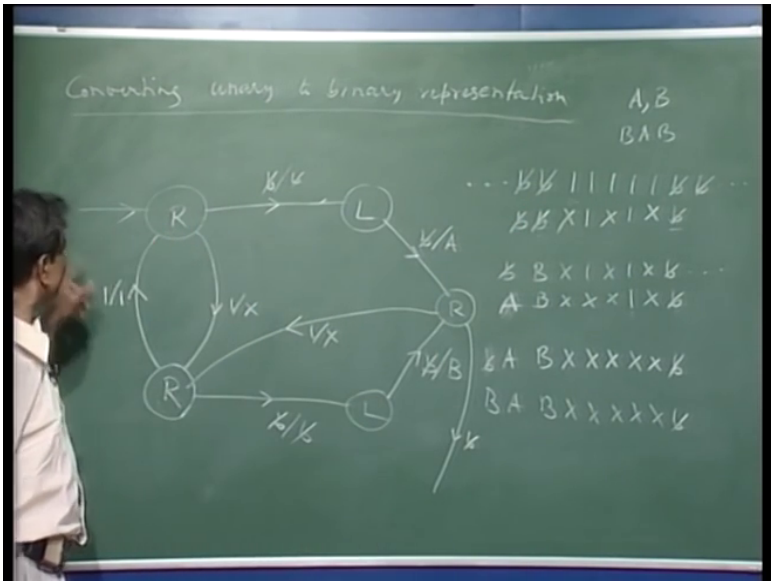
Course on Theory of Computation
By Professor Somenath Biswas
Department of Computer Science and Engineering
Indian Institute of Technology Kanpur
Lecture 36
Module 1
Example continued. Finiteness of TM description, TM configuration, Langauge acceptance, definition of recursively enumerable (r.e.) languages.

Let us now describe the Turing machine which will perform this conversion of unary representation of a number to its binary representation according to the strategy which we had outline.

(Refer Slide Time: 0:37)



So as an example think of this string of 1's which is of course 5 in unary and we would like to convert it to binary, remember that we will use for binary representation the symbols A and B where A would stand for 0 of the normal binary representation and B will stand for 1.

So this is 5 and therefore what we should get here BAB which is 101 which is the binary representation of this unary number. What we will do is we have the convention that our Turing machine will start at the left most 1 of the this input string. So the Turing machine will start here scanning this particular symbol, so let us see how we can carry out the strategy by means of a Turing machine.

So let us say this is the initial state and in this initial state it is scanning this left most 1. Now we said that as soon as this machine sees this 1 it will convert to an x, this 1 it converts to an x and what should it do? It should move to the right, right? Skipping the next one etcetera. So therefore this is a right moving state it comes to, and in this when it sees a 1 what should it do? It should just go back to this state so that it can cross of the next one.

So and as it so remember that initial it was here so this was converted to x, right? And the machine came here and in this when it sees a 1 it just skips that 1, so this 1 is written as 1 and it moves to this state and in this state it should be scanning this particular symbol, so therefore now we know that this also a right moving state. So this 1 is left is left as 1 in this state now it is scanning this particular symbol which will be converted to x it comes here and in this state it is now seeing this 1 so which will remain as 1 because 1 is not being changed, so 1 is being over written by 1 itself and then it moves to the right therefore the head will be here scanning this 1 but this particular 1 will be converted to x, right? And it comes here having converted to x, right?

And then the since it is a right moving state the head will be under this blank, but what does now seeing a blank mean? It means that the input is entirely scanned in this pass and also see what has happened is this 1 was cross and you found a 1 corresponding to this, this 1 was crossed and found the 1 corresponding to this, this 1 was crossed and you did not find a 1 corresponding to this particular 1 which was here. So that means this original length of the string is odd and therefore it should print B to the first blank cell available.

So therefore it keeps on now moving to the left, right? And in this as it moves to the left you know it just keeps on moving to the left, keeps on moving to the left when it reaches a blank symbol here it will convert it to B, why B? Because it knew that the string of 1's that it has considered in this pass is odd and the remainder by division with 2 is 1 and therefore our code of 1 being B it should print that B and so it is here.

So therefore having written this blank this blank by B so the situation will be like this what should it do? Now it should again start this process all over again on the remaining one, which are the remaining ones? The 1's which is the quotient of division by 2 and that is indeed see 5 when you divide by 2 the quotient is 2 and that is what you have here there are two 1's, right? So it will come here and then having written B it should go over this side so therefore it is a right

moving state and in this as it moves to the right it ignores this cross it sees a 1 and this 1 it is converted to x and this process begins again, okay. So now let us go over again, so this 1 is now changed to x having changed to x the machine is in this state this sees an x whatever self-loops we are not writing that means it remains there without changing the symbol.

So this x it skips this 1 takes the machine to the top state without converting that 1 into anything that 1 remains as 1 this 1 is here and this comes here in this right moving state and now it will again keep moving to the right sees an x which of course it ignores, now it sees a blank the head will be here in this state and blank means again that the current 1's we have already taken care of and in this case you see what had happened was this iteration what had happened this 1 we converted to x this 1 we could match it off to this and which remained 1 and now it is here seeing a blank which means that the number of 1's in this part left was even so it now should print a A and therefore it is similar to this except in this part what it remembers is that it should print an A whenever it sees a blank.

So in corresponding to this there is a left moving state and so this is actually as it is from here is moving to the right and when it keeps on moving to the left till it finds a blank which it changes to A, okay so let us go over at again what is going to happen this it will keep on after seeing this blank it would have come to this state and it will start moving to the left and skip all 1's all x's this B and when it sees this blank it will change it to A and then moves to the right, in this right what it should do? It will keep on going to the right till it sees this 1.

So these things do not change this 1 is converted to x, x blank now it sees so it is here, right? And what it finds that it has reached a blank. So now on blank in these cases it would go the left but now remembering the string of 1's that it had considered in this iteration was odd and therefore a B needs to be printed. So it will now again go over to the left and here this blank would be converted in the next step to B and as the machine converts the blank to B, so where is his machine now?

This machine was here it converted this blank to B and it is now in the right moving state and it will now it is not going to see anymore 1's because all the 1's have been crossed off so it will keep on moving to the right and what will happen it is going to see a blank the moment it sees a blank in this state, what does it mean? That means you know that there was nothing more to do

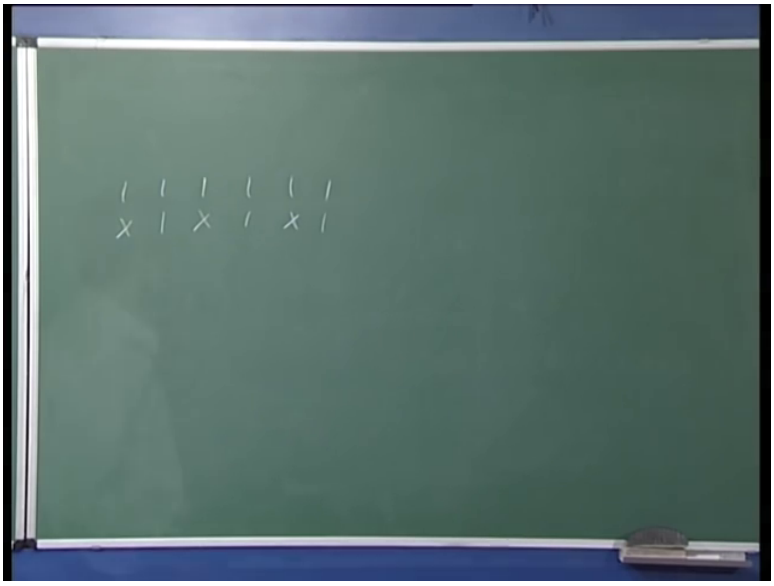
because all the 1's now have been converted to x's and that is why you are not seeing any 1 in the current space on which you need to work and since you found that all 1's have been taken care of therefore the job of conversion is over, okay.

So let us see this once more and remember here we are talking of a Turing machine where we have used our old convention that any transition which is not shown means the machine remains in that particular state without changing the symbol I wrote here blank but more technically or to be consistent with what we are writing I should have said blank is replaced by blank.

So let us go over once more very quickly this transition diagram so remember that the machine would start at the left most 1 of a block of 1's which are flanked on both sides by blanks the machine would start here and this state why this is the state how do we know from this transition diagram that this is the initial state because there an arrow coming from nowhere into this state which indicates this is the state in which we start the operation.

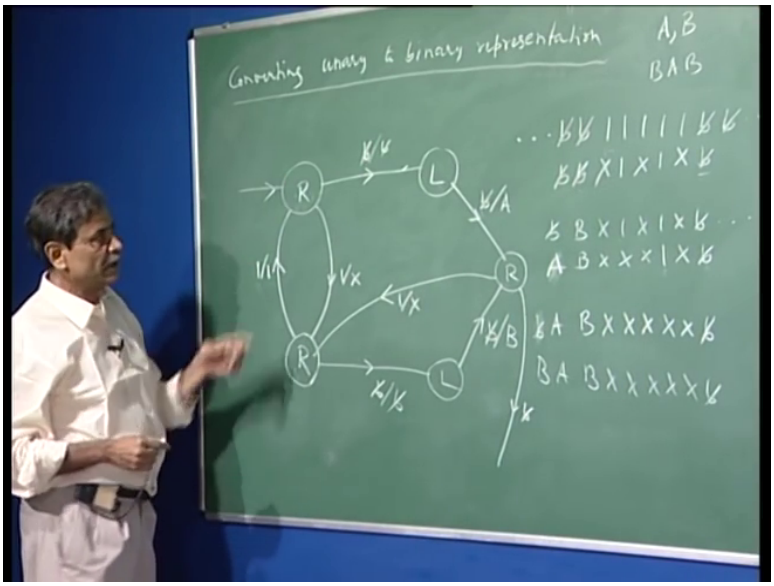
So in this state if you see 1 you convert it to x go to this right moving state that means the machine moves to the right and when it sees another 1 it does not bother to change it because this is the 1 which corresponding to the 1 which you had cross off. So 1 converts gets converted to x next 1 is not converted to anything it remains 1 but it means that we have corresponding to this 1 we have found a 1 so we are back in this particular state.

(Refer Slide Time: 14:01)



So this alternately so this we know that given a block of 1's what we are doing is we are putting it crossing it to x so first 1 next 1 we are leaving it as such crossing it to 1 next 1 as such crossing it to 1 leaving it as such but now when we see a blank that means that input space is over and depending on whether the machine is here or here it knows whether the input space had once which is whose length is either odd or even in this case it is even in this case is odd. So this is what is going on and here those transitions which we have not shown.

(Refer Slide Time: 14:50)

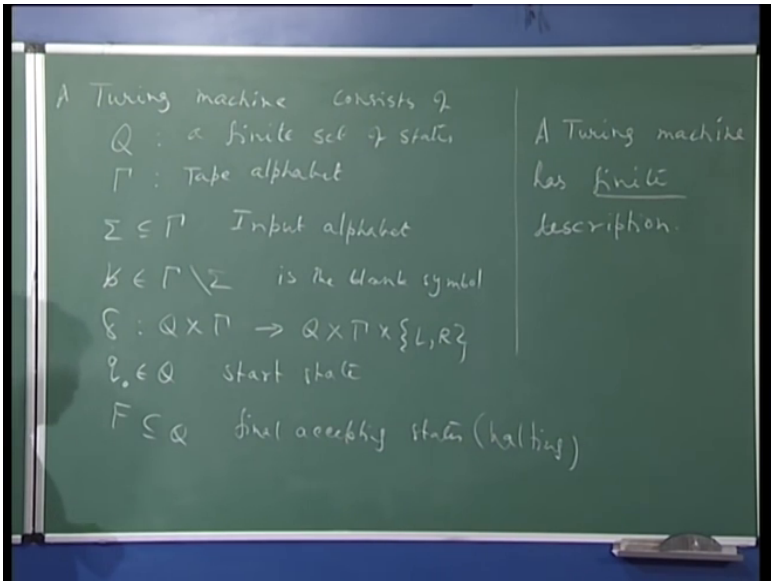


So for example in this if there was an x in this state if you see an x, what does it mean? It means the machine remains here and moves to the right that is the convention we have made use of. Notice that in this machine we have made a transformation of unary to binary and something again you do not know how it is it could have been done at all by finite state machine. So this is something extra that we can see that Turing machines can do which we could not do by finite state or by PDA, alright.

We have seen now several at least two examples of Turing machines both doing fairly interesting tasks the first example was checking whether a string of a's, b's and c's are equal number of a's, b's and c's and the second one was converting unary representation number into its binary representation but in this particular course of theory of computation our focus is not so much you know computing functions but our focus has been to recognize languages how do we recognize

languages or what is really what does it mean to say Turing machine recognizes a particular language.

(Refer Slide Time: 16:52)



As we has seen several examples of Turing machines now we know what are the components or what go define a Turing machine like if you look at this Turing machine first of all it has a finite number of states we can say a Turing machine consist of Q of finite set of states, then it uses tape alphabet so that is all the this is the set of all symbols which can appear on a particular Turing machine tape. Now a subset of this sigma which is a subset of this gamma is the input alphabet.

Then one particular symbol we made a special and that was the blank symbol you can say blank which is an element of tape symbol but which was not in the input is the blank symbol and the importance of the blank symbol is that this is the symbol which most of the cells in a Turing machine tape will have then of course we had a transition function and this transition function in this example was graphically or pictorially represented, but what is a transition function for deterministic Turing machine? It takes a state and a some symbol which appears on the tape so this is the present state and the symbol that is being scanned and corresponding to which there will be a next state and the symbol that will overwrite and the move.

So therefore transition function is a mapping which is from the set of states crossed with the set of tape symbols and its range is that means given a present state and the present symbol being scanned this transition function should tell what is the next state and therefore this component it

is Q what is the symbol being written therefore this we write and what is the move, so one of left or right.

Then you have of course a special state which is the start state, so q0 which is an element of Q is the start state of the Turing machine and in the context of language recognition we should have some final accepting states, so some F which is a subset of Q you know final accepting state and in these states once the machine if the machine at all reaches one of these states then the machine halts that is the convention we have so final accepting states are all halting.

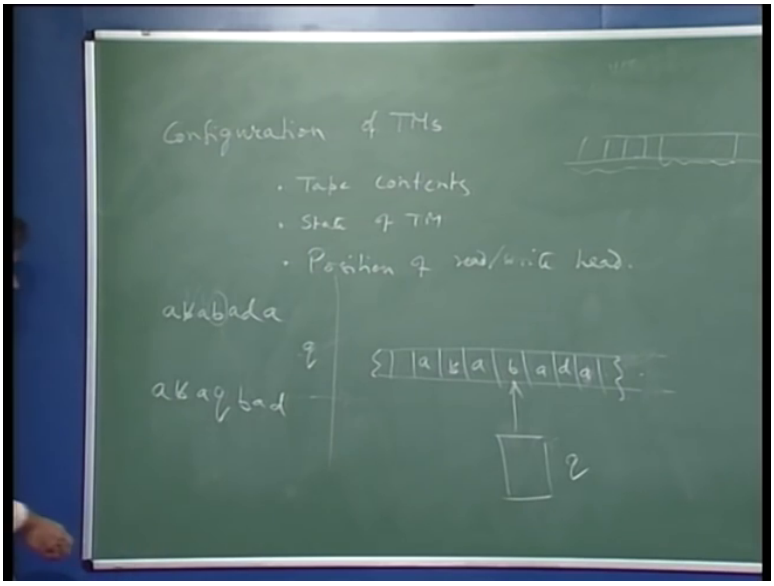
So if you specify all these then you specify a particular Turing machine different Turing machines of course can have different Q, gamma, sigma, blank symbol maybe different, transition functions will be different, etcetera. However notice that this is finite, this is also finite, this is also being a subset of gamma this is finite this is one particular symbol this is a map from Q cross gamma both are finite, so therefore and it is a deterministic machine we have said so corresponding to each one there will be one such component one Q, one symbol will tell me what is the next state symbol being written the direction of the move.

So point is even this is a finite function everything here is finite. So therefore the description of a Turing machine whether you use a diagram or whether you use these symbols and then elaborate each one separately whatever it is a Turing machine can be described finitely and that is a very important point, a Turing machine has a finite description. Now you must realize that this is what is expected, why? Because what we are trying is our overall motivation is to capture the intuitive notion of an algorithm by Turing machines no there can of course be many many infinitely many algorithms there can be infinitely many Turing machines that is fine but an algorithm is a finite object, is it?

Supposing I code an algorithm in my favorite language programming language whether it is C or java in that is a text and that text is something finite. So my every program is a finite you know I can think of a program as a finite text so it is something like a finite you know entity or an object. So if I am saying that our algorithms can be captured by Turing machines then to retain that correspondence since algorithms are finite things or entities Turing machines also must be finite entities and indeed these are as you can see a particular Turing machine is represented in

terms of these things and these are all finite things and therefore this particular Turing machine is also a finite entity.

(Refer Slide Time: 24:33)



Now let us consider what we mean by recognition language recognition by deterministic Turing machine and the notion we shall use here is the configuration of Turing machines we have spoken of configuration at least several times before when we talked off PDA for example we had a notion of configuration and configuration is kind of snapshot which tells me at what point the computing device is and this snapshot should have enough information that from that point onwards if I would like to carry out the computation by the computing device the configuration should have all the information to tell me to how to carry out that further those further steps of the computation.

Configuration of a Turing machine at least one tape Turing machine the kinds of Turing machines that we are talking of should have then what? It must tell me what are the symbols which are there now on the input and there order in which they are there that is the configuration must have that tape contents and it should also tell me the state of the machine and it should tell me where the head of the Turing machine is positioned currently.

So again remember that configuration should give me a snapshot which has complete information about the current affairs of that running of a particular Turing machine. So at any given time a Turing machine tape has some symbols and the read write head is positioned to be

reading a symbol on the tape and the machine is in some state q. The configuration of the Turing machine must consist of all these information, however tape is an infinite thing, is it? We said it extends to infinity in both directions.

Then how do I represent this infinite thing finitely, fortunately remember that if begin with we had only finitely many symbols on the tape initially and after a number of steps non blank cells will still be finite will be bounded because why because you see it is like this that suppose initially this much was the space of the tape which had which contained all the non-blank symbols you can imagine the input.

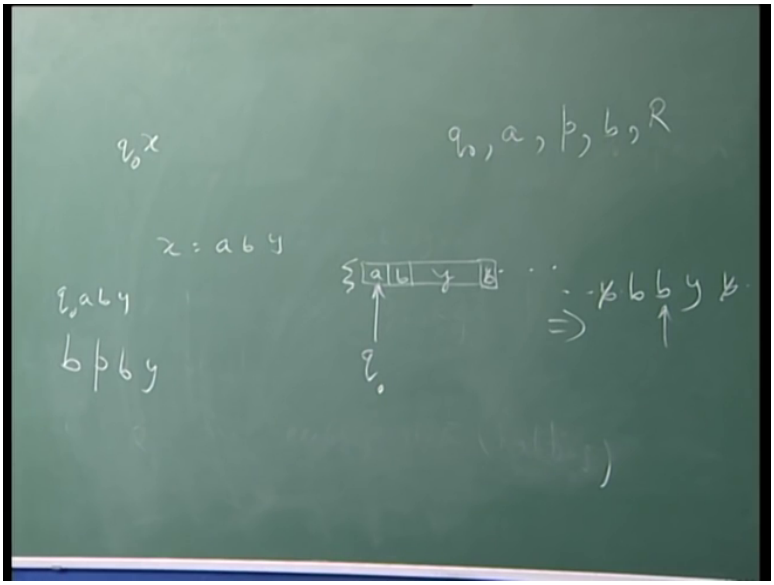
And then the machine after n steps may have moved to all these area in n steps it would have at best converted n more blank cells into non-blank to contain non-blank symbols. So after n steps the number of cells which would contain all the non-blank symbols will be the original ones plus n at most that is the bound. So in providing tape contents we do not have to of course say give all the blank cells all we do is so let us take this picture and let us say I had here a, blank, a, b, a, d, blank or let us say a and all these cells are blank.

So I would represent this situation by saying a blank a b a d a and that is all I write with the understanding all the symbols to the left of this and all the symbols to the right of this all these symbols are blank, right? So now the machine is in machine is scanning this particular b in state q in this picture, right? A convenient way of writing all this in one single line is to do this we will write a blank a and here we write this the state and then b a d.

So this is the way we will represent configuration so couple of things again to say here that this is a string which will contain symbols from the tape alphabet including blank like here we have one blank but the idea is the current tape contents is this and everything to the right are blanks everything to the left are blanks and now you see there is a state symbol appearing the idea is that this the machine is in this particular state scanning the symbol immediately to the right, so it is scanning this b, alright?

And if you notice that this very conveniently represents whatever information that particular picture had, alright?

(Refer Slide Time: 31:45)



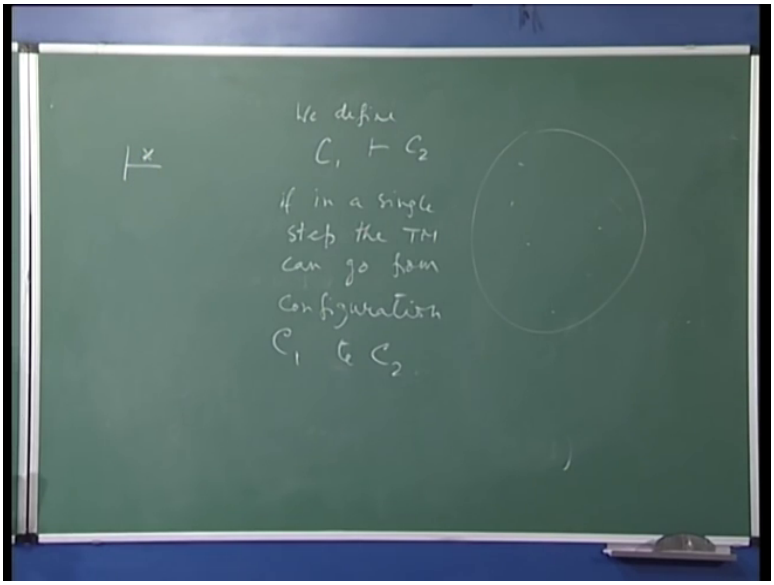
So now what we can say that initially you know we always say that initially the Turing machine is scanning the left symbol of the input that is a kind of convention we all have all the time. So what is the initial configuration? Initial configuration will be the input let us say x some string and the machine is going to be scanning the left most symbol of this input and that situation is very conveniently represented by this configuration and notice our so we are saying that the machine is in state q0 scanning the left most symbol of x which is our input.

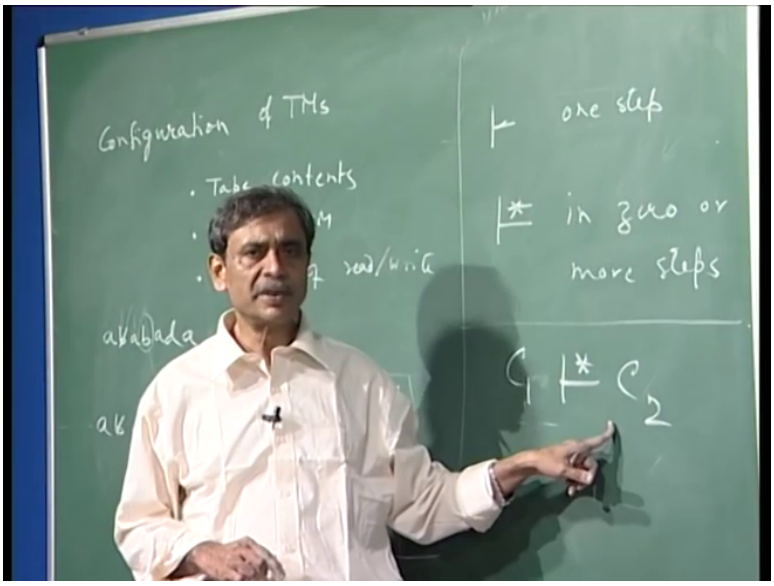
Now let us elaborate it a little bit so let us say x is a, b and y and suppose our Turing machine is such that in q0 if the input is symbol or not the input symbol the symbol which is being scanned is a then it goes to the state p writing b and moves to the right. So therefore what would happen initially in picture we may write like this these are all blanks here a then b and then this whole thing is y and then these are all blanks initially the machine is here in state q0, and this in one step what will be the situation? Of course these will be blanks and this a is converted to b so and of course nothing else changes.

So this a has become b there was a b here so that b is here and then the string y and then of course blank and so on and the head will be since the move is to the right the head will be here. So in terms of configuration what is the situation? Initially my configuration was q0, a, b, y and in one step the configuration became b p b p b y, right? So from this configuration we are getting

this particular configuration, why? Because the Turing machine had carried out one step of its activity.

(Refer Slide Time: 34:48)



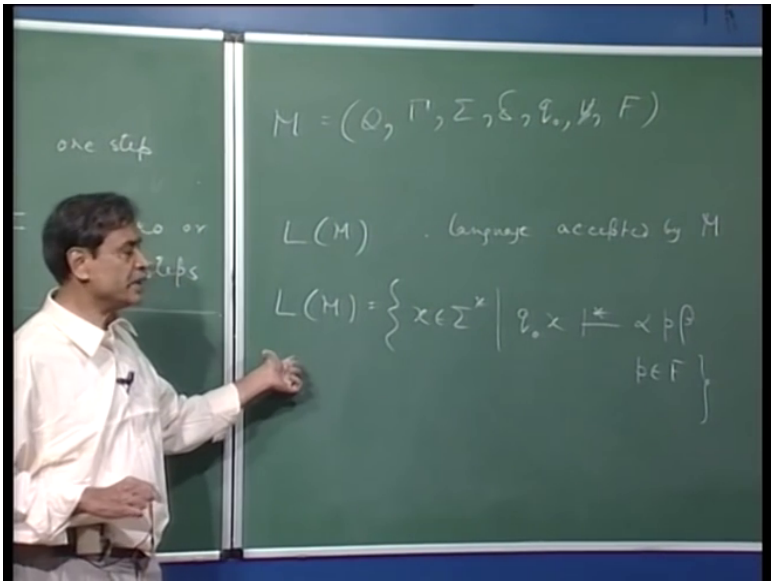


So what we can see that if we consider that (())(34:49) there is a space of configurations of a Turing machine there is a relation amongst configuration and what is that relation? Supposing there is configuration C 1 and the configuration C 2 we say C 1 derived C 2 in one step or C 1 is related by this relation if the Turing machine in one step can go from C 1 to C 2 so let us see let us this way put it we define C 1 you know then this relation symbol to C 2 if in a single step the Turing machine can go from configuration C 1 to C 2.

So therefore this is a binary relation relating configurations and of course you can define another relation which we can say so if you remember we are saying that this relations stands for machine so machine one step situation in a single step from configuration C 1 the machine goes to configuration C 2 this we will say in 0 or more steps one way of relating this to is that you know this particular relation this is again a relation on configurations and this particular relation is the reflexive and transitive closure of this relation even without going into those kinds of mathematical jargon intuitively the idea is very simple we will write C 1 goes to C 2 we will express like this for a Turing machine M in case the configuration C 1 from starting from configuration C 1 in 0 or more steps you can go to the configuration C 2, alright? Why are we doing all this? You are doing all this because very precisely we would like to capture the notion of accepting or recognizing a language by a Turing.

We are trying to understand very formally and very rigorously what does it mean to say a Turing machine recognizes a language and towards that we defined a relation on configurations and this relation we will relate to configurations provided from the first you can go to the second configuration in 0 or more steps using that Turing machine which you have in hand.

(Refer Slide Time: 38:38)



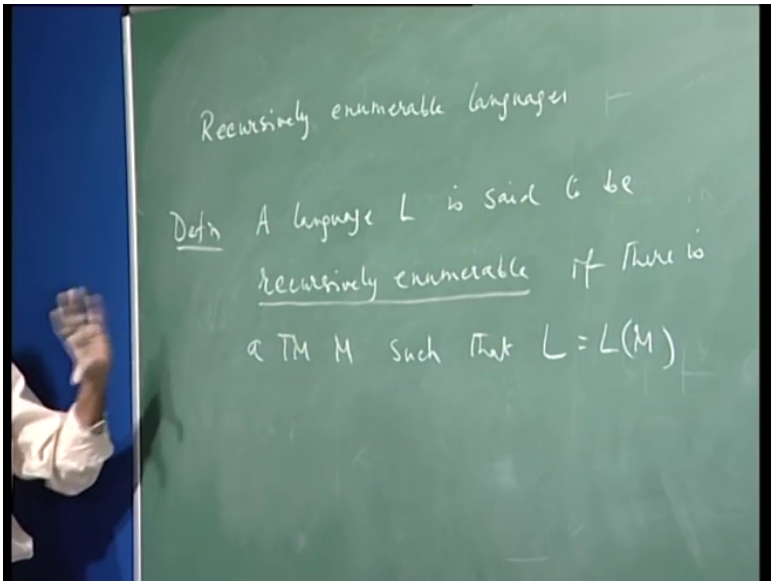
So to begin with a Turing machine M remember it has a number of states it tape uses an alphabet which we are calling gamma let us say the input again is over an alphabet sigma, then it has a transition function delta and it has an initial state q0 is a special symbol which is blank symbol

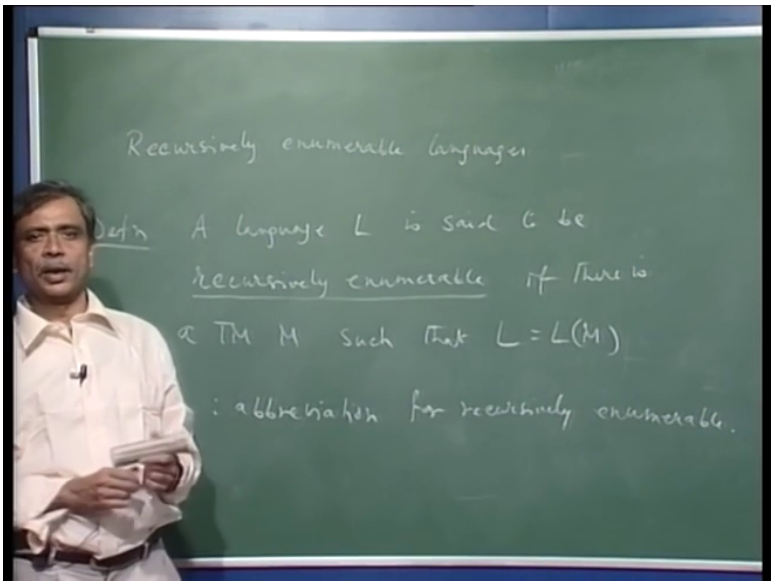
let us write it as this and then let us mention the states which are final accepting states so these are this is final accepting states.

So once more this is the set of states of the Turing machine this is the input alphabet or rather tape alphabet is the input alphabet is the transition function, the initial state, the special blank symbol and the final accepting state. And L M is the language accepted by Turing machine M this Turing machine so this stands for language accepted by M and using our terminology now we can very precisely say that this L M is this set of all strings x over alphabet input alphabet sigma such that q0 x which is a configuration which really stands for configuration in which the every part of the tape is blank except where the input x is written the machine is in its initial state q0 scanning the left most symbol of the input and from this configuration if in 0 or more steps you can reach a configuration where you know some alpha is here and maybe some p and beta where p is an element of F, okay.

So what we are saying is scanning the left most symbol of the input initially in the state q0 if in 0 or more steps the machine if it reaches a final accepting state in that case the original x is in the language accepted by the Turing machine, right? Now right away let me use another terminology the set of all languages for which we have a Turing machine recognizer Turing machine which can accept a language such a language is called recursively enumerable language.

(Refer Slide Time: 42:21)





So let me since this is an important concept let me note it down, so this is a definition a language L is said to be recursively enumerable if there is a Turing machine M such that this language L is the language accepted by the Turing machine M, right? So recursively enumerable languages are those languages which are accepted or you know we also use the term recognize accepted by Turing machine and this is the class of languages which will consider you know extensively in next classes but we already know from the first example that the language of equal number of a's is followed by equal number of b's and equal number of c's that means this language is a recursively enumerable language because we gave the Turing machine which recognize precisely this language.

So therefore this particular language which we know is not context free is a recursively enumerable language. So our goal will be to study this class of recursively enumerable languages and see which all languages at least intuitively which all languages are recursively enumerable and if there are languages which are not even recursively enumerable that will be our concern. Before we end let me say that we do not always write this whole thing recursively enumerable language we just simple write r e so this is an abbreviation for recursively enumerable.