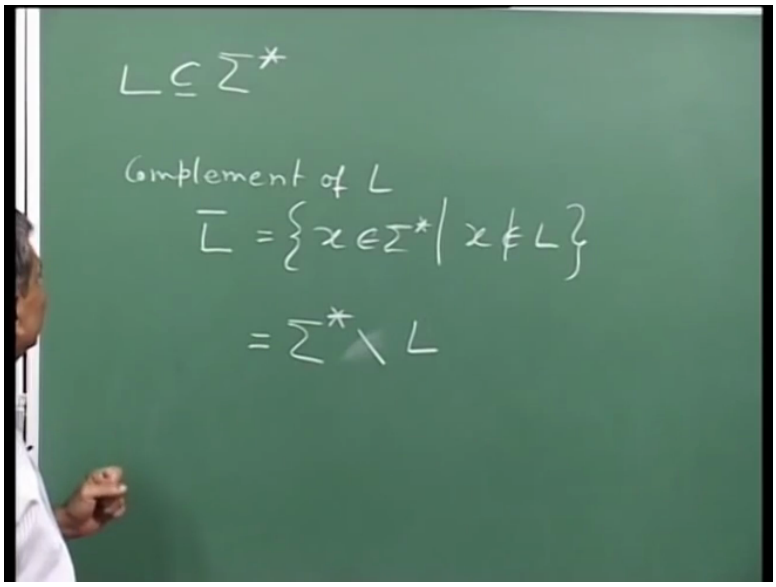
Theory of Computation
Prof. Somenath Biswas
Department of Computer Science and Engineering
Indian Institute of Technology Kanpur
Lecture 4
Regular Languages, their Closure Properties

(Refer Slide Time: 0:45)

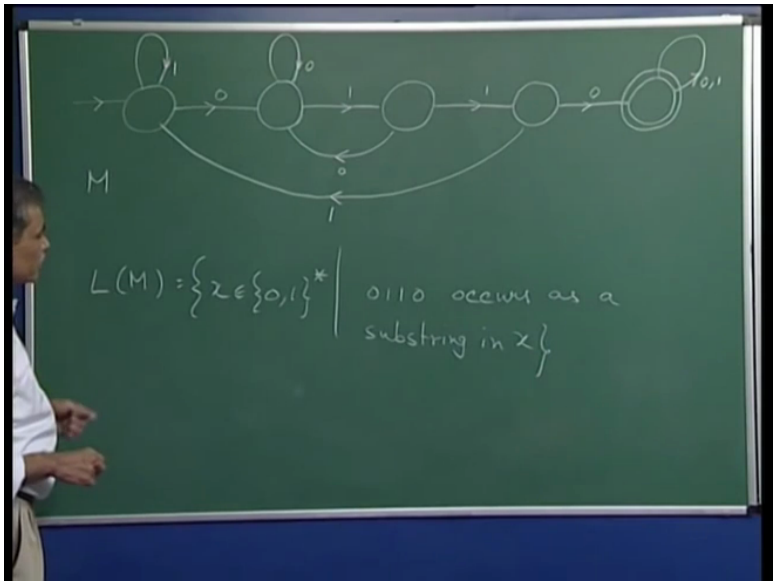


Today we will see some more examples of regular languages. Recall that a regular language is one which accepted by a DFA. It is, if you want to show a language is regular, one of the ways of doing it would be provide a DFA to accept that language. Now suppose I have an alphabet Sigma and let us say I have a language L which is made up of strings from this alphabet.

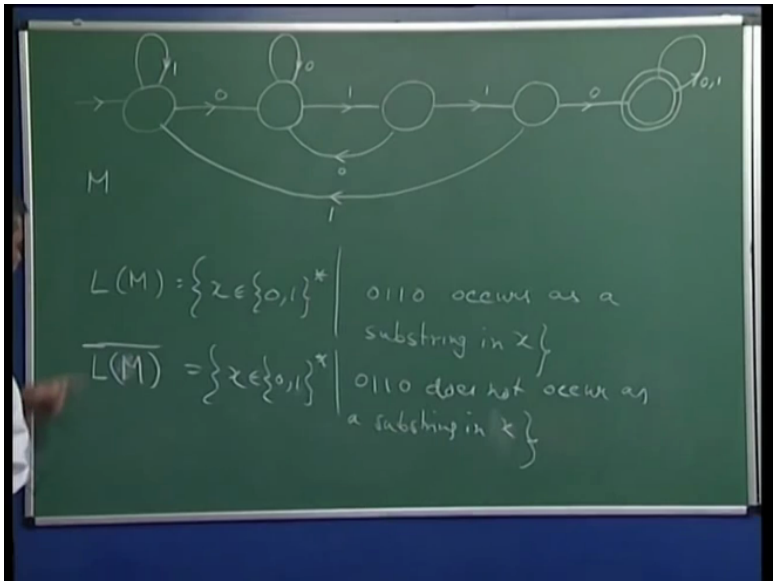
From L I can define another language which is called a compliment of L, compliment of L which is defined as, this is denoted as like this and essentially all those strings in Sigma star such that x does not belong to L. So another way of looking at this would be all those strings in Sigma star I am just rewriting it in the same in another manner. So this is the set of, so this is your Sigma star and L was here, so what is L compliment?

L compliment is all the strings which are outside of this L. So we can say that L compliment is equal to Sigma star minus L minus here is the set difference and as you know some people write it as this. So now what I am, what is very simple to see that if you give a DFA for L then you almost immediately have a DFA for L compliment?

(Refer Slide Time: 2:48)



(Refer Slide Time: 3:13)

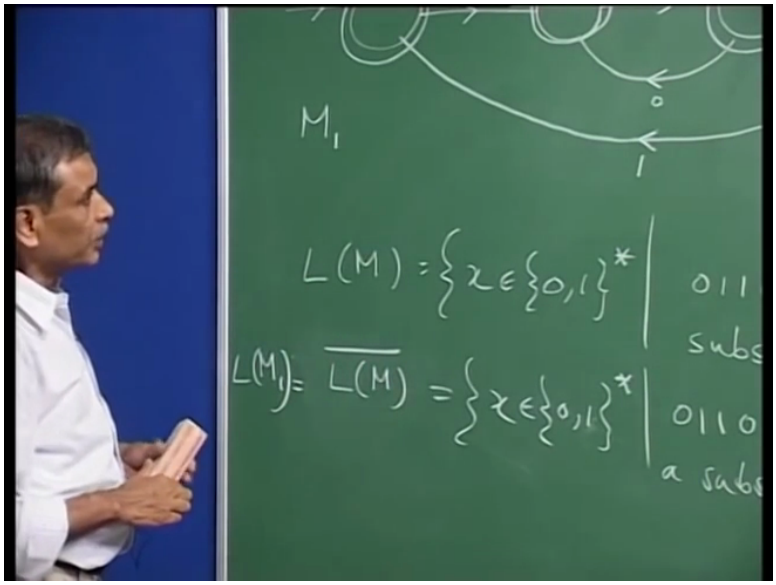


So let me give the idea through this example that we have taken last time, you recall that this DFA accepts the language, the set of all strings in which the substring 0 1 1 0 occurs somewhere in the string. Now what is the compliment of this language? So this is the language, so what is the compliment of the language accepted by M? It is by definition x in 01 star such that 0110 does not occur as a substrate in x.

Now notice a string here will take the machine, string in this language will take this machine M from its initial state to its final state. Now a string which is not in this language, where will this such a string will take the machine to it will take the machine to either this state or this state or this state or this state but not to this state therefore if I am interested in accepting only those strings which do not have 0 1 1 0 as substring then such strings will take this machine from the initial state to one of these first 4 states but never to this last state.

And therefore if I simply do this that means what I am doing here is, I am switching the final states to non-final states and vice versa that means, see this was the old, I took the old DFA and in that DFA I am switching the set of final states with the set of non-final states. So this is my new machine M1, I obtained from the old machine just by switching the set of final states with the set of non-final states and now do you see that the language accepted by M1 is precisely this language.

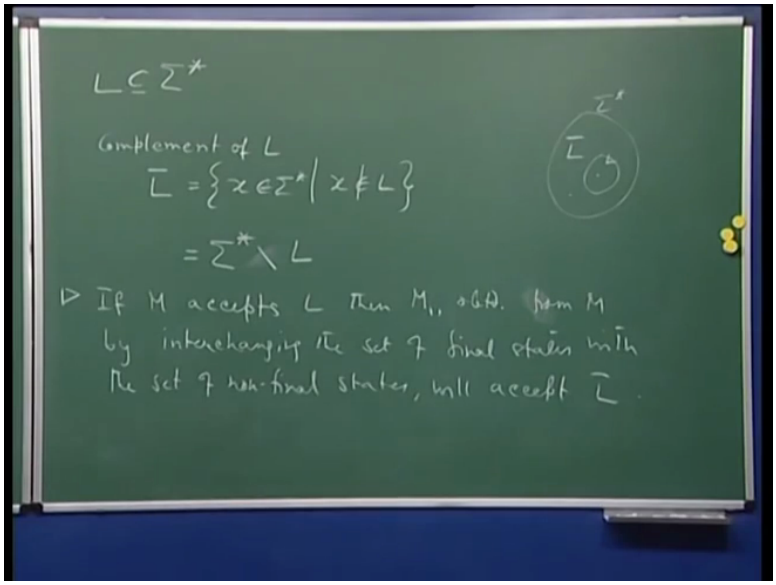
(Refer Slide Time: 5:45)



Let me write it clearly language accepted by this new machine M1 is the compliment of the language accepted by the old machine. So therefore, once more what is the reason? That since in the old machine, a string takes the machine to from its initial state to its final state if and only if 0 1 1 0 occurs as a substring the compliment set is consisting I mean consist of all the strings where 0 1 1 0 does not occur as a substring.

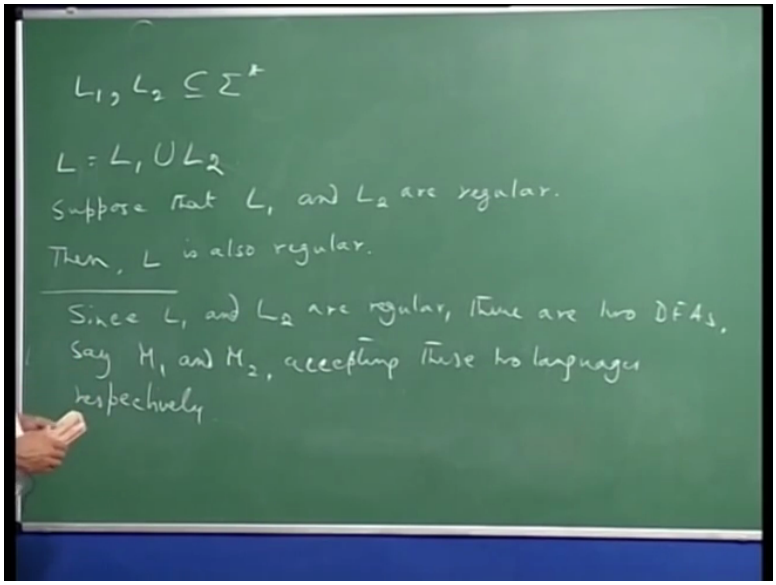
So such strings clearly would take the old machine from its initial state to one of the non-final states but if we are interested in accepting only those strings where 0 1 1 0 does not occur as a substring then all I do is to take make this flip, make the switch between the initial states and the sorry, the set of final states and the set of non-final states.

(Refer Slide Time: 7:03)



So in other words, so I can write the, what I am trying to say in a sentence like this if M accepts L then M1 obtained from M by switching or interchanging the set of final states with the set of non-final states will accept L compliment. If M accepts L then M1 obtained from M by interchanging the set of final states with the set of non-final states will accept the compliment language L, right? So therefore it is quite easy to get the compliment language if the original language was regular.

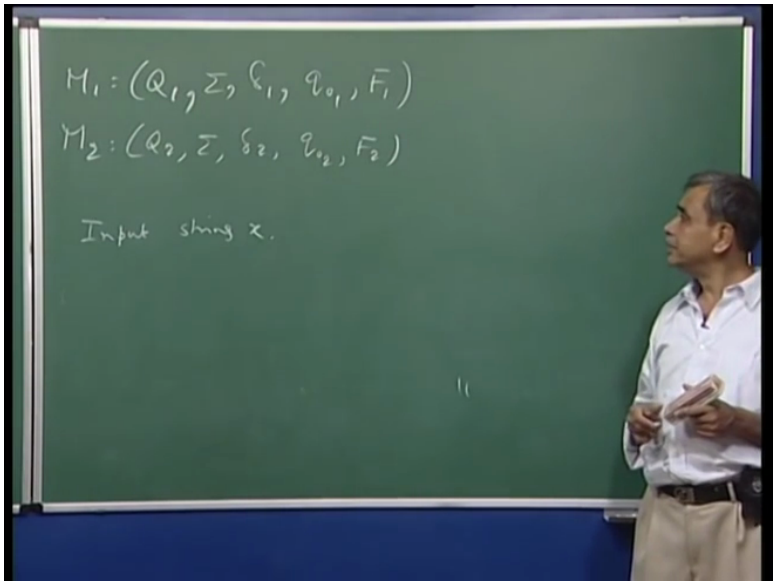
(Refer Slide Time: 8:58)



Now we can think of some other operations on sets and thereby get new languages from old languages for example, consider union they have 2 languages L1, L2 both are over the alphabet Sigma and let me consider this new language which is nothing but the union of L1, union L2 and now suppose that L1 and L2 are regular. So L1 and L2 are regular and new language L which is the union of these 2 languages L1 and L2. What I would like to show?

Then suppose that L1 and L2 are regular then L is also regular. So in order to prove this first let us understand intuitively, what we would like to? Since L1 and L2 are regular, there are 2 DFA's say M1 and M2 accepting these 2 languages, respectively, okay. So this is by definition, since L1 and L2 are regular by definition we will have 2 DFA's M1 and M2 accepting these 2 languages.

(Refer Slide Time: 11:42)

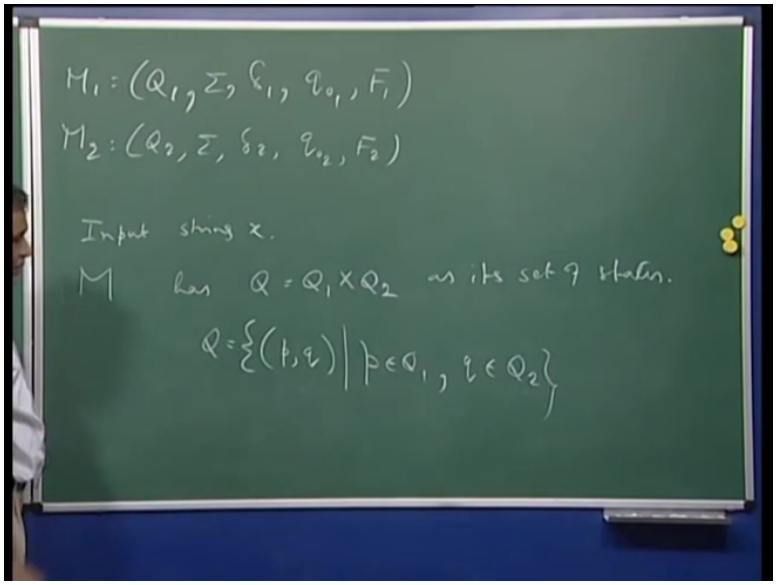


And let me say that M1 has set of states as Q1, of course its input alphabet is Sigma because it accepts L1 which is over the alphabet Sigma because it accepts L1 which is over the alphabet Sigma and its transition function is Delta 1 and let us say its initial state is q01 and its final states, that set is F1 and similarly for M2 let me write, Q2, Sigma, Delta2, q02 and F2.

Now we have these 2 machines and we would like to accept all those strings which are in this language that means if a string is either in L1's or in L2 or in both I should accept that string, my goal is to build the DFA which will accept any string which is accepted either by M1 or by M2. Now if we think of that can be done in this manner that let us say the input string is x, we run machine M1 on x, check whether it is going to its one of its final states that is F1 and then run M2 on x and check whether the machine M2 also goes to whether it goes to its final state and either of these 2 machines M1 or M2 goes to one of its final states then the string x should be accepted, that is clear.

But you see then we are going over, if this is what we try to do then we would like to go over the string x twice, once with M1 and other time with M2. Now that is not possible if I want to design a DFA because I scan my input only once, from left to Right I scan the input and my answer should be ready. So another possibility would be, instead of trying to do it serially which is what I had told you maybe what I can do is think of running both the machines parallelly and check if either of the machines goes to its, one of its final states and in that case we come up with an answer.

(Refer Slide Time: 14:48)

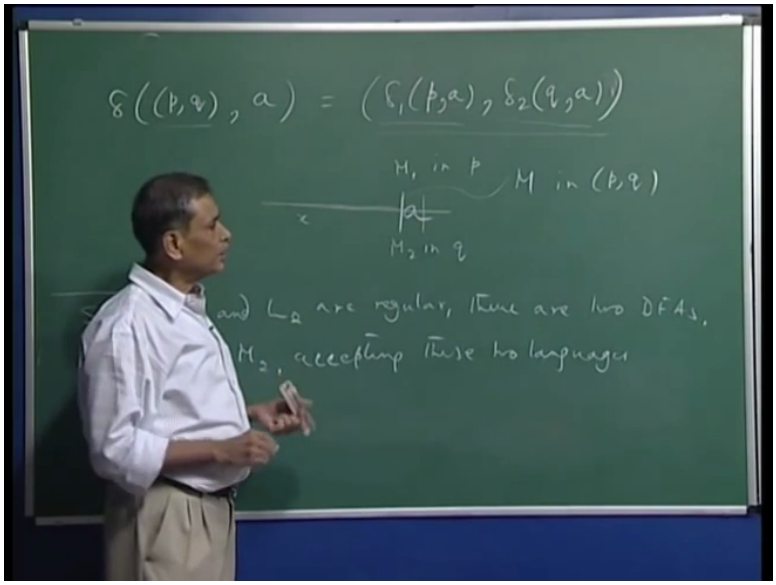


Now of course how do we do this? Now conceptually we can do so by a machine M, imagine the machine M has Q which is, okay. So imagine a machine DFA M which has its set of states as the cross product of the 2 sets of states, of the 2 old machines M1 and M2. So another way of saying what Q is? Q is P; q such that p is an element of Q1 and q is an element of Q2. Remember that Q1 is a set of states of M1; Q2 is a set of states of M2.

So my idea is something like this, the DFA I am trying to design will keep in its finite head, the information simultaneously about both the machines it will remember where the first machine would be after scanning a portion of the input and same time it will remember where the machine M2 will be after scanning the same part of the input. If this is my goal, in that case what I do is, I define the Delta of the machine M as follows.

Now remember the Delta, the state transition function takes a state of the machine, now I am trying to give the Delta for M but M states are pairs of something they are pairs and they are of the form P, q.

(Refer Slide Time: 17:00)



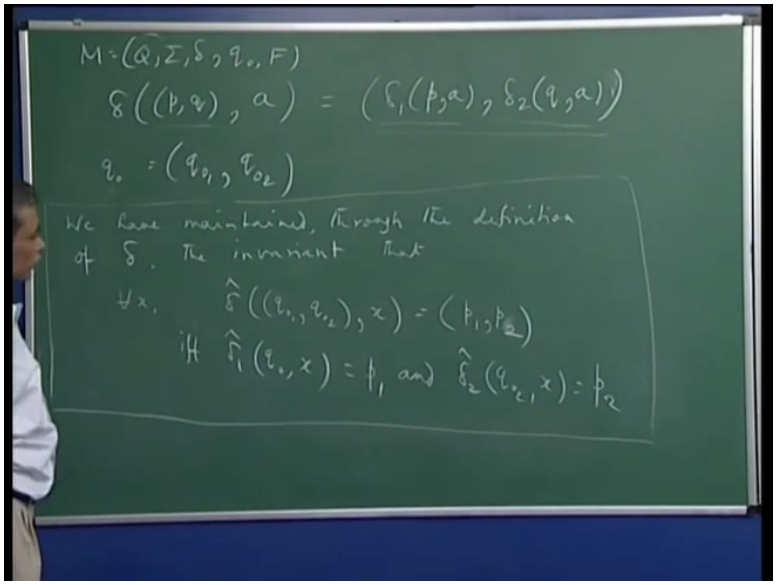
So suppose it is in a state which is remember this is one state and this one state is a state of the machine M and that one state as it happens here is a pair. First component is a state of the first machine M1, the second component is the state of the machine M2. So this pair which is a state, now in this state the machine M sees a symbol a. Now you would like to keep this as invariant that after scanning the symbol a, the machine and should remember where M1 would be and where M2 would be in the 2 components.

In that case it is fairly clear that the first part of the state of M should be, see again if this is again a pair then this is the first component of the pair and the second component of the pair's and this is a pair in which the first component is a state of machine M1, the second component is a state of the machine M2 and what we are saying our trying to do is something like this.

That suppose this is the input x and at some point machine M1, in is in state P, machine M2 is in state q. In other words what I am trying to say that suppose x was given to the machine M1 this part of the x would have taken M1 to p and the same part would have taken M2 to q and therefore this part would have taken M the new machine I am trying to design in the state P, q and now the next symbol is a.

Since after scanning this part M1 was in P, M1 now would be after scanning the symbol M1 would be in Delta1 p and M2 would be in Delta to q, a. So therefore this together will be the state in which this new machine M will be after scanning this part followed by the symbol a.

(Refer Slide Time: 20:16)

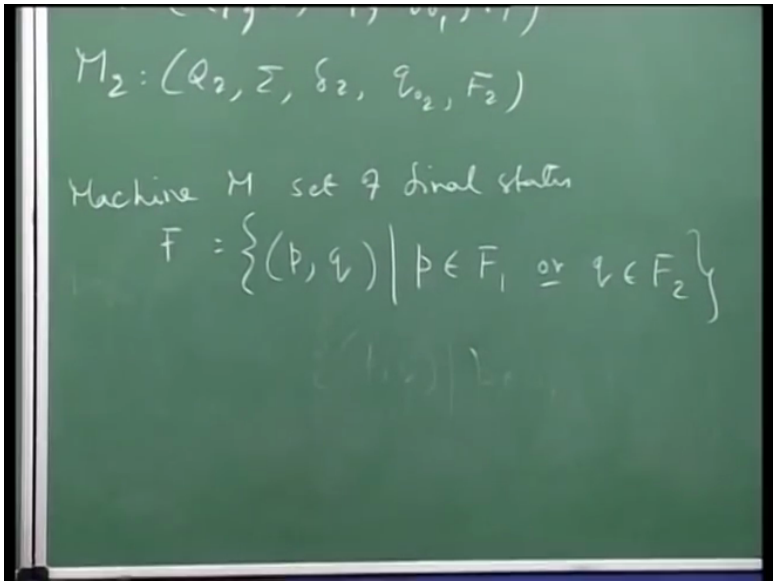


So if you would, if you do this if you define the Delta of the new machine M in this manner, so my new machine M remember has Q, Sigma, Delta, q0 and F. Q we have already said, what is the set of states for M? Is this Q1 cross Q2, Sigma is of course the alphabet, Delta is as defined here, now what should be q0? Remember q0 is a state of M and therefore it is a pair, it is a pair in which the first part is from machine M1 state and the second part of the pair is a state of machine M2 and this is what initially machine M should be and if my goal is to keep track of through the machine M simultaneously the 2 states in which M1 and M2 should be after scanning something.

So after scanning no string in other words after scanning Epsilon or in other words after beginning q0 therefore should be, remember that q01 was the initial state of M1 and q02 is was the initial state of M2, this manner what I have done? Is I have made sure initially the machine M is in a state which is a pair in which the first part is the initial state of the machine M1 and the second part is the initial state of machine M2 and we have maintained through the definition of Delta, the invariant that what is the invariant?

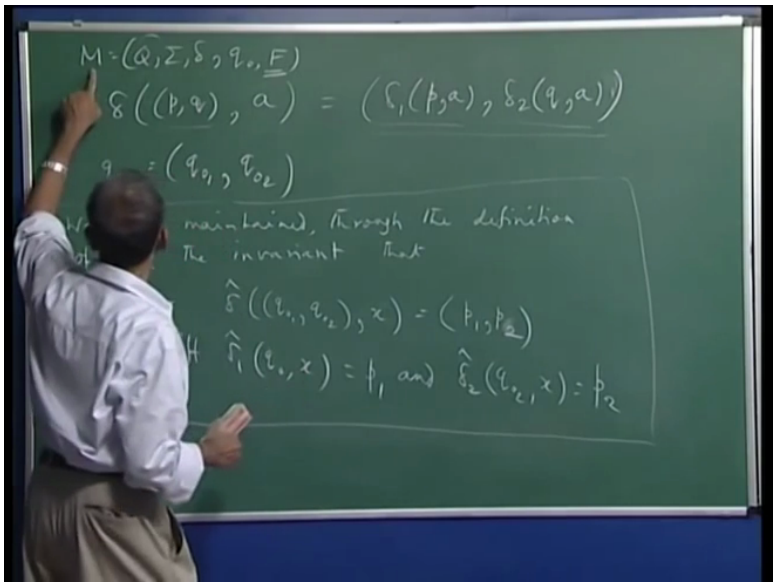
What can I assert? If this is the way I define my M, so I have defined all the way of, all these 4 components, it says I mean I can assert it that for all x Delta hat of q01, q02 which is of course the initial state of the machine M, so what we are saying is, for all x if the machine M on scanning x goes to the state which is a pair p1, p2 this will be the case if and only if Delta hat 1 of q0x is P1 and Delta hat of, for the machine to q02, x is P2 this follows from our definition, it is not very difficult to see at all and if you want to prove it.

(Refer Slide Time: 25:02)

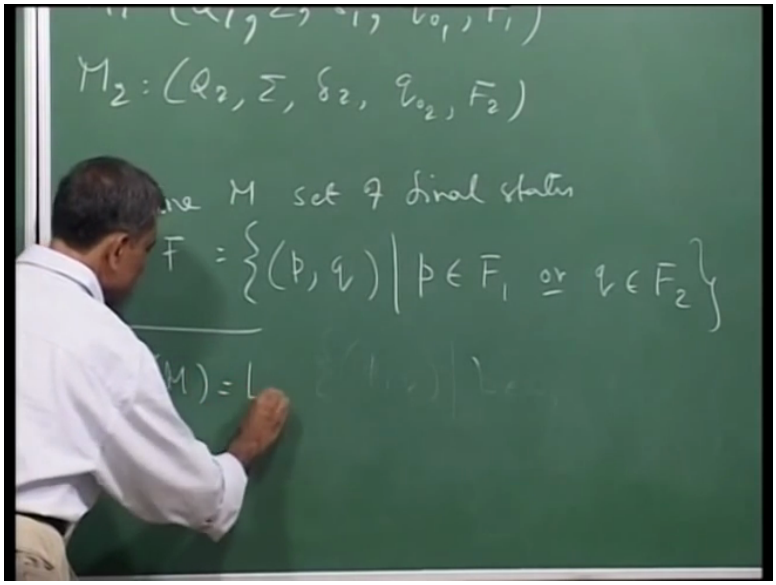


I want to prove this, if I wish to prove this statement rigorously then I prove it by induction, structural induction on the length of x and once you accept this which is not very difficult to see then if I define for machine M the final state set as, set of final states F as either, so it is a pair which either P is in F1, F1 is the set of final states of machine M1 or q is in F2.

(Refer Slide Time: 25:45)

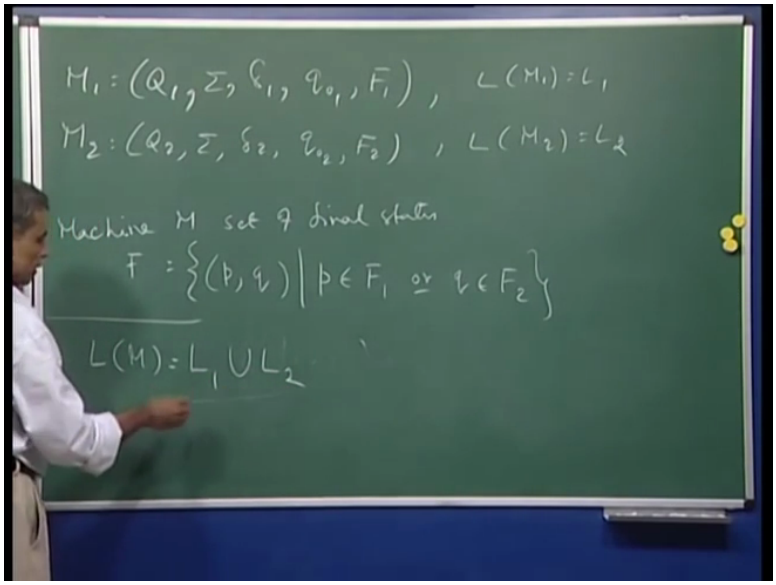


(Refer Slide Time: 25:58)



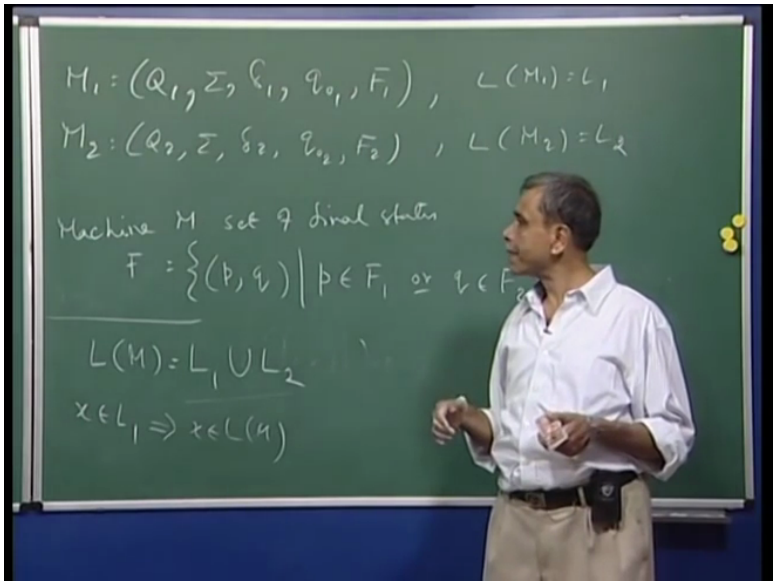
So now I have completely specified the machine M in this manner and now you will look at this and you should be able to see that the language accepted by M is the language L1, union L2, remember L1 was the language which was accepted by M1, so LM1 by definition was L1 and LM2 was L2 we define the machine M in this manner and we have managed to keep this and therefore this new machine M defined as this is going to accept L1 union L2, why?

(Refer Slide Time: 26:52)



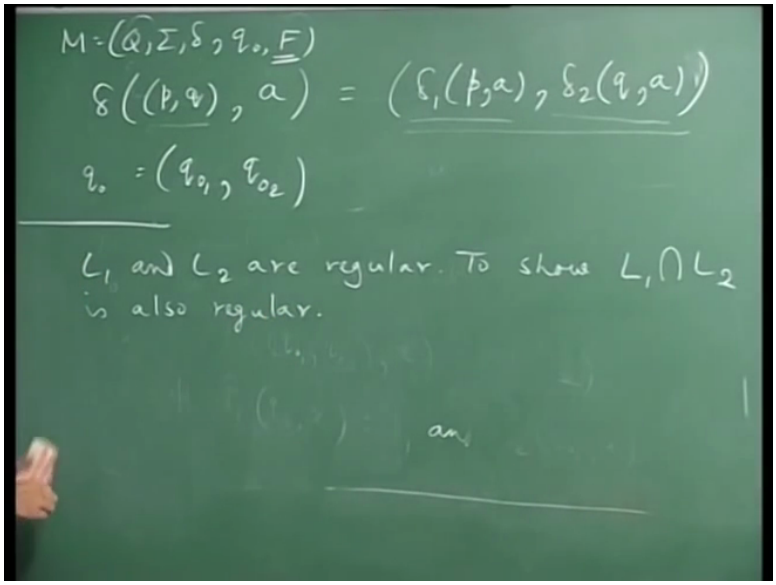
Very quickly because again this is a set equality, so let us suppose take a string from this side. So a string in L1 union L2 it means that that string is in neither in L1 or in L2. So without loss of generality suppose that string x is in L1, can you see that this string x, so let us say x is in L1 and I want to prove that then x is in language accepted by M, why? Suppose x is in L1 that means Delta 1 hat of q0, x is in some state P1 and since x is in l1, P1 is one of the final states of M1, right?

(Refer Slide Time: 28:39)

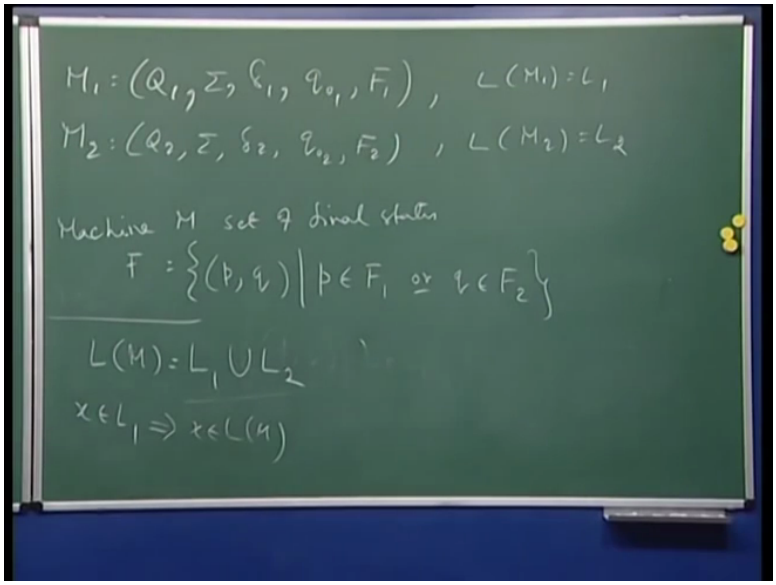


So this machine M on x would go to P1, P2 in which the first part is a final state of M1 and that state is going to be a final state of M because this is the definition of set of final states of M and therefore that string x is accepted and in a manner similar you can prove that if a string is here, that means if a string is accepted by the machine M then it has to belong to either L1 or L2 or perhaps both therefore we have proved that if we have 2 languages L1 and L2 both are regular then there union will also be a regular language.

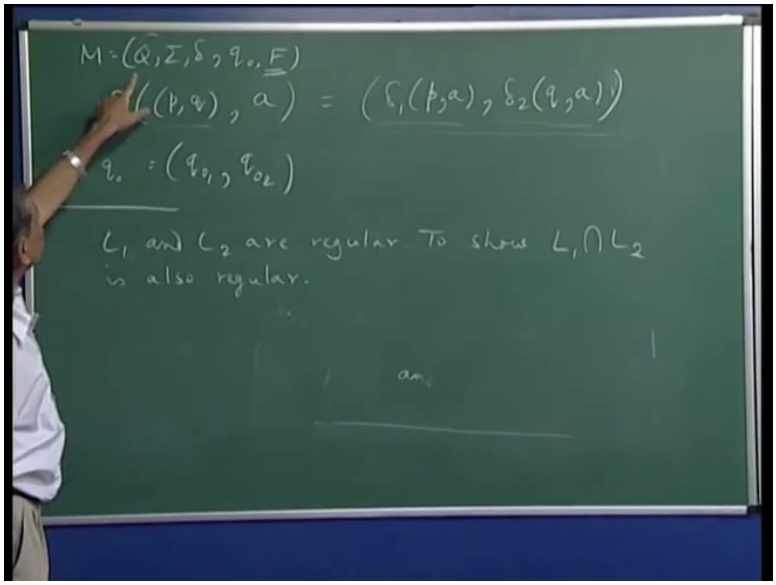
(Refer Slide Time: 29:18)



(Refer Slide Time: 29:29)

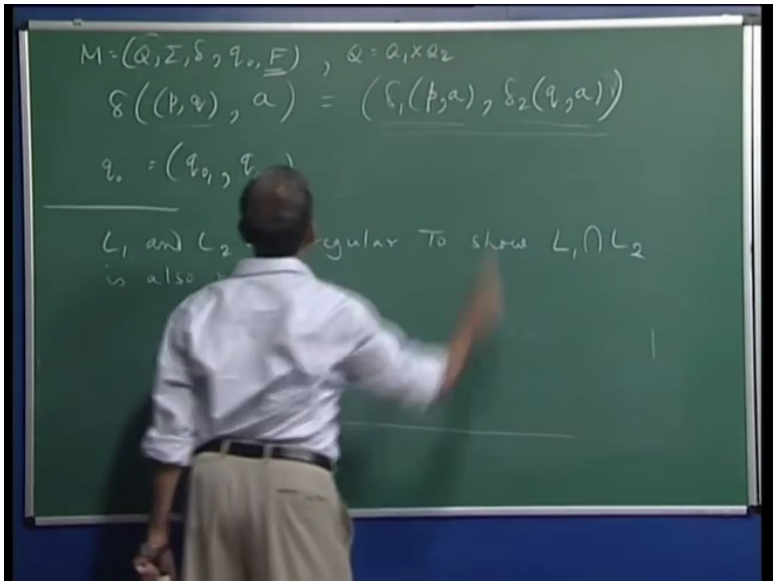


(Refer Slide Time: 29:36)

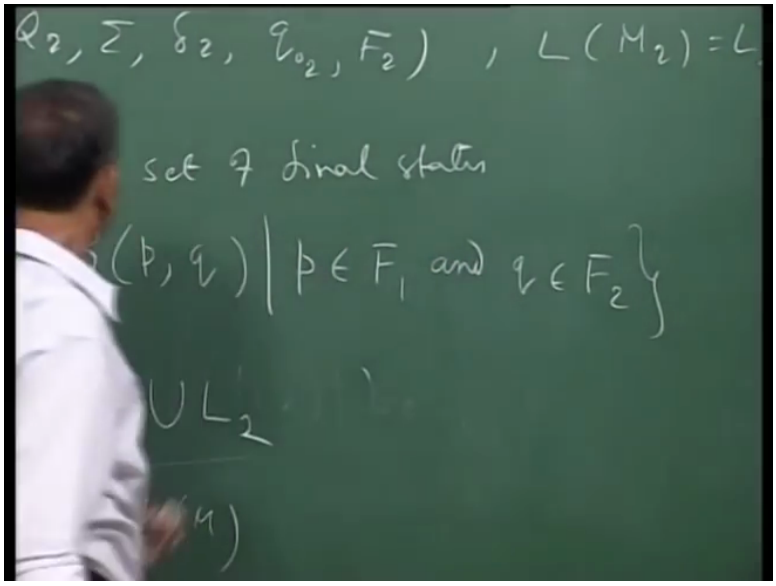


Now using very similar idea, in fact the definition of the new machine will not change much. So suppose I wanted to prove that L1 and L2 are regular to show now that L1 intersection L2 is also regular this again can be proved in a manner very similarly, I start with 2 DFA's, one for L1 one for L2 exactly like that, let us say M1 is for L1, M2 is for L2 then I take new machine M whose state keep track of the 2 states, one of M1 and one of M2 the other of M2 and therefore it states are the set of pairs exactly like this as we have defined the 4, right?

(Refer Slide Time: 30:02)

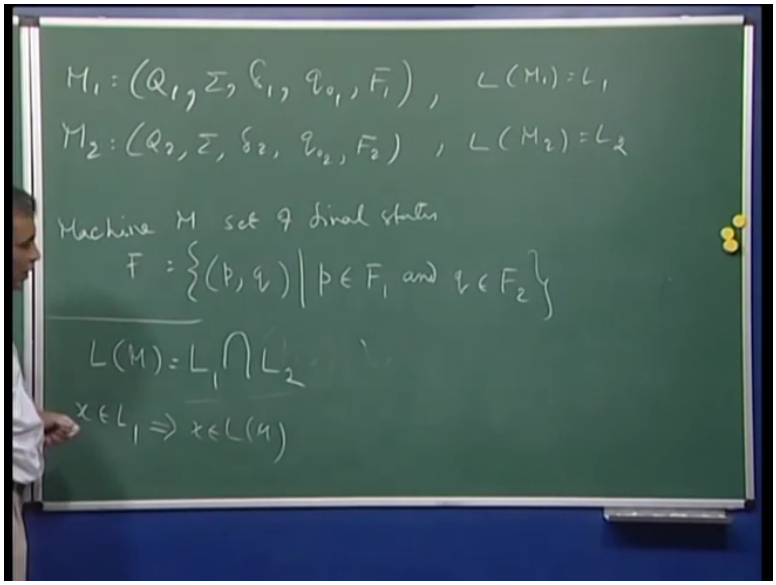


(Refer Slide Time: 30:11)



Here remember that Q was q1 cross q2 and now we just changed this or the set of final state definition into and, the same invariant remains true, so the new machine I am designing it is the same as the union machine the one machine I designed for the union for accepting the union of the 2 languages.

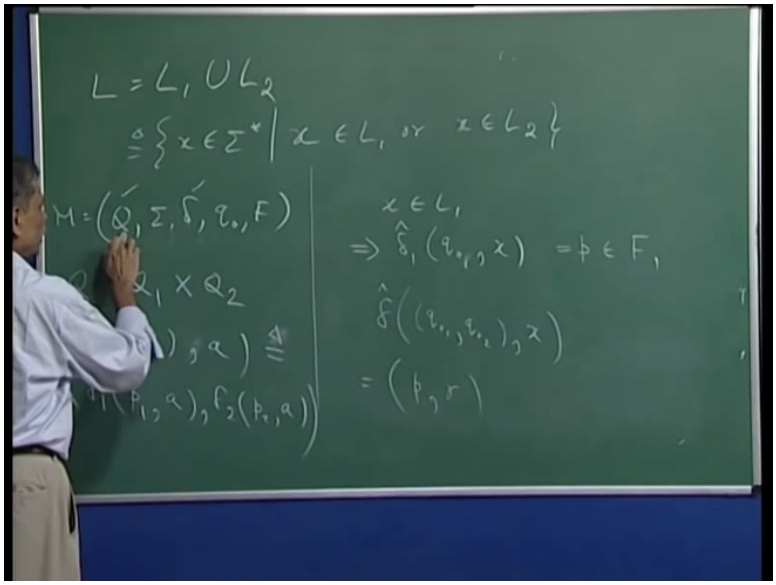
(Refer Slide Time: 30:44)



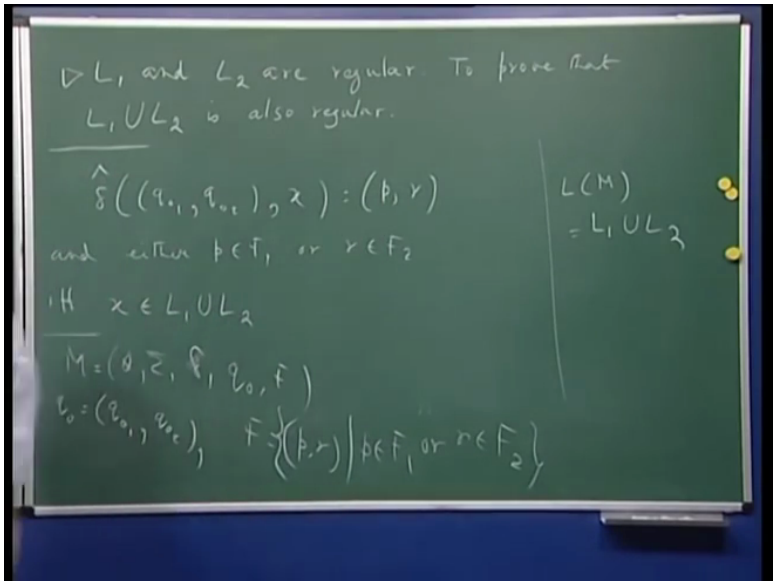
Only thing I changed was the definition of F and now you should be able to show that this new machine M precisely accepts the intersection of L1 and L2 then the proof will be something similar to that now consider a string x which is here that means it is both in L1 as well as in L2, so x is in L1 and, x is in L2 and I would like to show that x is accepted by LM and on this x, the machine M will go to a state which is a pair Pq.

Since x is accepted by M1, P has to be a final state of M1 and since x is accepted by M2, q has to be a final state of M2 and therefore this pair P,q is such that P is a final state of M1, q is the final state of M2 but precisely such states are the final states of my machine M and therefore the string x will be accepted by this machine. So anything here L1 intersection L2 once I changed, only thing I changed in the old definition, union definition is the definition of F and what I get is a machine which accepts the intersection of the 2 languages L1 and L2.

(Refer Slide Time: 32:24)



(Refer Slide Time: 33:07)

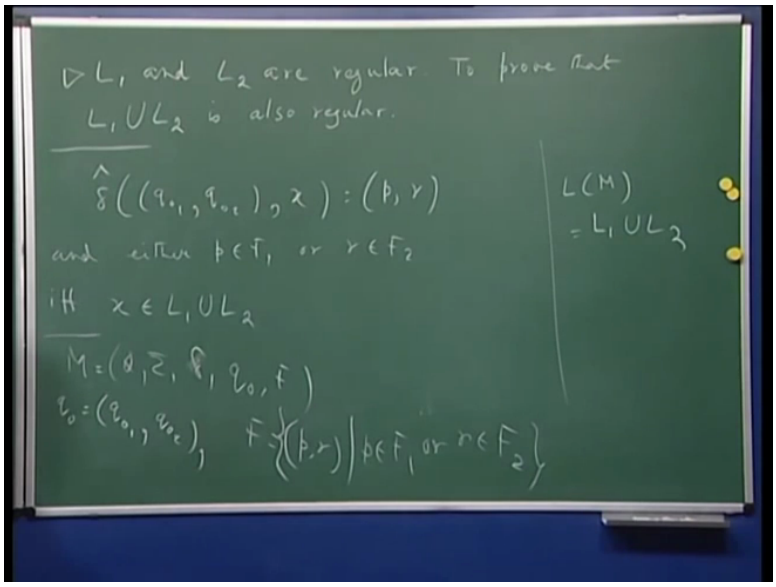


Here, Q here and now let me define its initial state and its set of final states as follows, we will say that q0 is the pair, q01, q02 and F is all p,r such that either P is in F1 or r is in F2 then once I define q0 the initial state of M as this and the set of final states as those tuples in which at least one of the components is one of the final states of the previous machines M1 M2.

This is not difficult to see that the language accepted by M is L1 union L2. Actually the statement if you prove it rigorously then immediately that follows and proving rigorously the statement is also not difficult because you will just use the properties of Delta 1 and Delta 2, the 2 transition functions for the 2 machines and therefore what I have? What I have is that if

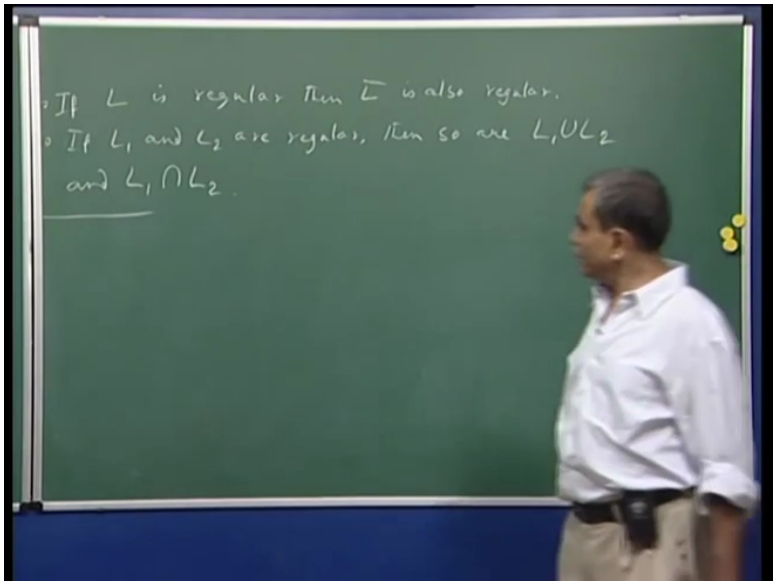
you give me 2 DFA's then in a very simple construction I can create another DFA M which is going to accept language which is the union of 2 languages, of the 2 DFA's.

(Refer Slide Time: 34:40)



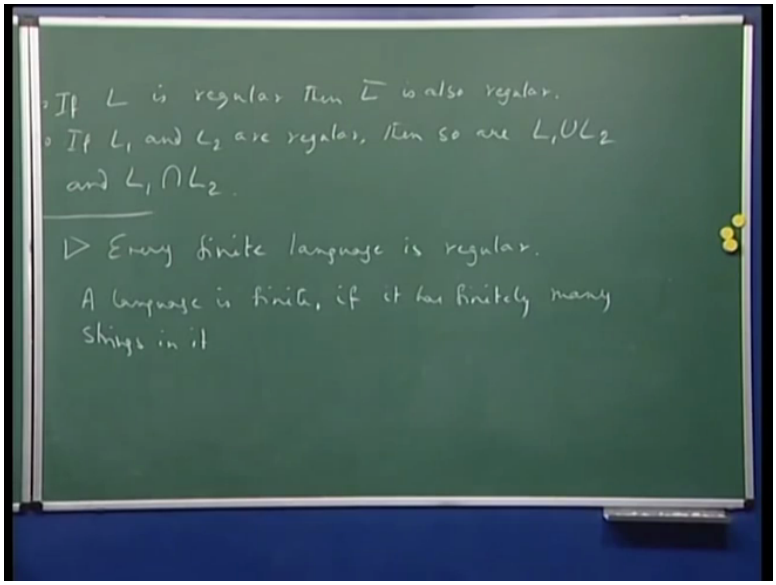
So therefore what we have done is to show this, if L1 and L2 are regular then you take the 2 DFA's which accept not the 2 DFA's, DFA for L1 take a DFA for L1 take a DFA for L2 and then use this construction and then argue that the new machine that we have obtained he will accept L1 union L2.

(Refer Slide Time: 35:30)



Now what about intersection? So we have seen number of facts about regular languages, one is that if L is regular then compliment of L is also regular the other facts we just proved that if L1 and L2 are regular then so are the 2 languages obtained by taking union and the intersection, okay. Besides this we have also seen a couple of very simple examples of regular languages like binary strings in which we have even number of 0's, also we took the example that binary strings in which the number of 0's as well as number of 1's both these numbers are even.

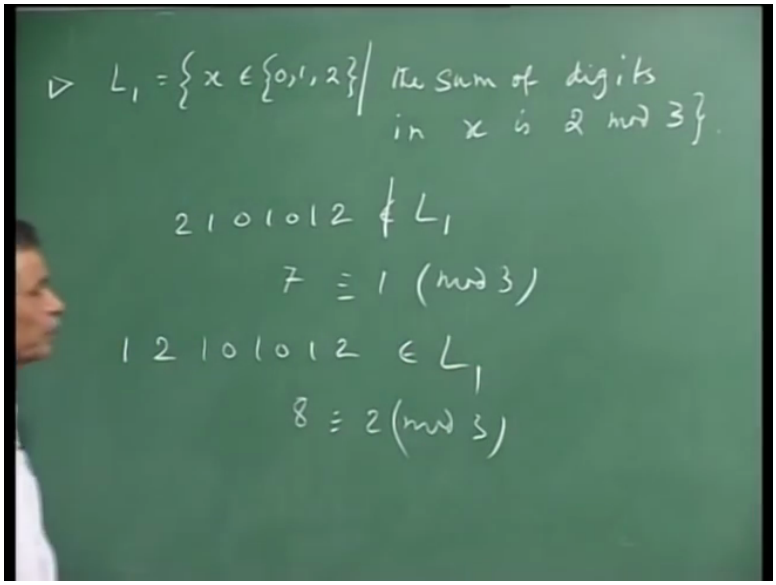
(Refer Slide Time: 37:07)



And it is fairly simple to prove that every finite language is regular, what do we mean by finite language? A language is finite if it has finitely many strings in it, very similar to the definition of finite set. You said a set is finite if it has finitely many elements and similarly a language is finite if it has finitely many members in it. Example would be, so this language L it has exactly 3 strings 0 1 0 0 1 and 1 0.

So it has 3 strings in it and that is a finite number therefore this L is finite. You can prove this without difficulty to fix your idea first maybe you can look at this example itself and how would you design a DFA to accept this language L and then you can generalize that argument that given any arbitrary finite language they can have a DFA to accept that language, okay. So we have a number of languages now we understand which are going to which are going to be regular.

(Refer Slide Time: 39:07)

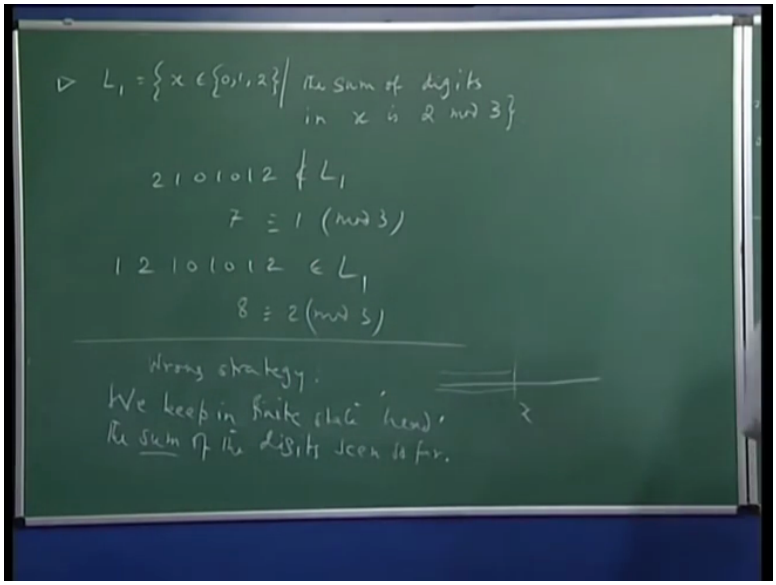


Before we precede to something else another topic, let us take few more interesting examples of regular languages. So let us take this, suppose I have this language L1 which are strings over 0, 1 and 2 and what I want is to define all those access to be in L1, if and only if the sum of digits in x is 2 mod 3, okay. So let us take some example strings 2 1 0 101 2, is this in L1?

Well, what is the sum of its digits 2 1, so 3, 4, 5, 7? So sum of its digits is same 7 is mod 3, right? It is not 2 mod 3 and therefore this is not in L1, okay. On the other hand just the other one I mean I just put 1 in front, 1 2 1 0 101 2 the sum of digits now is going to be 3, 1, 4, 5, 6, 8, so this is of course 8 is 2 mod 3, so therefore this is in the language L1.

And I would like to design a DFA for this language, so through these 1or 2 examples, now I am trying to fix once for all our idea about how to go ahead and design DFA's. By now you understand that if the DFA tries to remember the sum of the digits it has seen so far after scanning the prefix of an input string then that strategy is not going to work, why? So what is the strategy I have in mind?

(Refer Slide Time: 41:48)

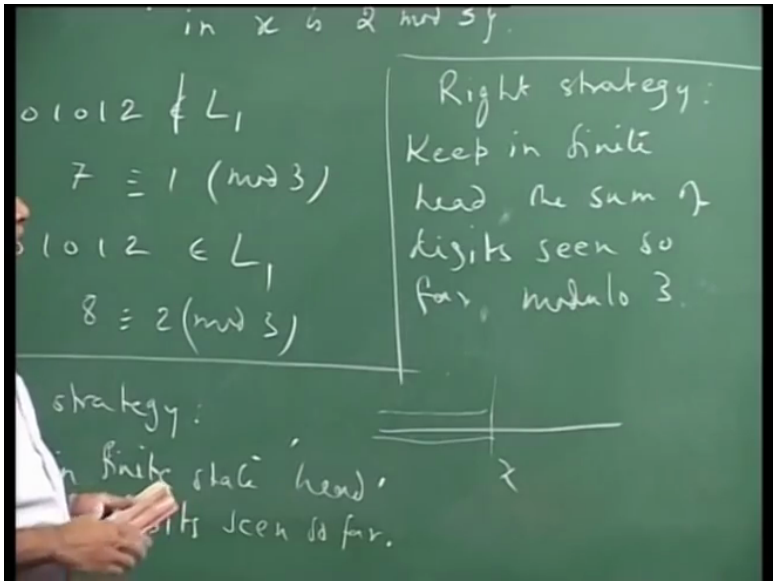


The wrong strategy in any case, suppose the string x is given to you and remember that idea of a design of a DFA is as we are scanning this string x from left to right we keep some information in the head and someone says that let us the information that we keep in the finite head of the machine is that we keep in finite state head that is the mind of the machine that is one of the states of the machine is going to keep track of the sum of the digits seen so far.

So you have seen all these digits, suppose you try to remember in the finite head of the machine this sum. Now that is wrong strategy, why? Because this sum can be arbitrarily large remember think of a string which is going from here to a very far ahead then that sum of the digits in that string is going to be a huge number if you take put some more symbols after that the number is going to become larger and larger.

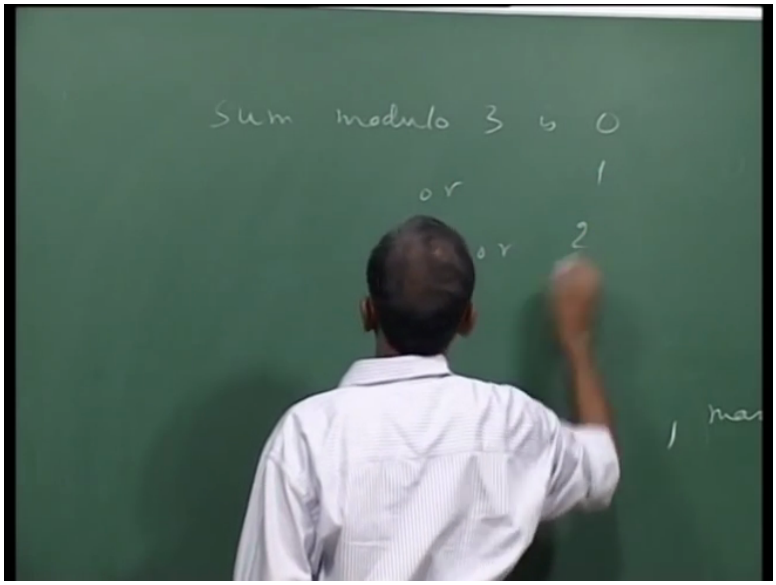
So that sum cannot be kept in the finite head of any DFA, however many states it may have it can store only finite amount of information and how do you solve this problem?

(Refer Slide Time: 44:04)

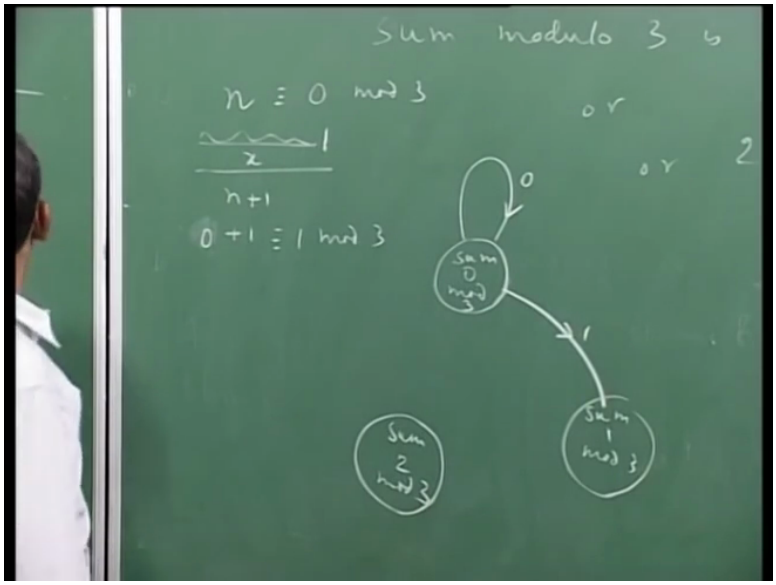


Well the right strategy would be, do not keep the sum but keep the sum modulo 3, keep in finite head the sum of digits seen so far modulo 3. If that you do at any given time, what this sum can be?

(Refer Slide Time: 45:00)



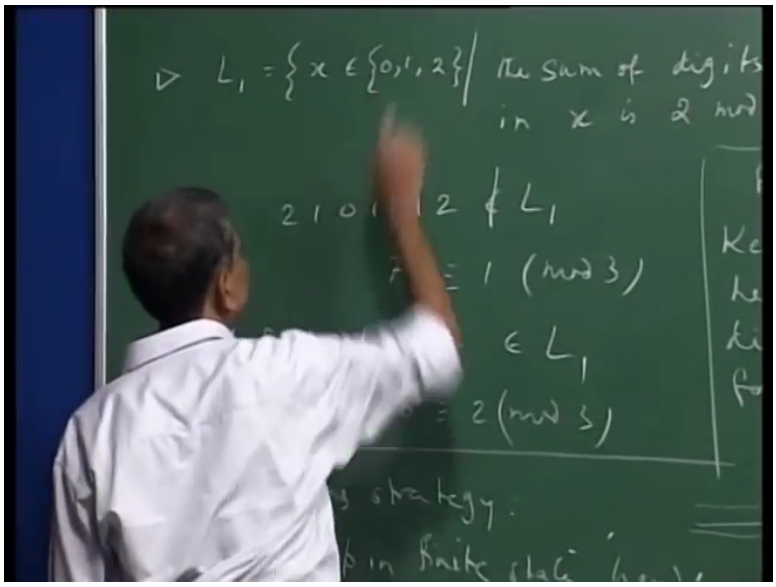
(Refer Slide Time: 45:19)



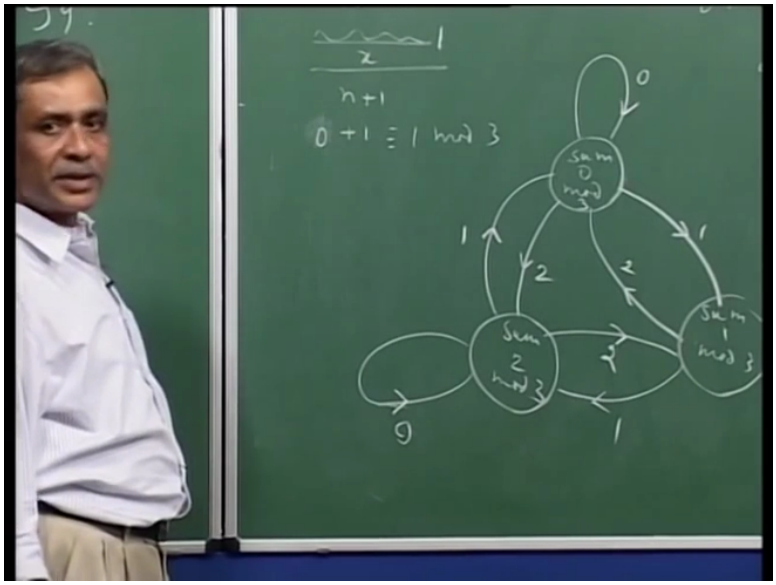
Modulo 3, there are only 3 possibilities either the sum is 0, sum modulo 3 is 0 or 1 or 2, right? These are the only 3 possibilities. So let us keep designate 3 states, so this state is the sum is 0 mod 3, sum is 1 mod 3 and sum is 2 mod 3. Let us say we are here the sum is 0 mod 3, now 1comes where should you go to, so let us say some of the digits seen so far is n which is since you are in this state then that some is 0 mod 3, now 1 came, so sum string x was there and up to this the sum was n and now 1 came therefore the sum up to this is going to be n plus 1.

So if n is 0 mod 3 then clearly n plus one is 1 mod 3 because n is 0 mod 3, 0 plus 1 is 1 mod 3, right. So it is easy to see on 1 from here you will go to this state where the sum is 1 mod 3 on 0 where will you go is not difficult to see by similar argument that will remain in this state and if a digit 2 comes.

(Refer Slide Time: 47:00)



(Refer Slide Time: 47:06)

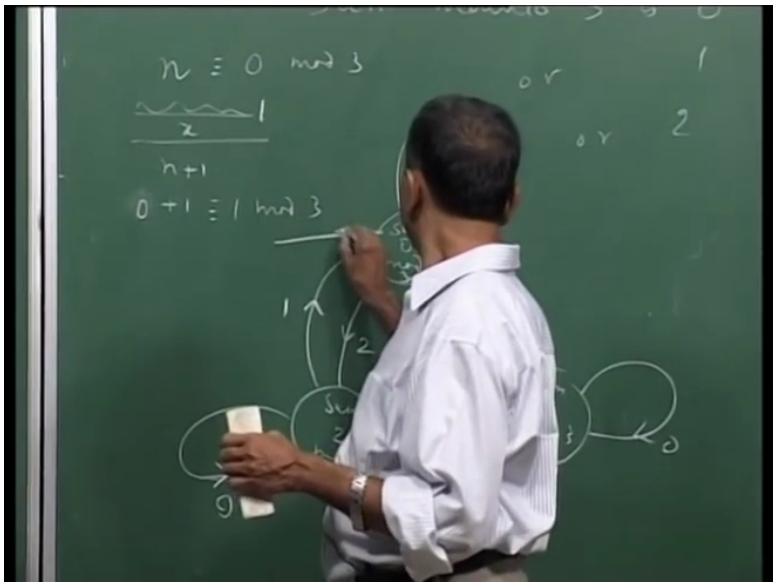


Remember that string here consists of these 3 symbols, so if 2 comes you will come here from this state if 1 0 comes you will remain here, if 1 comes you will come to this state, if 2 comes when you are here you will go back to, go to this state and in this state, so **e** you are

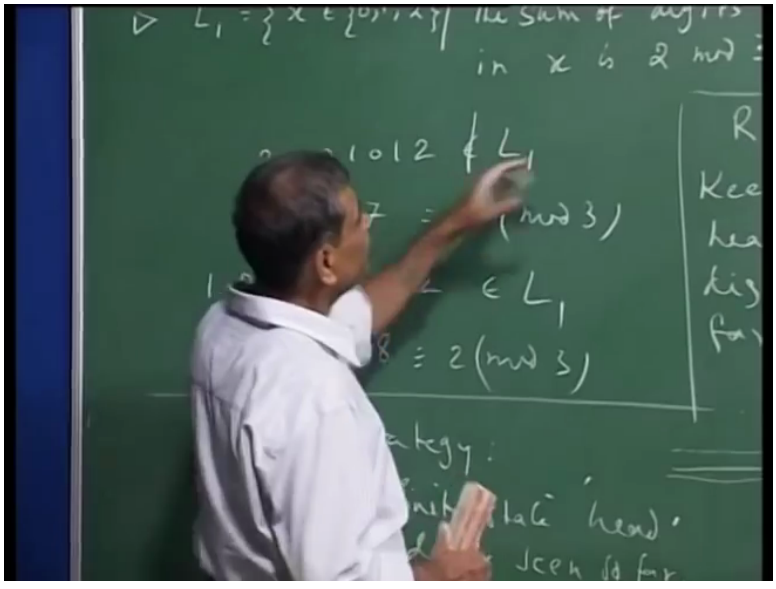
here that means what you have seen so far those digits adapt to a number which is 2 mod 3 and then 0 came where are we?

Then 0 came then of course you will remain here because the new sum is also going to be the old sum and therefore if the old sum was 2 mod 3 the new sum is also going to be 2 mod 3. If 1 comes then you will be here and if 2 come then you are going to be here, so this was 2. So this almost specifies the language accept the DFA accept 2 things I have given the transition function and have given the set of states but we have not said what is the initial state?

(Refer Slide Time: 49:00)

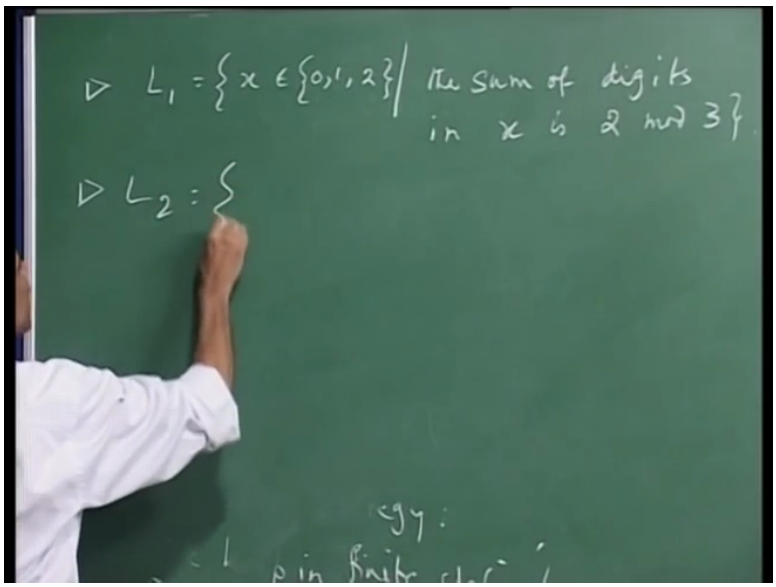


(Refer Slide Time: 49:14)



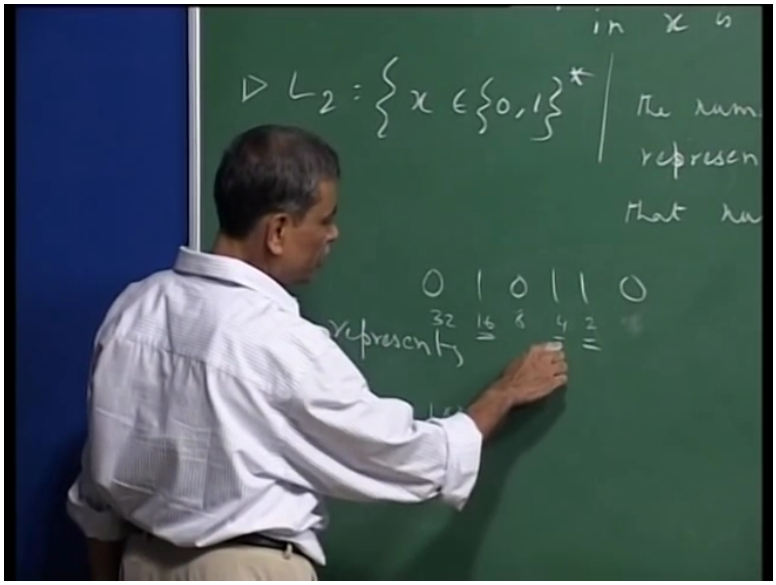
Initially with State you should be in, remember when you start you have seen an empty string, the sum of the digits there, there are no digits therefore the sum is 0, so you should be here which is the 0 mod 3 and you want to accept only those strings which take the machine where the sum of the digits is 2 mod 3 such strings will take the machine to this state and therefore we make this accepted. So therefore this machine will accept this language, okay.

(Refer Slide Time: 49:45)

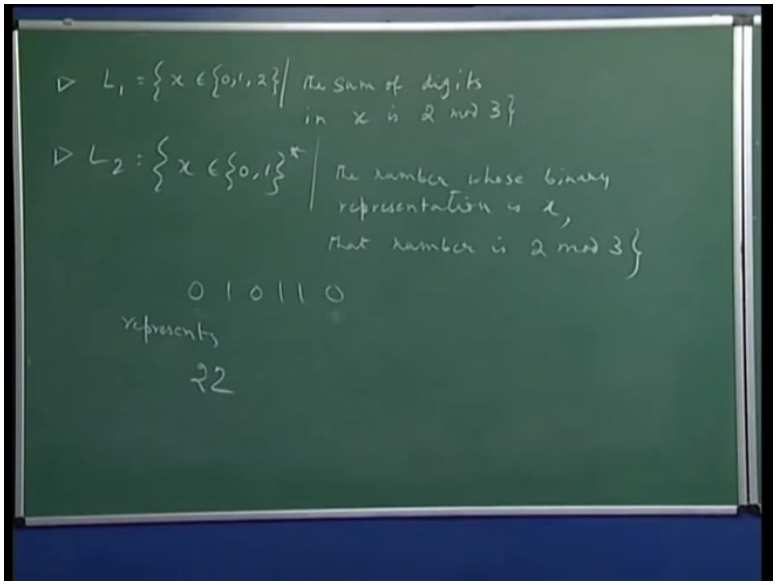


Now let us make this particular example a little more involved by defining another language L2. Now this time the strings are binary strings and will say the number whose binary representation is x that number is 2 mod 3. So what we mean by this is something like this. So we are talking of strings which are over 0, 1. So these are binary strings, so let us say I have 0 101 1 0, I can see such a string to be representing a number whose binary representation is this string and what is that number?

(Refer Slide Time: 51:27)



(Refer Slide Time: 51:58)

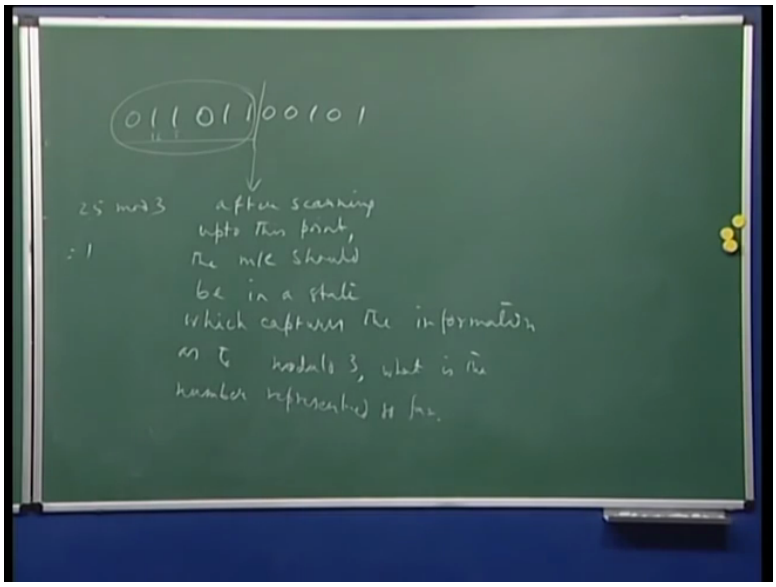


You know that, so you will take this is 1 multiplied by 0, so 1 multiplied by 0 plus, so 1 multiplied by 2, 4, 8, 16, 32, so basically we have a bit 1 here, here and here. So 16 plus 4, 20 plus 2, 22. So therefore we can say that this string represents the number 22. It is just the binary representation of this thing is of course usually this but if you put 0 in front the number does not change, so this is it.

So things are similar, in that case what is, we expect we should be able to have a DFA and indeed we will be able to have a DFA and again what we can do? Is to keep track of, not the number represented by the string you have seen so far but that number in modulo 3 that

information is what we have, we will keep in the finite head. So therefore the machine for L2 again will have 3 states.

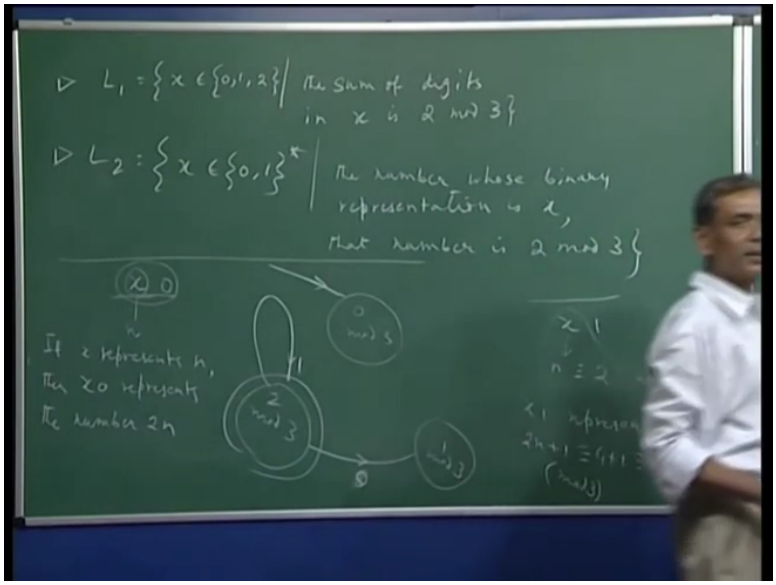
(Refer Slide Time: 52:33)



Let me say it in this way, what we are trying? We have input is this, some number, some bit string. Supposing this is the bit string, what we are trying to do is, that suppose I have seen up to this point, the DFA has seen up to this point then it should be after scanning up to this point the machine, the DFA should be in a state which captures the information as to, what is that information it should capture?

This is the string it has seen so far therefore it should capture the information as to modulo 3, what is the number represented so far? So if this is a bit string that you have seen so far then what is that number? This number is going to be 1 2 plus 2, 3 then this is 8, this is 16, right? 16 plus 8, 24 plus 3, 25. So if you see this if your DFA has seen scanned up to this point then it should record that this string is what is mod 3 of the string seen so far. So this is of course 1, right?

(Refer Slide Time: 55:06)



So then it is do the problem here, for this language again I will have 3 states, the I have seen an initial prefix of the string, input string and that string that prefix spells out number and that number is 0 mod 3, if I want what I wish to do is to that such a string should take the machine to this state, if the number seen so far is 1 mod 3 then it should take the machine to this state and if the number is 2 mod 3 then I would like the machine to come here.

So that way now we were clear what the 3 States should be, so what should be the transition? So let me take one example I mean one of these states, now they can be there are 2 transitions, right? Transition on 0 and transition on 1, correct? So suppose some string has taken the machine to this state and now 0 comes, if this x represented the number a the binary string here represented the number n then the binary string x0 represents, so let me write it clearly.

If x represents n then x0 represents, what? 2n, the number 2n clear, is not it? Supposing you have some string and you put, so whatever is the number here if you put 0 in front that means you are taking multiplying the number by 2 that we know from our understanding of binary representations of number. So what I know was, I would like to know which is the state the machine should be after scanning 0 from here?

If the machine was already here, the machine is in this state that means it has scanned a string x which is the binary representation of a number? Let us say n which is 2 mod 3, n is 2 mod 3 and now when the 0 came then the number became 2n, right? This n is 2 mod 3, so this is 2

into 2, n is you can replace n by 2 because we are going modulo 3, n is 2 mod 3, so this is 4 which is of course 1 in case of mod 3, right?

So if 0 comes I should come to this state, if 0 came and we are in this state we should be here and if 1 comes, so again argue the same way, so basically you had seen x and now 1 came and now suppose that x represented the number n which took the machine to this state and therefore since it is here this n is 2 mod 3. So this is about x, what is the number represented by the string x1?

x1 represents if you think just recall your understanding of binary representations or number. So if x represents n, x followed by 1 represents 2n plus 1, right? n was 2 mod 3, so this is 4 plus 1 mod 3 which is 5 mod 3 which is of course 2 mod 3, right? 5 mod 3 is 2, this is we are going modulo 3 that is why I am writing this equivalence rather than equality and so if 1 came I should remain here, this way you can fill up the other 2 states which we leave it as an exercise.

Again as before the initial state was this we can complete the rest of the DFA transition table in arguing in a manner like this I have done it for this state we can do the rest of the states ourselves. Now one point I want to make is this that you see that it is not the case is that DFA can do modular arithmetic, in fact it cannot even you know, it cannot divide anything's like that and yet we manage to, you remember that this will be the accepting state once we complete this diagram fully then this will accept we have we can design a DFA in this manner which will accept this language L2.

How did we manage to do that? So one way people say that look the machine itself whatever it is doing is very simple or a state it sees a symbol it just goes there or goes to some other thing, other state or the other, there are only 3 states and yet it manages to do something which looks like that it is required to do some arithmetic but the knowledge about modular arithmetic you are putting it yourselves as a designer in through your these transition diagrams.

Remember that for the machine it is very simple, if it is in this state if 1 comes it should come back here, if 0 comes it should go there. In order to figure what should be as a designer you need to know all this modular arithmetic, some of the simple facts about modular arithmetic and thereby you manage to put the 2 transitions for this state and similarly you can do for the rest of the states, other 2 states.

So sometimes we put it this way, the knowledge of modular arithmetic required for this problem has been hardware into this DFA. So while designing DFA when we, the lesson that I am trying to impart is that when we see such a thing we just do not go and say oh! Can a, can a DFA do modular arithmetic only then such a language can be accepted by DFA, no we do not do that.

DFA can only do very simple things like basically it is in a state and on a symbol it will make a transition but it is for us to figure out what the transition should be? In order to capture the essence of the modular arithmetic as we have done here