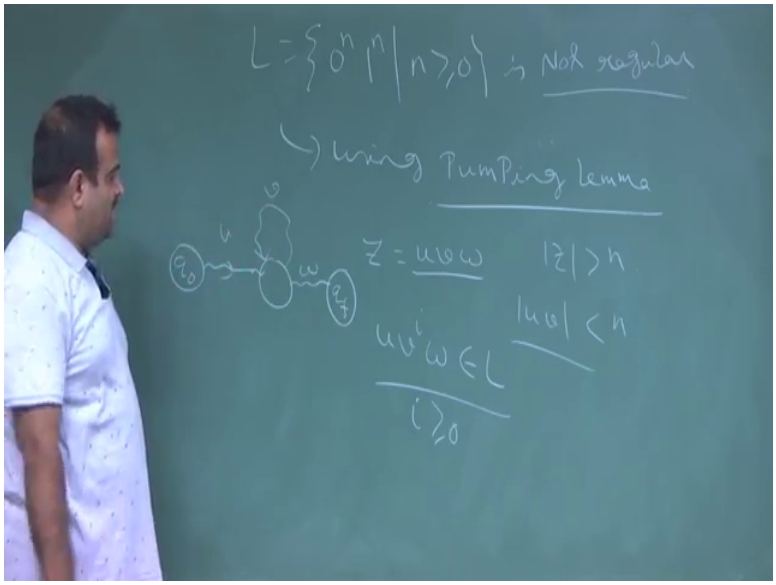
Introduction to Automata, Languages and Computation
Prof. Sourav Mukhopadhyay
Department of Mathematics
Indian Institute of Technology, Kharagpur

Lecture – 29
More on Pumping Lemma

Ok, so we are talking about Pumping Lemma, application of pumping lemma. So, we just have this is a powerful technique to proof a languages is non-regular. So, using this technique, we have seen the language like this.

(Refer Slide Time: 00:33)

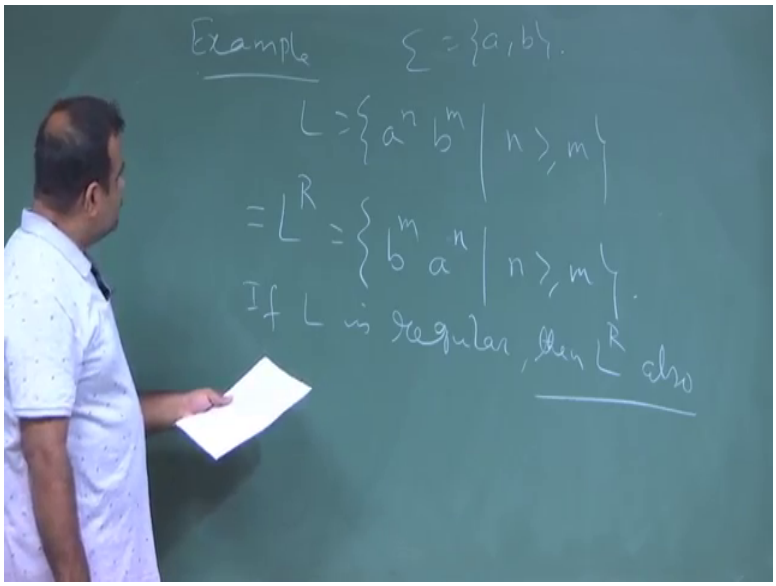


0 to the power n, 1 to the power n, it is not regular, it is not regular ok. So, it is a using we have seen this using pumping lemma. Pumping lemma is the necessary condition for a regular language. If your language is regular, then there will be a n, n is basically the number of regular language regular means we have a DFA which is accepting the language. So, n is the number of state in the DFA. So, then we have we have that n. And if we take any string which is greater than n, then which can be written as u v, where u v will be length is less than n and then this u v to the power i w belongs to n. This is necessary condition where i is greater than equal to 0. If i is 0, then u w, then we can pump here there.

So, it is q 0 with u, we go some state. Then from there with v we go here, and then with w we can go to the final state. So, since it is a loop we can keep on bumping there that is

why it is called pumping lemma. So, using this pumping lemma, we have seen few language which are not regular, which is not satisfying the pumping lemma and it is a necessary condition ok. So, with help of that we will show a few more language which is not regular.

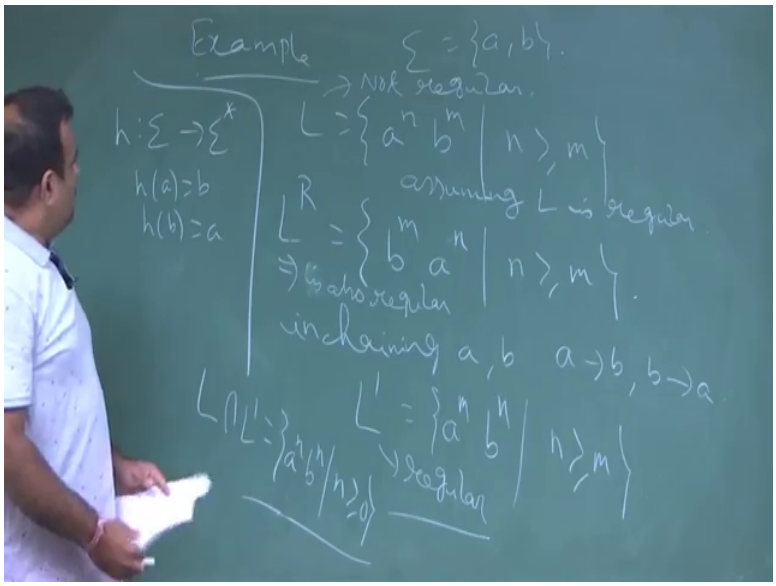
(Refer Slide Time: 02:35)



For example, we take this language a to the power n b to the power n; n is greater than equal to m. We show this language is not regular for sigma is on a, b ok. So, this means what this is the number of b's number of b's are more number of b's than a ok, because n is more. So, more number of a's than b's anyway.

So, if we take the reverse of this L of R, which will be basically this will come here a to the power m sorry a to the power m, this is just the reverse n is greater than this ok. Now, the if we have seen if a language is regular, the reversal is also a regular. Then so if L is regular then L prime also this we have seen. This is the property of a regular set ok. Now, in the in this, this is a, this is a L 1. So, here we just exchange this a, b. So, we just exchange this a, b.

(Refer Slide Time: 04:21)



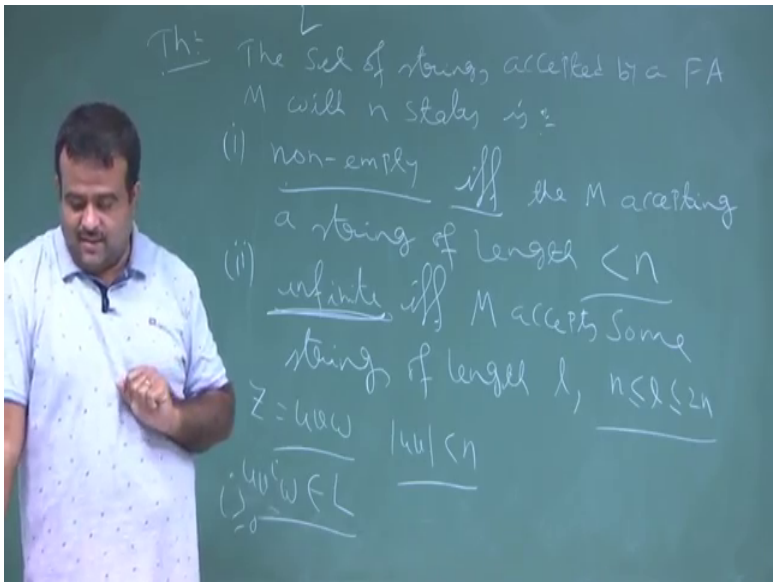
Interchanging a and b, that means, a is going to b b is going to a this mapping and this is a homomorphism. We have seen the homomorphism; this is a homomorphism. So, if you do that, then we get L prime which is basically a to the power m, b to the power n; we are exchanging a by b b by. So, a is going to b a particular string, and b is going to a a particular string. So, this is a under this mapping sigma is going to sigma star where h of a is b, h of b is a. So, this is a homomorphism which is a substitution. And we know the regular set is close under substitution. So, here n is greater than is equal to m ok.

So, if you look at this now if this is regular then L prime will be also regular, we are assuming this is regular. We are assuming this is regular assuming L is regular, that means, L prime is also regular by that assumption, this is also regular. Now, we know the regular set is close under substitution or in particular homomorphism, so that; that means, this is regular. So, L and L prime are regular.

Now, if you take the intersection, then what it will be? L and L prime, If you take the intersection L intersection in prime this will nothing but a to the power n b to the power n n greater than equal to 0. Now, since L is regular, L prime is also regular. So, their intersection is also regular that is a another properties of regular set. So, this has to be regular, but we know this is not regular, we have prove this using pumping lemma, so that means, what there is a contradiction.

Since this is not regular, so this cannot be regular then this cannot be regular then this cannot be regular. So, this implies contradiction; that means, this is not regular ok. So, this is one example. Even we can directly using pumping lemma to show this is not regular. So, this proof will be this direct proof will be there in the lecture note ok. So, this is this is one direct application of the pumping lemma. Now, we will prove a theorem which is coming from the pumping lemma on a regular set ok.

(Refer Slide Time: 07:39)



So, this theorem is telling, so suppose the set of string accepted by a finite automaton m with n states so that means, this set is regular. If we have a regular set, then we have this properties is either it is empty, or if it is non-empty then non-empty if and only if the finite automaton m is accepting a string of length less than n, accepting a string of length less than n.

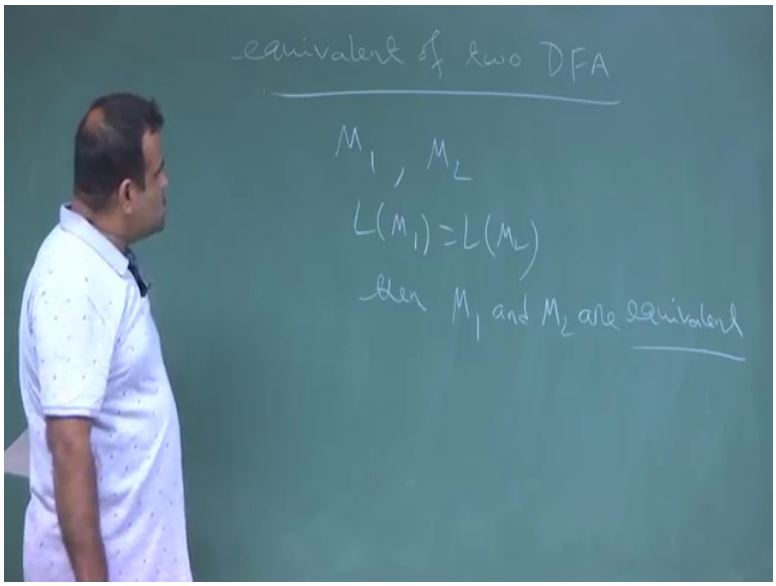
If it is accepting a string of m less than n, then the say L is that set, set of all string which is a regular set that that means, we should have a automaton to accept that, so that automata is n. Now, this set we want to check whether this set is empty or not. If it is non-empty means we should have a stream which is accepting, this string which is accepting by this automata and whose length is less than n ok. And when it will be infinite, it is infinite if and only if m accepts some string, of length L, where L is belongs to n less than L less than equal to 2 n.

It will be infinite if we have a string of more than length more than n then it will be an infinite length this is a theorem this can be prove using pumping lemma. So, first part is what? First part is if it is non-empty means if and only if, if it is non-empty that; that means, we have a string which is accepting it. And if we have a string which is accepting is then it is a non-empty.

And the second part second part is coming from the directly coming from the pumping lemma, that means, if we have a string off, so this is consist of n state now if we if we have a string of length L. So, then we can write this w, Z is u v w where u and v are less than n length then we can pump in v. So, then u v to the power i w will also belongs to L. So, this will give us the infiniteness using pumping lemma.

So, it will be either empty or finite or infinite. So, when it will be empty? If there is no string of length less than n; and when it is finite if we have a length of string less than n and when it is infinite, if we have at least one string of length more than n, then it is this ok. So, we will use this theorem to see whether R 2 DFA is equivalent; so this theorem will use. So, the details proof of this theorem will be given in the lecture note ok. So, we will use this theorem to see whether two DFAs are equivalent.

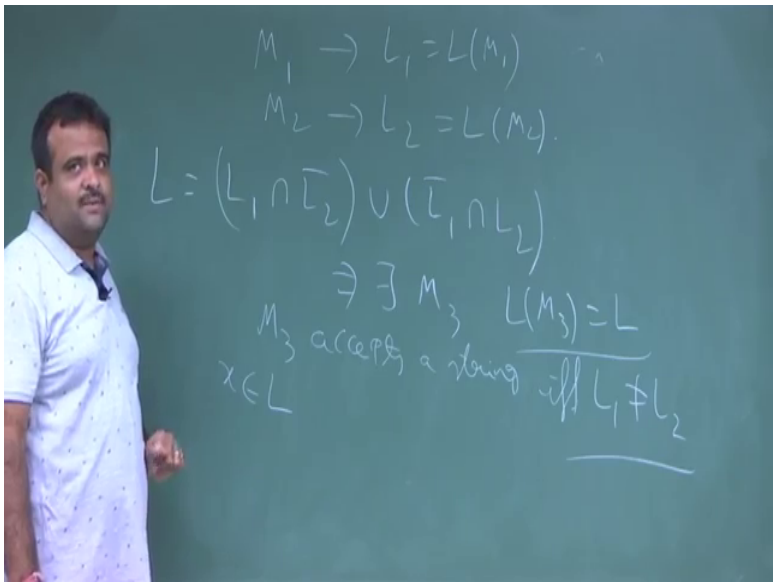
(Refer Slide Time: 12:03)



So, two DFA with finite automata, so that means, we want to see given two DFA say M 1 and M 2, whether they are equivalent or not. So, what do you mean that they are equivalent, equivalence means if they are accepting the same language the that means, if

L of M 1 is L of M 2 which is then we say these two DFA are equivalent, then M 1, M 2 are equivalence ok. So, yeah, so how to show this we have to. So, you have to check whether two DFAs are equivalent or not.

(Refer Slide Time: 13:13)



So, how to check this? So, to check this suppose M 1 is corresponding to L 1 which is nothing but L of M 1, and M 2 is corresponding to the language L 2 ok. Now, we want to see whether this L 1, L 2 are same or not. So, for this what we do? So since they are coming from this so these two are regular. So, now, we construct this set L 1 intersection L 2 complement union of L 1 compliment intersection L 2 ok. Now, this will denoted by L.

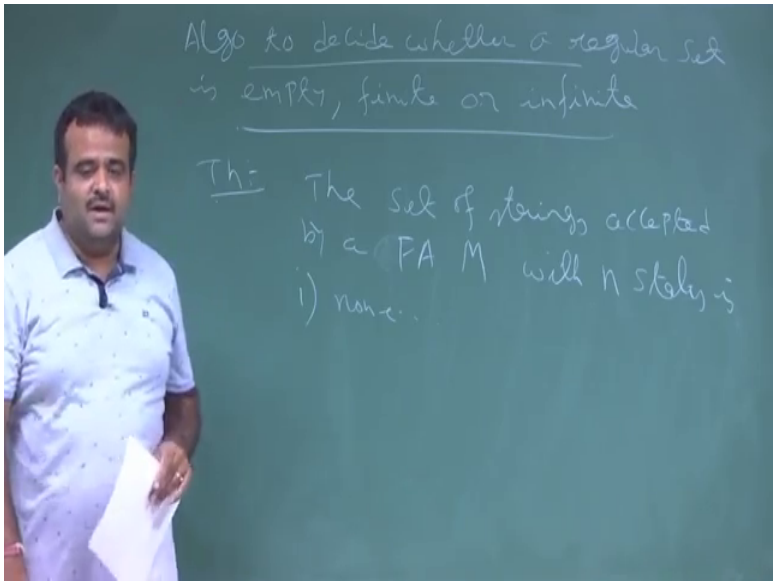
Now, what is L? Is L is regular, this L is regular? Yes, L is regular. Why? Because L 1 is regular L 2 is regular. So, L 2 complement, L 2 is regular, L 2 complement is regular, then we have this is the closure properties of the regular set. Now, we know intersection of two regular set is regular. So, this is regular, this is regular, we know the union of also regular. So, this is L is regular. So, once L is regular, that means, we have a DFA we have a DFA M 3 which is accepting this ok.

Now, if this is now you have to check whether this is empty or not. If the set is not empty, then these two DFA are not equivalent. So, if M 3 accept a string, string, string if and only if L 2 not equal to L 1, so that is the way we can check whether this is a regular set. Now, we have to check the whether this regular set is empty or finite or infinite, so

that we can do using some algorithm, algorithm to decide whether set regular set is empty or not, empty finite infinite.

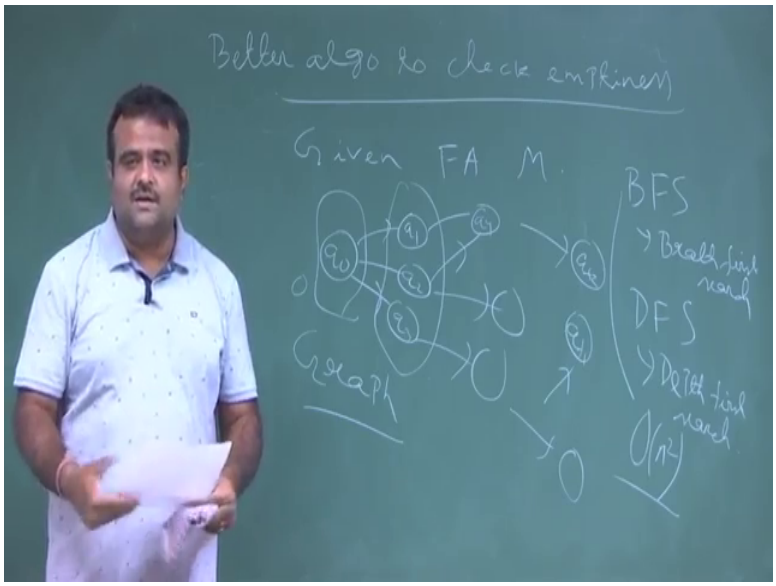
So, if it is not empty, if there is a string which is belongs to this, because if x is belongs to this L that means L 1, L 2 cannot be same. If they are same, these has to be empty because the way we have constructed this. So, now we will talk about how we can check whether a regular set is empty or not, empty finite infinite. So, this we can use the pumping lemma to check this.

(Refer Slide Time: 16:13)



So, this is the algorithm algo to decide whether regular set is empty comma finite or infinite. So, this is we have just now we have seen the theorem. So, since this is a regular set, so we have a automata which is accepting. So, set the set of string accepted by this by a finite automata M with n state, so this just now we have seen is non-empty, non-empty if we have a if and only if we have a language, we have a string of length less than n, and it will be infinity if we have a string of more than n. So, this, this, this way we can prove that using the pumping lemma.

(Refer Slide Time: 18:03)



But there are other ways also, we can see this because once we have a finite automata, then we can see whether this so this is the better algorithm algo to check to check the emptiness ok. So, you have given the automata, given the finite automata M. So, finite automata means we have some this is a graph, we have a graph, we have a starting state, then you have some state from the starting state say q 1 q 2, q 3 something like that. And we have some state from here something like that and then you have some final state q f 1 like this, and you have some transition on this. So, this is basically a graph; we have a graph.

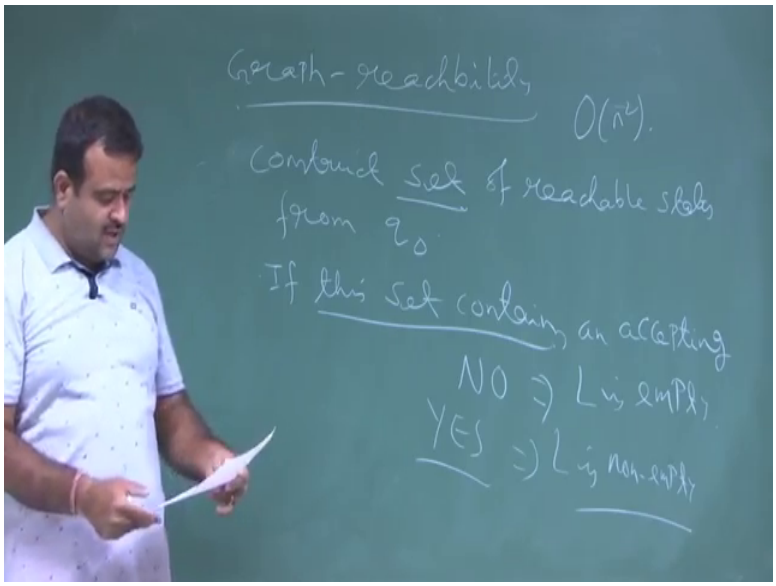
Now, in this graph, what we can do, we can do the graphs traversal. We know there are two ways we can do the graph traversal. What are two ways, one is BFS, the Breadth First Search, which search the graph level wise. This is the level 0, this is the next level like this. So, we reach to the all the nodes level wise, level 0, level 1 like this. From the level 1, we reach to the all the nodes which are connected with the level 0 which you are connected to the previous level node; so leveling. This is the one way we can do the search that is called BFS.

Another way is DFS. DFS is nothing but Depth First Search. So, you start with the vertex, and we keep on going until you stuck. Once we stuck we come back to that position and we explore the new path like this. So, what do we do? We do the graph search ok. We do the graph search. And we found from q 0 where we can reach all the

possible nodes we can found where we can reach. And in that reachable nodes, reachable states or the nodes we can say graph since it is nodes, the reachable states where from q 0, we can reach we can list it out, and that will take in the both the algorithm. We will take the time complexity order of n square with the time order of n square we can find that.

Now, so we have the list from q 0 where else we can go from q 0 by using the graph search. And then from there what we can do we can just check whether any of this reachable node is a final state or not ok. So, that if there is any final state, then it is non-empty; if there is no final state then it is empty. So, let us just write this ok.

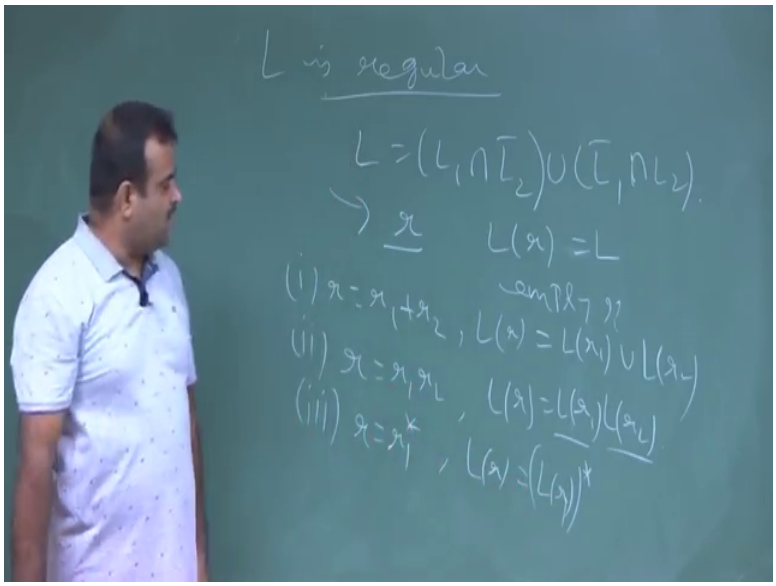
(Refer Slide Time: 21:41)



So, this is graph reachbility ok. So, how do you do, we construct a set of states or nodes set of reachable states from q 0. So, this we can do using the graph traversal or this BFS or DFS, and that can be done if there are n state in the automata that can be done order of n square a time complexity ok.

So, we construct this set. Now, if this set contained an accepting state, if this set contains an accepting states, then it is non-empty; we check that. And if the output is no, if there is no accepting state, then the language is L is empty; otherwise if it is accepting it is containing any accepting state, yes, then it is non-empty. And L is non-empty means that that DFS are not equivalent. The way we constructed the DFA they are not equivalent ok. So, even we can do it by using the regular expression.

(Refer Slide Time: 24:05)



So, how we can do it using the regular expression to check whether the graph is non-empty? So, we know r is L is regular, because the way we have constructed ok. Now, now, L is basically what? L is L 1 intersection L 2 complement union of L 1 complement intersection L 2. Now, we know these how to construct a DFA for this, because this is a we know the you for intersection we can construct a DFA, for complement also we can construct a DFA; we can reverse the states I mean the accepting state who are the non accepting state we can make it accepted make it the new accepting state, so that will give us the compliment.

Intersection also you have seen, union also you have seen. So, basically we can construct a finite automata for this. And then just now we have seen that by using the graph search we can check that, but there is another way to do that using the regular expression. So, this L, L we can have a regular expression for this r say ok. Now, we check whether L of r is L of r is basically L empty or not. So, this we can check using the number of operator on this regular expression. So, regular expression r is either three of this way r will be r 1 plus r 2. So, in that case, L of r is nothing but L of r 1 union L of r 2. Now, it will be empty if both are empty, if the regular expression r 1 and r 2 both will corresponding to the empty set, this is one possibilities.

Another possibilities is it is concatenation r 1 r 2. So, L of r is basically L of r 1 L of r 2, so that means what? That means, if empty means either one of this empty, then it will be

empty ok. No, both has to be empty, because if one is empty, then epsilon will come; so both has to be empty. So, that way you can check by the this inductive method. So, we are reducing the number of operator. And now this last one if r is r star that means, L of r is basically sorry r 1 star. So, we can check whether L of r 1 is empty or not; if L of r 1 is empty then this is empty set. So, this way also we can verify a set is a regular set is empty or not.

Thank you very much.