Theory of Computation

Prof. Kamala Krithivasan

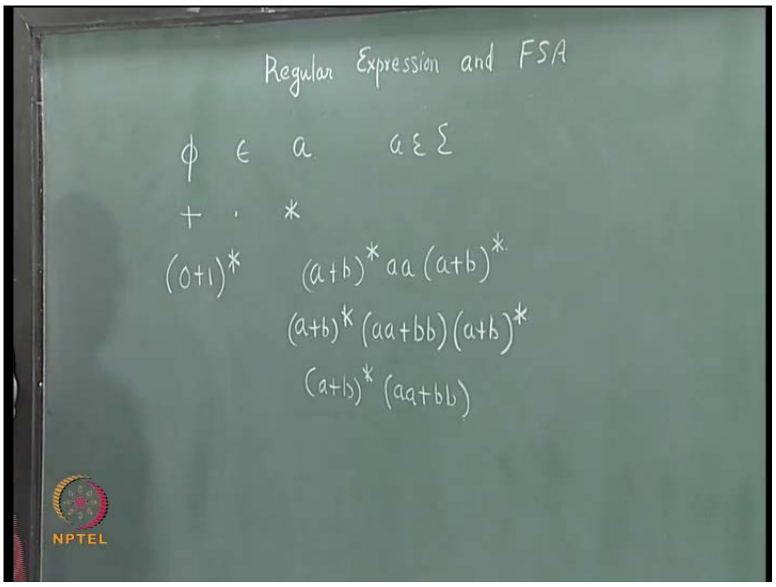
Department of Computer Science and Engineering

Indian Institute of Technology, Madras

Lecture No. # 13

Regular Expressions Regular Expressions to NFSA

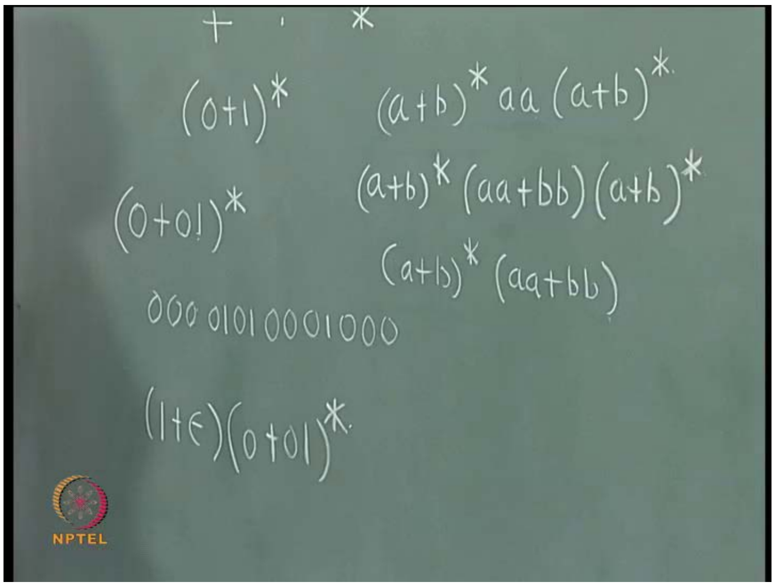
(Refer Slide Time: 00:15)


Regular Expression and FSA
$\phi \quad \epsilon \quad a \qquad a \in \Sigma$
$+ \quad \cdot \quad *$
$(0+1)^*$
$(a+b)^* aa (a+b)^*$
$(a+b)^* (aa+bb) (a+b)^*$
$(a+b)^* (aa+bb)$

We were considering regular expressions, regular expressions starts with phi epsilon or letters of sigma, a is a letter belonging to sigma, and then it uses the operation of plus, dot, star, a finite number of times, and you get a regular expression. We were considering some examples of regular expression; for example, 0 plus 1 star denotes, the set of all strings over 0 and 1 are all binary strings, a string of the form a plus b star a plus b star a a plus b star. This is a regular expression, it denotes the set of strings, which have at least one consecutive a a. Any string of a s and b s followed by a a a and any string of a s and b s.

So the language consists of all strings having at least one pair of consecutive a s. And a plus b star a a plus b b a plus b star, this denotes the set of strings, which have at least one consecutive pair of a s or one consecutive pair of b s. An expression of this form, a b star a a plus b b this represent the set of strings, which either end with a pair of a s or end with the pair of b s. Any string of a s and b s you can have, but it should end with the pair of a s or pair of b s. Now, look at this expression, 0 plus 0 1 star consider this expression 0 plus 0 1 star. What does the set represented by this expression? What is the set represented by this regular expression?

(Refer Slide Time: 02:40)



$+ \quad \cdot \quad *$

$(0+1)^*$ $\quad (a+b)^* aa (a+b)^*$

$(0+01)^*$ $\quad (a+b)^* (aa+bb) (a+b)^*$

$(a+b)^* (aa+bb)$

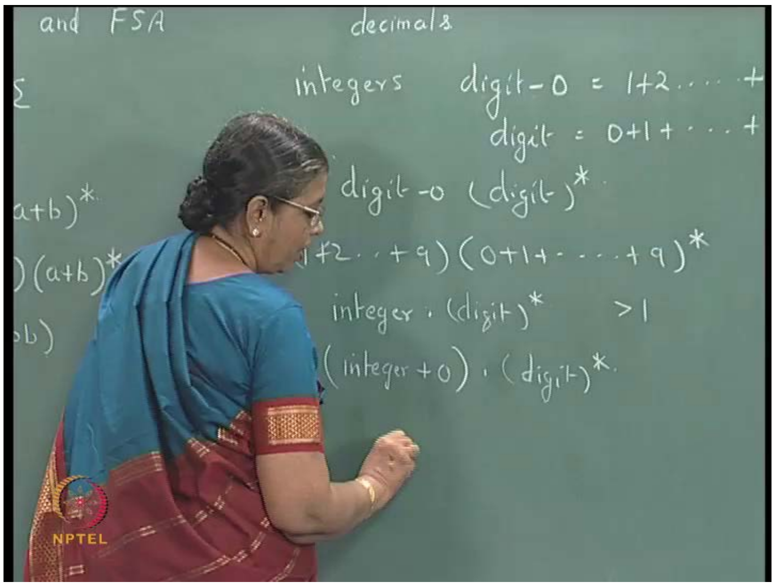
000 0101 0001000

$(1+\epsilon)(0+01)^*$

Now, you see that you can have any number of 0 s, because 0 0 0 you can have or you can have 0 1 0 1 a number of times again any number of 0s we can have, we can have 0 1 and so on. But you see that after a 1 occurs, either you will get a 0 or you will get a 0 1. So, when you get a 1 see 0 1 occurs this is a sequence followed by 0 s and 0 1 s. The set of strings represented by this regular expression consists of a sequence of 0s and 0 1s. So, when you have a 0 1 next, which will be followed either by a 0 1 or it can be followed by many 0s. Whatever, it is you can see that, you will not get two consecutive 1s after a 1 occurs, it will occur because you get a 0 1, and after that either you will get a 0 or a 0 1 so you will not get two consecutive 1s.

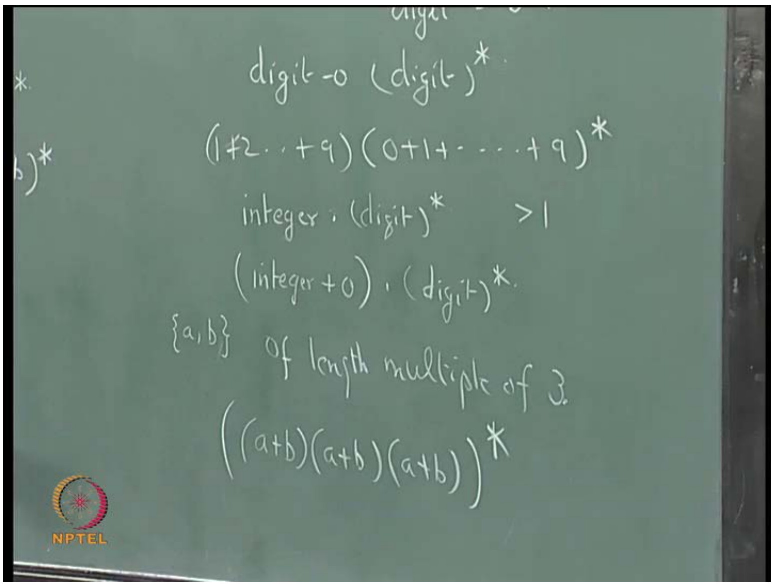
So, this expression represents a set of 0s and 1, strings over 0 and 1 or binary strings where you cannot have two consecutive 1s. And you must also note that, it begins with a 0. It can end with a 1 or a 0, but it begins with always with the 0. Now, if you allow the possibility that initially you can have a 1, again you cannot have 2 consecutive 1s. But it the string may begin with a 1. If you allow for that possibility, then you have 1 plus epsilon 0 plus 0 1 star. So, this denotes that either you begin with a 1, if you take epsilon that means, nothing is taken from this portion. So, it can begin with a 0, the string can begin with 0 or with 1 and then it will be followed by, some other things, but the main thing is you will not have two consecutive 1s.

(Refer Slide Time: 05:01)



Now, if you want to represent the set of decimals, we have already seen how to represent the set of FORTRAN identifiers, if you want to represent first the set of integers. Let me denote by digit 1 digit 0 digit minus 0 digit hyphen 0 this is 1 plus 2 plus 9. And digit is digit is 0 plus 1 plus 9. So, integers can be represented by digit hyphen 0, followed by digit star or in a sense the regular expression, if the alphabet is 0 to 9 it will be 1 plus 2 plus 9, followed by 0 plus 1 plus 9 star. If you want to represent decimal, it will be an integer followed by a dot, followed by a sequence of digits. So, it will be integer dot (No audio from 06:37 to 06:43) digit star.

(Refer Slide Time: 07:43)


digit - 0 (digit)*

(1+2..+9)(0+1+....+9)*

integer . (digit)* >1

(integer + 0) . (digit)*

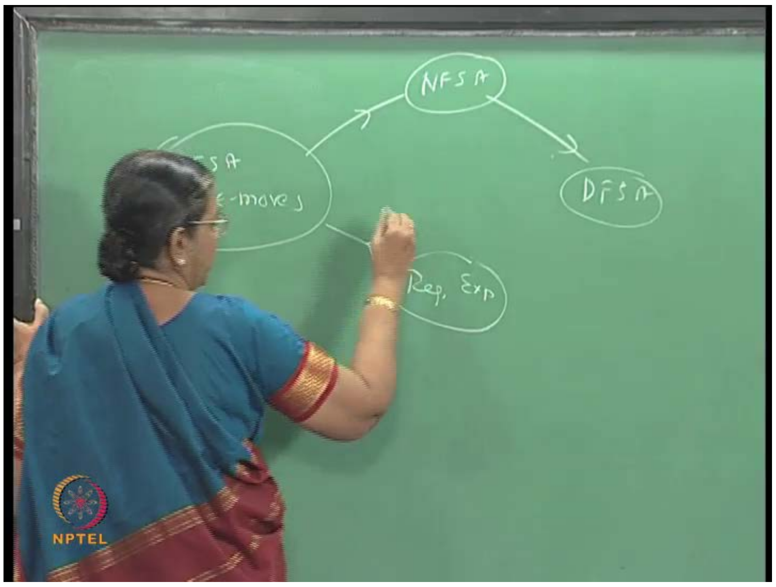
{a, b} of length multiple of 3

((a+b)(a+b)(a+b))*

So, we assume that, you are representing integers greater than 1. First portion you have, if you can allow for 0 also 0, something like that. In that case, you will have integer plus 0 dot digit star. Please note that, you also allow something like in this expression, you will also have something plus like this will also be in the language. So, like this you can express certain things, using regular expressions. Suppose, you want to represent the set of strings over a and b of length divisible by length, which is a multiple of 3. The regular expression for that will be you can have a plus b or you can have a plus b the length will be 3 this allows all strings.

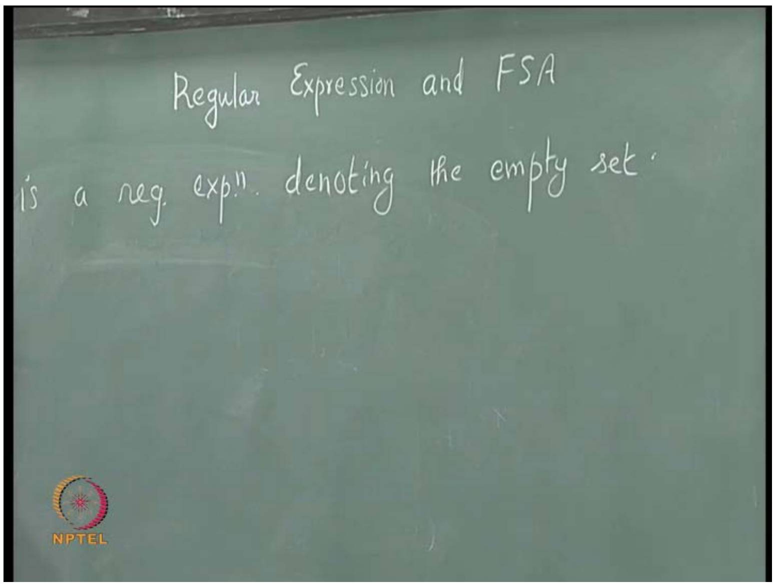
This regular expression represents the set of all strings over a b which are multiple of 3 epsilon is also allowed; when you have this star epsilon is also allowed here. And similarly, you can see that in this regular expression, epsilon will be in the language. Epsilon will not be in the language represented by this, because it has to end with 2 a s or 2 b s the minimum length string is a a or b b in this case.

(Refer Slide Time: 09:00)



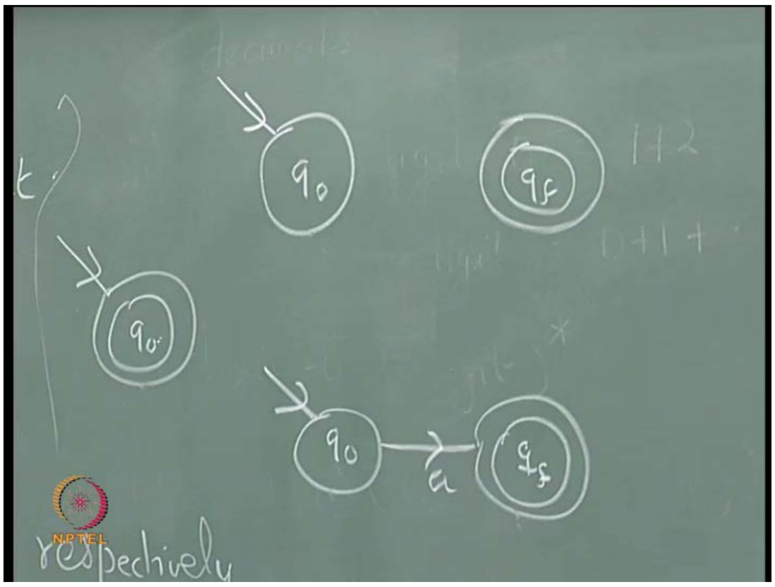
So, having defined this, we have to see that we have seen that N F S A, with epsilon moves and N F S A, D F S A and regular expressions. So, today these things we have already considered, today we shall consider this how to get, N F S A with epsilon moves from a regular expression.

(Refer Slide Time: 09:49)

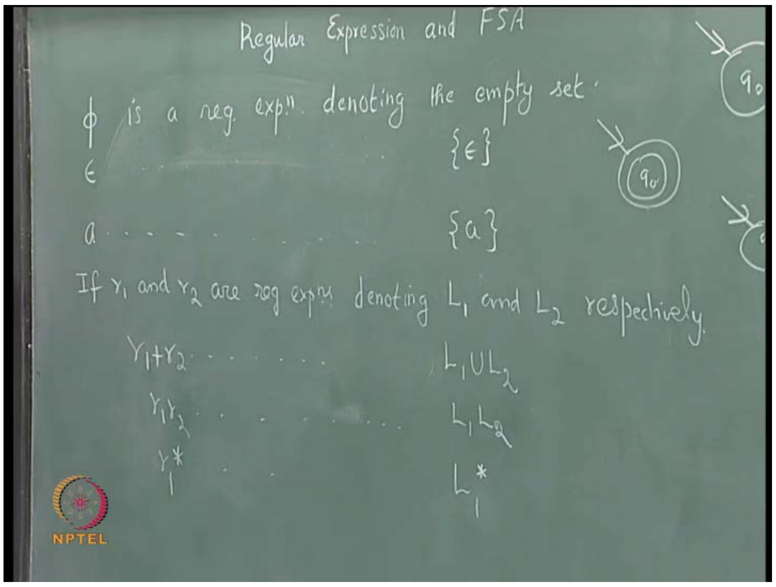

Regular Expression and FSA
is a reg. expn. denoting the empty set.

Now, given a regular expression, how to construct the non deterministic F S A with epsilon moves, this is what we will see now, how do you define a regular expression? You start from the empty set epsilon and symbols of sigma and perform the operation of union concatenation and star a finite number of times. Now, the definition of regular expression is like this, I will rewrite the expression though, we have considered it in the last class. Phi is a regular expression, denoting the empty set.

(Refer Slide Time: 10:36)



(Refer Slide Time: 11:08)



Regular Expression and FSA

ϕ is a reg. expn. denoting the empty set.

ϵ . . . . $\{\epsilon\}$

a . . . . $\{a\}$

If r_1 and r_2 are reg exprs denoting L_1 and L_2 respectively.

r_1+r_2 . . . . . . $L_1 \cup L_2$

$r_1 r_2$ . . . . $L_1 L_2$

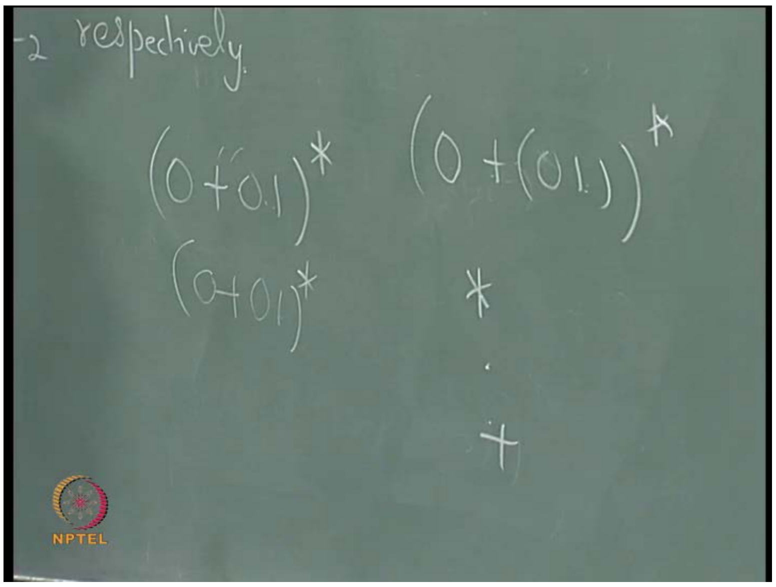
r_1^* . . . L_1^*

Now, how can you draw a diagram non deterministic F S A with epsilon most for this? You can have a diagram like this, q naught q f without any arc between them. What does this accept this diagram? There is no arc there it accepts the empty set no string will be accepted. This is the state diagram of a machine, which accepts the empty set then epsilon is a regular expression denoting a set having only epsilon. Epsilon is a regular expression, which denotes a regular set having only 1 string and that 1 string is the empty string epsilon. How can you represent it by, a F S A (No audio from from 11:33 to 11:38) there is only 1 state, which is the initial state and the final state. When will epsilon be accepted, if the initial state is a final state epsilon will be accepted there are no more arcs no transitions.

So, no other string will be accepted, then a is a regular expression denoting the set containing just one string, which is a. For each symbol in sigma a is a regular expression, which denotes a set having just one string which is a, how can you draw non diagram for these state diagrams, you can have something like this q naught a (No audio from 12:32 to 12:40) q f. So, a alone will be accepted. Then what was the definition, if r 1 and r 2 are regular expressions, denoting L 1 and L 2 respectively (No audio from 13:11 to 13:17) r 1 plus r 2 is a regular expression, which denotes L 1 union L 2 r 1 r 2 denotes a regular expression. It is a regular expression, which denotes the concatenation of L 1 and L 2, r 1 star is a regular expression, which denotes L 1 star this we have seen.

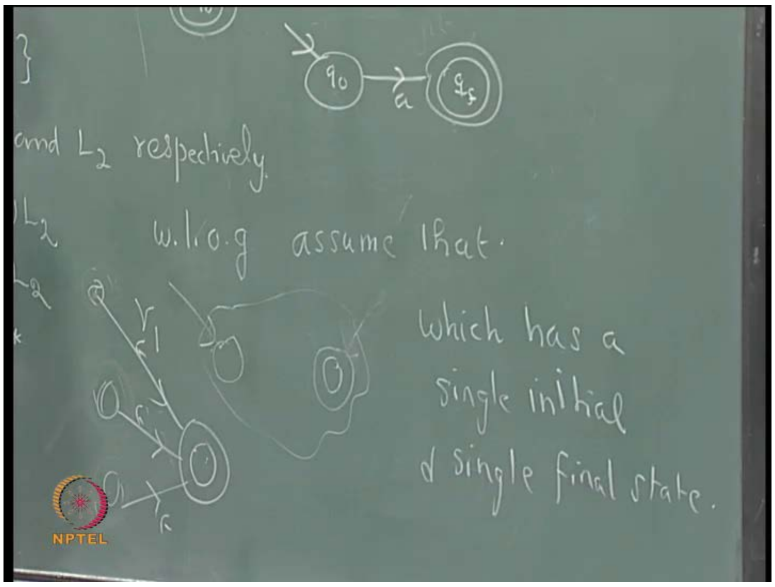
So, this is something like a bases class, you start with the empty set epsilon and symbols a. Then you use the operation of star plus concatenation a finite number of times and you get a regular expression. For example, if you look at this 0 plus 0 1 star this is a regular expression, 0 and 1 are symbols. So, first you are concatenating 0 1 then you are using union, then you are using star.

(Refer Slide Time: 14:12)



So, we do not write this as usually, we do not write it as this is what we mean 0 this is concatenation is performed a first, then this union then the star. Now, we do not write like this we usually write like this, because among these three operators star dot plus this has the highest precedence. This has highest precedence then concatenation, then union among these operators this is the way, you evaluate the expression. So, that is why because concatenation has higher precedence, than the union this parenthesis is omit.

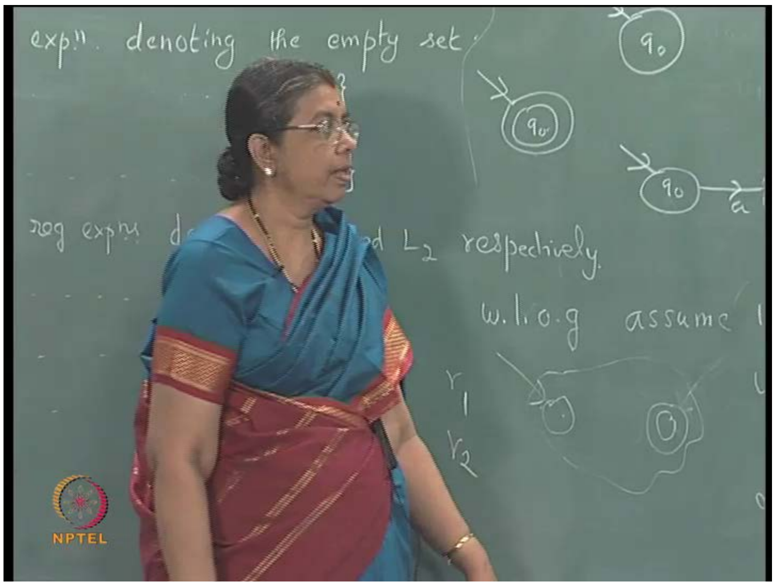
(Refer Slide Time: 15:36)



(No audio from from 15:21 to 15:32) Now, without loss of generality, you can assume that r 1 is represented by a state diagram like this, which has a single state. It has a single initial state single initial, and single final state. Usually, our initial state is only 1, but you can have several final states, that is what we have assumed. Now, without loss of generality, we can assume that there is a only 1 initial state that is always there. There is only 1 final state, there is only 1 final state and there is only 1 initial state. Similarly, r 2 also can be represented by a diagram, where you have a initial state and only 1 final state.

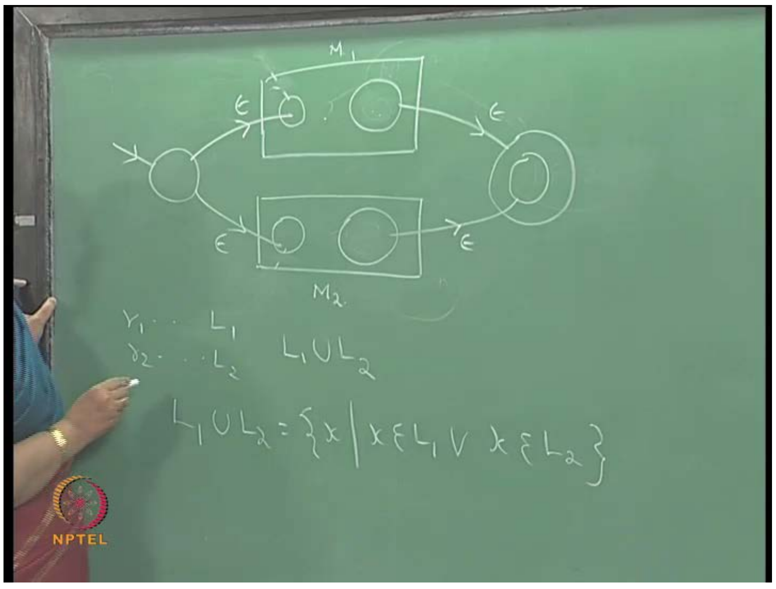
How can, we assume that how are we justified in assuming this, because we are considering transition diagram or a state diagram with epsilon moves. Suppose, there are say 3 final sates. You make them all non final, you have introduced a new final state make these non final (No audio from from 17:19 to 17:25). And have epsilon transitions to this final state and there is no arc leaving from the final state. So, instead of having 3 final sates, I can make them non final and have only 1 final state, where from each final state an epsilon transition leads to the final state. The language accepted by such as transition diagram will not be affected by this.

(Refer Slide Time: 18:22)


exp.n denoting the empty set
reg exp.ns d... d L₂ respectively
w.l.o.g assume
r₁
r₂
q₀
q₀
q₀
a

So, without loss of generality, you can assume that r 1 is represented by transition diagram or N F S A with epsilon moves, where there is the 1 initial state and only 1 final state. Similarly, r 2 is also represented by a diagram with 1 initial state and 1 final state.

(Refer Slide Time: 18:52)

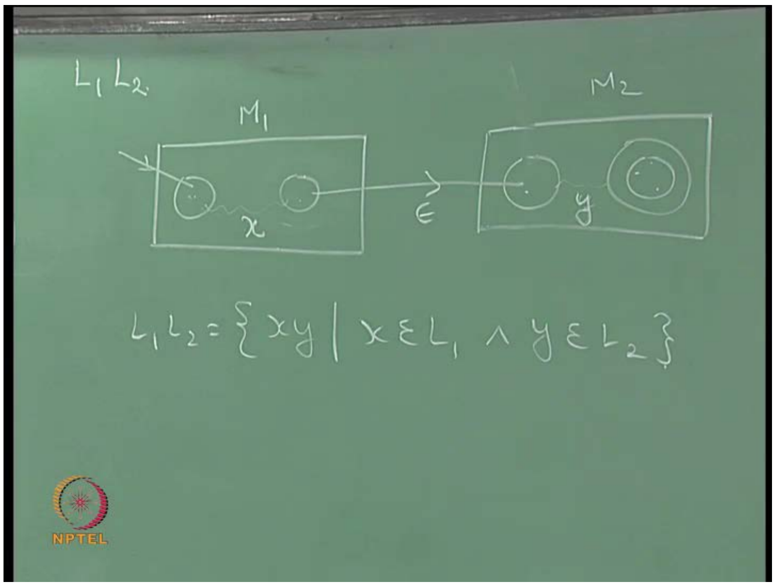


Now, how do we go about defining r 1 plus r 2? So, this r 1 is represented by M 1 with an initial state and the final state right, r 2 is represented by another transition diagram with an initial state and final state, this is M 2. So, r 1 is represented by this, r 2 is represented by this. How can you represent r 1 plus r 2 by a transition diagram or a N F S A with epsilon moves. I am using the word N F S A, with epsilon moves and transition diagrams in an equivalent manner, state diagram or transition diagram. What you do is you introduce a new initial state and remove them these are no more initial states. Introduce a new final state; these are no more final states then add epsilon arcs here add epsilon transitions (No audio from from 20:41 to 20:51) here.

So, this is a transition diagram or a non deterministic F S A with epsilon moves, this has got a single initial state and a single final state. And what does it accepts, if r 1 accepts L 1 r 2 accepts L 2 this accepts L 1 union L 2, why how do you define L 1 union L 2. L 1 union L 2 is x belongs to L 1 or x belongs to L 2. So, any string accepted by L 1 you will go from here to here through a path for M 1, this is the initial state and this is the final state. So, if a string is accepted by M 1 or if a string is in L 1 it will be accepted by M 1. So, that string will take you from this state to this final state. Some x say, then how will the same string be accepted by this diagram or this non deterministic F S A with epsilon moves, start from here through an epsilon transition, you go here then reading x you go to this state. Then through an epsilon transition, you go to the final state.

So, the string x will be accepted by this also, if it belongs to L 1, if it belongs to L 2 it will be accepted by the N F S A, M 2 with epsilon moves. That is suppose this string y is accepted y is in L 2. There will be a sequence of moves, which takes you from this initial state to the final state here. Because it is in L 2 or it is accepted by M 2. Now, how will this diagram accept that starting from here go to this state through an epsilon transition. Then after reading why you go to this state, then through an epsilon transition you reach here. So, the same string will be accepted. So, this diagram you can see that it is a non deterministic F S A, with epsilon moves and it will accept L 1 union L 2.

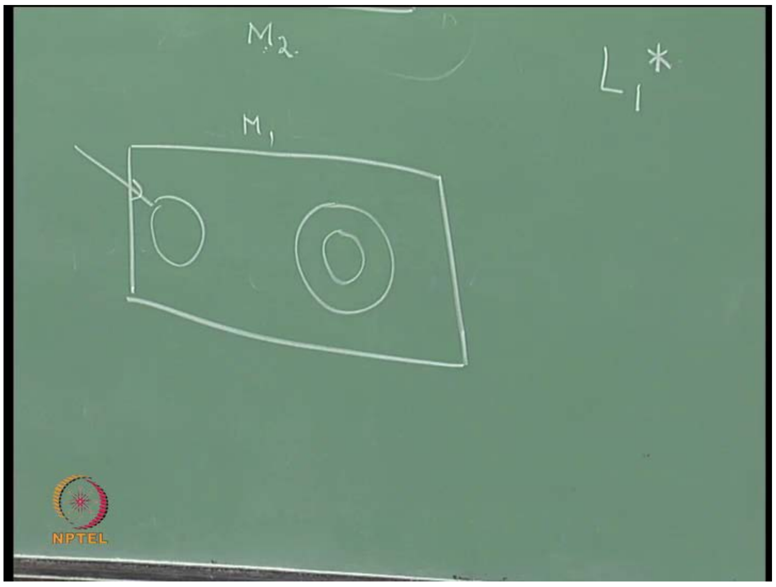
(Refer Slide Time: 23:56)



Now, the concatenation of L 1 and L 2 suppose, L 1 is accepted by a non deterministic F S A with epsilon moves M 1 with initial state and a final state. And L2 is accepted by M 2 with initial state and final state like this. Now, you want to accept L 1 L 2 what is L 1 L 2? L 1 L 2 consists of strings of the form x y x belongs to L 1 and y belongs to L 2. Now, how can you get this from this diagram, you have a diagram, where this will no longer be a final state. And this will no longer be an initial state, and there is an epsilon transition from here to here. This is an N F S A with epsilon moves what sort of strings will be accepted by this, if a string a string x can be accepted by M 1 by going from this state to this state and while going it reads the string x through the transitions.

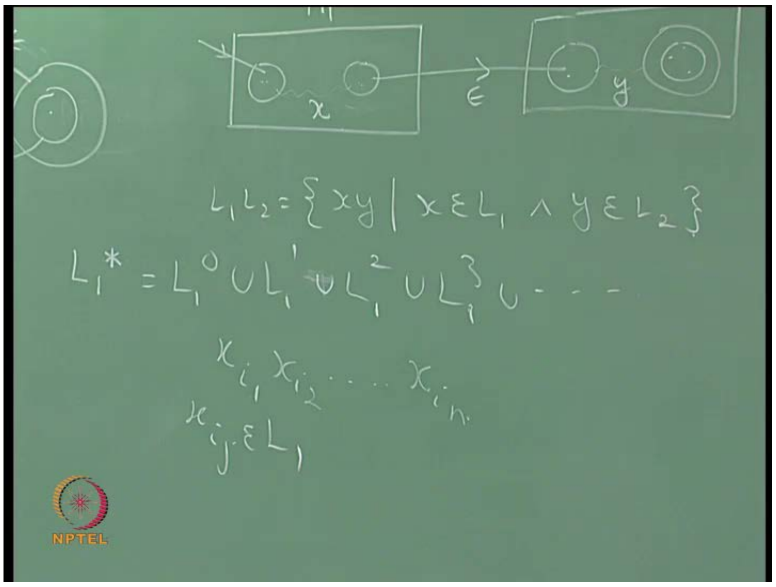
And a string y will be accepted by this machine, by going from this state to this state and while going from the initial state to the final state, it reads the string y including epsilon moves through the transitions. Now, what sort of strings will be accepted by this diagram this is the initial state. So, start from here and it reads a string of the form x and then it makes an epsilon transition to this then reads a string of the form y to reach a final state. So, the strings accepted by this transition diagram are the N F S A, with epsilon moves will be of the form x y, in between you have an epsilon transition.

(Refer Slide Time: 27:05)



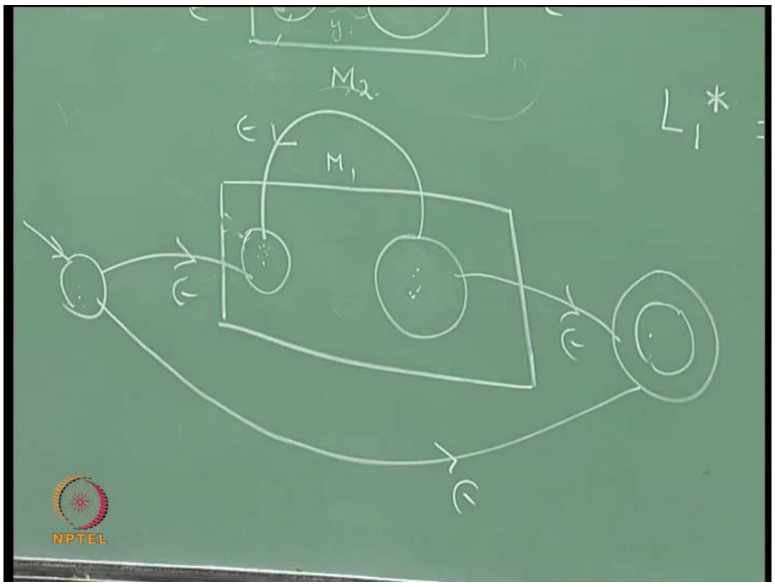
So, this accepts L 1 L 2 or the set represented by r 1 r 2. Next, we have L 1 star L 1 is represented by r 1 and it is represented by N F S A with epsilon moves.

(Refer Slide Time: 27:31)


x
ε
y
$L_1L_2 = \{xy \mid x \in L_1 \wedge y \in L_2\}$
$L_1^* = L_1^0 \cup L_1^1 \cup L_1^2 \cup L_1^3 \cup \cdots$
$x_{i_1} x_{i_2} \ldots x_{i_n}$
$x_{i_j} \in L_1$

Now, I want to have a N F S A with epsilon moves, which will accept L 1 star. What is L 1 star? L 1 star is L 1 0 union L 1 that is L 1. (No audio from 27:38 to 27:47) And what sort of strings will be accepted by L 1 star. See, it will have strings of the form x i 1, x i 2 x i n L 1 star will consists of strings of the form x i 1, x i 2 x i n, where each x i begin belongs to L 1. And L 1 0 represent just a empty string, so empty string will also be there in the language, L 1 star will contain the empty string also. So, the set of strings which are present in L 1 star will be of the forms x i 1 x i 2 x i n, where each x i j belongs to L 1.

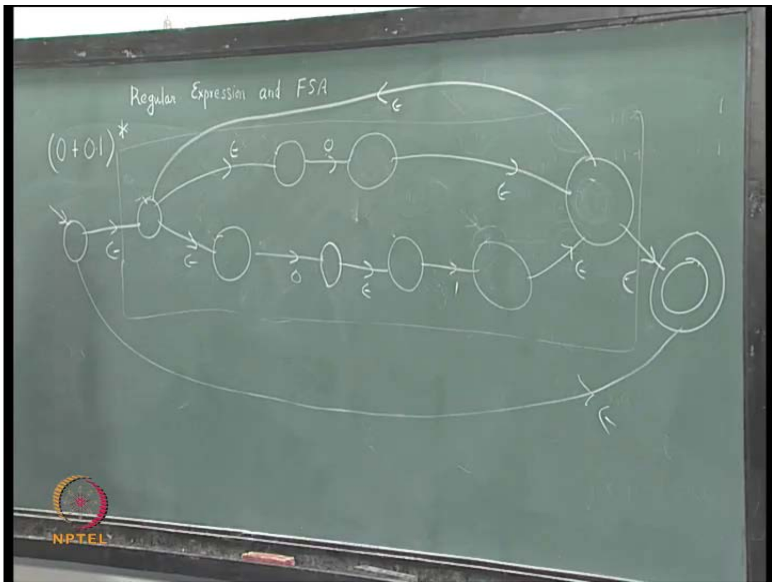
(Refer Slide Time: 28:52)



Now, how will you modify this state diagram, so that it accepts L 1 star. You have 1 new initial state and 1 final state this will be the initial state. Now, and that will be the final state, these states will not be initial and final states. (No audio from 29:10 to 29:16) And you add epsilon transitions here, when we do this, what is the language accepted, just after having this same as L 1 nothing has changed. Instead of change, you have made a new initial state and a new final state and starting from here, you go here any x you can read and go here. So, this just accepts L 1 only.

Now, you want to accept L 1 star. So, from here to here, you have an epsilon transition. So, what sort of strings will be accepted by this diagram, staring from here you go here, you can read a x i 1 and go here, go here again through epsilon transition read a x i 2 here go here, read a x i 3 here go here. Repeat this several times, after reading x i n it transit to the final state. So, strings of the form x i 1 x i 2 x i n will be accepted is this or something is missing, epsilon will not be a accepted unless this accepts epsilon L 1 0 consists of epsilon, you have to accept epsilon L 1. So, there should be r 1 are from the initial state to the final state with label epsilon.

(Refer Slide Time: 31:46)



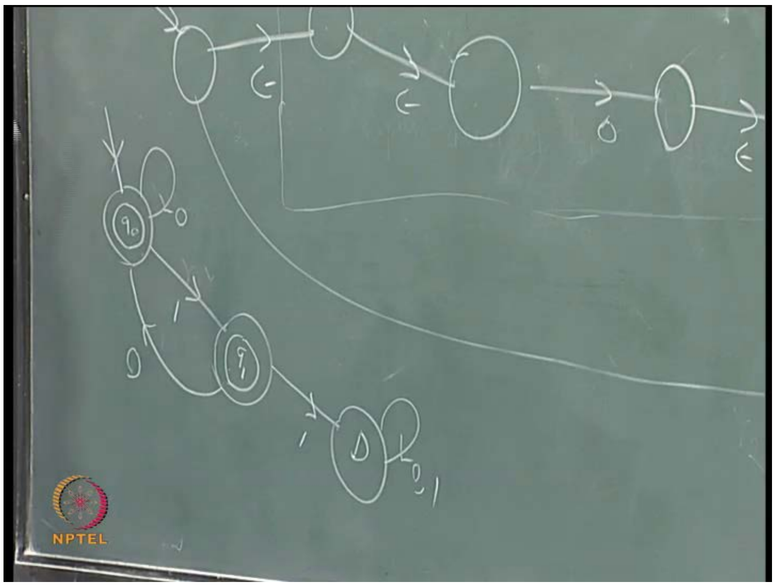
So, this diagram this N F S A, with epsilon moves will accept L 1 star. Step by step construction, you can use for let me take a simple expression. (No audio from 31:36 to 31:44) 0 plus 0 1 star, this represents the set, which does not have 2 consecutive 1s and it begins with a 0. Now, use the method use the method (No audio from 32:10 to 32:17) 0 will be accepted by something like this or I will write here. (No audio from 32:29 to 32:38) 0 will be accepted by something like this and 1 will be accepted by something like this. So, 0 1 in order to accept 0 1, this is not an initial state this is not a final state.

So, this will accept 0 1, then I have 0 plus 0 1 another 0 for 0, you can have something like this. So, repeatedly use the construction, how will you accept 0 plus 0 1 have a new initial state and a final state here. This will not be initial state, this will not be initial state, this will not be final state this will not be final state. Then have an epsilon transition here have an epsilon transition here, have an epsilon transition here. So, what is the language accepted by this it will be 0 plus 0 1. How many states you are having 1 2 3 4 5 6 7 8 states you are having. Now, you want to accept this star. So, use the construction then you will add 1 more initial state, 1 more final state this is the L 1 portion the whole portion this is the L 1 portion, you want to construct L 1 star.

So, you will not make this an initial state, this will not be made the final state and you will add epsilon transitions here an epsilon transition here, an epsilon transition here and then an epsilon transition. Systematically, if you construct using the construction we used you will construct this. Then to get the deterministic diagram deterministic, if I say what will you do? First of all you have to remove the epsilon transitions and get an N F S A. Then use the subset construction and get the deterministic number of states will be very large, the way you get number of states will be very large.

So, this is just to prove that for any regular expression we can get an N F S A with epsilon moves the construction which we used is to prove, that given any regular expression, you can construct an N F S A with epsilon moves, which will accept the set represented by the regular expression. But that may be a very unwieldy way of doing it for proof it is but generally, suppose you are given this. How will you construct the automaton? You will start from the scratch and say start with this initial state, then when you get a 0 any number of 0es you can read, because the epsilon is accepted this will also be a final state, so when you get a 1 you go to a state q 1.

(Refer Slide Time: 36:45)

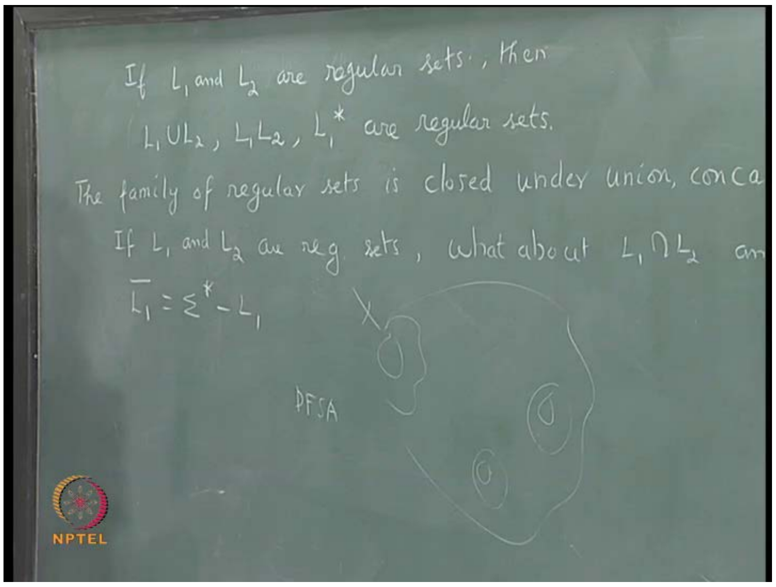


Now, you cannot get two consecutive 1s so if you get another 1 you will go to the dead state. But just this can be a final state, from this if you get a 0, you will go here, and from this if you get a 0 or a 1 you will go here. So, this will be the deterministic state diagram for this expression with has got only three states including the dead state. And it is easy in this a case, it is very easy to draw this rather than this way of proceeding. The way, we do this is to just a prove it is a proof by induction, why it is a proof by induction for the definition of regular expression? We have used induction means; please remember that it is not necessary that it should be 1 to n like that. Underline set is inductively defined what is underlying set?

The regular expression is inductively defined bases class is 5 epsilon and a r regular expressions, then induction class is you use the operators plus dot and dot is concatenation star to build more and more regular expressions. To show that that can be represented, that can be represented by N F S A with epsilon moves. Bases class, you show that phi epsilon and each symbol can be accepted by a N F S A. And then induction portion is if r 1 and r 2 can be represented by M 1 and M 2 r 1 plus r 2 can be represented by a state diagram, r 1 r 2 can be represented by a state diagram, r 1 star can be represented by a state diagram and so on.

So, in general you will not use this method for really constructing something, you will use a rather intuitive approach and try to construct the deterministic F S A straight away. In some cases, the deterministic F S A may be a bit difficult; you can construct the non deterministic F S A, and then construct the deterministic F S A.

(Refer Slide Time: 40:07)


If L_1 and L_2 are regular sets, then

$L_1 \cup L_2$, $L_1 L_2$, L_1^* are regular sets.

The family of regular sets is closed under union, conca

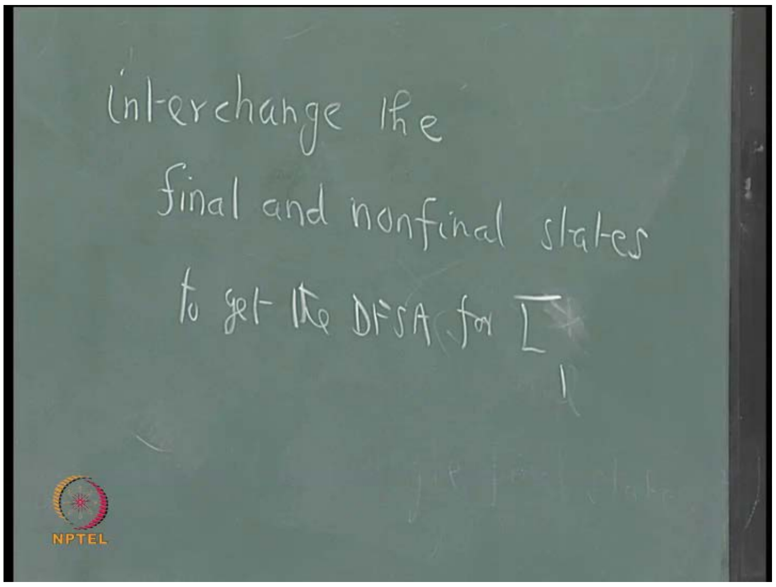
If L_1 and L_2 are reg sets, what about $L_1 \cap L_2$ an

$\overline{L_1} = \Sigma^* - L_1$

(No audio from 39:51 to 40:05) So, if L 1 and L 2 are regular sets, then what are regular sets? Regular sets are sets accepted by F S A, they are also generated by type 3 grammars. So, from this argument, we have shown that L 1 L 2 L 1 union L 2 L 1 star are regular sets. That is the family of regular sets; family of regular sets is the family contains all regular sets. So, the family of regular sets is closed under union concatenation and Kleene closure. So, because if L 1 and L 2 are regular sets L 1 union L 2 is also a regular sets. So, you say that the family of regular sets is closed under union it is closed at the concatenation it is closed under star.

Now, what about intersection and complementation if L 1 and L 2 are regular sets, what about L 1 intersection L 2 and L 1? How do you define L 1 bar? L 1 bar is sigma star minus L 1 complement set complement notation, sigma star is the set of all strings over sigma L 1 represents the subset of that. So, the complement of L 1 is the set of strings over sigma, which are not in L 1 is that is usual set theoretic notation. Now, if L 1 is accepted by a state diagram like that with an initial state and some final state. It is a D F S A. So, I am not making 1 D F S A, if L 1 is accepted by this diagram, how can you accept the complement? Just interchange the final and the non final state, has the same D F S A diagram same deterministic final state diagram.

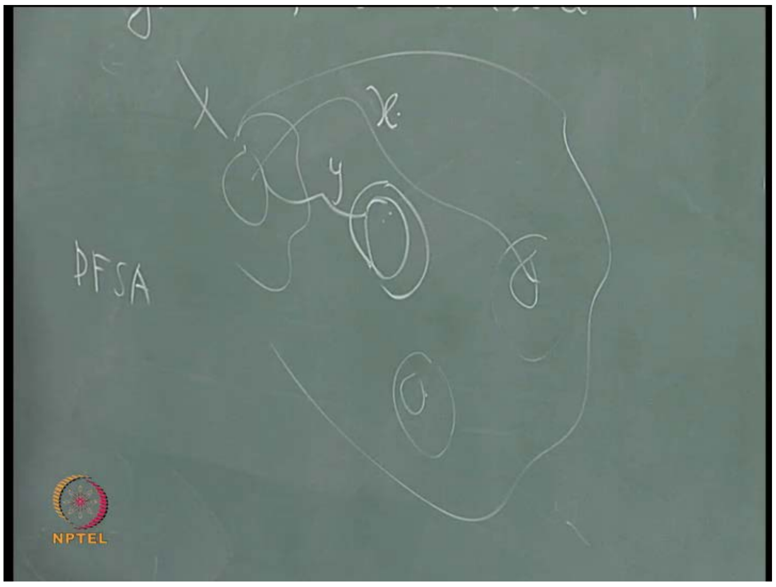
(Refer Slide Time: 43:44)



interchange the final and nonfinal states to get the DFSA for $\overline{L}_1$

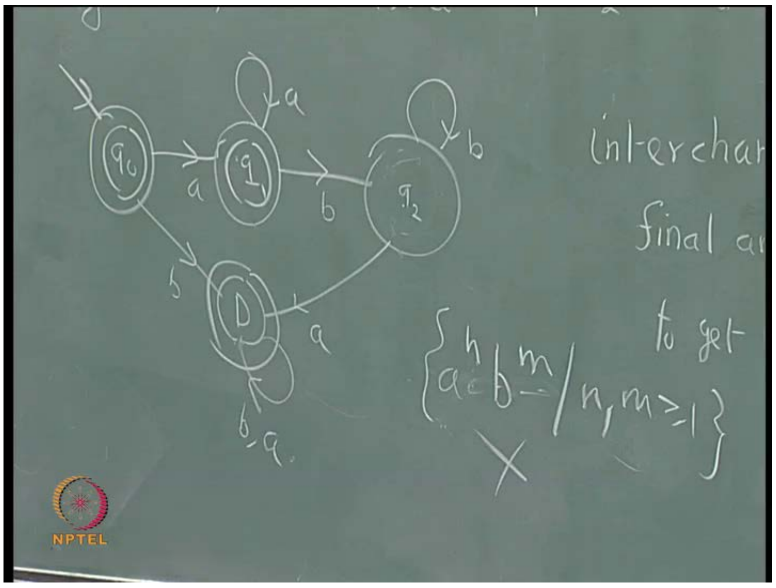
Interchange the final and non final states, to get the D F S A for L 1 bar that is all.

(Refer Slide Time: 44:21)



See, if a string is accepted by L 1, it will take you from a initial to initial state to a final state, if x is accepted it will be taken it will take you from a initial state to the final state. Now, you are making the final state and non final state. So, x cannot be accepted, if a string y takes you to a non final state and that will not be accepted by L 1, but you are making the non final states as final states. So, y will be accepted by the complement.

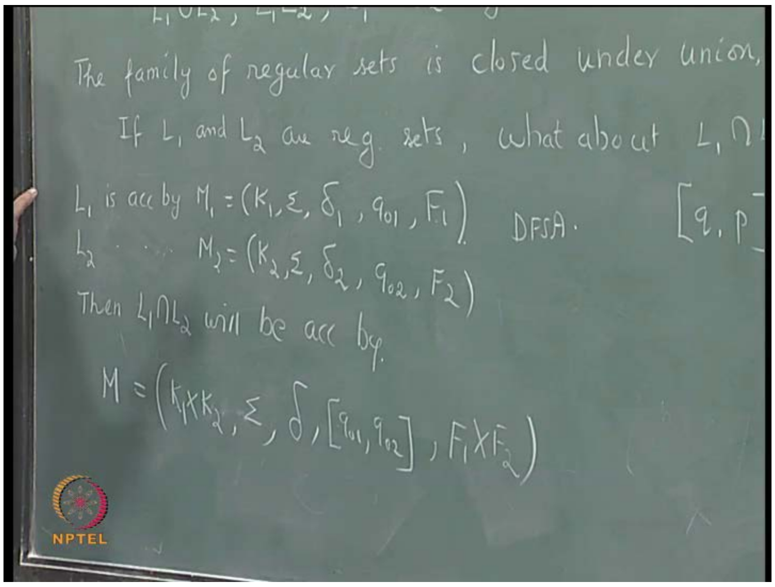
(Refer slide Time: 45:09)



Let me take a simple example, this example I have been repeating again and again, q naught a q 1 q 2 a b b and a dead state b a b a, what sort of strings will be accepted by this machine, a power n a string of a s a power n b power m n m greater than or equal to 1or the language accepted consist of a sequence of a s followed by a sequence of b s epsilon is not accepted by this machine. Now, you want to accept the complement, then you should accept all strings over a and b, which are not of this form. The complement should accept, all strings which are not of this form, from this diagram how will you get the diagram for the complement make this a non final state. And make everyone other state final state. (No audio from 46:44 to 46:51).

So, epsilon will be accepted epsilon is not there here epsilon will be accepted. A string of a, s al1 will not be in this language. But string of a, s al1 will be accepted by this diagram, then a power n b power n string of a s followed by a string of b s will not be accepted by this diagram. But if a string sequence of a s followed by a sequence of b s, then followed by something else, that will be accepted. A sequence of a, s followed by a sequence of b s then followed by something else will be accepted. So, whichever, string which is not of this form will be accepted by this diagram. So, you are just interchanging the final and the non final states to get the complement.

(Refer Slide Time: 48:21)



The family of regular sets is closed under union,

If L_1 and L_2 are reg. sets, what about $L_1 \cap$

L_1 is acc by $M_1 = (K_1, \Sigma, \delta_1, q_{01}, F_1)$ DFSA. $[q, p$

L_2 … $M_2 = (K_2, \Sigma, \delta_2, q_{02}, F_2)$

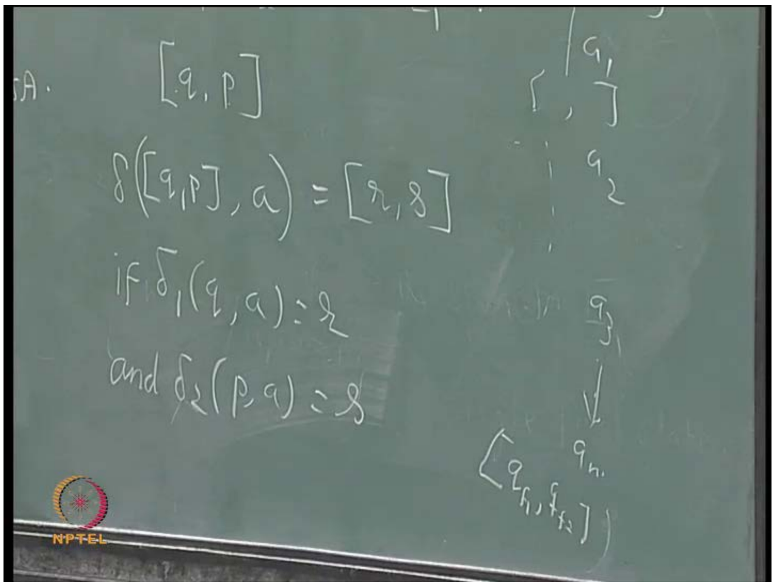
Then $L_1 \cap L_2$ will be acc by.

$M = (K_1 \times K_2, \Sigma, \delta, [q_{01}, q_{02}], F_1 \times F_2)$

Now, what about intersection, also it is closed L 1 intersection L 2 is also a regular set. (No audio from 48:08 to 48:20) Suppose, L 1 is accepted by M 1 equal to K 1 sigma, delta 1 q naught 1, F 1 and L 2 is accepted by M 2 equal to K 2 sigma delta 2 say q naught 2 F 2 sigma, I am using the same. So, L 1 is accepted by a final state automaton like this. L 2 is accepted by a final state automaton, it you can without loss of generality you can take them to be D F S A. Then L 1 intersection L 2 will be accepted by a machine M 1, M the set of states of M or ordered pairs, see if q is a state in K 1 q p states will be ordered pairs like this, q belong to K 1 p belonging to K 2.

So, the set of states is the ordered Cartesian product of K 1 and K 2 input alphabet of course, is sigma, you have to define delta the transition mapping. The initial state will be this pair that is a pair consisting of the initial state of M 1. And the initial state of M 2 and final states again F 1 cross F 2 Cartesian product of F 1 and F 2.

(Refer Slide Time: 50:47)



$[q, p]$

$\delta([q, p], a) = [r, s]$

if $\delta_1(q, a) = r$

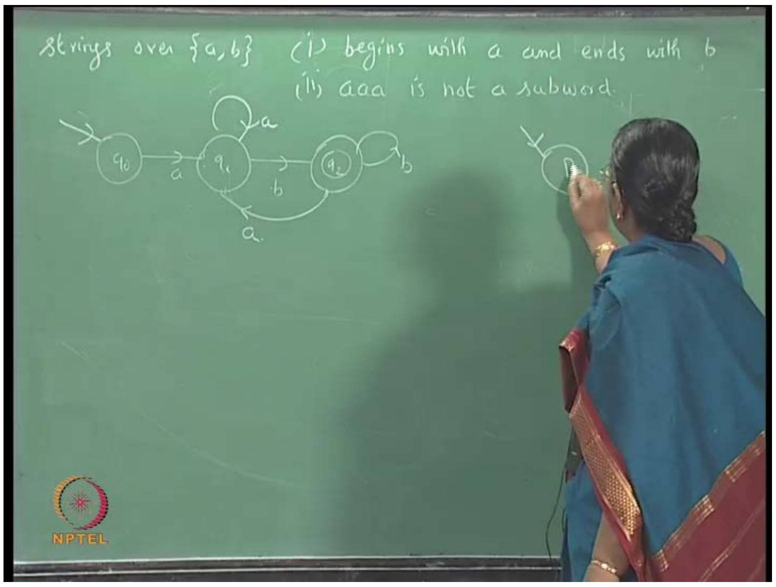
and $\delta_2(p, a) = s$

q_1
q_2
q_3
q_n
$[q_{f_1}, q_{f_2}]$

So, each state in this will be represented something like this and we have to define delta. How do you define delta of q p comma a equal to r s this r s, states are represent like this, when can you say like this, if delta 1 q a is equal to r and delta 2 p a is equal to s.(No audio from 51:20 to 51:30) So, what happens is suppose a string a 1 a 2 a 3 a n is accepted, starting from q naught 1 q naught 2 some states, you will go through and finally, end up with some q f 1 q f 2. In M you will accept a string of this form from through a sequence of states like this. Now, the first comp1nt takes care of the fact that it will be accepted by M 1. The second component will take care of the fact that will be accepted by M 2. So, only when it is accepted by both M 1 and M 2, it will be accepted by the machine.

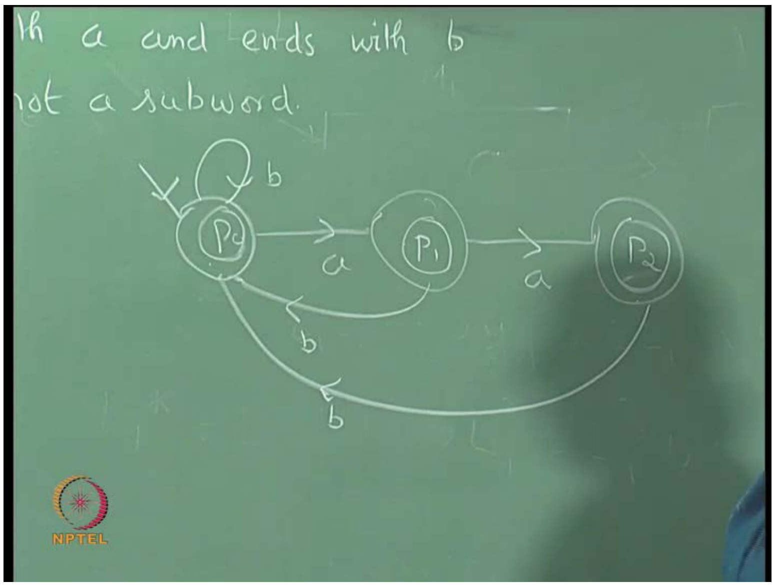
And vice versa, if it is means if it is accepted by this, it will be accepted by M 1 and M 2 and if it is accepted by M 1 and M 2. Because of the first component and the second component, it will be accepted by M. So, this makes sure that both M 1 and x the construction is very simple, we can illustrate with an example. The example, which we consider earlier (No audio from 53:00 to 53:14) strings over a b, with two conditions, which is a begins with a and ends with b a a a is not a sub word.

(Refer slide Time: 53:16)



So, one diagram will be first condition q naught a (No audio from 54:02 to 54:10) q 2 b am I correct? String begins with a, and ends with a b (No audio from 54:34 to 54:45) now, this will have only 1 a, you can have several a s followed by a b. So, this will take you to a state (No audio from 54:57 to 55:12) now a, any number of a s b and then a.

(Refer Slide Time: 55:47

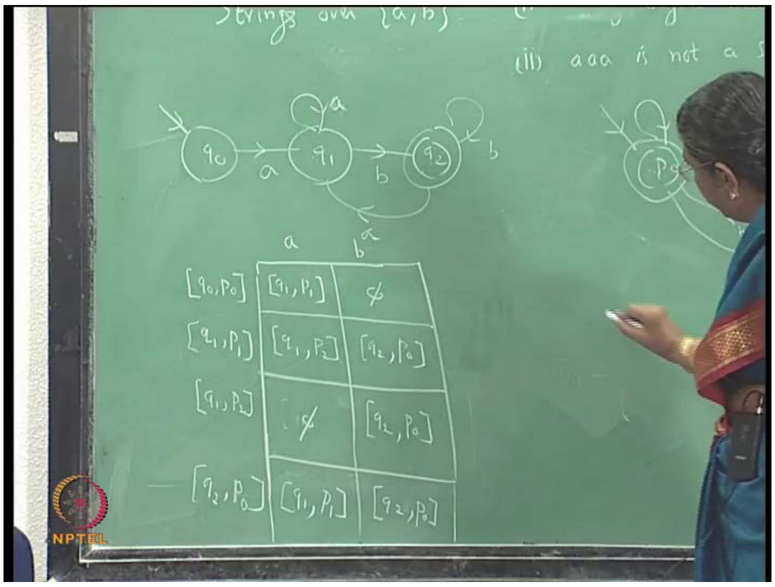


$$\begin{tikzcd}
\arrow[r] & p_0 \arrow[loop above, "b"] \arrow[r, "a"] & p_1 \arrow[r, "a"] \arrow[l, bend left, "b"] & p_2 \arrow[ll, bend left=60, "b"]
\end{tikzcd}$$

Then second condition is a I am not drawing the dead state here, because anyway dead state, if you go to it will not lead to anything. So, I am omitting the dead state, what about a second condition? p naught a a you as long as you get a b you can be here. If you get 1a you go here again, if you get a b you can go here a a, if you get a there you have to go to the dead state. But from here, if you get a, you go here from here if you get a b you here from here. You cannot get a if you get a, b you can go here. This will be the diagram p 1. I am not drawing the dead state here again, dead state I am omitting, because anyway dead state will not lead to anything. So, for otherwise you can also have a dead state and draw the diagram completely. But for convenience sake, I am omitting the dead state.

Now, what are the final states yet this satisfies the condition that a a a, is not the sub word. So, this can be a final state, just a b a b can be accepted this can be a final state this can also be a final state. So, we get this automaton for satisfying both the conditions, we start initially with the pair q naught p naught q naught is initial state of this p naught is the initial state for this.

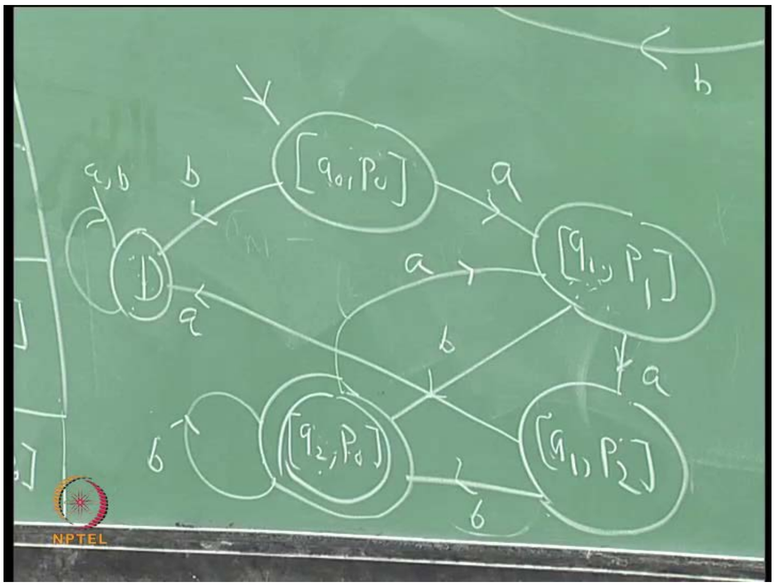
(Refer Slide Time: 57:07)



	a	b
$[q_0, p_0]$	$[q_1, p_1]$	ϕ
$[q_1, p_1]$	$[q_1, p_2]$	$[q_2, p_0]$
$[q_1, p_2]$	ϕ	$[q_2, p_0]$
$[q_2, p_0]$	$[q_1, p_1]$	$[q_2, p_0]$

Now, what are the next states q naught a, is q 1, and p naught a, is p 1. So, you get this pair here and from q naught, you cannot get a b. So, this will go to a dead state, then taking this pair q 1 p 1 from q 1. If you get a you go to q 1, from p 1 if you get a you go to p 2, from q 1 if you get a b you go to q 2 from p 1 if you get a b you go to p naught. Now, start with this pair q 1 p 2 (No audio from 58:22 to 58:30) from q 1, if you get a you go to q 1, but from p 2 you cannot get a. So, this will be really the empty state it will go to the dead state. Then from q 1 if you get a b you go to q 2, from p 2 if you get a b you go to p naught with this power on the left hand side, I mean starting with this pair q 2 p naught from q 2, if you get a, you go to q 1 from p naught if you get a you go to p 1. So, you get the pair q 1 p 1 for b from q 2 if you get a b you go to q 2 from p naught if you get a b you go to p naught.

(Refer Slide Time: 59:43)



So, we have considered all pairs and let us draw the state diagram for this with the dead state. So, you will find that q naught p naught is the initial state and the other states are q 1 p 1, q 1 p 1 q 1 p 2 and q 2 p naught, q 2 p naught. Then there will be a dead state also for the deterministic diagram. So, marking the transitions from q naught p naught, if you get a, you go to q 1, q 1 p 1. If you get a b you go to the dead state, then from the dead state, if you get a b you go to the dead state that may be denoted as phi in another row if you want. Now, from q 1 p 1 if you get a you go to q 1 p 2, if you get a b you go to q 2 p naught. From q 1 p 2 if you get a, you go to the dead state, if you get a b you go to q 2 p naught. From q 2 p naught, if you get a b you go to the q 2 p naught itself, if you get a, you go to q 1 p 1.

So, this completes the state diagram of the machine, you can very easily see that a b will be accepted a b will be accepted so any string. Now, what is the final state the final state is such that the first component should be a final state of the first machine, the second component should be a final state of the second machine. The second machine all three are final state in the first machine only q 2 is the final state. So, in this case, only this is the final state. So, you can see that a b will be accepted a b will be accepted, and if you get a a a it will go to the dead state. Any string after that will not be accepted a a a occurs as a sub word. From here again, if you get a a a another sub word it will go to the dead state. So, whenever, you get 3 a s it will go to the dead state.

So, the string will not be accepted. And reaching the final state, you have to reach through a transition labeled b and then afterwards, you can get any number of b s you want. So, the string ends with the p. The string begins with a, the first symbol, if it is a b the string will be rejected by going to the dead state. So, this diagram you can draw a straight away or you can draw these 2 diagrams and then combine them using this method. So, this shows that if L 1 and L 2 are regular sets L 1 intersection L 2 is also a regular set and can be accepted by a final state automaton.