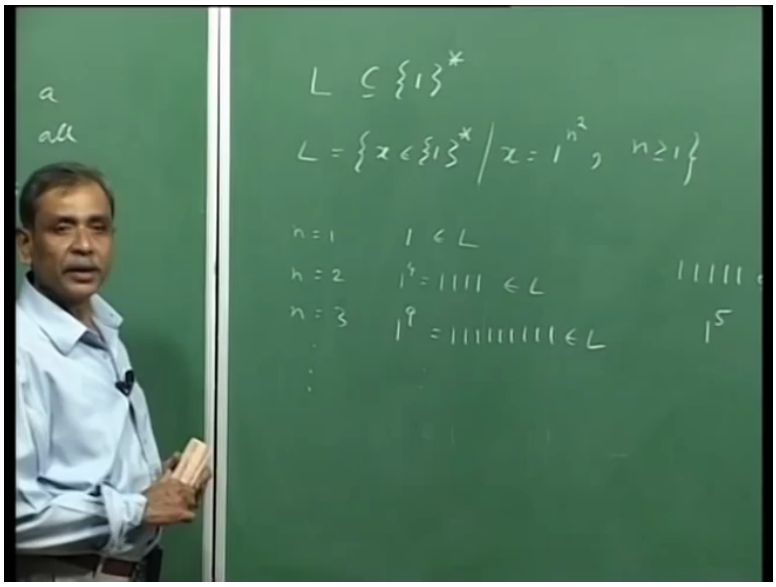
Theory of Computation
Prof. Somenath Biswas
Department of Computer Science and Engineering
Indian Institute of Technology Kanpur
Lecture 6
More examples of non-regular languages, proof of pumping lemma,
Pumping lemma as a game,
Converse of pumping lemma does not hold.

(Refer Slide Time: 0:43)

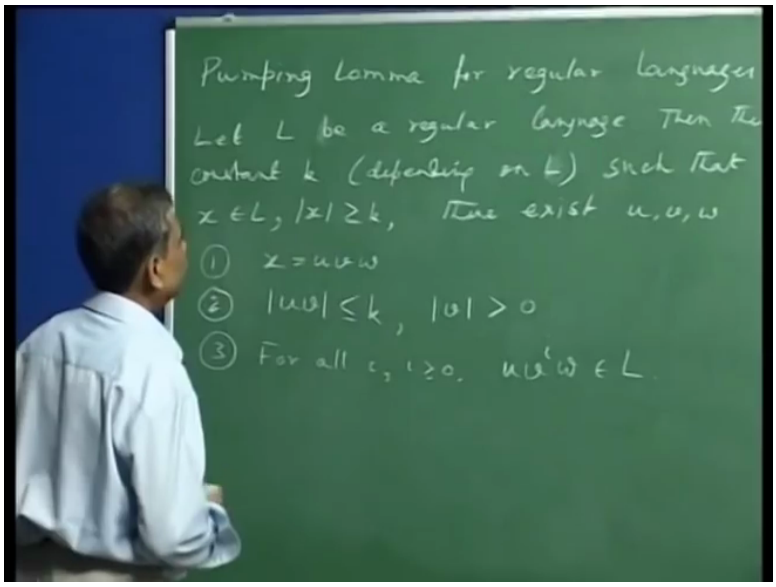


Let us start with another example of a language which is not regular and we will prove by pumping lemma that it is not regular language the language we will consider is this one, firstly this language is a unary language that means the alphabet for the language has only one symbol, so therefore the language in this case is a set of strings each of which, each string is just a bunch of 1's and the our language will satisfy this L is all those strings over the alphabet just one such that x is of the form 1 to the power n square where n is greater than 1.

So basically what we were saying is you can take n equal to 1, n equal to 2, n equal to 3 and so on, if n equal to 1 then the string that you get is 1 to the power n square which is 1, so basically 1, so this is in the language L when n equal to 2 then n square is 4, so there are 1 to the power 4 which is of course 1 (()) (2:20) of 4 1's this is also in the language L and n equal to 3 this is 9, so therefore this is 1 1 1 1 1 1 1 1 1 this thing is also in the language and so on, right?

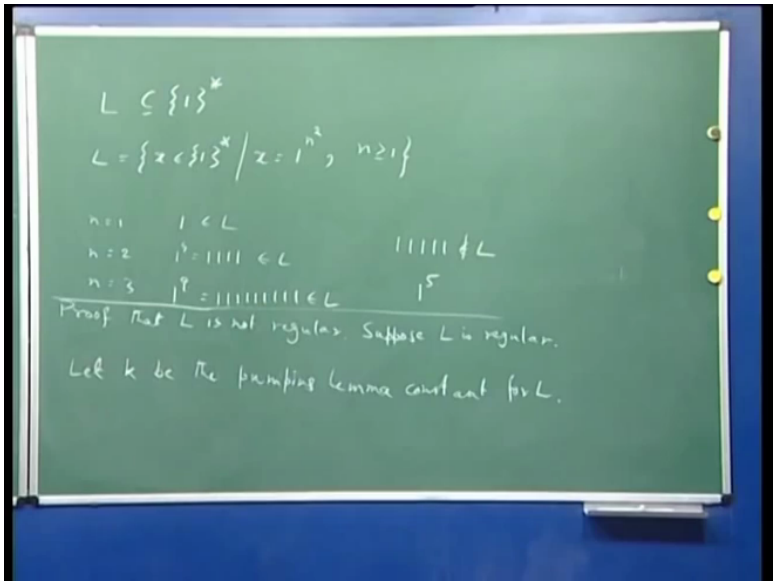
On the other hand this string 1 1 1 1 1 these 5 1's is not in the language, our language L because this is 1power 5 and 5 is not a perfect square, right. We would like to show that this language is not regular and we will use the pumping lemma.

(Refer Slide Time: 3:14)



You recall pumping lemma statement was this which we have gone through 1's and the this is saying in order to use this lemma to prove this particular language is not regular, we need to first of all select somebody will even this lemma says that there is a k, right? So will our proof should be able to handle any k that might be given, so we just keep it as an indeterminate and we say that let k be the.

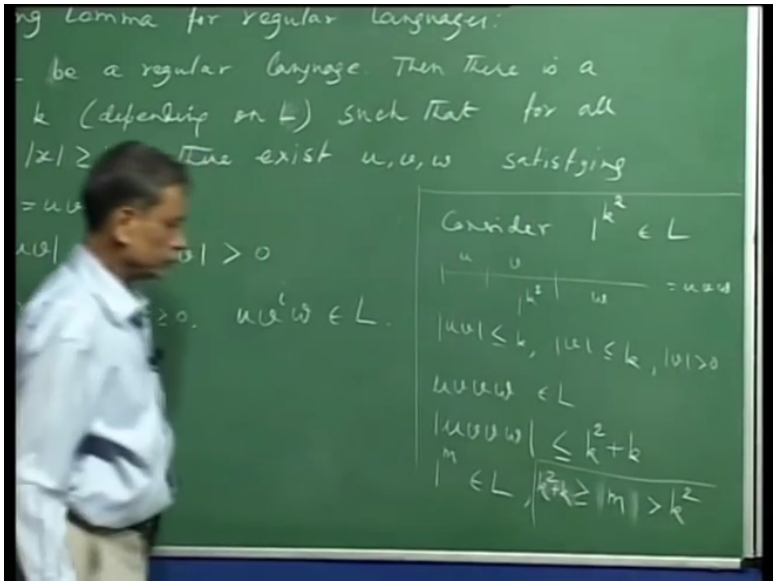
(Refer Slide Time: 4:14)



So we will of course, first we will have to say that our proof we are proving L is not regular and the proof is by contradiction. So we start by saying suppose L is regular and now we can invoke the pumping lemma for this because we have assumed L to be regular and then we derive the contradiction, we say let k be the pumping lemma constant for L. Now we need to select a string in order to invoke non-trivially this lemma some string in the language whose length is greater than k.

In fact it will be convenient to consider 1 k square itself and clearly this string is in the language, whatever be the case. So the pumping lemma constant k we are considering this string 1k to the power k square and now we invoke the pumping lemma, what does the lemma say? That this string 1 to per k square.

(Refer Slide Time: 6:00)

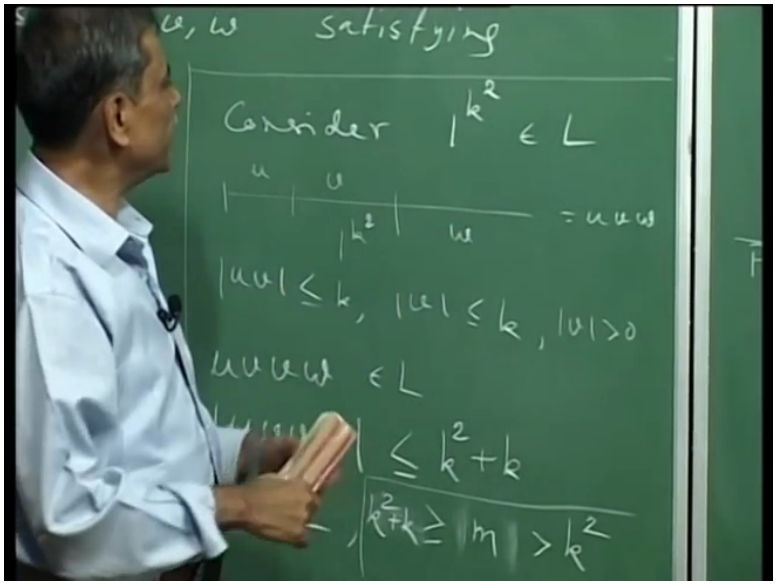


Let us say this is my string 1 to the power k square and there is u, v, w this whole string is u, v, w and this string was of course 1 to the power k square and there is u, v, w this whole string is u, v w and this string was of course 1 to the power k square. Such that u, v, w of course is a string x and u, v together its length is less than equal to k, so suppose this is u, this is v, this is w and my lemma is saying that length of u,v is less than equal to L and from the lemma I get also, so from here I am getting that length of v is less than equal to k but of course length of v is strictly greater than 0 the other condition length of v is strictly greater than 0.

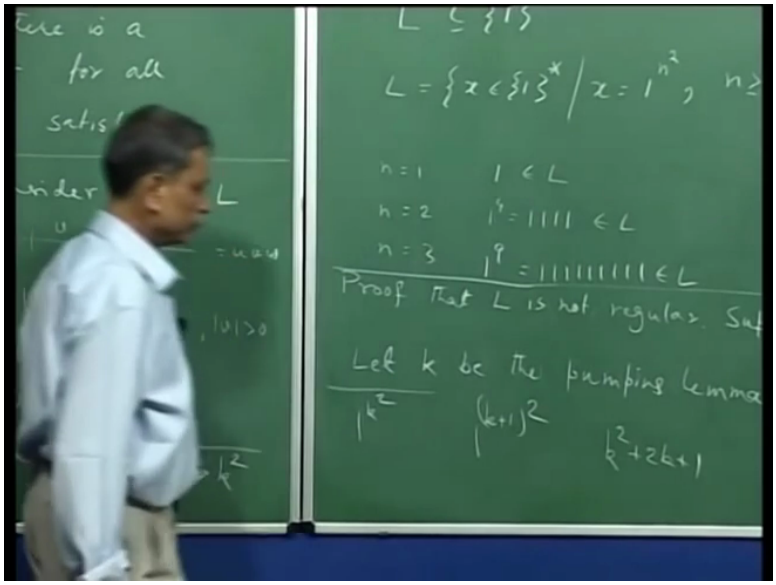
So v is a nonempty string its length maybe 1 or at most it can be k. Now consider the string uvvw this particular string is also in the language that is what the lemma is asserting I have used i equal to 2, so this is in the language L what can we say about the length of this string? An upper bound on the length of the string is uvvw, uvw is of course that length is k square plus we have added a v that length is at most k, so this is bounded by k square plus k.

Actually we have got a contradiction already because this the lemma is saying that some string length is k square plus k bounded by that and consisting only of 1, so in other words, let me write it like this the lemma is saying that 1 to the power m is in the language L where length of m this way is strictly greater than k square or rather the m itself is strictly greater than k square and less than equal to k square plus k. In other words it is saying that there is a string in the language whose length is greater than, strictly greater than 1 to the power k square and that string's length is bounded by k square plus k, can that be?

(Refer Slide Time: 9:42)



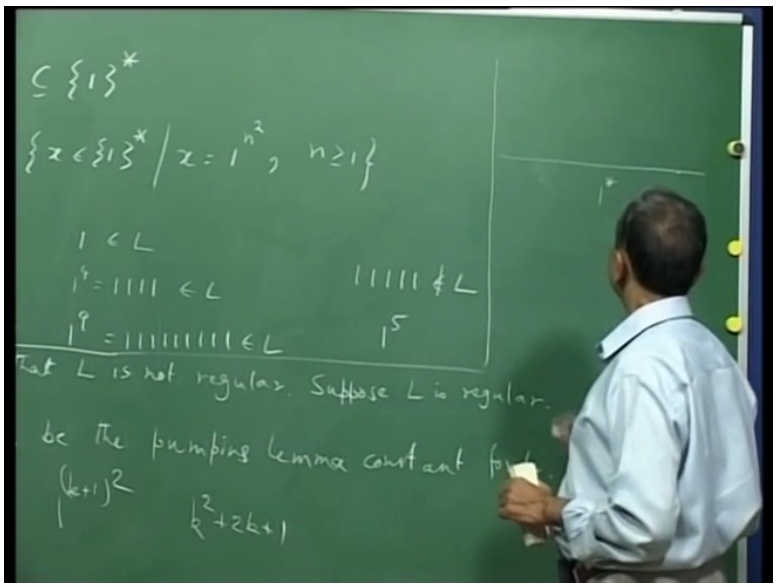
(Refer Slide Time: 10:16)



Because you see what is that string which is in the language and whose length is greater than this string. The very next string after 1 to the power k square will be 1 to the power k plus 1whole square and this is the next string, this is the string in the language of course and the very next string greater than this particular string will have length therefore k square plus 2k plus one.

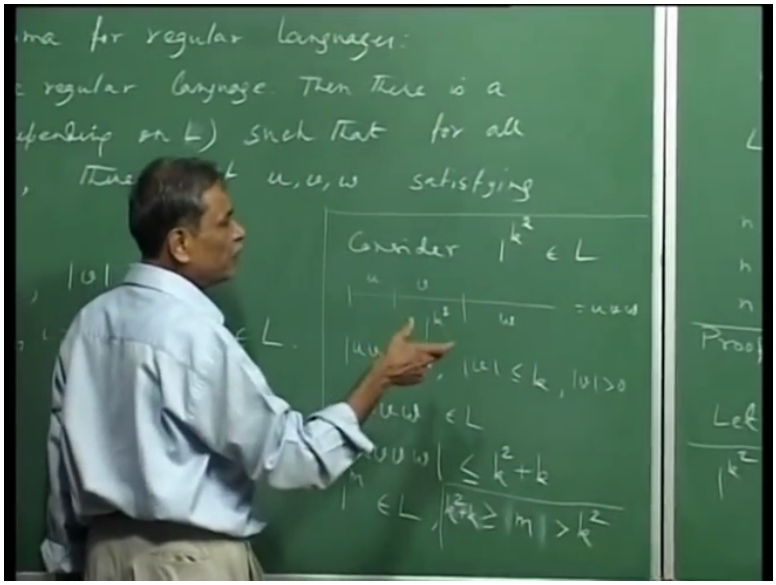
On the other hand the lemma is saying that there is a string whose length is at most k square plus k. So this is strictly less than k square plus 2k plus 1 and therefore you have the contradiction.

(Refer Slide Time: 10:57)

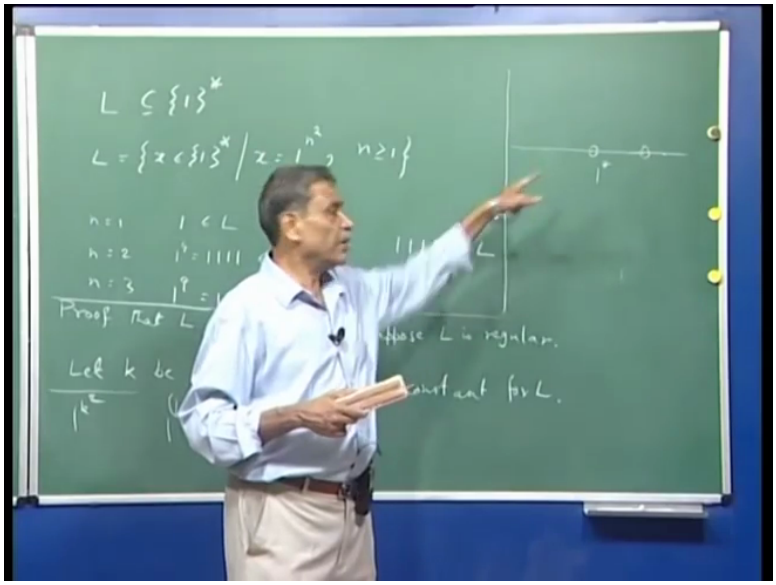


So really speaking what the fact that we are exploiting in this proof that if there is a string in the language supposing this is the set of all strings over the alphabet 1, so this is all the elements of 1 star, if we have a string in the language and the next string comes after a long gap and this gap keeps increasing as we go down, right? Because what is the gap here 2k plus 1 from k it is 2k plus 1, next one will be even bigger and (()) (11:23).

(Refer Slide Time: 11:31)

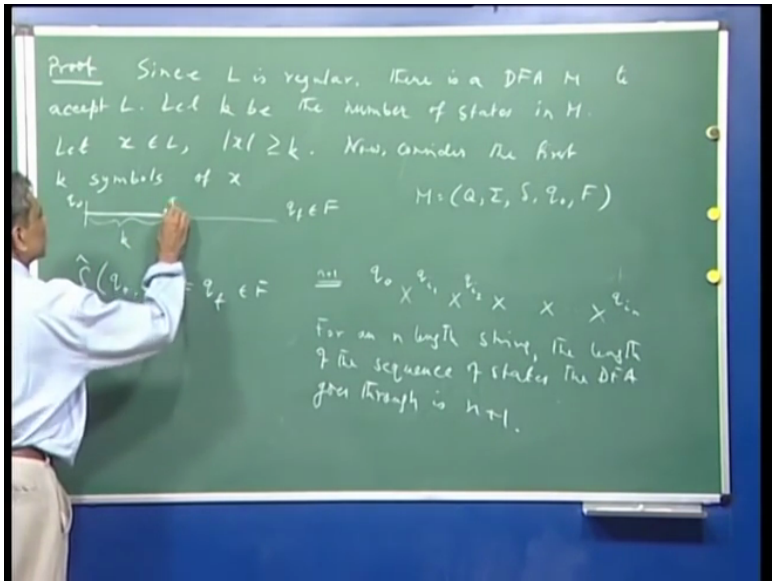


(Refer Slide Time: 11:40)

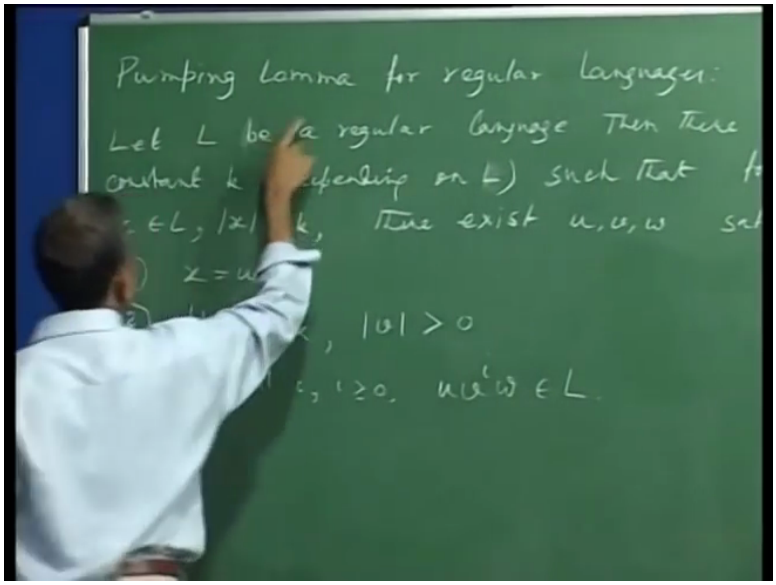


However lemma says that the next string gap is at most k and that is the fact, this particular fact is be contradicted by the implication of the lemma, alright. We have seen couple of examples of how to use the pumping lemma to prove some language to be non regular. Now let us prove the lemma which we have not done yet so far and the proof is fairly simple.

(Refer Slide Time: 12:02)



(Refer Slide Time: 12:05)



Because the moment you say L is regular, see pumping lemma starts from the assumption that L is regular, so we say since L is regular there is a DFA M to accept L, right. In other words L of M is the language L this is a DFA and let k be the number of states in M that means the cardinality of the set of states of M is k. Let x is in L and x, length of x is greater than equal to k, okay.

So this is your x and the length of x is greater than equal to k. Now consider the first k symbols of x. So let us say the first k symbols is this part of the string x, right? So the length of this is exactly k and what do we know of x? That x is a string in the language therefore if

you start the machine M at its initial q0 then the machine finally will be some state which is in the final state, one of the final states of accepting states of the machine M.

If I am using the old notation that M is, the DFA M is 5, the 5 tuple Q, Sigma, Delta, q0, F. So far what we have said is x is accepted by M. In other words Delta hat of q0, x is qf because we are saying that qf is the state it goes to which is an element of F, right? Now recall one thing that we noticed right in the beginning while discussing DFA. Suppose I have a string of symbols, no I do not mean 0's but let me just say these symbols and remember let me write it more carefully.

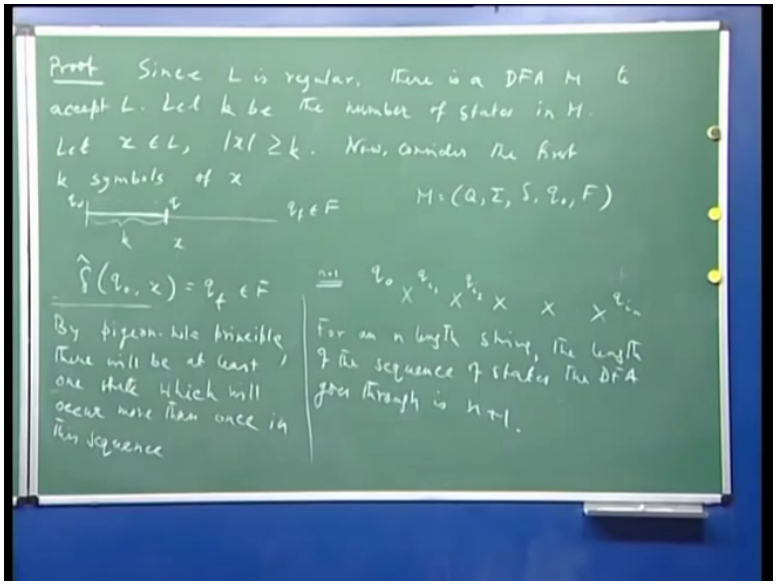
So this is one symbol, this is the next symbol, this is the next symbol, this is the next symbol, this is the next symbol and we consider the sequence of states the machine goes through as it reads the string, so it reads this from state q0 it goes to maybe a state qi1 then the next symbol it goes to qi2 and qin the way I have written that this sequence has, the sequence of states has is of length n plus 1 and what is the?

In this case what is the length of the sequence of symbols, it is n, right? Because on 1, if the string had already one symbol than the sequence of states the machine goes through will have 2 states and so on, you can see supposing it is 2, the string had only 2 symbols than the sequence will be 1, 2, 3 of 3 and so on. So if the string had n symbols than the sequence of states through which the machine goes that will have is precisely n plus 1 its length will be n plus 1.

The sequence of strings, sequence of states for an n length, so let me write it clearly that for an n length string the length of the sequence of states the DFA goes through is n plus 1, okay. Now this part of the string we said is exactly of length k, now what does it mean? The sequence of states through which the machine will go on this part of the string which is of length k this sequence is of length k plus 1 from the fact that we have written.

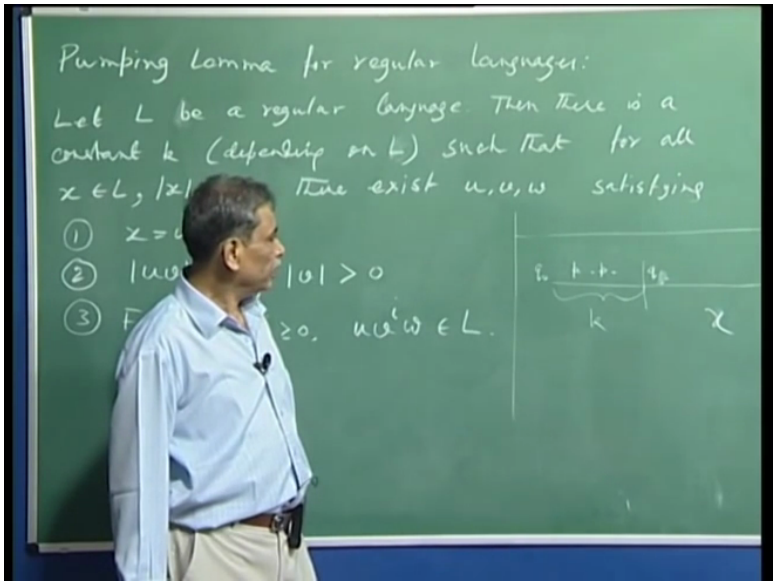
Now this is the most crucial observation that the sequence of states is a length k plus 1 and brutally how many states are there k because that is what we have assumed, let k be the number of states in M and sequence as one more than the total number of states, one more the length of the sequence is one more than the total number of states in the machine.

(Refer Slide Time: 19:52)



Therefore by pigeonhole principle there will be at least one state which will occur more than once in this sequence, which is this sequence we are talking about? The sequence of states the machine goes through on the first k symbols of the string x, right?

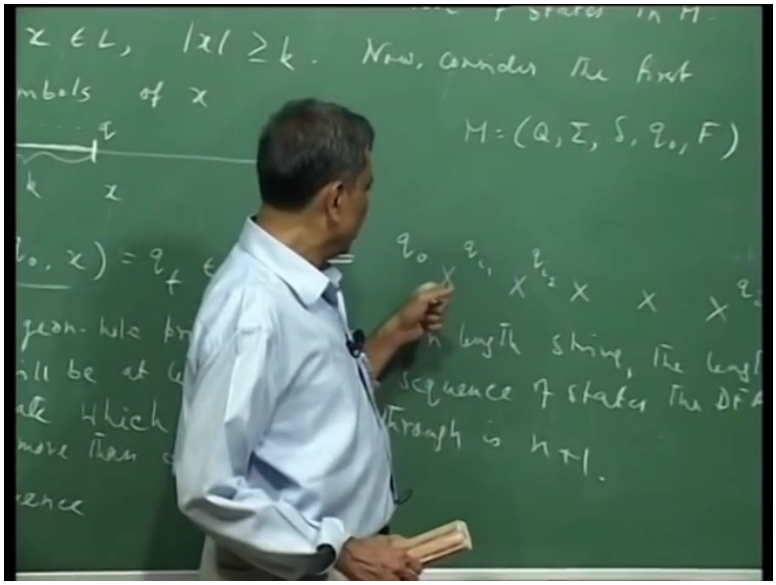
(Refer Slide Time: 21:00)



So in fact let me see if we can picture, some kind of picture that this is your string x, right? And this is the sequence of states, now we are talking of the first k symbols, the string consisting substring the first prefix, prefix of the string x of length k and here it is the sequence of states that we are talking of q0, sorry let us say q here finally of course it goes to Qf.

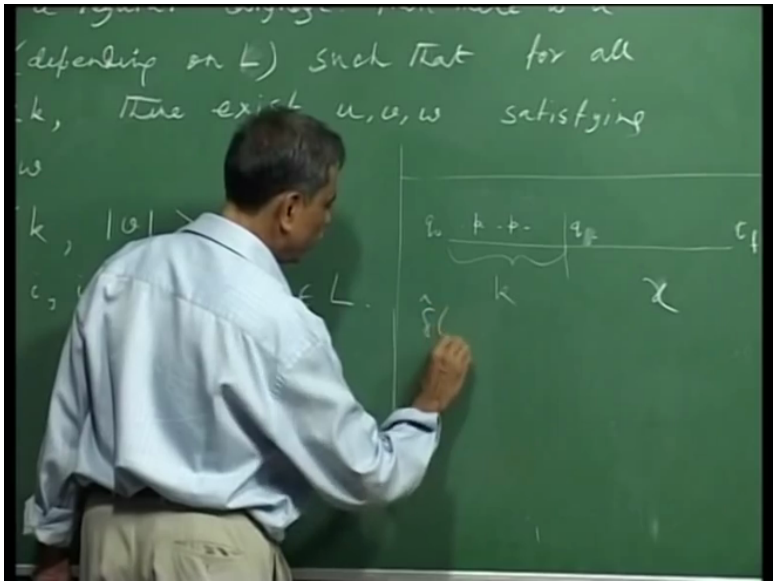
Now q0 to some q this sequence of states there must be some state which repeats at least once you can see this is a very simple use of the pigeonhole principle that we learn in the (()) (21:55) maths.

(Refer Slide Time: 22:07)

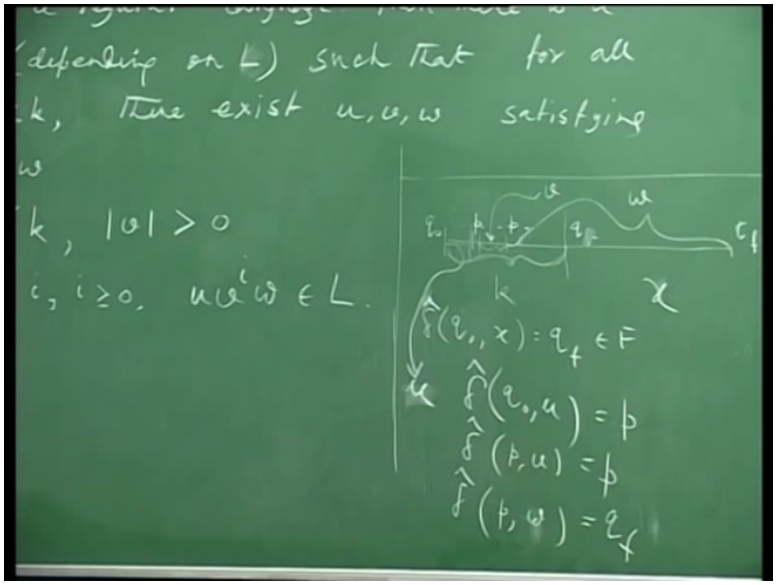


Now again if you look at this you can see between 2 states even if they are adjacent there will be at least one symbol, between 2 states in this sequence between any 2 states in this sequence will be at least one symbol even if the 2 states are adjacent like in this case.

(Refer Slide Time: 22:46)



(Refer Slide Time: 23:56)



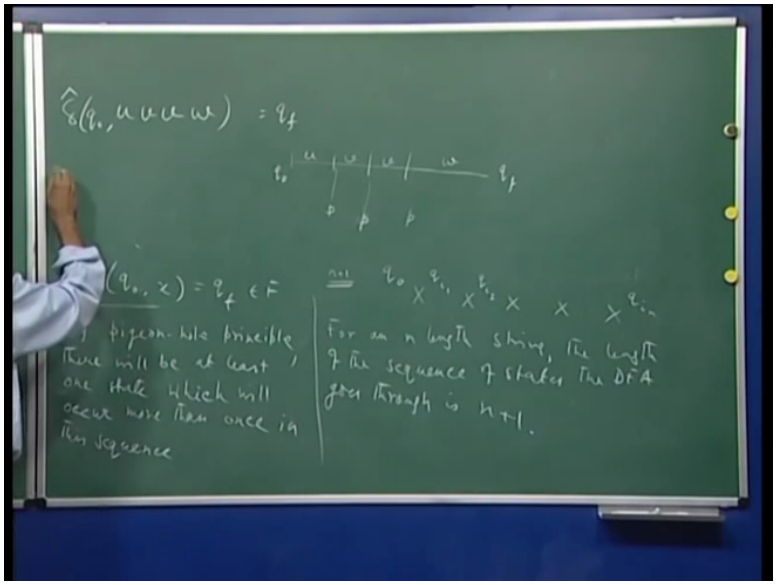
So what I can say is the point. Let me say, let us focus on this picture and what 2 things that we already know? That Delta hat of q0, x is qf which is in the one of the final states and let me call this to be this part, the part of the string which brings me from q0, brings the DFA from q0 to p which is this first occurrence of the state which occurs more than once this part let me call u, right?

And in state p from, right? The very from the very next symbol to this part, what is this part of the string doing? It is taking the machine from the state p back to p. So let me write it in this way that Delta hat of, this is no, this is Delta hat, Delta hat of q0, u what was u? u was

the part of this string the initial part of the string which took the machine from the initial state to the state p which is going to occur once more.

Delta hat to q0 to u is this p and this part let me call v, so Delta hat of p to v is also p, right? Because this v part, this u part took the machine from q0 to p, v part is taking the machine from p to p, right? And now you do not care about anything else but let me call all the rest of the string after v whatever is left in x, let me call it w. So what can I say about w? P, w will take the machine from the state p to qx, right, okay.

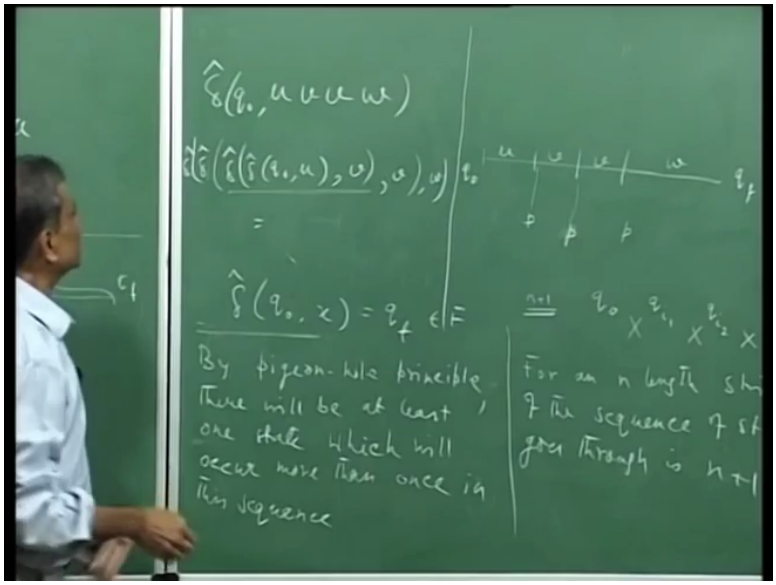
(Refer Slide Time: 25:53)



Now the rest of the proof is a simple application of these facts. Now what will happen on the string? u let us say vvw, what can we say the behaviour of the machine from q0 to this, now let us look at this we will just use these facts, so q0 I mean it is easy to see what is going to happen?q0 this is, we start from q0 first we go to look at the part of the string which is u the prefix of the string which is u that takes the machine, if this is your u then this is v, now we have another copy of v and then finally w.

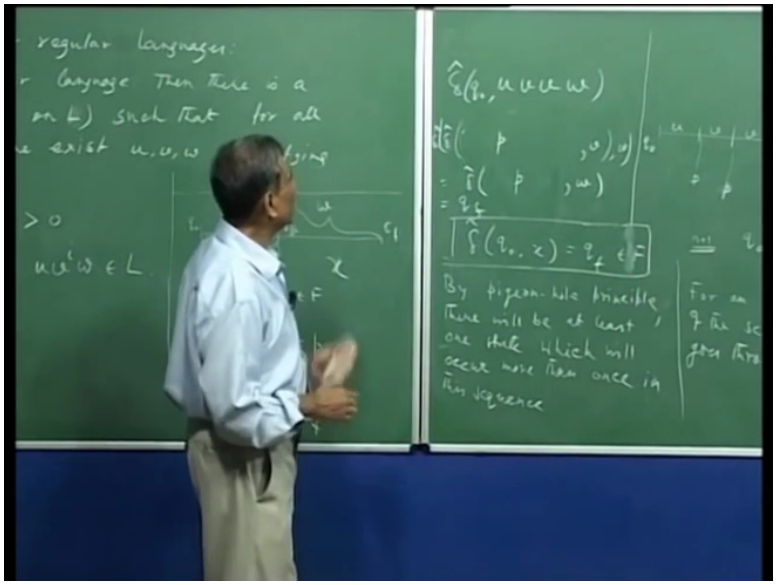
Now we have said the machine takes, machine goes from q0 to p on u, machine goes from p to p on v. Now another copy of v is there but machine cannot, there is no reason for machine to behave differently because this is a deterministic machine, so it sees v, so it goes again took p, in other words you see this is also going to be qf because now from p on w it will go to qf.

(Refer Slide Time: 27:43)



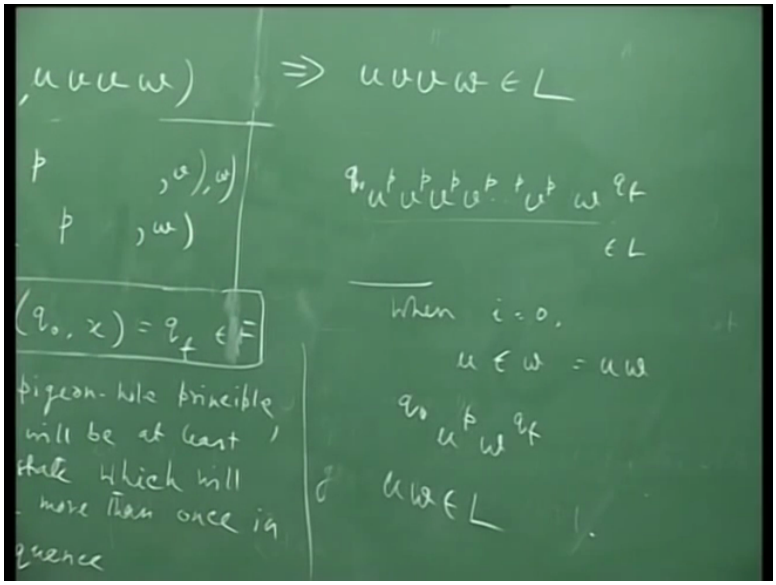
You can you can write it like this way that I can say that this is, what is this? Delta hat of q0u this is the state the machine goes to after seeing the first part u, now it sees v, now again it sees, so this is the state the machine will be after it sees u and v. Now it sees another copy of v but this whole thing is one state and now it sees v, so Delta hat this is going to be the state the machine is going to be after seeing uvv and then finally it has w, you will put another delta hat, right?

(Refer Slide Time: 28:53)



This is, what is this? Now I can I can argue it out using this facts, Delta hat of q0, u was p. So let me substitute p here then delta hat of p of p and v is p, so this whole thing is reduces to again p, so let me write it p then delta hat of p of v is again p, so you see this is let me write it clearly delta hat of p, v, w and this is delta hat of p of v is again p, delta hat of p of v is p. So this becomes p, Delta hat of p, w is qL, so this is what I had asserted earlier and this is now I have shown you, qf is of course an element of F.

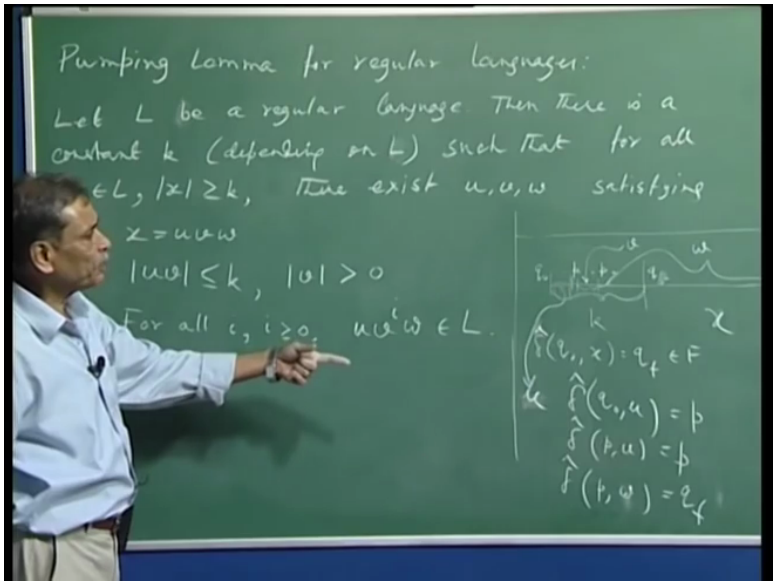
(Refer Slide Time: 30:30)



Now what does this mean? That since uvvw takes the machine from q0 to qf therefore what I have is that uvvw is also in the language L, right? And this argument will be true, if you have not just one extra copy but if you put any number of these, so let us say I had uvvvw, u is going to take q0 the machine from the state q0 to p according to this, v is going to take the machine from p to p, this v will also take the machine from p to p, this p to p, p to p and w will take the machine to qf. So therefore this string is also in the language L.

We have proved for all i greater than equal to 1, what happens when i equal to 0? That means we are talking of the string uw, when i equal to 0 that means we are talking of the string uv0 w, v0 by definition, v to the power 0 by definition is you recall in the beginning we talked of this is the empty string u Epsilon which is uw, so now let us look at uw the machine initially the state q0, q0 starting from q0 on u it will go to the state p, p on w it will go to qf. So again uw is in the language seen language L.

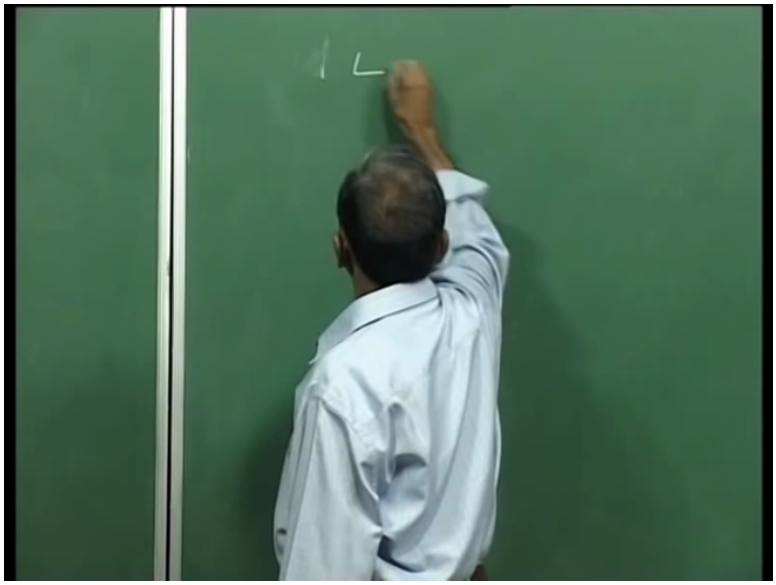
(Refer Slide Time: 32:40)



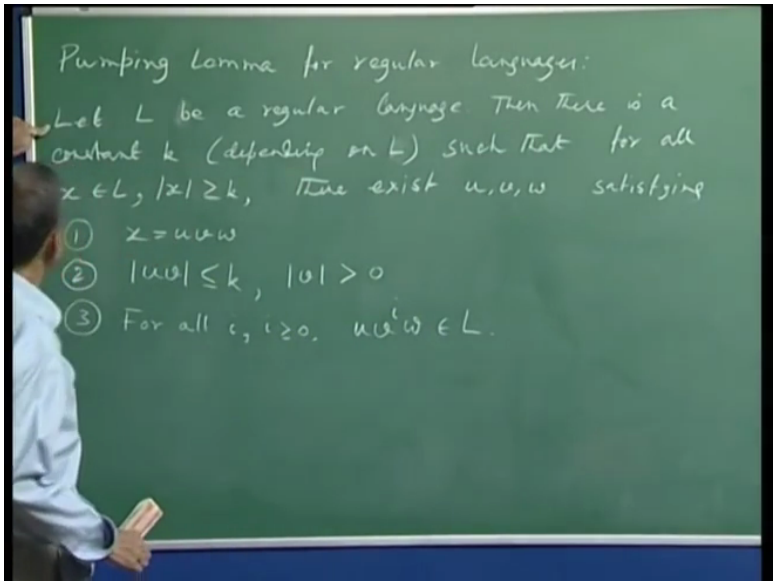
So therefore for all i as the lemma states greater than equal to 0, the string uv to the power i w is in the language L. If you want to be very formal you do this proof using induction ni and that is easy. Basically you will try to show that for all i you want to prove this using induction on i start by saying suppose i equal to 0 then we are talking about the string uw using this facts we know that uw then assume in that induction prove it is going to be that suppose this statement is proved for all i up to some n and then you would prove by induction.

I mean your induction step that uv to the power n plus 1w also will be in the language but the idea of the proof is very clear, this is how we will go about. Now several things we need to say about this lemma and we will say these things because sometimes we do not fully understand the content of this lemma or what it is saying exactly, we make mistakes in applying it.

(Refer Slide Time: 34:39)

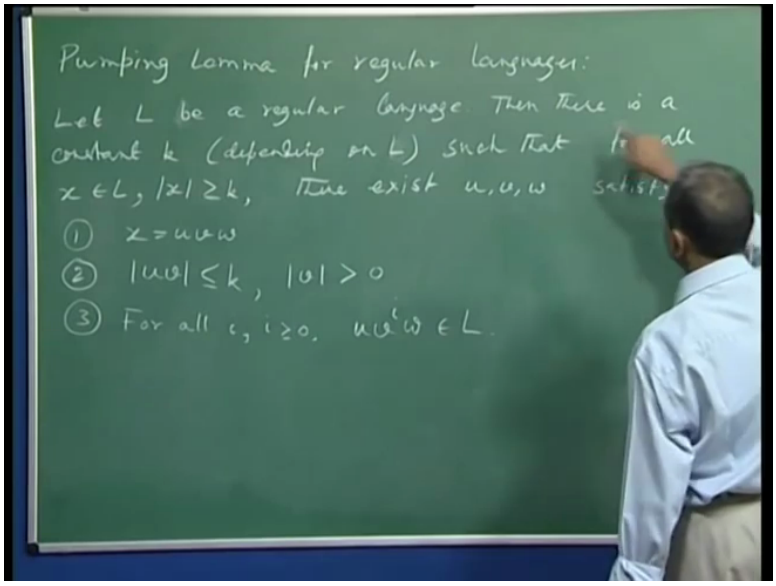


(Refer Slide Time: 34:55)

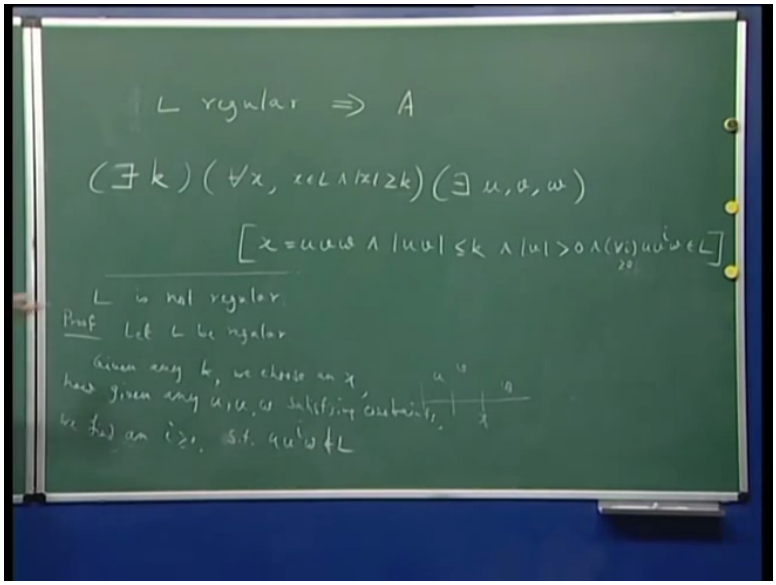


Now let us look at the logic way of looking at the statement, what is this assertion? You know it is saying first of all at the top-level, this lemma is of the form L regular implies certain things, so let me call that all these things that it implies. So you know we are saying the statement goes like this let L be regular then, so L is regular implies this whole thing then etc, so this part I am calling it A.

(Refer Slide Time: 35:17)



(Refer Slide Time: 35:26)



Now look at A, the statement A, what is A saying? A is basically this statement we start with this there is a constant k. There is constant k which depends on the language, so once you have fixed L, this is a constant k there is a constant k then it is saying such that for all x, right? x is in L and length of x is greater than k, after that what we are saying?

Now we are saying there exist u, v, w there exist u, v, w and now we are asserting certain things about the relationship between u, v, w and x and certain conditions on k and then we are saying certain other strings also will be there in the language. So let us write it in the

formalism of logic, it is saying that the string x is uvw that is the first thing we said then we are saying that we use this notation and already used it here.

Then we are saying that length of uv is less than equal to k and our next conjunct is length of v is greater than 0 and finally we are saying for all i uviw is also in the language L, for all i greater than equal to 0, right? This is the formalism of logic I have just rewritten that. The purpose for doing this is, clearly tell you when you were giving a proof use to show a particular language to be not regular.

It is very very necessary to keep these things (()) (37:56) intuitively in our mind. Suppose we use to show using this lemma that some language L is not regular, what do we do? We will first assume in our proof, the structure of the proof is what I am trying to tell you, we will first assume that let L be regular. The moment you say L is regular then this statement is true then that statement says there is a k.

Now in my proof, can I say for example? That K is 590 I cannot do that, why? Because this is not in the hand of the prover, there is this k therefore this k should be, my proof should be ready to use any k which is given to, right? Sometimes these kinds of proofs that can be seen as a game between 2 people, one is an adversary his goal is to prove you wrong and your goal is to prove L is not regular.

So what you are going to do? You your adversary is going to give you a k then you will pick up an x, right? So in that game the adversary moves are, adversary starts with this existential quantifier move, so he picks any k, so you should be ready in your prove to use any k whatever might be given but then it is your turn in the game you will take some language because this part is saying for all x therefore we have complete freedom to choose any x, so long that x is in the language and its length is greater than equal to k, right?

So the discovery part of the proof is to come up with some proper x given any k, right? We choose an x which is which will satisfy this. Now what is our goal? Is to come up with finally on pumping a string which is not in the language, by now you have seen the kind of proves that we have seen it may making use of the pumping lemma, so we choose x, now this existential quantifier.

So sometimes beginners they choose an x and then they say let u be this, v be this, w be this, that will be long that proof is wrong, what you have to now be prepared to carry out your

argument for any breakup of u, v, w of course which will be given by the adversary but adversary has to obey these, that uvw together is the string x and length of uv is less than equal to k and the length of v is strictly greater than 0.

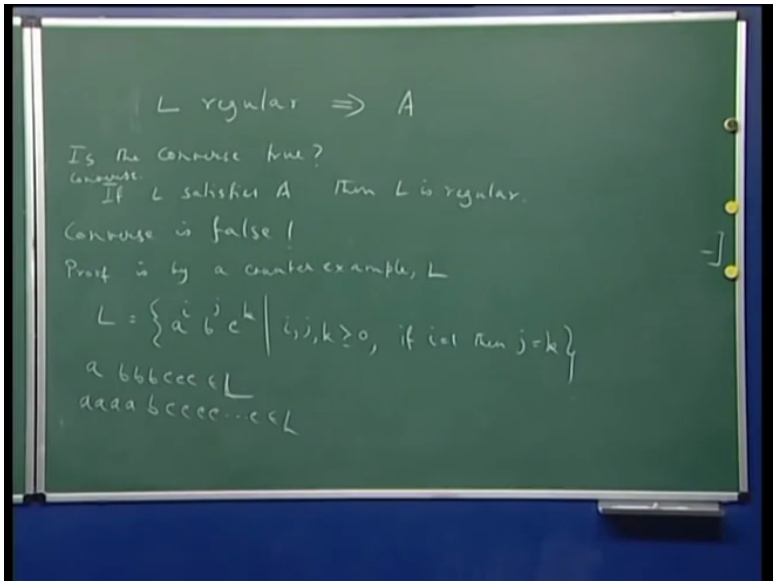
So once you have chosen x then what is the next part of the proof? So this is your x then for any breakup of uvw of the string x which obeys the constraints these 2 constraints, the total length of uv is less than equal to k and v is nonempty, but should be able to find an i, so adversary gives you uv and w. So we choose an x, now given any uvw satisfying constraints I have already written it.

You see now it is in my hand to choose an i, we find an i and to derive the contradiction this i should be such that uviw is not in the language and then therefore I win the game. So in that model of adversary versus the prover game adversary chooses k, the prover chooses some appropriate x in the language.

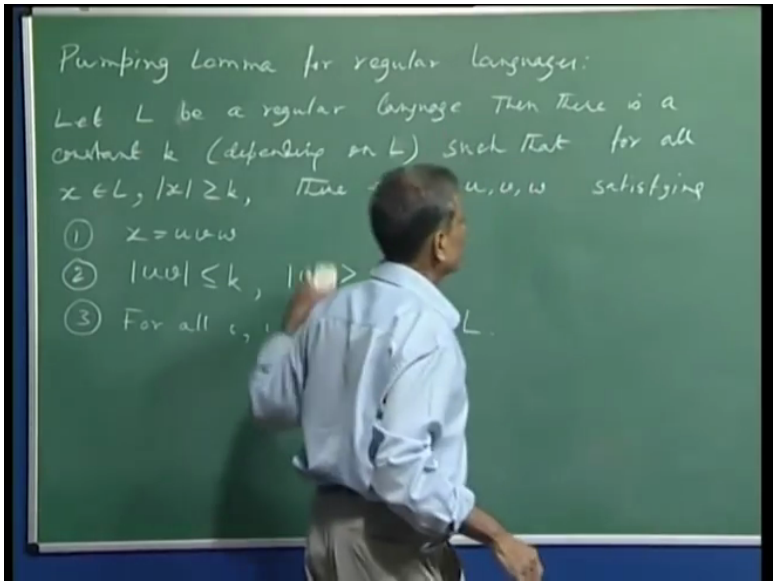
Now once that x is chosen adversary gives a breakup of x in terms of uvw satisfying these constraints then prover chooses an i and he manages to win the game, if he finds uviw which is not in the language because you got a contradiction because uviw according to the lemma it has to be in the language but by definition our i is such that, you have seen these examples that on pumping I am getting a string which the definition of the language tells me could not be in the language.

So therefore I get the contradiction and therefore the supposition L is regular brings me to a contradiction and therefore I conclude that L is not regular. So this is the overall structure of the proof. I want to emphasize again in these proofs when we use pumping lemma I should be able to use given any k then I choose an appropriate x then for any breakup of uvw of that x. I should be able to find out an i such that uviw is not in the language which I would like to show is not regular that is the first thing I said. I wanted to make a point the structure of the proof making use of the pumping lemma.

(Refer Slide Time: 46:17)



(Refer Slide Time: 46:33)



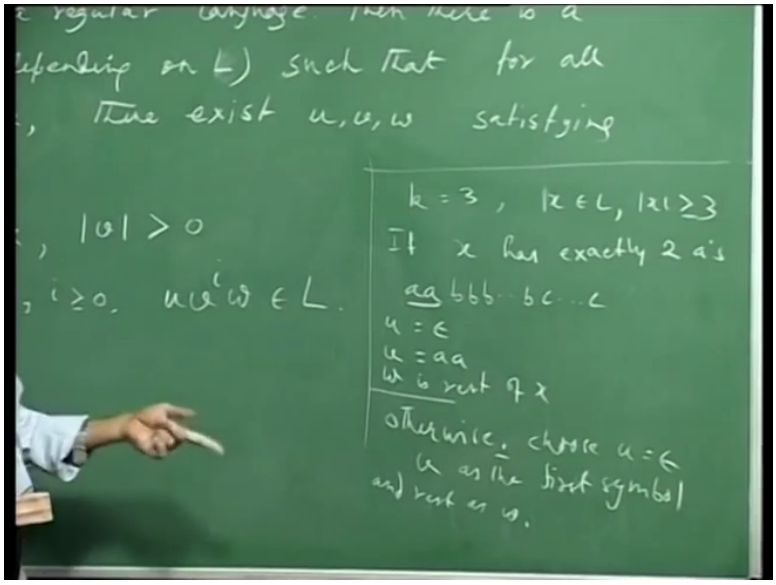
The second thing comes from here, just the top statement. The primary lemma says that if L is regular then certain conditions will be true, so a natural question is, is the converse true? So what is the converse? If L satisfies A, A is remember this part of the lemma, if L satisfies A then L is regular this is the converse, right? Actually it is not very difficult to see the converse is false and how do I show that converse is false? I actually give a counterexample in which the language A, language will satisfy these conditions A and yet the language is not regular.

So proof is by a counterexample and that counterexample language L, that L is defined to be ai, bj, ck, first thing is i, j, k these are all greater than equal to 0 and now the interesting part

the constraint is, if i is equal to 1 then j is equal to k. So for example the string a bbb ccc is in the language L, why? Because it is of the form, some bunch of a is followed by bunch of b's followed by bunch of c's and there is exactly one a, right? i equal to 1 then the number of b's is equal to the number of c's, so this is in the language.

Now on the other hand if I had a bunch of a's, more than one a's then there is no really constraints on the length of b's and c's, right? I could have had one b and maybe many c's this will be also in the language, okay. It is not too difficult to see what this language is and I would like to show this language satisfies all these which came in the you know, remember we are looking at from the "Then part", in fact let me say there is a, it says there is a constant k.

(Refer Slide Time: 49:55)



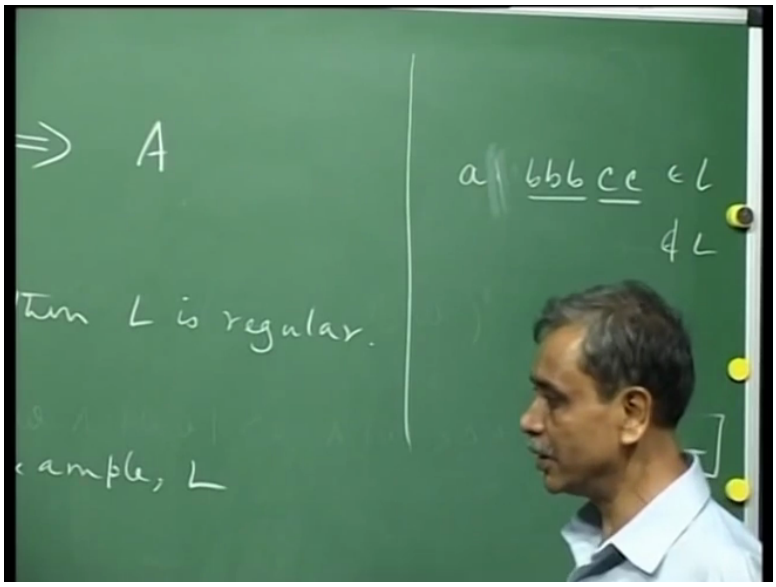
Let me just say the k is 3 and you take any string which is in the language whose length is 3 or more then what should be the uvw if I assert such that for all i, uviw is in the language and that rule is kind of simple also.

That if, so basically consider an x in the language and length of x is greater than equal to 3 and now if x has exactly 2 ways, right? So aa then I have a number of b's, according to since it is in the language L, if you see that there is no constraints on how many b's, how many c's should be there. So we have a number of, these number of c's then what I do is, in this case u is empty Epsilon, v is this part a and w is rest of x.

Just let us verify that on pumping we are not going out of the language because if I pump down that means I will take out these both a's because v will be taken out then I have just a bunch of b's, bunch of c's which is okay, there is no relation on the lengths of b's and c's, since i is not equal to 1 there need not be any relation. On the other hand if you pump any number of times you will get more and more a's but again you will not get a single a, which is when a constraint comes.

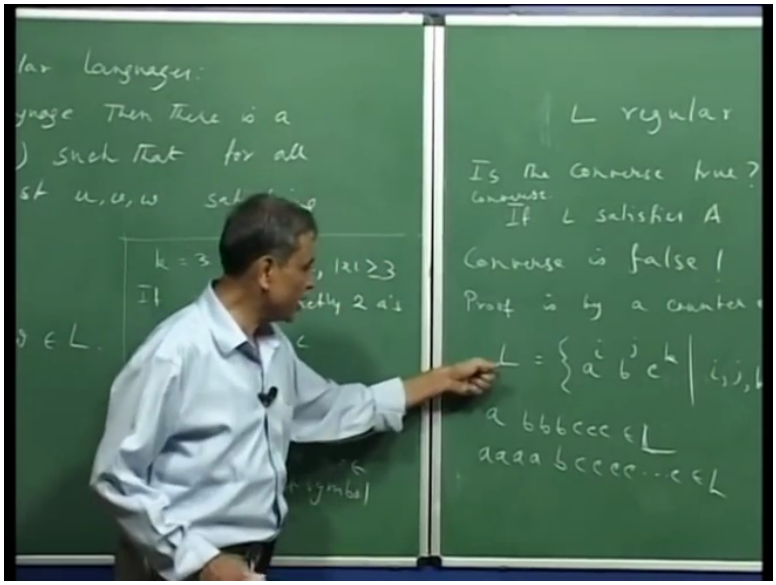
So it is easy to see if I choose v as this part both the a's then we can pump any amount of time and this results in string still will be in the language. If x has exactly 2 ways then this is the situation otherwise choose u as Epsilon, v as the first symbol of x and rest as, you should be able to verify that even in this case on pumping v it is just one symbol nothing is going to happen.

(Refer Slide Time: 53:27)

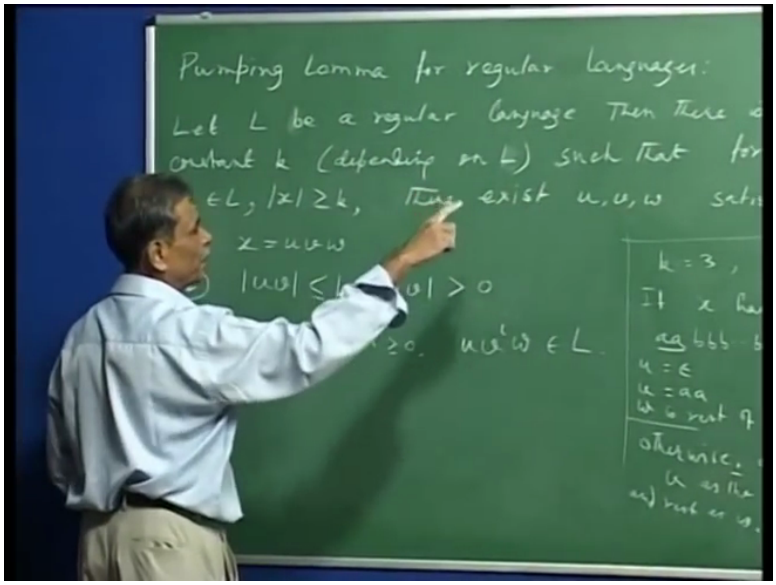


Now, do you see what was the point of, here the problem would be let us say with a string like this I had 2 a's and here I had 3 b's and 2 c's this will be in the language L, according to the definition this is in the language L and now if v was only one a for example then when I pump down I will have one a but now I have length of b's and c's do not match and therefore I will get a string which is not in the language L and thereby the condition a will not be satisfied, right?

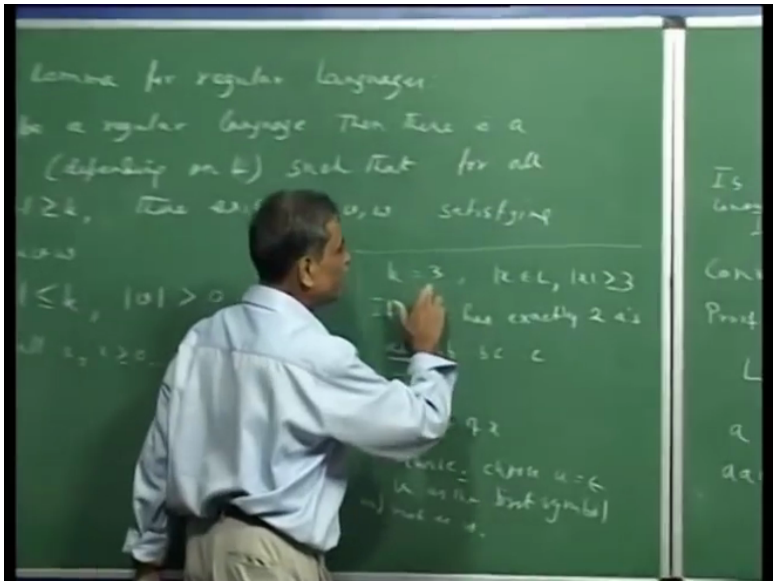
(Refer Slide Time: 54:26)



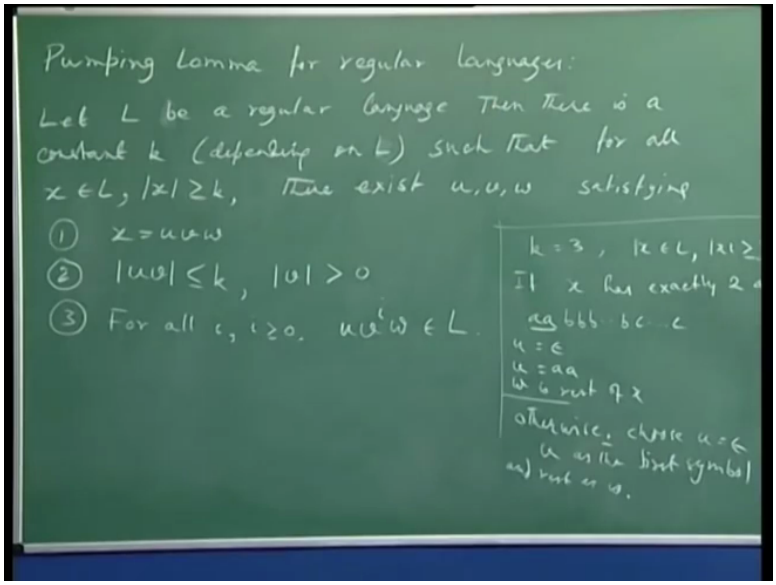
(Refer Slide Time: 54:29)



(Refer Slide Time: 54:33)



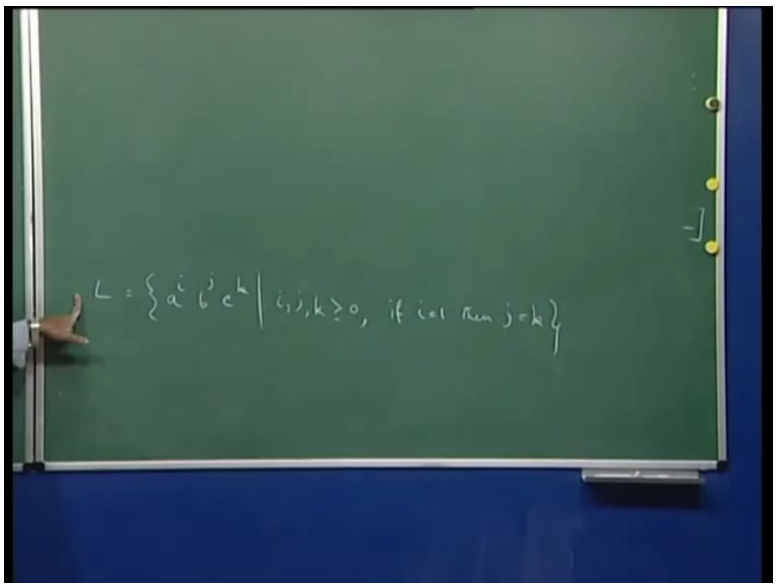
(Refer Slide Time: 55:24)



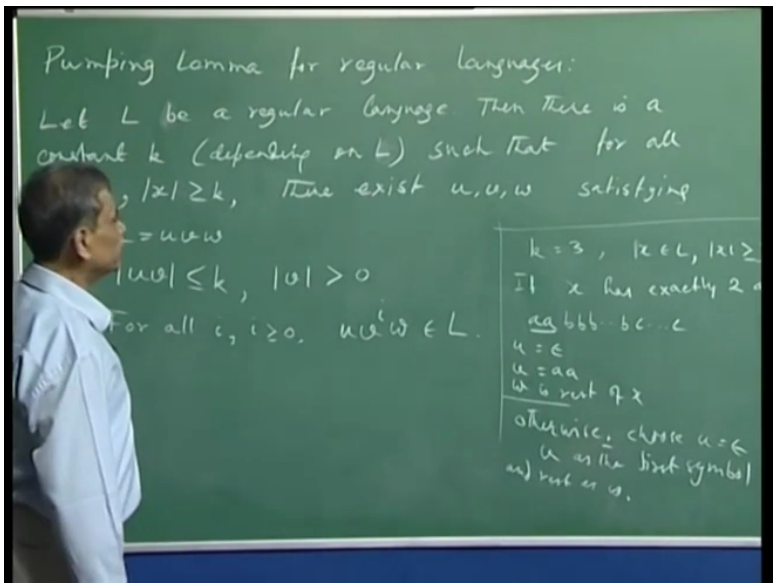
So overall conclusion is, for this language L the conclusion of pumping lemma holds however the language itself is not regular. So the converse does not therefore holds, here is the language L which satisfies all these things there is a constant k we took that k as 3 then we say take any string in the language whose length is 3 or more and then given such a string in the language we can break it up in uvw such that on pumping v we do not go out of the language therefore this conclusion of pumping lemma holds and yet the language L is not regular.

Actually how do I show that this particular language is not regular? I stated that this language L is not regular, how do I show? Because now I know that I cannot use this pumping lemma to prove this particular language to be not regular, why? Because here is a k which is 3 in fact for which you know the conclusion of the pumping lemma holds, so we do not get into any contradiction.

(Refer Slide Time: 56:15)



(Refer Slide Time: 56:24)



Now we can prove this to be not regular by using a slight generalisation of this lemma. Let me state the lemma, here is the language L which happens to be not regular but I cannot

prove that it is not regular by using this pumping lemma and I just showed you that pumping lemma is not going to give me any contradiction but then how do we show this L is not regular? We can show by means of a slight generalisation of this lemma which will consider next.