Theory of Computation

Prof. Kamala Krithivasan

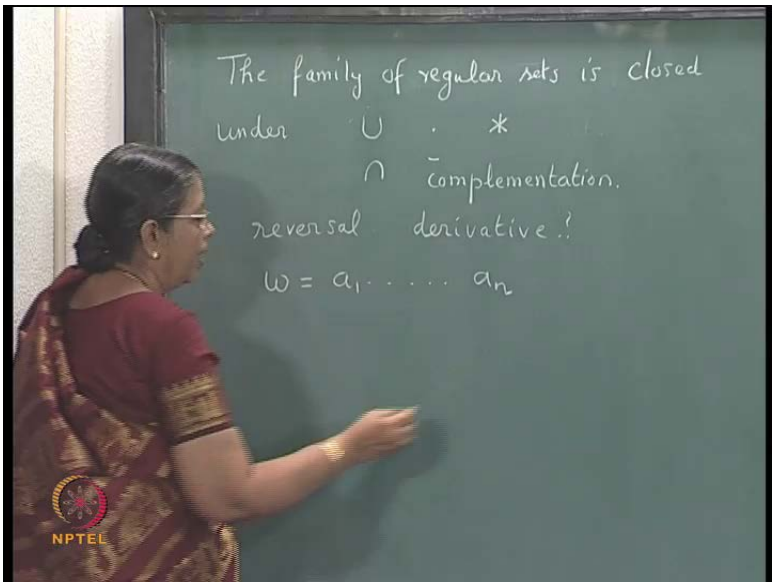
Department of Computer Science and Engineering

Indian Institute of Technology, Madras

Lecture No. # 14

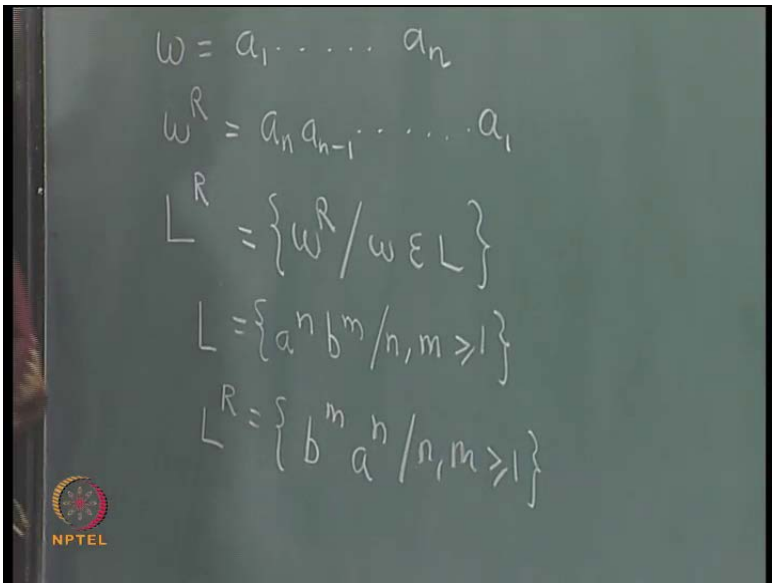
DFSA to Regular Expressions

(Refer Slide Time: 00:15)



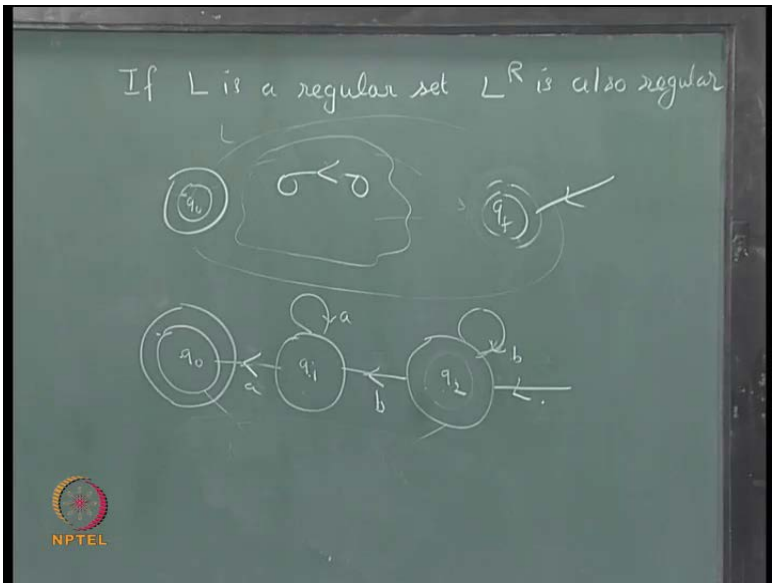
So, we have seen that the family of regular sets is closed under the following operations. The family of regular sets is closed under union concatenation and star, this we proved using regular expression. Then we also saw that it is closed under inter section and complementation. (No audio from 00:47 to 00:59) Usually these three operations considered together, and these three operations set theoretic operations are considered together. So, any way this union is counted in both the sets.

(Refer Slide Time: 01:49)



Now, we will consider two more operations - reversal and derivative with respect to a string. (No audio from 01:20 to 01:35) What do you mean by this will come to that in a moment first take the reverse. If w is a string a 1, a 2, a n the reversal of w, w R is taken as a n, a n minus 1, a 1. If L is a language then the reversal of L is denoted as L R and that is the set of w R w belongs to L the reversal of a language is take each string in L and then reverse it. It consists of all strings of the form w R where w belongs to L. For example, if I take L I will take this example a power n b power m this I have been considering several time. Because it is a good illustrative example L is equal to a power n, b power m; n, m greater than or equal to 1 this is a regular set. L power R take each string and reverse it. So, this will be b power m a power n; n, m greater than or equal to 1.

(Refer Slide Time: 03:07)



What we want to show that is if L is a regular set L R is also regular it is also a regular set. We want to show that how do you show this? It is very simple L is represented by is accepted by a finite state automaton with a state diagram. It can be a finite state deterministic finite state automaton nondeterministic finite state automata or nondeterministic finite state automata with epsilon it does not matter. And without loss you assume that there is only one start state and one final state because otherwise you can add epsilon transition sand make only final state.

So, have a finite state automaton with epsilon transition, if there are more than one final state create a new final state and have epsilon transition leading to that. So, without losses of generality assume that there is only one final state and one initial of course, initial state is so on. Now, how can we accept L R any string in L will take you through apart from q naught to q f? Now, the reversal you have to go from q f to q naught. And then in this diagram if there is an arc like this, from one node to another node change the arc change the direction of the arc.

Keep the same diagram each arc if it is directed this way make it directed this way and make this as a final state; make this as the initial state. So, if a string takes you to from q naught to q f the original diagram in the changed diagram. It will take you from because the arcs are all directed in the in the opposite direction it will take you from q f to q

naught by traversing to a n a n minus 1 and so on. So, this is a transition diagram or a nondeterministic f s a with epsilon moves.

So, obviously it accepts the regular set you can convert it into N F S A and then D F S A. Take this simple example which you have considered q naught, q 1, q 2 a power n, b power m I need not draw the dead state. I will tell you why it is not necessary, but if you want you can even draw the dead state does not matter. I will draw the dead state, but actually you can omit that also dead state b, a, a comma this accepts a power n. Now, I want to convert this diagram so that it express accepts b power m, a power m. So, reverse a direction of every arc, self loop will be same.

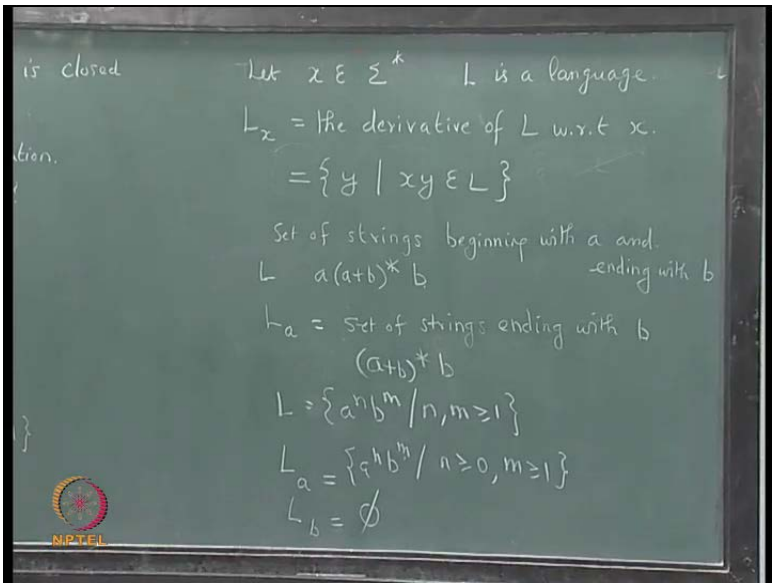
(No audio from 07:12 to 07:37)

Make this into initial state make this into the final state initial change in this example it. So, happens that you have only one final state and one final state if you have more final state you must create a new final state and draw epsilon arcs. Now, you can see that it will accept b power m, a power m. This is not really required the reason is note that this is a nondeterministic F S A. What you have obtained is a nondeterministic F S A, from b if we get from q 2 if we get a b you can go to q 2 or q 1 from q 1 if you get a you can go to q 1 or q naught.

So what you are getting is a nondeterministic F S A, but you know that we can convert a nondeterministic F S A into deterministic F S A. This is not at all necessary because from D you can reach any state. But starting from the q 2 we cannot reach d so it is not going to contribute to anything. So, this is not necessary you need not have considered this portion at all that is not going to affect the result in any way. So, we know that if L is a regular set L R is regular so the family of regular sets is also closed and reversible. Now, one more thing you have to consider which will be used for converting a regular expression D F S A, into a regular expression.

(No audio 09:15 to 09:36)

(Refer Slide Time: 09:36)



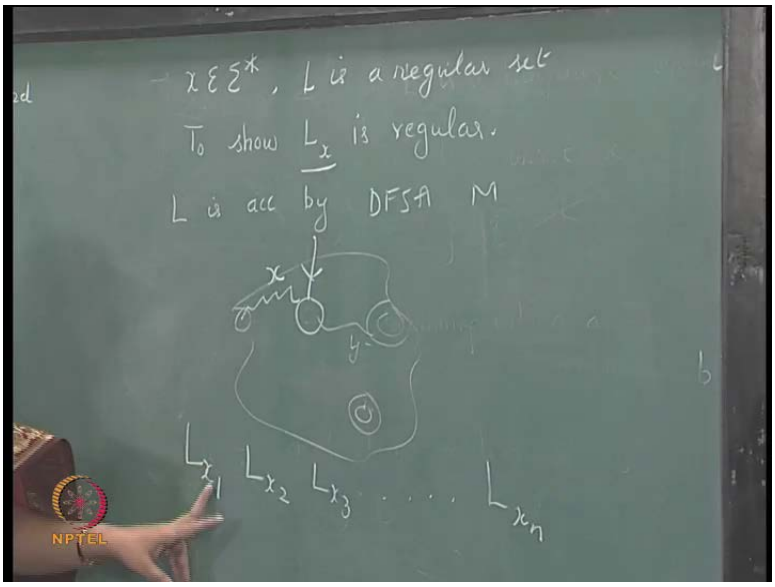
Let x belongs to sigma star x is a some string, we have considered in the alphabet sigma and x is some string. And L is a language L is a language it denote by L of x L with the subscript x this is called the derivative of L with respect to x. What is this language this is the set of strings, of the form y where x y belongs to L. (No audio from 10:35 to 10:41) that is in L you take strings, which begin with x, x is a string. So, consider from L the set consist set of strings, which begin with x and remove that x portion ok is that clear.

Let us see some example 0 0 0 plus 1 star 1, 1 or let us consider the example, which we considered yesterday. Set of strings, beginning with a and ending with b the regular expression for that was a, a plus b star b is not it this is a regular set. Now, what suppose I denote it by L is represented by this, that is the set of strings, over a and b which begin with a, end with the b. Now, what can you say about L a the derivative of L with respect to a, you take all strings, beginning with a and remove the first a. So what will be that that will be just this any string of a (s) and b (s) ending with the b is not it.

So, L of a will be set of strings, ending with b. That is all. It will be represented by a plus b star b and you know that it is a regular set. Now if you take L to be this, a power n, b power m, n comma m greater than or equal to 1. What is L of a (No audio 13:13 to 13:22) what is L of a? a power n, b power m yes (( )) n greater than or equal to 0, m greater than or equal to1. What is L of b? There are no strings beginning with b so L of b

is empty. Now, you can see that we are taking L, L of a is a regular set here also there if we take L L of a is a regular set, L of b is also a regular set phi empty set is a regular set.

(Refer Slide Time: 14:38)



Now, what you want to show is…

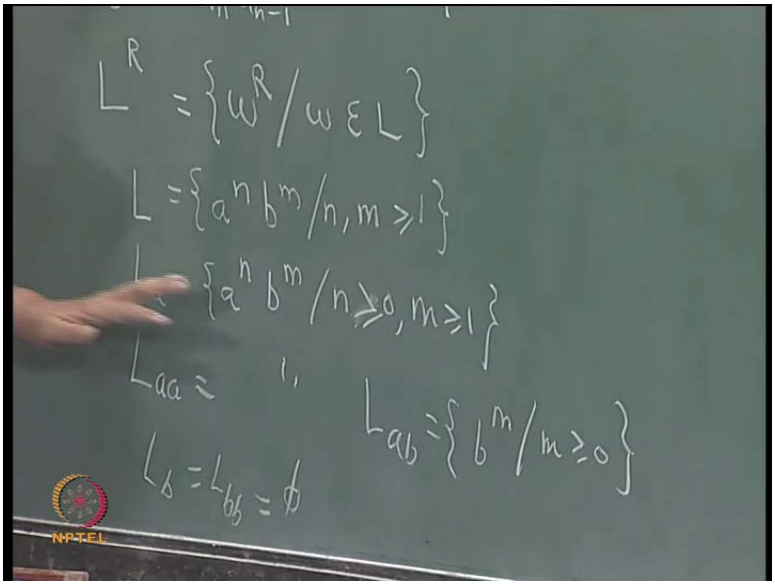
(No audio 14:19 to 14:36)

X, is a string in sigma star L of L is a regular set to show L of x is regular the derivative of L with respect to s is a regular set how do you it is very simple how, do you show that. (( )) Take a deterministic diagram without loss of generality L is accepted by a D F S A. Then what change L is accepted by L is accepted by D F S A M some diagram with one initial state may be some final states. Now, how will we get L of x (( )) make (( )) yeah the same diagram we can have remove the initial state starting from here go through a path for x you will reach some state make this initial state that is all.

Start from the initial node go through the path corresponding to x you will reach one state. I am taking D F S A if it is N F S A there will be many states and so on. I do not want I mean for simplicity I am taking D F S A. So, starting from q naught after reading x I go to a particular state only one state. Make this the initial state same diagram, but they are making this initial state. With the original one accepts something like x y, this automaton will accept y. So, this changed diagram will accept it represents D F S A, which will accept the derivative of L with respect to x.

So what do you get if L is a regular set L of x is regular (No audio 17:05 to 17:12) I have taken a string, now say I take a regular set L I consider L x 1, L x 2, L x 3, L x n like that for different string, how many distinct derivatives can I have?

(Refer Slide Time: 17:52)



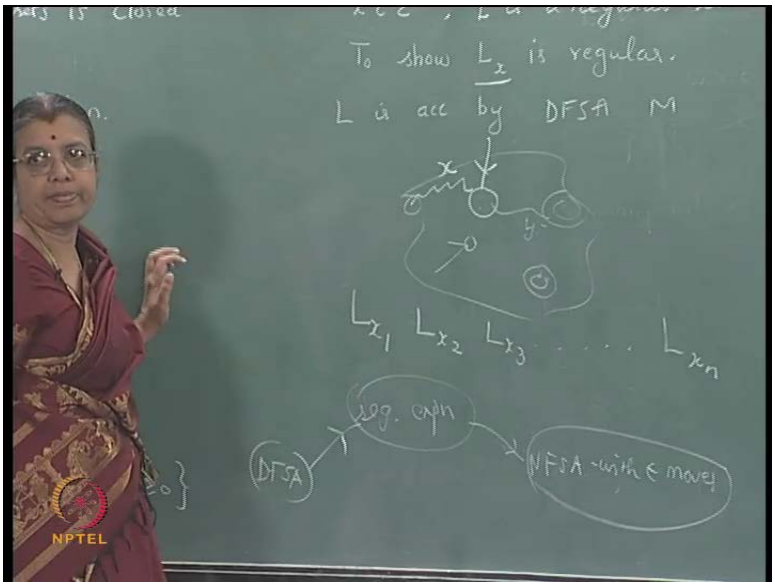
For let us take this example, L is a power n, b power m, L of a is a power n, b power m, n greater than or equal to 0 m greater than or equal to1. What can you say about L a a, what will be L a a same as L a. What can you say L b is empty L b b will also be empty, what can you say about L a b what is L a b (( )) b power m, m.

So, this is L a power n, b power m, n greater than or equal to 0 this is one derivative, this is one derivative, this is one derivative empty set and so on. Now, similarly, for anything L x 1, L x 2, L x 3, L x n are some derivatives, how many of them after sometime they have to be equal. How many distinct derivatives can you have? (( )) It depends on the number of states; see while for considering L of x I am making this into initial state similarly. I can make each one of the state the latest possibility is each one of the states can be made as a initial states.

So, if there are n states in the D F S A, at most they will be n distinct derivative. Afterwards they will become equal more and more strings, you consider some of them will be equal. In this diagram I can make each one of the states as a initial state and get a regular set. So, for x 1 I may make this as the initial state for x 2 I may make this as the

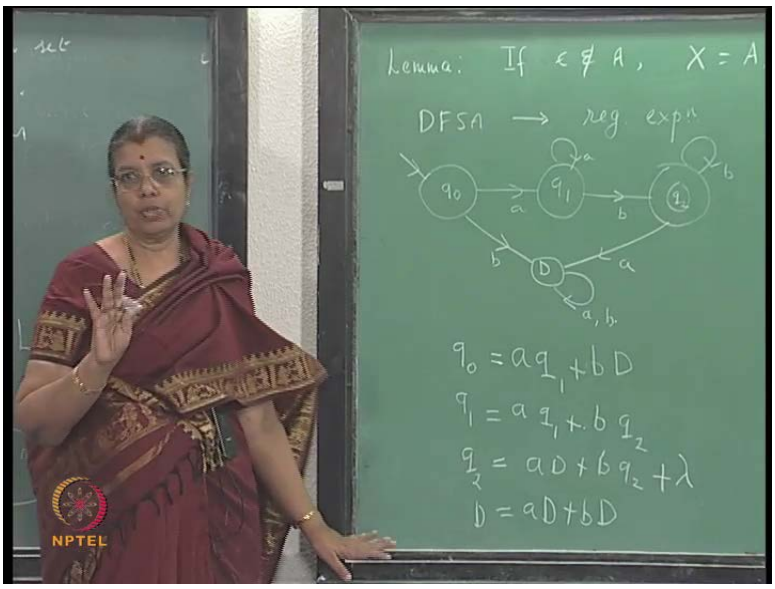
initial state. But again for x 3 I may make come to this itself is not it, so at the most I can have n distinct derivatives this we will use now.

(Refer Slide Time: 20:49)



We have considered the, this given a regular expression how to construct N F S A, with epsilon move this we consider. Now, today we shall consider given D F S A how to find the regular expression. I will recall one theorem which we studied in the last semester.

(Refer Slide Time: 21:19)



This lemma theorem whatever it is we studied this in the last semester please recall this result. If epsilon does not belong to A, X is equal to A X plus B has a unique solution X

is equal to A star B. Did we not study this how many of you remember? So, we will make use of this. I will assume this result if you have forgotten the proof please go through your notes and try to remember.

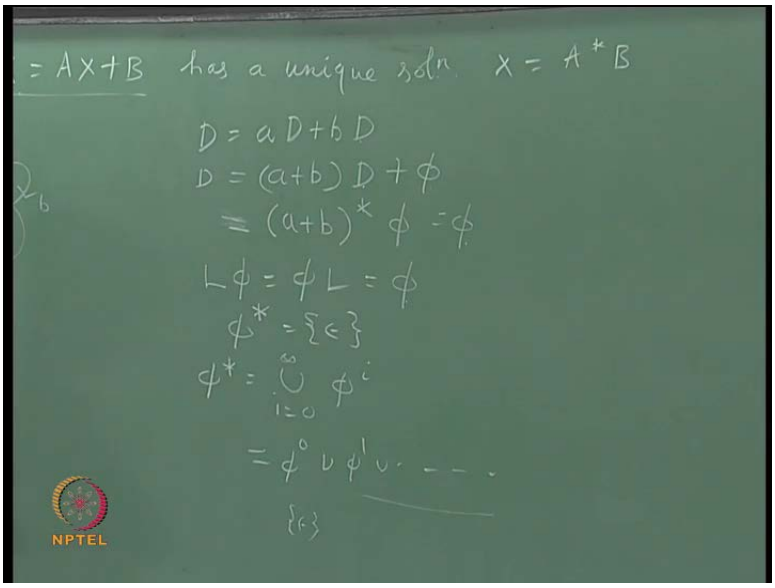
If epsilon belongs to A, the solution will not be unique A star B will still be a solution, but it will not be unique. So, we will not bother about that because this is the result we will make use of. Now, our aim is given D F S A how you will construct the regular expression. Again I will start working out with an example will have 2 or 3 examples. The method should be clear and while working out we know that the procedure is correct give the argument that the procedure is procedure is correct. So, let me start with this same a power and b power.

(No audio 23:04 to 23:34)

How many states are there? There are 4 states so for 4 states I will write 4 equations. How do you write the equations first start with q naught q naught equal to it goes to a q 1, so a q 1 starting with the b it goes to d, b D. q 1, but q 1 is a q 1 plus b q 2, q 2 is a D plus b q 2. But if is it is a final state we must add one lambda also lambda or epsilon I will use lambda for empty string plus lambda this denotes the empty string. Then D is equal to a D plus straightforward how you write the equations I am starting with the deterministic F S A. So, it is very simple how you write down the equations you must look at these as unknowns.

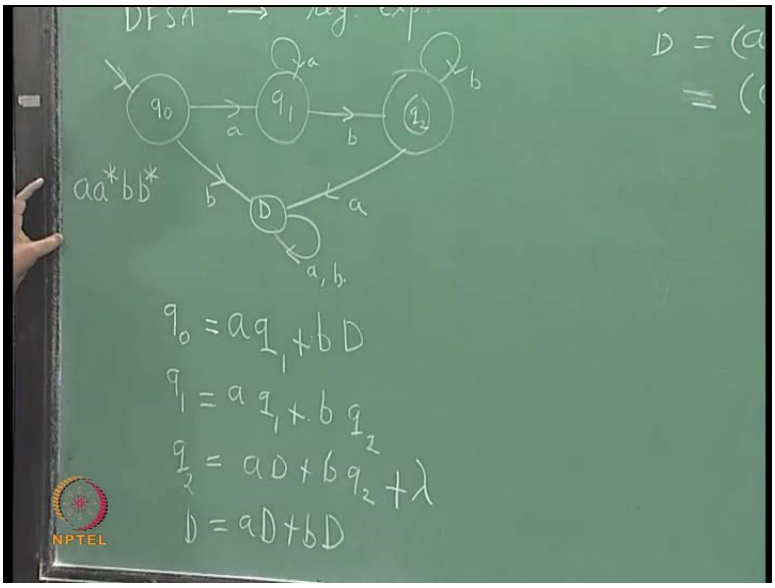
In the sense you are having 4 unknowns and 4 equations and you must be able to solve, but the thing is they are not simultaneous equations as you study it earlier they are equations involving regular expressions. So, you have to use a different technique and the technique you use is you will be making use of this lemma. Only two things you use in solving you have to solve for q naught q naught is the initial state. So, there are 4 equations with 4 unknowns, you should be able to solve. And for solving that you use two techniques one is substitution another is this lemma only these two.

(Refer Slide Time: 26:22)



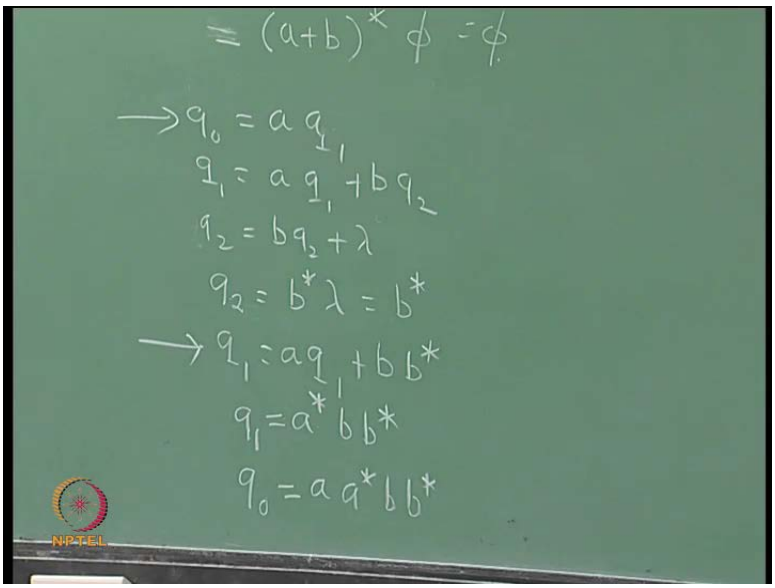
So, let us see how to solve this first take D is equal to a d plus b D. And that you can write as a plus b D. I can write it as a plus d plus phi can also write it that way is not it. Now, this is of the form D is equal to a plus b D phi. So, actually it is of this form is not it. So you can solve for d use this lemma it is of the form X is to equal to A X plus B and epsilon does not belong to this. So, use the lemma what will be the solution for D what will be the solution a plus b yes and something concatenated is phi is phi, see you have this result this also we studied earlier L phi is equal to phi L is equal to phi. But what is phi star this again you must remember phi star is not empty it is having one string phi star is epsilon. Why phi star is epsilon phi star union of i is equal to 0 to infinity phi i. So, it will be phi 0 union phi 1 union etcetera all this will be empty, but phi 0 is epsilon by definition.

(Refer Slide Time: 29:24)



(No audio 28:48 to 29:07)

Before proceeding further what is what will be the answer you have to get. It represents a power n, b power m a power n, b power m n greater than or equal to 1 m greater than or equal to what will be the regular expression. You have to get this answer is not it a a star b b star is not it. This is the answer you have to get so let us see whether you get it D is empty. So, when D is empty this will become empty, this will become empty.



So, the equations will now reduce to q naught is a q a q 1, q 1 is a q 1 plus b q 2, q 2 is b q 2 plus lambda. Now, we have got rid of 1 equation and 1 unknown that is all now we

are ending up with 3 equations with 3 unknowns. Each time when you do the substitution or solving after that you must get rid of 1 unknown and 1 equation. Now, look at the last one it is again of this form. So what is the answer of q 2 q 2 is b star lambda that is b star. Now, use this in the other equations use this in this so what do you get you get q 1 is equal to a q 1 plus b instead of q 2 you write b star. Now, we are having only these 2 equations.

So, we have got rid of 1 more unknown and 1 more equation we are having 2 equations with 2 unknowns. Now, saw again this is of the form A X plus B. So what will be the solution for this q 1 will be a star b b star is not it this star followed by this. So, this is a star b b star use this in the first equation, q naught is a q 1 that is a a star b solving for q naught you have got the answer. Now, this method will work because each time we are trying to get rid of 1 equation and 1 unknown and we are using either that lemma or substitution.

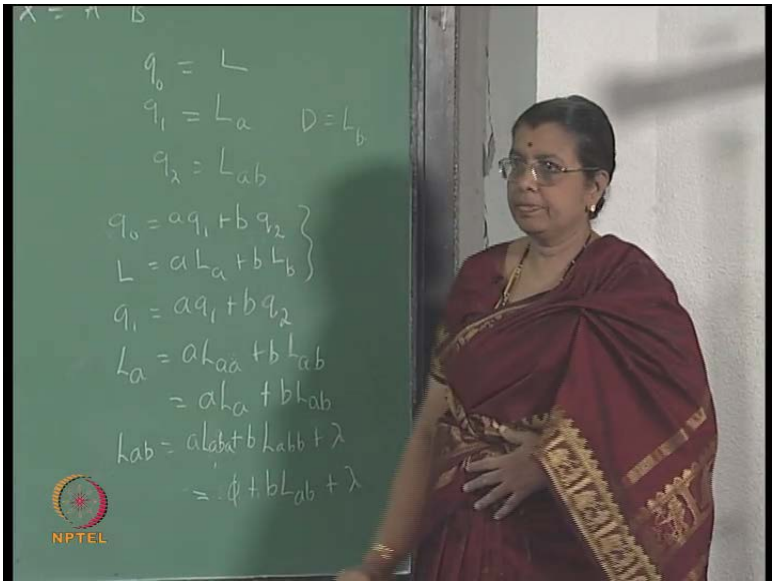
So, finally, we should be able to solve for q naught. Now, why we do this and what is the underlying meaning behind this why you should do this. Now when you solve for D, D is empty when you solve for any variable here take for example, q 1 what is q 1? q 1 is a star b b star is not it finally, you got this q 1 is a star b b star it is a set of strings, which have take you from that state to the final state. So, the set of things which take you from that state to the final state is represented by that when you solve for that variable. So, d means the set of strings which take you from d to the final state, but no string takes you from d to the final state it is empty.

So, you are getting the empty state d is empty q 2 q 2 represent the set of strings, which take you from q 2 the final state. What is that you can go through this several times? So, b star so when you solved what we got q 2 is this term q 1 is the set of strings which take you from q 1 to final state. so that will be a star b b star that is what we have got for q 1 q naught is a set of strings, which take you from q naught to the final state. So that is you know that it is a a star p b star finally, we have to solve for the initial state see suppose I have say ten states or something like that you will get 10 equations.

But then you have to be while solving you have to use your sort of intuition you may you must try to get rid of quite easily. How in which order you will proceed which one you will try to remove first that is not there is no hard and fast rule you have to look into the

rules. And then try to see which one you can get rid of easily and then solve finally, you have to solve for the initial state. That represent the set of strings, which take you from an initial state to the final state and that is the answer you want.

(Refer Slide Time: 35:28)



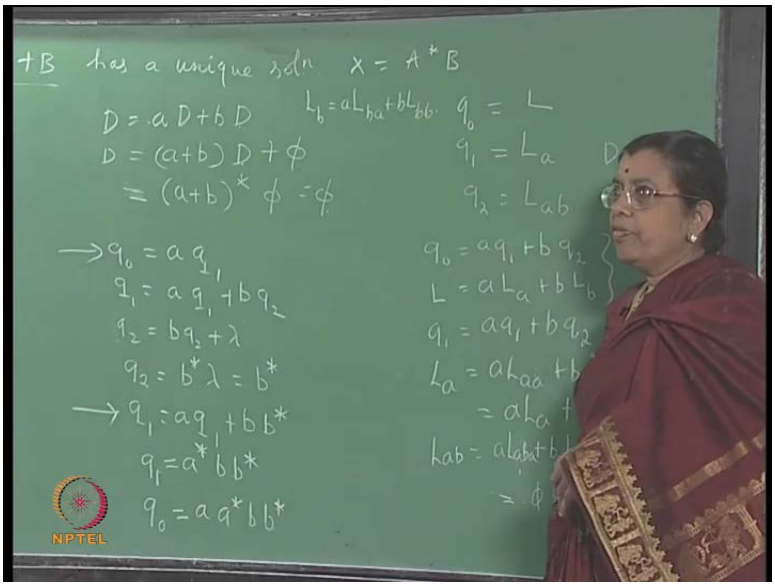
Now, how are we justified in writing this equation and what is q 1 what is q 2 and so on. We are justified in writing this equation because of that in reality you can see that q naught represents L, the language q 1 represents L a, the derivative of a L with respect to a q 2 represents L a b. So, the first equation q naught is equal to a q 1, D is of course, L b. q naught is a q 1 plus b q 2 this is what we have written instead of I can write L is equal to a L a plus b L b. This is what we had written and anything for that matter some L some A or something like that on the left hand side you write it as a A a plus b A b you write like this.

What is that really mean this represents the set of strings; it represents the set of strings which take you from a particular state to the final state. Now in this place so here for example, q naught represents the set of strings which take you from q naught to the final state. Now, this represents the derivative of A with respect to a. I will so the set of strings, some of them can begin with a some of them can begin with b a is set of strings. Among the set of strings some of them will begin with a, some of them will begin with b. So, this represents the set of strings which begin with a set of strings which begin with a is a then the derivative of A with respect to a is not it.

And the set of strings which begin with b is this for example, here you see q naught represents the set of strings which take you from q naught to the final state. If among them some of them may begin with the a some of them may begin with the b. This represent the set of strings which begin with the a this represent the set of strings which begin with the b. That is why it is empty no string in this case it is empty because no strings begins with the b. So, we are justified in writing like this. This equation is really this and the next equation is q 1 is a q 1 plus b q 2 that is L a is a L a a plus b L a b, but we see we saw that L a a is the same as a L a is not it.

So, this is and L a b a L a a is the same. As the L a this we have earlier seen some are we have written, L a is this l a a is also this L b L b b they are all empty. L a b is equal to b power m, m greater than or equal to 0 if you consider L a b b this is also the same. Set of strings beginning with a b b and then remove the a b b it will be b power m m greater than or equal to. So, continuing our equation L a a is the same as L a this is l a b. Then the third equation is q 2 q 2 is a D plus b q 2 plus lambda, this is as L q 2 is L a b L a b is a L a b a plus b L a b b plus lambda. And this is again empty because strings, starting with a b a, is empty here. So, this is a D a phi it will be or L b I can this will be empty and this is just L a b itself.

(Refer Slide Time: 41:24)



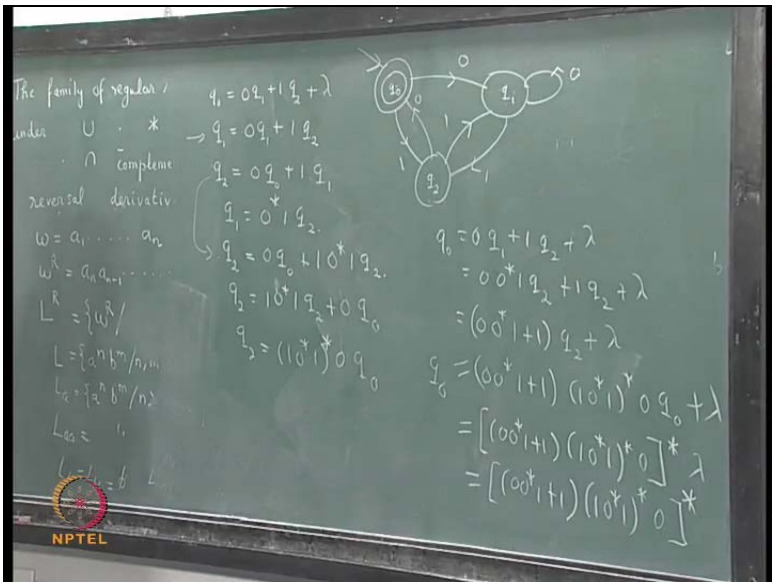
And this equation D is equal to a D plus b D really represents L d is equal to a L b a plus b L b. So, we are justified in writing the equations in that manner and why this lambda

here this really this q 2 is equal to where is that. q 2 is the set of strings which take you from q 2 to the final state. Some of them may begin with a some of them may begin with b, but those beginning with a are empty that is why this becomes empty. Some of them may begin with the b that is why you get this.

But if it is a final state the empty string also takes you from that state to the final state is not it. So, in order to accommodate for that from q 2 you have a set of strings which take you from q 2 to a final state. This accounts for the set of strings, which begin with a, this accounts for the set of strings which begin with b and this accounts for the empty string. Because it is a final state you are not adding lambda anywhere if it is not a final state you are not adding lambda.

It is a final state only you are adding lambda this is the explanation why we get. Why we are justified in writing such equations. So, given a D F S A, you can construct the regular expression in the first in this manner first you write n equations with n variables. Because you have n equations with n variables you will be able to solve and the two techniques which you use for substitution and this lemma. And both while substituting it is we are justified and substituting and while using this lemma also you are justified so finally, you get the correct solution as on.

(Refer Slide Time: 43:58)



So, let me take one more example and solve

(No audio 43:43 to 44:26)

0, 1 look at this diagram with 3 states L is a set of strings, over 0 and one accepted by this F S A. Is this the deterministic f s a? (No audio from 44:48 to 44:55) this is a deterministic and this solution works for a deterministic F S A not considering the nondeterministic F S A. That is why you are able to write uniquely the next thing. So, let us find the regular expression what sort of strings will be accepted by this machine one condition will be the string has to end with (No audio from 45:24 to 45:30). What sort of strings will be accepted by this machine or sort of strings will not be accepted by this machine.(No audio from 45:36 to 45:42)

The string has to end with the 0 epsilon will be accepted of course. But apart from that any other string it has to end with the 0. Let us find the regular expression corresponding to this D F S A. So, how do you go about doing that let us first write down the equations q naught is 0 q 1 plus 1 q 2 plus lambda. Because q naught is the initial is a final state. And equation for q 1 will be q 1 is 0 q 1 plus 1 q 2 and q 2 the 0 q naught plus 0 q naught plus 1 you have to solve for q naught. As I told you, you have to look into the equation and try to solve whichever is easier and so on. So, take this from this you will get q 1 is equal to 0 star 1 q 2.

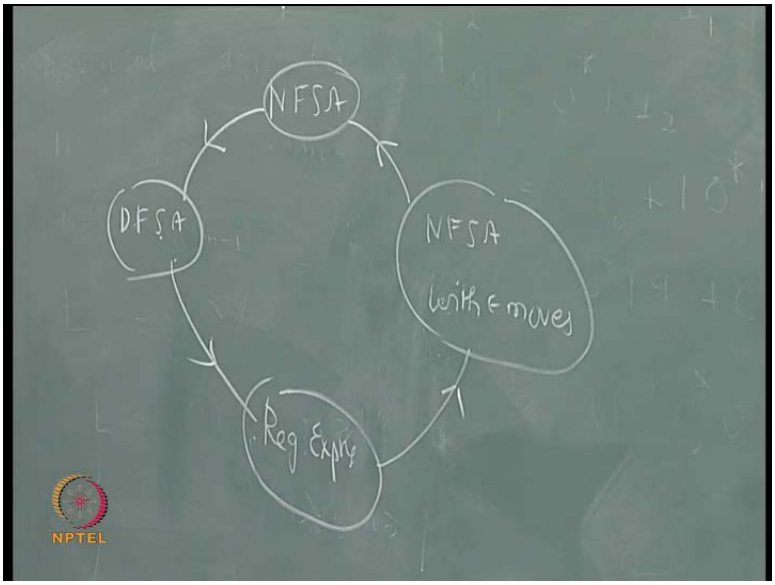
(No audio from 47:27 to 47:356) use that here q 2 is equal to 0 q naught plus 1 instead of q 1 you write 0 star 1 q 2. Use this in this equation you will get this. So that will be q 2 is 1 0 star 1 q 2 plus 0 q naught. Use the lemma again so q 2 will be 1 0 star 1 0 q naught. (No audio from 48:41 to 48:47) Look at the first one q naught is 0 q 1 plus 1 q 2 plus lambda that is equal to 0 q 1 is 0 star 1 q 2. So, 0 star 1 q 2 plus 1 q 2 plus lambda that is 0 0 star 1 plus 1 into q 2 plus lambda, but what is q 2 q 2 is 1 0 star 1 star 0 q naught so use that here 0 0 star 1 plus 1 instead of q 2 you write this 1 0 star 1 star 0 q naught plus lambda.

So, q naught is this q naught plus lambda, it is again of the form X is equal to A X plus B where B is lambda. So, if you use expression if you use that X is equal to A X plus B is a star b you will get 0 0 star 1 plus 1 1 0 star 1 star 0 star lambda. This is nothing, but that lambda you can remove 0 0 star 1 plus 1 1 0 star 1 star 0 (No audio from 51:00 to 51:06) that is starting from q naught how will you reach q 2 0 0 star 1 or just 1. That is this portion starting, from q 2 how will you get q 2 again q 2 1 0 star 1 that you can repeat

several times that is why that 1 0 star 1 star is here. Then you go back to 0 q naught and this itself you can repeat several times and that is why the star outside.

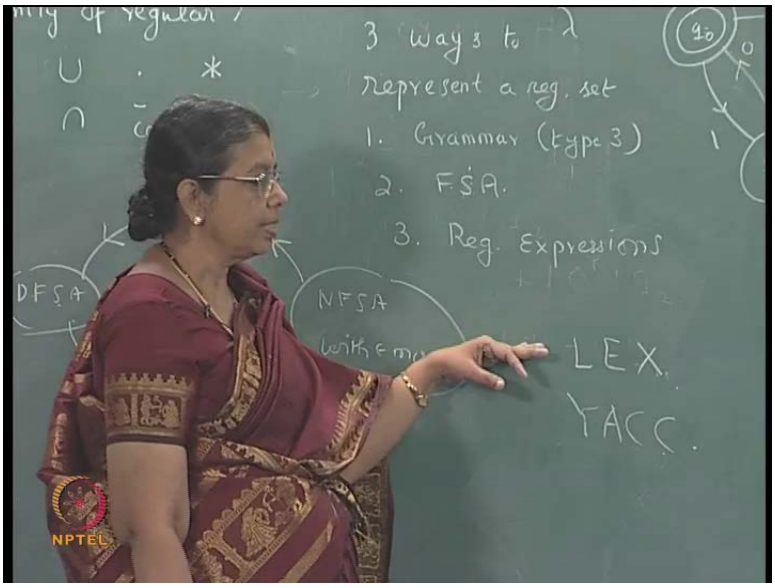
Now, you can see that apart from epsilon the string will end with the 0 it cannot end with the 1 and then each time a string within the bracket is considered there will be 1 1 contributing from this and even number of one is contributed from this. So, always the blocks of one will be of odd length they will not be of even length. So, this method helps us to get the regular expression from the final state automata so for a regular set. (No audio from 52:35 to 52:44)

(Refer Slide Time: 52:50)



We have this is complete now you have D F S A, you have N F S A, you have N F S A with epsilon move sand you have regular expressions. So, we have seen given an N F S A, with epsilon move how to construct the N F S A without epsilon moves. And by subset construction from a N F S A, you can construct the D F S A, given a regular expression you can construct the N F S A with epsilon moves. This also we have considered and today what we have considered is given a regular expression how to find the given D F S A how to find the regular expression.

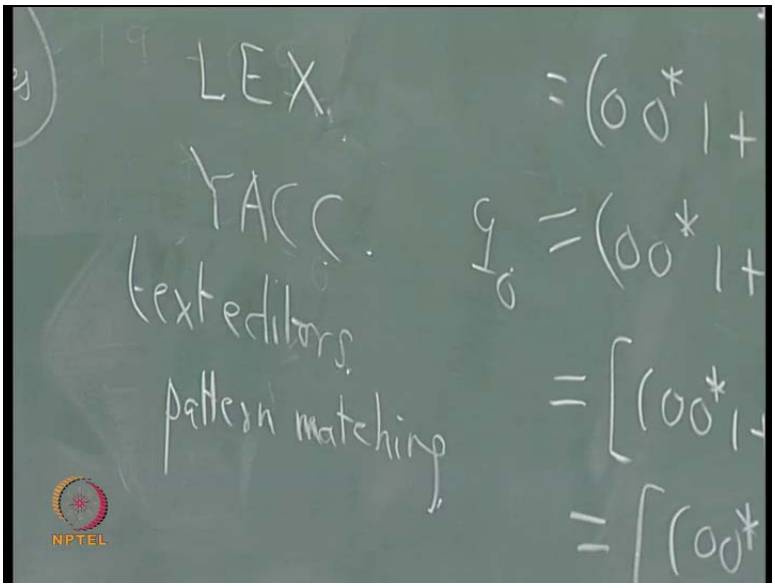
(Refer Slide Time: 53:49)



So, regular sets can be represented in 3 different ways 1, 3 ways to represent a regular set 1 is grammar this type 3 second is F S A, this is an acceptance device. We have shown the equivalence between the two earlier and another is by means of regular expressions. We have seen the equivalence between these two norms. So, all these 3 are different ways of representing regular set. The idea of regular expressions is very useful in L E X is a automatic lexical analyser generator which is used as part of a compiler generating device.

The compiler has 2 parts the parser and I mean syntax part and the analysis part and the synthesis part the analysis part consists of the lexical analyser and the parser. And nowadays you have automatic way of generating the lexical analyser. And the parser you have a program called a L E X, which is a automatic lexical analyser generator and you have a program called Y A C C, which is automatic parts of generator a lot of context free grammars and parsing ideas will be used here lot of regular expressions idea will be used here.

(Refer Slide Time: 55:56)



I will spend may be a few minutes on this sometime later how the L E X works or the idea behind the automatic lexical analyser generator. Not only that in text editors also the idea of regular expression is used that is for pattern matching.(No audio from 56:3 to 56:13) So, you want to see you have a big text in spell check and all you use this quite often you have a text. And you want to find out supposing you have misspelt something you want to check what is, that word and then check wherever you have that word you have to replace it by the correct spelling and so on. So, for example, something like theory instead of writing like this you have written like this some where you will check and then replace it. And this or for that matter some name you have misspelt then use the spell check you use the spell check what it is this ultimately gives us some idea of regular expression or a finite state automata.