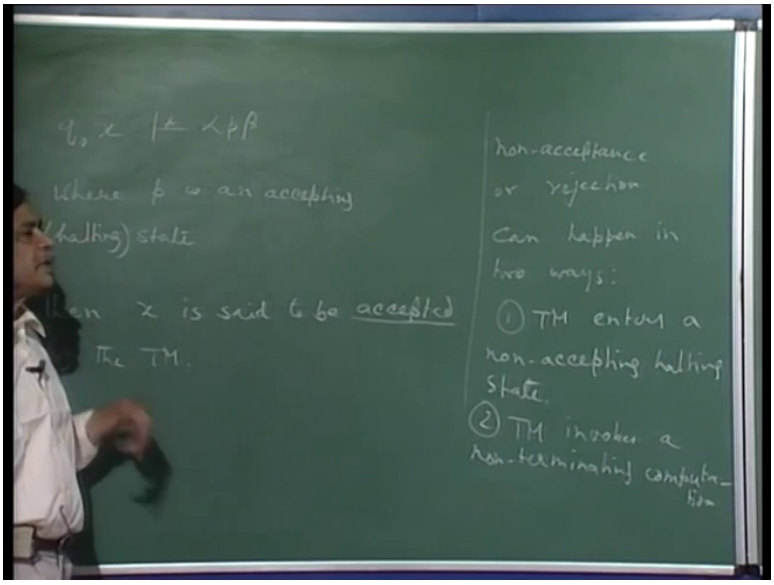
Course on Theory of Computation
By Professor Somenath Biswas
Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur
Lecture 37
Module 1
Notion of non-acceptance or rejection of a string by a TM. Multitrack TM, its equivalence to standard TM. Multitape TMs

We should be very clear about the notion of acceptance by Turing machine because this is a somewhat this notion is somewhat different from the notion of acceptance as we were familiar in case of finite state machines. Now recall what we said about a string being accepted by a Turing machine.

(Refer Slide Time: 0:43)



So we said that the configuration initial configuration was Turing machine was in the initial state q0 and scanning the left most symbol of the input string let us say the input string is x then we said that in if 0 or more steps from this configuration the Turing machine goes to a configuration alpha p beta where p is an accepting halting state all accepting states will by definition will make them halting.

So when we say the machine enters an accepting states you know at that time the machine has decided that it is going to accept the string so therefore there is no reason for the Turing machine to operate any further. So when from an from the initial configuration it reaches a configuration

where this state is an accepting which is halting state then we say if such a thing is possible then we say that the input x is accepted by the Turing machine M said to be accepted by the Turing machine and we also used the word rejected a string is rejected by a Turing machine.

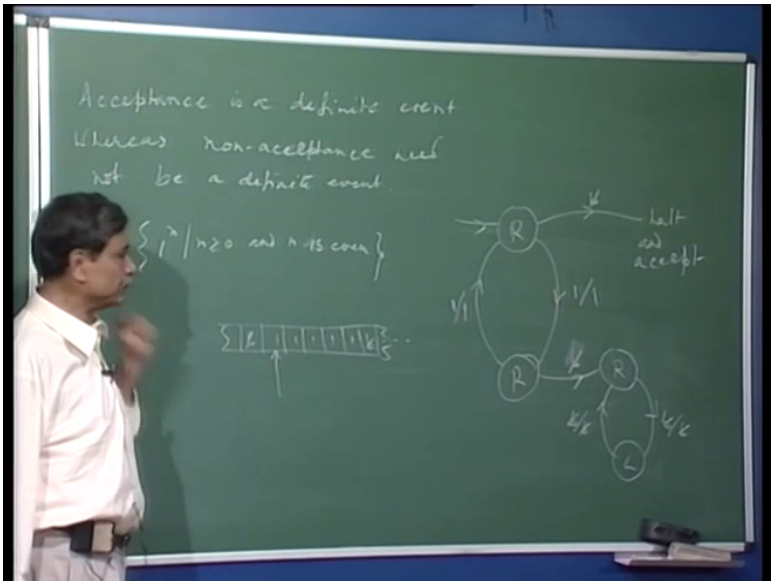
What does rejection of a string mean? Rejection is a word which is equivalent to the string is not accepted, string is not accepted means that this situation will not come about. Now you can see that non-acceptance or rejection can happen in two ways and what are these two ways, one the machine enters a state which is halting but not accepting TM enters a non-accepting halting state that is after some time on the input the Turing machine enters a state from there no further computation will occur and that particular state is not an accepting or a final state.

In that case of course this situation has not happened and therefore that input string is non-accepted or rejected, however there is yet another way a string can be not accepted or rejected and that is Turing machine invokes a non-terminating computation. What we mean by this that the Turing machine keeps on working and it never comes to a halting state and therefore it has not reached an accepting state with by definition is halting and the work of the Turing machine goes on forever but in that case this situation would not have not occurred, and what is that situation? In 0 or more steps which is always finite can be arbitrarily large in some number of steps the machine reaching a final accepting state.

So in this situation again this will not come about and therefore the input is not accepted or another way of saying that is input is rejected. The point I am making is important that acceptance of a string is an event in the sense that the string invokes a computation which finally reaches a state which is accepting and halting, whereas non-acceptance need not be a definite event in the sense a definite one this corresponds to non-acceptance by a definite event it has reached a state which is a non-accepting but halting state.

So at that time the Turing machine would know that the input string is not accepted but on the other hand if this is the case that happens Turing machine goes on working, goes on working so there is not a definite event by which one can say that the Turing machine would know or you would know if you are just looking at the process of the computation of the Turing machine then the string is rejected.

(Refer Slide Time: 6:45)



So the point is acceptance is a definite event, what is that event? And that event is the Turing machine reaching and accepting halting state this reaching and accepting halting state is a definite event. So therefore acceptance is a definite event, whereas non-acceptance need not be a definite event and that is the case when the machine invokes a non-terminating computation.

Just to illustrate this point let us take a very trivial example of let us say the language it is the all blocks of 1 strings which have length even, so this is the language L it is a very simple Turing machine to accept it or to recognize this language L and in this case in this state if you see a blank then you halt in a and not accept. So remember what input would have been written like this that there is a block of 1's flanked by blanks on both directions and if you start the Turing machine here if you just trace you know starting from here it sees the first 1 it does not change it but moves to the right and so on it sees here this 1 in this state and it again does not change this 1 but moves to the right and finally it will be here in this state and here it is seeing a 1, right?

So what is going to happen? This 1 it was here it has come here and this 1 it is here and going there and so on and the after some time it is going to see blank, right? Now in the beginning if it was the if the string was 0 length 0 then of course it sees blank and should halt an accept because that means n is equal to 0. On the other hand if there is see basically a string of length even length drives the machine here string of length odd will drive the machine final here and at this

after moving to the right if it sees a blank then it knows the string that it has seen is odd and therefore it will halt and not accept.

And in this case if it sees the end of the input which is a blank symbol it knows that it has seen what it has seen as an even length string and the input is over and therefore it should halt and accept. In this trivial example both acceptance and non-acceptance they are definite events, right? Because when it accepts of course acceptance we said is always a definite event it halts and accept but here when it is not accepting a string it is halting but not accepting. Now suppose what we do here is that here instead of halt and not accept we make the machine go into an infinite loop and that can be something like this.

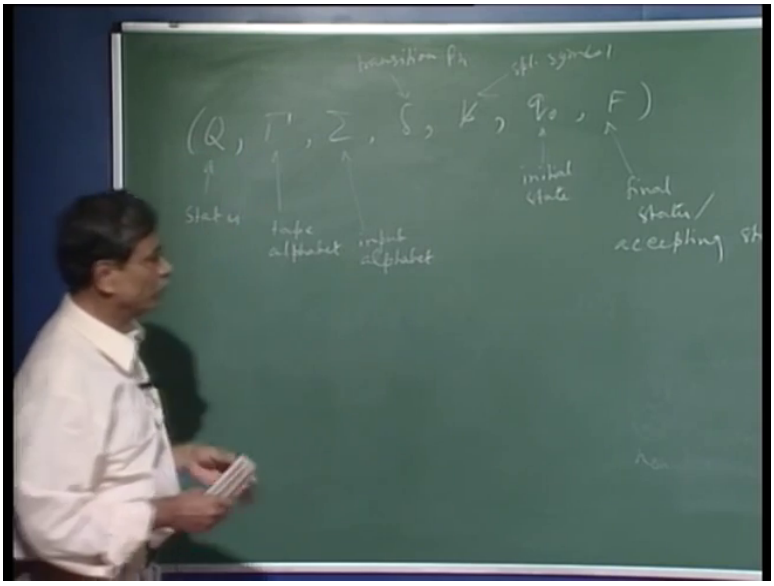
So what happens after it has seen an even number, odd now sorry odd number of 1's it will be here we will see a blank and therefore it will go to this state will move to the right and again of course it sees a blank because everything was blank here it keeps the blank the symbol as such goes to the left and then it will again see a blank it again does not change that blank symbol goes to the right. So it just goes on in this loop infinitely on and on and on and on.

Now here what is happening on an string of length even the machine is will halt in an accepting state on an string of once which is of length odd the machine is going into an infinite loop, what is the language this particular Turing machine recognize or accept? Clearly the strings which take the machine to the halting and accepting situation here those are the strings which are in the language accepted by the Turing machine. So that is same as the one before whereas here it is doing some non-terminating computation on strings of length odd but that does not change the situation so far as the strings of even length are concerned. So my point is even this machine which is evoking an infinite computation on strings which are not accepted this particular machine also accepts this same language.

So the two machines although you know this is slightly way of doing is but it is perfectly alright to clam that this machine is also a machine which accepts this particular language. So therefore as you can see that the non-acceptance or rejection of the strings which are not in the language in this trivial example those are strings which are of length of odd and they invoke finally and non-terminating computation so that is fine but even then those strings are not accepted because these

strings these odd length strings do not come to this situation where the machine goes into a halting and accepting state.

(Refer Slide Time: 15:41)



So we have clearly need to understand this particular point one more point I would like to make here and then go over to you know generalizations and restrictions of our basic model and that is the point I am saying is that this is of course a convenient way of representing a Turing machine but we have said formally a Turing machine is what, formally a Turing machine is a number of things basically it is a tuple that is the way you say what something formally is that a Turing machine is a tuple consist of set of states and you know tape alphabet, input alphabet, then transition function, then a special symbol blank this is the symbol which is initially there in most of the tape cells except the part which are occupied by except those cells which are occupied by the input and what else do you need to specify a Turing machine you must say where would the Turing machine start which state the Turing machine start in so this is the initial state and we must say also which are the final accepting halting states.

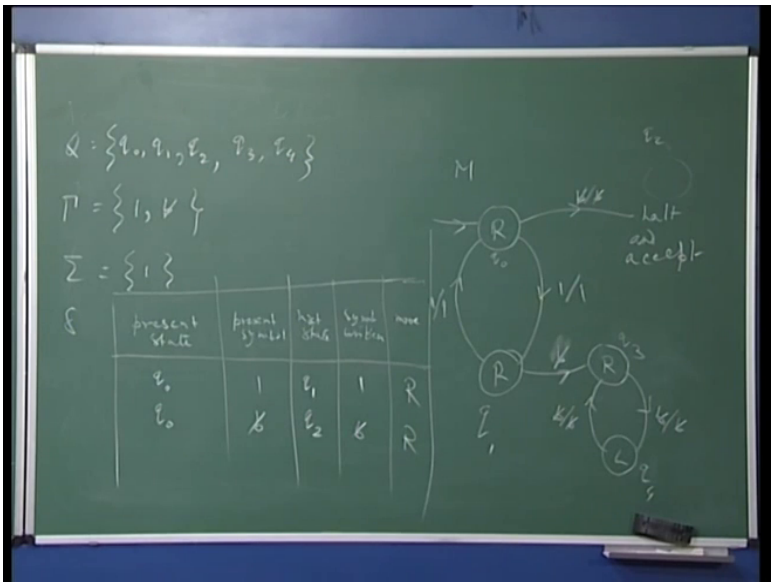
So I will not say again and again accepting final halting states because we assume always that a final or an accepting state is a haling state. So let me just simple say these are final states or also equivalently we call them accepting states. So I would like to make sure that although informally we describe Turing machines simple Turing machines using diagrams like this transition diagrams like this formally we will represent Turing machines through a such a tuple by

definitely specifying what the set of states are, what is the tape alphabet and so on, remember that this is a finite all these are finite entities because this is for a particular Turing machine the set of states is finite and so on.

So let us take this particular example and workout so that we are absolutely clear that the formal way of specifying a Turing machine by means of such tuples and then elaborating each one of them separately and diagram like this they are not a you know you can go from one to another. So what would you say set of states of this machine if you say this machine is M so we can clearly see that the we have of course these 4 states but also there was another state here which is the really is stood for the final halt or accepting state in which of course you halt.

So suppose I name this as q0, this is q1, this as this state which is the halt and accept as q2, q3, q4.

(Refer Slide Time: 20:12)



Now basically we are saying that okay the set of states Q will be q0, q1, q2, q3, q4 and the tape alphabet consists of in this example remember what we had initially was just a block of 1's flanked on both sides by blanks, so and we are not writing anything other than 1 or blank, right? So this even a blank you write blank that is okay and this is it. So the tape alphabet consists of 1 blank input alphabet input was a string of 1's only, right?

So input alphabet is just 1 consist of the symbol 1 and now delta, right? Delta you have to specify by given by giving the map which would tell me given a present state and given the symbol being scanned what will be the next state etcetera etcetera. So we can see delta I can specify by means of a table the table of quintuples and given something like this present state, present symbol, next state, symbol written and move direction of move. So just to be very clear for the if the present state it q0 in q0 if you see a 1 then you move to state q1 you write of course do not change that symbol to anything else write it as 1 itself and move to the right.

So this is the arrow which corresponds to this arrow corresponds to this row of this table. In q0 you can see a blank if you see a blank what you do? You go to this halt and accept state which we are calling it q2 and move in which direction maybe we just right it does not matter and so on you can fill up this table the only thing if you q2 since we are taking q2 as our halt state that means from here nothing happens, so there will be no quintuple with q2 in this column that means once we reach q2 then the Turing machine has nothing to do and therefore another way of saying is that it halt, okay.

So all I am trying to say in this that the way this was a finite object this diagram and this had all the information corresponds to giving a very formal way of specifying a Turing machine. So like this arrow which coming from nowhere into this state q0 it signifies that q0 is the initial state which we have to mention here, right? What is the special symbol that is not really written here I mean from this diagram you cannot really make out maybe that what is the symbol which is there initially for most of the cells and that is one limitation of such a diagram but you can otherwise specify that the symbol blank is what will be there in the most of the cells in the beginning all the cells except where the input present.

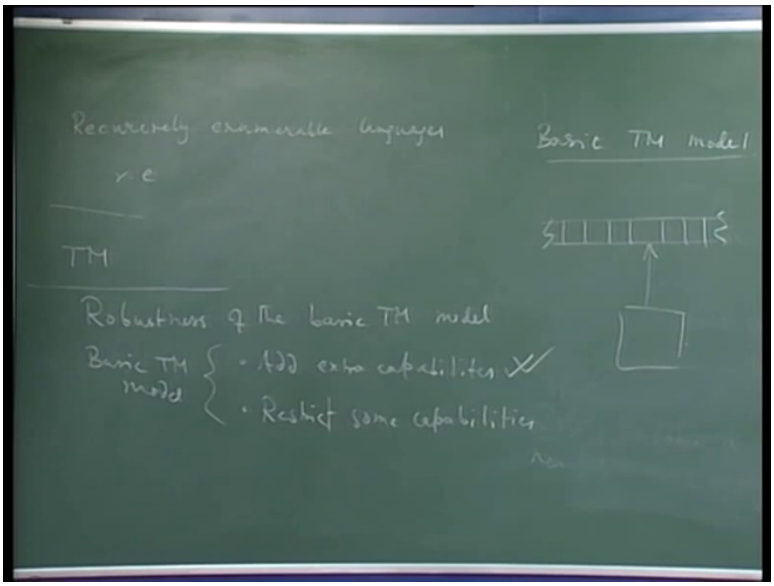
We are kind of assuming that the input alphabet does not self-contain blank without any loss of generality. Now from here now that we understand acceptance and non-acceptance of a Turing machine clearly and also we understand informal definition and formal definition of a Turing machine by means of you know specifying all these tuples this particular tuple and then specifying individually each one of these.

See these are all each one of these is a finite object like this delta which is the transition function is a finite table (())(25:42) made the point why it will be a finite table simply because both set of

states and the tape alphabet these are finite sets therefore it cannot have more than cardinality of q multiplied by cardinality of gamma rows it cannot have more them so many more rows than cardinality of q multiplied by cardinality of gamma and then therefore this table is also going to be finite.

So Turing machine is a finite object whether represented by means of diagram like this or by means of more formally a table and tuple in any case the point I am making and I am hope it is absolutely clear that one particular Turing machine is a finite object, alright? So now from here we will look at a certain generalizations or restrictions of Turing machines our basic Turing machine model and argue that neither the generalizations nor certain restrictions make any difference so far as the language recognition capability of our of Turing machines are concerned.

(Refer Slide Time: 27:25)



So just to get back to the motivation we define a class of languages called recursively enumerable languages and we said a language is recursively enumerable or for short a language is r.e if there is a Turing machine to recognize that language, right? And our Turing machine was model was you know that basic Turing machine model that it has a tap which is going on to infinity in both direction and there is a finite control with finite states it can the machine can move up and down changing state, changing symbols and cells and so on.

And we would ultimately like to claim that by means of whatever algorithm that you may think of you cannot ever recognize a language which is not recursively enumerable and also we will

also claim the other way that if a language is recursively enumerable then that language decision problem at least given a string whether it is if it is accepted we can have an algorithm which will halt and say the string is accepted.

So basically we would like to as I said right in the beginning in our discussion of Turing machines that we would like to equate the informal notion of algorithm with notion of Turing machine computability. So therefore the question which becomes important is that you know we are working with we have defined this particular model which you may call the basic Turing machine model. So the question which might arise very naturally is that look maybe that if you change this model a little bit without intuitively losing the idea of effectively doing computation maybe then the class of languages which are recognized by you know changing the basic model will be different from this class.

So in that case this class will not be robust in fact what should we do to convince ourselves that this is a good enough model we do not need to think of any restrictions or any generalization which are natural enough. So you may say our the point we are trying to prove is the robustness of the basic Turing machine model, two thing we can do to the basic Turing machine model either we can you know add some extra capabilities or restrict some capabilities.

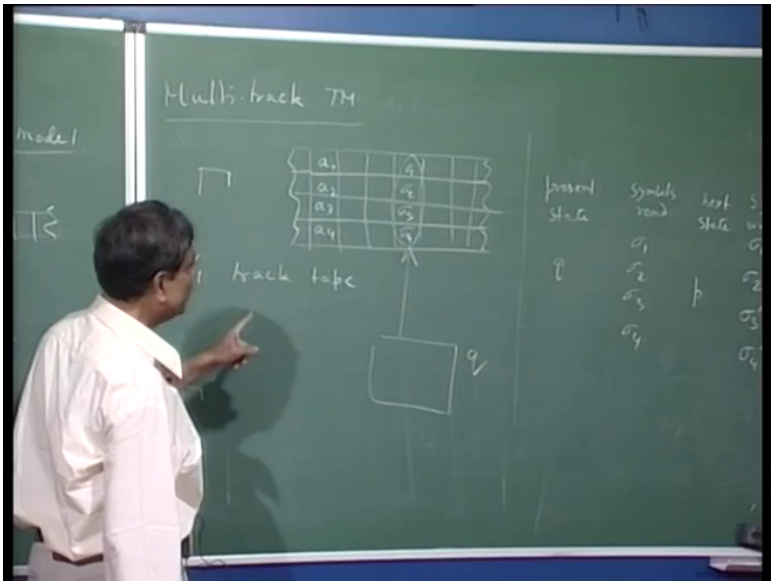
So what I am saying is suppose we do this two our basic Turing machine model and then find suppose then find that whether we add some extra capabilities or whether we restrict some capabilities you know some natural things we do not want to do something very unnatural once. On the basic Turing machine model then the new machine or class of machines that we get do not change the class of languages accepted.

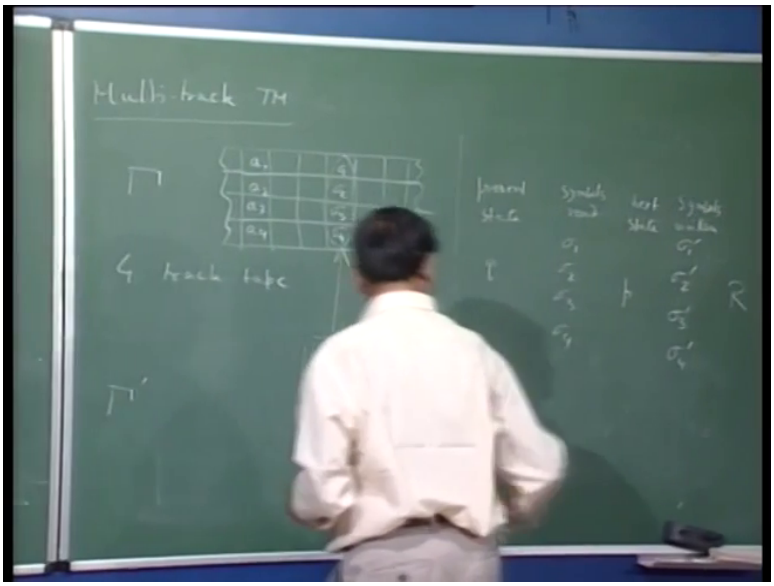
In that case we can claim that this set of or this class of languages defined through or basic Turing machine model is a good grasp because you know changing small things here and there do not make any difference to the class of languages which are or which is recognized. So first of all we will try to add a few capabilities to the basic Turing machine model this is model and let us see what we get.

You know we are talking about tape and in our basic Turing machine model the tape you may say has a single track in the sense that the tape consist of is known array or just arrangement of linear arrangement of cells and each cell can have exactly one symbol with you might have known you

might be knowing about you know real life tape devices and there people talk of tracks that a tape has not just such tapes are not just a linear arrangement of cells which can contain exactly one symbol but the kind of picture that will have for a multi-track tape will be something like this, right?

(Refer Slide Time: 34:08)





This is 4 tracks and each track is like this thing this tape, right? So you may have a symbol a1, a2, a3, a4 so basically on the 4 tracks this is a 4 track tape and you can have different symbols occurring at different tracks and then what is your read write head would do, suppose instead of a single track Turing machine we had a 4 track tape Turing machine. Now we can see it is

reasonable that if this is my read write head and that what happens in real life tapes with multiple tracks simultaneously the read write head will read all the 4 symbols on the 4 tracks, is it clear?

So if the tape head is here it is going to read all these 4 symbols the machine is in some particular state and depending on the contents of these 4 cells it will make it move and that move will consist of changing into a new state writing symbols here in this 4 cells and then moving the head either 1 step to the left or one step to the right. So in other words what we are describing is a multi-track Turing machine, right?

We have added an extra capability to our basic Turing machine model the basic Turing machine model had only a single track in its tape but in a multi-track Turing machine we have several tracks in this particular example picture the tape has 4 tracks we are now considering therefore what we call multi-track Turing machine how do we specify the transition function of a multi-track Turing machine.

Remember what such a multi-track Turing machine would do in a single step that it will be in a certain state at a certain time so that is the present state, okay. It will be reading 4 symbols in this particular example on the 4 tracks because the read head is here so let us see it is reading a sigma 1, sigma 2, sigma 3, sigma 4 these are 4 symbols it is reading and supposing the machine is in state q. So it is reading on the 4 tracks sigma 1, sigma 2, sigma 3 and sigma 4.

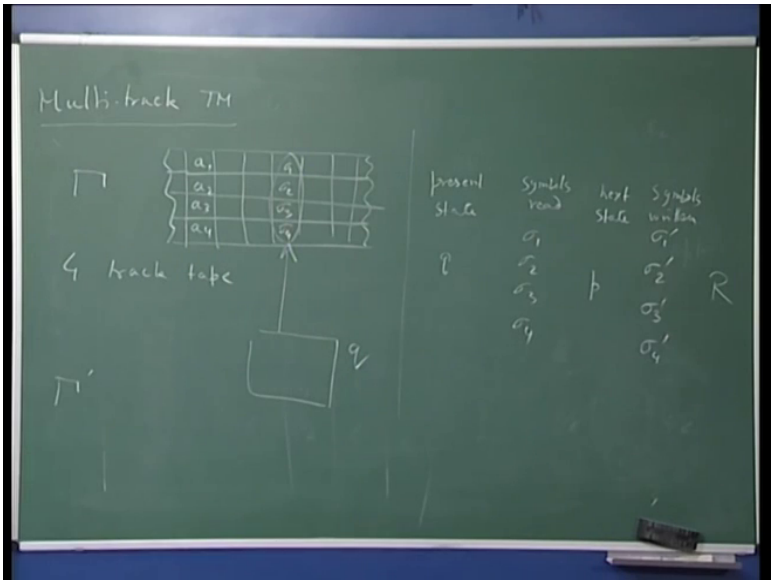
Now what would such a machine do? Its transition function must specify what will be the next state if on track 1 you are reading sigma 1, on track 2 you are reading the symbol sigma 2, on track 3 you are reading the symbol sigma 3 and on track 4 you are reading the symbols sigma 4. So in such a case maybe the next state is p and symbols written it must specify maybe we will just say that sigma 1 this is changed to sigma prime, sigma 2 prime, sigma 3 prime, sigma 4 prime, okay.

So it will obviously the have the makes sense to say that it can simultaneously change all these 4 symbols, so those are the 4 symbols changed and then the move, so this entire this head which reads all the symbols in all 4 tracks simultaneously either will move 1 step to the left or one to the right, so let us say it moves to the in this case for example let us say it moves to the right.

Now this is of course different from our basic model because our basic model there was only one track and so therefore it could read only one symbol at a time, if you you know it is kind of obvious if you think like this we just see this that suppose instead of thinking of all these four symbols being different I mean different symbol of course they may be different but what I mean is thinking of this as a four tuple suppose I think of it as a single entity coding this four tuple.

So let us say what I mean is that supposing I have a code which will code or which will have a single symbol for all possible I mean for every four tuple of symbols of the gamma that is the tape alphabet of the multi-track machine imagine you are defining a gamma prime which is basically you have a single symbol single distinct symbol for every such four tuple, right? So let us instead of four I can illustrate more simply the basic idea by just two tracks and the point is not lost because you can just whatever I will do for two you can do it for four.

(Refer Slide Time: 42:00)



So let us say you had and the your tape alphabet gamma was this let us see 0, 1 and of course we have blank always and that is the you know special symbol. So supposing you had only tracks then on this what all symbols which can occur? You can have 00, 01, 10, 11 then you can have 0 blank, right? Then you can have blank 0, right? The you have blank blank and anything else obviously you would also see 00 is taken care of so that is fine so two blank and similarly you would have you know 1 and blank supposing 1, 1 which is there already so 1 blank, blank 1 and blank blank is taken care of.

So suppose you say you code this as A, right? This as A, this as B, this as C, this as D, this as E, this as F, this as G and this is H and this is I, right? So now imagine if I had a tuple like what this two track machine q you know something like let us say 0 blank, p, blank blank, q to conceptually you can think of to represent this tuple as this 5 tuple quintuple as q for 0 blank we had the symbol E it goes to state p for blank blank we had the symbol G, right? And of course I should say what is the move, right? Here I should say let say move to the right so we move to the right.

Now this way therefore just by changing the alphabet here originally the two track machine had 0, 1 and blank and here I have q, E, p, G, R etcetera. Now this is the quintuple corresponding to that and so I have changed the alphabet set input alphabet not input alphabet the tape alphabet from 0, 1, blank to this. Now what is the new blank symbol or what was what will be blank here you may say G is the blank because you know you would assume a blank means on blanks on both the tracks, right?

So G will act as the special symbol which will be there most of the places and initially maybe the input is written on one track other tracks are blank which is fine I can take care of this way. So what I am saying is that a single step of a multi-track Turing machine like this can be taken care of by a basic Turing machine model machine by extending the alphabet set, right? So here originally you had 0, 1, blank and here we have A, B, C, D, E, F, G also we had something more, right? H and I.

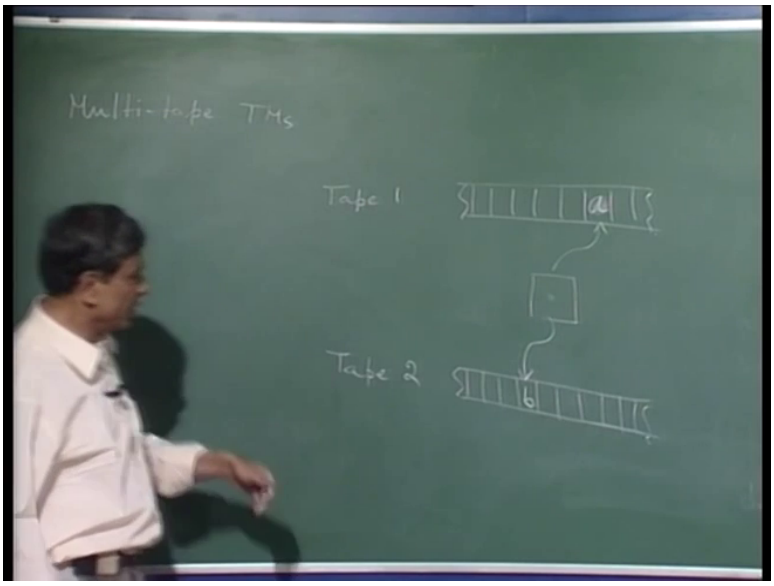
So your original alphabet for the multi-track machine at three symbols and you have here so many other symbols and that takes care of this. Therefore it is now should be fairly simple to convince ourselves that if one could recognize a language by means of a multi-track Turing machine by essentially changing the alphabet set the tape alphabet set I can do whatever the multi-track Turing machine was do by a basic Turing machine model.

So therefore adding the capability of having multiple tracks on the tape does not really add to the recognition capability, in the sense what are the languages which can be recognized that class of languages that class of languages will not change just because from a single track you had multiple-tracks. So this is one possible generalization that one could think of and we have found

that it need not or it will not change the definition of the class of languages recognized by Turing machine.

So now let us make a slightly more interesting generalization and that generalization is that instead of 1 tape whether single track or multi-track that does not matter because multi-track corresponds to I mean I can simulate by a single track that I know, however if we had more than 1 tape then can we do something which with a single tape we cannot do, alright? So what we are going to discuss now is multi-tape Turing machines.

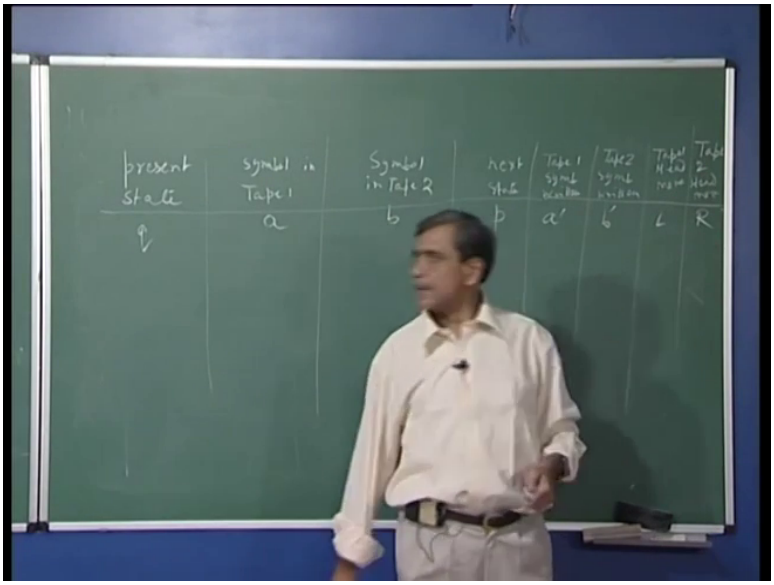
(Refer Slide Time: 48:35)



So multi-tape Turing machines let me make this simplest case for multiple-tape Turing machine that it has two tapes consider a Turing machine which has two tapes there is of course we will assume that each tape is as before two way infinite tape and there is of course 1 you know as usual the finite control of the Turing machine which basically means the Turing machine this black box or whatever you call it can stay in number of states is finite of course.

Now so it is possible since there are two tapes this device must have the capability of reading from both the tapes and changing. So therefore we need to have in this case two read write head your one read write head for tape 1 the other read write head is for tape 2, right? So imagine at some point of time this on tape 1 the cell which the tape 1 read head is reading is A and let us say the tape 2 read write head is reading the symbol B, what happened? What is the transition one step transition of such a Turing machine?

(Refer Slide Time: 50:55)



So again let us see what is the current you know scenario this as we are seeing in this picture let us see the machine is in state q so present state is q and symbol in tape 1 is A, symbol being read in is B. Now you expect that the machine in state q depending on what symbols it sees in tape 1 and what is the symbol it sees in tape 2 it will make a decision about which is this next state it should go to. So it is corresponding move will be you know will be specified by seeing what is the next state which is so let us say next state is p now you have to also say what are the symbols that the machine would write in tape 1 and in tape 2, right? So we say tape 1 symbol written, tape 2 symbol written. So let us say it writes A prime here and writes B prime here and some other symbol so B is changed to B prime in this tape A is changed to A prime in this state.

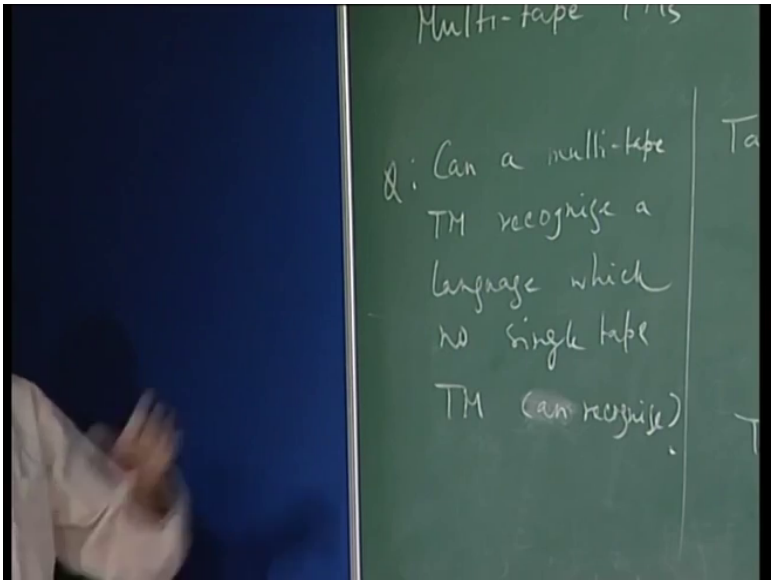
And now the move but you see there are two heads two read write heads and obviously a single move will restrict supposing we just a left then we would mean that both the heads must move to the left or if we just say write you know we may do you want to restrict that both head 1 and head 2 move in the same direction, no since we are talking of generalizing the capability of Turing machine these moves can also be independent in the sense one can move to the left, other can move to the write or this can move to the right, this can move to the left or both can move to the left both can move to the right.

So we will say tape 1 head moves and tape 2 head moves, right? So this maybe tape 1 will move head will move to the left and tape 2 head will move to the right. So you see a move of a two

tape Turing machine to describe it we need such a Tuple, right? So present state what are the symbols which the machine is reading on the two tapes? Depending on these three pieces of information the machine must say what is the next state it will go to, what are the two symbols it would write on the two tapes and what are the directions in which the two heads will move, right?

Now since we have got an interesting generalization can we also like we did for multi-track case can we also take care of such generalization by means of our single tape Turing machine, right? So again please be very careful in what is our objective we have defined another way in another way Turing machine the Turing machine which has two tapes our question is that can there be a language which is recognized by such a machine which cannot be recognized by a single tape basic Turing machine model.

(Refer Slide Time: 56:25)



So that is the question we would like to answer, so let me write the question clearly that can a multi-tape Turing machine recognize a language which no single tape Turing machine can, this can is, can a multi-tape Turing machine recognize a language which no single tape Turing machine can recognize. If the answer is positive to this question then of course it means that we will have a bigger class of languages recognized by multi-tape Turing machines then the class of languages recognized by single tape Turing machine.

What we will prove and again not to you know the proof is not too difficult the answer is no, so again our single tape Turing machine is good enough if you focus our attention to single tape Turing machine that is good enough.