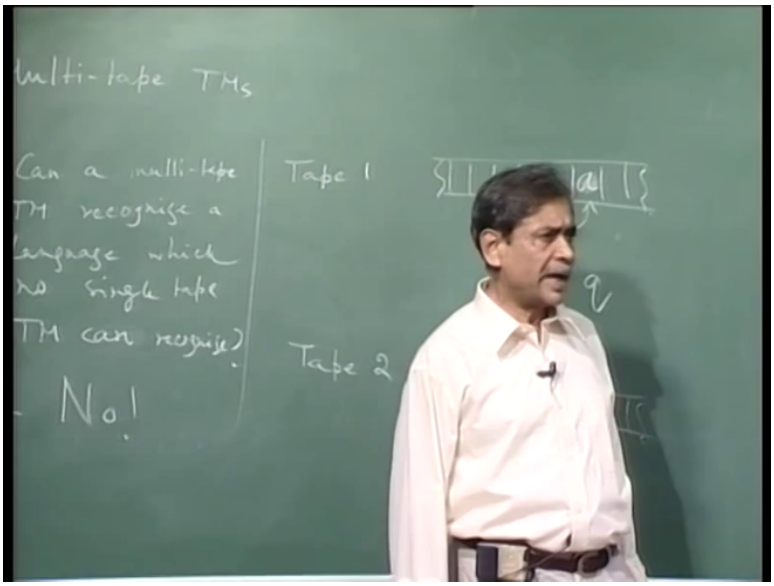
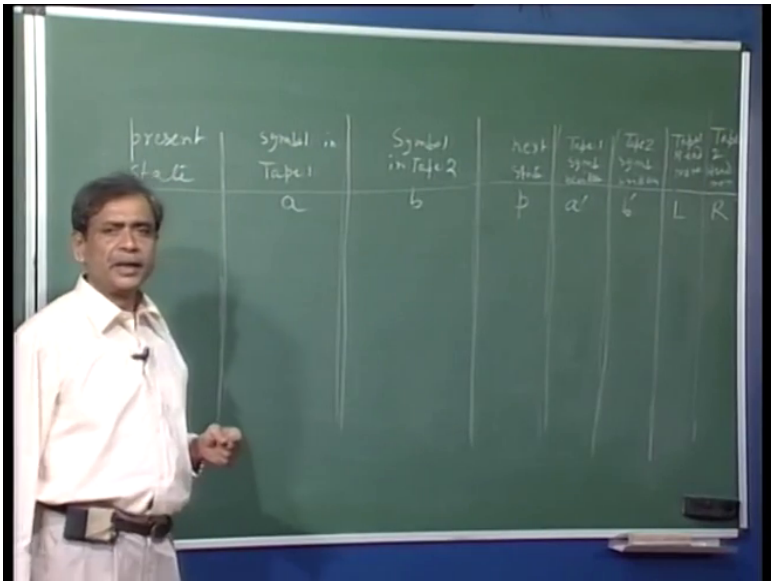
Course on Theory of Computation
By Professor Somenath Biswas
Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur
Lecture 38
Module 1
Simulation of multitape TMs by basic model. Nondeterministic TM (NDTM). Equivalence of NDTMs with deterministic TMs.

(Refer Slide Time: 0:28)



We are asking the question can a multi-tape Turing machine recognize a language which no single tape Turing machine can recognize and we would like to show the answer is no, what it means is that if there is a language which we recognize by a multi-tape Turing machine that same language can be also recognized by a single tape basic Turing machine the kind of Turing machine we first defined.

(Refer Slide Time: 1:25)



And this proof of this answer we would like to illustrate by means of an example where we have two tapes and the same idea that we use in this example can carry over to multi-tape machines with number of tapes which is more than two and in this case we had said that a move of such a machine will be described now by not a quintuple or a tuple of 5 elements but it will have 8 elements, why? Because you need to describe given a present state and the 2 symbols in the 2 tape in this example we said that the present state of the Turing machine is q and in tape 1 symbol is a, tape 2 symbol is b if such a situation occurs then the next state let for example is p and symbol that is written on tape 1 is a prime and symbol that is written on tape 2 is b prime.

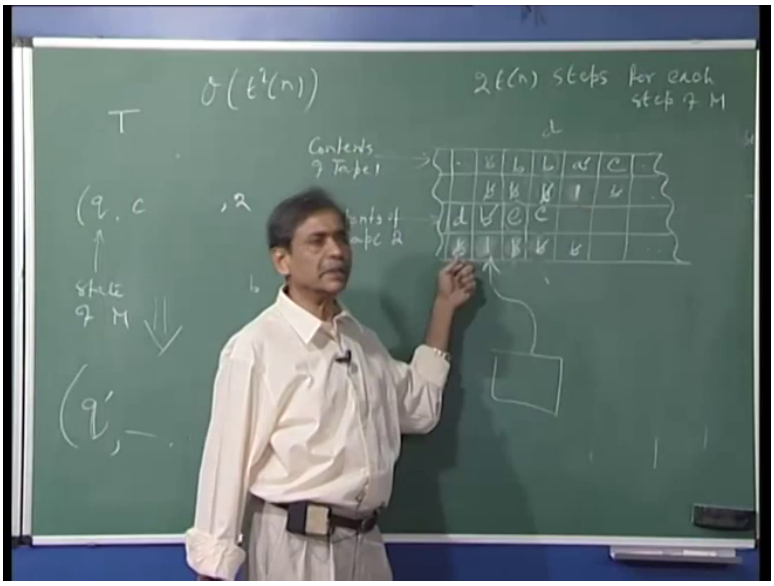
Now since there are 2 read write heads there movement need not be synchronized in the sense one can move to the left and other can move to the right or both can move to the left or both can move to the right. For this example we are saying that in this particular move when the present state is q and the symbol in tape 1 is a and the symbol is tape 2 is b. In tape 1 you the machine writes a prime in tape 2 machine writes b prime and the tape head for the first step is that movement is to the left and tape 2 head movement is to the right.

So after the next step from here is going to be that a and instead of a, a prime would be written instead of b, b prime would be written the same place and the tape head in first tape will move to the left by 1 cell and for tape 2 the head movement is to the right. So next time instant you will find the Turing machine the 2 tape Turing machine in this position, of course there are other

symbols and this kind of tuples have given then this machine will work on from one step to the next step and to the next step.

Now how can we do what this machine does by means of a single tape Turing machine. So there are several questions we need to first settle that first question being how shall we represent the contents of the two tape on a single tape, there are various ways of doing it but what is convenient and at least for this lecture the what we can do is to we will consider a tape a single tape of course which has 4 tracks, so let us be clear what we are doing.

(Refer Slide Time: 5:22)



This is a 2 tape Turing machine and this is doing something let us say it is recognizing a language and we would like to recognize the same language by means of a single tape Turing machine let us call that Turing machine as T and right now what I am describing you is at this single tape Turing machine T we will have 4 tracks. Now the significance of this 4 tracks will be that in this first track we will have the contents of tape 1 of the 2 tape machine M, so for example if here somewhere we will have and contents of tape 2 will be written in the third track.

So let us say here we had d b prime c c here we will have it not be aligned but in this case you see there is really no question of alignment between this tape and this tape, so let us just to say what we mean is we will say that this is here d b prime c c, I am sorry this we will write here in this tape. So this track will have contents of tape 1 and this third tape will have contents of tape 2, fine but we also need to represent the positions of the head, right?

So in this example situation the tape head of first step is scanning the symbol b, this b so the tape head of you know this particular fact we can represent by the situation of this kind. So the point is the second track will be blank except there will be 1 in 1 cell to represent that the cell corresponding to this cell right above your that is the cell which the tape 1 head is scanning and similarly for track 4. So in track 4 we will have this particular c is being scanned so 1 here and all these are blank, alright? These are whatever some symbol are there, maybe blank we do not know.

So you what we are going to do is that one movement or one move of this machine M will be carried out by a number of moves of this machine T, T is a single tape Turing machine with 4 tracks and what it does? It will can imagine that will carry out a number of steps to simulate one step of this machine, one step of this machine of course means that the 2 symbols are updated under 2 heads and this state is changed and the 2 heads move you know in directions appropriate.

So this is what we would like to carry out and now the I will try to give you the overall idea here, so imagine the beginning of simulating one step of this machine by this single tape Turing machine at that time in the beginning of this simulation of one single move of this machine we can assume that the tape head this is remember this is a single tape Turing machine so it has only one head so that particular tape head will be scanning that cell, okay which is which contains the left of the two symbols under the two heads.

So situation is like this that you see on tape 1 this b is being scanned and on tape 2 this c is being scanned. Now when we represented like this the b that is being scanned is to the right of this c, so we will imagine and that is the kind of invariant you can assume what is invariant that is at the beginning of simulation of one step of this machine the tape head of one tape Turing machine is positioned to that cell which contains the left of the two symbols being scanned by the two tape machine.

So in because of that we can assume that in the beginning this is where the tape head of the single tape machine is. And now you see from this what it knows as it is scanning all these four symbols because this is a single tape Turing machine with 4 tracks as we had explained that in a move it is scanning all the symbols in the track it will be scanning, so therefore it sees b on the top track blank on the second track c here and 1 immediately it knows that this is the symbol

which is being scanned by the tape 2 of the two tape Turing machine tape 2 because the right see the of all these of these 4 symbols the one which is here is under c which is on tape third track so which is the which corresponds of course to the tape 2.

So now it knows that the machine M on tape 2 is scanning c and let us say it is in a it also knows what is the state of the Turing machine M so which is of course q. So now the state of T is going to be composite object one of the things one of the components of T's tape T's state is going to be the state of M, so it will it remembers in its finite head what is the tape what is the state of Turing machine M and now what it can do you see right now does it know what will be the move of machine M, no because it does not know what is the symbol tape 1 head is scanning.

So however it sees the moment as it is scanning this it knows the tape 2 symbol is c, so let us say it will carry this information after reading this it puts the information that the tape 2 symbol here which is being scanned is c. So let us say that it can carry that symbol in a component c and now it knows for the tape 1 symbol it needs to move its head to the right it can be somewhere here but it knows because of our invariant that the second symbol is going to be on the right of the symbol that you first have seen in the beginning of the simulation of one step of M.

So it will move to the right this tape head is going to will move to the right till it finds a 1 in the second track it will find immediately because in this example it is right here so it moves to the right you see having picked up symbol c in its head there is no problem because the number of symbols is always finite for any Turing machine therefore for M also. So that can be stored which particular symbol is being had been read can be kept in the state.

So now it moves to the right and sees that b is the symbol which is being read by the first tape head. So now it knows the machine is in state q which machine this machine it is trying to simulate machine is in state q it is scanning a the situation is it is scanning b on the first tape c on the second tape and now it fully knows capital T this machine knows what is the present state of the machine and the two symbols being scanned.

So that information is enough for T to know what state M will be in the next instant next step and what will be the two symbols which will update the current symbols in two tapes, right? So therefore since it knows the information it knows what is to be done, so for example if the situation was that in state q if you are reading b on the firsts tape here c here he will update b to

let us say d and c to let us say e and this head will move one step to the left and this head will move one step to the right supposing this is the situation that would have happened and the new state would have been q prime.

So the simulating machine would know all that because after all that is a finite piece of information which is the transition function of this machine. So from q what is the in this case it is the second tape remember because it is that is the that was to the left so essentially one of the components here would be that which is to the left of I mean relatively speaking the two heads which is to the left and which is to the right, so for this representation is concerned.

So this component two is saying the tape 2 head was relatively when once we put like this was to the left. So it has all that information so now it can complete one step simulation of this machine by updating the situations, right? So what was the situation we said? That the symbol would have been would have become d so tape 1 symbol should have become d this should become d and what is the symbol here the symbol here was we said that it will write e and move to the left it will write e and move to the left, right? So what it should do, so let me circle it that this is the situation that we want but and tape 1 head will move one step to the right so this 1 should be made blank and the 1 should come here and this 1 should be made blank and it should come here because the tape 2 is moving to the left and tape 1 is moving to the right, okay.

So let me emphasize once the simulating machine knows the state present state and the two symbols on the two heads it knows what the all the updates that are going to place it is only a matter of correctly showing or are doing the updates here in the representation and so you can you can see what it is going to do because it had said that this the Turing machine first head will move to the right so therefore the what was 1 here that is made blank and since it head is supposed to move one step to the right the 1 in this track will be positioned to the next and similarly since here the head this head was moving to the left and this will come here.

So all these update will be done and then the machine will move to the all the way to the left as it moves to the left you see as it moves down it can update this particular symbol which will do and it will come here writing 1 and now it is positioned correctly you see that in the second tape it is scanning b prime which it is and on the first tape it is scanning a prime which is also the situation and now from all this it is ready for the starting of simulation of the next step of this machine so

that time this will be since it has move to state q prime the composite state here will be q prime, right? And right now it has not read what was to the left symbol so you can say anything something is written and still is it is the same situation that is the tape 2 is on the tape 2 head is relatively to the left and so this 2 remains and I am not shown certain other things here that is another component which is for doing all these kinds of things.

See for example it remembers that it should put a whatever see from the when it sees the two symbols on the two tapes for this machine then at that time it knows that what will be the new symbols but that time the head of this machine is positioned under one of those which it can update immediately but then it should remember what was the other symbol so that will be another component it will remember that and as it moves down when it sees the correct place it will make the update and so on.

So you see that the idea is that this machine has in its states several components and of course it needs it something for doing some information or some state information for doing its own jobs namely moving to the left moving to the right. So for that also you need to have some information in its head and so all that is also is a part of the composite state information of this machine.

So to summarize this whole thing what I am saying is in the beginning, beginning of what? In the beginning of simulating one step of this particular machine we assume that the machine head is reading that symbol here which is of course there are 4 symbol basically there are 4 symbols in 4 tracks that particular position its head is which corresponds to the symbol which is being relatively to the left of the two tapes in that representation.

And it also has in its finite state the state the machine that is being simulated the current state somewhere here and then this machine will move to the right to look for the symbol on the other track once it knows both the symbols it is going to update one symbol and then move to the left to update the second symbol also simultaneously it is going to update simultaneously in this process the 1's remember these 1's are really markers as to the position of the two heads.

Now one slight detail that you should remember that in this step that we had given example for the relative position of the two heads are not changing but can so happen that you know right now it is to the left this head is to the left of this but this head can keep moving to the right and

this head can keep moving to the left and then sometime the head here will become relative to this particular head to the left for that representation.

So when that happens this will be updated and that is that situation is for that situation to happen the two heads must have been adjusant or simultaneously in the same you know basically in the same four tuple of the track. So it is not a very difficult thing if you think about that the simulating machine will know when the relative position of the two heads are changing, alright? For each step of this machine I said that the simulating machine will make a number of steps how large that number can be? So you see what is the worst case? The time simulating time is spend most of it really is that looking for where the other tape symbol was.

So the point is how far apart the in that representation these two heads can be, in the worst case you see what can happen is that in every step one of the heads moved in one direction let us say this head kept moving to the left, this head kept moving to the right. So in if this machine has taken t(n) steps then the relatively the two heads here two ones here will be at a distance of 2 t(n), right? Is that clear what I am saying? What I am saying is that the worst case situation is, worst case for what? Worst case for the in that representation relatively how far away the two heads can be.

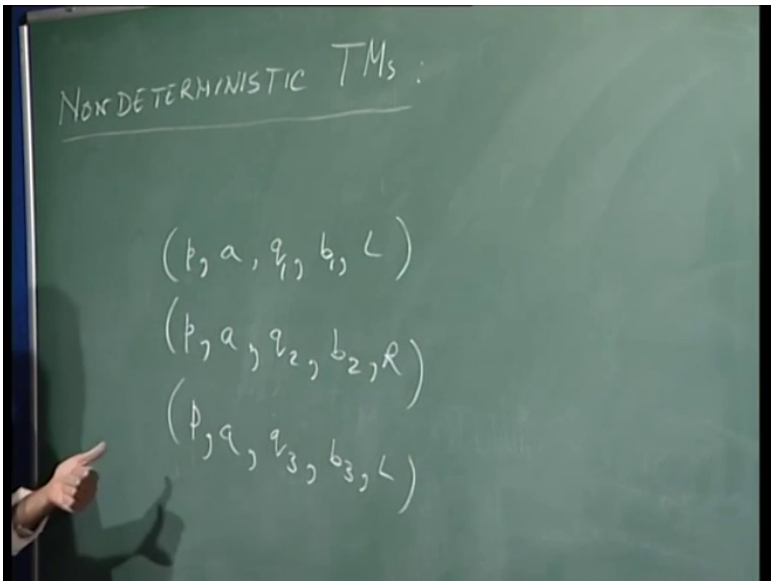
In t(n) steps the two heads can be around 2 t(n) you know cells away here. So in the worst case if this machine to recognize something to t(n) steps in the worst case that particular machine will take two t(n) steps for each tape of M, now M is taking t(n) steps this is taking two t(n) steps for each step of that. So total number of steps is going to be ordered t square n, so there is a quadratic blow up in this simulation something was happing in the t(n) steps you would require t square n steps here but that is okay because you know ultimately we will see that our concern is that if a simulation can be done in polynomial time than that is good enough for us there is a quadratic blow up but this is a polynomial if this is ordered t(n) this is order t square n.

So the blow up is the simulation cost first thing is quadratic which is okay. So that completes our discussion that multi-tape Turing machines can be simulated by a single tape Turing machine and therefore having a number of tapes does not add extra power in terms of recognition of languages over machines which have only single tape. So this our basic model is robust enough although I should say in fact we will see examples that having a number of tapes can be convenient but the

point that is we made in this is that it may be convenient to have a number of tapes to carry some work but it is not essential if pressed you can do it with a single tape. Our next topic is non-deterministic Turing machines we have not defined what non-deterministic Turing machines are but from our knowledge of finite states machines and push down automata it is not difficult to define the in an appropriate manner what is a non-deterministic Turing machine.

Non-determinism means that in a particular situation the next move can be taken as one of several possibilities and the choice is made non-deterministically.

(Refer Slide Time: 32:37)



So if you go back to our notion of a single tape Turing machine we had present state let us say p and at that time let us say the symbol that is being scanned is a, now if the machine is deterministic that this will completely determine what the next state is what the symbol that would be written here and which direction the head will move but suppose we have a Turing machine where for same or identical values for the first two components we had several different quintuples.

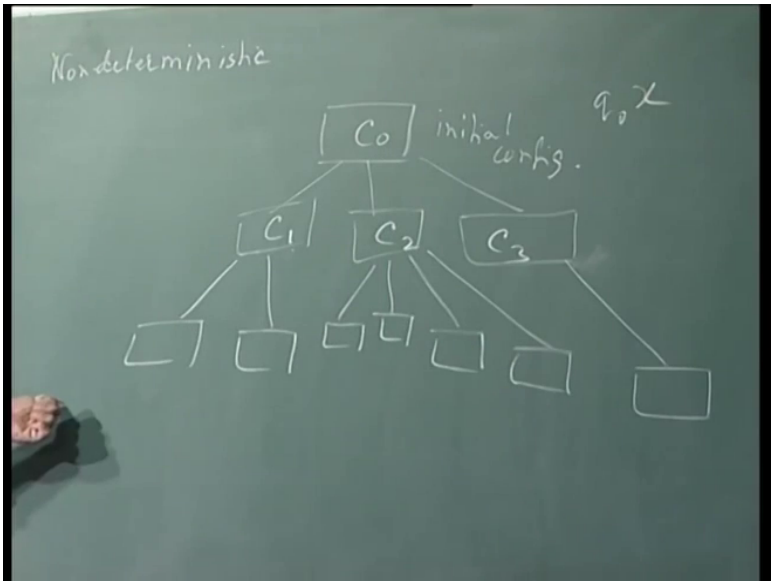
So suppose we had also we had p, a, q2, b2 let us say R and let us p, a, q3, b3, L. So suppose our Turing machine that we had in that definition of the Turing machine the situation was that we had a quintuple p, a and this as well as we had another quintuple you see this is a distinct quintuple in the sense q1 and q2 are there different maybe they are also different. So for the same present state and present symbol we had a number of quintuples.

So in that case what can we say? The machine when the present state is p and it is scanning the symbol a it has three choices either it can go to state q1 writing b1 in the current cell and moving to the left or it can go to state q2 writing b2 and then move to the right or it can choose to go to state q3 writing b3 in a and move to the left. Now non-determinism means that the machine non-deterministically we will choose one of these three quintuples when the present state is p and the current symbol being scanned is a.

So really speaking a non-deterministic Turing machine also will be described in a manner formally as a set of quintuples and of course it will have an initial state that we make it unique and it will have a number of accepting halting states, it will have tape symbols, it will have input symbols all those things are similar only thing in the set of quintuples that go to define the Turing machine in case of non-deterministic Turing machine we may have several quintuples 0 or more quintuples for the same single same present state and present symbol combination.

Now what happens, what happen as we said the in an execution of a particular run of a non-deterministic Turing machine Turing machine will non-deterministically chose one of the quintuples one of the possibilities it can make use of and go ahead.

(Refer Slide Time: 36:46)



So this situation we can visualize in a following way that initially the Turing machine was in a certain configuration and for a single tape Turing machine that configuration is that you recall our convention of writing configurations there is the initial state in that initial state it would be

scanning the left most symbol of the input if the input is x and of course rest of the tape is blank and that is the initial configuration and that we write like this.

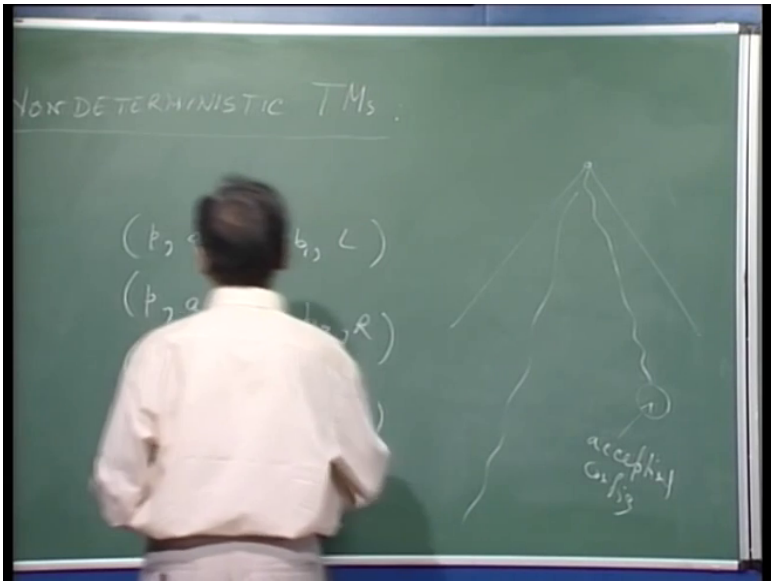
Now in case of deterministic Turing machine from this configuration deterministically or uniquely the machine will go to another configuration and then another configuration and so on. So the computation is of course in case of a deterministic Turing machine if we think in terms of configurations that it will be a linear chain of configurations starting from the initial configuration this.

Now this is the situation when the machine is deterministic, now what happens when the machine is non-deterministic? In case of non-deterministic machine such a linear chain of configurations a representing the execution of a Turing machine is no longer possible, why? If we want to capture what all things that could have happened, see because this is the initial configuration now there may be from this if such was the situation there are three things there maybe three configurations so this if your initial configuration c0 we have let us say is possible that from c0 the machine can go to c1 from c0 the machine can go to c2 from c0 the machine can go to c3.

Remember when we speak of non-deterministic machine as opposed to deterministic machines we cannot say from configuration this the machine will go to the configuration that, all we can say is that the machine is in some present configuration then there are a number of configurations and the machine can go to anyone of them. So in this example situation supposing from initial configuration c0 it can go to anyone of these, maybe from here it can go to these two configurations maybe from here it can go to four configurations and maybe from this it can go to only one configuration.

So the point I would like to make is in case of non-deterministic machine if you want to capture fully all possibilities then you should think in terms of a tree of configurations, so this is called a configuration tree, right?

(Refer Slide Time: 41:00)



So in general this is from the starting initial configuration we have a tree of configurations in case the machine is non-deterministic, right? So when we such a machine accepts the input the machine accepts the input if for anyone of these you know supposing you can there is a path from the root in this tree such that in that path you reach an accepting configuration, okay.

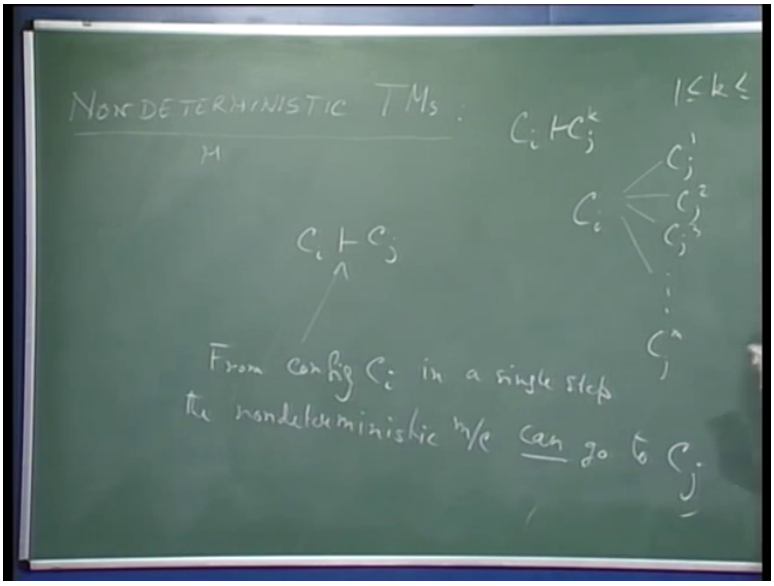
In that case we say the input would be accepted by the non-deterministic Turing machine the definition is very similar to other non-deterministic situations that you might have known namely finite machines and pushdown automata at least for these two you have seen non-deterministic versions. So you remember that a non-deterministic machine for a non-deterministic machine we say that the input is accepted if there is a sequence of non-deterministic choices which can lead the machine from the initial configuration when accepting configuration, right?

So basically a path in such a tree means what? That means that you are or the machine is exercising certain choices in moving down the tree those are the non-deterministic choices. So therefore once more let me emphasize that acceptance by a non-deterministic machine can be seen as a path in the configuration tree path from the root to a in the configuration tree where that path end in an accepting configuration, right?

Now a the tree itself can even be infinite in that some paths here maybe nonterminating you know some parts here maybe nonterminating that is fine but still we say since there is one particular path from the root to an accepting configuration the input is accepted. Let us formally

capture the notion of acceptance, how would we do that basically we will update the same formulism that we used to explain what we mean acceptance by a deterministic machine.

(Refer Slide Time: 44:17)



So you recall that we had a relationship from between configurations Ci and Cj and we read this as you know in the context of a particular Turing machine M in case of deterministic thing if the machine is in configuration Ci then the next configuration will be Cj and that situation we represented by means of you this notation which is really speaking that this you know you read it in many ways you know this is a turnstile symbol and you can say that Ci in one step leads to Cj.

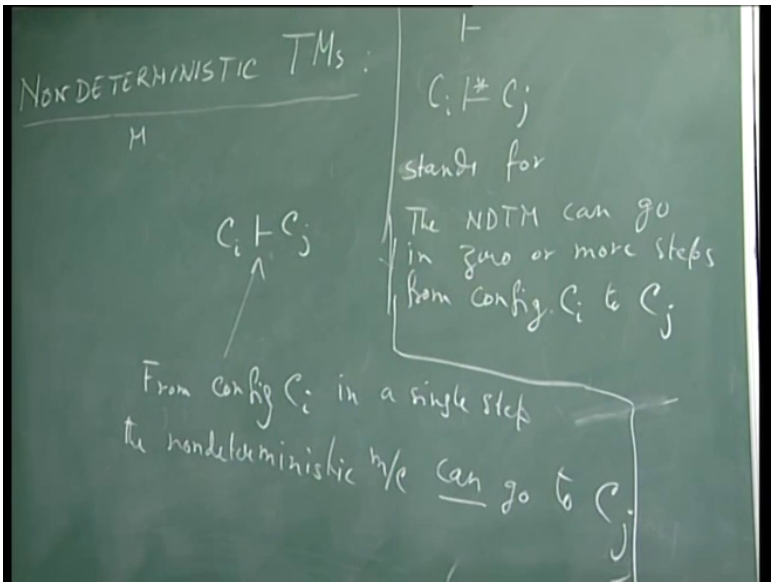
Now remember that in case of non-deterministic machine in one step Ci may go to several things like Cj 1, Cj 2, Cj 3 dot dot dot Cj m, right? But there will be a number of configurations the machine can go to. So for non-deterministic machine the same one step you know relationship one step derivation relationship amongst configurations we will can use the same thing provided we understand this as from configuration Ci in a single step the non-deterministic machine can go to configuration Cj.

So there is no determinism so we cannot say from Ci the machine will go to Cj but if there are several possibilities so essentially in this situation we would say that Ci can go to any of this Cj, so let us say write you know k here k can be in this example k can be anything from 1 to m. So let us repeat once more, suppose I had a Turing machine in which is a non-deterministic Turing

machine and in from configuration Ci the machine can go to all these configurations then what we can say is we will write that Ci in one step can go to Cj k for all these values of k, right?

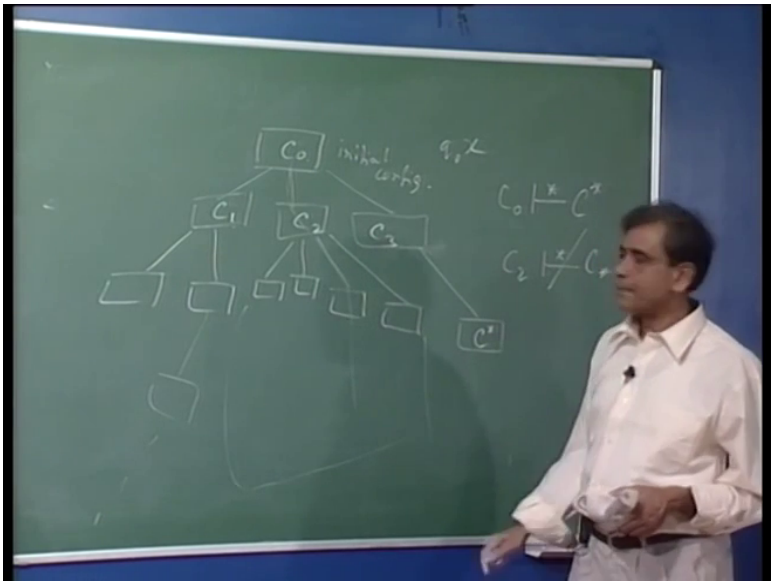
So what compare to the deterministic case we are changing the meaning of this symbol or the relationship there in case of non-deterministic machine the relationship was between two configurations Ci, Cj such that Ci from configuration Ci the machine will go to configuration Cj in case of non-deterministic machine all that we do to update is that we say again these two configurations are related provided from configuration Ci in one step the machine can go to the configuration Cj.

(Refer Slide Time: 48:47)



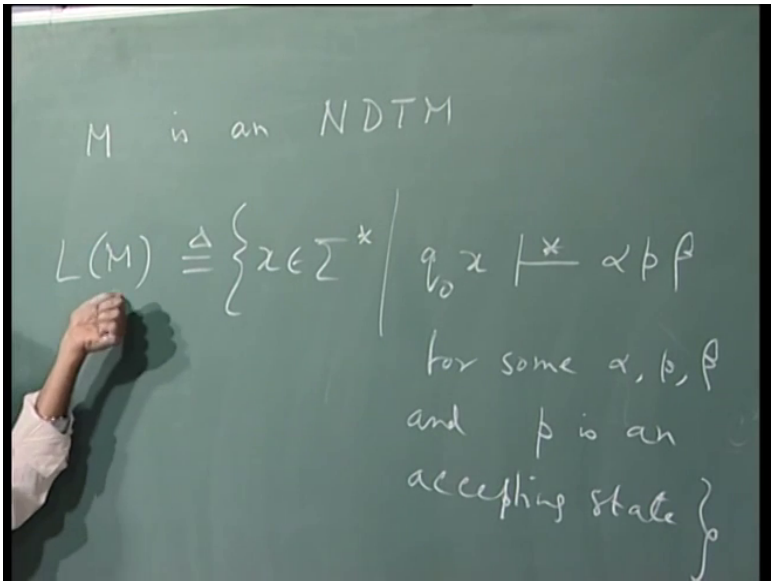
So once I understand this then what we can do is the same thing that we did for deterministic machines what we did was from this particular relationship or you know relation amongst configuration we defined another relation which stood for so this means that in 0 or more steps the machine under discussion can go from the configuration Ci to configuration Cj means or stands for configuration let us write it this way that the machine the non-deterministic Turing machine can go in 0 or more steps from configuration Ci to Cj, alright?

(Refer Slide Time: 50:16)



So in this configuration tree what it would mean? That for example this is C you know maybe star here, so clearly we can write C0 in 0 or more steps can go to some configuration C0 you can go to C star but you see from C 2 you cannot go to C star because I mean if C star does not occur in the sub tree here you know it may the same C star same configuration may occur somewhere here but if you that situation is not the case then from C2 you cannot write that you can go to C star, right? So essentially in terms of the tree any node and its anyone its ancestors will be related by this relation, right. So once we understand this relation in the as a simple generalization of a single step what can happen in single steps then it is easy to write what the language accepted by a particular Turing machine is.

(Refer Slide Time: 51:51)

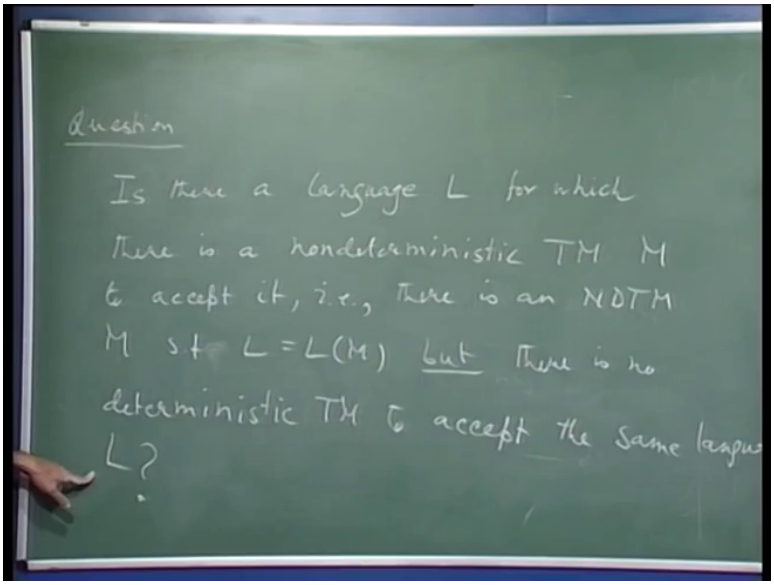


So suppose M is and NDTM, NDTM always is the abbreviation one uses for non-deterministic Turing machine. So M is an NDTM and the language which is accepted by this NDTM is defined to be this is the symbol to say that this is by definition is the set of all strings over the input alphabet sigma such that q0 x remember this, sorry this is not sigma 0 q0 x qo being the initial state of the machine M, so q0 x is the initial configuration in 0 or more steps you can go to some alpha p beta for some alpha p beta and this p is an accepting state, alright?

So essentially the idea is very simple that if starting a non-deterministic Turing machine in its initial configuration if there exist a sequence of configurations such that the machine can go can go from one configuration to the next leading to a configuration which is an accepting configuration in that case we say the string is in the language of the machine M. Now therefore the question which now is I am sure you are asking is that see clearly non-determinism seemingly adds power because it know can make choices non-deterministic choices the non-deterministic machine can make non-deterministic choices.

Now is this power such that that with this power a non-deterministic machine can recognize a language which is not there in the class of languages recognized by deterministic machine.

(Refer Slide Time: 55:08)



So the question that we have to answer for or is there a language M for which there is a non-deterministic Turing machine M to accept it that is there is an NDTM M such that this language L is the language accepted by the same DTM and or we can say but there is no deterministic Turing machine the same language L, okay?

So the question is, is there a language L for which there is a non-deterministic Turing machine M to accept it but there is no deterministic Turing machine to accept the same language here, if the answer to this question is yes, then non-determinism indeed adds power extra power recognition power over deterministic Turing machine and if the answer is no, non-determinism will just be a convenient tool to express some languages to express some algorithms but whatever it does in terms of language recognition could be done by deterministic machine.

So essentially this question is about whether non-determinism in case of Turing machine adds extra power over deterministic machines for recognition purposes or not.