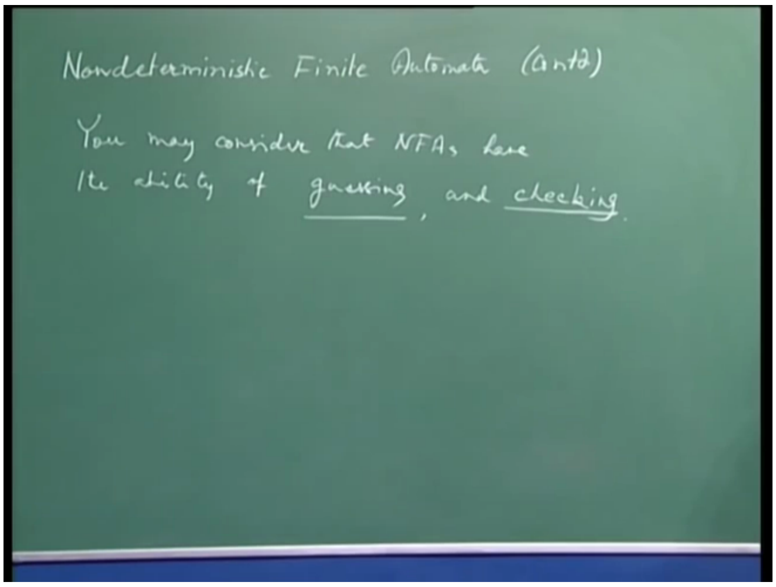
Theory of Computation

Prof. Somenath Biswas
Department of Computer Science and Engineering
Indian Institute of Technology Kanpur

Lecture 9

‘Guess and verify’ paradigms for nondeterministic.

(Refer Slide Time: 0:21)

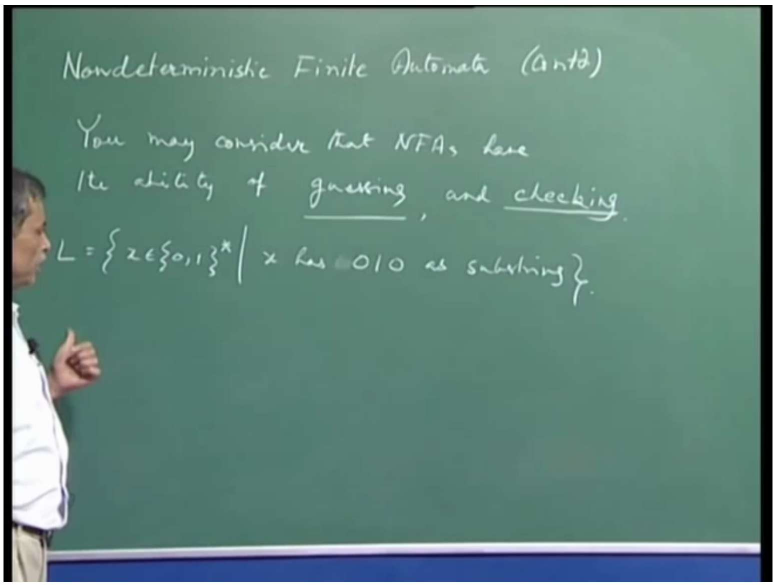


Nondeterministic Finite Automata (contd)

You may consider that NFAs have the ability of guessing, and checking.

We will continue with our discussion on nondeterministic finite automata. There is one particular way of thinking of NFA’s nondeterministic finite automata which is very useful that is this that you may consider that NFA’s have the ability of guessing and then checking. So the work of an NFA the way or what it is doing that is the language it is accepting it is useful to think in terms of this guess and check methodology.

(Refer Slide Time: 1:50)



Nondeterministic Finite Automata (Contd)

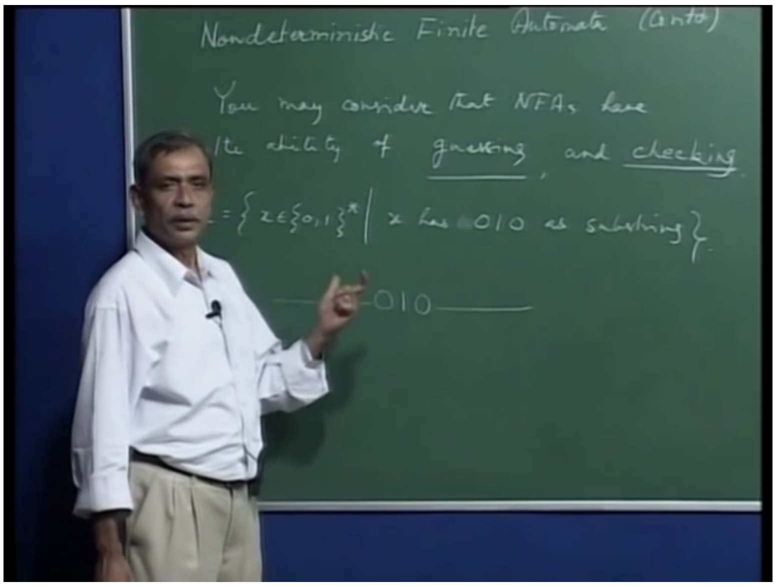
You may consider that NFAs have the ability of guessing, and checking.

$L = \{ x \in \{0,1\}^* \mid x \text{ has } 010 \text{ as substring} \}$.

Let me first take very simple problem and then through that problem I will try to explain this guess and check paradigm. Consider this language L this is the set of all strings over 0 1 such that x has 0 1 0 as substring. You have seen this a similar problem before and you know how to do it in DFA but let us see and convince ourselves that how this view of guess and check gives us a very simple NFA for this language.

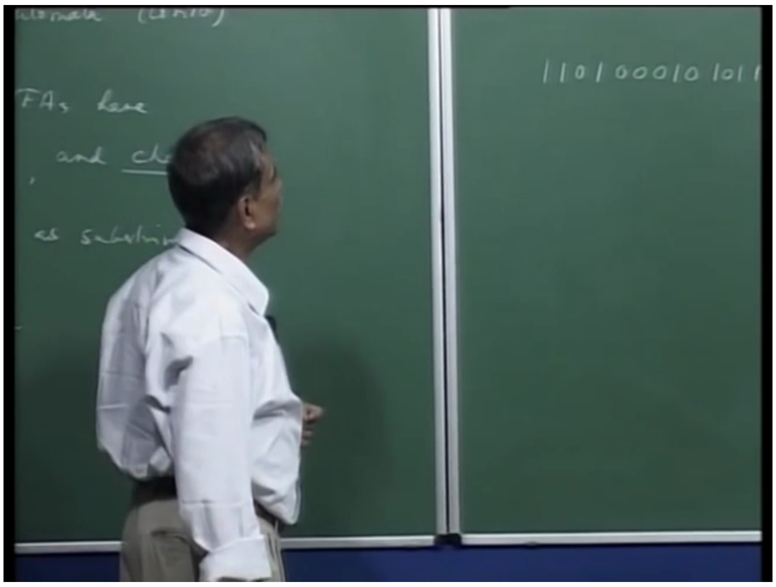
Now when you are thinking in terms of this guess and check first of all we consider only the string is in the language, we are not bothered with the strings which are not in the language, right? So what is the property of all strings which are in the language?

(Refer Slide Time: 2:57)


Nondeterministic Finite Automata (Contd)
You may consider that NFAs have
the ability of guessing, and checking
L = { x ∈ {0,1}* | x has 010 as substring }.
010

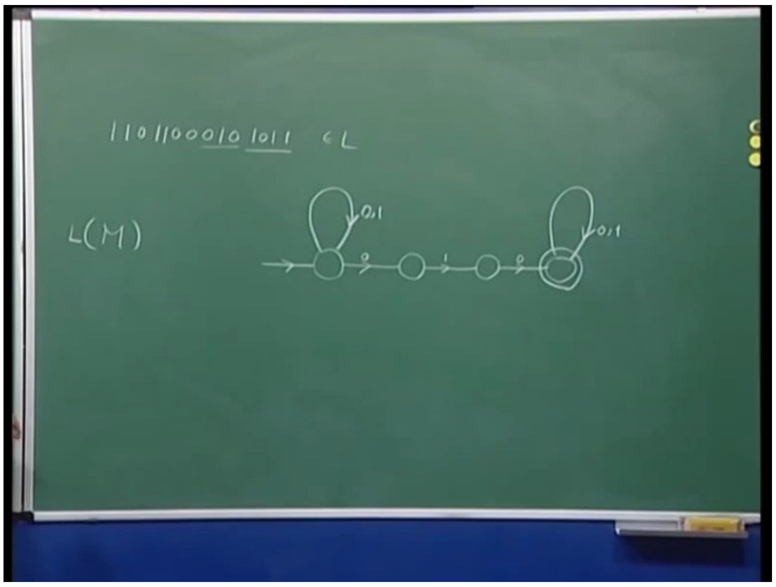
So such a string maybe anything but there will be somewhere this substring will occur and then there may be another part non-empty part following this 0 1 0. So imagine an automaton which is looking at the input bit by bit then it guesses when the 0, the 0 of the interesting substring 0 1 0 comes in the input.

(Refer Slide Time: 3:45)



But let us take a particular example. Let us say we had 1 1 0 then we had 0 0 then we had 0 1 0 and 101 1 this is the set.

(Refer Slide Time: 4:11)



To make it a little more interesting the same problem let me make the occurrence of 0 1 0 as, the first occurrence of 0 1 0 here. So as you can see in this string 0 1 0 is occurring, so therefore this particular string is in the language. Now imagine an automaton M and this is the input which is given to this automaton, it sees this first one then it sees the second one then it sees 0.

Now as I said it has the ability to guess of the 0's which is the 0 which is of interest? And that you can see it is in the 0. So in a way what this automaton does? It will wait it will keep on skipping these symbols one after another guess which is the 0 of the substring 0 1 0 in the language in the string, right? So that is this 0 then it needs to verify that starting from here there is indeed a substring 0 1 0 starts.

So let us that is the checking part, so how does an automaton check that 0 1 0 this just this substring 0 1 0? So it requires these states, this is in the state it is in some state and then it sees this 0 it comes to this sees this 1, sees this 0, so the point I am trying to make is this that it is like the DFA, right? We see in this state it remembers that I have seen the first 0, here it remembers I have seen 0 1 and then here it says 0 1 0 but the entire string is of course more than this.

Now imagine how automaton looks something like this, that there is this transition because of which the machine can remain in this state either on input 0 or input 1 and non-determinism is coming here, first of all that you see on 0 from the state either you can come to this state or one can go to this state, this guess and check paradigm of understanding non-determinism is that the machine guesses which is the 0 at which the substring 0 1 0 in the string is starting.

So let us say it makes this case here and now the automaton has the requirement of very fine that it is indeed the 0 of 0 1 0. So then it 0 then 1 then 0 it is in this state and now after that once it has verified that 0 1 0 is there then of course in this part of the string comes or you know in this particular concrete example 101 1 come this substring comes. So here it should remain irrespective of whatever is the input bit and if the machine can come by here then we should say the string is accepted, right?

Now in particular notice that after means that after 0 if another 0 came then the verification fails and in case of NFA what does it mean? This is your and now 0 came what would happen? Then of course the no transition is defined, so that particular computation path is aborted, so but before proceeding let me again say in this very simple automaton example of an NFA, how that guess and check paradigms, how the guess and check methodology is being brought into play?

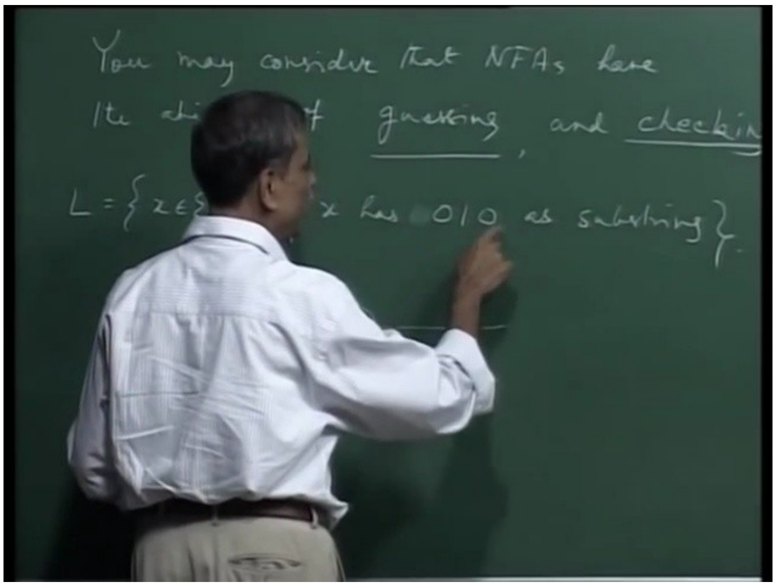
Whatever be the input, the machine is trying to guess in the initial state where is the first 0 of 0 1 0. Now it guesses and this is what you have to always assume that the machine will guess correctly, what happens if it does not guess correctly? I will come to that a little later but let us say it guesses correctly but that is not enough it should now verify that it is indeed the first 0 of 0 1 0, so that it does by making these transitions and at this point machine has verified it had guessed and that this is the right 0 and then when it comes here it has checked that it guesses correct.

And once it is convinced that it's that is the machine's guess check is correct, the machine is convinced that is guess was correct then it has no hesitation in accepting this string, okay. So this is it, this is L, this is the language accepted by this machine which accepts this particular language L, okay. Now sometimes this question ask what happens if the machine guesses wrongly?

Now suppose the machine guessed that this is the 0 of 0 1 then what would happen, of course it will, so on 1 it remains here on the second one it remains here, now it guesses that this is the 0 and now it will try to verify the next 3 bits are 0 1 0 it sees 0, fine so it comes here it sees one it will come here, now instead of a 0 which it is expecting it sees 1, this 1 then computation aborts.

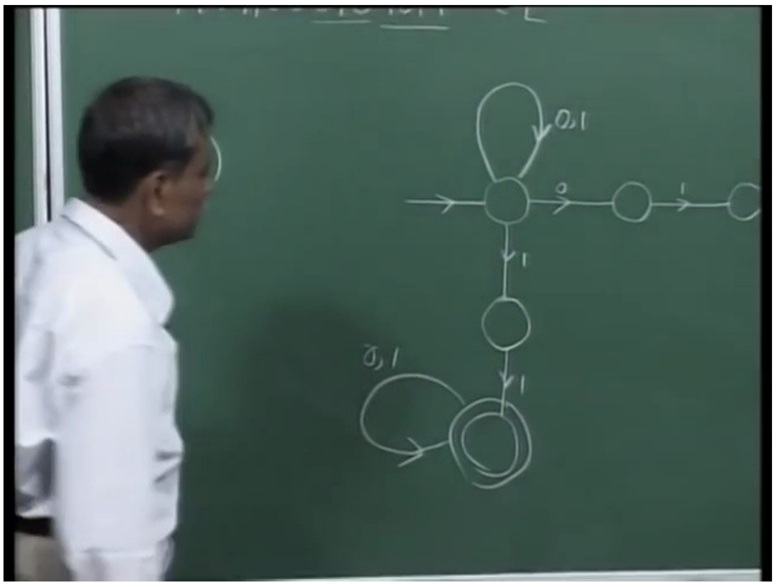
So are wrong guess on the part of NFA will mean a computation path either which is aborted or which leads to non-acceptance, correct guess and if the string is in the language would mean that the input will be accepted and the machine will be in the accepting state or final state. So in the same way we can think of the other language, so in fact in fact now you see it is so simple to say that is the example we had given earlier.

(Refer Slide Time: 11:55)

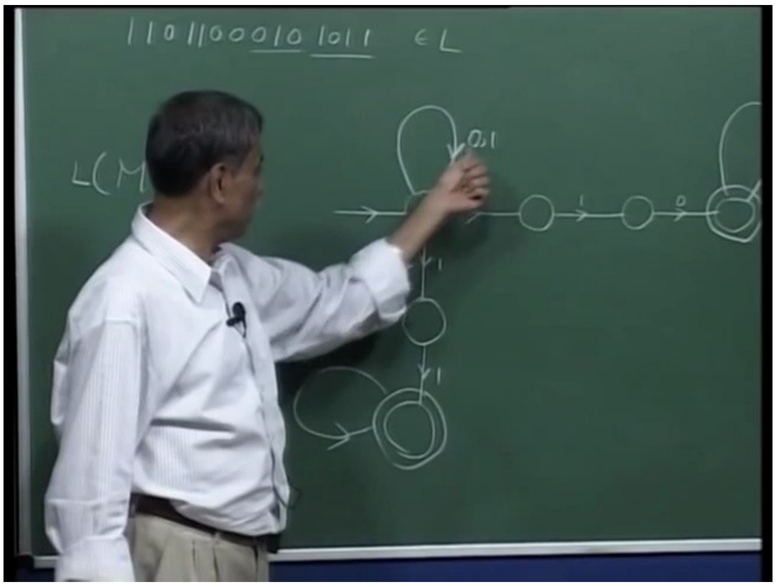

You may consider that NFAs have
the ability of guessing, and checkin
L = { x ∈ Σ x has 010 as substring }.

Similar example that the language x has 0 1 0 as substring or 1 1 as a substring, so I can just say or 1 1 as a substring then there are 2 things the machine guesses given a string in the language, firstly is the string in the language because 0 1 0 there or is it because the substring 1 1 is there in the language, right?

(Refer Slide Time: 12:41)



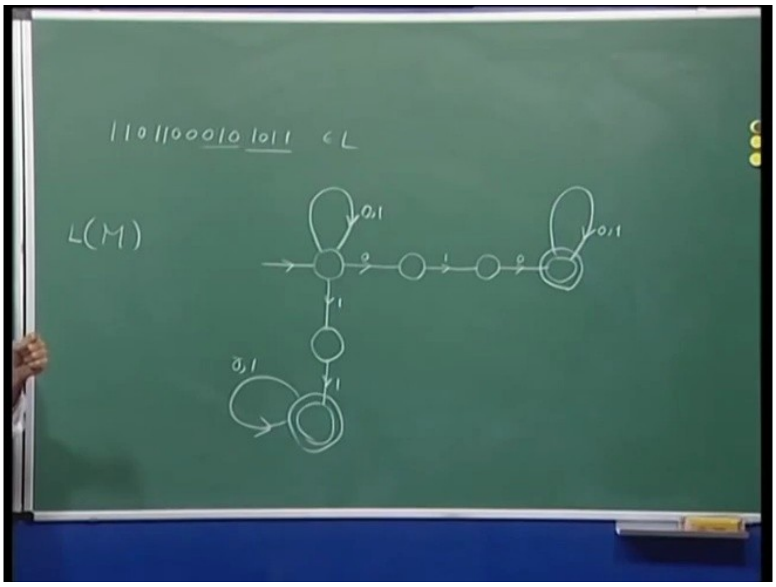
(Refer Slide Time: 12:52)



So you remember in that case the NFA would look like this. So now you see on both 0 and 1 from this state it can make it has 2 transitions defined on both either 0 or 1. Now, so let us say I have again this string, this is of course in the language for both because 0 1 0 is there 1 1 is there but let us say the machine guesses that first I will accept the string because 1 1 is there and then it skips the1 it remains here, it remains here at 0.

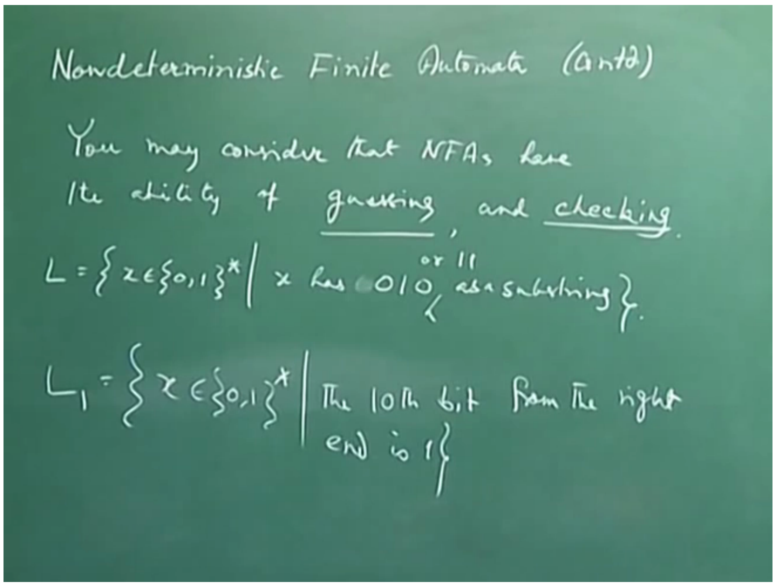
Now it has already guessed and now it verifies that 1 1 is here, so on the first one of 1 1 it will come here, second one it comes here, so therefore on these 2 ones convinces the machine that the substring 1 1 indeed occurs in the language and therefore whatever comes later it will remain in the state, right?

(Refer Slide Time: 13:13)



So this is the same thing, so it is so easy you see that in thinking in terms of guess and check this method to repeat myself again that NFA as if to accept a string it is guessing where the property which that makes the string in the language where in the that property is to be found in the string and then it is verifying it is checking that indeed that property is being obeyed by the string and after that it remains, it is in the final state.

(Refer Slide Time: 14:43)



Nondeterministic Finite Automata (Contd)

You may consider that NFAs have the ability of <u>guessing</u>, and <u>checking</u>.

$L = \{ x \in \{0,1\}^* \mid x \text{ has } 010 \text{ or } 11 \text{ as a substring} \}$.

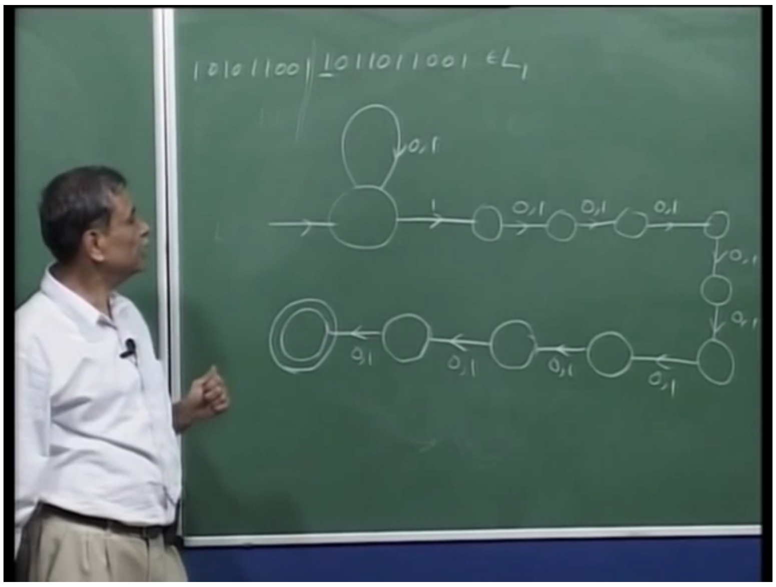
$L_1 = \{ x \in \{0,1\}^* \mid \text{The 10th bit from the right end is } 1 \}$

The other example of NFA that we had seen, so let us say something similar, so we had seen another language like this let me call it L1 is x in 0 1 star, right? Now let me say the, why not? Let us say the 10th bit from the right end is 1.

And then suppose an input comes something like this we will we will make the 10th bit 1 here, so I will 1st write the 10th bit 1011011, right? And this suppose this is the end and then I have something, supposing this is the input and this should be there in the language L1, clearly because 1, 2, 3, 4, 5, 6, 7, 8, 9, 10th bit from the right end is 1.

Now suppose I want to design an NFA to accept this language, you have already seen what the NFA should be but I am thinking now in terms of this guess and check. So what should be the guess? The gas should be that this is the 10th bit and verification should be consisting of 2 things that this 10th bit is 1 and after that there are exactly 9 bits no more no less.

(Refer Slide Time: 16:43)



So what, let us say the machine is initially in a state and on 0 and 1 it remains here and then when the 10th bit comes which is from the right end the 10th bit, 10th bit from the right end, that time now it should verify that this is indeed the 10th bit, so one, 2, 3, 4, 5, 6, 7, 8 bit will lead me here, 9th bit here and 10th bit. So you see so let me complete this. Do you care what is the 9th bit from the right end is?

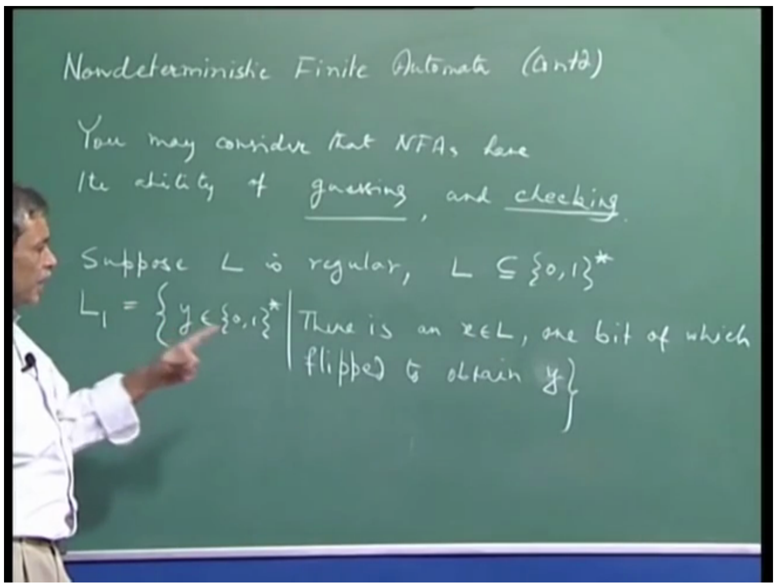
You do not, so for verification should mean that 10th bit from the right end is 1 and after that there are exactly 9 more bits. So this either on 0 or 1 it will go here, here, either on 0 or 1, 0 or 1, 0 or 1, 0 or 1, so 1, 2, 3, 4, 5, 6, 7, 8 or 1, 0 or 1 and 10th bit comes here takes me here and now how do I verify this is indeed there are no more bits after this is by having no transition?

So if by this you started from here and then let us say there are 9, exactly 9 other bits 1 comes here and then if there are more bits then what happens? Then the NFA computation is not defined therefore the next state is not defined, that string or the guess where the 10th bit is not 1 but let us say 12th bit from the right end is 1 then there will be 2 extra bits, so at you are here and you will find one more input bit has come and then the NFA's computation from here by defining no transition out of this state I am ensuring that if this is not the 10th bit from the right end. So the string is not going to lead me lead the NFA to the accepting state this final state.

So point I am trying to make is there are 2 aspects guess and check. You see there are many 1's coming before this but the 10th bit from the right end was this, right? 1, 2, 3, 4, 5, 6, 8, 9, 10, 1 came the machine guessed correctly let us say but this is not the one which is the 10th bit from the right, this 1 is also not, this 1 is also not, when this one, so all up to this point the machine remained in this initial state and when the 10th bit I mean 10th bit from the right end came it guessed correctly that is how you should imagine that it guesses correctly and then now it is verifying it is indeed the 10th bit from here, okay.

So these are 2 examples that we had seen and I have tried to explain by means of this guess and check methodology and this is actually how you, if you think that way the design of an NFA becomes quite simple in many cases. Let us take new example and that example will be as following.

(Refer Slide Time: 21:38)



Nondeterministic Finite Automata (contd)

You may consider that NFAs have the ability of <u>guessing</u>, and <u>checking</u>.

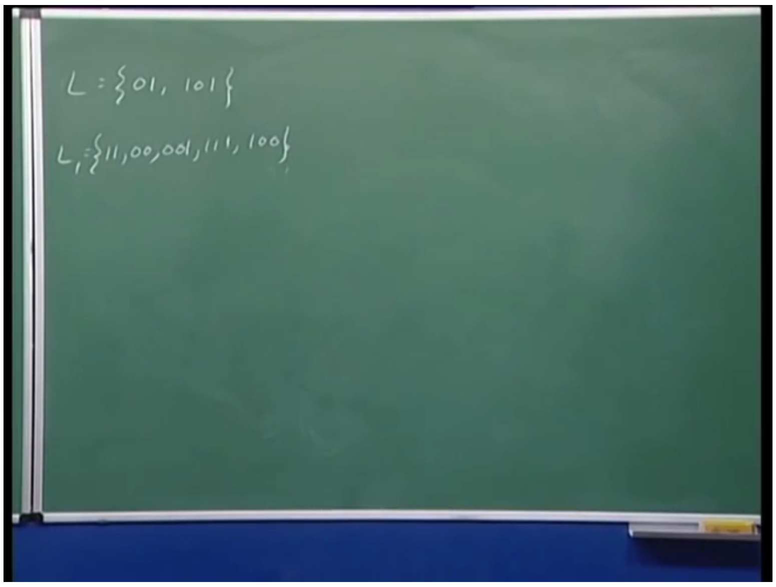
Suppose L is regular, $L \subseteq \{0,1\}^*$

$L_1 = \{ y \in \{0,1\}^* \mid$ There is an $x \in L$, one bit of which flipped to obtain $y \}$

Our next example for NFA where I would like to again explain exemplify this guess and check methodology is this suppose L is regular and now I am defining another language using this L as follows we are saying that L1 and without loss of generality, let me say an L is a subset of 01 star this L consists of binary strings and L1 is all y in 01 star, so again of course binary strings such that there is an x in L, you see that is how I am using this L to define this new language L1.

There is an x in L from which or let us say which is flipped or one bit of which is flipped. There is an x in L, one bit of which is flipped to obtain this y, okay. Alright let us let us see it clearly. So you are saying L1 is that language where this language consist of binary strings and if you take string in L1 you will find an x in L and such that if you flip one particular bit of this x, that bit maybe anywhere in x and exactly one bit.

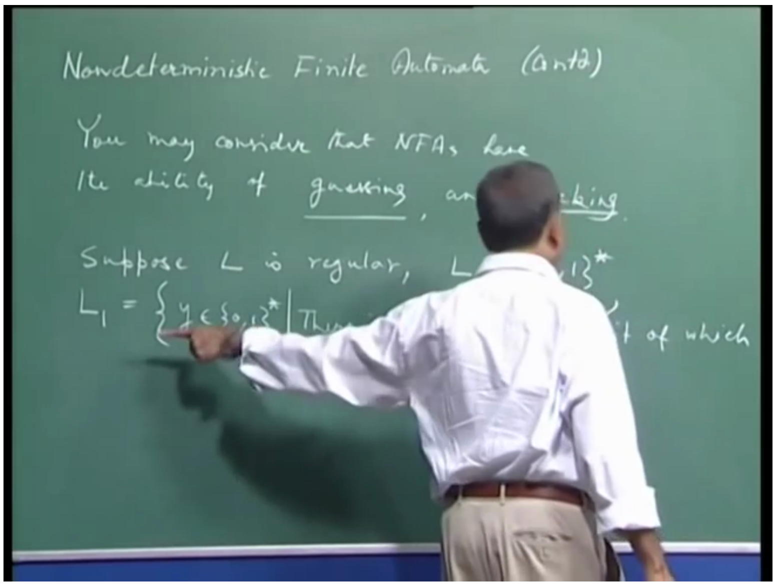
(Refer Slide Time: 24:29)



$L = \{01, 101\}$

$L_1 = \{11, 00, 001, 111, 100\}$

(Refer Slide Time: 24:51)



Nondeterministic Finite Automata (contd)

You may consider that NFAs have the ability of guessing, an[illegible]king.

Suppose L is regular, L [illegible] $1\}^*$

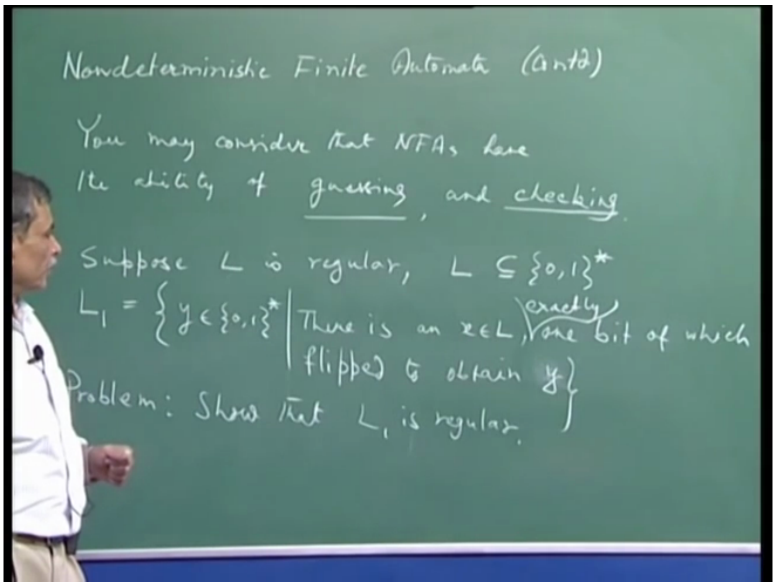
$L_1 = \{ y \in \{0,1\}^* \mid$ Th [illegible] t of which

So let me emphasize this one by saying that this is exactly one bit of which is flipped to obtain this particular string y.

So let me make this clear supposing my language was like this 01, 101, so what is L1? Now you see take this string 01 you flip this bit you will get the string 11. Now this 11 should be in L1 because I have obtained 11 by flipping this bit of 0. So similarly I could have flipped this bit also, so then I would have gotten the string 00, right? So from 01 by flipping one bit I would get these 2 strings, 101 if I flip the 1st bit I will get 001, if I flip the 2nd bit I will get 111, if I flip the last bit I will get 100, so therefore my language L1 is 11, 00, 001, 111 and 100, right?

Now if I ask you that tell me why this bit is in this language L1? You will say oh! You see I have asked you why this particular string is language L1. You will say oh! You see that is because you flip this bit, the 2nd bit you will get 101 and that is in the language L, right?

(Refer Slide Time: 26:27)



Nondeterministic Finite Automata (Contd)

You may consider that NFAs have the ability of <u>guessing</u>, and <u>checking</u>.

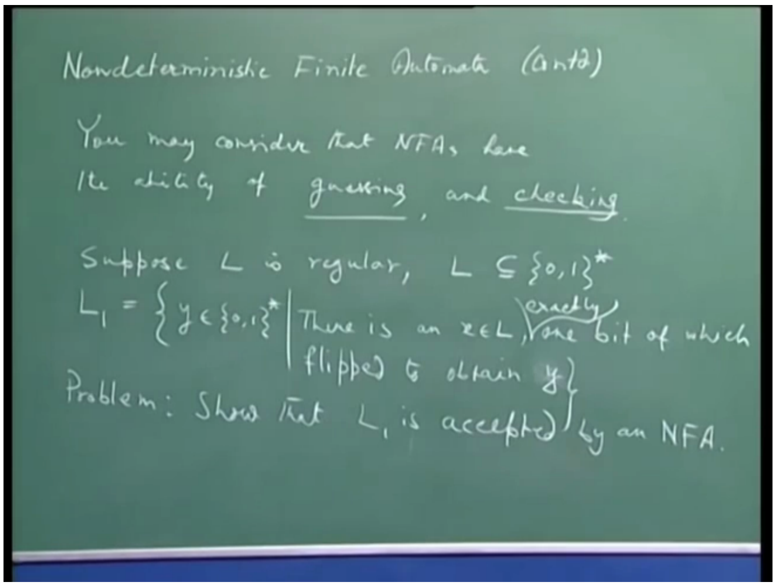
Suppose L is regular, $L \subseteq \{0,1\}^*$

$L_1 = \{ y \in \{0,1\}^* \mid$ There is an $x \in L$, $\overset{\text{exactly}}{\text{one}}$ bit of which flipped to obtain $y \}$

Problem: Show that L_1 is regular.

It is clear what the language L1 is and the problem is to show that L1 is regular. Now before we try to design, so I was saying basically what we are trying to do? That we will design and NFA for L1 but I have not yet shown you that this will make L1 to be regular but let me just show that L1 is accepted by an NFA.

(Refer Slide Time: 27:22)



Nondeterministic Finite Automata (contd)

You may consider that NFAs have the ability of <u>guessing</u>, and <u>checking</u>.

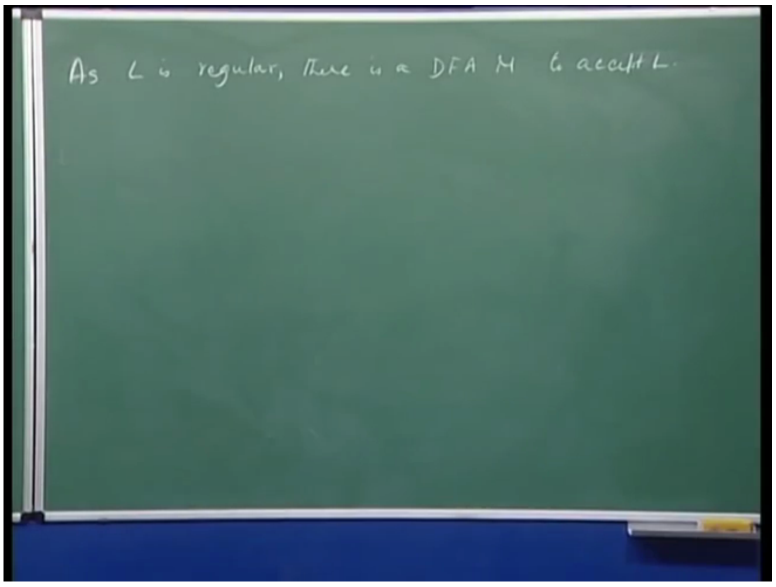
Suppose L is regular, $L \subseteq \{0,1\}^*$

$L_1 = \{ y \in \{0,1\}^* \mid$ There is an $x \in L$, exactly one bit of which flipped to obtain $y \}$

Problem: Show that L_1 is accepted by an NFA.

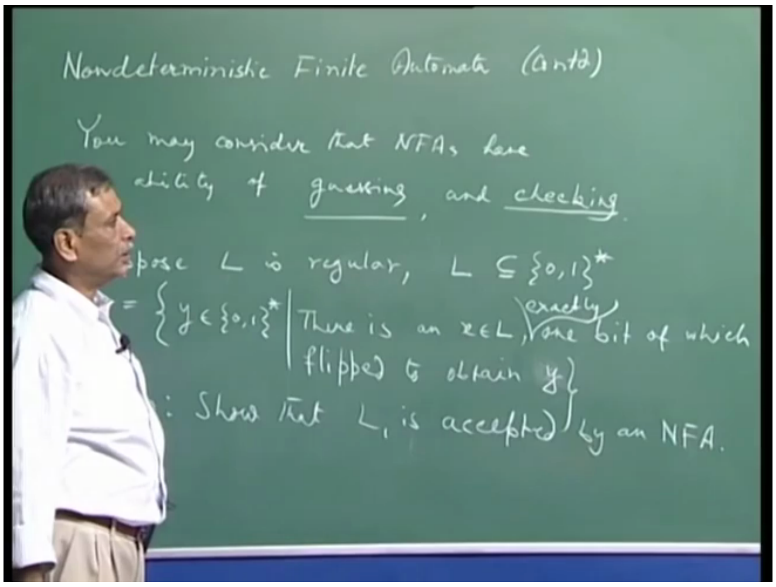
At this point this should be the more appropriate question and later on we will see the set of languages accepted by NFA's. Precisely the set of languages accepted by DFA, so once we know that I could have invoked that question to claim once I show that L1 is accepted by an NFA that L1 is also regular but at this point it is specific to ourselves only prove that there is an NFA to accept L1, okay.

(Refer Slide Time: 28:06)


As L is regular, there is a DFA M to accept L.

So let me now remove this and once when you are confronted by this problem, of course one thing you know, one thing is that since L is regular there is a DFA M, so I can write like this as L is regular, there is a DFA M to accept L, of course that comes by definition of regularity and now I want to design an NFA. So my NFA let us say, so let me first of all say this M has the states Q and then of course its alphabet is 01 binary alphabet we are using and its transition its function is defined by Delta q0 and F, these are some.

(Refer Slide Time: 29:31)



Nondeterministic Finite Automata (contd)

You may consider that NFAs have ability of guessing, and checking.

pose L is regular, $L \subseteq \{0,1\}^*$

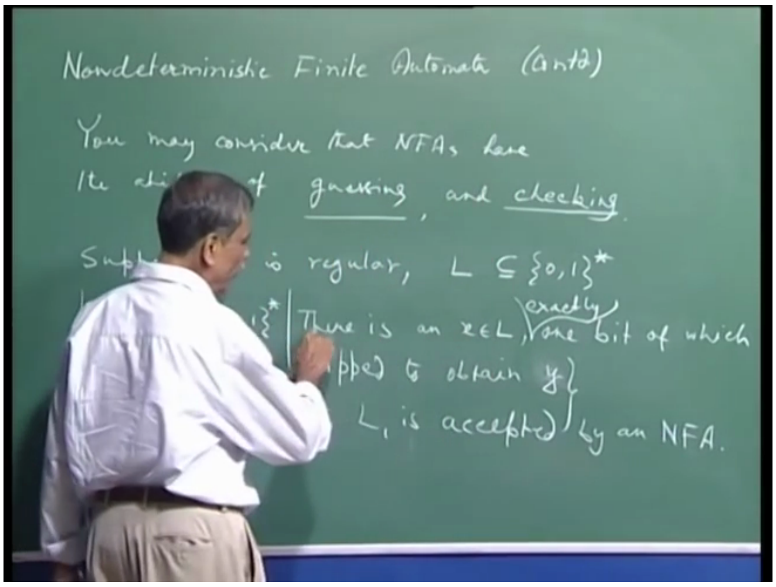
$= \{ y \in \{0,1\}^* \mid$ There is an $x \in L$, exactly one bit of which flipped to obtain $y \}$

: Show that L_1 is accepted by an NFA.

Now I would like to define M1 an NFA such that language accepted by M1 is precisely the language L1 which we have defined here, right? So let us think in terms of this guess and check business. Remember for designing that NFA all I am interested in, is that if you give me any string of the language my NFA will be able to accept it and we will think of the work of the NFA in terms of that guess and check, alright.

So you have given me a y and this y happens to be in L1, right? This is a bit string, now what should the NFA do? Guess and check, so NFA has to guess which bit is the bit which was left from string of L in order to get this particular string y. So let us say, let us just think conceptually first that this is the bit that was flipped, okay. So bit that is (()) (30:45) that is here but that is not enough it is not saying I have guessed that therefore this.

(Refer Slide Time: 31:02)



Nondeterministic Finite Automata (contd)

You may consider that NFAs have the abi[illegible] of guessing, and checking.

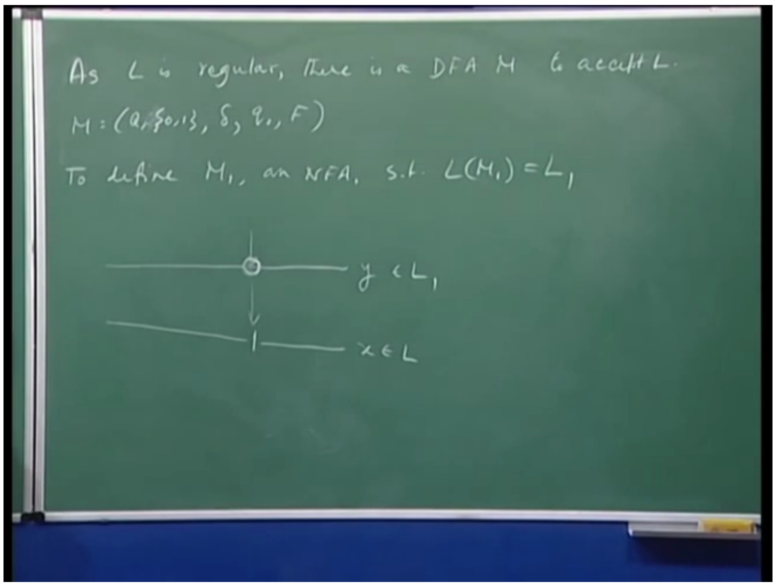
Supp[illegible] is regular, $L \subseteq \{0,1\}^*$

[illegible] $1\}^* \mid$ There is an $x \in L$, exactly one bit of which [illegible]pped to obtain $y\}$

L_1 is accepted by an NFA.

But you need to verify that this fact, what is you need to verify? There is an x in L whose one bit flipped exactly one bit flipped gives me the string that you have given me as input. Suppose you, just think in this way that suppose there was 0 here and by flipping I will get 1 here, so everything else remains same, what can you say about the string, new string?

(Refer Slide Time: 33:14)



As L is regular, there is a DFA M to accept L.

$M = (Q, \{0,1\}, \delta, q_0, F)$

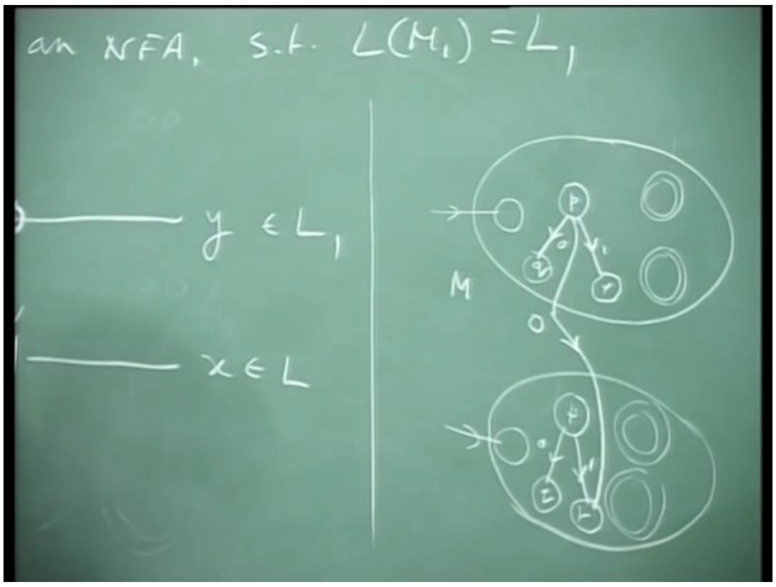
To define M_1, an NFA, s.t. $L(M_1) = L_1$

So this bit was 0 here you have flipped this 1 rest of the bits are exactly identical and now if this string is in the language L1 then clearly this string should be in the, this is the x which is in L. How do you verify that this string is in language L? The DFA obviously should accept, right? So now we are getting a glimmer of what is to be done. So imagine a machine NFA which on successive bits behaves as if it is the DFA M.

On one exactly one particular input bit it flips the bit, so it was 0 here and it assumed as if the bit is 1 and then sees what the DFA would have done on this particular bit that is the state now the machine goes to, machine in the sense the DFA goes to, this M goes to, our NFA is keeping track of that and then rest of the bits it behaves exactly like the machine M the DFA, okay.

So let me repeat, to verify that y is in L1 what our NFA will do? In the beginning it will keep track of the states, the machine, the DFA is in on successive bits than on 1 particular bit of it is 0, it will make the transition as if or it will see which is the state the DFA would go to had this bit been instead of 01 and then from that state onwards it just keeps track of what the DFA would do till the rest of the input and then at this point the NFA check will be over if the DFA is found to be in an accepting state.

(Refer Slide Time: 34:16)



Let me describe this pictorial, imagine this is a DFA, this is the DFA M and this is the initial state and there may be a number of final states, let us just say for the sake of this example illustration rather that there are 2 final states. Imagine we are thinking of a machine NFA which is consisting of 2 copies of this. So this is, so 2 copies of course look like this and then imagine this is one of the states, okay.

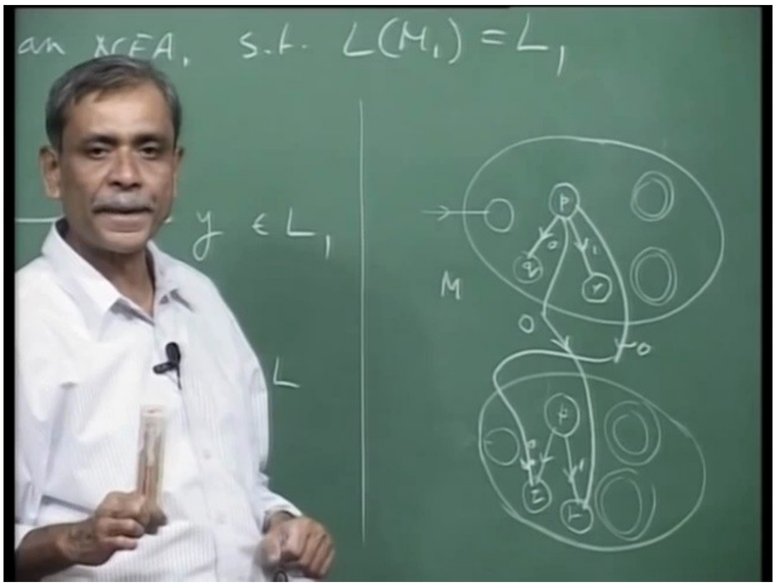
There are many other states but let me focus on one particular state and for the sake of illustration let us say on this is a DFA remember, on 0 it goes here and on 1 it comes here, of course in the copy also, so let me even give names for this P, q, r. So as I was saying that this is one real estate and on 0 this DFA M would go to q, on 1 it goes to r, on the copy also of course the similar same thing will happen, identical thing the state P, this state q, this state r, right? 0 and then 1.

Now imagine my NFA is making use of these 2 copies, so initially remember up to this point it is behaving exactly like the DFA. So imagine at that time this is making use of this copy and suppose it sees now 0 and the machine has guessed, the NFA has guessed that this is the 0 which was flipped, right?

To from a string which was in L, so actually a DFA would have seen 1, so therefore the NFA on 0 also it can of course go here, it also on 0 can go to this state r, the state r, okay. Now just look this, what I am trying to say is this, that normally when the DFA is working of course from the state P on 0 you go to q, on 0 if you go to r that means what?

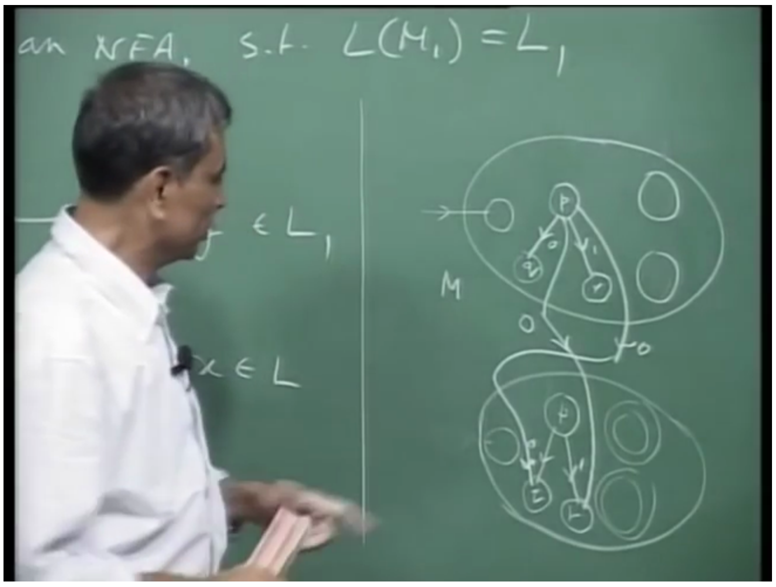
Another way of looking at it is that as if the bit 0 was flipped and because of that when we flip 0, you would see 1 and the DFA therefore should go to the state r, okay. So similarly from this state P on 1, so normally on 1 it would go to r but if that particular bit 1 is the bit which was flipped then on 1 it should go to as if it had seen 0, the DFA would have seen 0 and it would have gone to this q instead, okay.

(Refer Slide Time: 38:27)



Now then after that the machine remains here, okay. So now of course this is, so let me complete at least the picture wise pictorially what the NFA looks will be, this is the initial state. Initial state is the initial state of the first copy, from every string of the first copy these new transitions are defined, so either on 0 that it can make a transition to its copy that will be the normal DFA working or what can be done is to imagine that, that 0 is flipped and then what happens? Then it would go to the copy here which the DFA would have gone from here on seeing 1, so that is why you are coming.

(Refer Slide Time: 39:31)

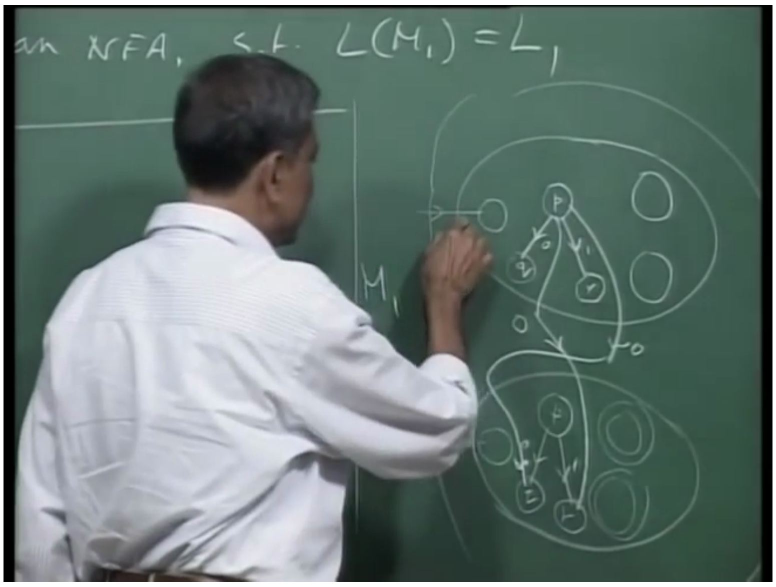


And then these are no longer, these are which were the final states, here are not final states of the NFA but these 2 the final states of the DFA in the second copy of the final states of the NFA.

Now through this picture let me explain that guess and check the way it is happening, the NFA starts its operation on this copy of DFA. By the way why it is an NFA? Because you see on every state, on every symbol in this copy in this part of the state space of NFA, by the way the NFA states will be the set of these states plus the set of these states. So either on 0 it can come here or on 0 it can come here and so therefore it is an NFA and not a DFA because on 0 it has 2 transitions from here and similarly from 1, 2 transitions from here and that is the case for every state, I have just shown you the case with one particular state but that is precisely what you will do for every, write it more formally over here later on.

So let us understand the guess and check idea from here, the machine NFA is working on the string it sees successive input, it remains in this copy till it finds the bit which it guesses had it been flipped? Then along with the flipped bit the string would have been accepted by DFA. So it considers the effect of that flip comes to this copy and then again behaves like the DFA.

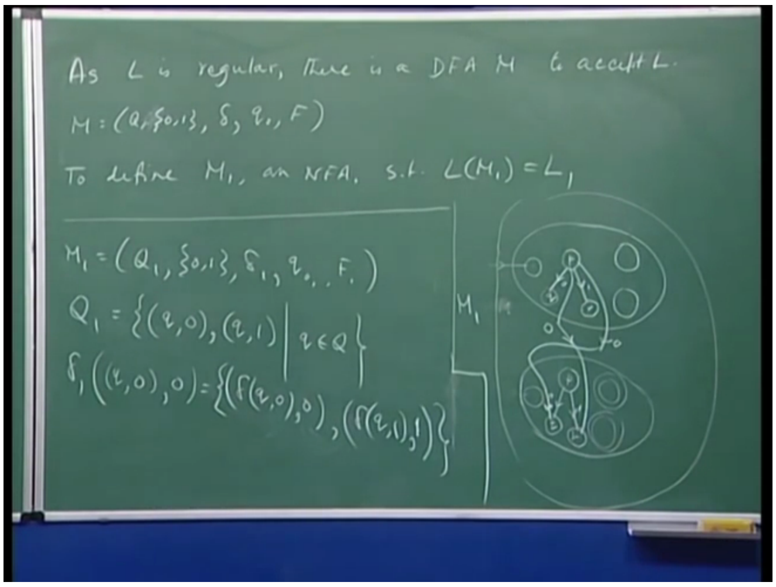
(Refer Slide Time: 41:58)



So it is doing 2 things that you see to go from here to one of the final states on an input, what is going to happen? You may be here somewhere and then at some point you have to come to this path because the accepting states final states of the entire machine is, so this M1 is this entire thing, right? The initial state is this and the final states accepting states are these 2. So you can see we are very fine that exactly one flip of the input, whether or not to take the DFA from the initial state to one of the final states, it's one of its final states.

How I am making sure that exactly one flip? You see you can the NFA has the ability consider flipping any bit but after flipping, the effect of flipping it what happens in this copy? And then no more flips can be entertained. So exactly 1 flip is required to go from here to here, alright.

(Refer Slide Time: 43:27)



As L is regular, there is a DFA M to accept L.

$M = (Q, \{0,1\}, \delta, q_0, F)$

To define M_1, an NFA, s.t. $L(M_1) = L_1$

$M_1 = (Q_1, \{0,1\}, \delta_1, q_{01}, F_1)$

$Q_1 = \{(q,0), (q,1) \mid q \in Q\}$

$\delta_1((q,0),0) = \{(\delta(q,0),0), (\delta(q,1),1)\}$

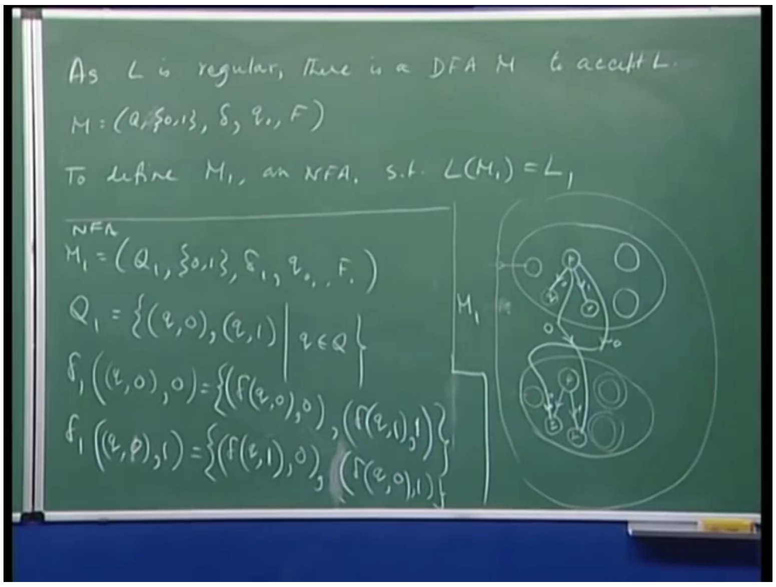
So let me more formally define this NFA M1. So as I said that M is this, so what should be the states of M1? So let me say M1 is Q1, 01, Delta 1, q01, F1, right? So I would think of the way I have done it in this picture that Q1 is, there are 2 things what we need to remember is whether I am in this copy of DFA or in this copy of DFA and so therefore let me say, okay.

So Q1 you can see it has exactly twice the number of states of Q because for every state Q I have 2 states q0 and q1 and as like to imagine q0 are these states the first copy and q1 are these, so how do I describe Delta 1? So Delta 1, now first let me describe suppose, remember a transition function takes 2 things an input symbol and the state of the machine, here suppose the state is q0 and I have 0.

Now the machine can remain in this copy, so that would mean essentially saying that it can remain in state Delta of q and then this 0 has come. Remember this q is the state of the DFA, 0 is the input symbol which has come, so from q the DFA would go to Delta q0 and it is still in the first copy, so that to indicate that I mean it is zero and this is one state but it can go to another state also and which is that?

It considers this 0 as 1 and that would mean Delta of q 1 and now it should be in this copy that I indicate by 1, okay. Alright, is this clear? So once more, states of machine M1 which is an NFA they keep track of two things what is the state of DFA and whether it is using the first copy of DFA or the second copy of DFA? So here it is, that is why states are q 0 means it is in the top copy, q1 means it is the q of the second copy, right?

(Refer Slide Time: 47:32)



As L is regular, there is a DFA M to accept L.

$M = (Q, \{0,1\}, \delta, q_0, F)$

To define M_1, an NFA, s.t. $L(M_1) = L_1$

NFA

$M_1 = (Q_1, \{0,1\}, \delta_1, q_{01}, F_1)$

$Q_1 = \{(q,0), (q,1) \mid q \in Q\}$

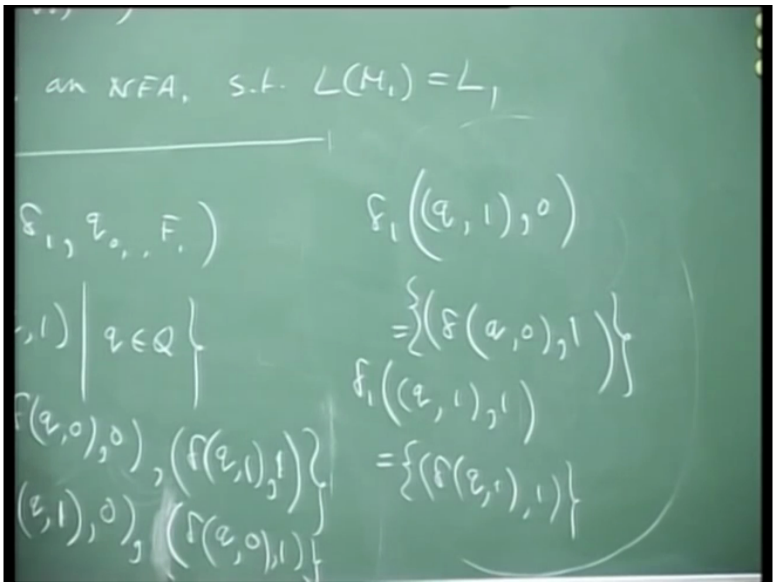
$\delta_1((q,0),0) = \{(\delta(q,0),0), (\delta(q,1),1)\}$

$\delta_1((q,0),1) = \{(\delta(q,1),0), (\delta(q,0),1)\}$

This is the set of states for Q, for M1 the NFA and so let me state very clearly and this is NFA and of course the transition function tells me that it is an NFA because from this state on the symbol there are two ways of going, right? Similarly what is Delta 1 of q1, sorry q0 on1, right? Now you can see if the NFA chooses to remain in the first copy than it is just this that this is Delta from on q it has seen 1, so it is 1.

Please do not get confused between this Delta 1 and this Delta, now you are talking of the DFA q on 1, q on 1 this is the state and it is going to this state that means it is assuming that it is still in the first copy, so therefore it is zero but it can also take this one to be the flip bit, right? In that case the state would be Delta of q, 1 has come flip it which is 0, q0, right? But now it is in the second copy, so let me write it clearly and this is one, okay, right.

(Refer Slide Time: 49:51)



an NFA, s.t. $L(M_1) = L_1$

$\delta_1, q_0, F)$

$,1) \mid q \in Q\}$

$\delta(q,0),0), (\delta(q,1),1)\}$

$(q,1),0), (\delta(q,0),1)\}$

$\delta_1((q,1),0)$

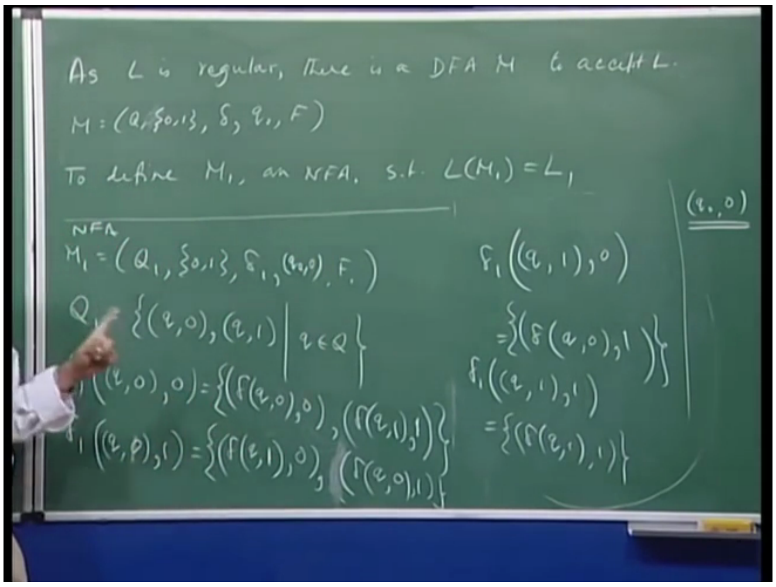
$= \{(\delta(q,0),1)\}$

$\delta_1((q,1),1)$

$= \{(\delta(q,1),1)\}$

Now the role of the picture is over since we are thinking in terms of doing it explicitly, writing out explicitly. So what can you say about Delta 1 of q1, 0? Now q1 means, we are talking of the second copy, so it should be Delta of q, 0 and it will remain in the second part, right? It is an NFA we will write like this as a subset of states, remember that this whole thing is a state of the NFA and similarly Delta 1 of q 1 1 is Delta of, from q it has seen 1, so q 1 and since it is in the second copy it will remain in the second copy.

(Refer Slide Time: 51:28)



As L is regular, there is a DFA M to accept L.

$M = (Q, \{0,1\}, \delta, q_0, F)$

To define M_1, an NFA, s.t. $L(M_1) = L_1$

NFA

$M_1 = (Q_1, \{0,1\}, \delta_1, (q_0,0), F_1)$

$Q_1 = \{(q,0), (q,1) \mid q \in Q\}$

$\delta_1((q,0),0) = \{(\delta(q,0),0), (\delta(q,1),1)\}$

$\delta_1((q,0),1) = \{(\delta(q,1),0), (\delta(q,0),1)\}$

$\delta_1((q,1),0) = \{(\delta(q,0),1)\}$

$\delta_1((q,1),1) = \{(\delta(q,1),1)\}$

$\underline{\underline{(q_0,0)}}$

And other two things for the NFA I have said Q1, what is 01, what is Delta one, q0 will be what? Now you see q0 is going to be the state it is the q0 of the DFA and the first copy, so this state will be taken as, let me write it this way that it is q0 0 this is the state and F1 which are the states? All those states which are ending with second component that is 1 and the State component of the DFA is the final state, right?

But the main point of this example was that guess and check methodology. This NFA has the ability I mean it is using its ability of guessing which is the bit which is flipped? See every time it can guess oh! This is, in that picture if you went back to that picture which I had just rubbed out that for every input bit it is seeing, it has the choice of guessing whether it is the bit which it should consider flipped and then see what the DFA would do and then it is verifying also, that one bit is flipped, exactly one bit is flipped and on this flip one bit new string will take the machine, the DFA from the initial state to one of its final states.

So this is a slightly more involved use of, for our understanding of guess and check methodology of describing NFA's and their work and this is one way and a very useful way whenever non-determinism is there and we will encounter non-determinism in other kinds of automata be it touring machines or be it pushed down Automata wherever non-determinism is there, it is often in fact most often it is very useful to think in terms of this guess and check paradigms.