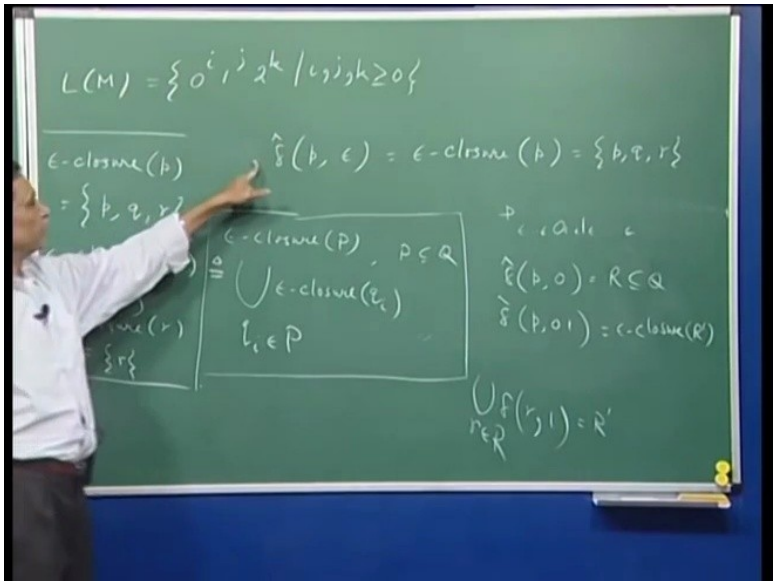
(Refer Slide Time: 32:51)



So let me define R dash more clearly, what is R dash therefore? R dash is, this is the union of all r in capital R. So once more, so Delta hat of P of 0 with any number of epsilons etc can take me to one of these states R, now 1 comes just using that one from that these set of states we can go to any one of these states, the machine can go to any one of the state's R dash but now after that 1 of course there can be any number of epsilons which the machine can use, so we take Epsilon.

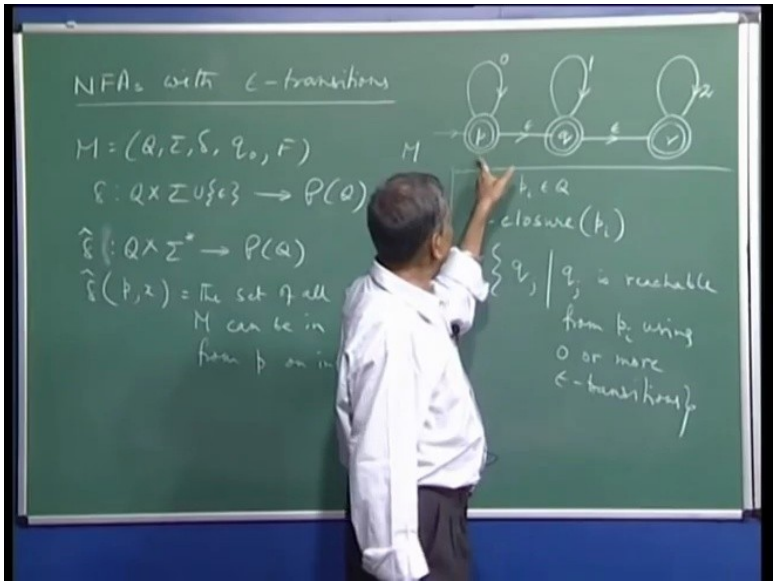
You see Epsilon closure is the built-in devise of taking care of 0 or more number of Epsilon transitions, you think it will come clear because we were just saying which all states one can reach using 0 or more number of Epsilon transitions, alright.

(Refer Slide Time: 34:18)

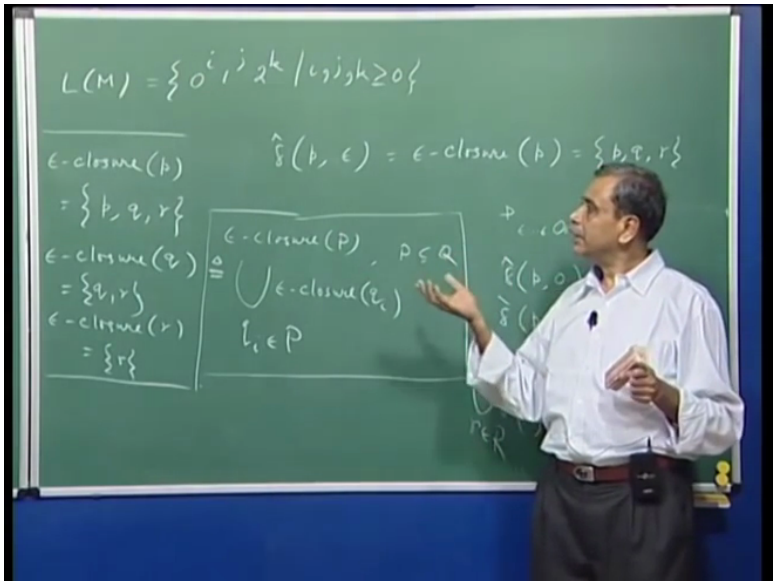


So this definition from that example we can complete inductively the definition of Delta hat because we have already given the base condition and then the induction will follow using this idea.

(Refer Slide Time: 34:28)



(Refer Slide Time: 34:49)

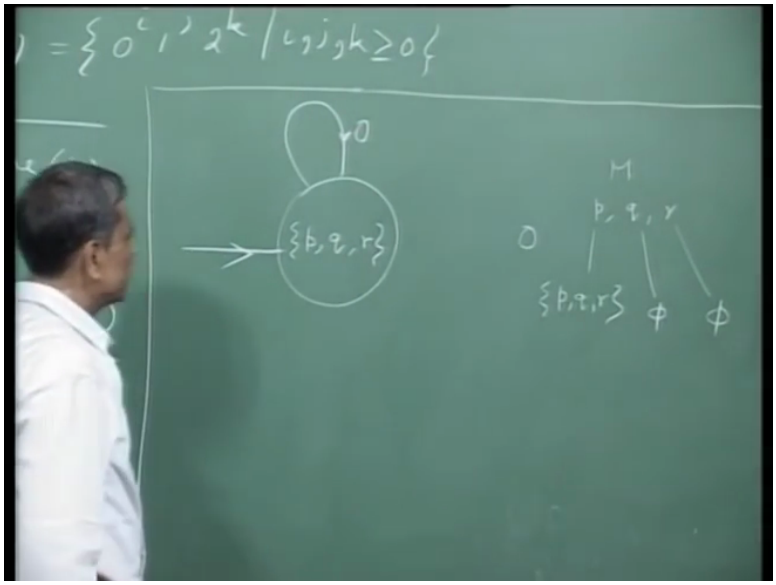


Instead of writing all that, let me do it for this particular example, what the corresponding DFA will be if I take care of Epsilon transitions? In other words what we are saying is from an NFA using some idea we found from an NFA we can go to a DFA which will accept the same language and I will show you that I can do the same for this (()) (34:58) also which has Epsilon transitions.

So I am attempting to get a equivalent DFA equivalent in the sense which will accept the same language as this NFA Epsilon transitions using the idea that we had used to construct equivalent DFA from NFA's. So let us see initially which all states it remember the DFA general will have states the subsets of the states space of M, of course and set of states is P, q and r.

So let me ask initially which of this subset of states the machine can be? It has not seen any input but then it could have seen it could have used the Epsilon initially of course the machine is in the state P but without consuming any input symbol just using Epsilon it can also be in q, it can also be in r.

(Refer Slide Time: 36:16)



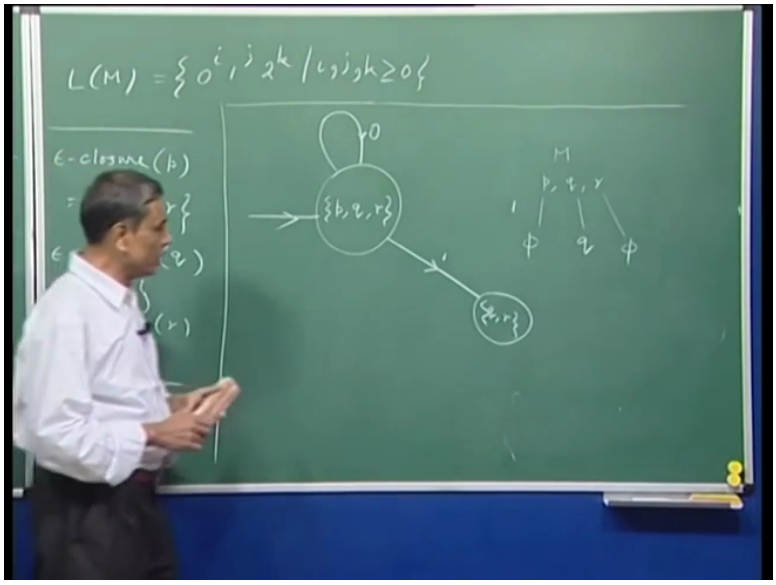
So my DFA clearly will have a state corresponding to p, q, r, alright this is the initial state of the DFA, if I managed to build the DFA and now let me think of the transitions from this state on the symbols and if there are only 3 symbols 0, 1 and 2. So on 0 which are the states that NFA can be? Suppose it is in anyone of these states, NFA is in anyone of these states p, q or r.

Which all states it can be when it sees 0? So the M is in anyone of these states p, q or r and now 0 came, right? P on 0, of course it moves to P but then on this same 0 it is just used one 0 but it can now used some Epsilons and therefore I need to take the Epsilon closure of this which is Epsilon closure of P I know is p, q, r from P of course it can go to p, q, r.

From q on 0 which all states it can be if it is already in q this machine M now 0 has come, you see there is no transition defined on q and therefore q on 0, so this is the effect of input symbol 0 will be in the State Phi, subset of states which is empty and similarly from r you can see on 0 it will go to there are no transitions, so this is on 0 from r it can go to the empty scale only.

So on 0 if the machine was in anyone of these states the machine M was in anyone of these states it can be in anyone of these states p, q, r by taking the union of these that is what I mean, so you see therefore on 0 if I am thinking in terms of a DFA which had this state which is a subset of this set of states of M on 0 it comes back to the same state and now let us consider what happens from this state 1 comes.

(Refer Slide Time: 39:11)

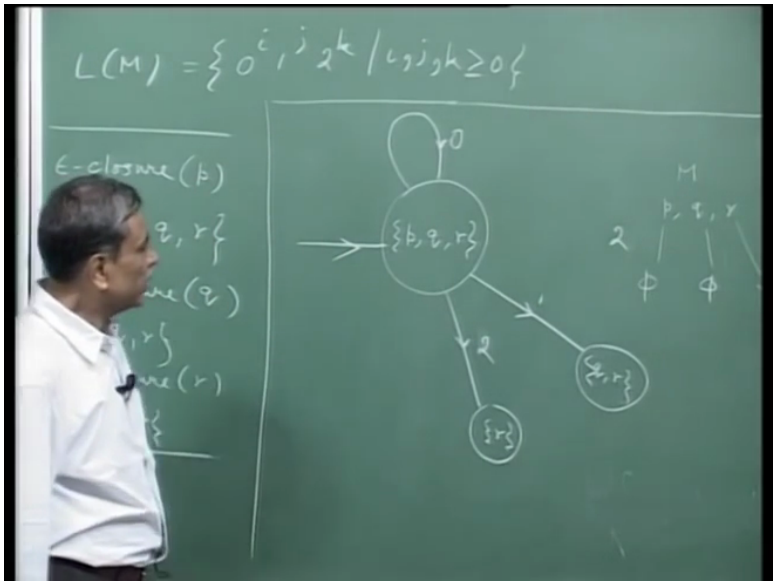


Again what is the meaning? That the machine M is in this state that means it is in anyone of these states p, q or r and now 1 has come, right? Now from P on 1 which all states the machine can be? Do I need to consider that is first it will take some Epsilon and then the q and then to, then what is the effect? I do not need to.

See the way I said that if I am using this idea clearly then I just see I am in P, 1 has come can I go to any state actually from P on 1, let us say I do not go anywhere, so this is empty but from q on 1 I can come to q and from r on 1, from r on 1 nothing is defined, so it is empty state but now I need to take the enclosure of q, enclosure of q is qr, so on 1 it will go to state, which is this state.

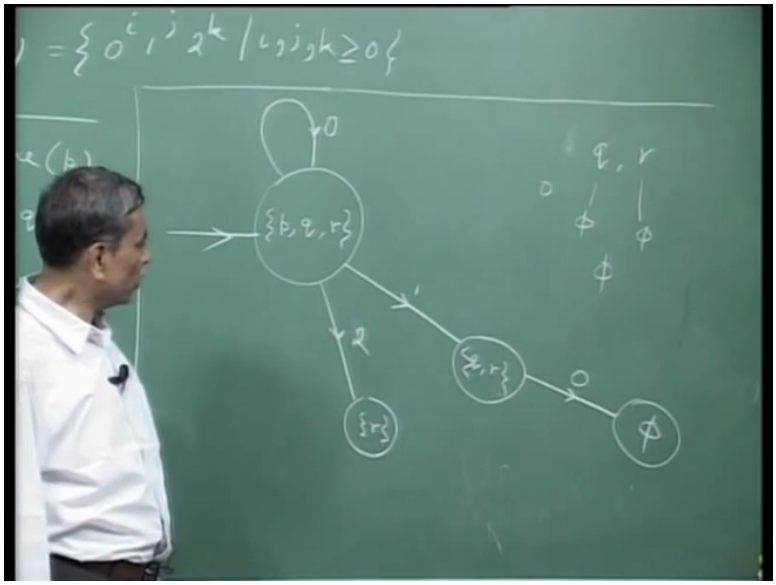
This is the same idea that we had used for NFA but only thing we are doing is we are taking care of the a possible Epsilon transition and in doing all this if you notice what we are doing that the machine is already here that means having considered possible Epsilons before, now it is in here and now it is about a particular symbol, so now 0 1 and now let us say 2 comes here which all states the machine can be?

(Refer Slide Time: 41:15)



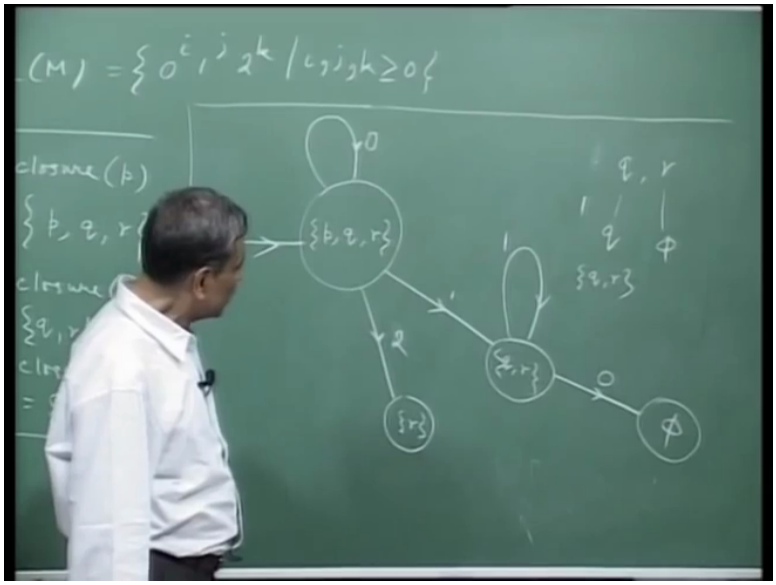
So you can see the same ideas phi P on 2 it will there is no transition defined, q on 2 no transition defined but it is r on 2 it can be here back r but now I need to take the closure of this, closure of r is of course r itself, okay.

(Refer Slide Time: 42:06)



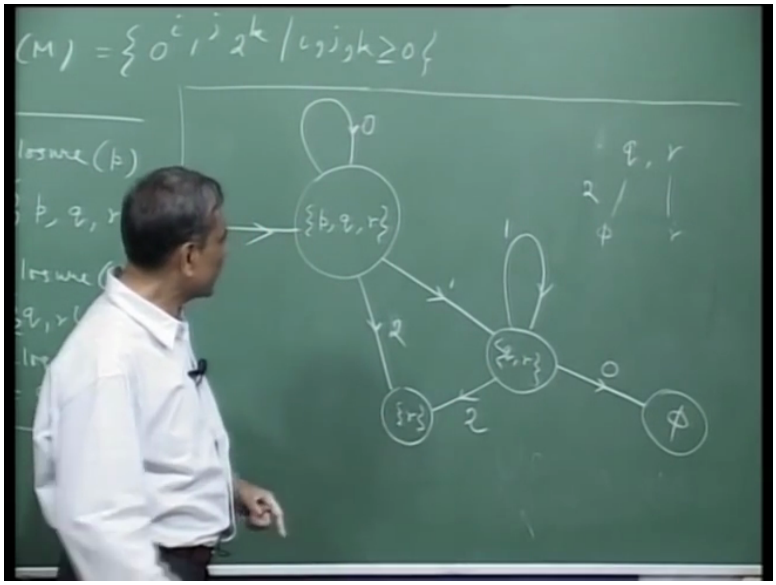
So I am in q and r and let me expand these states, supposing capital M machine is in one of these states q, r and now 0 comes. You see from q on 0 cannot go anywhere, no transition is defined from r on 0, no transition is defined take the Epsilon closure of this, so it Epsilon closure also will give me only the empty state set of states, empty set and so therefore from here on 0 I go to state which basically corresponds to the empty subset of the set of states of M.

(Refer Slide Time: 42:50)



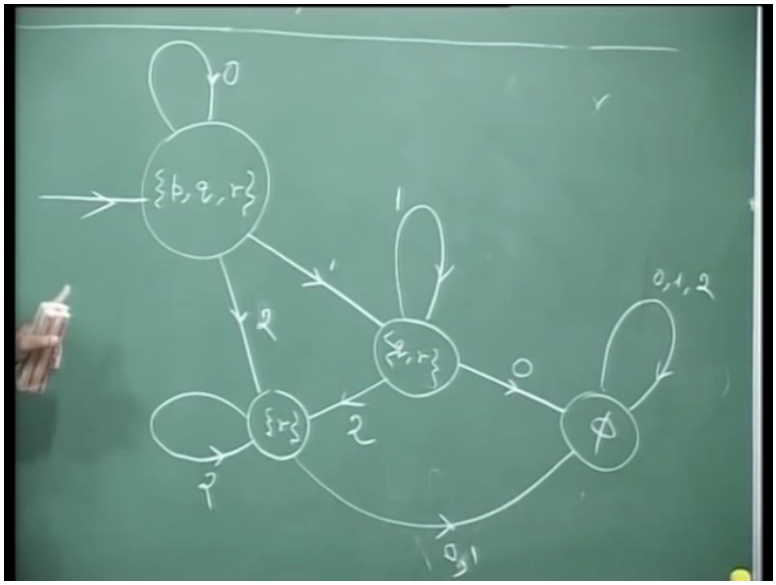
What about 1? If one comes again let us do this, from q on 1 I can remain here I can go to q, r on 1 there is no transition defined, so this is empty and now I will take the Epsilon closure of this which is q, r, Epsilon closure of q is q, r, so if 1 comes here we are back to this state.

(Refer Slide Time: 43:38)



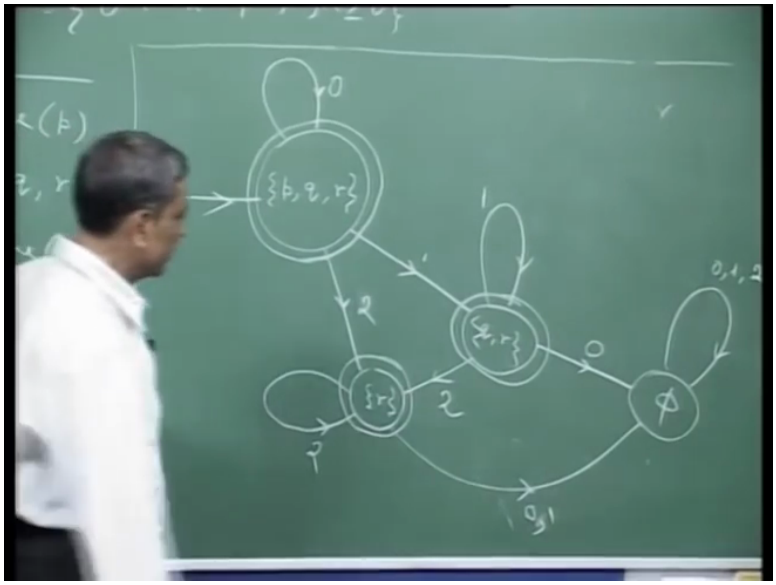
And if 2 comes, so let us do that we are in anyone of these states and now 2 came from q on 2 we have no transition defined, so this is q on 2 we go to empty set, r on 2 we come to r and now take the closure which of course means we are in this, okay. So this state is completely defined for all the symbols its transitions, this state also and now we state but that is easy to see r on 0, what will happen?

(Refer Slide Time: 44:23)



So if this is r the machine is in r, now 0 comes no transition is defined which means we come here, r on 1 again the same situation, so we will come here 0 as well as on 1, from this state go to this one however on 2 we remain here and then take the closure which is r itself and if 1 is in this state the machine is in empty state then whatever be the transition input symbol 1 remains there, right.

(Refer Slide Time: 45:39)

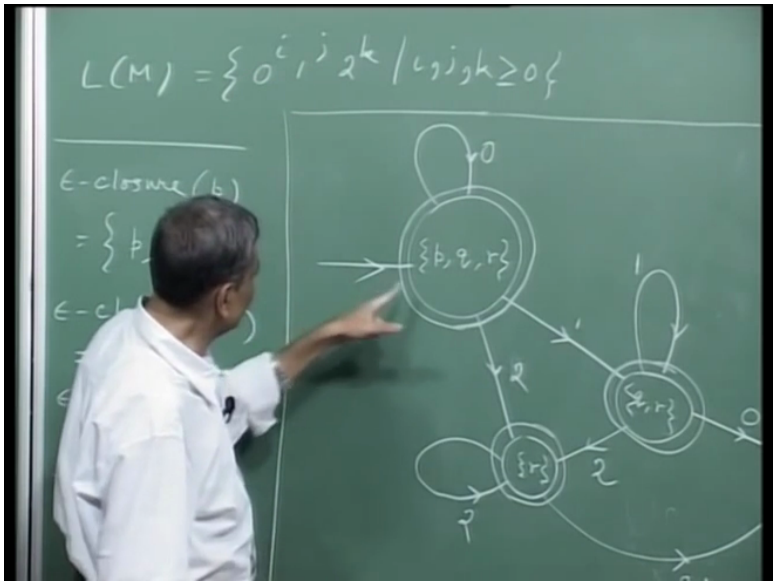


So what should be the final states as before in case of NFA construction NFA to DFA construction? All those subsets in which has at least one final state. Now you see all these 3 states P, q and r were final states, so therefore wherever P, q and r anyone of these occur, which is this, as well as this, as well as this all these 3 states are final and of course there is this is not a final state.

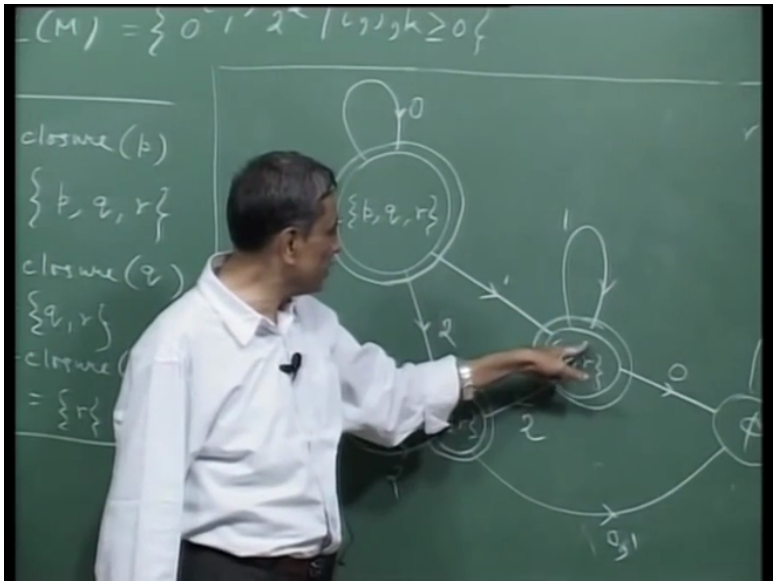
Now by the way this is a DFA clearly because from every state there is exactly on every symbol 0, 1 or 2 there is exactly one transition and it is the construction was similar from NFA to DFA only thing here we took care of Epsilon transitions by taking care of Epsilon closure and now let us just understand, step back and understand whether this machine indeed accepts this language?

You see you start from this, now this is a DFA we start here and now any number of 0's are there, 0 or more number of 0's, 1 is here and if that j and k are 0, of course there is nothing else then it is in this state and that is an accepting state and therefore that string will be accepted.

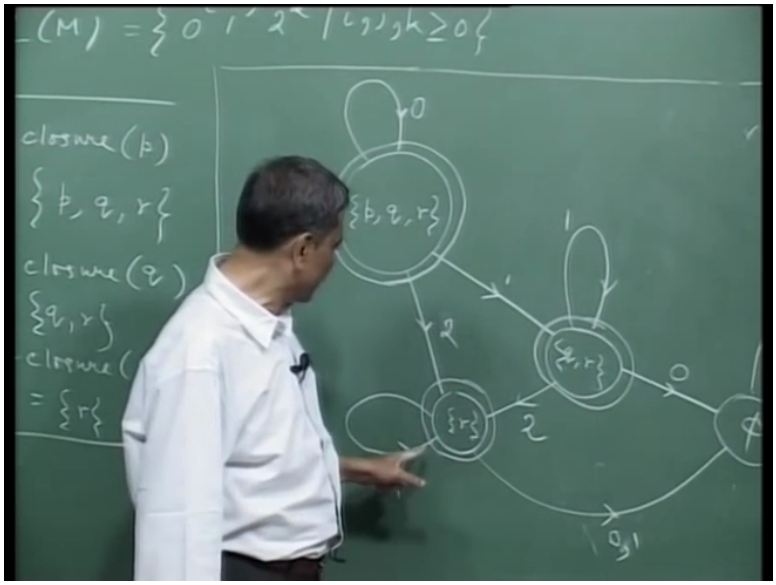
(Refer Slide Time: 47:03)



(Refer Slide Time: 47:09)

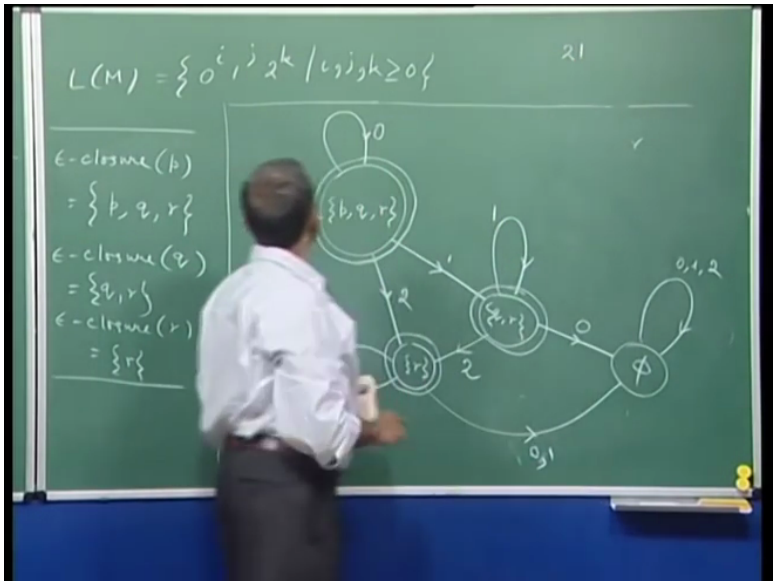


(Refer Slide Time: 47:13)

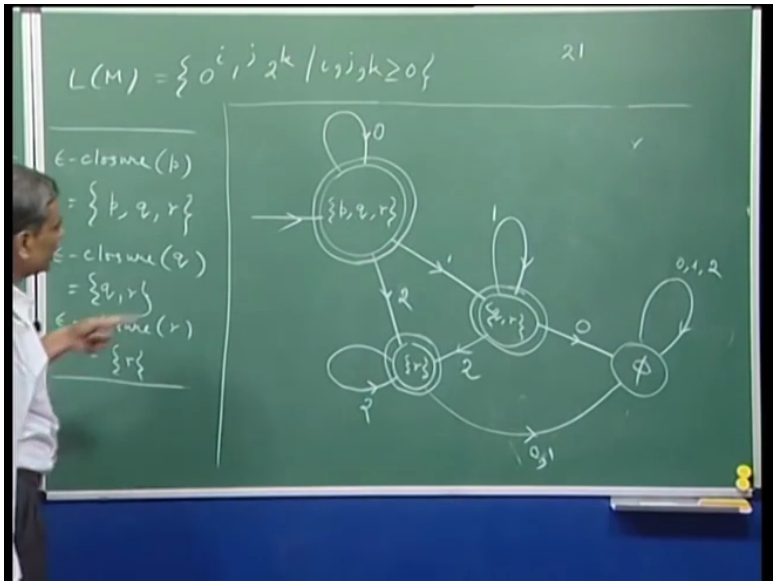


On the other hand let us say the string was some 1's, no 0's, some 1's and 2's, so the machine will start here, some 1's came, so it will be here and then on 1's it will be here and then some 2's came it here and it will be here and then finally some 1's followed by some 2's on such strings the machine will be in this state which is an accepting state and that is all right.

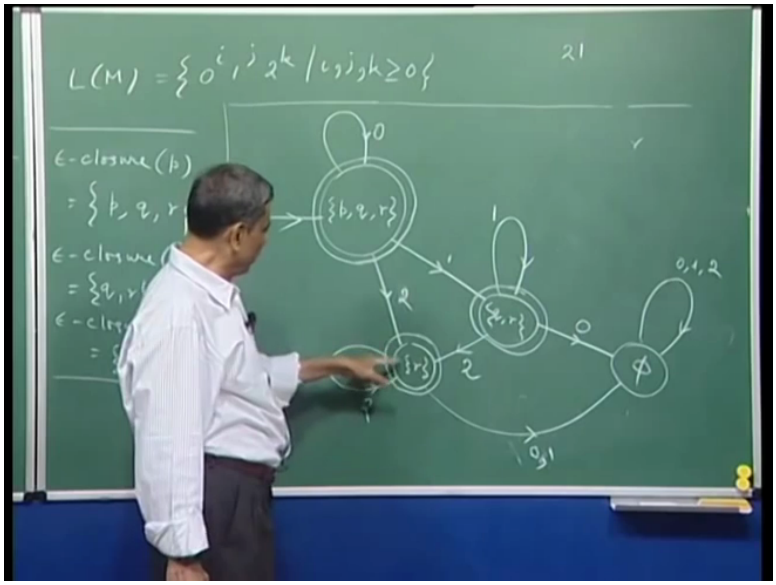
(Refer Slide Time: 47:35)



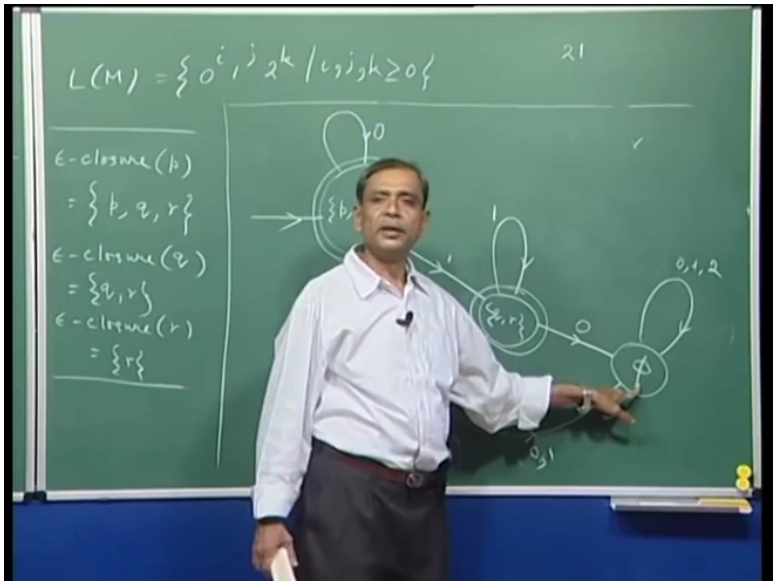
(Refer Slide Time: 47:51)



(Refer Slide Time: 47:53)

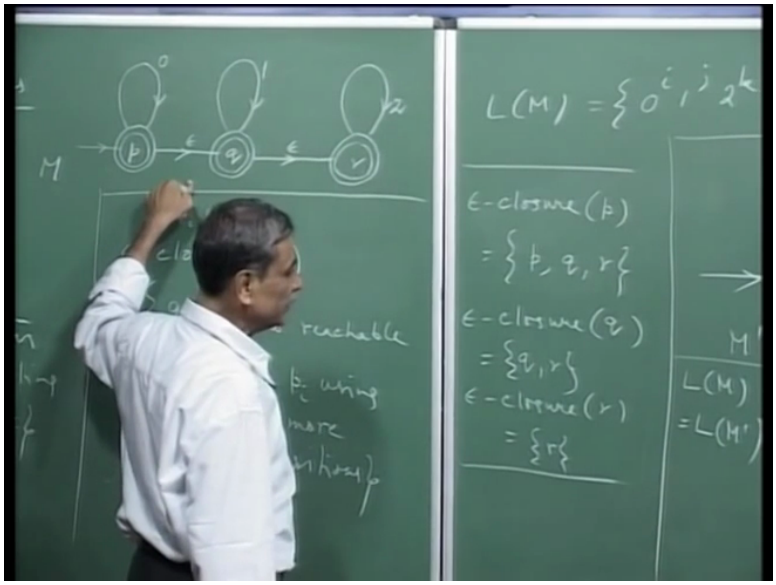


(Refer Slide Time: 47:58)



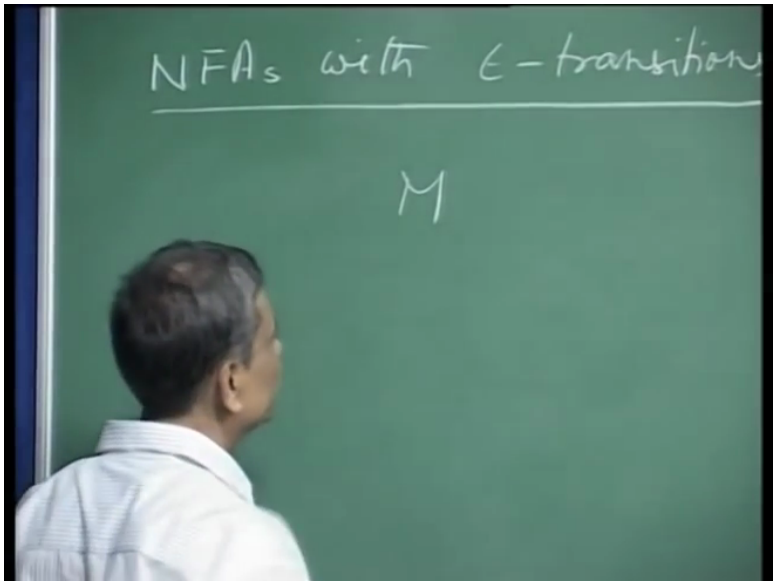
But on the other hand let us take 2 1, this just the string 2 1 and clearly this string according to this definition should not be accepted by this NFA that we kind of once argued and what about this DFA? What I am claiming to be the equivalent DFA, so initially you are here, now 2 came, 1 is here and now 1 came, on 1 you go here which is of course not an accepting state. So therefore the string 2 1 will not be accepted.

(Refer Slide Time: 48:28)



And this machine M dash which is DFA I claim using or understanding so far that we have they will this machine M dash and this Epsilon machine the NFA with Epsilon transition they both these machines accept the same language.

(Refer Slide Time: 48:42)

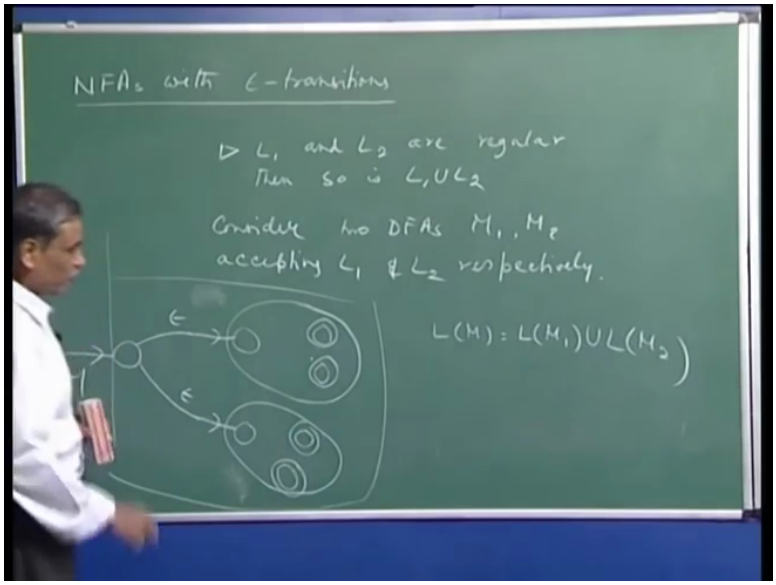


I have tried to show through an example though that every NFA with Epsilon transition there is DFA which accepts the same language and the construction that we have seen similar to NFA to DFA construction is only we took care of Epsilon transitions and the heart of the matter in a way is the Epsilon closure notion and the details that the proof the formal proof of this idea that the construction is correct can be seen in any textbook of theory of computation.

But what I would like to now is to quickly show you of why such machines can be of use? Because the reason for that is, the reason for asking this question is that we know that any NFA with Epsilon transitions there is a corresponding DFA to accept the same language as that of that machine, so what is good with such NFA's with Epsilon transitions? And you see the reason is same as the case with NFA's.

What we saw with NFA's was that yes NFA's do not accept any language which is not regular that is which cannot be accepted by a DFA and yet NFA is an interesting concept because for certain languages it was very easy, much easier to construct an NFA to accept than DFA's. Similar is the case with NFA's with Epsilon transitions.

(Refer Slide Time: 50:49)



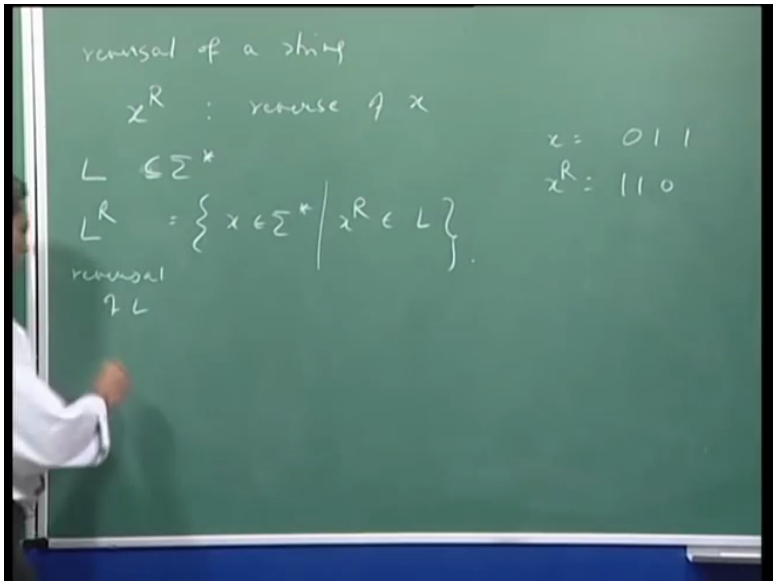
So let me give you one or 2 examples. Supposing I would like to show that L1 and L2 are regular then so is that is L1 union L2, what I mean is L1 and L2 are regular then L1 union L2 is also regular we actually prove this using a few lectures back actually we gave a proof for this I consider 2 DFA's M1, M2 accepting L1 and L2 respectively then another DFA we constructed which as it state we space had the Cartesian product of the 2 state spaces, M1 and M2 state spaces but you see using now this notion of Epsilon transition it becomes so simple this proof.

So we will have this, so I have these 2 DFA's, so this is M1 and I had another machine for L2 and in some number of final states and consider this new so this is M2. Now just consider another machine which is obtained like this that I add a new state and from this state I have Epsilon transitions to the 2 initial states of the 2 machines M1 and M2.

Now it is quite easy to see that this M, this is now one machine, right? This machine M will accept, right? Because supposing a string is in L, LM1 union LM2. Supposing in particular it is in this L1 so it will be accepted by this machine, on that string the machine chooses to take this Epsilon non-deterministically chooses from this to come to this state and then you know it is like a DFA here, so it goes to one of the accepting states.

So for the combined machine all these are accepting states, the accepting states of M1 union the accepting states of M2, right?

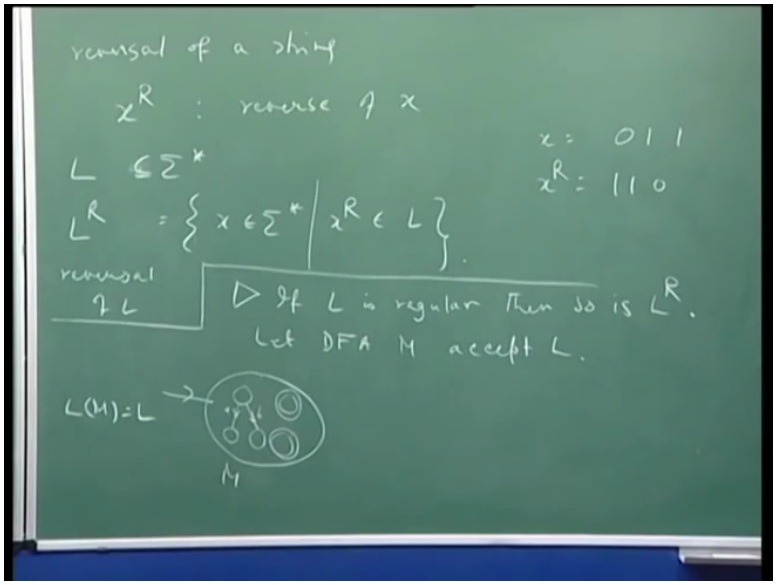
(Refer Slide Time: 54:11)



Now let us take another example which is this a little more involved than this vertical example, for that I need to use the notion of reversal of a string and that idea is very simple. So reversal of a string if you give me a string x then I use xR to denote reverse of x. So what do I mean by this? So supposing x was 0 1 1, supposing this is x, x reverse is of course you write the string the other way around, so 1 1 0.

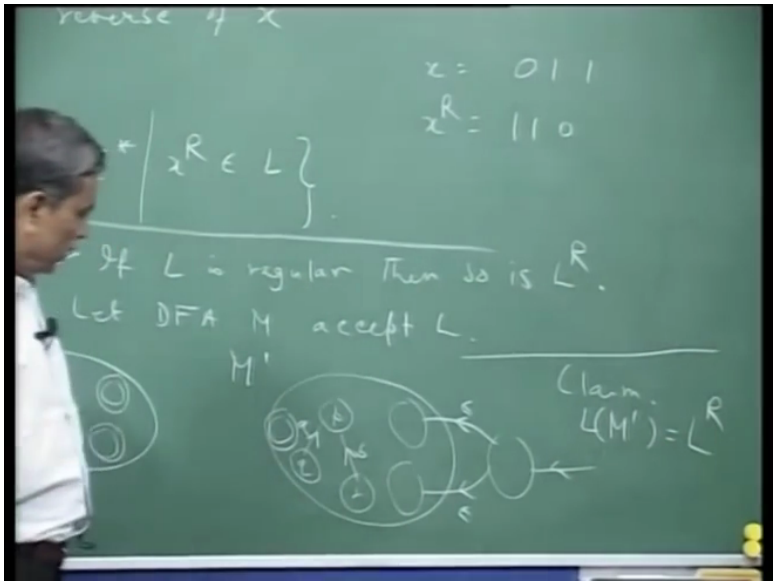
So using this idea of course you can talk of reversal of a language. So L is a language, so L is a subset of Sigma star. What is LR? This is the reversal of L, it is the set of all strings x such that x in Sigma star, xR is in L.

(Refer Slide Time: 55:36)



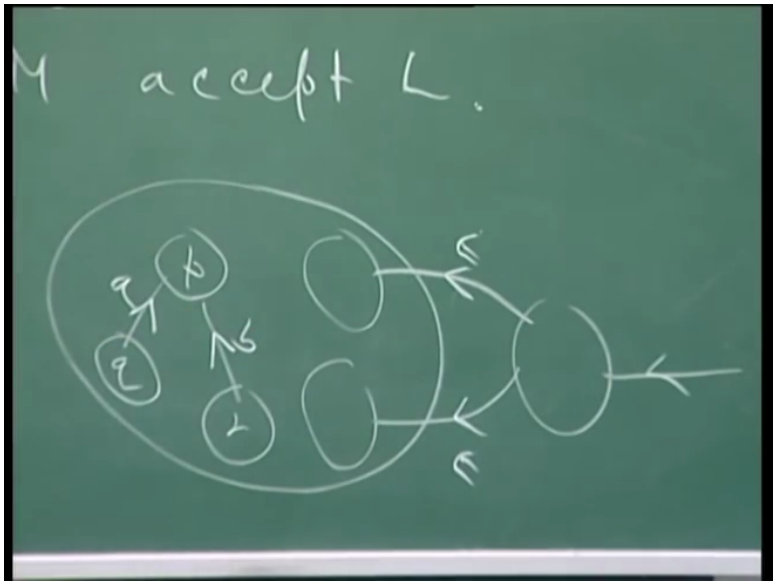
And the idea of the use of Epsilon is from this result that if L is regular then so is LR. So let me give you the construction and then one can argue why it is so? L is regular, so let DFA M accept L, so in picture I have this DFA and it has number of accepting states, this is M and essentially LM is L and supposing there is on symbol a it goes here, on symbol b it comes here, right?

(Refer Slide Time: 56:51)



Now consider I have a machine which is like this, we take the same machine what we do is from the old accepting states of this machine I do not Mark them as accepting now but I have a new initial state and I have Epsilon transitions to these and this was the situation supposing this is P, this is q, this is r, P, q, r, what we do is in this machine now we reverse all the arrows. So there was an arrow which means there was a transition from P on a it could go to the DFA went to q.

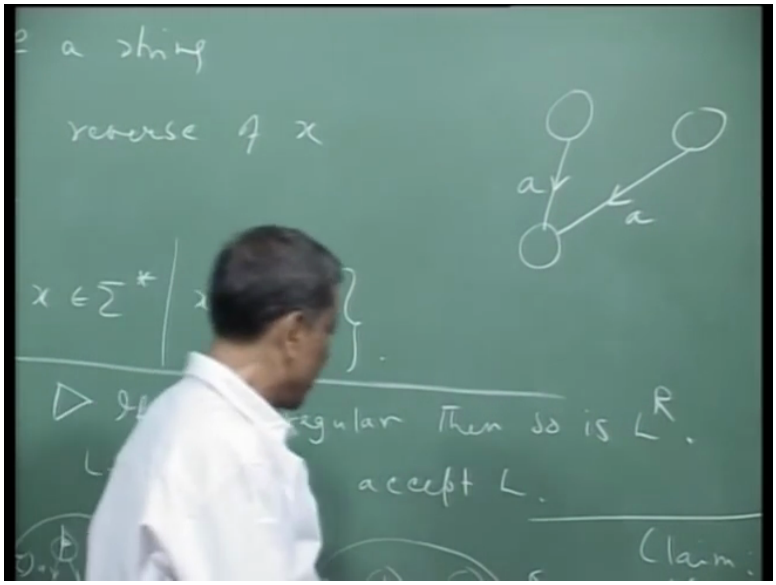
(Refer Slide Time: 57:57)



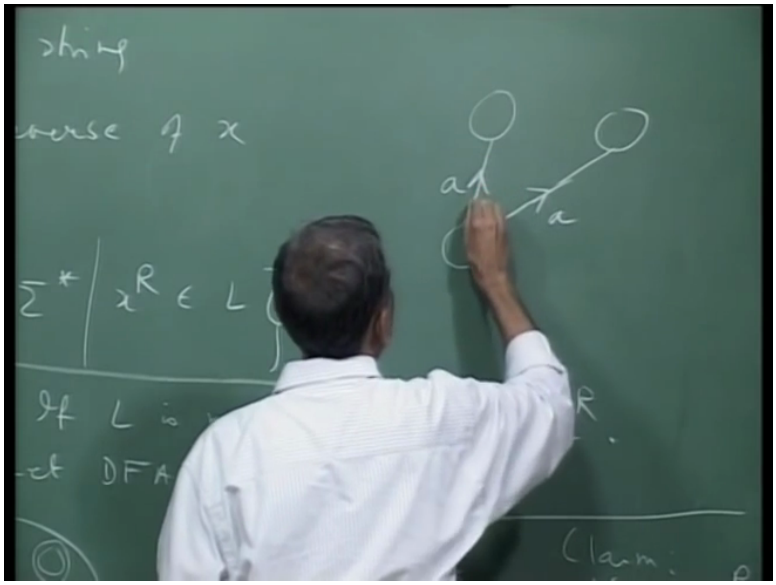
Now I reverse the arrow and this had though initial state here that is the one which I make final states, this construction is very simple I am going from machine M, I am first of all I reverse all the arrows in the transition diagram and Mark the old initial state is the only final state and then I add a new state I have Epsilon transitions to the old final states and now I claim that this machine M dash, language accepted by M dash is LR.

First of all let us understand why this is an NFA, of course 2 reasons one is that there are Epsilon transitions, so non-deterministically from the initial state this machine can either go here for here but non-determinism also comes because we have reversed arrows.

(Refer Slide Time: 59:27)



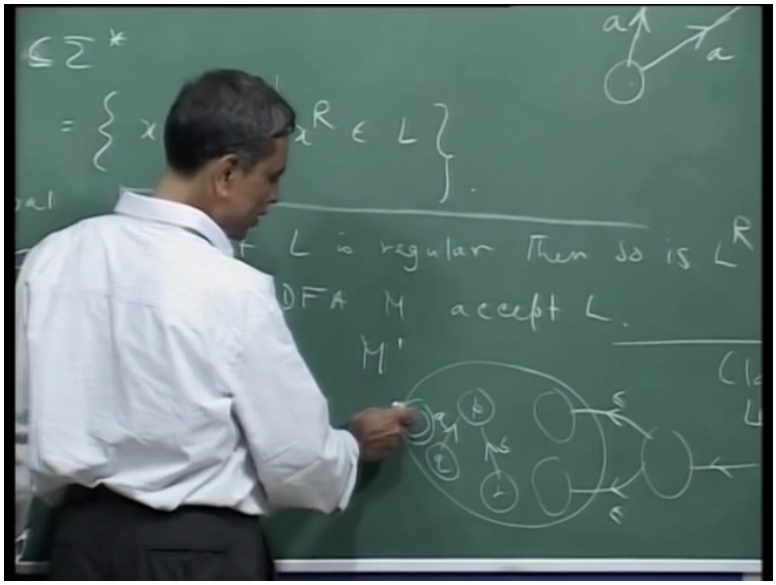
(Refer Slide Time: 59:52)



Consider for example that it is possible in a DFA that on a you came here as well as on a you came here when you reverse the arrows as we did here the direction of the arrows, so this is now going this way and this is going this way, you see that on reversal of the arrow directions from this state on a, one could either go here or could go there, there are 2 transitions defined.

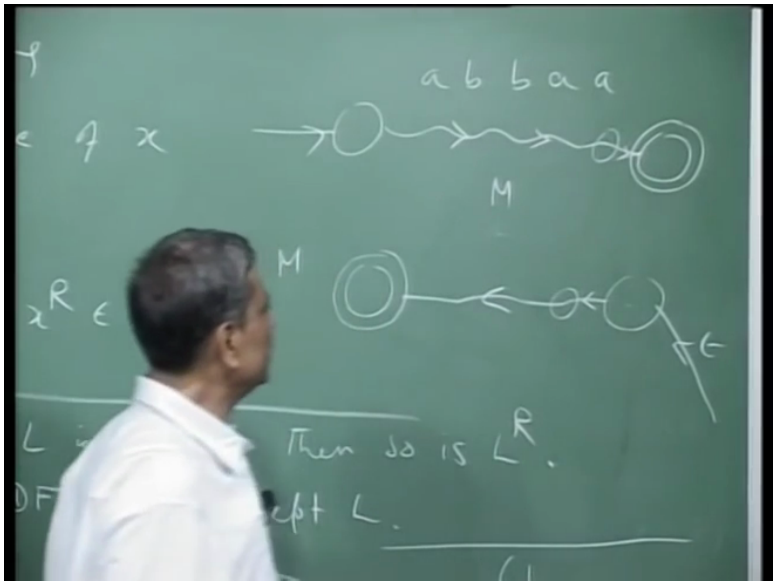
And similarly you will be able to make an example where from a state on a symbol, nothing no transition is defined because of the reversal of the arrows and very quickly why if x is in the language why xR will be accepted? And the reason is this because x is accepted there was a path in the DFA from the initial to one of the final states, right? And what is xR? It is just the string in the reverse manner.

(Refer Slide Time: 61:06)



So now the last symbol will be the first symbol, so whatever the final state x had gone to this machine, this new machine will come to that state on Epsilon transition it can I mean it can choose to do so and then it will follow the arrows in the reverse manner to come here and this is the accepting state.

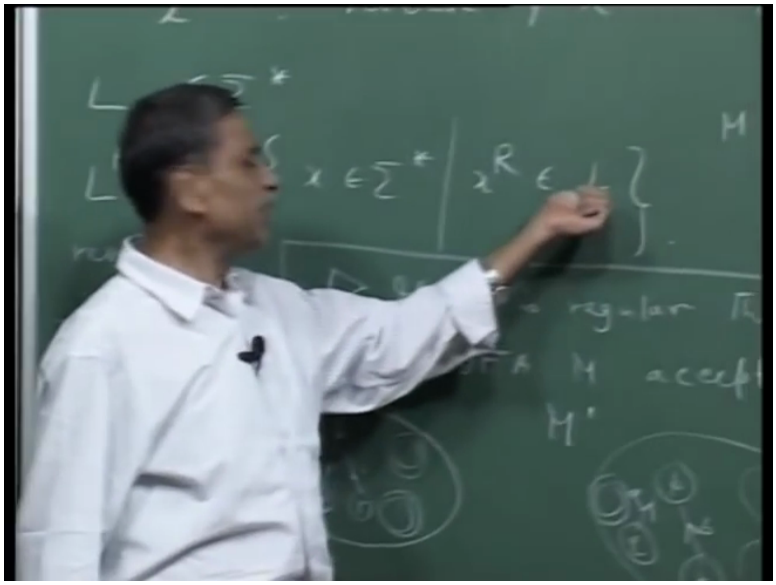
(Refer Slide Time: 61:26)



So in other words once more that suppose there was a string x which took the machine from the initial state of M to one of the final states, let me let me draw it here the x had taken to this final state this is in M, right? So let us think of a b b a a some string like this and now this, what happens here?

The reverse machine M dash, there is a path from this state to this state on the same symbols but going the other way a a b b a because a took you from here some state here to here, right? Supposing this was the last state and those transition here but now the arrows are reversed, so this last a can take the machine from here to here and here to here, it is easy to see that this particular path, there is a path from this machine from the state to this one.

(Refer Slide Time: 62:53)



But how did the machine go here? Non-deterministically from the initials, so it is not difficult to see that the way we have defined this new machine will accept precisely all those strings whose reversal is a string in the language.