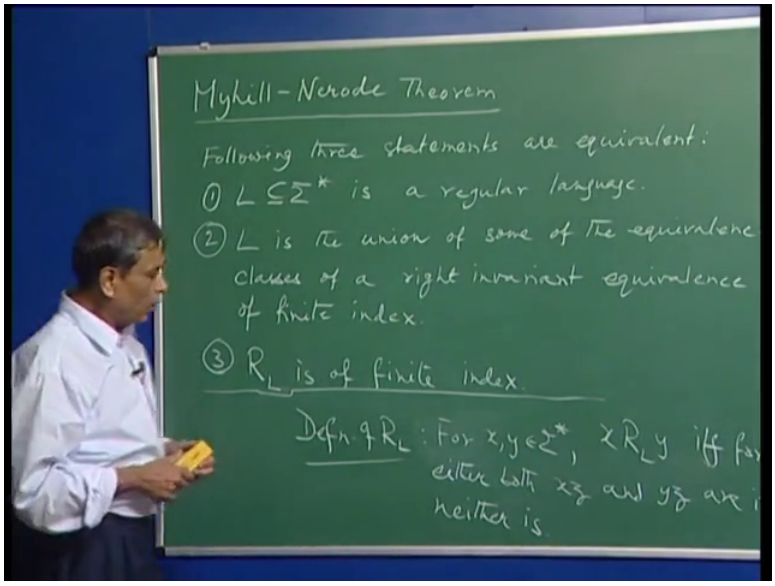
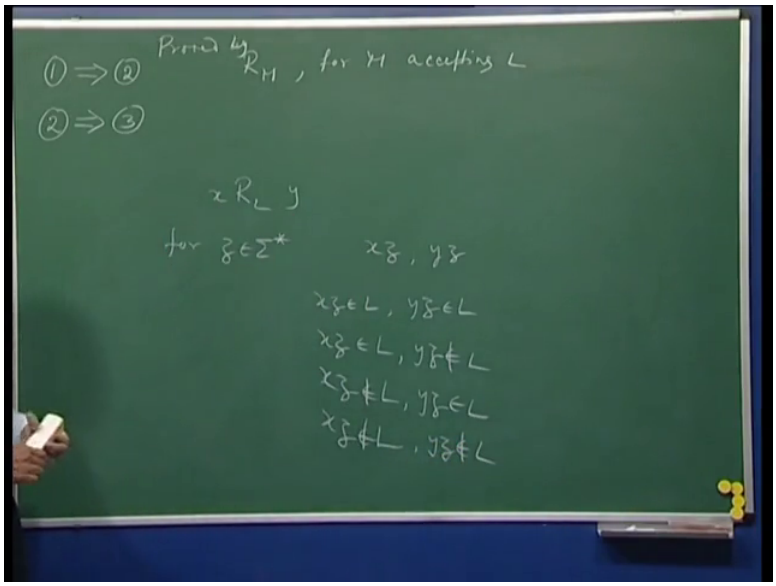
Theory of Computation.
Professor Somenath Biswas.
Department of Computer Science & Engineering.
Indian Institute of Technology, Kanpur.
Lecture-17.
Continuation of proof of Myhill-Nerode Theorem.

(Refer Slide Time: 0:37)





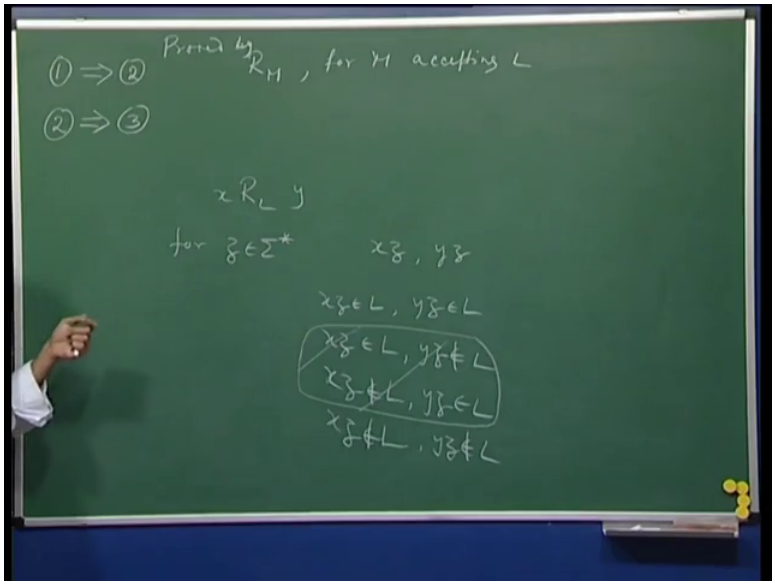
We are proving myhill nerode theorem and that theorem says that the following 3 statements are equivalent and we propose to prove this theorem by 1st showing 1 implies 2, then 2 implies 3 and then 3 implies 1. Last time we completed the proof of 1 implies 2 and just to remind ourselves what the proof was, we start with the assumption that l is regular. L is regular means there is a dfa m which accepts the language l and then we considered rm for

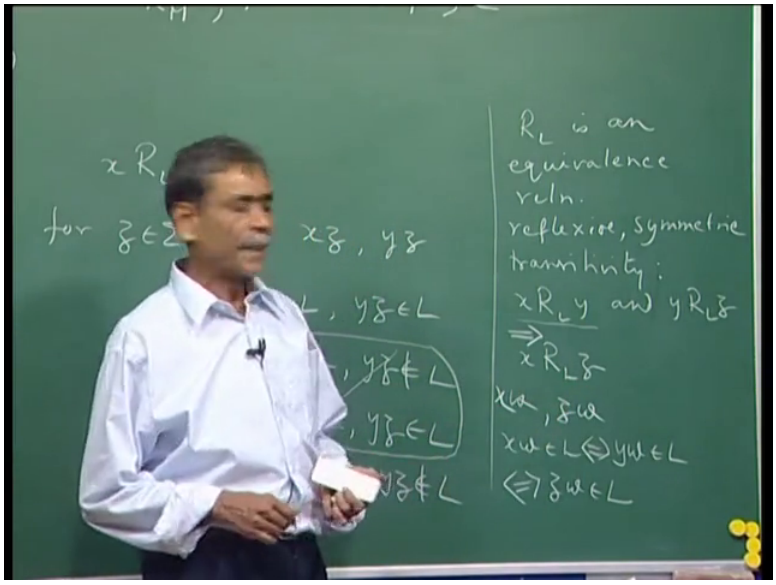
that machine m, we defined an equivalence relation rm last time and we found that time the rm, equivalence relation rm for m accepting l, we found that of course it was an equivalence relation and rm, the equivalence relation was right invariant, we proved that.

And it had finitely many equivalence classes, so therefore it is of finite index and of course the language is the union of some of its equivalent classes. So 1 implies 2, we can say proved by rm you know, just considering the equivalence relation rm. Now we would like to prove 2 implies the statement 3, here we start with the assumption that l is such a language satisfying the conditions here and then we would like to show that the equivalence relation rl for this language is of finite index. We defined rl last time and it was like this, again rl is an equivalence relation on sigma star which relates 2 finite strings you know in sigma star, this relation is pairs of such strings.

And we say that x rl y if and only if for all z in sigma star, maybe i will write it here. So when x rl y hold, it will hold only when for all z, so take any string z in sigma star, when you take the 2 strings xz and yz, right by concatenating z 1st to x and then to y, you get these 2 strings. Now you see what are the possibilities xz in l and yz in l, xz in l but yz is not in l, xz is not in l and yz in l and both xz and yz both of them are not in l. Right, these are the only possibilities that can happen when you get these 2 strings, either both in the language or exactly one of them in the language or none of them in the language.

(Refer Slide Time: 5:07)



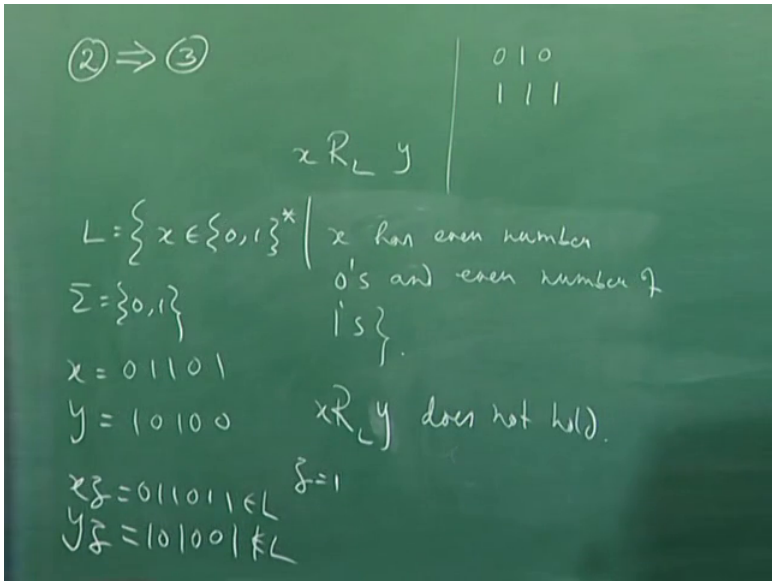


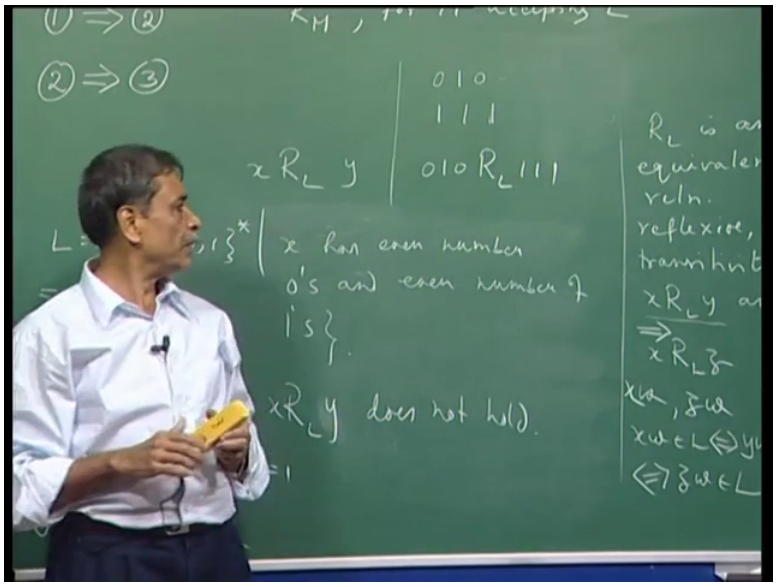
Now for x rl y to hold, these 2 cases must not be there. So you see for 2 strings x and y, if it is the case that for every z, when you form these who strings xz and yz, either both of them will be in the language or both of them will not be so language. So only in such a case we say that x is related to y. How do we check whether x is related to y, of course here is the definition but that is not the point right now, the point is to understand the definition. 1st of all let us quickly see that rl is an equivalence relation. See, reflexivity and symmetry are quite obvious, in case of reflexivity you need to check the condition for xz and xz of course you have only one string xz, and so therefore the conditions will be true, these things that we have written.

Symmetry is again see this the way the definition is, either both of them xz and yz in the language or none of them is in the language. So this definition the way we are saying is symmetric, so this is true. And finally transitivity, let us take, say, suppose that x rl y and y rl z. Right. And what i need to show that, i would like to show that if x rl y and y rl z, then x rl z. Now the reason for that is, you take any string w now, so x are z will be true if both the strings xw, xw and zw satisfies this condition. Now let us say xw is in the language, right. Then from this fact what do you get, since x is related to y, so suppose xw is the language implies yw is also in the language and now from the fact that y is related z, i get zw is also is a language, right.

And actually this is in fact an if and only if statement. So xw in the language because of this condition i can say if and only if yw is in the language and yw is in the language because of y being related to z if and only if zw is in the language. So what i have is xw is in the language if and only if zw is in the language, so x rl z. So it is clear that rl is an equivalence relation.

(Refer Slide Time: 9:11)





And just to fix our ideas we will just consider an old language that we had considered before, let us say l is a set of finite binary string such that x has even number of zeros and even number of ones, right. So now of course sigma is the binary alphabet 0, 1. Now consider x to be some string let say 01101 and y to be 10100. Will they be in the language? Or rather, of course we know that both of them 01101, this is not in the language because this has even number of zeros but odd number of 1s and this has odd number of zeros and even number of ones. So both are not the language, so that is okay and, but we would like to know if x rl z. So you see just consider the string, i am sorry x rl y, these 2 strings, x and y we are considering and we would like to know if x rl y holds or not for this language l.

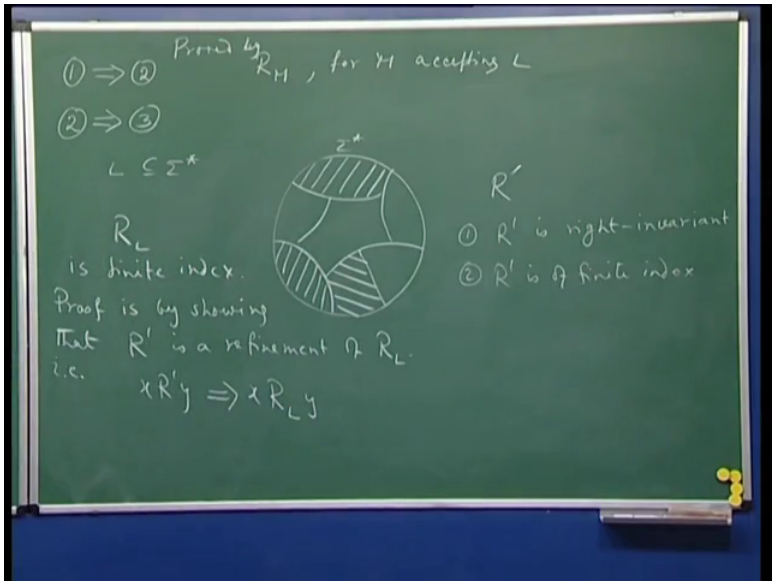
Now actually x rl y is not true, does not hold, why? Because just consider this string z to be 1, so when you get xz, that will be the string, so we write xz here, 01101 and then 1. Xz is in the language because it has even number of ones and even number of zeros, this is in the fridge and now take yz, yz is what, 10100 and then z again 1. This has odd number of ones and odd number of zeros, so this is not in the language. So that shows that these 2 strings 01101 and 10100, these 2 strings are not related by the relation rl for the language l.

On the other hand if you take another example for the same language l, that is the context, supposing i say 010 and 111, it is, you need to argue but you should be able to see, for example any string that you take, see the point is this, this has even number of zeros and odd number of ones, the string also has even number of zeros and odd number of ones, right. Now for this string to become a string in the language after augmenting some z, that z must have even zeros and odd ones, so that together they have even number of zeros an even number of ones. So now if you add the same string here, what will happen, this already had odd number of ones, you added through this string some more ones, so 2 or number of ones, when you add them, their number becomes even.

So the total number of ones will become even here and since it is even now zeros, only zeros earlier, so again that together, this entire string will have even number of zeros and even number of ones. So whenever this string is going to be in the language by adding some w, that same w also will make the string when added, this w when added to this string to the right, that string also will be in the language. And similarly we should be able to argue that any string, if it, you know if you take any w, if that w does not make the entire string in the language, that cases will be the case if and only if same w1 added will not make the string in the language.

So you see what i am trying to say for this language l. The 2 strings 010 is related by this relation r. Okay. So we are more or less clear about the relation rl and you should be able to see that this rl is defined for any language l whatsoever. Because you know, we just follow the definition, so and this is an equivalence relation, therefore it will partition sigma star in some equivalence classes. Now what the statement 3 is saying that this rl is of finite index, right. So let us prove this, that is 2 is true, that implies 3.

(Refer Slide Time: 16:07)



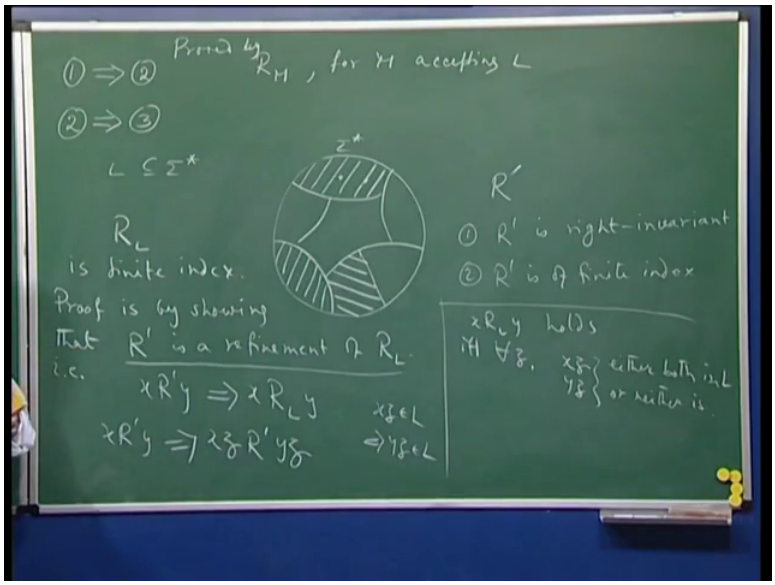
We now prove that the statement 2 implies statement 3. So you see statement 2 is saying that we have a language l in sigma star such that that there is, so this is sigma star, what this language l is? L is the union of some of the equivalence classes of an equivalence relation on sigma star satisfying these 2 conditions that that equivalence relation is right invariant and it is of finite index. So let that equivalence relation be r dash. Now 1st of all it says that one of the things it says that r dash is a finite index and r dash is an equivalence relation, so therefore it partitions the set sigma star in some manner, right. So these are the equivalence classes induced on sigma star by the relation r dash.

2 things we know already, one that the very fact that it is equivalence it is inducing this partition of course means r dash is an equivalence relation, we have made use of that, 2 other things is r dash is right invariant and r dash is a finite index. Which means the number of equivalence classes for r dash is finite. So the number of such equivalence classes is finite and now we are saying that l is the union of some of the equivalence classes. So the language is solely consists of you take some equivalence classes entirely, maybe this one also, this is just a picture of course.

So the statement 2 is saying that my language consists of the union of these strings and these strings and these strings. And now from here i would like to show that that language l which is the union of some of the equivalence classes of r dash where r dash is an equivalence relation which is right invariant and it is of finite index, that language l, for that language l if i consider rl, rl is a finite index, this i need to now prove and which will show the implication 2 to 3. Well the proof is by showing that any such r dash is a refinement of rl. We defined

refinement last time and to prove that r dash is a refinement of rl what i need to show that is x r dash y, that means whenever 2 strings are related by r dash, that means take any 2 strings in any one of these equivalence classes, that would imply these 2 strings are also related by rl. This is what i need to prove.

(Refer Slide Time: 20:59)



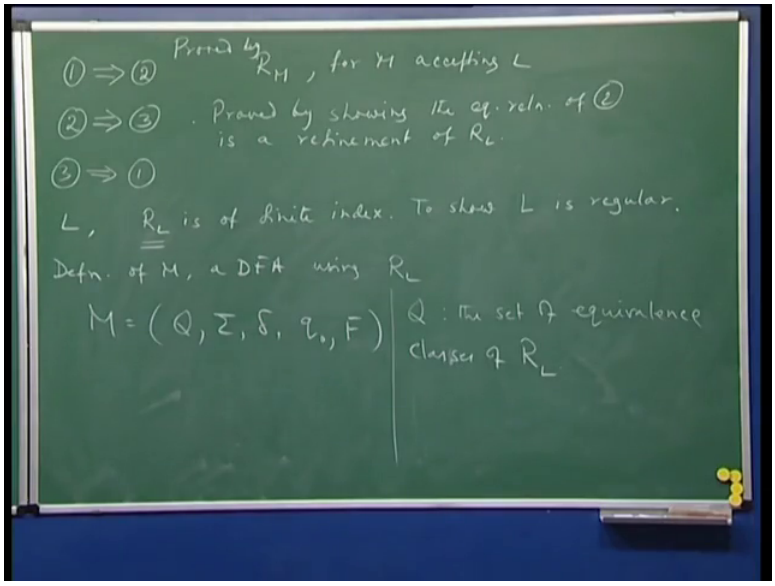
So of course i know from the definition, so let us that, this will be true if i can show, remember it is just a definition. X rl y holds if and only if for all z, xz and yz these 2 strings either both in l or neither is. Okay. So now consider the 2 strings xz and yz, right. Because r dash is right invariant i immediately have x r dash y. If i take any z, it is the case because the relation r dash is right invariant, what i have is xz is also related by r dash yz, that follows, that is something which follows directly from the definition of rz. Right. Now suppose xz is in the language, that means what that xz is either here or it is here or it is here or it is here or it is here because remember the language l is the union of some of the equivalence classes of r dash.

So suppose xz is here, now what is r dash saying? R dash is saying that x xz and yz, of course if they are related, that means yz also is in the same equivalence class. So if xz falls here, then we have assumed that xz is in the language so it is in one of these dashed equivalence classes. And now since it is related by r dash yz is also the same equivalence class. Which means if you assume xz is in l, that implies yz is in l, it cannot be the case that xz is in l and yz not in l. And of course this is true the other way also that if you assume yz in l, the same argument shows that xz will also be in l. Therefore if xz is not in l, yz also cannot be in l.

So that shows what, that whenever x is related to y by this r dash, i can no z such that exactly one of xz and yz will be the language, the other not in the language. Therefore clearly x is related to y by rl as well. And that proves that r dash is a refinement of rl. Now remember our notion of refinement. If something is a refinement of some other equivalence relation what does it mean, we discussed that last time, it just means that r dash at most breaks up some of the equivalence classes of rl into new equivalence classes. Right. In other words the equivalence classes of rl are made up of by, made up by combining some of the qualis classes entirely of r dash.

What does that mean, that number of equivalence classes of rl is less than or equal to the number of equivalence classes of r dash, right. Because r dash equivalence classes are made up by breaking apart some of the equivalence classes of rl. So now if says r dash is a finite index, that is what 2 says, rl also has to be a finite index. And that proves this that is ended the statement 3 that rl is finite index. So once more that our idea was this proof, we made this assumption and let r dash be such an equivalence relation, then we show that r dash is a refinement of rl and since r dash is a refinement of rl, number of equivalence classes of rl is no more than the number of equivalence classes of r dash.

(Refer Slide Time: 27:04)

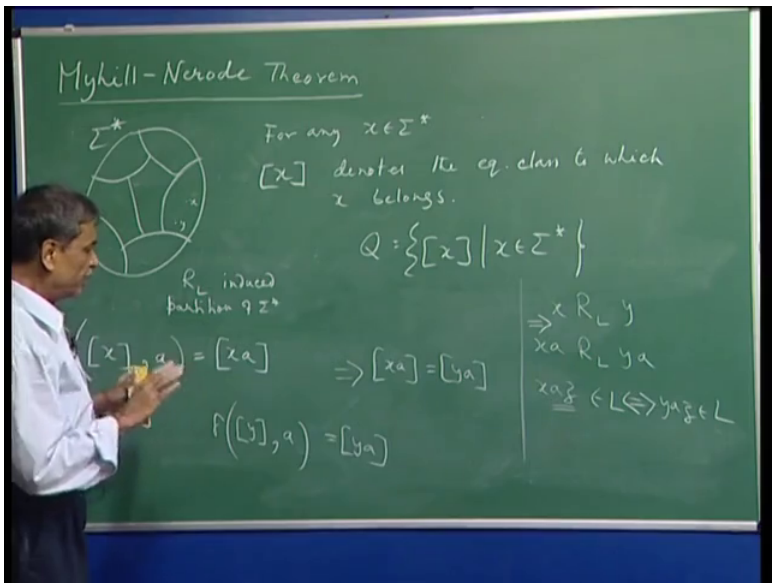


And since r dash itself was of finite index, that it has finitely many equivalence classes, rl will also have finitely many equivalence classes and therefore rl is of finite index. To complete the proof of myhill nerode theorem, we now need to show that 3 implies 1. The statement 3 implies 1. Now what is statement 3 saying, that i have a language l and the relation rl is of finite index. From there i would like to show, to show rl is, the language l is

regular. So from the assumption that rl is of finite index, we need to show the language l is regular. And the way we will show the language l is regular by actually constructing or defining the dfa for l using this relation rl.

So here is the definition of m, a dfa using the equivalence relation rl and we will show that this dfa m which we are going to define now will accept the language. So 1st of all my dfa m is going to be as usual q, sigma, delta, q0 and f, then i need to tell what this q, delta, q0 and f are, sigma is of course i know that sigma is the language, sigma is the alphabet l is defined over. Now we define q to be the set of equivalence classes of the equivalence relation rl. Now let me use this part of the board.

(Refer Slide Time: 29:36)



Remember that with our assumption is rl of finite index. Now rl is an equivalence relation on sigma star, the set of all strings over sigma and what more, rl is of finite index. So the number of equivalence classes rl induces on sigma star, that number is finite. Now what we are saying is that i will have an equivalence class standing for a state of my dfa. You might like to think of this way, that for every equivalence class, this is rl induced partition of sigma star and these are the equivalence classes, an example of course just a picture. And so i would like to say that these, i have state of dfa for this, for this, for this, for this. So in this case here there are 1, 2, in this picture there are 1, 2, 3, 4, 5, 6, 7 equivalence classes.

In such a case q will consist of 7 states, q will have 7 elements. Now i think you know what what each of these equivalence classes consist of. They consist of strings, right. Now how does one denote an equivalence class? One way of denoting is that take any element here let

us say x which is a string and you might remember that x denotes, so let me in fact write this, so for any x in sigma star, this is a standard notation that this denotes, this stands for the equivalence class to which x belongs. So my definition for q therefore is of this machine m, i can write q to be like this, in this manner. Okay.

Although it looks that for every x in sigma star i am doing something but the number of these equivalence classes is finite because that is what i have, statement 3 says that rl is of finite index, therefore there are only finitely many equivalence classes. So this set, the right-hand side set is finite and therefore we can take that set to be the set of states. Alright. So after that how do i define delta? Now remember the way we defined the transition function is, what was delta for any dfa, delta was a mapping from q cross sigma to q. In other words q, if, delta will look like this that it will take a state, it will take a symbol and it will tell me what does the dfa which states the dfa goes to from state q on symbol a.

We will define delta buyers i said the delta has 2 arguments, one is the state and one is the symbol. So states of this kind, equivalence classes of rl. So let us say i have this state and the symbol is a, right. Then we define it to be that state to which the string xa belongs. Now this definition looks alright, however we need to ensure one thing. Remember that equivalence class x consists of many strings, so suppose y is also here. So in other words the equivalence class x is same as the equivalence class of y. So the way we are doing it of course, that this equivalence class we are representing by the string x, you know while we write like this, this equivalence class we are representing by an element of the equivalence class.

And then we said that this is the equivalence class and the state corresponding to that is the state in which my machine will go if it is in this state and the symbol a comes. A natural question therefore is that since the state is same, whether you represented it as this or this, this is the equality, it is the same state because this is the same equivalence class, then this, for this definition to be what we call this definition to be well-defined. It must be the case that if i had taken some other representative for the class, let us say y in this case, then delta y and then a, of course by this definition i will get y a, they must be same, xa and y a must be same, only then this definition is meaningful, otherwise it is meaningless, right.

As it turns out that will be the case, why? Remember what do we know of x and y, that they belong to the same equivalence class of rl. So in other words, let me now write it here. I am trying to prove my definition, this is the definition of delta and that definition is done properly, it does not depend on which representative of the equivalence class i take. That is

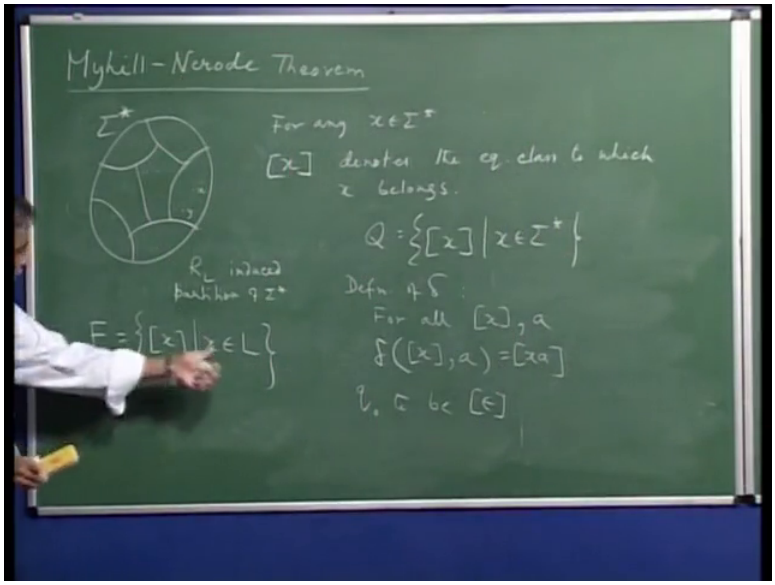
why i am saying in order to make this particular definition you know to convince you that this is a proper definition i need to show this that given that there is another string y in the same equivalence class, so that is why we are writing the equivalence class of x is same as equivalence class of y, then this is should show that the equivalence class of x a is same as the equivalence class of y a, given that x and y belong to the same equivalence class.

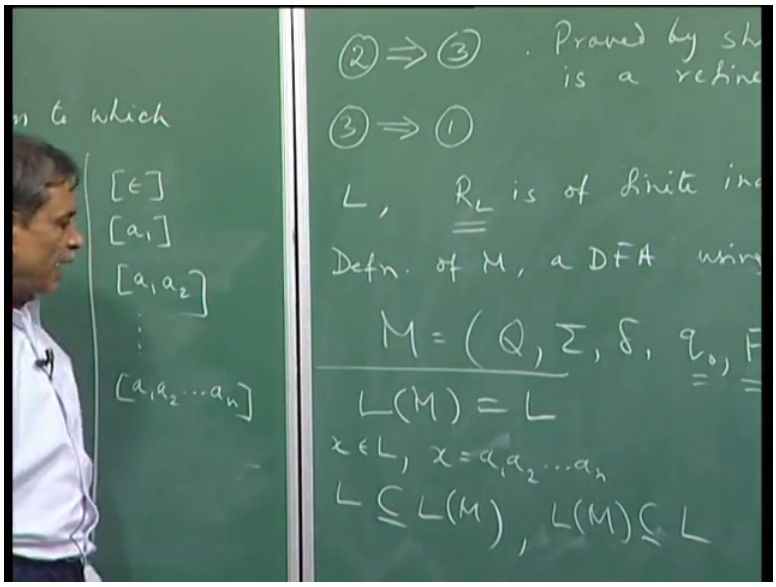
Now x and y belong to the same equivalence class means this, right. That is the definition, they are related by rl and this will be the case if x a also i can show to be related to ya. Now to show this that x a is related by the same relation y a, related by the same relation to ya, i need to show that for any z, x az and y az, whenever x az is in the language, this means if and only if y az is also in the language. Right. Now remember x az, x az in the language, now consider this as the entire string a az which you are appending 2, which you are concatenating through x.

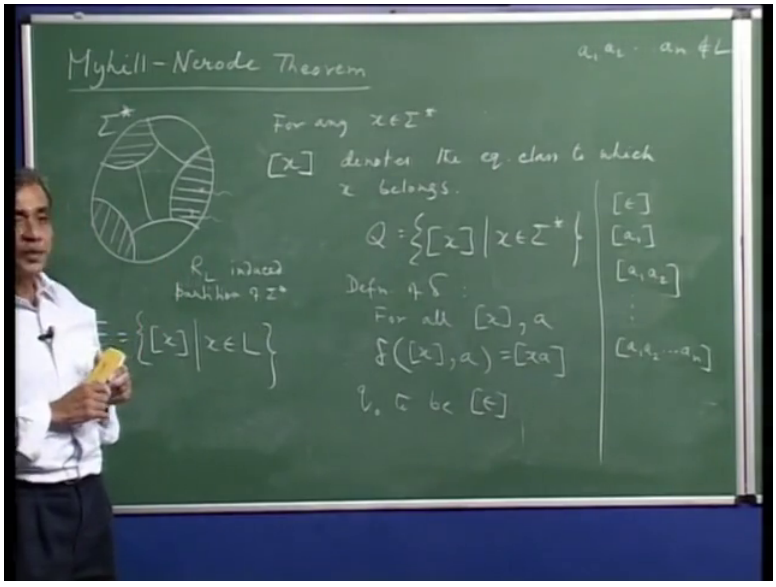
Now since x and y are related, x with this string appended, if that is in l, since x is related to y, then y az also will be in the language and vice versa. So therefore it is indeed the case that x rl y implies xa rl y a and therefore this definition is well, i mean it is well-defined, right, so we have done that. So my definition for delta is complete. And in order to show what the machine is, i need to provide the definition for q0 and s. Now what is q0, q0 is the state, initial state by the machine begins.

Now it should not be too difficult to see intuitively that if i take q0 to be that state of that equivalence class to which the string epsilon belongs. Remember epsilon is the empty string and epsilon will be in one of these classes, one of these equivalence classes, that equivalence class is the initial state of the machine. See we are you know we have considered that equivalence classes of rl, they constitute the states of the machine m, right. So please keep that in mind, so at one level we are talking of equivalence classes which are strings but at the other level we are thinking in terms of that this entire equivalence class is a state. It takes a little while maybe to appreciate this but really there is no difficulty, think of the way we have done it here.

(Refer Slide Time: 42:13)







At least q you can see there will be finitely many elements, delta just the way we have defined makes sense. So this also makes sense that q0 is something, q0 is the state of the machine m and so therefore it has to be an equivalence class of rl because that machine m, the states of the machine m articles classes of rl and we are defining the initial state to be that equivalence class to which the empty string belongs. Now we will see of course that this makes sense but before that let me complete the definition and finally the set s, f is a subset of q, remember, this the final states, so this i put it this way, all those equivalence classes such that…

So you see what we are saying, again x is in the language, of course languages in front, that is fine but you think of these strings which are in the language, there are only finitely many equivalence classes, so we will distribute to some of these equivalence classes. Those are the classes, they went thought of as states of the machine m are my final states of the machine for the services definition but does it make sense? 1st of all that this machine as we have as we have defined, does it accept the language l, that is what i need to show, right. We need to show that 3 implies 1 and in that starting from the definition of rl i define this machine m but ultimately what i need to show that this machine m the language accepted by this machine m, i need to show that this language is nothing but the language l.

Well look at it this way. So 1st of all it me show that if i take a string in the language, then that string will be accepted by this machine m. Let us see how. So suppose x is in the language and the string x consists of a1, a2, a n, these are the symbols, the string is of length n and this string x is consist of this symbol a1, a2 up to an. Now how does the machine behave on this string? Initially the machine is in the state epsilon, then the symbol a1 comes, right. Now look at the definition of delta, if it is in that state epsilon, then an, what is the next step, it will be in this state, the state to which the string in a1 belongs.

Then the machines m will be in this state, now a2 a, the machine will be in this state. By definition, definition of our delta. So now finally the machine after scanning all these symbols, the n symbols, it will be this state. This string a1 through an is the accepted provided this state is a final state. But of course that is the case because since this is an element, this string is in the is an element in l is a language is is one of the strings in the language, so you replace a1, a2, a n here, that is in the language, so therefore this state, this state will be a final state. So this proof is very simple, we just followed the definition and

what we see, finally we are in this state and that state is of course the final state. So this is accepted, right.

Now what else what actually i have shown is that this language l is a subset of lm, right because that is what i just showed that if you take any string in the language, that string will be accepted by that machine, right. Now what about the other way? I need to show that lm is a subset of the language l, one way of proving that is that if i take a string which is not in the language, then i should show this machine, the dfa that i have defined will go to a state which is not a final state, right. So you know this is like this, this, i am trying to show this part. So what i need to show that starting from x in l, i mean starting from x in lm i should show that the string x is in the language.

And now use the contrapositive of this statement that, this i can show, one-way of showing this is saying that if x is not in the language, i should be able to show that x is not in lm. In other words if i take a string which is not in the language, then i if i manage to show that string does not take the machine to a final state, then i am done. So again and again the thing is very simple, supposing that string, now let say a1, a2, a n is not in the language l, so how would the machine behave on this string? The same way that we have shown, at the end of this string when this string is scanned by my machine m, that will be in this state. But this is a string which is not in the language, right.

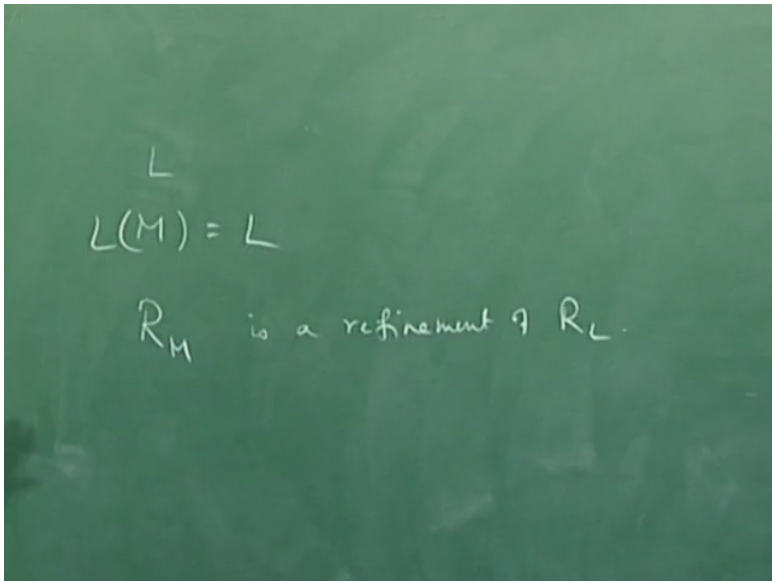
And so therefore this is not one of the final states this equivalence class which the string belongs is not one of the final states of the machine, because by definition final states of the machine is all those equivalence classes which are consist, which consist of strings which are in the language. Now you might wonder, then this is again, it looks kind of we are just waving hands and proving things but this proof is rigorous. You might wonder what if i have a string, i have an equivalence class of rl, like can it happen that x is in the language and y is not in the language? Cannot ever happen because consider this, just the string epsilon, x epsilon will be in the language but y epsilon will not be in the language and that would mean that x and y would not be the same equivalence class.

So really one of the ways rl, one of the implications of the definition of rl is the language l consists entirely of the union of, that is some of these equivalence classes we take their union and that gives me the language l. It cannot be that some string from here and some string from here will be in the language l. So that we can we can prove and therefore all that makes sense what we said. And that completes the proof that 3, the statement 3 of myhill nerode

theorem implies 1. Just to wrap up, we started with the assumption called for a language l to have rl which is of finite index, that means it has finitely many equivalence classes.

From there i defined a dfa and then i showed that dfa accepts the same language l, so therefore 3 statements, statement 3 was that rl is of finite index, statement one was l is regular and that is proved because my dfa that i defined this m that are defined, this is the definition of the m, that indeed accept the language l. And you see the nice thing about one of the many important things that it tells me is what, is that the notion of minimisation of states. Let us see why so. Myhill nerode theorem tells me what is the minimise automaton, what will be the minimise automaton, minimise dfa for a regular language l.

(Refer Slide Time: 52:11)



You see supposing l is regular, then that means that i have a dfa m such that lm, language accepted by that machine is the language l, all right by definition. And then if you consider rm, what do i know, that rm is a refinement of rl, that was a statement of the basically one implies 2, that is what we said. Now then that means what? Whatever machine that you may take, that machine, the states, number of states cannot be more than, sorry cannot be less than equal to the number of states of number of states of the dfa that we defined through rl. Once more, to recall that from the definition of rl gives me a dfa for m dash, for l.

And if you if you if you just go back to the statements of the myhill nerode theorem, what you know, what can see immediately that any machine m accepting l, the number of states in that machine has to be equal to or more than the number of states defined through rl dfa. Remember we just used a dfa definition from rl, now minimisation of states will be that

coming to the dfa which corresponds to rl. From rl that definition of the dfa that we got, that is indeed the best dfa in terms of number of states and this is what we will follow-up in our next lecture.