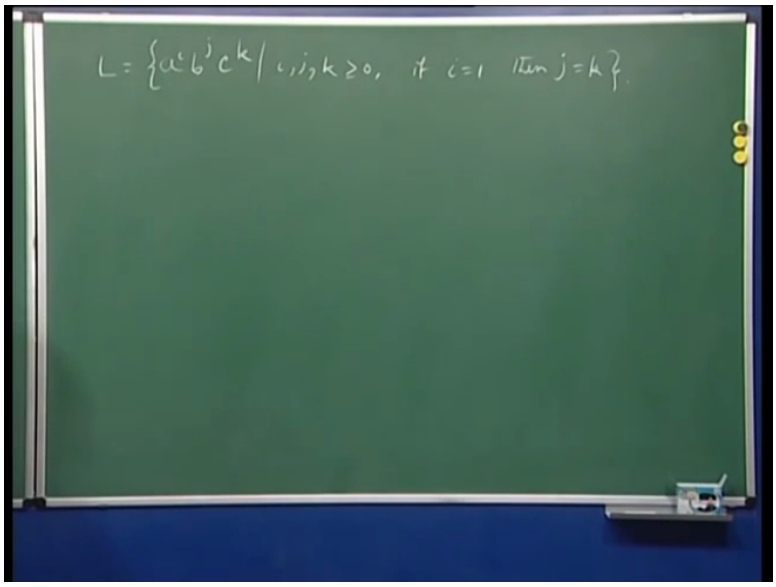
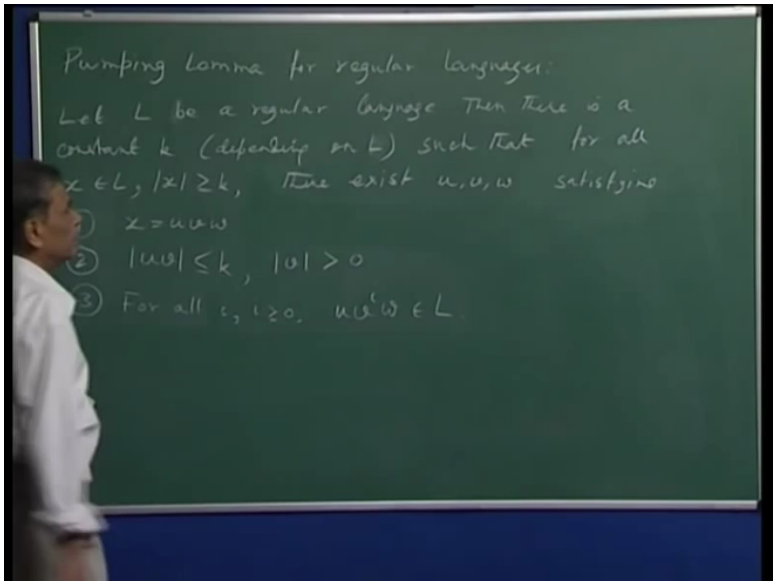
Theory of Computation
Prof. Somenath Biswas
Department of Computer Science and Engineering
Indian Institute of Technology Kanpur
Lecture 7
A generalisation of pumping lemma,
Non-deterministic finite automata (NFAs),
Computation trees for NFAs.

(Refer Slide Time: 0:21)

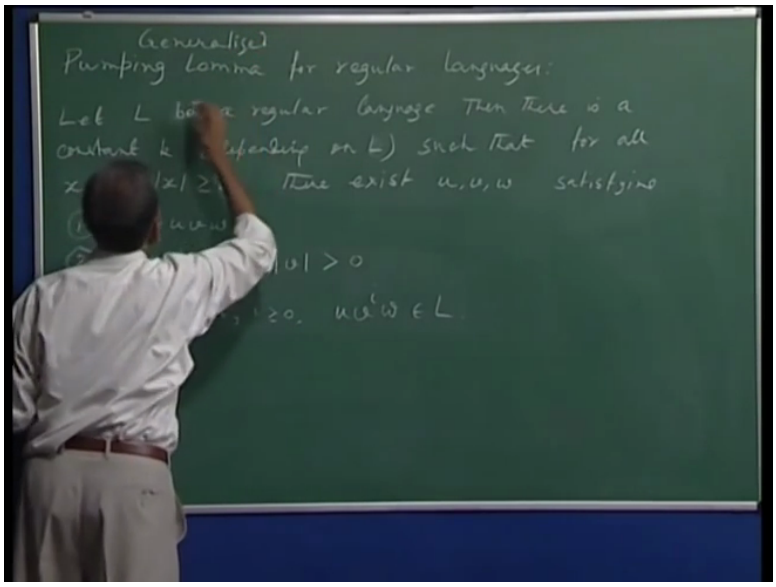


(Refer Slide Time: 0:27)

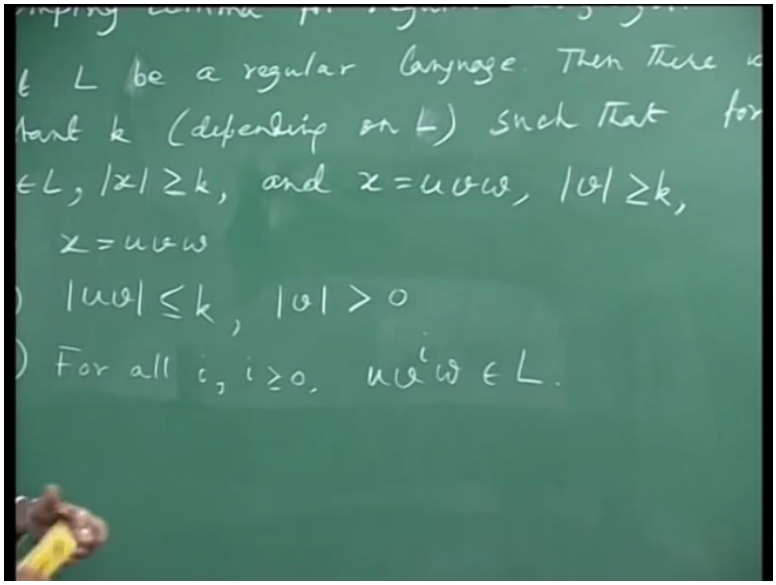


We have seen that this language cannot be proved to be not regular by using the pumping lemma that we have studied, we have proved this lemma. Now I had claimed that this language is not regular at the same time this particular form of pumping lemma is unable to show that. We can make a generalisation which is not too difficult to prove. So let me state that.

(Refer Slide Time: 1:02)

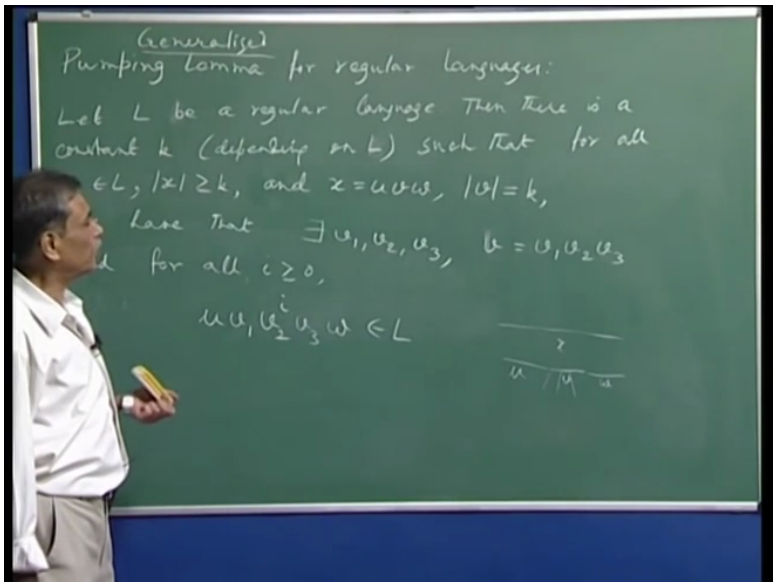


(Refer Slide Time: 1:49)



So let me say generalized pumping lemma for regular languages and we state something like this let L be a regular language then there is a constant k depending on L such that for all x in the language, x is greater than equal to k and instead of saying that there exist uvw, we will say here and x is equal to uvw where the length of v is greater than equal to k. We have no constraint on u and w and of course these things will change.

(Refer Slide Time: 2:05)



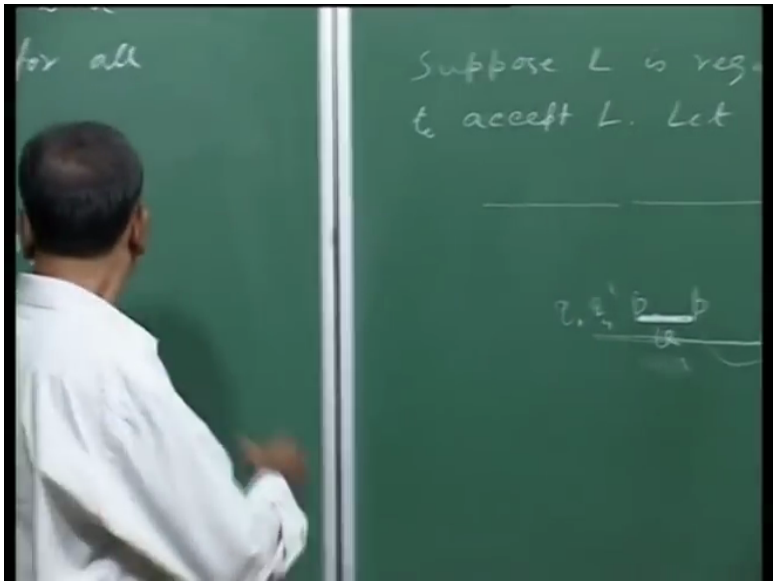
So let me remove these, right now what it is saying? That if you say L is a regular language then there is a constant k depending on L such that for all x in L, length of x is greater than equal to k and x is of the form uvw where v is of length greater than k. In other words it says

that v is a substring of x maybe anywhere in x whose length is greater than equal to k, in fact let me just this is okay, without loss of generally we can even say this that v is equal to k. x is greater than equal to k and x is uvw v is equal to k, we have that there exists v1, v2, v3 such that this string b is equal to v1, v2, v3 and for all i greater than equal to 0 the string u, v1, v2 to the power i, v3 w is also in the language L.

It is kind of similar to the old statement except what we were saying is that you take any string of length, any string in the language L whose length is greater than k and suppose you consider any substring in that string x whose length is equal to k that is v then it is possible to break up that v in terms of v1, v2, v3 such that see originally the string x was u then v then w this was the string.

Now we are breaking this in 3 parts or what we are saying is this v can be broken up in 3 parts v1, v2, v3 such that u, v1 this v2 any power of it followed by v3 and w this whole thing will be in the language. Now the proof is really really very similar to the idea that we have used to prove our original form of pumping lemma.

(Refer Slide Time: 5:25)

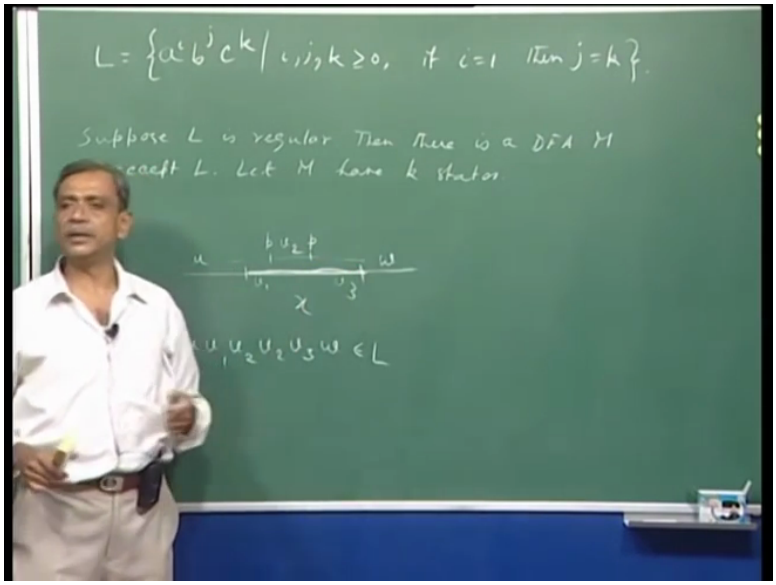


See v is of length k, right? How did that old proof go? Old proof use the fact that suppose L is regular then there is a DFA M to accept L then we say length M have k states and that k was the k that came here and the whole idea was that we took the first part of the string which is of length k in the old proof and then we said that string which we had called uv and the old proof, right?

And which was we said that look in this if you see the first k symbols of the string then consider the state sequence which takes the machine from q0 to a final state qf because the entire string is accepted there will be such a sequence of states and consider the working of the machine on the first k symbol in this there are k symbols then the state sequence here ui1 etc they will be up to this point when it is just over, the first k symbols are over.

There will be k plus 1 exactly k plus 1 states and since there are only k states, one of the states here must repeat itself without any State repeating I cannot construct a sequence of k plus 1 states using only k states and then we said the path in between these 2 states the part of the string which took the machine from p to back to p was v and then the argument was given that if you put any number of v's one after another and leave the first part and the last part this part was w, the new string will still be accepted.

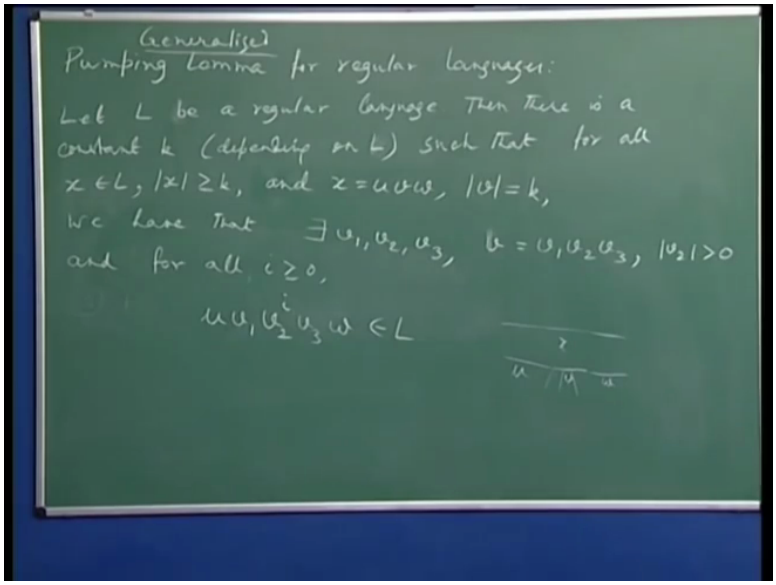
(Refer Slide Time: 8:41)



And now the wall all that we are seeing that consider now I am talking of the proof or the general case, what we said? Let this be x and let v be any substring of x which is of length k and then in this, for this substring which has k symbols consider the sequence of states the machine will go through first coming here and then coming here in that sequence there will be K plus 1 states the same reason because there are k symbols, so one of the states will repeat and that part which will take the machine from the same state p back to the same state p that we identify in this proof as v2, right?

And this part was v1, this part was v3 this will v2 and this part was u and this part was w and since v2 was taking the machine from p to p any number of copies after the first v2 will also take the machine from initial state to the final state, total string now then supposing you put v2 twice then it will be u, v1, v2, v2, v3, w this string also will be in the language and you can pump v2 any number of times and if you pump it 0 times that means this v, v2 you have removed that is all right, this string then will be v3 w which will take the machine from p to the final state. So that is the proof of the generalized lemma and that can be easily used to prove this particular language to be not regular.

(Refer Slide Time: 11:03)



Now we will consider the string, so now we are proving that L is not regular by using the generalized form of the pumping lemma. The proof will go like this that let L be regular and let k be the constant of the lemma and now consider this string a b the power k, C the power k which is surely in the language L because if i is equal to 1 then we require the number of b's to be equal to number of c's, so the string is in the language.

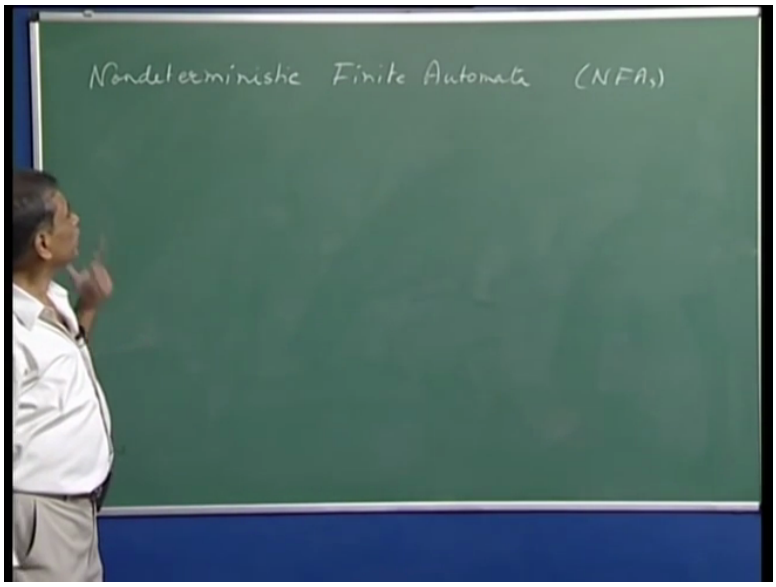
Now breakup this string as this is the u, this entire set of b is will constitute the v for the generalized lemma and this is your w and this says that there will exist v1, v2, v3 whose concatenation is this string v, right? So we can write this as u, v1, v2, v3 then w because w is all the c's there are k such c's this whole thing is your b's and this is u is only an a and I should have this, of course very importantly like in the old case, old form of the lemma that length of v2 is greater than 0 in the generalized form and remember the reason for that is when in the state sequence the state repeats even if these are consecutive states there will be at least one symbol, so repetition means there will be some symbols in between and that those symbols constitute the string v2 therefore length of v2 is greater than 0.

So now you can see that on pumping some number of times the number of b's will increase, nothing else will change in particular number of a's will not change and number of c's will not change, right? And therefore we will get a string in which I will have on pumping we will get the string ablc k where L is strictly greater than k, why? Because we have pump v let us say twice and therefore originally the string had kb's the v2 had had at least one v, now you

have added 1 or more b's, so the number of b's have increased from k to something higher number.

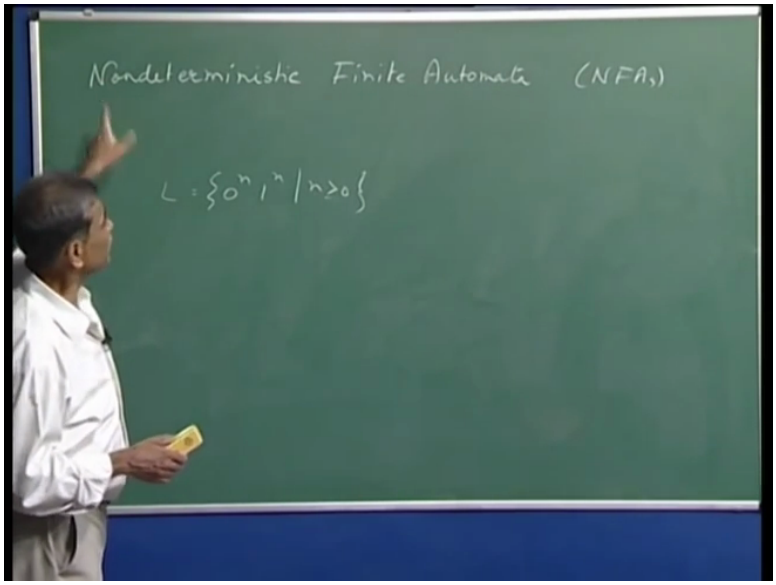
And now the lemma states that this will be in the language but that contradicts the definition of the language that there is only a and yet the length of b's is not same as the length of c's, right? So the general form of lemma, does indeed prove that the language L is not regular, do remember what I initially forgot to write was crucial that the v2 part is non-empty but that I am sure you will appreciate comes from the proof, once we have it in mind that v2 part is the part of the string within v that takes the machine from the same state to the same state and since these 2 states are occurring separately v2 part must be nonempty.

(Refer Slide Time: 16:42)



Our next topic is a generalisation of the finite automata that we had seen so far and we call this generalized form of finite automata as nondeterministic finite automata and before we give the definition and see examples of this kind of automata, let us appreciate why we are interested in generalizing our automata that we have seen so far, we define this class of languages, regular languages and we had proved many languages to be regular and we know that by definition the regular language is the language which is accepted by a DFA but we also saw the that DFA's are unable to accept some very simple looking languages.

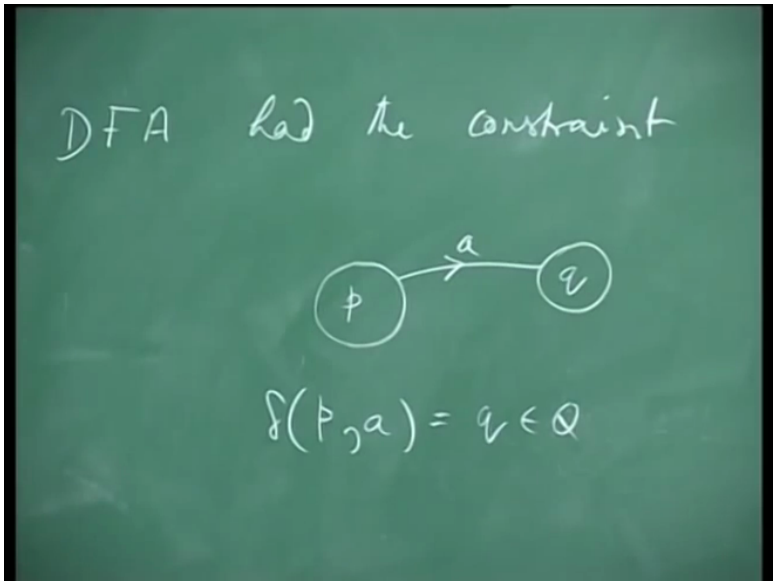
(Refer Slide Time: 17:49)



For example this is a very simple looking u degree that we are just saying the language is in which we have 0's and 1's in that order and the number of 0's is same as number of 1's, this is something this is a language no DFA can accept, so it is not regular, so obviously you would like to generalize the class of languages for which we can have, we can define formally automata and other devices for accepting.

Now it is because of that reason we think of generalizing finite automata, so the way we think of the generalisation, although by the way let me just say that although we are saying this is generalized form you will turn out that this form also will not be able to accept anything other than regular languages and yet this is an important topic because of the reasons that will (()) (19:02) later.

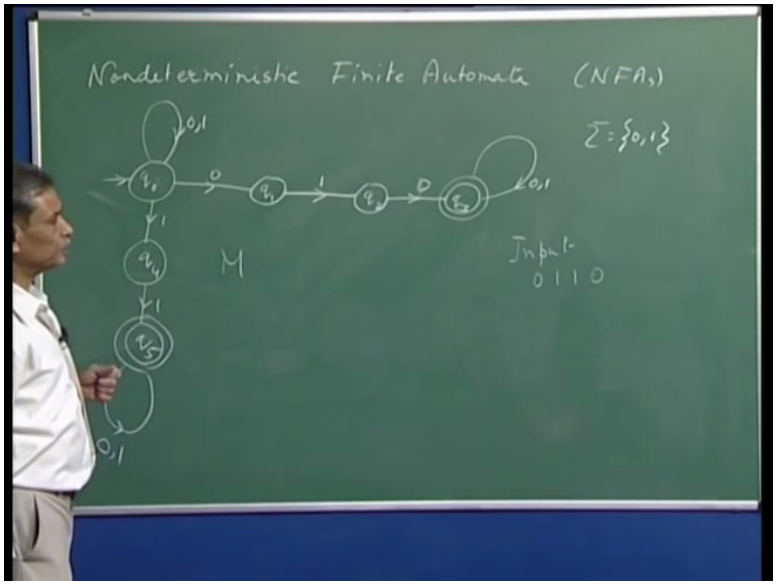
(Refer Slide Time: 19:11)



If you go back to DFA definition we had the constraint that call it constraint or call it the way it was defined was suppose the machine is in some state and then a symbol comes a, right? So this is p then the machine will go to some other state q, right? In other words we said that Delta p, a will be some state q in all set of states.

So what is happening is that from a state on when a symbol comes a there is a unique transition that is possible, right? So from every state one adds on every symbol there is a unique next that make sense the way we have defined the automata but suppose someone asks this question that what happens if I relax this? What happens if I say that from a state on a symbol there can be 0, 1 or more transitions?

(Refer Slide Time: 21:13)



Now let me first give an example, you see, what we mean by saying that from a state there can be 0, 1 or more transitions out on the same symbol. So consider this automaton, okay. Look at this, this is certainly not that, is all I have given you all the arrows that I need to define this automaton, you see in this state on 0 you could come back to this state because of this transition, also there is a transition out of this state on the same symbol 0 to this particular state.

Similarly on 1 from this is state there is a transition back to the same state also there is a transition to this state. So from this state on both 0 and 1 both the symbol 0 as well as 1, we have not one but 2 transitions out of that state. On the other hand look at this state, in this state we have a transition on symbol 1 but there is no transition out of this state on the symbol 0 implicitly I am assuming that the alphabet is binary.

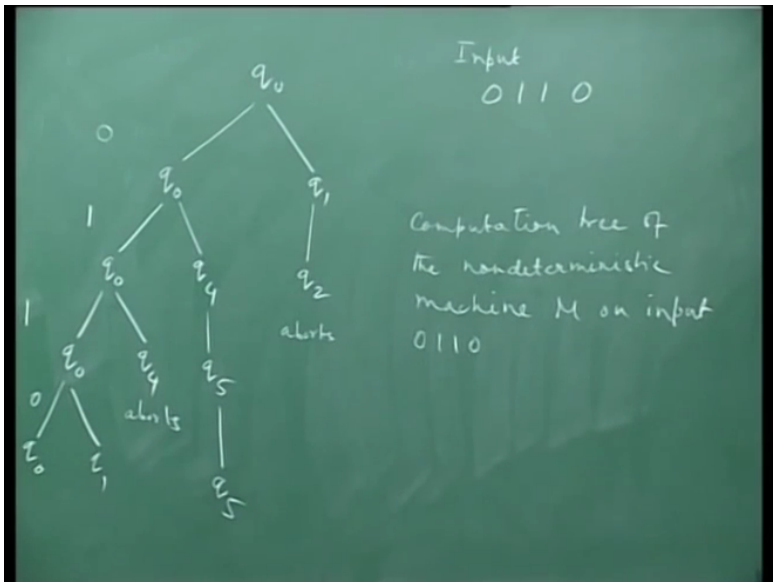
So let me write it clearly here that although my alphabet has 2 symbols, in this state for example on 0 we have no transition defined, similarly from this state there is on symbol 1 there are no transitions defined. So at least this particular automaton certainly has that property that we talked of that from one state there can be either 0, 1 or more transitions from the same symbol.

Once we make this kind of an automaton at least defined this kind of an automaton, what does it mean to say that this automaton has an input and is doing the computation of the input. So this we should be clear about, what does it mean to say that we have a computation

of this automaton? Now like a DFA this NFA also have some input which is a string over the alphabet.

So let us say our input is 0 1 1 0 and let me name these states, so this is q0, this is q1, q2, q3, q4, q5, so there are total 6 states and as before an arrow coming from nowhere it means that this is the initial state, so the this NFA, this automaton if I name it M, initially it is in its state q0, okay.

(Refer Slide Time: 26:13)



So that is easy to see, so here is this input 0 1 1 0 and initially the machine is in state q0. Now let us say the input is there and the input, the first symbol of the input is 0, so what happens, in fact this is what the first question you will ask that there are 2 transitions out of initial state on 0 then what happens, what does the machine do, which state the machine go to on 0, starting from the initial state q0?

Well conceptually the way to think about these machines will be that it can go to all these arrows etc all what they are saying is that since there are 2 transitions on 0, out of this state the machine can go to one of these 2 states either q0 or q1, which state it will go to? We will say non-deterministically machine makes a choice to take one of these 2 transitions but what?

So the initial symbol is 0 we can imagine a picture like this if I write out all possibilities, so from q0 on 0 the machine can go to q0 itself as well as it can go to the state q, let me complete this and then we will discuss a little more about what we mean by computation of

an NFA. Then the next symbol 1 came from q0 on 1 you see it can remain in q0, so this is the next symbol 1 that if and on 1 it can also go to q4, what about q1?

See had it gone to q1 from the first 0, now it is seeing the second symbol 1 from q1 on 1 it has to go to q2 there is no other choice than the third symbol 1 comes and from q0 as we have seen earlier it has on 1 it can either go to q0 or also it can of course go to q4 and from q4 on 1 where can it go to, see this is the state in which it saw 1 then these are the 2 possible states it could go to from q4 on 1, if you see there is only unique state it can go to which is q5.

From q2 had the machine take in this path from q0 that the machine on the first 0 chose to go to q1 and then q1 when it saw 1 it had no choice really except to go to q2 which is okay. Now on q2 it sees the same 1 but what happens on q2 on 1 there is no transition defined, which state the machine goes to? It is not defined, right? So we say that if machine had taken these choices than the computation aborts, right?

There is this path is nothing much to do because on q2 1 had come and there is nothing else to do other than the machine abort, by abort what I mean is let me make it clear, see the suppose the machine was in this, in state q2 and 1 came since no transition is defined then that computation cannot proceed any further and that is what we mean by the computation aborting, the computation no more nothing else is defined about the computation.

Now you see another 0 comes, the machine supposing it had taken this these choices from q0 onwards 0 it went to q0 on q0 on second I mean second symbol which was 1remain in q0 and this it can again as 2 choices from q0 on 0 it can go to q0 itself or it can go to q1 from q4 on 0 we have a problem again because q4 you see there is no transition defined on 0, so this path aborts, this computation aborts there is machine could be in q5, q5 on the 0 it will remain in q5.

So one thing is clear even if you define our machine in this way where there can be 0, 1 or more transitions out of a state as in this case, what we can do is that given an input we can see which all states the machine can be at the end of this string and you can see this is, this is the kind of tree is not it?

This is a tree and the at this level when I have 0 1 1 0 each level is talking about what are the states the machine can be after seeing that part that much of the that some initial prefix of the input when I have this tree and this is called the computation tree of the nondeterministic

machine. Machine M on input 0 1 1 0, I would like to emphasize that while talking of nondeterministic machines we of course see that the machine is faced with certain choices at certain points in the computation.

For example here if the machine was in this state after reading 0 or 1 it has the choice of either going to q0 or q4, what happens, you may ask what actually happens, if I start the machine on this input what actually happens? What actually happens is one way of thinking of this is the machine follows one of these parts, which path the machine will follow actually?

We do not see because all we see the machine can follow this, can follow this, can follow this; this is like a human being, right? I mean you are walking on the road and the road bifurcates, we cannot say which 4q will take I have the choice, if I am faced with the road we are going and there is a circle and there are 2 roads, the road is bifurcating I can either take the left part of the right part I have the choice.

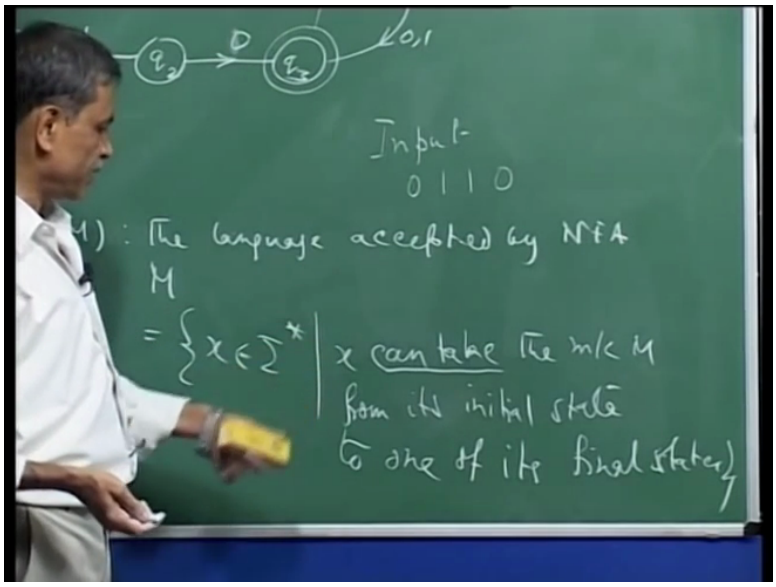
Similarly this machine I mean at least intuitively at a conceptual level we can think in this way, did the machine has the choice? And sometimes if you run it many times maybe sometimes it follow this path, maybe sometimes it follows this path and so on but one more thing that there is no probability involved, we are not saying that with certain probability it will take this path, this state or it will go to this, it is not probabilistic machine although such a definition is possible.

We will not consider probabilistic finite automata here but that is possible but here it is not that it is based any probability etc associated with these choices that it takes one transition with certain probability, another transition with another probability it is really really the best way to think of these machines is the machine is making a choice. Next question that comes is how do we realize (()) (37:21) and how do I realize in a machine choice, right?

Those of you who know a little bit of electric I mean you have seen some first forces in electrical engineering you have seen flip flops it is not difficult to see that I can realize a DFA using hardware it is very easy to see I can simulate a DFA, I can write a program which will see it, simulate the working of a DFA but can I write a program to simulate or to work like an NFA, I do not know I mean there is no such machine I know of which makes choices. So then what good is this kind of machine you may ask?

See what we can do is, we can use this theoretical construct of non-determinism to define languages. Now what is the language accepted by NFA, right? Suppose in that definition it is clear like in the case of NFA, se in case of NFA sorry DFA we wrote it we presented a DFA and then we said that there is a unique language associated with this machine which is the language accepted by the DFA that will be the similar case even with NFA's.

(Refer Slide Time: 39:09)

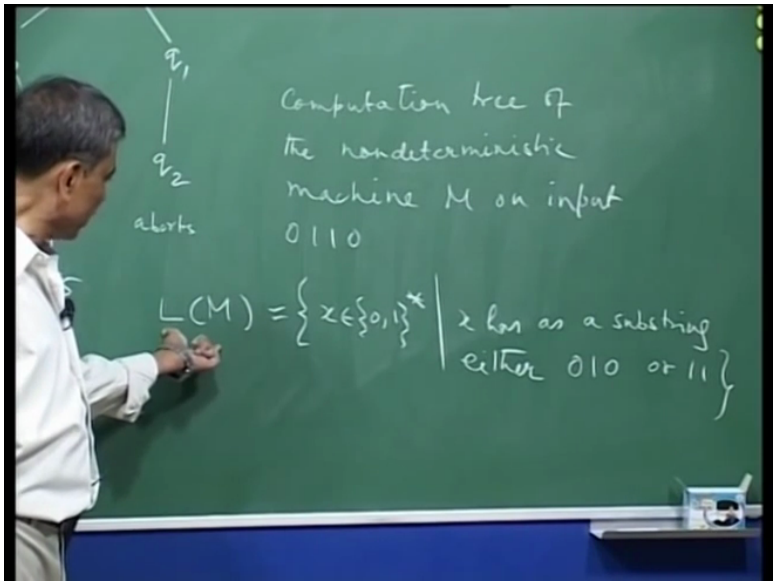


So let me informally write the language accepted by M, LM the language accepted by M, NFA M to be the set of all strings over my alphabet, all finite strings over my alphabet and what is the qualification on these strings such that x can take the machine M from a initial state to one of its, now operative phrase is can take, x can take the machine M from its initial state to one of its final states.

Let us look at this computation tree for the example machine that I have drawn. Is this string in the language accepted by the machine M? Well, which are the states the machine can go to on the input 0 1 1 0 that I have seen q0, q1 and q5 and as it happens q5 is a final state. So do you see that the machine M can go from its initial state q0 to one of its final states q5 on the input 0 1 1 0's therefore this string is in the language accepted by the machine M, the way we have defined.

This is an informal definition as I am saying, we will formalise it later but let us work with this informal definition for a little while more. I have just told you that the string 0 1 1 0 can take the machine M from its initial state q0 to one of its final states as it happens here q5 therefore we will say that 0 1 1 0 is in the language accepted by M.

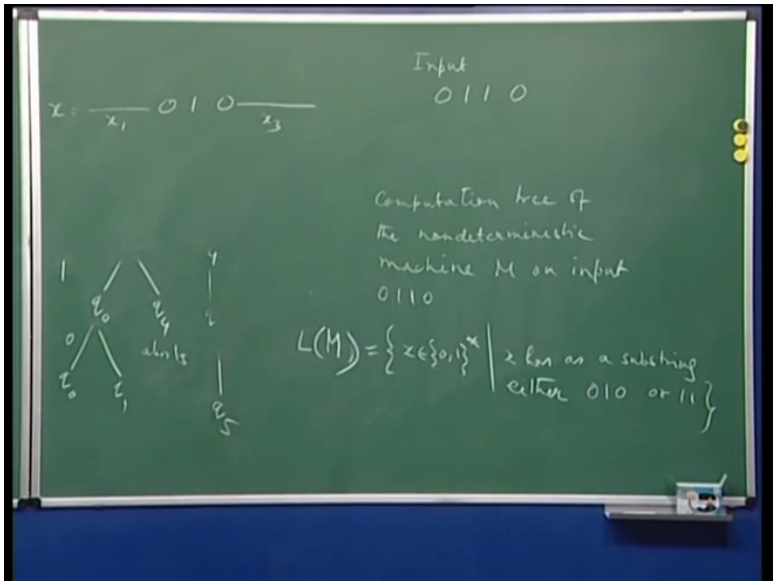
(Refer Slide Time: 41:54) 42:32



Can we see what precisely is this language? I claim that LM is the set of all binary strings 0, all finite binary strings such that x has as a substring either 0 1 0 or 1 1. I claim that this is the language accepted by the machine M the notion of the language accepted by a nondeterministic machine, remember is the set of strings each of which can take the machine from its initial state to one of the final states.

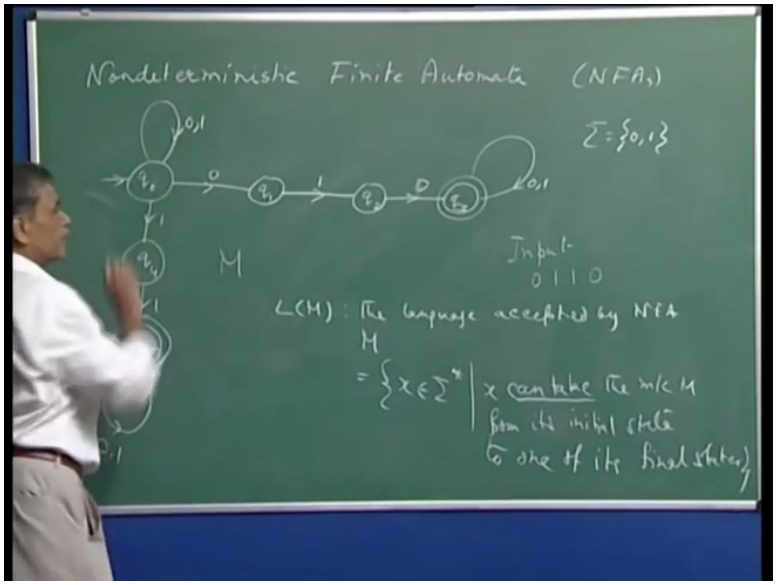
So can we argue this out? That using that notion of acceptance the language accepted by M is precisely this, alright. First of all think of a string which has either 0 1 0 or 1 1 as a substring. So in particular think of a string, we do not need this anymore, so let us think of a string, first this is there, think of a string which has as a substring suppose 0 1 0 our string is x.

(Refer Slide Time: 43:32)

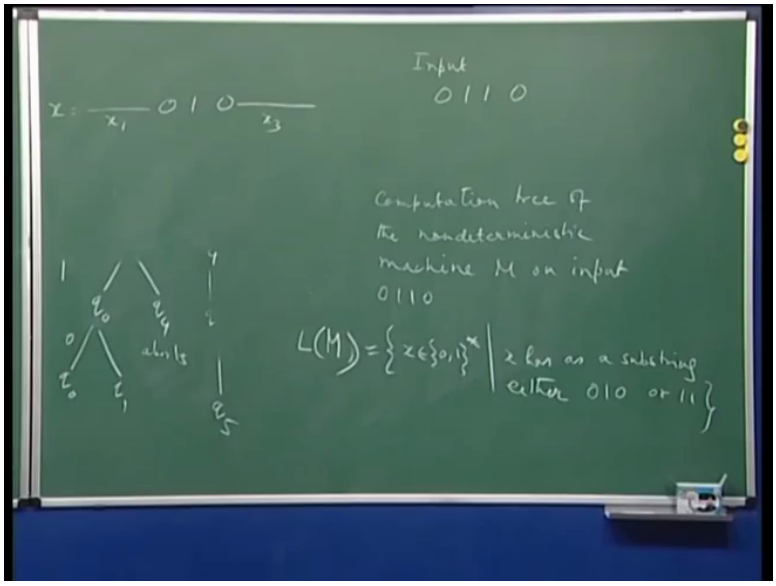


So there is some initial part and followed by that substring there is some other part, so call this first part as x1 and the last part as x3. Now what the machine can do? Remember I am not saying I can never say about NFA's that machine is going to do this or the machine will do precisely this but what I can definitely say is the machine can do such and such thing using its choices.

(Refer Slide Time: 44:19)

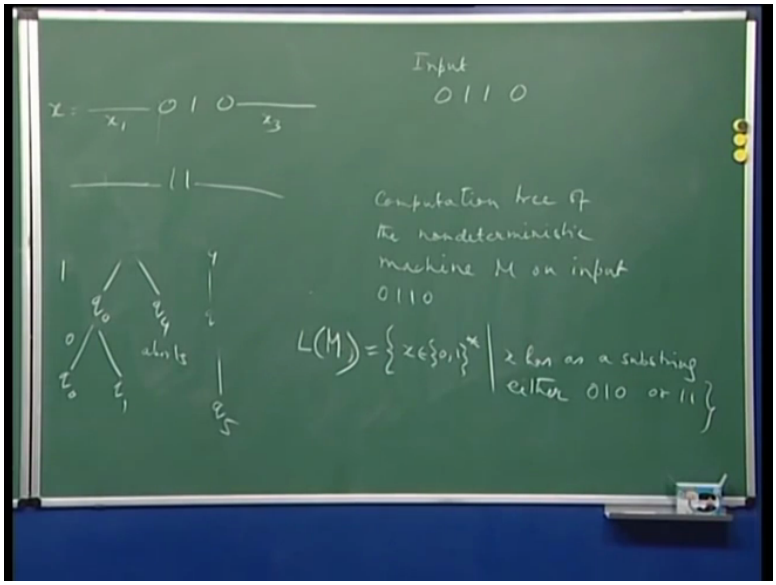


(Refer Slide Time: 45:17)



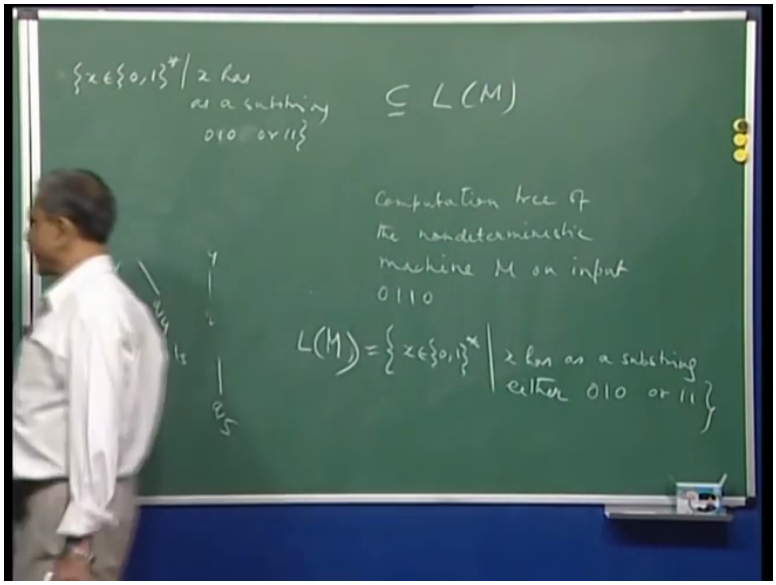
So what the machine can do that on x1 it remains in this state, right? Imagine x11 was some 0 or 1 something 1, some this is x, your x1, right? So what the machine, machine of course could have done on 0 it could have right on the first 0 it could have gone to either q1 but it chose to remain in this state q0 on this part x1. Now on this 0 which is the first symbol of the substring of interest the machine makes this transition to q1 and then 1 comes it has to go to q2 it does so then 0 comes it comes to q3 and then it remains in q3 because you know of this loop.

(Refer Slide Time: 46:00)



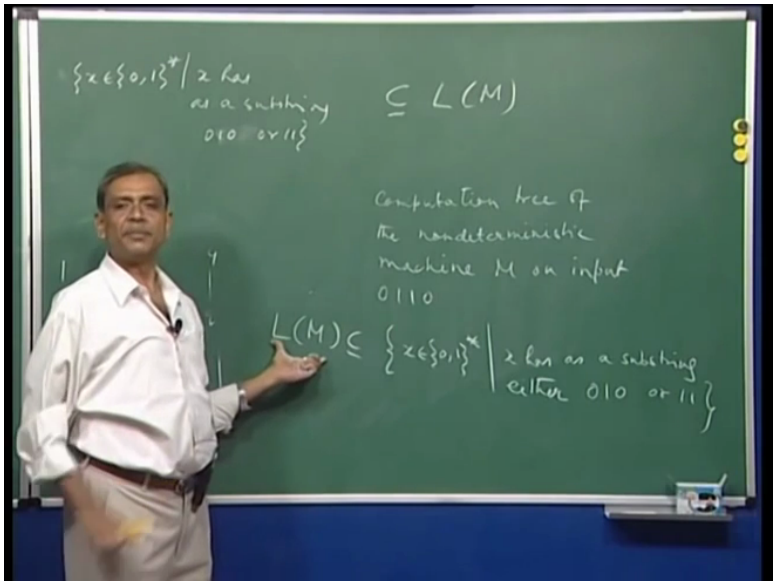
So do you see that on a string x which has as a substring 0 1 0 it is possible for the machine to from the initial state, its initial state q0 to this final state q3 and similar is the case that on any string which has a substring as a substring 1 1 the machine what it can do? On this path it will choose to remain in q0 and on this 1, now there may be one switch might have happened before but on all those 1's it remain in q0 and then on this particular 1, this 1 1 is the substring of interest in in q4 on q4 then 1 came it went to q5 and then the rest of the string it will remain in q5 then therefore again on this string the machine has the capability machine can go from its initial state to one of its final states in this case it is q5.

(Refer Slide Time: 47:16)



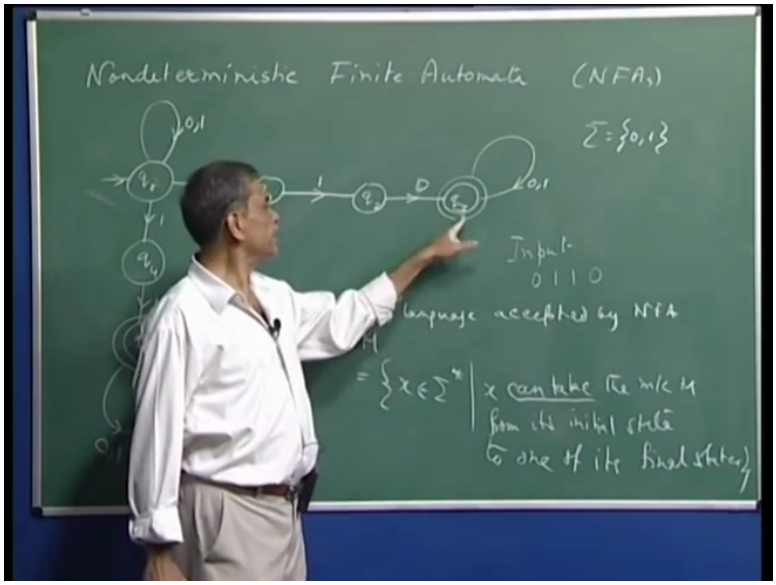
So my claim is that if you if you work with this definition of the language accepted by the nondeterministic machine M then all these strings which has as a substring either 0 1 0 or 1 1 such strings will be accepted and therefore what I have really shown to you that the language accepted let me write it this way that x, I am just repeating that x has a string 0 1 0 or 1 1 this set this set is a subset of the language accepted by our machine but if I want to show equality I need to prove the other way.

(Refer Slide Time: 48:04)



I need to prove also, along with this I need also to show that suppose a string is accepted by this machine by that what we mean the string takes the machine from, not take solution I am sorry, such a string can take the machine, so any string here by definition is a string which can take the machine from the initial state to one of its final states then such a string has to have either 0 1 0 or 1 1 as a substring.

(Refer Slide Time: 48:42)



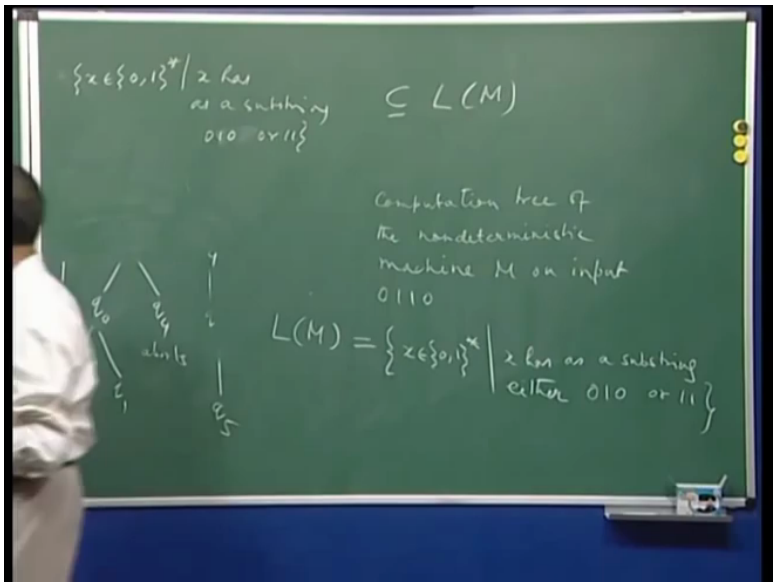
Again that can be looked at in this manner, that imagine that string, now that string finally and imagine that those choices because of which the machine went from q0 to one of these final states, consider in particular that the machine went to q3, initially it was in q0 then that string came and that machine exercise its choices and finally at the end of the string it is in this state.

Now do you see that string has to have 0 1 and 0 in succession, why? Because finally we are talking of a path which went from 0, q0 to q3 that path there might be many this loop that path might have taken but since it came from here if I some point in the path q2 must have come immediately after which q3 came and that transition is possible only on 0 from q2 to q3 and then it came it came to q2 and the state immediately before that must have been q1, must have been 1 because only way the machine can come to this state q2 is from this state on 1, right?

See that is the only transition and so the machine sometime came to q1 after which there was 1 0 and then some part of the string and then it came to, now you are saying it came to q1 but it could come to q1 only on 0. So therefore put together this 0 1 0 must occur in any path, right? Whose levels are these such symbols you understand what I mean, any path from q0 to q3, if I think of any path which takes the machine from q0 to q3 that path must be labeled with 0 1 0, alright.

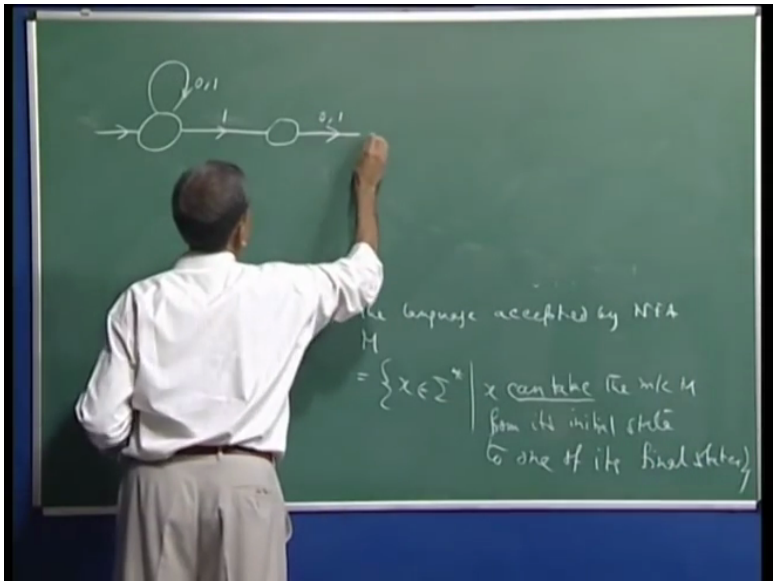
Which means that if the machine went to q0 to q3 on a string then that string must have 0 1 0 as a substring. Similarly on another way of accepting a string would be that it finally went to q5 and for that the string must have 1 1 as substring, so that will prove this assertion.

(Refer Slide Time: 51:29)

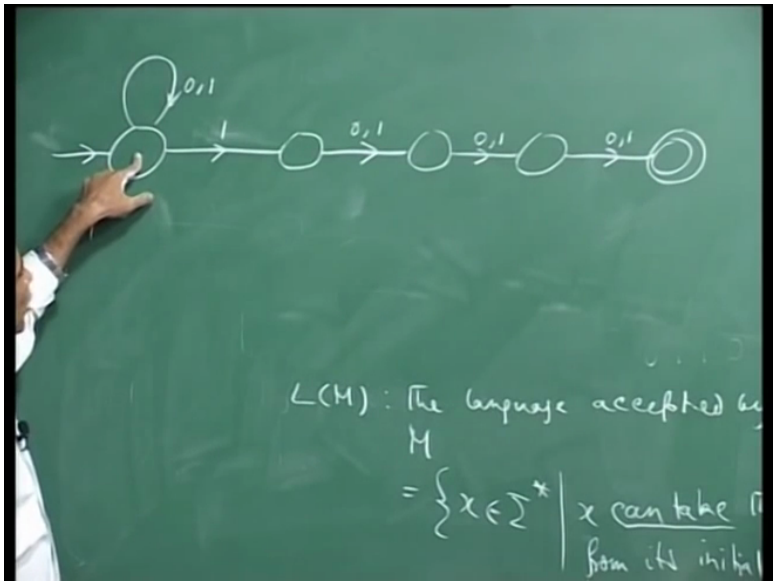


So therefore I can claim put together that this is a subset of LM and LM is a subset of this therefore it is an equal, alright.

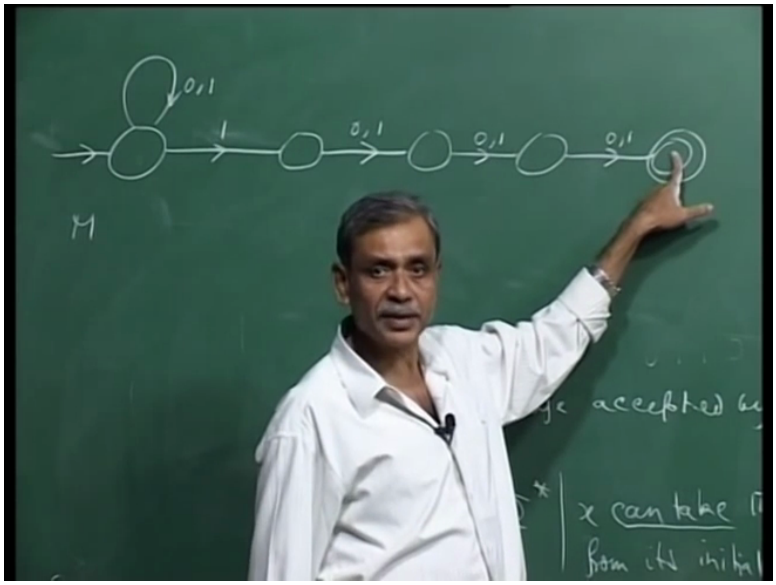
(Refer Slide Time: 51:48)



(Refer Slide Time: 52:21)

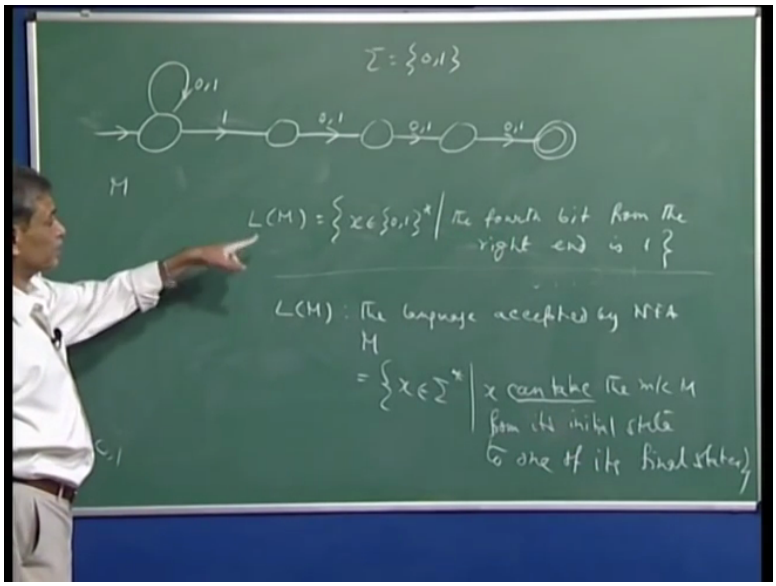


(Refer Slide Time: 52:32)



So let us see another example we will keep this definition here but the example will change, so consider this machine, well this is my another machine M which is again a nondeterministic machine, it is not a deterministic machine because from this state on 1 it could go to either here or it can remain in this state and you see there is no transition out of the state which is the only final state the machine has neither on 0 nor on 1 it has a transition.

(Refer Slide Time: 52:45)

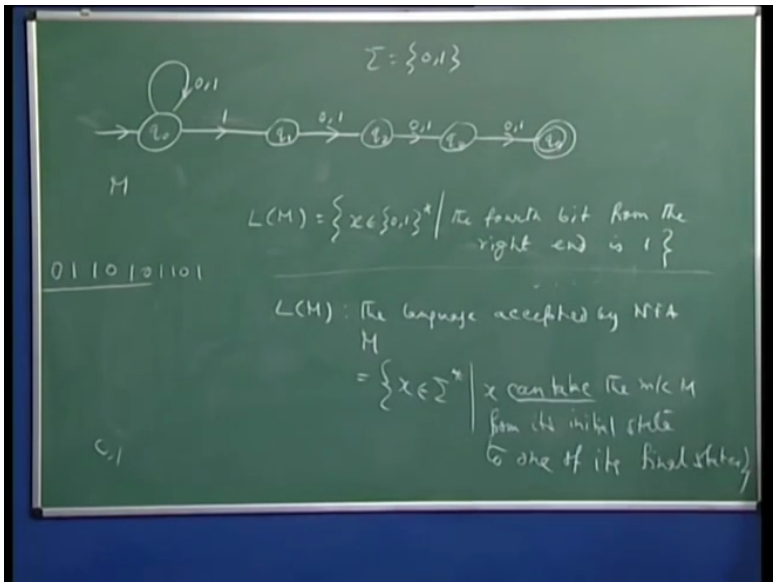


If I call this machine M and I now claim that LM again my alphabet it is binary I claim that LM is a set of all binary strings finite binary strings such that the fourth bit from the right end is 1. Now you have encountered this sort of languages before when we were discussing with

the DFA and remember that even prove that for this language to be accepted by a DFA that DFA must have at least 2 to the power 4 states and in fact we showed that 2 to the power 4 states will suffice but no DFA can accept this language with less than 2 to the power 4 states that we prove.

2 to the power 4 is of course 16, so that DFA to accept this language would require 16 states but if I am correct in making this assertion then I have an NFA to accept the same language with much fewer strings namely 5 strings.

(Refer Slide Time: 54:27)



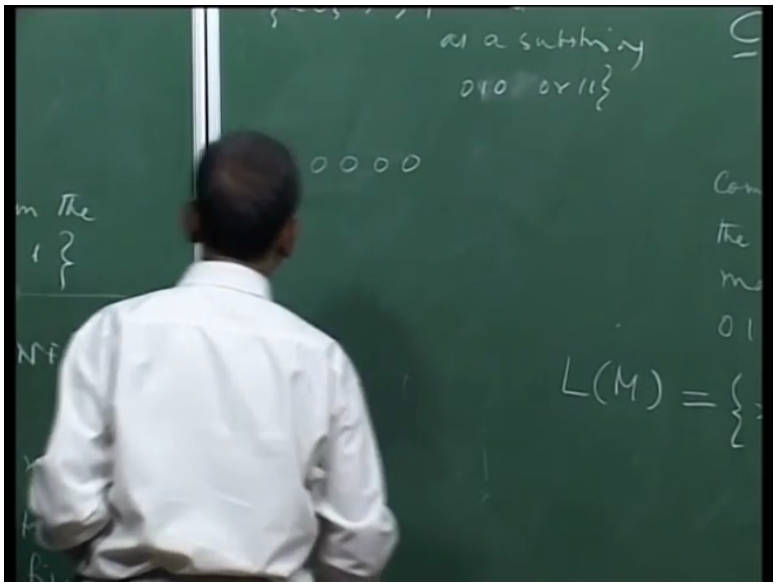
Well, imagine I have a string whose fourth bit from the right is one and let us say I have some more bits here. Now it is very easy to see that this string will be accepted by this machine, why? This string, remember, why? Because this string can take the machine from this state to that state, why? Because you know on up to this point before just a bit before the fourth bit from the right, this is the fourth bit from the right 1, 2, 3, 4 all these time the machine chose to remain in this state.

And now this 1came it had the choice of being here as well as going there but now it exercise the choice of going to this state. Now once on this it went to this particular state, supposing I name them as q0, q1, q2, q3, q4 then of course the next one keep either on 0 or over 1 from q1 it will go to q2, it went to q2 then this 0 came it went to q3 and then after q3 this 1 came it is now in q4, right?

So this string can take the machine from q0 to q4. So it is fairly easy to see that any string which has 1 as its fourth bit from the right end such a string can take the machine from its initial state to its final state and similarly I can argue the other way round that suppose I have a string which is accepted by this machine which is in the language, so that string will spell out a path from here to here and you see once the machine reaches here there is no transition out, if those any further symbol than what would have happened? The machine computation would have aborted, right?

So it must be that path, it must be such that the fourth bit from in that path when I look at the symbols the fourth bit from the right-hand must be 1, right? So it can it can look at this example and convince yourself in the same manner if need be you can draw a few a computation trees as we did in the first case for some strings.

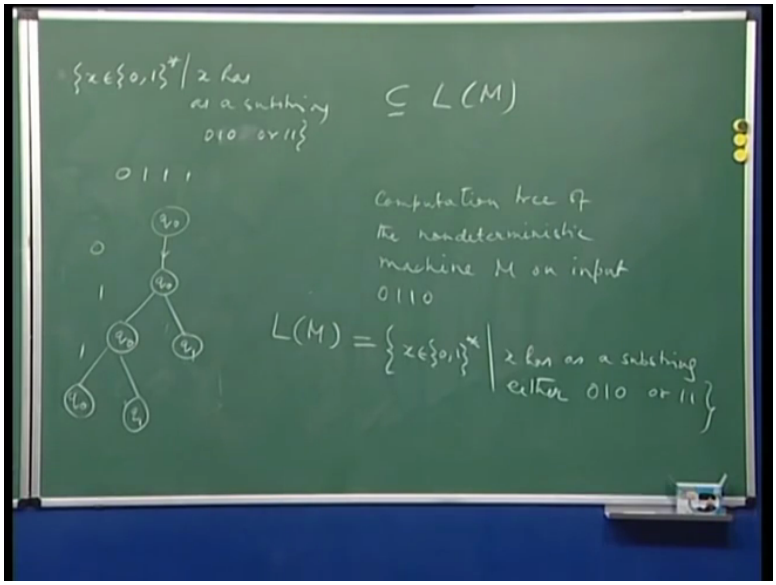
(Refer Slide Time: 58:00)



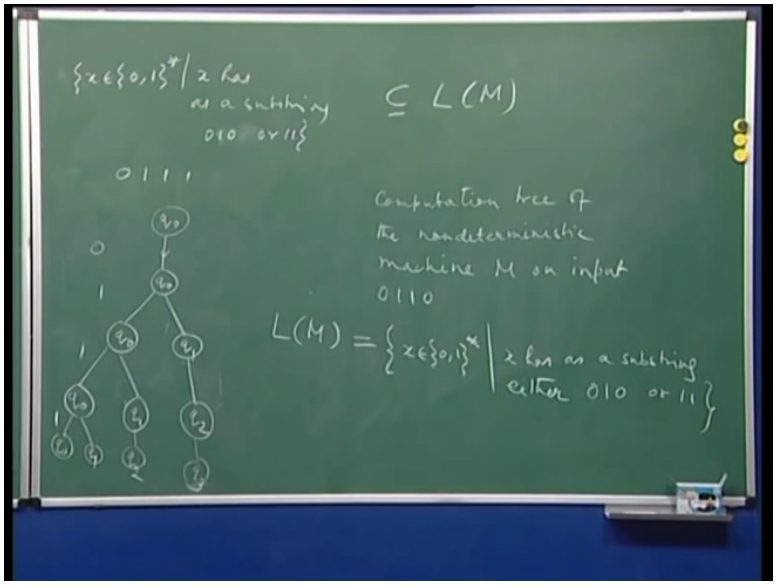
For example let us take a computation tree for a string which is not in the language, my claim language. So let us say 0 0 0 0 the fourth bit is not here, so in fact on this of course the machine will always remain in q0 that is clear but let us say it is 0 1 1 something like this. So think of 0 1 1 1 this string, this should not be in the language because the fourth bit from the right end is 0 and not 1.

So what happens? Let me just go do this, initially the machine is in q0, the 0 came the machine will remain in q0 on the first to 0 and then 1 came on 1 the machine has the choice to go to either q1 or q0, so this is the 1, on the next 1 again it can go to either q0 or q1 which is fine, on q1 so this is the next one that we are talking about.

(Refer Slide Time: 59:09)



(Refer Slide Time: 60:16)



From q1 on 1 it can go to q2, in fact it will go to q2 it has no other choice and then the final 1 came, what is that? What will happen to that? When this final 1 comes it will be either in q0 or in q1, right? From q1 when the 1 comes it will go to q2 and from q2 when the 1 comes it will go to q3, so these are the only 3, only states the machine can be either in q0, q1, q2 or q3 but it will not be in q4, it cannot be in q4 on the input 0 1 1 1. So therefore this string is not a string which can take the machine from its initial state to its final state q4, so therefore it is not in the language.

We will see other examples of NFA's but before we leave this example, we have seen 2 examples of NFA's and we have informally describe the languages accepted by these 2 NFA's with will continue but before we continue and take up further examples of NFA's we will need to understand the conceptual way in which most people understand the working of NFA's and that is what we are going to describe next.