Formal Languages and Automata Theory
Prof. Dr. K. V. Krishna
Department of Mathematics
Indian Institute of Technology, Guwahati

Module - 7
Properties of Regular Languages
Lecture - 3
Pumping Lemma

We have discussed some properties of regular languages, particularly we have concentrated under closure properties. That is the set of regular languages that closed with respect to certain operations like union concatenation and cleanly star that is from the definition. Other than that we have observed some of the operations like homomorphism, substitution, inverse homomorphism and quotient reversal.

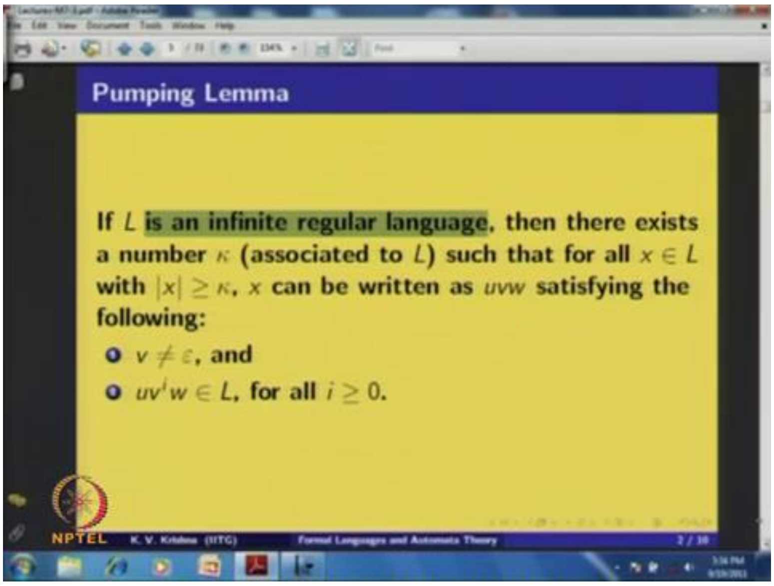
So, several other properties that we have looked into and understood that the class of regular languages is closed with respect to those operations. And, these properties are very essential in order to understand, new regular languages to see that they are regular. Thus, you may see the new language that we are going to observe it is regular, you may apply these operations on the known regular languages to come up with the these things.

Now, in this lecture I concentrate on talking about non regularity; that means, to understand the given language is non regular. So far, we have one tool called the main legner theorem that is actually characteristic property of regular languages, so what we have observed there, you know can be whatever regular languages are corresponding that right invariant equational relation, it is a finite index.

So, we have observed that the point and through that relation right invariant equational relation. We have to understand, what is the number of sequence classes and you see that the index, if it is finite we say it is regular, if it is not finite we say it is non regular. But, that particular result is of course, very fantastic result, but for practical purposes sometimes given a language to understand the number of equivalence facility, we have to little more you know we have to work and really as certain that, whether it is regular or non regularity this kind off.

So, in order to get this property that non regularity, we now look at it very practical and very nice tool, now they called pumping lemma that we will discuss in this lecture. So first, the essentially you know this is a property of the regular languages, so given a regular language, I purpose this pumping lemma and see how this can be used to understand on non regularity.

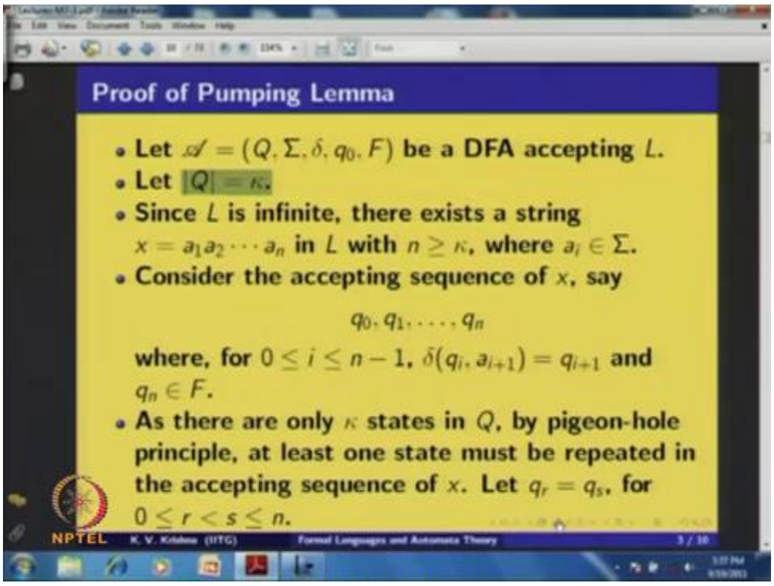
(Refer Slide Time: 03:00)



First, let me state the lemma, if L is a finite regular language, then there exists a number k associated to L. Of course such that for all x in L, whose length is bigger than k that string can be written as UVW satisfying the following condition. So, one condition here is the middle portion, if we have partition that as UVW, the middle portion is non empty, and for all i greater than or equal to 0, UV power i W, all these strings an finitely strings they are all be in L.

So, this is what is pumping lemma what I am setting here is for any infinite regular language, this property holds. First, let us look at the proof of this result, then we will discuss about the applications of those things.

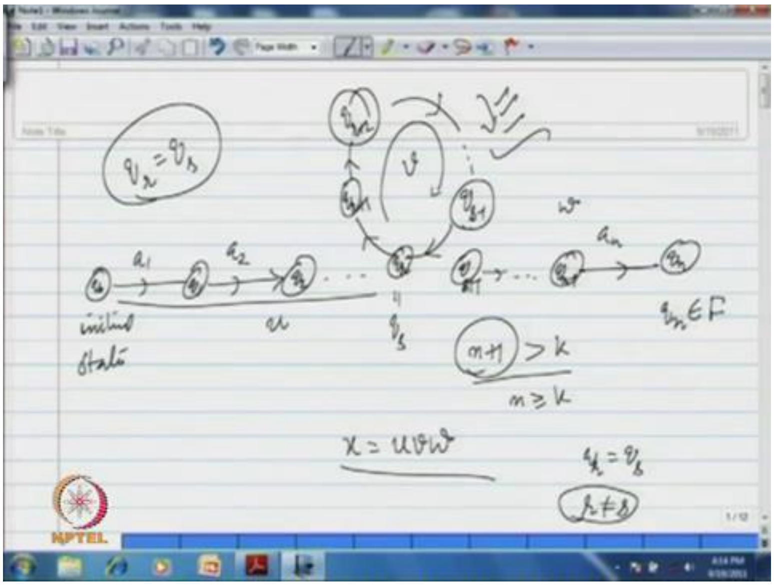
(Refer Slide Time: 04:15)



Since, the given language is regular, you can have a DFA, let me say A Q sigma, delta q naught, F have a DFA accepting L. Assume the kappa k, whatever you see that the number of states you set it as k, since L is infinite, you can always find the string a 1, a 2, a n, let me call say x, a 1, a 2, a n in L, whose length is bigger than or equal to k. Here, the language is infinite I am using because, since it is infinite language, you can always find a string whose length is bigger than or equal to k.

Now, this string since it is in L, we will look at the accepting sequence, let me call the intermediate states to reach to the final state, like the accepting sequence q naught q 1, q n then you apply this string a 1,a 2, a n in the DFA.

(Refer Slide Time: 05:33)



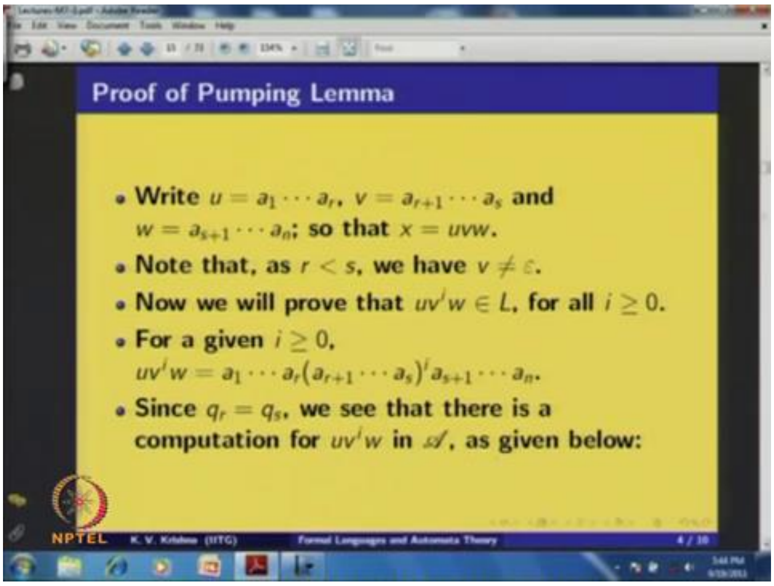
So, that is you can have this kind of situation, in q naught you are applying a 1 to come to q 1, and then in q 1 you apply a 2 to come to q 2 and so on. By the time you apply a n, you are reaching to the state q n and since this string is accepted by the given DFA, this is initial state, and this is a final state, q n is an element of F, so this is the situation here.

Now, as there only k states in Q, thus number of states we have assumed that is k. By pigeonhole principle, at least one state must be repeated in the accepting sequence of x; that means, you see here, the number of states I have written, because here a 1, a 2, a n when I am supplying as an input, I can clearly see here exactly n plus 1 states, so this n plus 1 is bigger than k, because n is greater than or equal to k.

So, you can see that by using pigeonhole principle among these states at least one state should be repeated there, because there only k distinct states here. Now, you notice here that, here there are n plus 1 states and that is bigger than k, and hence by pigeonhole principle at least one state should be repeated in this sequence. Now, let me assume that q r is equal to q s for r and s, they are between 0 and n, and r is less than s.

So that means, the picture now will be looking like this is q r, let me say and this is equal to q s, now the q r plus 1 and q r, because these are coinciding q r plus 2 and so on. Now, this is q s minus 1, this is q s again, this is same as q s, because q r is equal to q s, and now of course, q s plus 1 is here and so on, so the situation is like this, now you have this kind of picture on the accepting sequence.

(Refer Slide Time: 08:19)



Now, because of this, I have the facility to break this string in the following form, that is the string from here to this situation. This place q r state I will call it as u, and the string that is actually in this loop I called it as v, and the remaining string I let us write it as w; that means, a 1, a 2, a r till a r, let me call it as U, a r plus 1 a s that is V, and a s plus 1 to a n, let us write it as W.

Thus x is clearly partition as uvw and more over, here this string V is non empty because q r is equal to q s, but r is different from s. Because of this at least some string should go in this loop and you observe that V is not empty string, now we will prove that, UV power i W for all i greater than or equal to 0 is a string in L. First, for given i greater than or equal to 0, UV power i W, that is essentially a 1, a 2, a r, a r plus 1 to a s power i, a s plus 1 to a n.

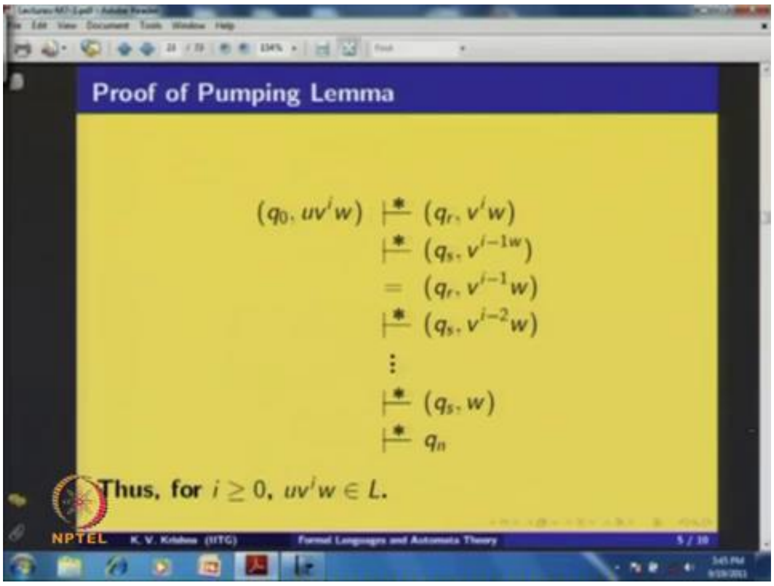
In the picture, you can clearly observe this property, UV power i W is accepted by the DFA, the DFA a that you can quickly observe. But of course, we have to give an analytical observation here, first you understand very quickly that by avoiding the loop here from q naught to q n you can reach; that means, v power i, where i equal to 0, if you look at that is what is just u w, you can clearly see in this picture that if you say.

If you apply the string u w that means with i equal to 0, v power i w, that string. If you substitute, if you apply in this initial state q naught, you see you will go to q r after u get, so over you are at q r and w starts from here. And, you can quickly observe that u w is accepted by this by the given DFA, because, you are reaching the reaching to the final state q n itself.

If, i is equal to 1 that the string x, that is clearly in the language, if i is equal to 2 essentially you have u v 2 w. Now, u you are reaching here, v 2 means essentially you once, once again you go through the loop, so that means v square, then w. Similarly, in this loop if you keep going for several times here in this loop, if you are going several times.

So, that much power that you can raise to v and the result in string that is u v power i w is going to the final state q n and thus you can quickly observe through this diagram that u v power i w for all i greater than or equal to 0 is accepted by the DFA. Now, that is what we will demonstrate here analytically, since q r is equal to q s, we see that there is a computation for U V power i W in A as given below.

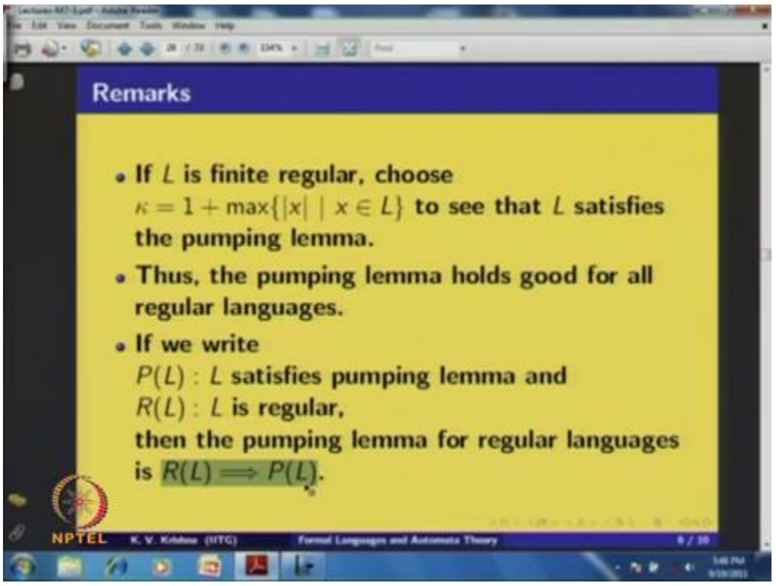
(Refer Slide Time: 12:01)



How that is, you consider the string for arbitrary i q naught U V power i W in finitely many steps of course, you are reaching to q r by after finishing U, as I have mentioned in this figure. Now, V power i W, you will keep going to this V in that loop for that many times, as you required that is what is the situation you reach to q s, that is then the rest of the thing is at first, once you consume 1 V and the rest is V power i minus W and since you know q r is equal to q s.

So, I again take back the state q r, and apply once again then I get reduced by 1 more V here and so on you keep reducing that continuing. Once, the v component is finishing the string, when the rest of the string is W. then you consider q s and in q s you can apply W to reach to q n. And, thus the string U V power i W is in L for all i greater than or equal to 0 and hence we have that result.

(Refer Slide Time: 13:08)

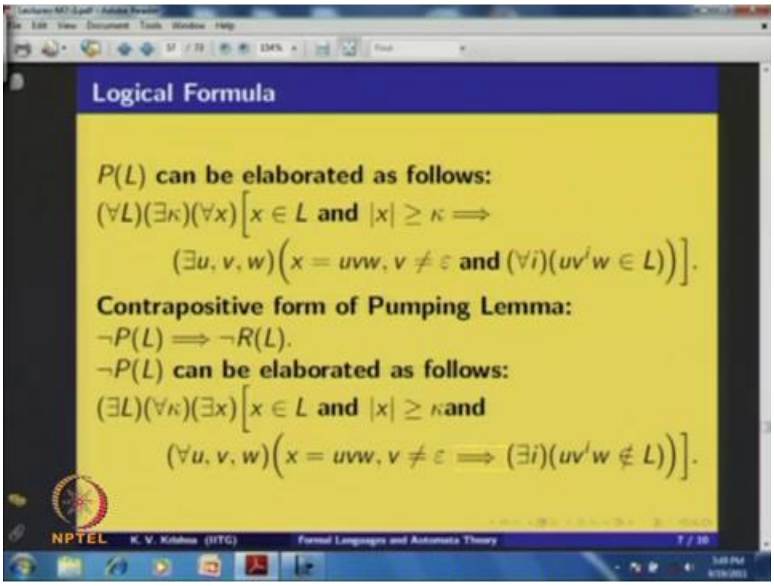


Now, let me look at some remarks on pumping lemma, if L is a finite regular language, then if you choose k to be for example, 1 plus maximum of all length of the all strings in L, there are only finitely many you consider the maximum say plus 1. Now, with this k the language L satisfies the pumping lemma, because in pumping lemma it is given that if you take any string whose length is bigger than or equal to k. Then, it can be partitioned into so on and the string U power U V power i W is in L for r is greater than or equal to 0.

But, there is no string in L whose length is bigger than or equal to k here, because we have considered k to be the maximum of all those strings plus 1 length. Thus you do not have any string and hence the result is vacuously true, so for a finite regular language the pumping lemma is vacuously true. And, thus I can compute that this pumping lemma holds good for all regular languages, because we have proved for infinite regular languages.

Now, from this observation you will see that for a finite regular language, a pumping lemma holds and hence this you have. Now, if we write P L the statement L, satisfies pumping lemma and if I denote it by L is regular by R L. Then, the pumping lemma for regular languages can be given by R L implies P L, because if L is regular then L satisfies pumping lemma.

(Refer Slide Time: 15:00)



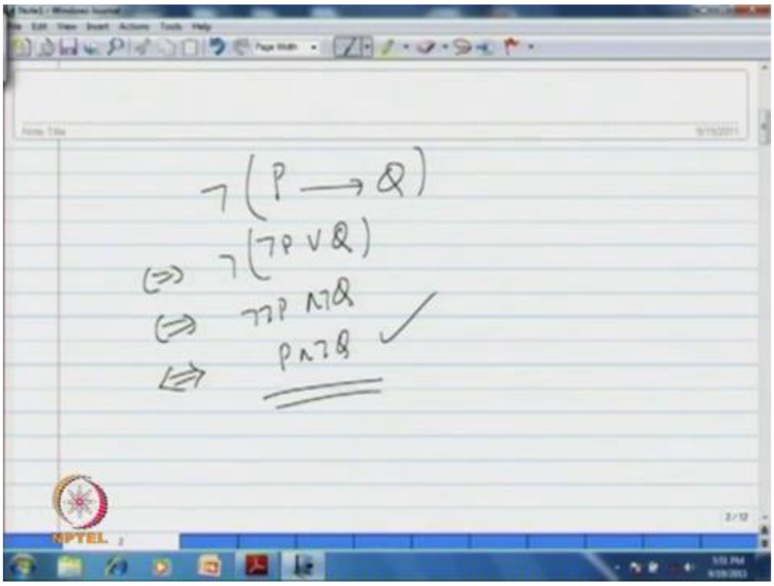
Now, this P L pumping lemma for regular language can be elaborated by logical formula as follows. For all regular languages, there is a k, such that if you take any string in L, whose length is bigger than or equal to k. Then, there exists UVW strings such that x is UVW, V non empty for all i U V power i W is in L, this is the formulation of P L pumping lemma.

Once again, you see because this is very important logical formulation, what is the observation here, if you take any regular language, there is a number k such that for all strings whose length is bigger than or equal to k. That string can partition such that so and so happens, that is U V power i W is in L for all i greater than or equal to 0. Now, let us look at contra positive form of pumping lemma, thus as I have mentioned this pumping lemma, we will use to understand non regularity; that means, essentially we will look at the contra positive a form of the pumping lemma.

If a language does not satisfy pumping lemma then it is not regular, that is written by negation P L implies negation R L. So, now we have to look at negation P L essentially, this negation P L from the above this we can quickly write, because of this, whenever you have the quantifies for all are there exists. If you apply negation of the statement there will be reverse, so for all become there exists and there exists become for all.

So look at here, so for a language that is if you here in pumping lemma, it says there exists a number, here for all k for all numbers corresponding to this L. If you choose, you can identify a string; that means, there exists a string x such that the length of x is bigger than or equal to k. And, for all strings UVW, x is such that x is of the form UVW, and V non empty, that implies there exists an i such that U V power i W is not in L, so this is a contra positive forms.

(Refer Slide Time: 17:55)

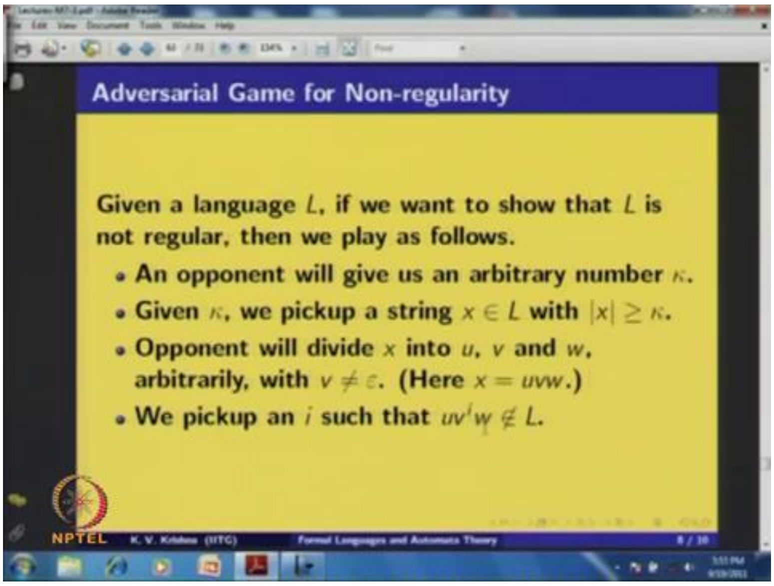


You can quickly see that, whenever you have the statement for example, P arrow Q, you know that this is negation P or Q and when you have the negation of this statement. This is equal to at the negation of this is equivalent to negation of negation P and Q, so that we used this logical formula to obtain. Because, here you have two things, this statement implies this you can call this P and this is Q, P arrow Q.

Now, once you applied that negation what you will be writing that is P n negation Q, that is what exactly we are writing this P and negation of this statement. So, using this we have formulated this contra positive form of the pumping lemma, thus what you have to do here, if you want to observe a language is not regular. For all k you are given a arbitrary k you have to identify a string, whose length is bigger than or equal to that k.

And, if you partition the string in any you know in any form that U with UVW; that means, for all choices of UVW with x is equal to UVW and V non empty. You have to identify i, such that U V power i is not in L, so this is what is point we have to observe to understand a language is not regular.

(Refer Slide Time: 19:40)

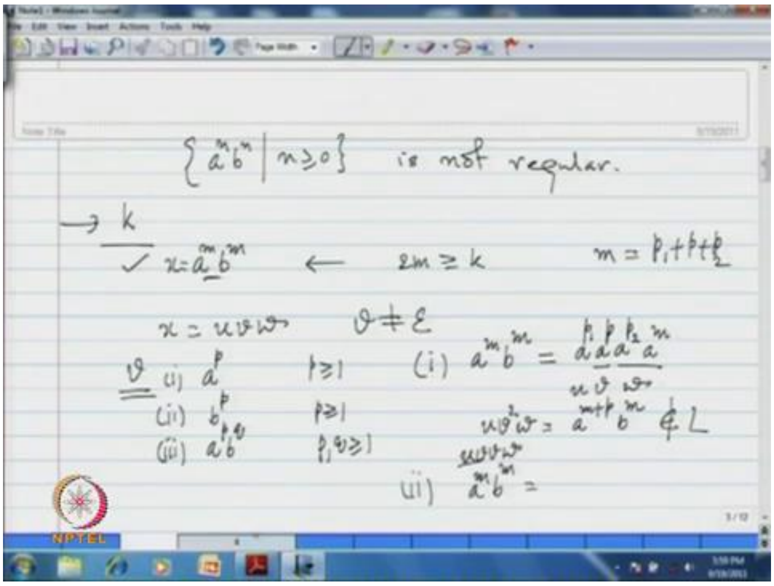


This can be better explain through an adversarial game for non regularity; that means, given a language L, if you want to show L is not regular, then we play as follows. So that means, here an opponent will give us an arbitrary number k, because you have to do it for all k. So, you take a number from your opponent an arbitrary number, then you pickup smartly a string x, whose length is bigger than or equal to k.

Once, it is the ball in your court; obviously, you can behave smartly, so here, k is given by the opponent, because this is an arbitrary thing. So, you choose because there exists x in L that is what is a point here, so you choose some string smartly, whose length is bigger than or equal to k. And, then since for all choices of UVW, where x is of the form of UVW is equal to x.

So that means, since it is for all forms you leave the choice to the opponent. So, opponent will divide as for his wish is the form UVW and of course, we will maintain that V non empty, then you have to identify i such that U V power i W is not in L.

(Refer Slide Time: 21:01)



Now, let me give you an example to show that the language is non regular, if you consider the language, because we have already observed this a power n, b power n, such that n greater than or equal to 0, you know this is non regular. This we have observed through mainly road theorem, now let me for apply pumping lemma to observe this is not regular. So, you take an arbitrary number k that is given to you now you chose some string, because any string is of the form a power n, b power n.

So, I do not have too much choice, only thing is on the length you have the choice, but let me choose a power m, b power m such that this length of this; that means, 2 m is greater than or equal to k. We choose this string, because this is given to you, now the opponent will divide this is into u v w, how you will divide with v naught equal to with v non empty, how you will divide that, it is of is choose and thus the possibility of v, we have to look at.

Now, since v is non empty in the string a power m, b power m, the possibility for v is it maybe of the forms say a power p, p greater than or equal to 1. Because, if b is completely within a or b maybe of the form b power p of course, p greater than or equal to 1. Some, non empty string r, if it is in between then you can have some a, so let me call a power p, b power q of course, here p plus q greater than or equal to 1.

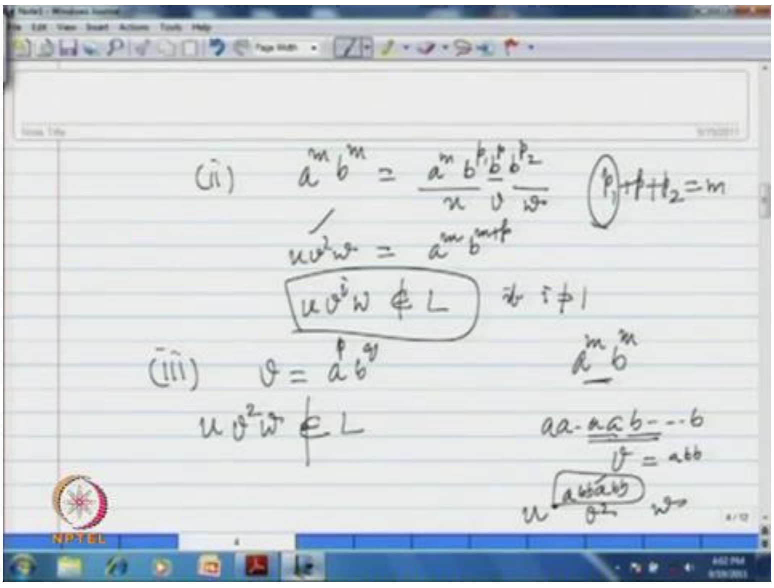
But, here since we have assumed that, in case of q is 0, we have already the first case if p is 0, the second case is already there. So, p and q both are greater than or equal to 1 here in this case, there are the three possibilities for the non empty string v to be non empty. Now, let us look at the first case the string a power m, b power m, what is the situation this, the first say let me call p 1, a power p that is a string v and a power let me say p 2 a power p 2 and then a power m.

So, here p 1 plus p 2 is m, clearly now what do you do this is now automatically u is a power p 1, v and rest of the string is w here. If you now raise the power of p to for example, say to that is the string u v 2 w what will happen here, the number of a s here will be a power m, plus p, then b power m, this is the string. Since, we have in raise 2 to the power 2, this string is nothing else, but u v v w, this u v w when you take u v here, there are the component of w, there are m a s and since we have 1 more v here.

So, a power b is getting added to this, so thus the string is a power m plus number of a here, so the number of b is m. But you see v is clearly not a string in L, this is a power m, b power m, the string is number of a s followed the number of b s will be in L. If the number a is equal to number of b, that is the condition here, so this is not a string in L, similarly, if you consider the second case, you can do the same thing because a power m, b power m is there.

Now, u will be here the v is completely in b say, let me say this is b power p is here. Now, before b power p you can have all a, so that means, a power m and some number of a whatever is possible, so that is that you can have so let me write this case.

(Refer Slide Time: 26:14)



In case two, same stop a power m, b power m, is a power m, b power p 1, b power p, b power p 2 with p 1 plus p plus p 2 is equal to m of course, p 1 can be 0, then just before this v you do not have any b s. Now, this is u, this is v non empty, p greater than or equal to 1 and the rest of the string is w here, again similar to the previous case, if you raise the power of v to 2, you can clearly observe the number of b will increase. But the number of a is m, only for example u v square w that is a power m, b power m plus p here is this.

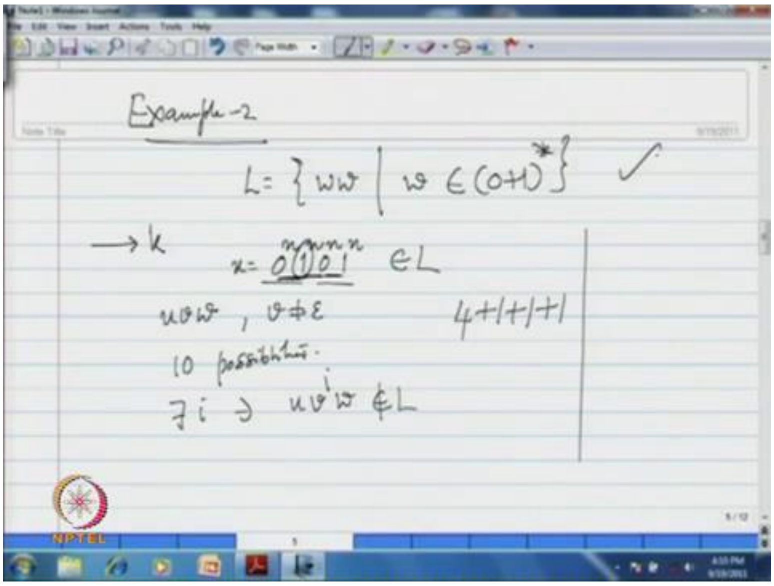
In this two cases it is not only the power 2, you observe that u v power i w is not in L, if i is not equal to 1, even if you make it i is 0, then the number of in the first case, the number of a will reduce and number of b is m. Thus you can quickly see that that is not a string in L, if you raise to the power more than 1, if it is not equal to 1, the number of a or b in first case and second case respectively will increase to observe that this string is not in L in both the cases.

Now, in the third case since v is assume to be of the form say a power p, b power q, so this is the string in between, so we have some number of a s and then the same number of b s. So, this is a string somewhere in between you have so many number of a s, p number of a s, say q number of b you have. Now, once you raise the power here maybe square for example, u v square w, what will happen since here number of a s is assume to b s greater than or equal to 1.

You can clearly see that, by raising this power for example, 1 a and 2 b s in place of v, once you write b square here a b b again a b b will come; that means, after a in the portion of b square. This is a portion of b square of course, you have the same u and w as it is. Now, in the portion of v square you observe that a b b, a b b is coming, if v is say a b b and you can clearly see that after a b have come after b.

Again, a s has come and b s have come, this is not the format of the language the string in the language and thus this is also not a string in L. So all the three cases, we have observed these are the three possibilities for the string, if b kappa not arbitrary string a power m, b power m, these are the three possibilities for non empty v. And, you observe that all the three cases, we you identified some i such that u v power i w is not in L and hence we can conclude that the language is not regular.

(Refer Slide Time: 29:46)



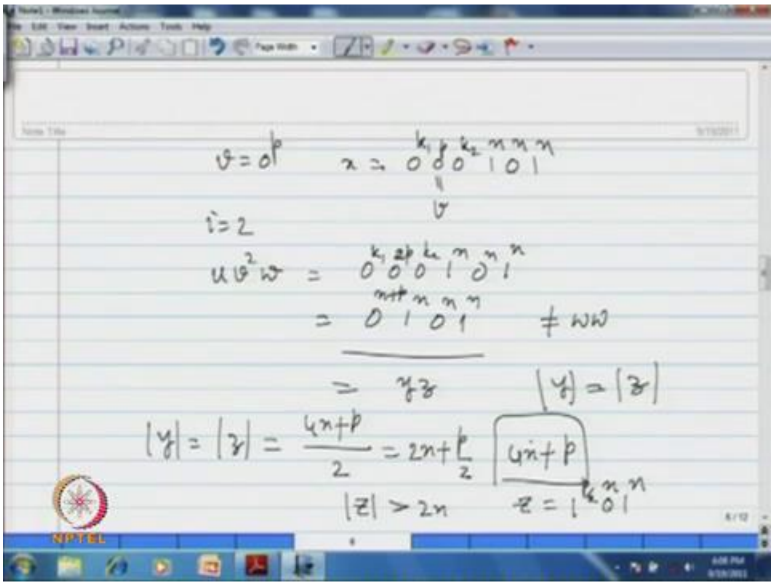
Let, me give one more example, if I consider the language say all those strings of the form w w, such that w is in star, in the string of the form w w. Now, again pickup an arbitrary k, this is given to you now you can smartly choose a string of course, by taking as string arbitrarily in this the number of the number of cases that you have to argue will increase and you may not have to control to understand that.

So, here smartly for example, if I choose say 0 power n, 1 power n, 0 power n, 1 power n, you can clearly see this is a string in this language, because the same whatever w is a first half I have taken I have written the same thing. So, this is it if I choose the string now this string will be divided u v w, now the possibilities for v, because v is non empty. Now, you look at carefully this v can be completely in the first block of 0s or maybe the first blocks of ones or the second block of 0s or maybe the second block of ones.

So, there are four possibilities and then it can be you know in between the first block of 0s and ones, that is one more possibility. Now, you see and similarly here another possibility and what else now it can include all this and it can actually intersect this, so you see there are several possibilities here you can observe that there are 10 possibilities. So in each possibility of v, we have to observe that there is i such that u v power i w is not in L.

Now, let me quickly discuss one case and then we proceed to understand I leave it as an exercise to understand that each case you can identify n i such that this is not in L.

(Refer Slide Time: 32:33)



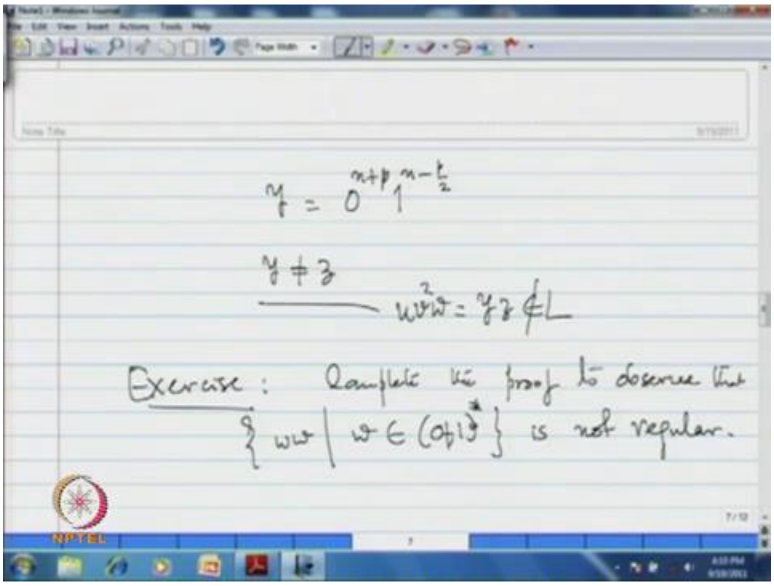
For example if v is say 0 power p, it is a first block of 0s such that x is equal to say 0 power k 1 and this is under consideration that is 0 power p. This is what is v and the rest of the 0s, maybe let me call it as k 2 and the remaining strings that is 1 power n, 0 power n, 1 power n, if this is a situation now you choose say again i equal to 2. Now, the string u v square w, that is 0 k 1 here p plus 2 p 0 k 2, 1 power n, 0 power n, again 1 power n; that means, essentially the number of 0s here is n plus p, 1 power n, 0 power n, 1 power n.

Now, we have to observe that this is not of the form w w for any w, if this is of the form w w you can have two strings say x is already used x is a given string, say this is y z if you divide with mode y is equal to mode z. Now, observe that this y and z are different because of it is of the form w w, then we want to observe the contradiction, instead what I am doing this string you assume it is of the form say y z with lengths are same.

Unless, y equal to z you know this is not of the form w w, so here now the length you observe this is this is 4 n plus p is the length. Now, what is the length of z and if you compare with the length of y, so if you divide this mode y is equal to mode z. This is 4 n plus p by 2 say let me call it as 2 n plus p by 2, now you look at z is suffix of the result in string u v square w here.

And thus, this length of z is greater than 2 n and you observe then this z is of the form, z is equal to this because this is there is a shifting here. Let me say 1 power say p by 2 number of ones, whatever is coming and then this is 0 power n, 1 power n, this is what is coming and whereas, y will be the prefix of the string.

(Refer Slide Time: 36:00)



So, y will be now this is 0 power n plus p, 1 power n minus p by 2, because this many number of ones are shifted I mean you have to consider in z. So, you clearly see that these two strings are different and hence you conclude that this y z is not in L that is u v square w y z this is not in L, thus in this case we have identified n i such that u v power i w is not in L.

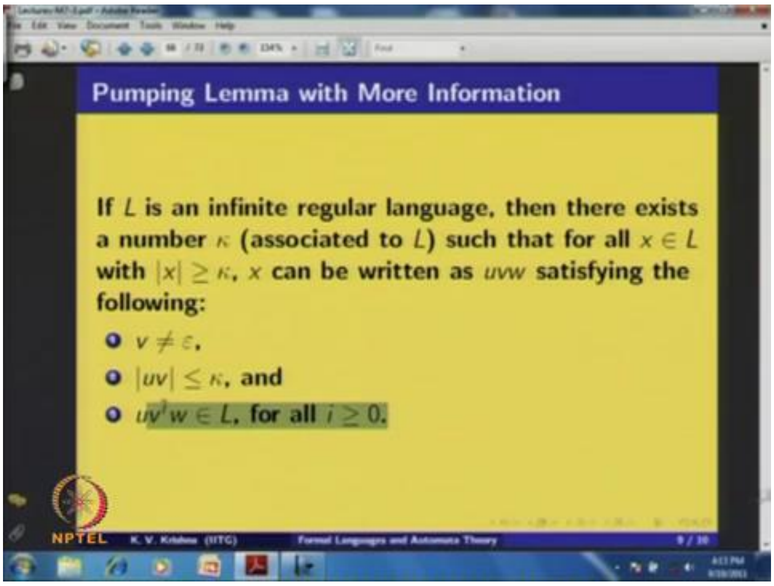
Now, if you consider case two; you can have maybe the v is in here the block of ones, case 3 it is the block of 0s, case 4 is the block of ones. Likewise, you can argue and understand that there is i; that means, u v power i w is not in L, in each case you can observe that and understand that the language is not regular. So, an exercise here complete the proof to observe that w w such that w is in 0, 1 star, so I can put this plus regular expression is not regular.

Now, let me make a point here in the first example, if you are given k you choose a string of course, you have a choice of choosing, but since the first language 0 power n, 1 power n, any string of the form 0 power n and 1 power n. There are three cases to discuss, if you look at the second example, the strings of the form w w, for any string w in 0 over 0 1. Now, given a k you can smartly choose a string here I have the choice say I have chosen 0 power n and 1 power n, because this is such a nice string that you can observe the cases very quickly.

If you choose an arbitrary w in over 0 1 the situation will be complicated and the cases to discuss will be little bit difficult, because the w arbitrarily when you have chosen how you have to argue that the result in string at some point to the phi. Whether, it is in L or not to say that it is a little bit difficult, still when I have chosen this string and I have observed that there are 10 cases. Now, all the 10 cases I have to argue and I have to identify n i to prove that it is the language is a not regular

Now, I give you some more information about pumping lemma, then we can in fact, observe more in pumping lemma to understand very quickly that the given language is not regular.

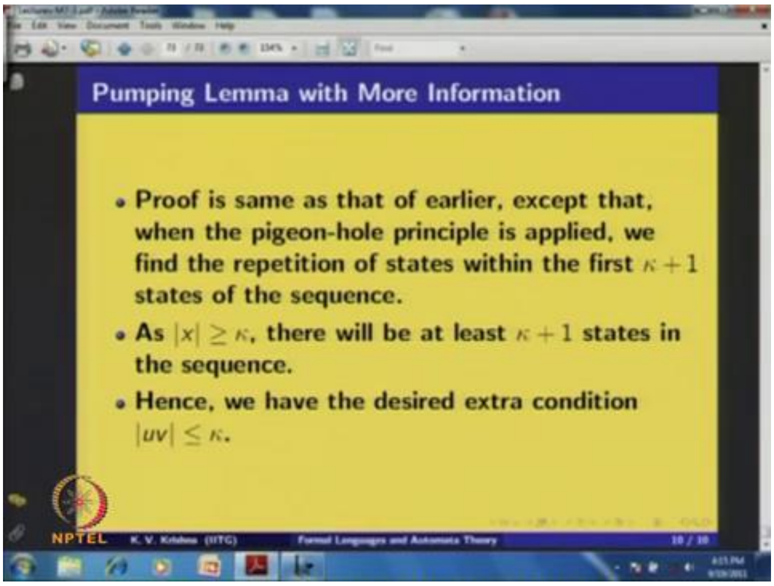
(Refer Slide Time: 39:34)



That is the version of this pumping lemma is follows, if L is an infinite regular language then there exists a number k such that for all x in L, with mode x greater than or equal to k x can be written as UVW satisfying the following. Because, everything is same the division, now here V non empty as earlier, but here one more important point is the length of U V is less than or equal to k.

So, you can divide UVW not only satisfying the V is non empty, in addition to that the first two strings. When you are dividing you can just look at the division of UVW such that the length of U V less than or equal to the given k. So, and of course, when you have such a division U V power i w is in L for all i greater than or equal to 0.

(Refer Slide Time: 40:28)

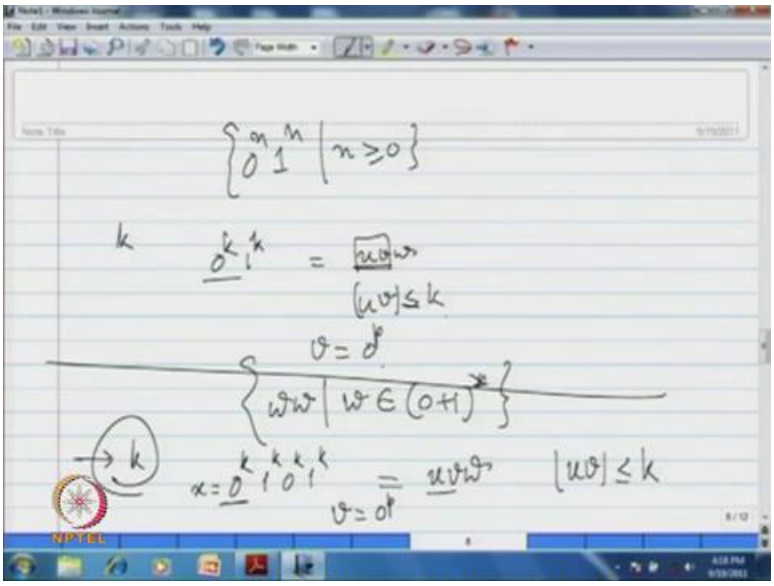


So, this version of pumping lemma if you are looking for the proof is essentially same as earlier, only thing is when you are looking for the proof I simply mention that by pigeonhole principle. There is r and s such that q r is equal to q s that is what we said in this diagram here, what I will suggest you instead of just choosing some arbitrary place where q r and q s are coinciding. But there, wherever the repetition is coming what do you keep going from the left side from q naught q 1, q 2 and so on.

Whenever, the first repetition comes that repetition you identify that, then this u and the loop v, the length will always less than or equal to k, I give you this argument, when you are applying the pigeonhole principle, you just concentrate on looking at the pigeonhole principle. The repetition of state within the first k plus 1 states of the sequence, why do you get the repetition with first k plus 1 state, because there is only k state, once you cross the number k, automatically and definitely you can have one state repeated.

So, within the first k plus 1 states, you should certainly have 1 state repeated, since there only k states and thus you can have the length of u v less than or equal to the k, so this loop you can always identify which in the first k plus 1 states. So, that is reason why you have this condition u v is less than or equal to k with this condition, you have an advantage you know that you need not discuss too many cases for example, of the language a power m, b power m, when you have considered.

(Refer Slide Time: 42:41)



Now, let me give you in case of 0 power n, 1 power n we have argued 3 cases, once you give me k what I will do I will choose 0 power k, 1 power k, this string I will choose, now if you divide this u v w, this u v the portion as mentioned is less than or equal to k. And, hence the v is essentially of the form 0 power p only there is no choice of getting 0 power p, because this u v less than or equal to k and you know the first k symbols are 0s only and hence you observe v can have just 0s only.

And, thus you can discuss the only one case here, similarly when you look at the second example w w such that w is in that 0 1 star in this example, if you take this example you given k you can choose now these strings 0 k, 1 k, 0 k, 1 k. Smartly, you can choose these strings, now you look at this is an element of this whose length is greater than or equal to k.

Thus, the choice of choosing the string is in my hands now, you are giving me k and I choose this swing, clearly this is a string of length greater than or equal to k. Now, one whenever you divide this u v w of the form u v w with u v length is less than or equal to k, you clearly observe that this entire string u v has to be the first block of 0s only.

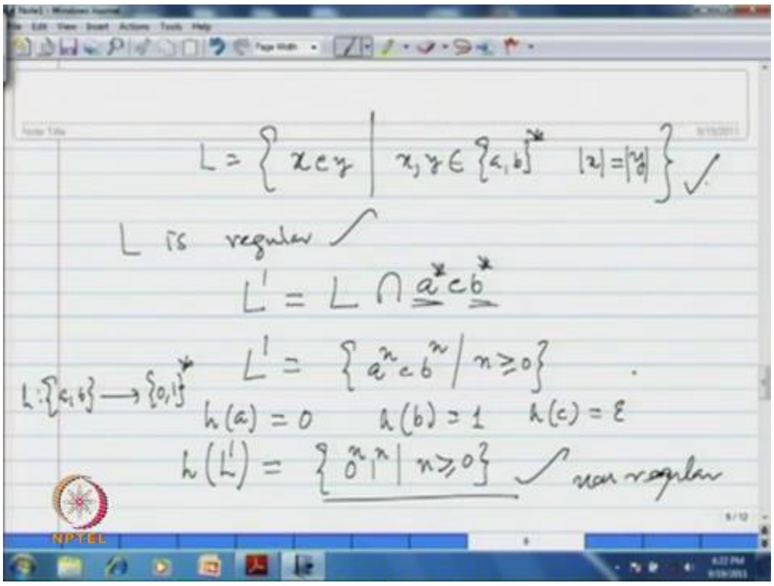
In particular, v is of the form only 0 power p for some p positive and hence only one case I will discuss to conclude quickly that this is not regular. Because, again the pumping lemma, it is saying that it is not only v non empty, whenever you divide u v w, because for all choices of u v w with the length of u v is less than or equal to the chosen number k, u v power i w belongs to L.

For all i, that is what is pumping lemma saying to observe the non regularity, by considering this version you can quickly conclude that a language is non regular, that the number cases will drastically go down by adding this additional condition. Now, we have observed several closure properties of the regular languages and pumping lemma also we have observed as the property of regular language.

Now, all these properties are very useful in not only you know identifying a language is regular to identify some to a language non regular also. Because to understand that a language is suppose you have established certain languages are non regular, you can apply closure properties. If you assume some language is not to observe some language is not regular, it is not just the pumping lemma you will you may use the closure properties.

Also if you assume that it is regular somehow the closure properties if you apply using the closure properties, the result in language if you can somewhat identify that it is non regular. Then, as for the closure properties you can quickly conclude that using the properties of closure properties and pumping lemma, the languages are non regular. So, earlier I gave example to using closure properties to observe that certain languages are regular.

(Refer Slide Time: 46:35)



Now, let me give you one example using that you can see that languages are non regular, let me consider the language say x c y such that, x and y strings of a and b and c is a fixed symbol, length of y is equal to length of x. If I consider this language you can understand any string x and y you choose in between I put c the symbol and this x and y should be of the same length over symbols of a and b.

Now, if you assume L is regular, now you understand that this language L dash, this intersection with a star c, b star, this is also regular. Because a star c, b star is regular I have assumed all is regular intersection of two regular language is regular. Now, you understand what is this L dash, L dash is then a power n, c, b power n, such that n greater than or equal to 0, you can observe this quickly.

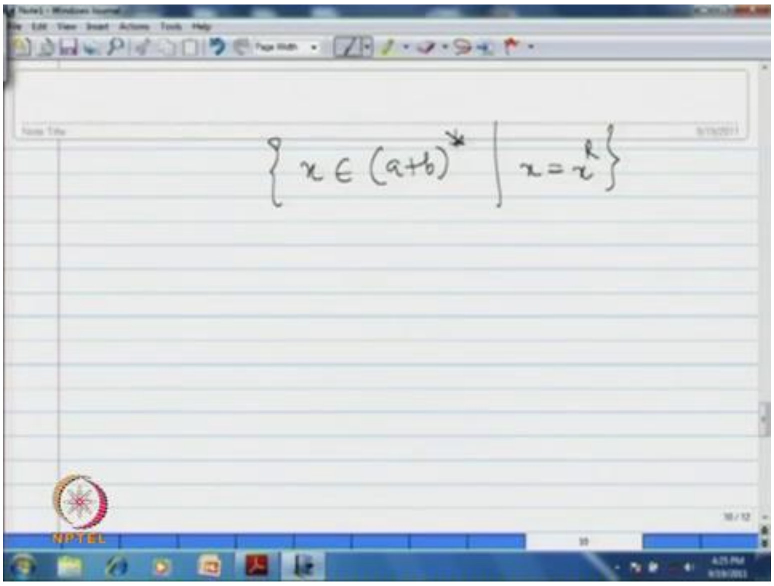
Because, length of x is equal to length of y, so x c y is then of the form a power n, c b power n form, because the lengths only should be same in this c is fixed here, you have to choose a here from here, b from here. And, since the lengths should be same that is a power n, b power n form, now using homomorphism if I send h of a to 0 and say h of b to 1 and say h of c to say empty string.

Suppose, we send it like this, then the under this homomorphism; that means, I essentially mapping from this to 0 1 star, so the mapping is essentially h is from a b to 0 1 star. So, what is the image of under this homomorphism what is the image of L dash, image of L dash, if you observe carefully this is 0 power n, 1 power n such that n greater than or equal to 0, you see to we know this is a non regular language.

So, if you assume L regular, then this L dash that is also regular and since L dash it is regular, homomorphism image of the regular language is regular. So h of L dash should also be regular, but we have got a non regular language and hence our assumption that L is regular wrong. So, using these closure properties we have concluded that, this is a non regular language, so thus these closure properties are useful to understand non regularity of languages.

So, from this discussion you have observed that there is a non regular language coming out of this, but using closure properties of the regular language intersection homomorphism. We should always have regular language you have got a non regular language and hence our assumption is wrong, so L is non regular.

(Refer Slide Time: 50:27)



Now, let me give you an exercise to understand that it is non regular you can apply pumping lemma to prove that all those strings which are palindromes power a b you can observe that this is a non regular language.