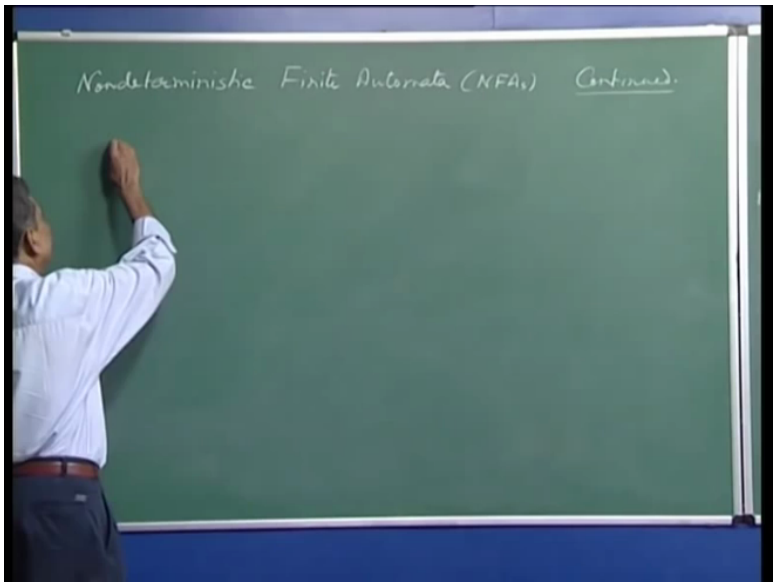
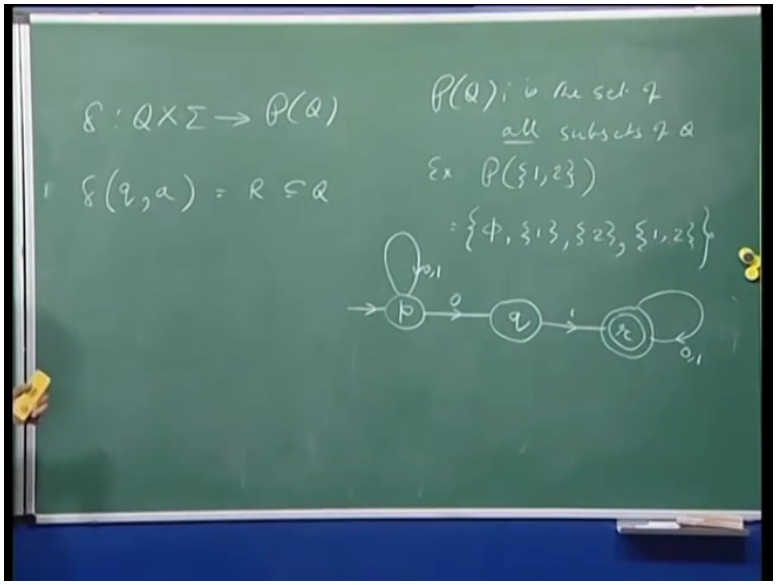
Theory of Computation
Prof. Somenath Biswas
Department of Computer Science and Engineering
Indian Institute of Technology Kanpur
Lecture 8
Formal description of NFA,
Language accepted by NFA,
Such languages are also regular.

(Refer Slide Time: 0:27)



So far we have discussed NFA's at an informal level, let us define NFA's formally, an NFA when we use the formal route is a 5 tuple Q, Sigma, Delta, q0, F as DFA's are where Q is the finite set of states of the machine M. In other words an NFA also has number of states and that number is finite, Sigma is the alphabet, right? And which is a finite set of symbols, Delta is the transition function and it is here, an NFA and DFA would differ in general q0 is the initial state of the machine M and F is the set of final part accepting states. We have 2 terminologies for the same thing either you can say final states or accepting states. Clearly this is a subset of Q.

(Refer Slide Time: 2:34)



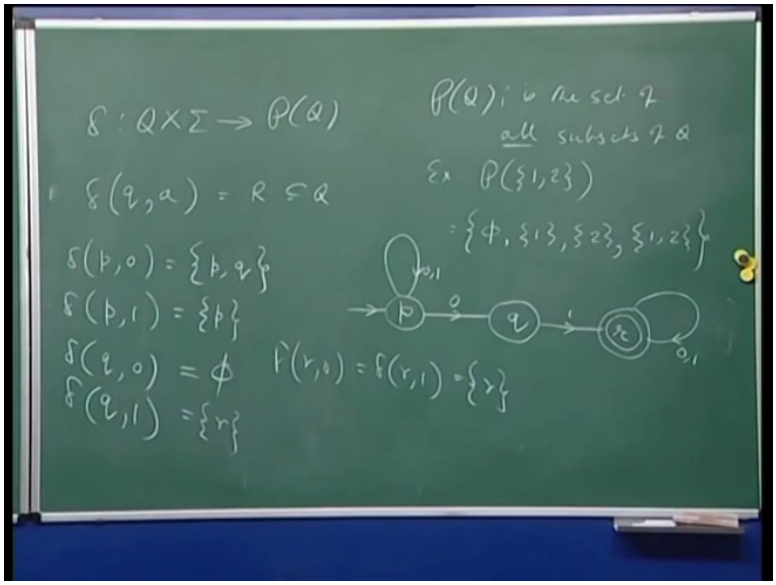
And now let me describe the transition function for NFA's, so Delta is a mapping from Q cross Sigma to power set of Q. Now you recall what power set is, power set of Q is the set of all subsets of Q, example would be, simple example let us see you have power set of the set 1 and 2 this is going to be, this is a set remember power set is a set and its elements are the subsets of the set here.

So the empty set is a subset of this set 1, 2, the singleton 1, the singleton 2 and the entire set itself. So in general if Q had N elements, Q had N states then power set of Q will have 2 to the power N elements each of which is a subset of the set Q. So now this is saying the Delta is a map which has 2 arguments that is the domain is the cross product of Q cross Q and Sigma and its range is this power set.

So for example Delta of let us say q and a symbol a, so the first argument of Delta comes from the set capital Q, the second argument comes from the set Sigma, so this is a state of the NFA, this is the symbol of Sigma and this will be some R which is a subset of Q. So let us say how it captures our informal understanding of NFA's.

Let us take a very simple example and let me name these states as, this is P, this is q and this is r and you can see what this NFA accepts, the language NFA accepts, it accepts the set of all strings in which the string 0 1 occurs as a substring we have seen similar examples before.

(Refer Slide Time: 6:05)



But in this picture the transition diagram formally would write like this Delta P of 0, right? Now which are the states the machine can go to? In this diagram you can see it can either go to P or it can go to q on 0, so therefore this is, so the understanding of this is that if we have Delta q and a is R which is a subset of Q the machine from state q on symbol a can go to any one of these states of M which are in R.

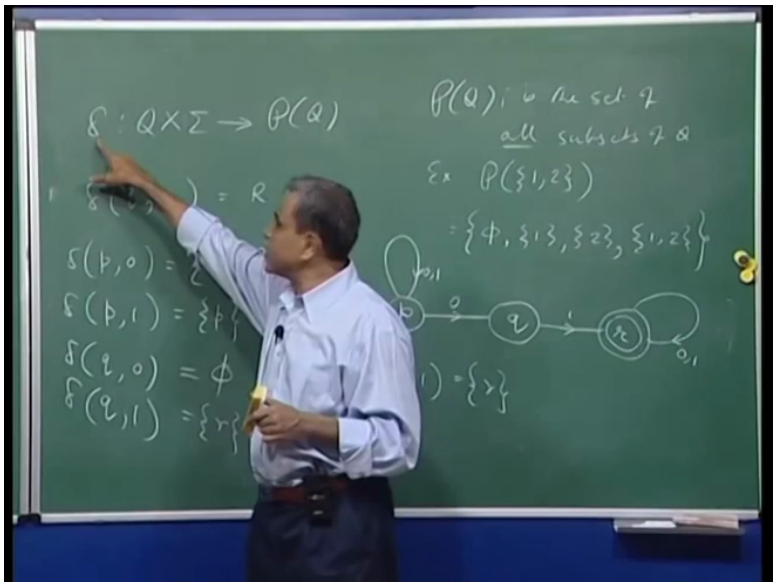
So like here the machine R machine, this particular machine from P on 0 it can go to you see it can go to P on 0, it can also go to q on 0, so the set of all states through which the machine can go to that is the value for this case, right? So similarly what is Delta P of 1? Delta P of 1 you can see it from P on 1 it comes back to P but we write like, since they have to be a subset even for a singleton case this is the set of states it can go to, maybe Singleton.

In fact this set can also be empty, as you can see from the next state q, Delta what is q of 0? See there is no transition defined from q on the symbol 0. So which are the set of states the machine can go to from q on 0, none of the states. So therefore it is the empty set but empty set remember is always member of the power set, so that is all right our definition of Delta allows empty set here, R can be empty as in this case.

And similarly you can write Delta q of 1, Delta q of 1 is only the set consisting of the symbol r and to complete our description for this case Delta r of 0 as well as Delta r of 1 both of this once the machine is in state r it does not matter whatever is the symbol it remains in the state r, so this is r, right.

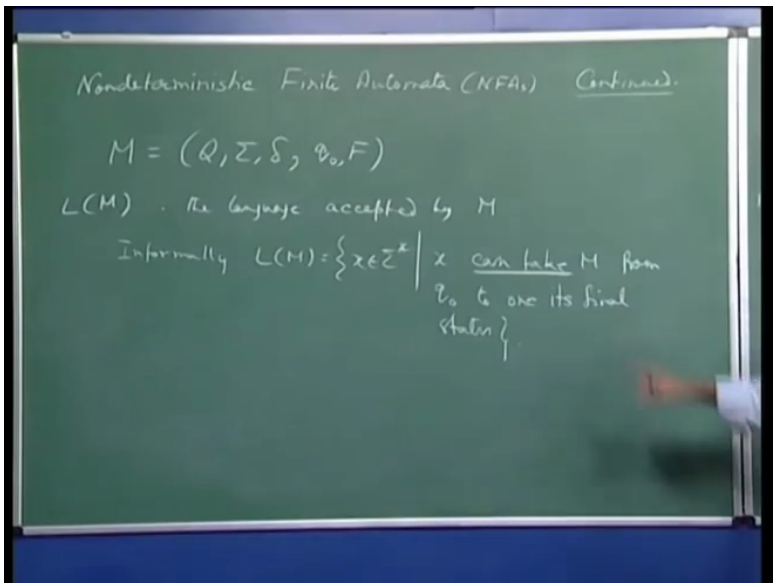
So just to emphasize once more in case of an NFA when we say that Delta q, a from state q on a is equal to R which is a subset of Q, what we mean is, our understanding is that if R, R is the set of all states the machine can go to from the state q on symbol a, right.

(Refer Slide Time: 10:01)



We have defined formerly what NFA's but the next important thing we need to define is what the language accepted by an NFA is? How do we define that formula? In case of DFA what did we do? We extended this Delta which was from Q cross Sigma to Q cross Sigma star, exactly that is what we are also going to do but keeping this definition in mind for Delta in case of NFA.

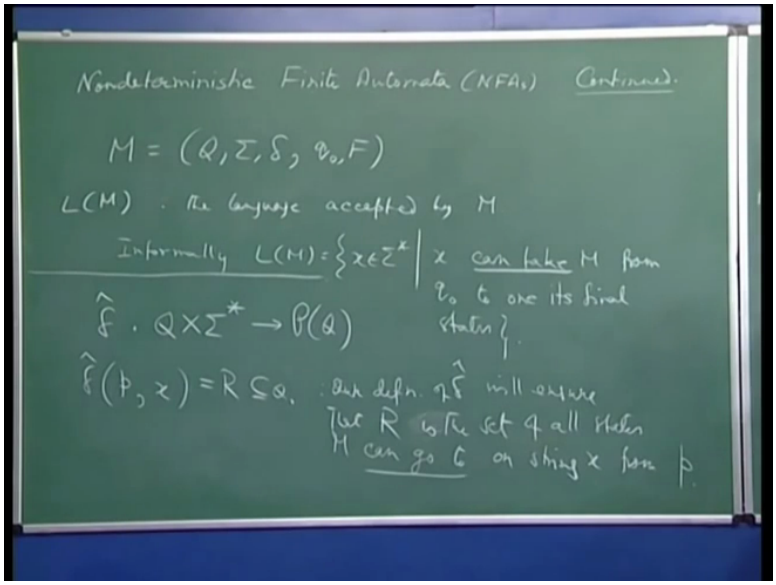
(Refer Slide Time: 10:22)



Now we would define LM the language accepted by an NFA M, informally what did we say the language was? Informally we said LM was the set of all strings in Sigma star such that x can take M from q0 to one of its final states, right? Remember I emphasize that because it is a nondeterministic machine a string can give rise to many different computational paths and depending on the nondeterministic choices made by the machine that particular computation it will be either in this state or in that state.

However this thing is well-defined we look at all possible state's the machine can go to after reading the the string x and that is one of its states it can go to happens to be one or more, happens to be a final state then we say that the string x is acceptable. So this is of course we have stated it informally, how do we formalise it?

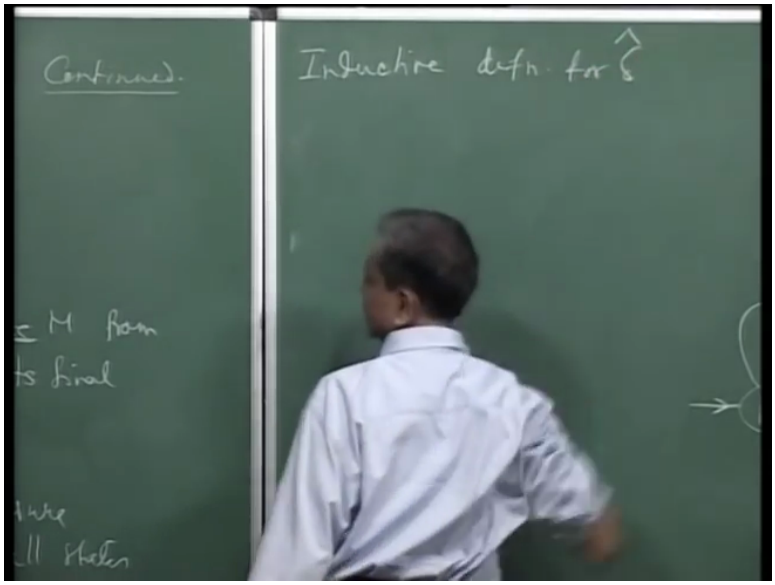
(Refer Slide Time: 12:12)



As I said we will do it in terms of Delta hat, so let us remember what Delta hat was? Delta hat is a function which again takes 2 arguments but the second argument, first argument is a state and second argument is a string, right? Is a finite string over the alphabet Sigma, right? And its range is as before the power set of Q. So intent is, intent is this we would like to define Delta hat this manner that suppose Delta hat of P and x.

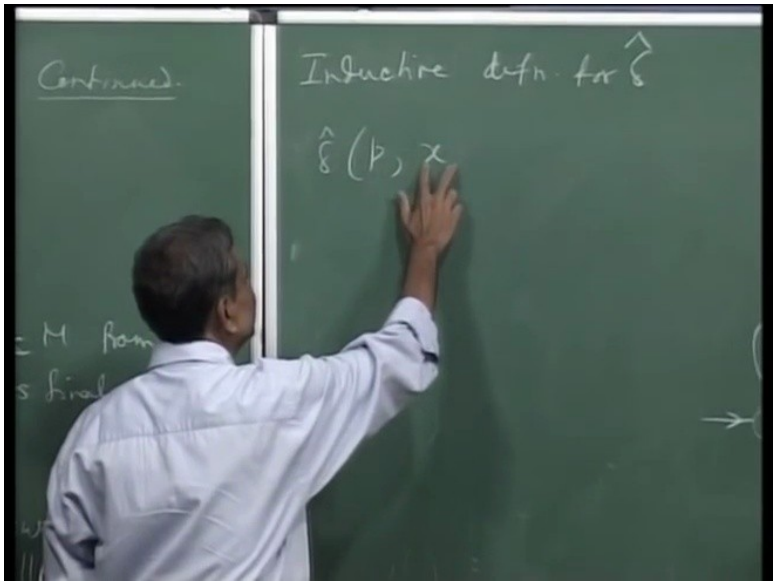
Now x is a string remember which is in Sigma star, if this happens to be R which is of course a subset of Q then we would like our definition should ensure definition of Delta hat will ensure that this capital R is the set of all states, the R is the set of all states M can go to can go to I will again emphasize the fact its backdrop possibility it can go here it can go there and which are all the states it can go to? That is what, is the subset of QR on string x starting from state P, right?

(Refer Slide Time: 14:37)



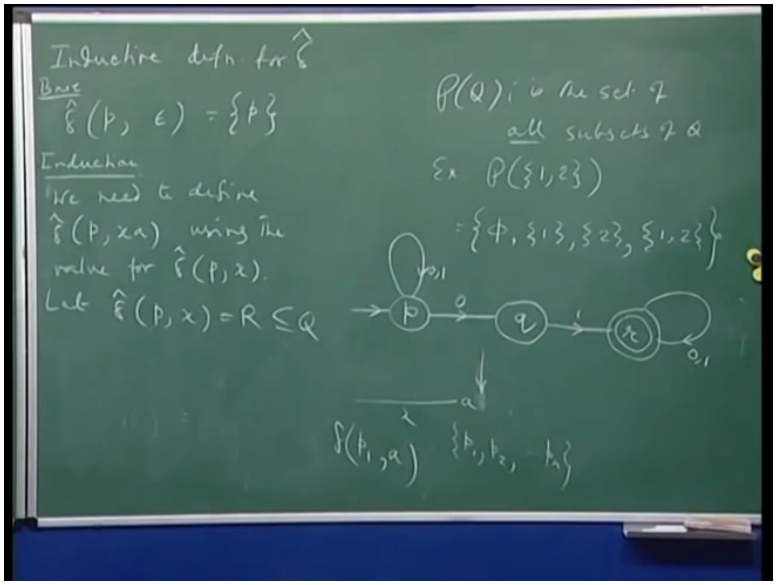
This is what we would like to ensure and how do we do that? Well, we will do it in this manner inductively we will define Delta hat using the definition of Delta and remember the style is same as the way we did it for DFA, this is what the informally what we would like Delta hat to satisfy, so what is Delta hat? Remember Delta hat takes 2 arguments P and some string x.

(Refer Slide Time: 15:28)



Now I need to define this function Delta hat for all possible second arguments, argument here but then that is an infinite set we cannot just write down as we could do it for Delta. So inductively you will define, inductively this induction is on x.

(Refer Slide Time: 15:38)

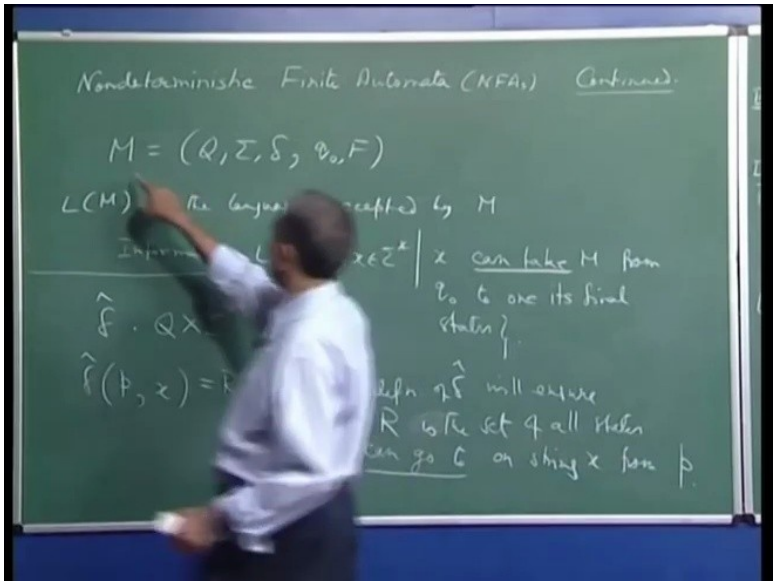


So what is the base case? The base case is when the string is empty the null string. So P and it is presented with the null string Epsilon what happens, what we want to ensure? That the set of all states the machine can go to from P on this string but then there are no symbols it has seen, so it will not make any transition it cannot make any transition it will remain in the state P, so clearly this is going to be this, okay.

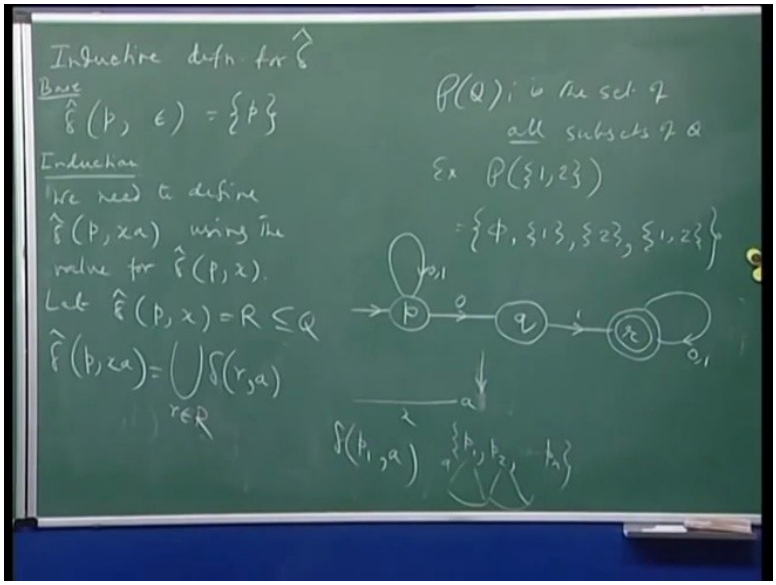
So this is the base case and for the induction part what do I have to do? We need to define Delta hat P, x a using the value for Delta hat P x, right? This is how all the inductive definition goes that supposing you already know how to do it for the string X then a new symbol comes a which is added to its right then for the string x say what will be the value of Delta hat?

So let Delta hat of p of x be R which is of course the subset of Q, remember that means this is the set of all states the machine can go to from inductively we assume that this assertion will hold here from the state P on the string x, right? So just before I write down formally what is happening, so let us say this part is x and this part is a, right? And after reading x the machine is in some states let us say it can be in one of these states.

(Refer Slide Time: 18:34)



(Refer Slide Time: 18:30)

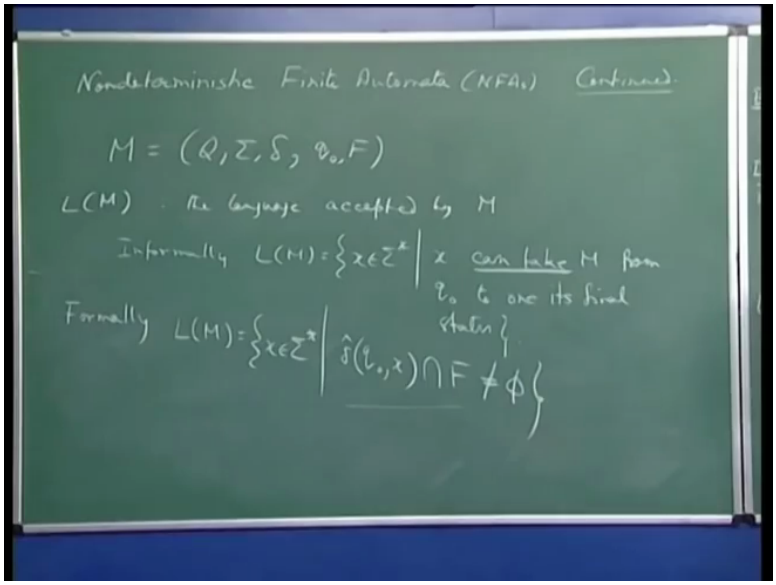


Now what other states the machine can be after reading a at this point. So it on x it could have reached the state P1 and now it is reading the symbol a, so had it reached P1 then it would have gone to one of these states P1, a this is defined because Delta we already know when you gave me the machine M, Delta P 1 of a tells me which all states the machine can be on reading a from the state P1. So this is going to be a collection of states, so these are the states the machine can be after reading a.

Similarly from P2 on x it could have gone to P2 and now a came again there will be a bunch of states defined the machine which the machine can go to any one of which the machine can go to on symbol a and so on, so therefore it's not difficult to see the set of states the machine can be after reading this symbol a is the union of all these states, right? So therefore we would write like this, this way we will that Delta hat of P of xa is the union of, their supposing r is a state, r is an element which is of course the state in capital R.

Remember what was R, capital R is the set of states the machine can be starting from the state P on reading x and now the next symbol we need to take care say Delta r, a this is going to give me a subset of states, so had the machine reached r then when it read a the machine would have gone to, could have gone to anyone of these states and the same is true for every other element of R, so we take the union so you see this completely defines Delta hat, okay.

(Refer Slide Time: 20:45)



And in terms of Delta hat we can now formally define the language accepted by the machine M this was the informal definition and formally now that we have all this machinery we don't need to write English sentences or in any national language sentences to explain what the language accepted is LM is the set of all strings in Sigma star such that Delta hat of q0 x, what is this? This is the set of all states the machine can be starting from q0 on this x on reading the string x and what we want?

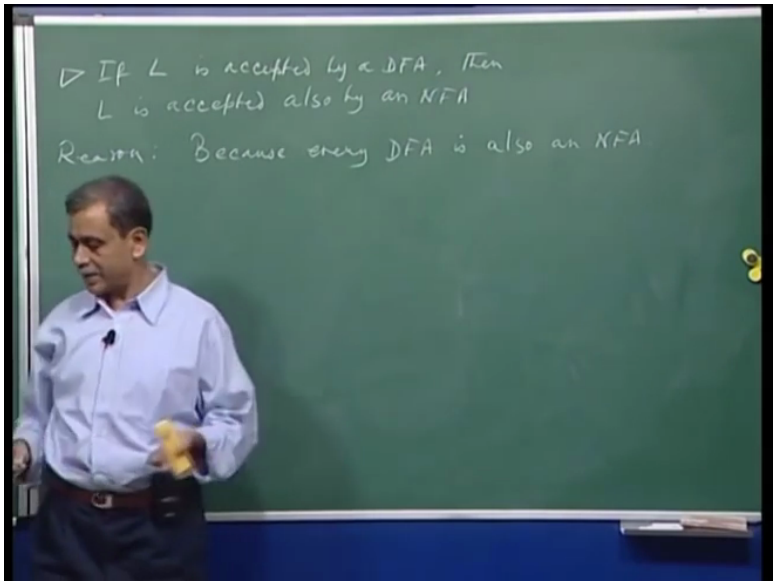
We would say that this string x is in the language LM if anyone of these states happens to be one of the final states. So that is easy to write formally is this, what we are saying is look at the collection of states the machine can be on reading x starting from the initial state intersect that set with the set of final states, if that overall thing that you get is nonempty that means it can go to one of the final states then of course x is in the language and these are precisely the strings which are in the language because we are stating this is equal.

So this is the way to define the language accepted by the nondeterministic machine M. Now this one very important question that we need to address and that is we had defined DFA's and we had then talked of the languages accepted the class of languages accepted by DFA's and now we have defined NFA's and now again we can talk of the class of languages defined by NFA's.

So I have 2 classes of languages, one define through DFA's the other defined through NFA's, what is the relationship between these 2 classes? In particular it is easy to see that if a

language is accepted by a DFA then it is also a language accepted by (()) (23:38). So let me let me write it down.

(Refer Slide Time: 23:45)

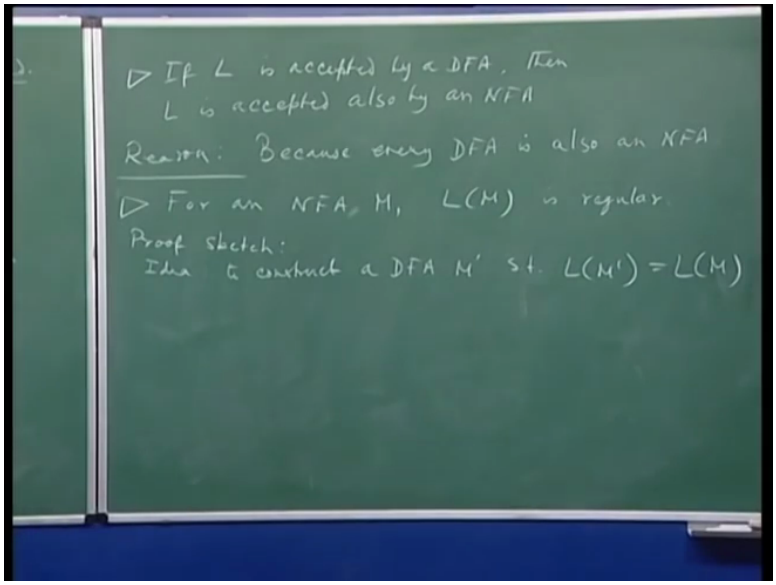


If we stated, if L is accepted by a DFA that means there is a DFA to accept L then L is accepted also by an NFA. Now the reason for this is actually a trivial reason, why? Because every DFA is also an NFA, does it make sense, this statement? It does because what is an NFA? NFA just states that from a state on a symbol that can be 0 1 or more transitions, right? Where is DFA's from a state on a particular symbol there exactly one transition, so therefore the DFA case is a sub case.

DFA situation is subsumed by the situation of NFA, so you can think of every DFA without any problem that it is an NFA, it is that NFA is choosing to behave deterministically that's all. So in terms of class of languages therefore it is clear that the class of languages accepted by NFA's can be something bigger it cannot be less, right.

So the question therefore which comes to our mind is that is there a language which can be accepted by an NFA but not by any DFA and the answer is no. In other words these 2 classes, the classes of the class of languages accepted by DFA on one hand, class of languages by NFA's on the other hand these 2 classes are actually same and therefore since the class of languages accepted by DFA's we termed it as the regular languages, class of regular languages NFA's also precisely accepts regular language.

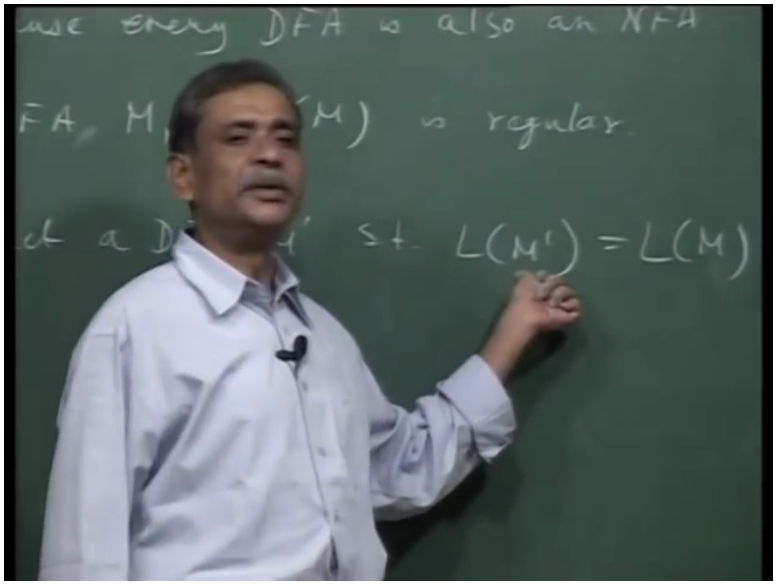
(Refer Slide Time: 26:37)



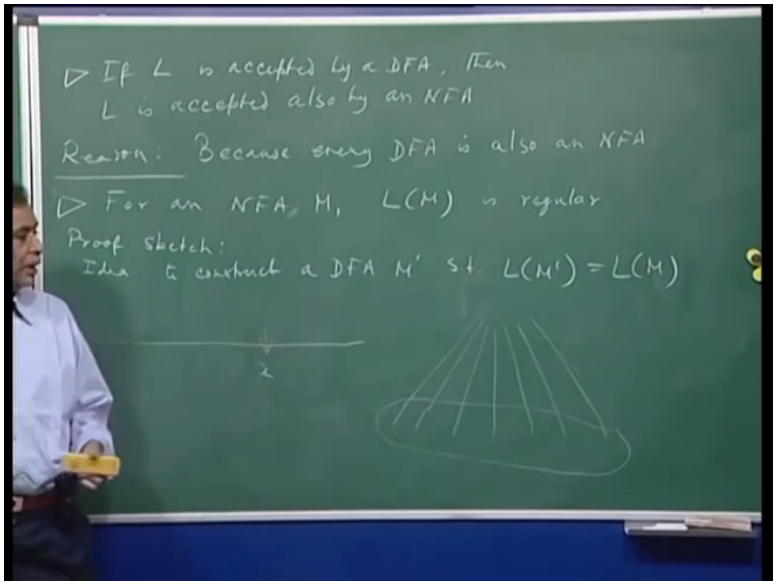
So this requires of course proof, so what we are claiming is that if you give me an NFA, for an NFA M the language accepted by M, I will put it this way is regular. When I say LM is regular then what I mean is because by definition there is a DFA to accept this language LM, the language accepted by the NFA M and we prove this by actually constructing given any NFA M we will see how to construct a DFA from that M, the definition of that M which accepts the same language.

So let me now give a proof sketch, right? So the idea is to construct a DFA and let me call it M dash such that the language accepted by this DFA is same as the language accepted by the NFA that you had given me, right?

(Refer Slide Time: 28:24)



(Refer Slide Time: 28:44)



So before we proceed further let us see what can be the strategy of the DFA M dash, right? So we have seen several examples for the design of DFA's and remember this is a string and our M dash has to decide by the time it reaches here the end of the string that whether or not x would have been accepted by the NFA. So as it is going on what all information it can keep through its finite head that will help in doing this task, right?

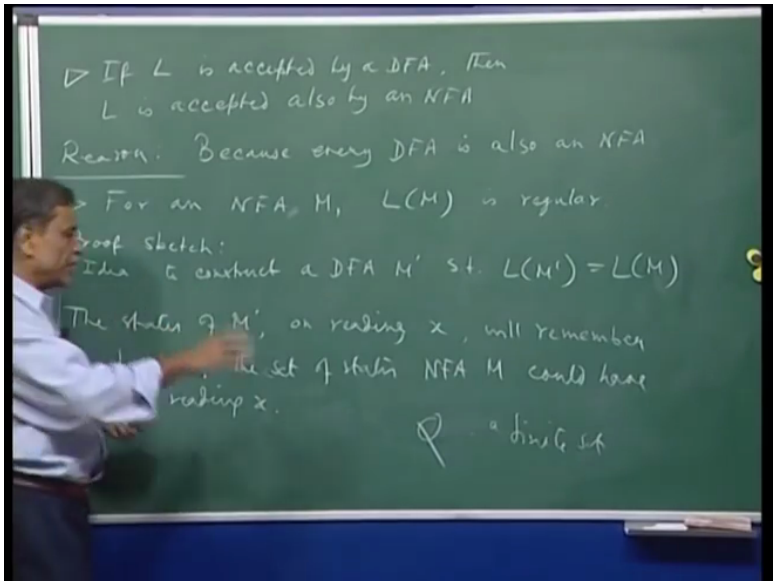
So in particular what should be the states of the DFA such that as the DFA moves from symbol to symbol when it reaches the end of the string it will know whether or not the string is in the language of the NFA? Now let me tell you is an idea which won't work which is this, that can we do this that of course we know that on a string x there are many possible computation paths.

So one could be that let the DFA keep track of all these paths, right? But that idea won't work, why? Because the total amount of information if you would like the total number of states that would be required will be infinite because as we have longer and longer strings we will have it's like a you remember that computation tree is like a tree and its keeps growing it keeps growing and we can't keep the entire tree for every input string in a state.

In other words because a DFA can work only with finite number of strings then you would say okay, is not really necessary to keep track of all these computation paths, why not keep track of the these sequence that is if it had taken this part then it would have been in this state on x, had it taken this part? So this will be a sequence of states, can we keep track of that in the finite states of DFA?

Again the answer is no because this sequence also keeps growing arbitrarily large with larger and larger input but all this is overcame we don't really need, the DFA does not really need to keep all the information which is there either in the entire computation tree or even in the frontier of the tree, what it needs to keep track of?

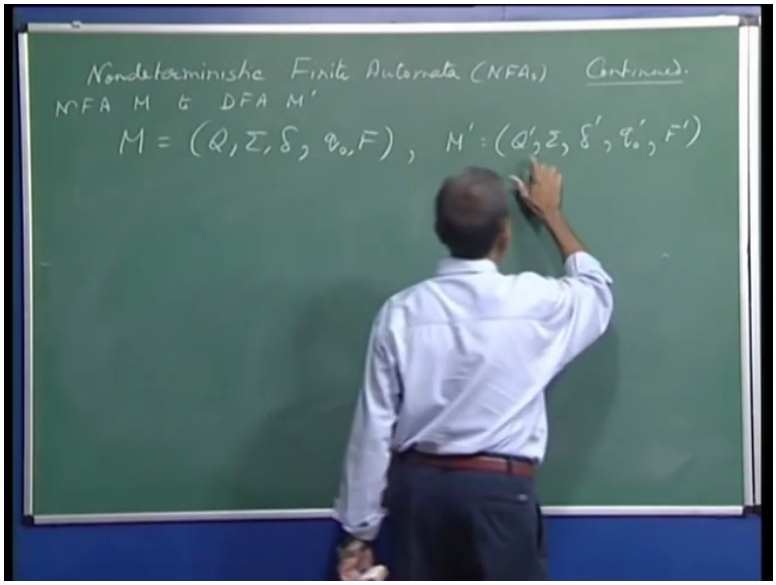
(Refer Slide Time: 32:18)



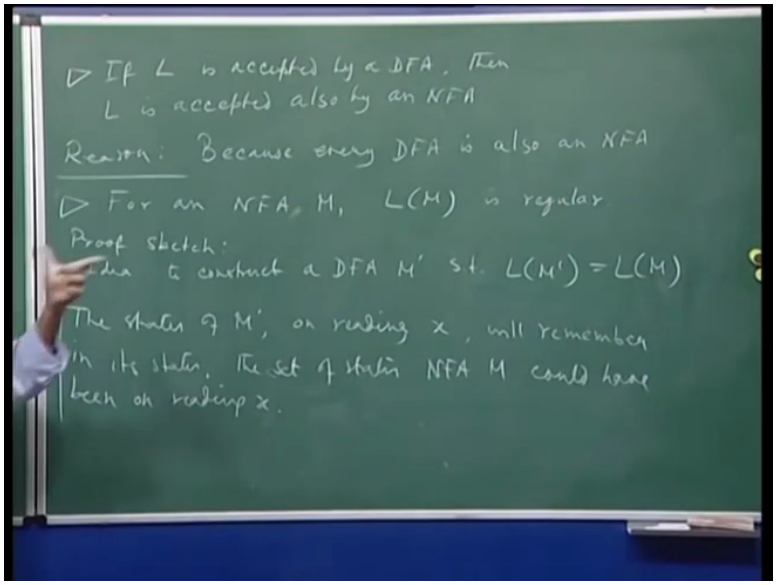
So let me write it down the states of M dash on reading some string x will remember in it's states the set of states NFA M could have been on reading the same string x. Now why should this idea work? So you know this is the invariant the DFA will keep fixed that it will remember in one of its states. Precisely the set of states the NFA would have been (()) (33:29) on reading the string x.

Now you see this actually the total amount of information is finite, why? Because NFA has the states, the set of states of NFA is Q and Q is a finite set and whatever be the string that you give me the NFA will be on reading some string x it will be I know in one of the subsets of Q. So how many different subsets can be there a finite set like Q? Clearly that will also be finite. So this information that the DFA needs to keep track of therefore is a finite amount of information and that does the work.

(Refer Slide Time: 34:46)

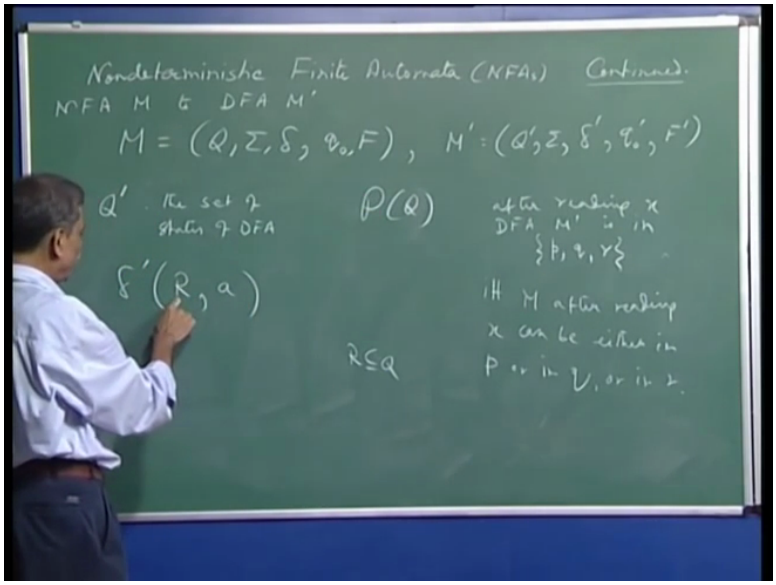


(Refer Slide Time: 35:26)



So let me now give the little more detail on this idea, from the NFA M we would like to get to a DFA M dash which is equivalent in the sense M dash the DFA accepts the same language as that of the NFA. So if our NFA was this Q, Sigma, Delta, q0, F. M dash let me write it as Q dash etc and now our strategy for this DFA M dash we said is going to be that M dash is going to keep track through its you know states, the set of states M could be after reading part of the in during some initial part of reading, right.

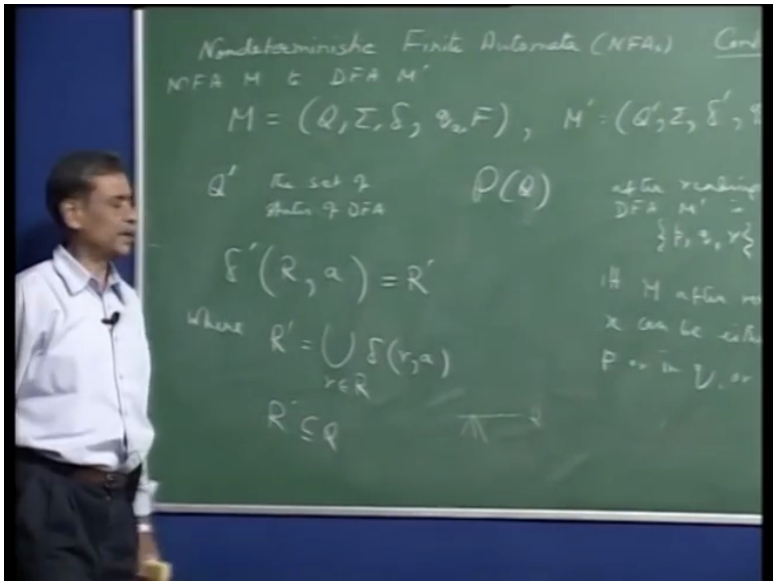
(Refer Slide Time: 35:33)



So then the simplest thing would be to define Q dash, which is remember the set of states of DFA to be the power set of Q, right? So for example if the DFA is in this state where each p, q and r is a state of the NFA after reading x DFA M dash is in, now this is a state of the DFA what we would like to ensure is if and only if M after reading x can be either in P or in q or in r, this is what we would like to ensure.

So for that we are saying the DFA set of states is the set of all subsets of states space of the NFA, alright this is not too difficult to see to implement that idea and the only thing is now what we do about, how do I define the transition function for the DFA? So now Delta dash is going to be, now is R and a, what is R? R is a subset of Q which is the NFA set of states but this is now a state of the way we have defined it.

(Refer Slide Time: 38:03)

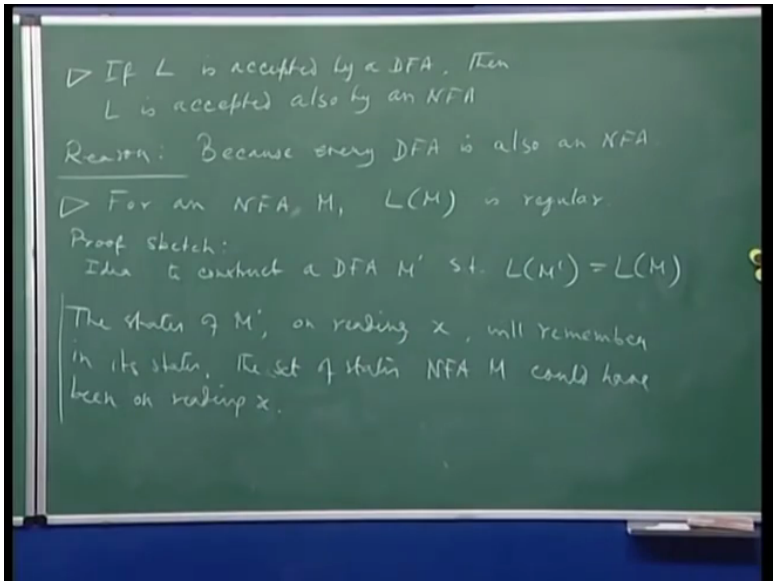


It is now a state of DFA R and a symbol has come. So according to this idea this should be equal to R dash and what is R dash? Where R dash is the union of I mean we have discussed this idea before but it is essentially what we were saying is that suppose the NFA is one of these states R on reading something, some part of a some string and now a came, right?

So from this small r it can go to these are the states the NFA can go to, so I collect these states, collect these all these states take their union, now I get a subset of states of M and let me define that to be the value for Delta dash R a and if you see this actually implements this idea that we have stated here one more thing to notice that there is nothing nondeterministic about it.

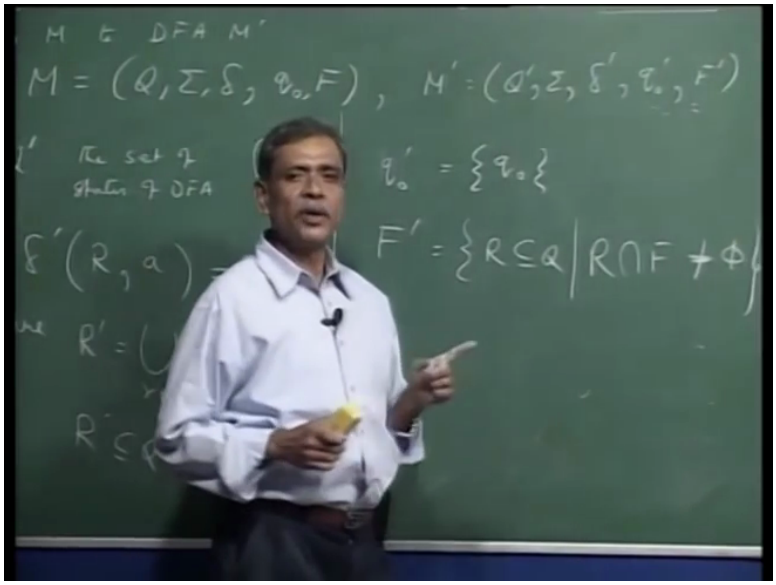
This is the state of DFA, this is another state of the DFA, is not it? Because R dash is also a subset of Q, so there is one state, a symbol and we are giving a definite state of DFA that we have (()) (39:38).

(Refer Slide Time: 39:51)



Now I will not prove it formally but if you see that this idea being implemented through this definition of Delta dash then it is not going to be difficult to see that this M dash is going to accept the same language but of course I need to tell you what these other 2 components of the DFA would be, so that is easy.

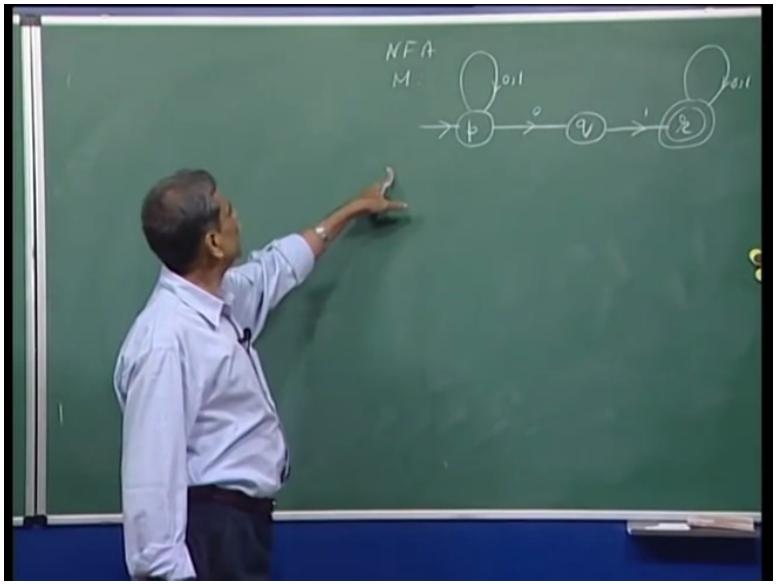
(Refer Slide Time: 40:23)



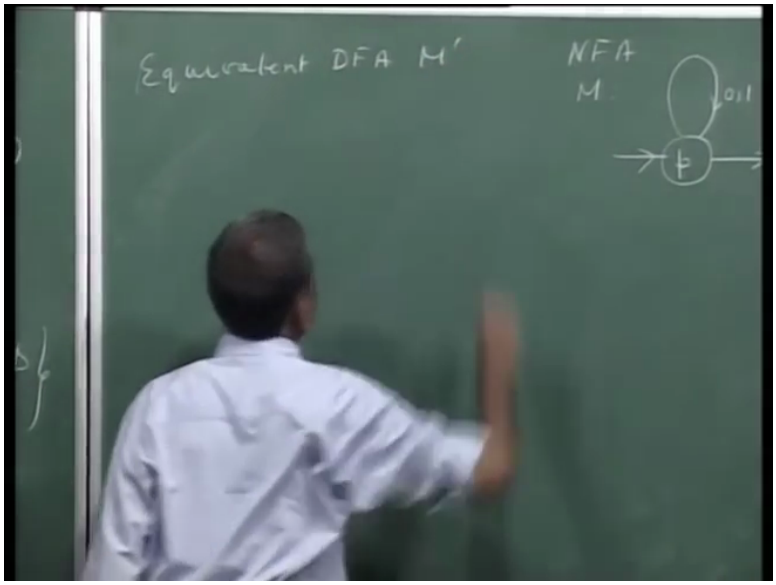
So q0 dash the initial state of the DFA it is clearly should be the set of states the NFA could be initial and that is of course q0, q0 is the initial state of the NFA and this is this is the set of states the NFA could be initially and therefore the corresponding subset is our state of the DFA and what should be F dash? When will the DFA know the strong would have been accepted by the NFA if it finds that the NFA can be in one of the states on reading this string x and one of these states is a final state of the NFA then the string would have been accepted.

So it is clear F dash we can define it this way that the set of all R, now this is the state of a DFA this is of course the set of states of the NFA such that R intersection F, F is the set of final states of the NFA is. So we have completely describe what M dash is, we have described what Q dash is, sigma of course same as that of NFA, sigma and then Delta dash we have defined and F dash we have define. So the DFA is totally defined.

(Refer Slide Time: 42:25)



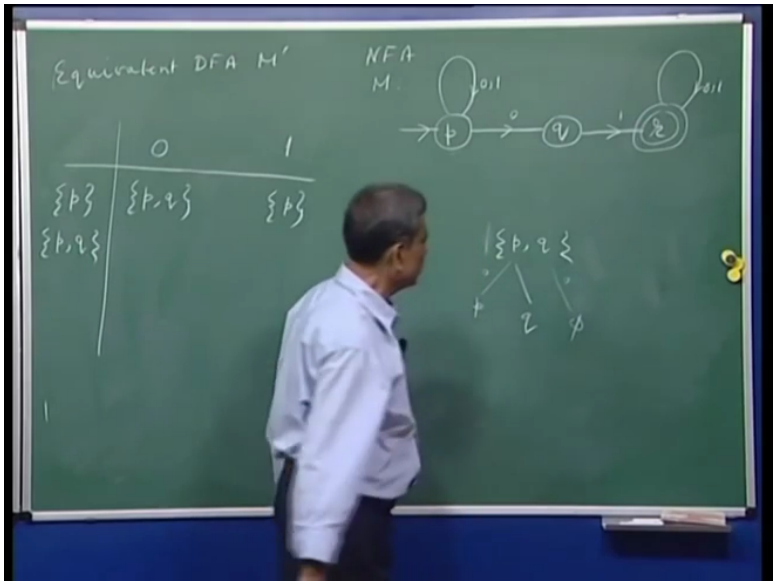
(Refer Slide Time: 43:02)



Now let me give you an example of going from given an NFA to a DFA which accepts the same language. As an example consider this NFA and you can see of course what does this NFA do? It accepts the set of all strings each of which has as a substring somewhere in the string 0 1 and we would like to using this idea that we have developed so far going from NFA to DFA we would like to get an equivalent DFA M dash.

Now we described of course in detail what the set of states etc will be but given such a problem it is instead of trying to first, see this NFA has 3 states and if you follow blindly that whatever you said you will start out with 8 states 2 to the power 3 is 8 but you see some of the states could be in accessible from the initial state, there is there is no way you can reach one of these subsets.

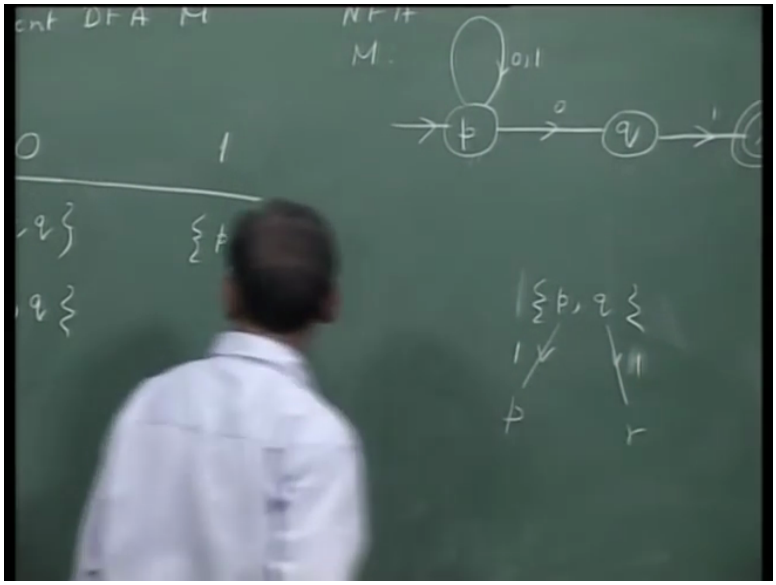
(Refer Slide Time: 43:57)



So there is no point keeping those, so we will start with the initial state of the DFA and see which all states it can reach on 0 and on 1, basically the symbols of the alphabet. So the initial state of the DFA is of course the initial set consisting of, this is the initial state of the NFA which is P and here is the state and there the symbol can be either 0 or 1, so from P on 0, let us see this properly once.

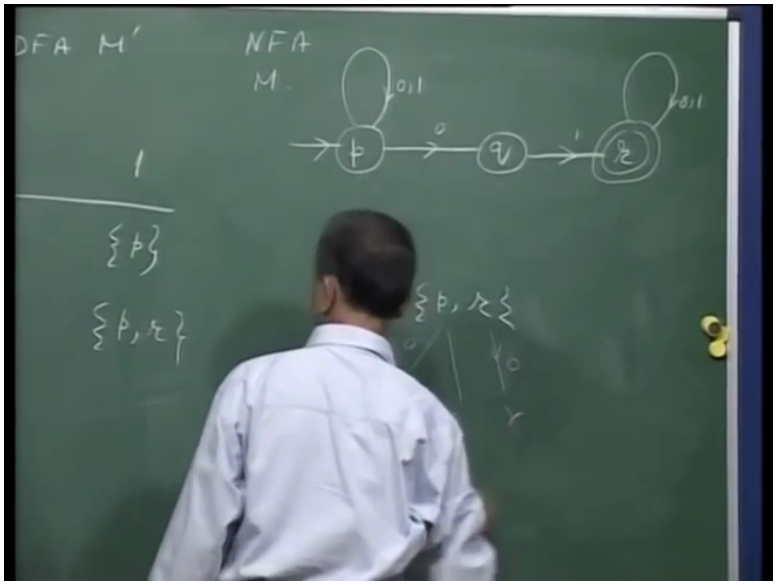
From P on 0 it can go to P as well as it can go to q, right? So therefore we write this and we can see from P on 1 then it has no other choice it has to remain in P, right? So now we have found another state which is reachable from initial state, so let P take care of the state. So the DFA is in this state Pq and let us say 0. So what is the state the DFA would be? So we see individually in fact that is what we elaborated there.

(Refer Slide Time: 46:40)



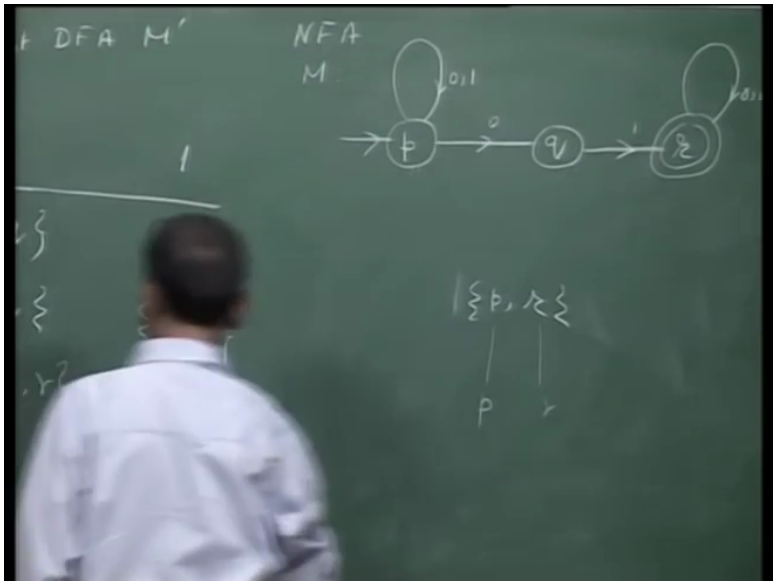
From P on 0 the NFA can go to either p or q and from q on 0 what is the states it can go to? From q on 0 transition is defined, so from q on 0 the NFA goes to the empty state, so take the union of all the states and that is P and now let us consider from state Pq on 1 which all states it can go to. From P on 1 it can go to of course P, from q on 1 this the NFA has no choice but it will go to R, so right on 1 goes to r, so from Pq on 1 it goes to P, r. So you have found discovered one more state is reachable from the initial state and it will take care of this.

(Refer Slide Time: 47:09)



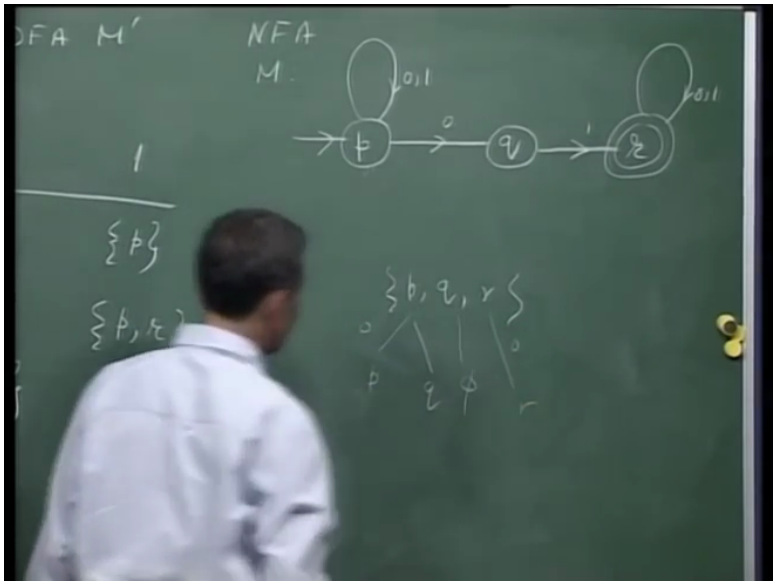
So Pr, we will do the same thing, so now it is P and r and now it came on 0 from P you can be either in P or in q, from r when 0 comes you will remain in r, right? So the state DFA will be P, q, r and what about 1?

(Refer Slide Time: 47:30)



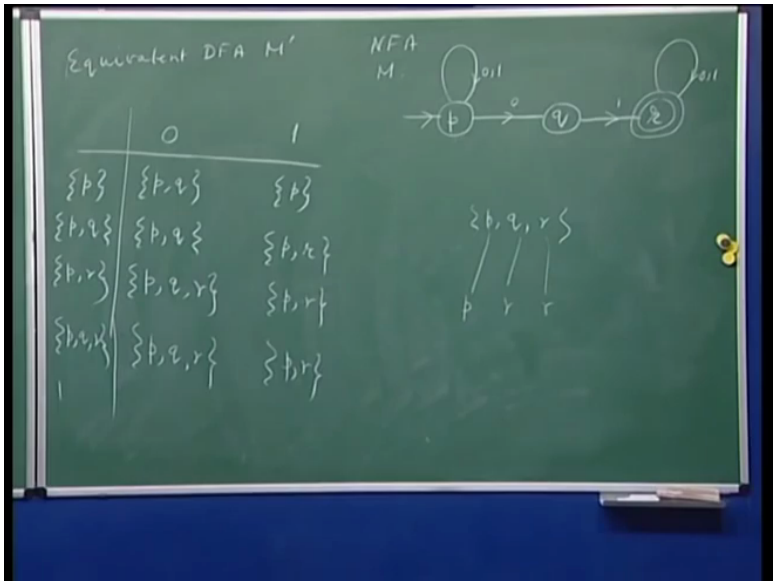
So from P the NFA on 1 can go to only P and from r on 1 you will see it can only go to r, so it is Pr, Pr is already taken care of but again we have discovered a new state P, q, r.

(Refer Slide Time: 48:09)



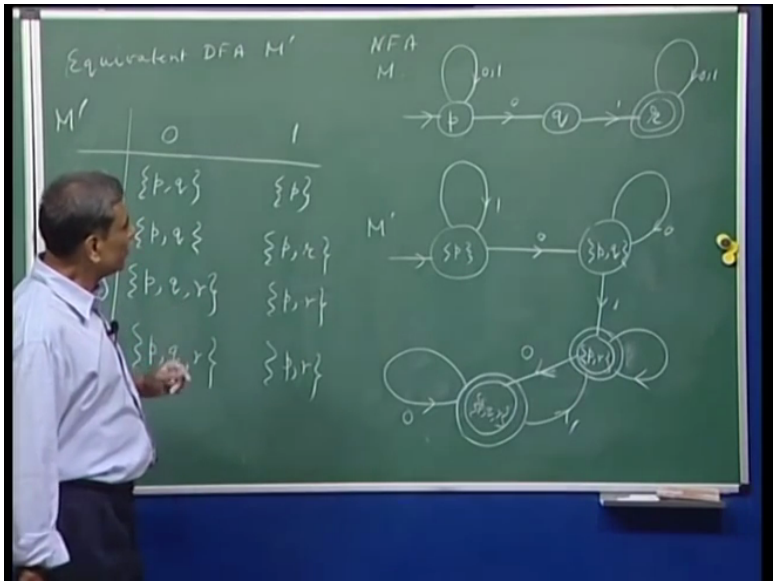
So P, q, r, from P on 0 it can go of course to either P or to q, q on 0 it goes to empty, r on 0 it goes to r. So P, q, r on 0 it goes to P, q, r.

(Refer Slide Time: 48:39)



And now consider the1 case symbol 1 from p on 1 it has no choice but to come to P the NFA, q on 1 it can go to r, r on 1 it can go to r take the union, so it is Pr, right? Now you see all the states, these are the states which the machine can go to starting from I mean these are the reachable P, Pq etc. So therefore I have completed the description of the DFA. By the way this is called a transition table because here we are listing out all the set of states of the DFA and here we have number of columns, one column per symbol and basically such a thing tells that if the machine is in this state and 0 came which is the state it can go to.

(Refer Slide Time: 49:48)

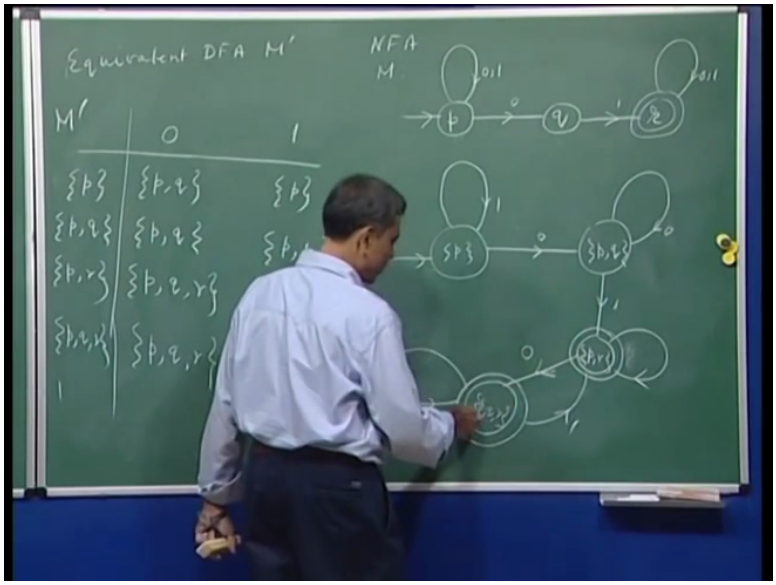


Let us draw the corresponding transition diagram, so that for the DFA M dash and you can see it has 4 states, the initial state is this P and on 0 it goes to a state called this state and one it remains in the same state initial state from Pq on 0 it remains in the same state on 1 it goes to a state Pr from Pr on 0 it goes to a state P, q, r and on 1 it remains, right? What are the final states?

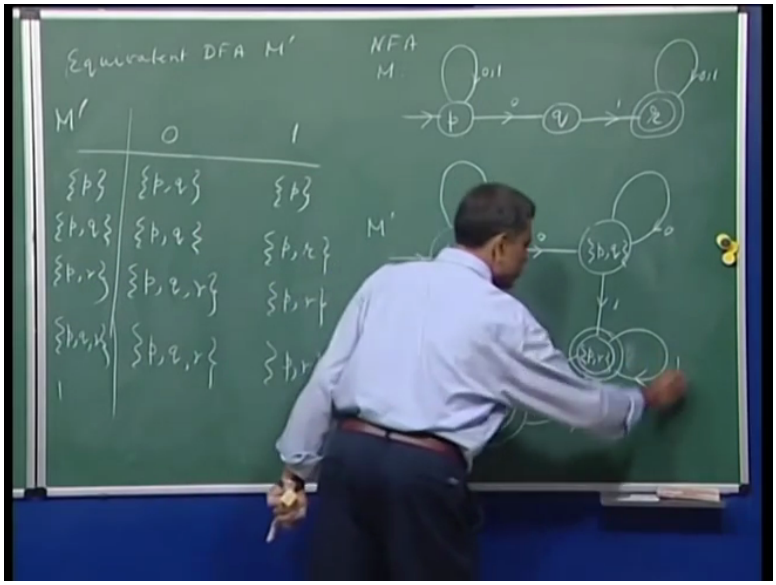
All those which have a final state is one of the members, all those subsets, so we can see the final state here is only r, so r occurs here, r occurs here. So these are the 2 things, oh! By the way we should have taken care of our Pr transitions and even this state transition P, q, r on 0 it remains here and p, q, r from this state on 1 you come to, so this is the DFA according to our idea.

Now 2 points, first of all I mentioned it once, if you had done mechanically you would have written 8states and then you would have found the states are not reachable, so those states we do not have to really show. So this is the DFA, now does it make sense can we informally see that M and M dash except the same language? And of course I know it will accept the set of all strings which has 0 1 somewhere.

(Refer Slide Time: 52:52)



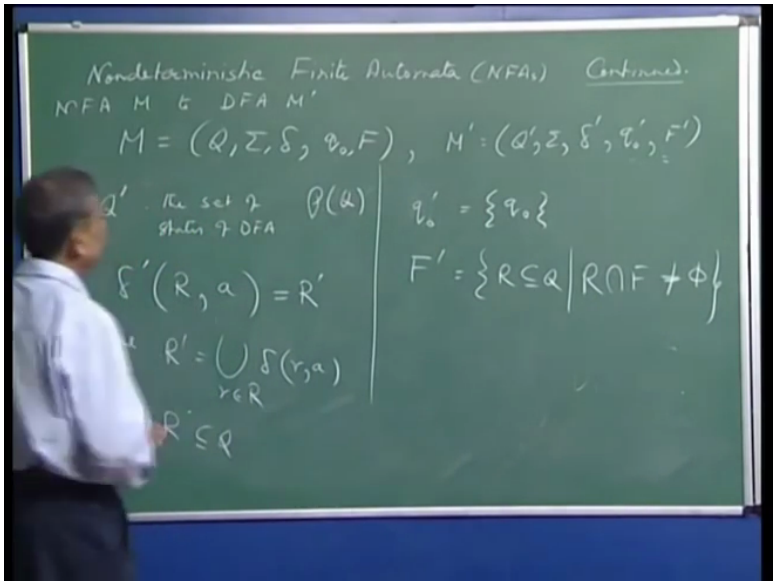
(Refer Slide Time: 53:02)



And you see what this DFA does? This DFA waits on 1 here and if 0 comes it is here that means it has seen 0, if it sees another 0 at this point, it says okay I have seen 0 which could be the first part of the interesting string 0 1 and once it finds the 1 then of course it is either in this state or in this state, I should have put down what that transition diagram symbol was Pr on 1 it means here.

So you see once it has seen 0 and 1, see then in consecutively, right? First 1's it escapes, it remains here then it sees 0 then, Oh! By the way, what happens on 1? Here, so it has seen 0 if your 1 then you have seen the interesting thing, so now whatever happens, whatever be the symbol which would come later on we should accept the string and therefore it is either of these 2 states, this is not the best DFA that you could design but of course this DFA accepts precisely the same language namely the set of all strings in which 0 1 occurs as a substrate.

(Refer Slide Time: 53:53)



So today in this lecture what we did was formally we defined what an NFA is and then we showed a very important result which is that given any NFA I can construct a DFA such that there is DFA accept the same language as the NFA, so therefore the class of NFA's does not have any language which is not accepted by a DFA. In other words NFA's also accept precisely the class of regular languages.