Theory of Computation

Prof. Kamala Krithivasan

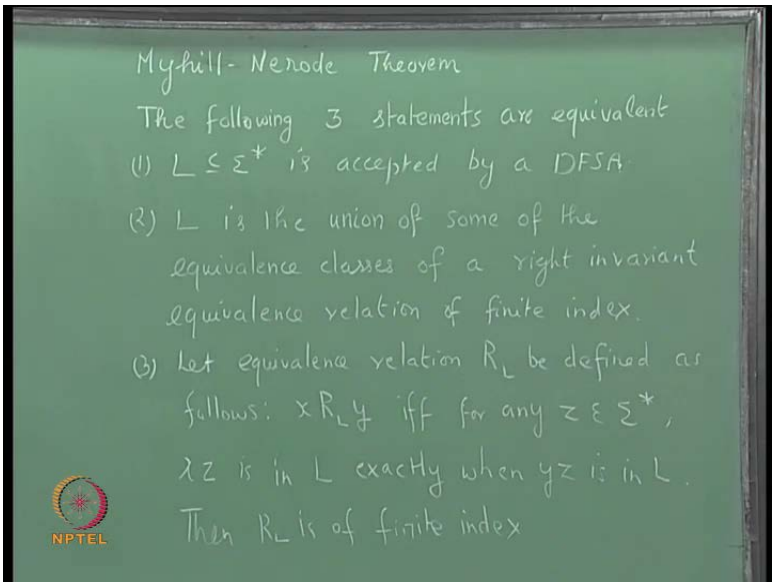
Department of Computer Science and Engineering

Indian Institute of Technology, Madras

Lecture No. # 18

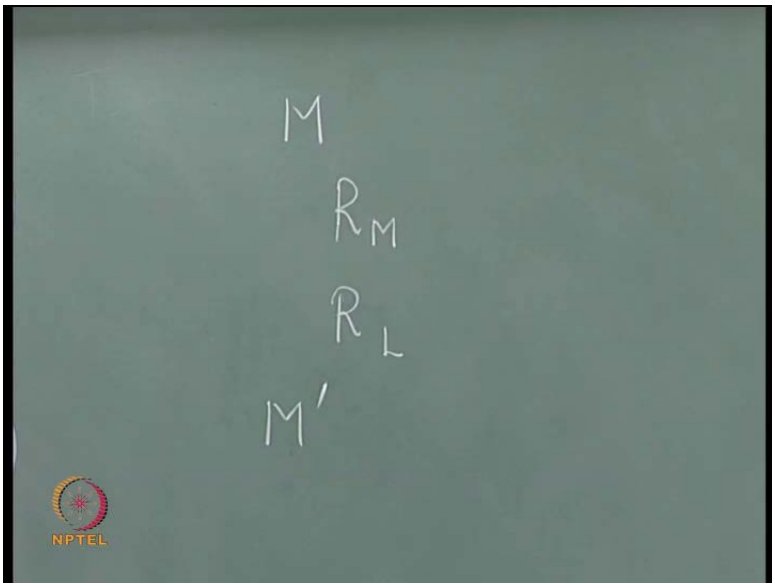
Minimization of DFSA

(Refer Slide Time: 00:21)



We have been considering Myhill-Nerode theorem; let us go through the statement again. Myhill-Nerode theorem states like this. The following three statements are equivalent. L contained in sigma star is accepted by a FSA, we can take as a DFSA. Second statement is L is a union of some of the equivalence classes of a right is invariant equivalence relation of finite index. And third condition is that equivalence relation R L be defined as follows. x R L Y if and only if for any z in sigma star x z is in L exactly when y z is in L, then R L is of finite index. Now, we have proved this result by showing that 1 implies 2, 2 implies 3, and 3 implies 1.

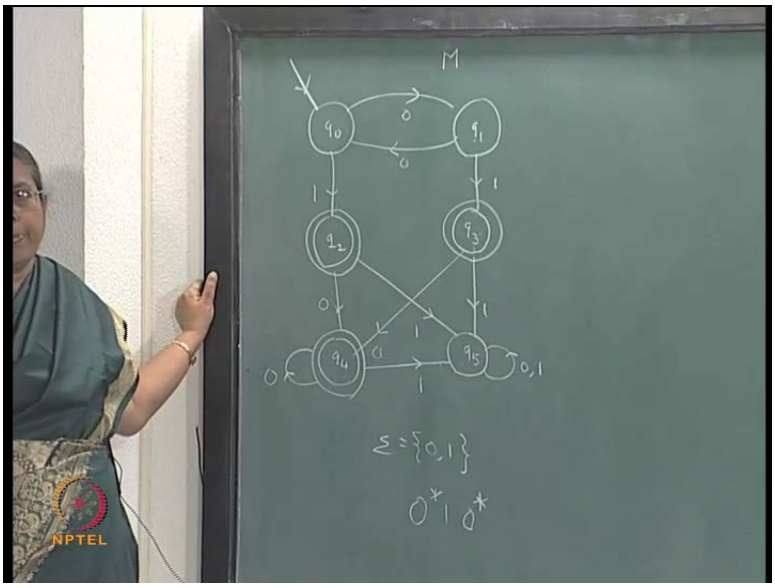
(Refer Slide Time: 01:08)



So, actually we started with a machine M and defined an equivalence relation R M on M. R M was a finite index it is right invariant. And L is a union of some of the equivalence classes of R M. Then we saw that if we take R L R M will be a refinement of R L. So, the index of R L or a number of equivalence classes in R L will be less than R equal to the number of equivalence classes in R M. Or index of R L will be less than or equal to the index of R M. Because this is finite index this will be finite index.

And from this you constructed a D F S A M dash for a third proof. That is 3 implies 1 from the relation R L you constructed a machine M dash corresponding to each equivalence class of M dash. You had a straight in M dash and equivalence class, which contains the empty string, is the it corresponds to the start state and if x is in L then an equivalence class corresponding to that will be a final state here that is what we read.

(Refer Slide Time: 02:35)

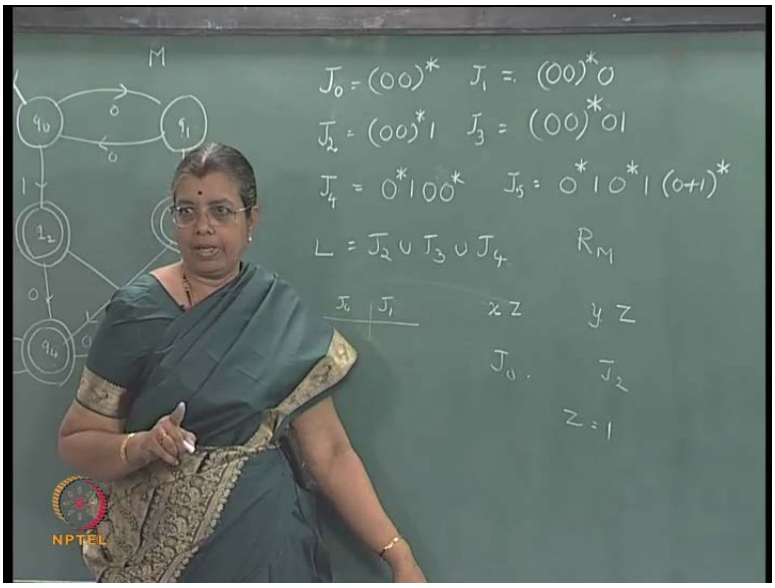


Now, we illustrated that with one example let us take one more example here look at this state diagram it is a D F S A the alphabet is 0 1 here. Now, what sort of strings will be accepted by the machine q 2 q 3 q 4 are final states. So, starting from q naught you can read any zeros and if you go to a if you read a 1 you go to a final state. Then if you read some more zeros still you remain in a final state. But if you read 1 one extra you go to a non final state.

Once you go to q 5 you keep on remaining in the q 5 and no string will be accepted q 5 is a non final state. So, what sort strings will be accepted, any string having just exactly 1 1 if it does not have any 1 also it will not be accepted. If it has got 1 1 it will be accepted it can be followed by some zeros also. But if it has got two ones it will not be accepted 2 or more ones it will not be accepted. So, actually it corresponds to 0 star 1 0 it corresponds to this regular expression the language accepted corresponds to the regular expression 0 star 1 0 star.

Now, if you consider this as A M by the machine M M induces an equivalence relation on sigma star. What is that equivalence relation the set of strings which take you from q naught to q naught is 1 class. q naught q 1 a 1 class q naught q 2 is 1 class q naught to q 3 is 1 class and so on.

(Refer Slide Time: 04:30)



So, 1 class J q naught I will write it as J q naught or may be J naught simpler J naught is a equivalence class which contains a strings which take you from q naught to q naught.

What sort of strings will take you from q naught to q naught q naught to q naught epsilon will be there. Than 0 0 0 0 0 0 will take you to q naught even number of zeros will take you to q naught. So, J naught contains the strings corresponding to 0 0 star even number of zeros. J 1 J 1 is a set of strings which take you from q naught to q 1. What sort of strings take you from q naught to q 1 odd number of zeros 0 or 0 0 0 0 0 0 0 0 and so on. So, J 1 corresponds to 0 0 star 0 I mean the. I am writing the instead of writing the language I am writing the regular expression for that.

And what is J 2 J 2 is a set of strings which take you from q naught to q 2 what sort of strings will take you from q naught to q 2. 1 takes you and any even number of zeros followed a 1 will take you to q 2. So, that can be represented by the regular expression 0 0 star followed by a 1. And J 3 is a set of strings, which take you from q naught to q 3. What sort of strings will take from q naught to q 3? Odd number of zeros followed by a 1. So, that will be 0 0 star 0 1 and what is a equivalence class corresponding to q 4 that is set of strings which take you from q naught to q 4 J 4. That is after getting an even number of zeros or an odd number of zeros you get a 1 and then you get 1 more 0.

It can be followed by any number of zeros. So, J 4 corresponds to you can have odd or even number of zeros it does not matter. Then 1, then at least 1 0 you can have any zeros

followed by a 1 and then at least 1 0 it can be followed by negative. J 5 is a set of strings which take you after reading 2 one b s you will go here in between you can read zeros also. So, it will be 0 star 1 then you can read some more 0 stars I mean some more zeros and another a 1. So, if you read number of zeros and get a 1 you will go here, then you can read zeros or you need not read zeros if you read a 1 one you will go here. Then after words you may read 10 or 1 does not matter.

(No audio 09:06 to 09:20)

Now, look at the set of strings on sigma star they will belong to any 1 of this class take any binary string it will belong to one of these classes. For example, if we take 0 1 0 0 1 something like this to which class it will belong 2 one s means it will belong to J 5. If the string contains 2 one s or more it will belong to J 5 if it does not contain 1 it will belong to this if it contains only 1 one it will belong to 1 of them. If I take for example, this it will belong to this class. Now, what is L? L contains these three classes these three are the final states so L is a union of the equivalence classes corresponding to these three.

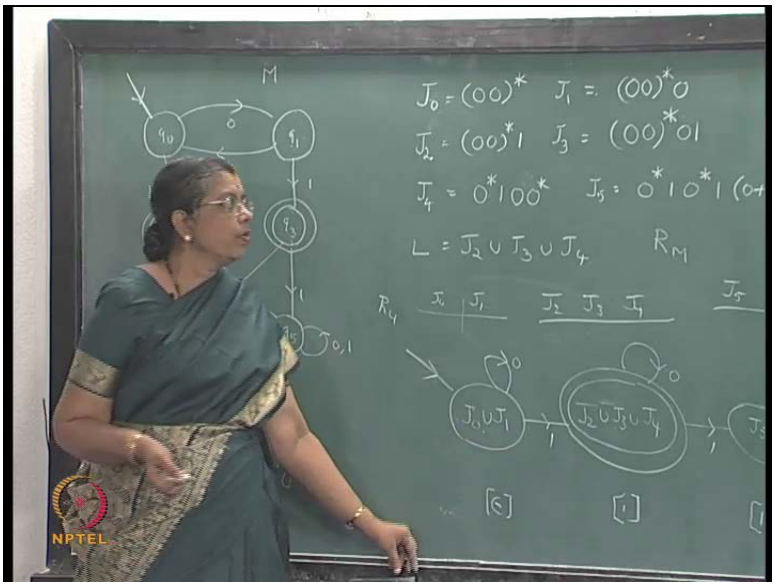
So, L is J 2 union J 3 union J 4. Now, according to R M that is this is a machine M and we have defined R M as set strings which take you from initial state to a particular state. The each 1 corresponds to a equivalence class. So it has got 6 equivalence classes like this and R language is the union of three of them. Now, consider R L R L will be a refinement of I mean R M will be a refinement of R L. So, some of these equivalence classes can be merged in R L in R L somehow the equivalence classes can be merged. See which of them can be a merged. You see if you take a string from J naught or J 1 it will be of the form a sequence of zeros.

If you take a string which belongs to the equivalence class for this or J naught and if you take a string from J 1 they are sequence of 0 whether it is odd or even is the difference. If it is followed by a string, which has 1 one you will accept it. See I take 2 strings x y 1 belongs to J naught another belongs to J 1 that means x is even number of zeros y is odd number of zeros. If I take a z like this then if z contains only zeros it will not be accepted. Whether it is x z or y z both will not be accepted if z contains just 1 one both of them will be accepted. If z contains two ones or more one s both of them will not be accepted.

So, whatever may be z x z and y z will both be accepted or both be rejected is that clear. So, you can in R L if you consider R L, J naught and J 1 you can put in the same equivalence class. And what about J 2 J 3 J 4 suppose I take 1 string from, I will consider J naught and J 2 now J naught and J 2. x belongs to J naught y belongs to J 2 that means x is a string of zeros alone y contains 1 one. Now, if I take z and I take z to be 1 x z will be accepted because it has 1 one y z will not be accepted because it has got 2 one s. So, x z can be accepted y z need not be accepted will not be accepted. And I choose z in such a way that 1 is accepted the other is not accepted.

So, that does not satisfy the condition of R L is not it. So, z J naught and J 2 cannot be in the same equivalence class of R L you cannot merge. Similar argument holds for J naught and J 3 J naught and J 4 J 1 and J 2 and so on you cannot combine any of this with this.

(Refer Slide Time: 14:58)



But look at J 2 and J 3 J 2 and J 3 take a string x and y. x has 1 one y has 1 one so if you take z, if z has zeros alone both x z and y z will be accepted. If z contains 1 one both 1 one or more one s x z and y z both will not be accepted. So, whatever way you choose that either x z and y z will both be accepted or x z and y z will not be accepted. So, they will be in the same equivalence class a similar equivalence similar argument you can give for J for also. So, actually in R L J 2 J 3 J 4 will be merged.

And J 5 is of course, at if you take 1 string from J 4 and J 5 say for example, J 4 if you take and J 5 if you take, take x and y. This has got 1 one this has got 2 one s or more one s if I take z such that z is equal to 10 x z still has 1 10 y z has more zeros so x z will be accepted y z will naught be accepted. So, you cannot set of put J 4 and J 5 in the same equivalence class of R L they will be different.

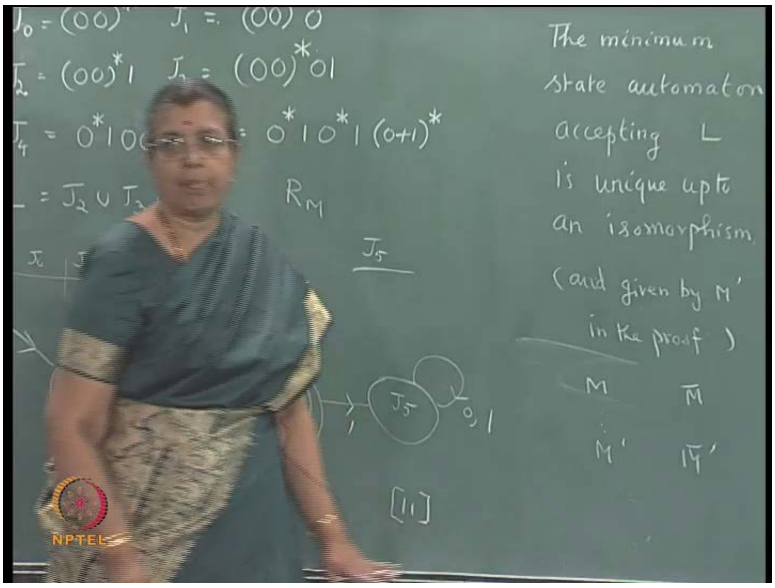
So, J 5 forms a separate class to all together. So, according to R L there are only 3 equivalence classes. Strings which do not have one s strings which have 1 one s string which have 2 or more one s. So, if you put them as a J naught union J 1 J 2 union J 3 J 4 J 5, and epsilon will belong to J naught the epsilon belong to this that will be the initial state. And J 2 J 3 J 4 contains the final states or a strings in them are accepted so that will be the final state. The transition diagram will be like this, as long as you read 10 you are here if you read 1 you go here if you read 1 more 1 you go here.

(No audio 18:12 to 18:25)

How did I get this transition diagram? By the way you have define see this has epsilon in that. Actually epsilon is the equivalence class containing epsilon is this. The equivalence class containing, 1 is this the equivalence class containing 2 one s is this. So, what is the equivalence epsilon concatenated with a I am sorry 0. That is again here now the same equivalence class so this R goes here. What is epsilon concatenated with 1? One that equivalence classes this so this R goes here and with 1 if you concatenate 0 1 0 to which equivalence class 1 0 belongs it will be here. So, 0 goes here if you concatenated with 1 and 1 one 1 belongs to this class so arc 1 goes here.

1 one 0 belongs this 1 one 1 also belongs to this class so that R goes like that. This is the way we have constructed the minimum state this is the minimum state automaton. This is also a deterministic automaton this is also a deterministic automaton, but this is the minimum state automaton.

(Refer Slide Time: 20:14)

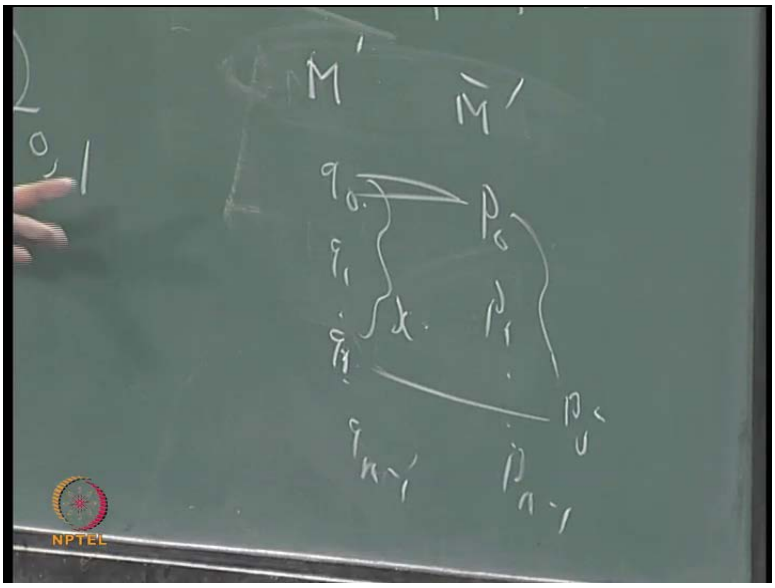


Ok, so you have the result the minimum state automaton, accepting a language a regular set up is unique up to an isomorphism. And actually it is given in the theorem and given by M dash in the proof in the proof of Myhill-Nerode theorem. So, in the Myhill-Nerode theorem is started with M and N constructed M dash for R L.

And that gives you the minimum state automaton. What is an isomorphism? Renaming updates instead of calling them as like this I can call it as some q naught q L or P naught P 1 P 2 something like that that is what it means. Now, how are you justified in sayings that how are you justified in saying this. First of all in the theorem if you look you started with L accepted by M constructed the relation R M and we constructed R L and then M dash this is the way the proof find. Now, we saw that R M is a refinement of R L so the number of equivalence classes in R L is less than R equal to the number of equivalence classes in R L.

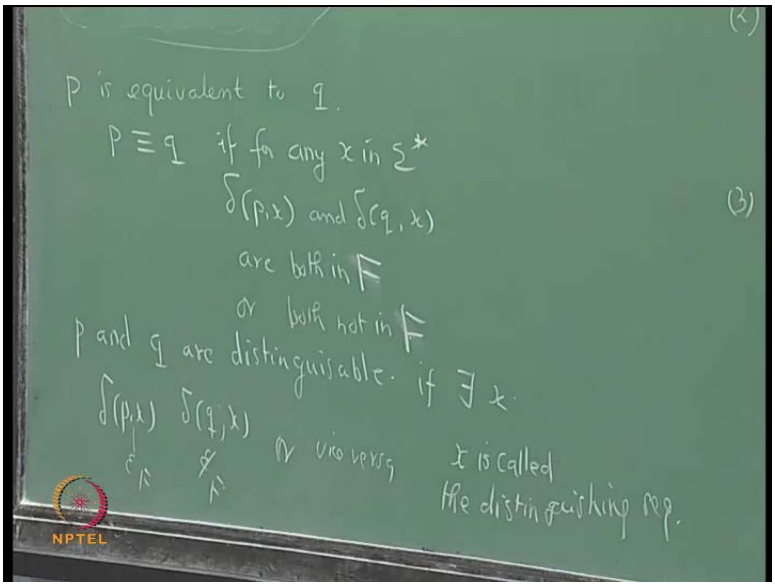
So, any d F a if you start with you can reduce and have machine corresponding to R L. And if two have the same number of states, you can say so one has same number of states. Suppose I start with this I start with M another one say M M bar say then I reduce according to R L and get M dash here M bar dash both are R L for L. So, actually they will be the they will have same number of states, which is equal to the number of equivalence classes of R L. Now, you can identify each 1 of them.

(Refer Slide Time: 22:57)



I have 2 automaton M dash and M bar dash this has got q naught q 1 q n minus 1 say this has got P naught P 1 P n minus 1.Now, the initial states correspond to each other and if a string x you from q naught to q 1 another string x will take you from P naught to some P J. And q y corresponds to P J, like that each state can be identified with another state. So, that will be this identification is consistent your identif[ier]- your making a correspondence between the states of M dash and M bar dash. And this is consistent because they are all depending due to the equivalence classes of R L. So, the minimum state automaton is unique up to an isomorphism.

(Refer Slide Time: 24:14)

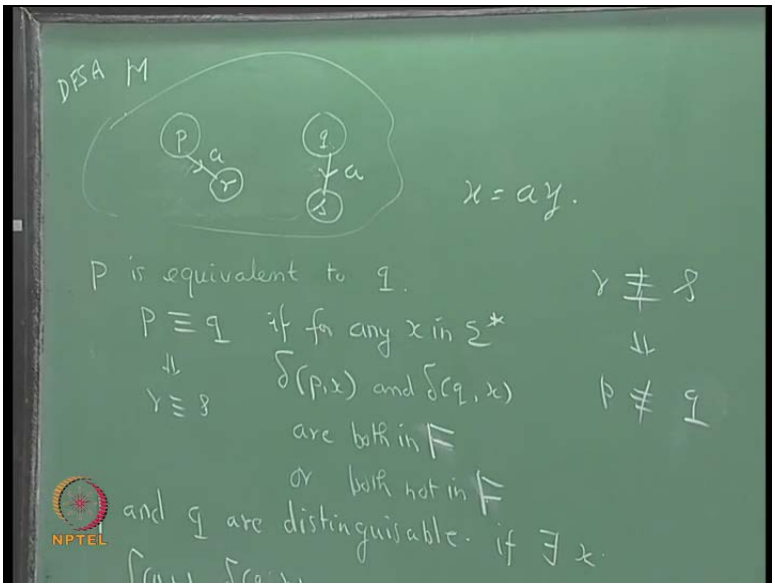


Now, we learn a few more definitions, I will use that example again. Now you see that you are given DFSA M and in that you are having two states P q two states. You are having in the diagram state diagram of M you say P is equivalent to q. You say you define like this you say P is equivalent to q or you write as P if for any x in sigma underlying alphabet. Delta of P x and delta of q x are both n L are both not in L. This is why how we defined R L similar to defining R L. Two states P and q are equivalent if you take any string starting from P after reading x you go to state starting from q after reading a particular string you go to state another state.

Both of them should be final states or both of them should be non-final states or are both in L both in F or both not in F final state. Delta P x is a state that is what this meant. Starting from P after reading a string x you go here starting from q after reading x you got to another state both of them should be final states or both of them should be non-final states for any x. Then you say P and q are equivalent. (No audio 26:30 to 26:40) P and q are distinguishable, if there exist a x if there is a x such that delta of P x and delta of q x if we consider.

This may belong to F and this does not belong to F 1 of them may belong to F other may not belong F or vice versa. This may belong to F this may not belong to F if there is a string x which takes you from P to a final state. But takes q takes you from q to a non-final state or the other way around then the states are distinguishable. And x is called a distinguishing x is called the distinguishing sequence x is called a distinguishing sequence.

(Refer Slide Time: 28:14)



Now, look at this example, you take q naught and q 1. If you get 1 or if you get string which has one one it will take you to final state. From both otherwise it will take you to a non-final state. So, any string x you take either it will take you from q naught any string x if you take either it will take you to a final states.

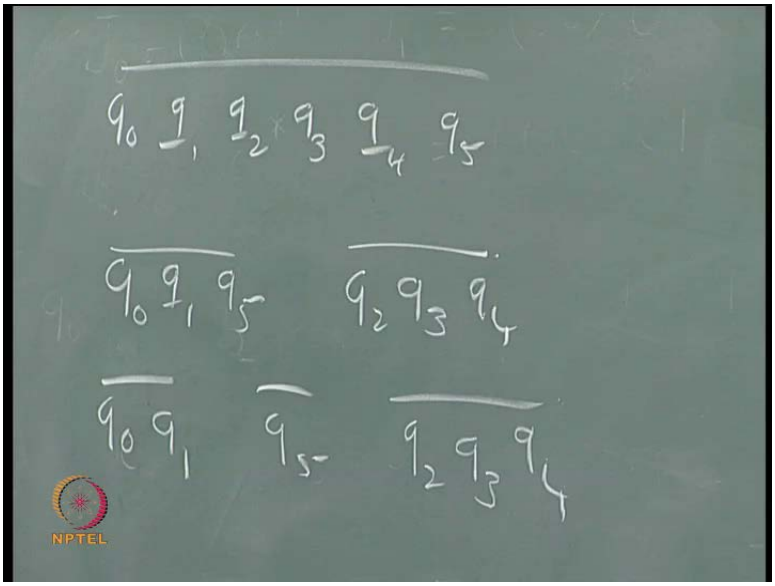
Starting from q naught or q 1 or it will take you to (( )) so they are equivalent. Similarly if you consider q naught and q 2 they will not be, from q naught if you get a 1 you go to a final state. From q 2 if you get a 1 go to a non-final state. So, 1 is the distinguishing sequence here q naught and q 2 are distinguishable and, 1 is the distinguishing sequence. Now, coming back here P is equivalent to q whenever you if you take any string x it should take you to a final state and this will take you to a final state. Now, I have 1 symbol suppose some a it takes meet to state R. The same a from q it takes you to s. Now, if P and q are equivalent what can I say about R and s?

They have to be equivalent any string takes you to a final state or takes you to a non-final state in both cases. So, if it begins with a it will go to R and then it will be followed by the x you are writing as some a y that is all. So, if a y is x from P if a y is accepted from q a y will be accepted that is from R y will be accepted from s y will be accepted. So, R and s will be equivalent, so if P is equivalent to q you would imply R is equivalent to s. The other way around if R is not equivalent to s or controversy if R is not equivalent to s

there is a distinguishing sequence which takes you in one case to a final state in another case to a non-final state.

That means that string y which takes you in one case to a final state in another case to non-final state. Consider a y in one case it will take q 2 a final state and 1 case it will take you to a non-final state. So, if R and s are distinguishable P and q will be distinguishable. So, the other way around R are distinguishable I am writing it not equivalent that would imply P and q are distinguishable. So, this is the concept so we have seen in this case or it is a class equivalence class how we can group the equivalence class. And so on so for every example we cannot do that to get minimize minimum state automaton.

(Refer Slide Time: 32:23)



The shortcut is like this are with have all of them q naught q 1 q 2 q 3 q 4 q 5 in 1 block. Then separate them into 2 blocks having final and non-final states. So, q naught q 1 q 5 the possibilities these can be equivalent then you keep on separating them so that only equivalence equivalence state come in 1 block. You partition you are going to partition q 2 q 3 q 4 this is 1 block why? Epsilon takes you from this to the final state; epsilon takes you from this to a non-final state. So, if we take 1 from this 1 from this they are distinguishable epsilon is the distinguishing sequence.

These are the final states these are the non-final states. And epsilon takes you from each 1 of them to a final state epsilon take you from each 1 of them to a non-final state. So,

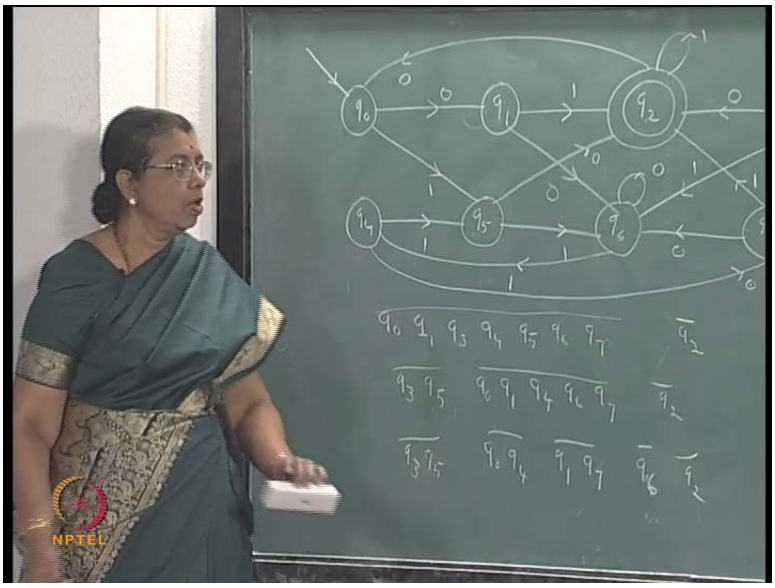
they are distinguishable so this is a first separation. Now, look at this q 2 q 3 q 4 q 2 q 3 q 4, after getting a 0 from q 2 or q 3 or q 4 where do you go q 4 the next state is the same in this in this example it is so happen it is the same. So, after going to q 4 either it will be accepted or not accepted and so on is not it and this is a final state. And after reading a 1 where do you go from all 3 states you go to the same state.

So, further (( )) of this is not possible it is not possible to further divide this block. Look at this block q naught q 1 q 5, from q naught or q 1 if you get a zero where do you go to the same block q naught or q 1. From q 5 if you get a zero you go to q 5 same block so from q naught or q 1 or q 5 what are the zero successes. Zero successes means the states which you go after reading a 0 so they that is q naught q 1 q 5 only they are in the same block. So, nothing happens now look at 1 look at the 1 success what are the 1 success? Of q naught q 1 q 5 q naught and q 1 the 1 successor are q 2 and q 3 for q 5 it is this.

So, they are in different blocks q 5 q 2 q 3 they are in different blocks so you have split them you split them q 2 q 3 and q 5 are distinguishable. So, q naught q 1 should be distinguishable from q 5. Is that clear q 2 q 3 q 4 now we have seen that this cannot be further divided this is a single state. Now q naught and q 1 look at q naught and q 1 what are the zero successes q naught and q 1 the same block, what are the 1 successes q 2 and q 3 they are in the same block. So, further division not possible. So, you get only 3 states these 2 states can be grouped together they are equivalent q naught and q 1 are equivalent.

You can group them together q 2 q 3 q 4 or equivalent you can group them together this a idea.

(Refer Slide Time: 36:57)



One more example we will take

(No audio 36:29 to 36:55)

q naught q 1 q 2 q 3 this is the final state and this is the initial state, q 4 q 5 q 6 q 7. Now, the transition are marked as follows, 0 1 1 0 0 1 0 1 0.

(No audio 38:06 to 38:49)

This is a state diagram of a machine we want to find out the minimum state automaton.

(No audio 38:59 to 38:15)

Let us find the minimum state automaton in this procedure. So, all of them will be in 1 block first therefore, only 1 final state and others are non-final. So, first you split in to non-final and final state. So, q naught q 1 q 3 q 4 q 5 q 6 q 7 are in one block q 2 alone is in one block. Now, look at the zero successes of them the zero successes, of q 3 and q 5 q 3 and q 5 are in this block. The zero successes of q 3 and q 5 are in this block, but the zeros successes of q naught q 1 q 4 q 6 q 7 are in this block. So, you split them the zeros successes of q 3 and q 5 are in this block. So, you put q 3 and q 5 separately and q naught q 1 q 4 q 6 q 7 in one block.

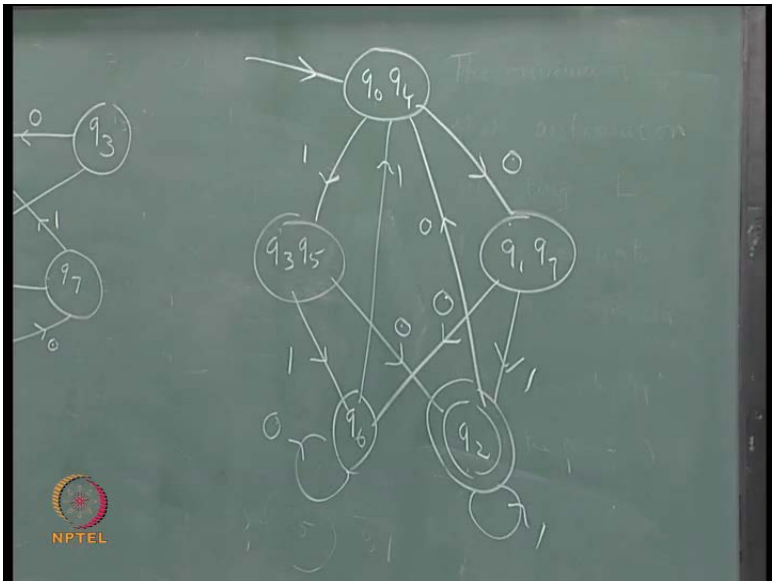
Keep on doing this whether further split is possible, q 3 and q 5 what are the zero successes of q 3 and q 5 q 2? What are the one successes q 6 same state? So, further split

here is not possible look at this q naught q 1 q 4 q 6 q 7. What are the zero successes of them q naught q naught q 1 q 1 is q 6 in the same block q 4 is q 7 the same block q 6 same block q 7 the same block. But consider the one success what are the zero successes of q naught q 1 etcetera. The one successor of q 1 and q 7 are in one block. The one successor of q naught and what is q 4 q 4 are here in this block. The one successor of q 6, is here. See the one successor of q naught and q 4 they are in one block.

So, q naught and q 4 you can split the one successor of q 1 and q 7 are in this block. q naught q 4 the one successor of q 6 is in one block q 6 q 3 q 5 so this is split in to this like this. Now, you have to see this for this no for this split is possible these are single states. So, q naught and q 4 q 1 and q 7 whether for the split is possible you have to see q naught and q 4 what are the zero successes. q 1 and q 7 they are in the same block what are the one successes same state q 5.

So, at this stage you cannot separate them out q 1 and q 7 see whether they can be separated q 1 and q 7 what are the zero successes? The zero successes are 6 q 6 and one success are q 2 same state.So, you cannot split further so further split is not possible in this case, so the minimum state automaton will have 5 states where q 3 and q 5 will be merged q naught and q 4 will be merged q 1 q 7 will be merged and so on so if you draw the diagram for minimum state automaton.

(Refer Slide Time: 44:54)



(No audio 44:39 to 44: 50) q naught q 4 will be 1 state, q 3 q 5 will be one state, q 1 q 7 will be one state, q 6 will be one state, q 2 will be one state. Which will be the initial state the 2 1 2 which q naught belongs so this is the initial state q 2 is the only final state. Now, draw the transitions q naught is q 1 q 4 0 is q 7. So, if you get a 0 you go to this from here if you get 1 q naught 1 is q 5 q 4 1 is q 5.

So, this get 1 you go here from this q 3 q 5 q 3 q 5. If we get a 0, you go to q 2. If you get 1, you go to q 6 from q 1 q 7; if you get 1, you go to q 2; if you get a 0, you go to q 6. From q 6 is to get a 0 you go to q 6 if you get 1 you go to q 4 from q 2 if you get 1 you go to q 1 if you get a 0 you go to q naught. So, this is the minimum state automaton corresponding to this. So, given any machine we can find the minimum state automaton in a systematic manner. The idea of these distinguishing sequences is very useful when you apply the idea of final state automata to computer networks.

You expect the system to be in a particular state. The whole network you are you it is in a particular state after certain things happen. Some messages and from 1 to another or some from this to another and so on tell be the whole system will be in a another state. Then after some certain things happens it will be an another state. So, the state of the status of a network can be represented as a state of a final state automaton. And you start with some initial configuration, then after something you expect it to be in a particular state. You expect the whole system to be in a particular status.

But you do not know whether that will happen or not by experimenting you find that it may go to a different thing so you want to check. So, for that you have to give proper inputs and check whether what is happening to the system whole network. Send messages from one to another or send package from this to other and so on. But what is that you are going to do what sort of thing you are going to do how many times you have to test. That really corresponds to you have to find out the distinguishing sequences. If you represent it as a network from one state to another which is the distinguishing sequence.

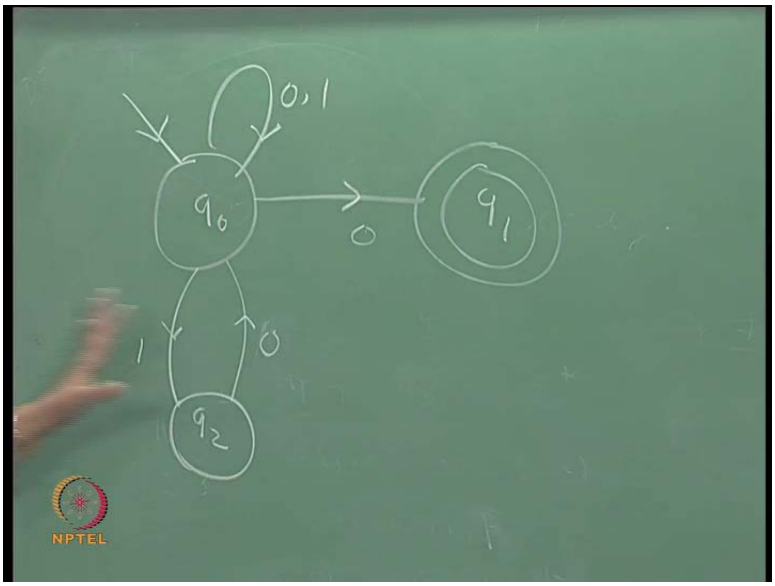
And we have to experiment only for those distinguishing sequences. We need not have to go through everything only for those distinguishing sequences you have to find out whether the system behaves as you want it to be or it is behaving in a different way. So, me sort of that idea it s what I am telling you may not be the exact thing. But a similar

idea is used in for finding out whether the computer network the system behaves in a way you want it to behave or it behaves in a different way. And puts your into trouble ok. So, lots of finite state automata ideas are used in that context.

Now having then the minimization of deterministic automata we have considered deterministic finite state automata. And minimized in fact if you remember what we did in the earlier classes. We considered a non-deterministic FSA having two zeros in a middle or having two one s in the middle and then constructed the DFSA that was not difficult. But the strings yeah that we ended up with nine states or something like that by subset construction. But the nine state can be minimized and you get only 4 states and those with ended with 2 zeros or ended with 1 2 one s again we constructed.

And we got the minimum at that end the construction itself gave you the minimum state automata. But the subset construction as it is may not give you the minimum state automaton, but you can minimize that. So, what we did is from the non-deterministic F s a, you constructed the deterministic FSA, which had many states and then you can minimize that using this procedure. You will get the minimum state automaton. Now, given a non-deterministic FSA can you apply this procedure? Can you apply the equivalent states and merge the something let us see it is not possible really.

(Refer Slide Time: 50:57)



So, let us see this example, q naught.

(No audio 51:01 to 51:48)

See what is a language accepted by this machine it is a non-deterministic machine because from q naught if you get a 10 or 1 you can go here. From q naught if we get a 0 you have can go here or here from q naught if we get 1 you can go here or here. So, it is a non-deterministic FSA. What sort of language will be accepted this is a final state? So any string ending with a 10 will accepted any binary string ending with a 10 will be accepted. Now, we if you try to minimize this automaton q naught and q 1 it is a final state this is not a final state they cannot be equivalent. And among this q naught takes you to the final 0 takes you from q naught to a final state from here to a non-final state.

So, these two cannot be equivalent. If you applied try to apply the minimization procedure find out which states are equivalent this being a final state cannot be equivalent to any of them. And these two 0 takes you from here to a non-final state from here to here final state. Because the 0 takes you from here to q naught and give one two possibilities are there. But anyway they are not equivalent they are not equivalent. So, you cannot minimize this but, is this the minimum state non-deterministic FSA, no this is not at all necessary. The minimum state n FSA, is just this.

So, this minimization procedure does not work for non-deterministic FSA, it will not work so this is about minimization of DFSA. Next we have to consider final automata with output which we have earlier considered in the first class like a serial adder parity checker and so on. So, in that case also we can have a set of minimization procedure. But what you would like to consider is how we define the output does that output depend on the state alone or this output depend on the state and the input.

We have to check depending upon that we call it as a Moore machine or mealy machine this will shall consider in the next lecture yes. (( )) State q 3 is not reachable from q naught that is correct. So, before you start the minimization procedure you can remove the inaccessible states. But in this case q 3 merges with q 5, but q 5 is reachable so it does not create any problem starting from q naught q 3 will not be reached, but q 5 will be reached.