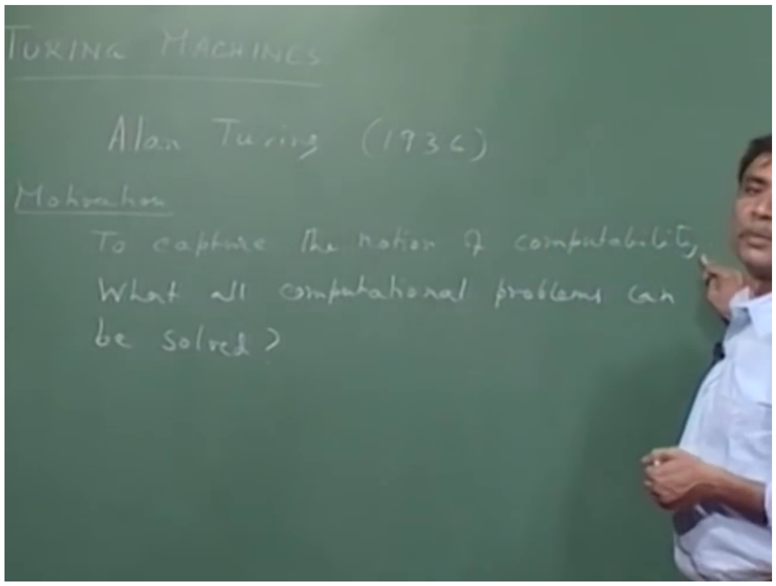
Course on Theory of Computation
By Professor Somenath Biswas
Department of Computer Science and Engineering
Indian Institute of Technology Kanpur
Lecture 34
Module 1
Turing machine (TM): Motivation, Informal definition, example, trasnistion diagram

From this lecture onwards we will spend some time on this topic Turing machines. Turing machines the concept was defined by a British logician, mathematician, computer scientist whose name is Alan Turing and the definition came in a very very famous paper that Turing wrote in 1936. Before we get into the details of what Turing machines are how to you know define things with or work with Turing machines we should have some understanding we should have some appreciation why did Turing define in 1936 what we now call Turing machines.

(Refer Slide Time: 1:24)



TURING MACHINES

Alan Turing (1936)

Motivation

To capture the notion of computability,
What all computational problems can
be solved?

Now the real motivation is to capture the notion of computability, what we mean by to capture the notion of computability? You know today of course we are very familiar with computers, programming but at the same time sometimes we ask that can all computational problems be solved by our computers maybe something that I cannot do today something if at all there is something which I let us say cannot do today maybe in future when we have you know some other kind of digital computers or some other kinds of computers, not electronic computers as we know today but maybe quantum computers, maybe DNA computers can we then do something can we write certain algorithm which could not have been possible to write by using today's computers.

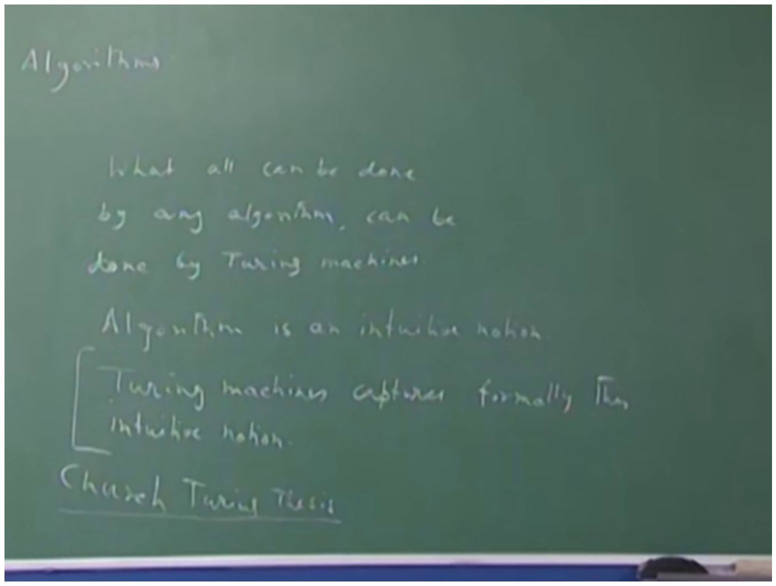
So that means what all things can be done by computers, so when I say to capture the notion of computability essentially what I mean is I would like to answer this questions this particular question, what all computational problems can be solved this is the question we are trying to even today this a question that we have and way back in 1936 before digital computers where even there Alan Turing defined this Turing machines and that today we understand that it totally captures the notion of computability.

Now if we understand what cannot be done by Turing machines and what can be done by Turing machines and then if it is indeed true that what Turing machines can do is precisely what any computer ever can do, then just by studying Turing machines we will know the power as well as limitations of computability, alright? So now we can see therefore this notion is of course very important but before again we discuss the details of Turing machines one question arises that how is it possible that we define something once for all in fact it was defined way back in 1936 and the claim that we are making is that whatever be computers in future or today they cannot do something Turing machines cannot do and whatever they can do Turing machines can do.

So that means we are capturing once for all the notion of computability, how is at all possible that today we define something and we are saying that look this definitions stands forever so far as a concept like computability is concerned, alright? So I am just trying to tell you how not just important how even difficult such a notion is because once for all we are trying to capture a notion like computability whereas we know that computers change every day and yet we are trying to say that look what Turing machines can or cannot do that precisely defines what computers can do, alright?

Of course we know that computational problems are solved by algorithms, so another way of saying that if at all there exist an algorithm to solve a problem that algorithm can be implanted on Turing machine.

(Refer Slide Time: 6:40)



Algorithms

What all can be done by any algorithm, can be done by Turing machines

Algorithm is an intuitive notion.

Turing machines captures formally this intuitive notion.

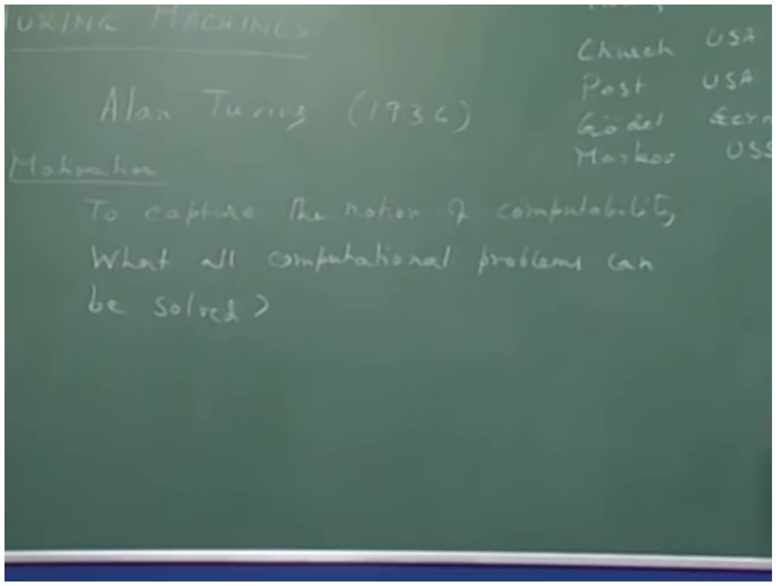
Church Turing Thesis

Now can one give proof of this statement, what is that statement? What all can be done by any algorithm can be done by Turing machines. Now if you notice supposing you say that I want a proof for the statement, what does it mean? Then is of course one needs to know that since proof is something formal then one must understand formally what an algorithm is but algorithm as a notion is an intuitive notion, why do I say so? Because much before anything like digital computers where there to implement algorithms we of course had algorithms, right?

For example in way back in 300 BC you Euclid provided his famous Eulid's algorithm to compute GCD of two numbers, there is no question of any computers to implement that algorithm then but mathematicians down the ages they have intuitively understood the notion of algorithms. So point I am making is that algorithm is an intuitive notion and what Turing machine does is to capture this intuitive notion formally, so let me say this Turing machines captures formally this intuitive notion.

Now really speaking that this statement cannot be proved because as we shall see Turing machine is of course is a formal concept in the sense that you can define it with clarity in fact we will be making use of Turing machines to define many algorithms but on the other hand this intuitive notion of algorithms as we said it is an intuitive notion. So this kind of statement is saying that this formal notion captures this intuitive notion, right? So really speaking we cannot have a proof for this what we have is a thesis, so this thesis is called Church Turing thesis.

(Refer Slide Time: 10:35)



URING MACHINES

Alan Turing (1936)

Motivation

To capture the notion of computability,
What all computational problems can
be solved?

Church USA
Post USA
Gödel Ger
Markov USS

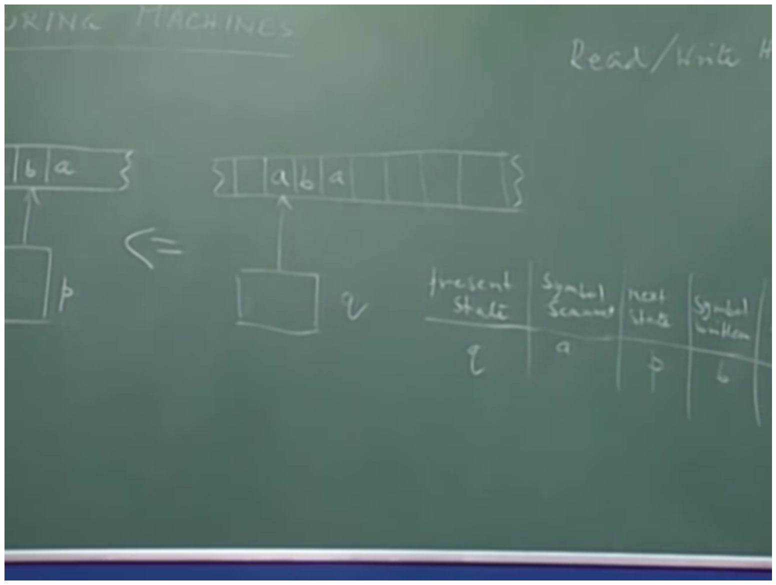
So let me write it here what is Church Turing thesis, what can be done algorithmically can be done by Turing machines and vice versa that is other way is that what cannot be done by Turing cannot be done by any algorithm whatsoever. So this is the so called Church Turing thesis, and very briefly I will just tell you why people have faith in this thesis? It is because at around the same time at around 1936 there were many other not many other let us say 5, 6 other people in different countries, for example Turing himself was in UK, Church in USA, Post in USA, Gödel in Germany and the Markov in USSR they are independently tried to define formally this notion of algorithm that is they were trying to say give a formal definition of what can be performed computationally.

And they all actually came up with different model, Turing came up with Turing machines, Church came up with Lambda calculus, Post came up with a certain rewriting system, Gödel defined what is known as general recursive functions, Markov defined Markov process they all look very different and yet when it you know when people try to look at all these things simultaneously they found that the class of problems which can be solved by Turing machines is precisely what can be done by Lambda calculus, precisely what can be done by Post system or Gödel's general recursive functions or my Markov process.

So that (())(13:44) you know some kind of faith that this must be correct and since then of course there have been some attempts by many philosophers of science, logicians to very very elaborately examine and very critically see that is it is this thesis correct in the sense that maybe that algorithms are more than what you know there are algorithms which could not be performed by Turing machines or vice versa but you know all these attempts have shown that no, indeed whatever you can think of to be reasonable definition of an algorithm you see that it can be performed by Turing machines.

So with this little background I would like to go into the details of what Turing machines are how we make use of them, alright.

(Refer Slide Time: 15:08)



Turing machines let me draw a diagram trying to describe what a Turing machine is first of all what we have is a tape and we are already familiar with the notion of a tape for example we have seen finite state machines the input would be on the tape as well as for push down automata the input would be on the tape.

So this is a tape that Turing machines uses so there is a difference between the tape finite state machines use and Turing machine use you see this jagged lines on two ends they indicate that you should imagine this tape to be continuing to infinity in both directions, once more that this tape continue to infinity in this direction as well as in this direction and remember as before a tape consists of cells this is a cell, this is a cell, this is a cell and the cell can take on a cell you can have a symbol let us say a, on this cell you can have another symbol and so on.

Now this is you can say is the tape of a Turing machine and there is this looks very much like finite state machine diagram that you recall this is the if you like to call the control component of the Turing machine and it has it can be in a number of states, so let us say at any particular time it is in state q and this is its head this is the head of the Turing machine which is at this time scanning this symbol a.

Now in case of Turing machine this head not only can read the symbol which it is scanning but it can also change the symbol, so therefore this head that we have we would like to call it a read write head recall that finite state machines only had a head for reading a symbol from the tape but Turing machines head reads the symbol and it can change that means it can write another symbol maybe the same symbol.

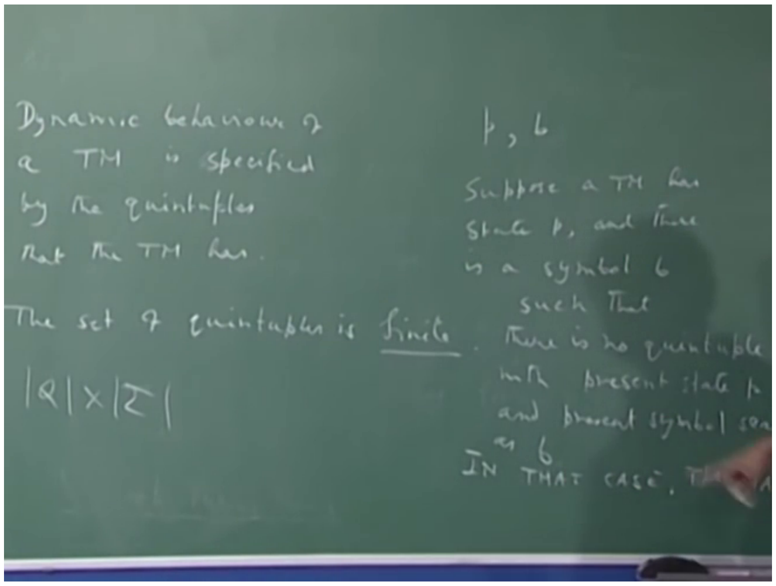
Now how does the computation or how does the working of Turing machine to be described. So here to begin with let us say the machine is in some state q and scanning this symbol a so we can say present state is q symbol can in this case is a and now there will be a next state symbol written and move. So let me just write this first, so suppose the present state is q and the symbol scanned is a then suppose here we have p, here we have b and here we have R.

So you see there are 5 things q, a, p, b, R what we mean is that whenever this particular Turing machine is in state q scanning the symbol a its next state is going to be p, right? And it will write the symbol b in place of a so this symbol a is overwritten by this symbol b and then this read write head will move right by exactly 1 cell.

So from here if this description is to be followed the picture would be let us see that we are following this because at this time it is in state q scanning symbol a and next state is indeed p now it has recall it has what here the symbol was a and now this symbol is b but the read write head has moved in the next instant 1 step to right, so this is called the move, okay. Let us now be clear about what is happening I will repeat that at any given time a Turing machine is in a certain state scanning a certain symbol on the tape and depending on the present state and the symbol that is being scanned at that time it moves to a next state in the next time instant having overwritten the symbol that was there by this and the it will make a move only by 1 cell either to the left or to the right in this case if this is the one it is following then the move is by one step to the right.

Now is now you can see therefore I can describe completely the dynamic behavior of a Turing machine by listing down these kinds of quintuplets let me say what is it why we call it a quintuplet because there are 5 things here present state, symbol scan, next state, symbol return and the direction of the move, right?

(Refer Slide Time: 22:40)



Dynamic behaviour of a TM is specified by the quintuples that the TM has.

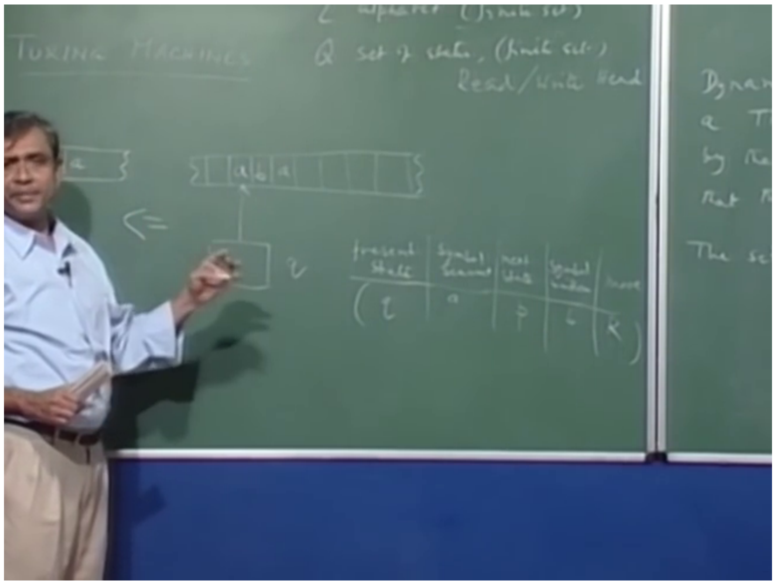
The set of quintuples is finite.

$|Q| \times |\Sigma|$

p, b

Suppose a TM has state p, and there is a symbol b such that there is no quintuple with present state p and present symbol sca as b

IN THAT CASE, T



So dynamic behavior of a you know I will keep writing this TM abbreviation for Turing machine dynamic behavior of a TM is specifies by the quintuples that the TM has, okay.

So remember different Turing machines can have the set of quintuples different but one particular Turing machine we associate a set of quintuples with that Turing machine and that will tell me if I am running that Turing machine what how will the Turing machine behave on a tape when something is written. Though by the way this set of quintuples for a particular Turing machine it is easy to see that this set is finite, why? Alright, because as before we assumed that the alphabet that we work with is a finite alphabet, right? Sigma is the alphabet which is a finite set of course as before.

And the important point is the number of states any particular Turing machine will have is also finite, so Q is the set of states which is also a finite set and now we are not talking of non-determinism if you think currently only in terms of deterministic Turing machines determinism means that there is a definite move at each given time. So how many quintuples can be there if the particular Turing machine as the set of states Q and the alphabet it works on is sigma clearly Q cross sigma not what is mean is if there are 10 states the machine had and 5 symbols and the total number of number of quintuples would be what, at most 50 why I say at most I will come to that because maybe that on certain state and for a certain symbol no quintuple is defined.

So in that case what happens? Supposing we had a state p and a symbol b but those no quintuple with state present state p and symbol scan b, so let us write it suppose a Turing machine has state p and there is a symbol b such that there is no quintuple with present state p and present symbol can as b supposing this is the case that means what that the machine if it ever reaches the state p and scanning the symbol b there is nothing defined, in that case of course we say the machine halts in that case we say TM halts.

Let us once more recapitulate whatever we are saying, very briefly the Turing machine will have a two way infinite tape the tape going to infinity in both direction the tape is divided in cells each cell can contain one symbol from a finite alphabet sigma the machine can be at any given time in one particular state of a finite set of states which we are calling Q and what happens when the present state is Q and the symbol being scanned is a is that some next state will be defined what symbol to be overwritten in this place that will also given and the direction of the move you remember that Turing machine move is always by one square either to the left or to the right.

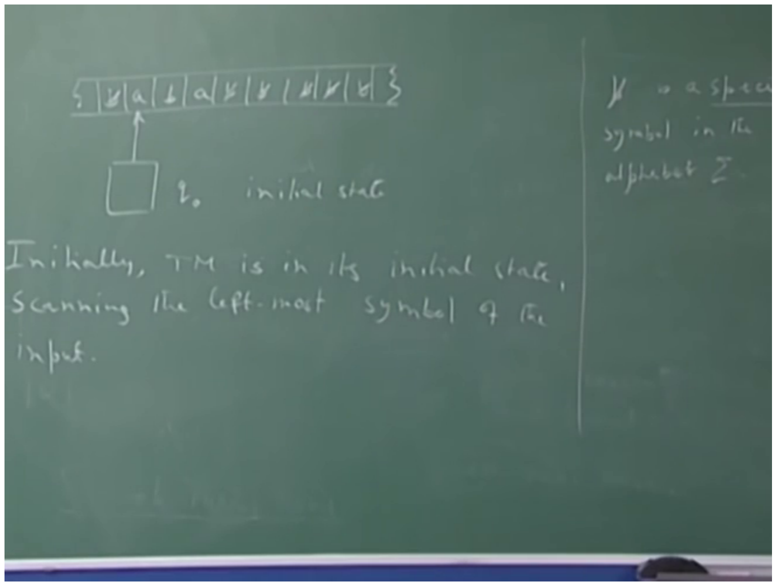
So depending on present state depending on the symbol being scanned some next state is defined some symbol what is to be overwritten is defined and the direction of the move is defined whether left or right and such a thing we call a quintuple and clearly the set of quintuples which we associate with a Turing machine defines the dynamic behavior that means what happens from you know instant to instant even that the machine is at a certain instant you know given doing something basically it is in a state and scanning a symbol how it is behavior will evolve over time that we can say, alright?

We also mentioned something about halting and that is actually a very simple way of defining halting that when will this whole process stop if at all ever it will stop is we can say that it will halt the machines you know this scanning, overwriting, going to a next state and moving this kinds of behavior will sees in case it is in a particular state scanning a particular symbol for which no corresponding quintuple is defined, so that is what we said here about that if the Turing machine finds itself in a state scanning a particular symbol for which there is no quintuple is defined in that case of course you know there is no defined activity and therefore the machine we can say as halted because it cannot do anything further and that is that is the definition of Turing machine.

You know we have to be a little more careful because we have said what happens from one instant to another where how did this whole thing begin, in other words initially what is the situation. We have a notion of initial state which you have seen in case of finite state machines, right? Initial we say the machine is initially in its initial state and also in case of finite state machine we said initially the machine is scanning the left most symbol of the input (())(31:37) case of finite state machine in case of PDA we had one more besides this of course this things hold but we also had a stack and we also had to say whatever the initial stack contents will be.

Now all those things we need to specify should say what is the case with Turing machines initially.

(Refer Slide Time: 32:05)



Initially, TM is in its initial state, scanning the left-most symbol of the input.

b̸ is a spec
symbol in the
alphabet Σ.

So initially we can we will assume that Turing machine is scanning the left most symbol of its input which is written on this tape somewhere, so let us say input is ABA initially the Turing machine is scanning this symbol in a state called which we are denoting as q0 basically this is the initial state of a particular Turing machine.

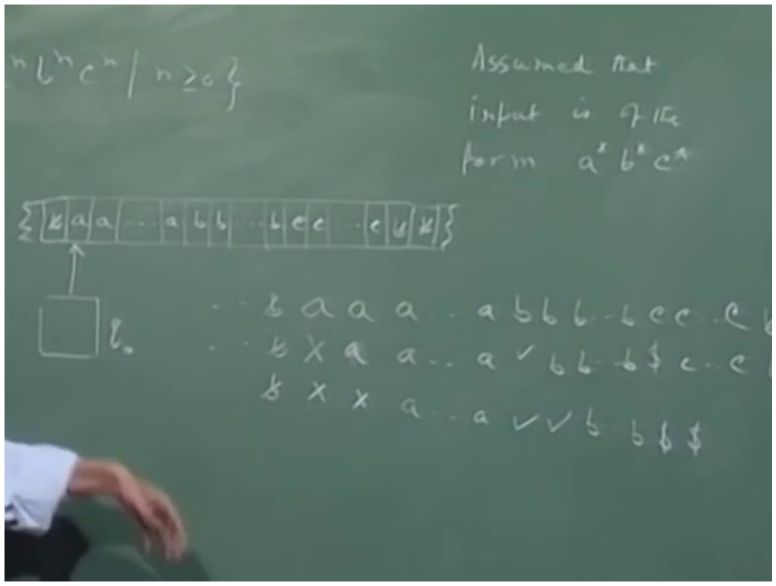
So initially that time 0 if you wish the machine is in its initial state to initially Turing machine is in its initial state scanning the left most symbol of the input and remember that input has to be always some finite string so in this trivial example the input is a string of three symbol it is scanning the left most thing, but what about the rest of the cells? Right? What is that initially in all those cells which are not being used by the input.

So what do we expect and that is precisely what happens in our definition also that these are all blank, right? Imagine it is like a copy book when you start working on a copy in the beginning the copy book is maybe some problem is given so that you know that is your input rest of the copy book is empty and that is what we mean by saying all the cells initially which are not occupied by the input string they are blank.

Now of course what is blank? Blank is a special symbol so blank I denote like this the symbol for blank is this, this is what is convention blank is a special symbol in the alphabet sigma, why is it special? Because this blank symbol is what most of the input will have, right? Initially the, sorry the blank symbol is what most of the tape will have actually all the time but to begin with at least initially we can see the input is some finite string which is there the machine is scanning the left most symbol of the input and rest of the cells contain the special symbol called black, alright?

So now let us at least give one example to convince you that it can suddenly do at least more than what PDA is push down automata could do, remember ultimately we would like to claim and we would like to convince you that this simple device that we are describing can do all computations but to begin with let us have one example where we see the power of Turing machines over push down automata.

(Refer Slide Time: 37:06)



So let us take this problem, the problem that we will tackle now for our simple first example of a capability of Turing machine is that we would like to recognize strings of this form, so what does it mean that imagine on the input we have a number of a’s followed by a number of b’s and then followed by number of c’s. We will assume our input is such that it has an indeed has a number of a’s followed by a number of b’s followed by c’s of course in general input could have been anything but let us at least assume that input is of the form a star, b star, c star that it has 0 or more a’s to begin with (())(38:17) more b’s then it has 0 or more c’s.

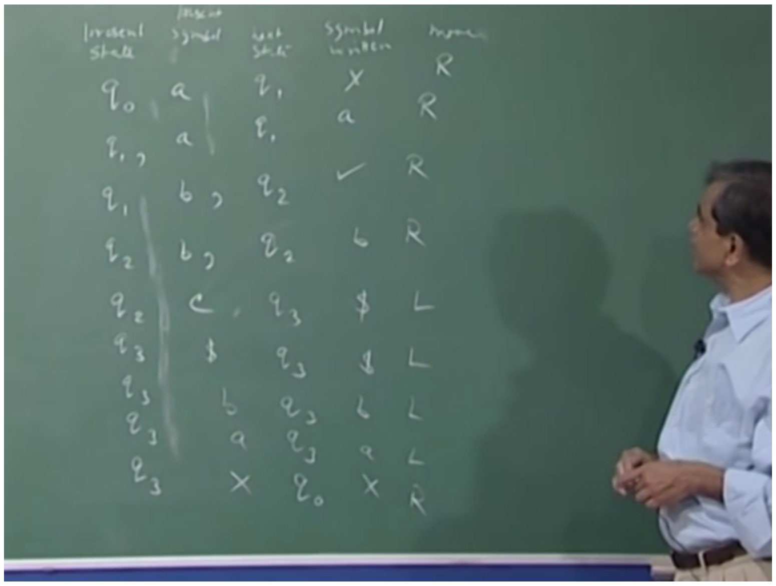
So this is the tape of the Turing machine initially look like this and we would our Turing machine would like to make sure that the block of a’s number of a’s in the block of a's that number is same as number of b’s and the number of c’s this the input starts here so initially the Turing machine is scanning this symbol a in its initial state q0, right. So how would such a device the Turing machine that we have described do this task of verifying that the number of a's same as number of b’s same as number of c’s.

One way of doing this the strategy could be that see so let us just see its sees an a and then it will make sure that corresponding to this a there is a b and a c. So what you might do or the Turing machine might do is that it will instead of a it will write a cross and then kind of move all the way when it sees a b it will put a tick, so that means corresponding to this a which has been crossed we have found a b which is now tick and again it will move to the right to find a corresponding c and let us say it will change it to dollar. So after some time the tape will look like this of course these here we have blanks here also we have blank, right? So what it has done we will see how the details are worked out by a Turing machine but basically it has found an a and corresponding to this a it has found a b and a c but of course now it has to do for all other a’s.

So again so therefore the machine will come back make this a as cross of course this cross exists this remember this b has been already checked off because we took this b to corresponding to this a and therefore we will tick the next b and now corresponding to the second a we will make this dollar to come in place of this c. So for example you can see what is happening this simple idea that machine sees an a it remembers that it has seen an a so therefore it must find a b and a c to correspond to this a, why does it put a cross? Because it needs to remember that yes this a I have seen and I am now going to check off 1 b and 1 c.

So basically it puts a cross for the a that it wants to make sure that there will exist a b and a c to corresponding to that a the b that finds it will put a tick and a c that it finds it will put a dollar and will go on like this, so let us at least before get in to more details let us see that this can be at least this part that I am describing can indeed be done by the machine kind of machine that we have described.

(Refer Slide Time: 43:28)



Present State	Present Symbol	Next State	Symbol written	move
q_0	a	q_1	X	R
q_1	a	q_1	a	R
q_1	b	q_2	✓	R
q_2	b	q_2	b	R
q_2	c	q_3	$	L
q_3	$	q_3	$	L
q_3	b	q_3	b	L
q_3	a	q_3	a	L
q_3	X	q_0	X	R

So let us say as we said initially the machine is in state q0 that is its initial state it is scanning this so let us recall this is present state and this is the symbol present symbol and it goes to next state and of course remember the quintuple the symbol overwritten and move direction of the move.

So the initial what we said was initially as it scans an a it will overwrite that symbol by a by that a by a cross. Now should it remain in this state q0? Not really because this now needs to remember that I have seen an a corresponding to which I must find a b and a c, so it moves to a let us say state q1 and which direction should it move as we had seen that will move to the right, good.

Now the machine will be in state q1 in this kind of case you can see it will see a's what should it do with a's it should not change a's and it will remain in the same state q1 and remember that a need not be changed of course because what in state q1 what you are looking for is a b to corresponding to correspond to the a which has been crossed of course it moves to the right, maybe then what will find? It will find sooner or later you know these block of a's will be over in state q1 we are going to see a b.

And we said that this b will be checked off by putting a cross putting a tick there and it should move to another state because now it should find a corresponding c, so let us say it moves to state q2 and it will still move to right. Then in q2 it will see maybe more b's it should not change a state by seeing a b because in state q2 what you are looking for is a c to correspond to the a which you have crossed, so the b remains and still moves to the right to find the first c or the c corresponding to the a that you have crossed.

So in q2 whenever let us now it sees a c and it should move to a state q3 by writing what should we do? We have said for the c it will print a dollar, and now what it should do? It should now start moving to the left because this job of checking that for a particular a there was a particular b and a particular c that job is done, so now it moves to the left in this state q3 and in state q3 in general what is it going to encounter? It will encounter not all it you know as it is scanning now coming back you can imagine that suppose it has done this it has put this dollar and then it may see some more dollars it may see some more b's, it may see some more ticks, it may see some more a's of course but it should start by crossing that a which is just next to the previous cross.

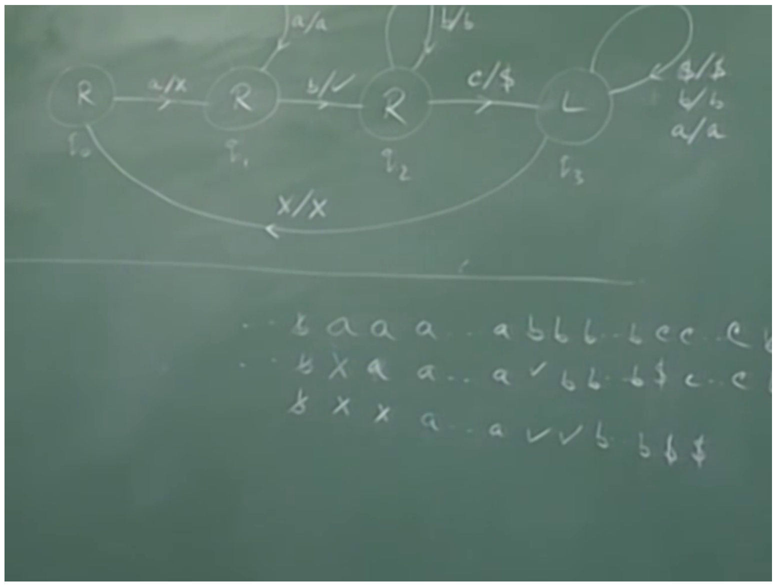
So it should come all the way till it sees a cross and then move to the right, so basically in q3 if you see a dollar you print a dollar remain in q3 keep move still keep moving to the left and in q3 if you see a b you remain in q3 do not change that symbol b still move to the left if you see an a you remain in q3 keep it a till move to the left till you find a cross and let me put a comma everywhere so that or let us remove all the commas. In q3 now when you see an X of course that means you have found the right most a the last a that you have crossed off, so of course that cross remains.

And now what you should do? You should move 1 step to the right that is the Turing machine should move 1 square to the right and start this checking for as a again, right? So basically it moves to this state q0, okay, right. And you know of course we have not yet fully described the thing but already you can see that this will becoming a little too messy you know this writing this quintuples each one separately can we do a little better just for describing simple Turing machines as this, yes we can use some diagram transition diagram for you know for Turing machines and of course that diagram is something it will be something similar to the transition diagrams of finite state machines but it should of course contain extra information like the direction of the move and if whatever the symbol being scanned what is the symbol being written because after all a move should describe a move as given by a quintuple, alright?

So now you see one interesting thing about this example that look at this q1 state q1, right? Whenever you are entering this q1 you are moving to the right, right? Similarly whenever you are entering the state q2 you are moving to the right? We have entered q2 here moving to the by moving to the right we have entered q3 here we move to the left we have entered q3 here again move to the left we have entered q3 here we move to the left we have entered q3 here moved to the left and q0 is the only place we have entered q0 we have moved to the right.

So this observation makes the task of drawing a transition diagram for a Turing machine simple, let us see how.

(Refer Slide Time: 52:35)



So at least in so far we had defined 4 states this is q0, q1, q2, q3 and now look at the first quintuple in q0 if you see a symbol a you put a cross and why do not and go to state q1, so it is like this and this will be your convention that a cross that means in q0 if I see an a then the symbol that I write is cross and then move to state q1, however where is the direction? Now we are now in this transition we are entering q1 but one thing we noticed about this state q1 so far that wherever whenever you enter q1 you move to the right.

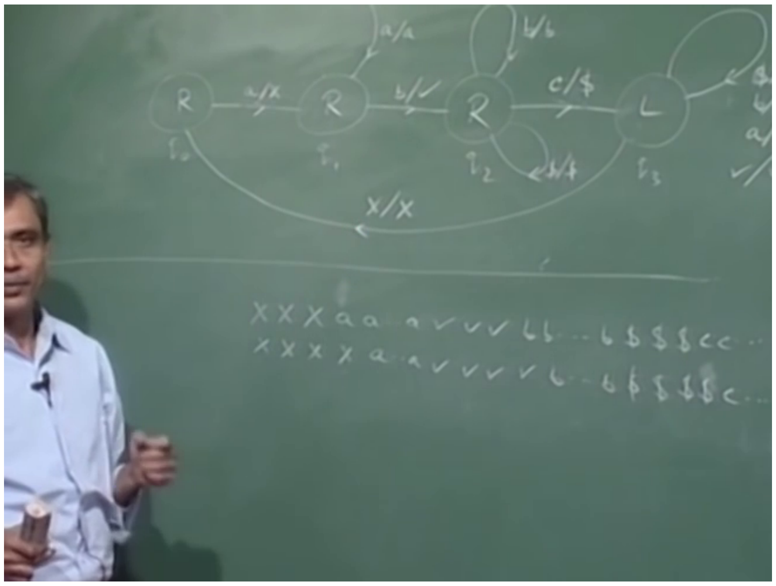
So why not place the symbol here right, that means now you see that all the five things of the quintuple they are described if I am in state q0 and seeing the symbol a then I write the symbol cross go to this state by moving to the right we are associating a direction with the state itself and the idea is whenever an arrow ends in that state then that transition must be following the direction which is written inside this circle which represents R, fine.

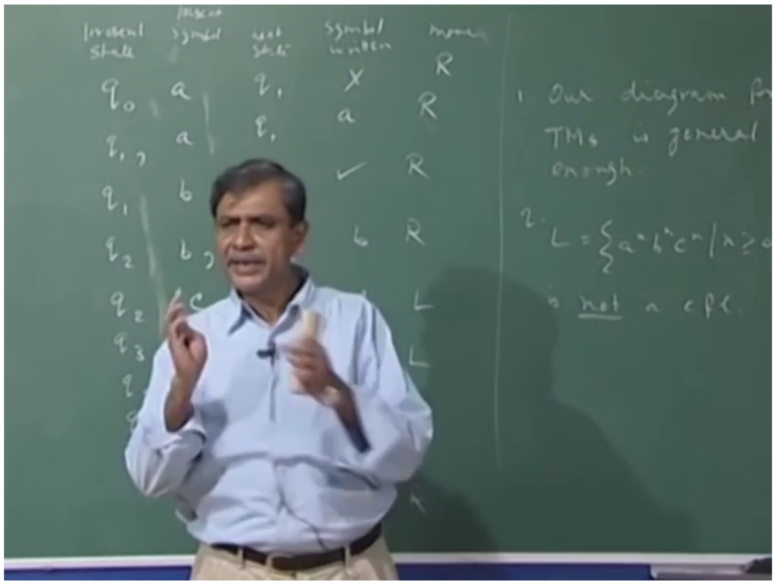
So let us do ahead and do this, in q1 if you see an a what did you do we again we move to the right by writing an a, so essentially in q1 if I see an a I will write a and I will remain in state q1 if I see a b then I move to state q2 by writing a tick in that place of b and then I move to the right. So therefore of course you will see that this is the right moving state in q1 if I see a b I will put a tick move to the right and now we are q2 and you in q2 if I see a b we will remain in the state but we will make a right move, in q2 if I see a c what we do? We go to a state q3 because this c will change to a dollar and in q3 when we enter we have entered q3 here in all these places we have entered q3 and moved to the left.

So we can write the direction here and in q3 if you see so let us go ahead in q3 if I see a dollar I keep it as such so this is a self-loop really but by make a move to the left in q3 if I see a b I still remain in the same state but I move to the left so I will just write like this in q3 if I see a still I remain in the same state q3 and keep the same symbol there so a is replaced or rewritten by a itself and there move to the left and now in q3 if I see an X then what we do we go to q0 of course we do not change that symbol the same symbol is written back but I moved to the right. So you can see that q0 is a.

This diagram actually at least the iteration part of the Turing machine very clearly explained. So let us just follow it once iteration is what? Iteration is that given an a starting with an a checking it off with b and a c and then coming that is then coming back for the next one.

(Refer Slide Time: 58:36)





So let us just see here so the iteration begins with the state q0 so in general maybe I had a number of crosses and then a block of a's then a number of ticks actually equal number of ticks as we have put crosses then some b's then equal number of dollars then comes c's then of course black.

So we can now see this Turing machine working in iterations and this iteration is that tick an a make a cross and corresponding to this a find the left most b that you can find make it so this will become a cross this will become another tick, this b will become another tick corresponding to this c you will make it a dollar, right? And then you should come back and possession yourself at the end of the iteration to start the next iteration here, so, okay. And at the end of the iteration you should be here.

So let us just see that is happening remember in the beginning of the iteration we are scanning this a and you see how convenient this particular diagram will be to understand this iteration. So remember that we the machine is here in state q0 and it sees an a therefore it makes it an x which it has and now where will the machine be, it will be after making it a cross it will be here. Now here it sees an a what should we do? It writes the same a it will see all these a's it will keep on being in the same state q1 till it sees a b this particular b which will change it into this tick and move to the right so the Turing machine head is now where having put the tick it is here, right? And then you can see it is if it sees a number of b's it will skip but it will remain in the same state till it will see a c but now we can see that in this state it will also encounter some dollars but those dollars also it will retain and till it finds a c and that c it will change it to a dollar now it will move left.

So when it will start moving left? This c after changing it to a dollar it is in this state c it has changed it to a dollar it is in this state which we had called q3 and in q3 it is basically it is a left moving state till you find a cross and then you move one step to the right, is that happening? So in q3 you are here you have moved to the left having changed the c to a dollar this particular c it was changed to a dollar and now you have moved to the left, so the Turing machine is going to see a dollar, so what does it do on a dollar it writes the same dollar comes back to the same state by moving to the left and so on.

So it will after some dollars it will see some b's it will still keep moving to the left it will see some ticks should it do anything, not really so tick also it will retain and it will see some a's again it remains in the same state it keeps moving to the left, right? Because it is in this state sees an a writes an a and comes back to this state by moving to the right it sees an a it will come here, it will come here sooner or later it will if this a left of which there is a cross. So it will come back having seen this a it will come back to this state and now the state that the symbol that it will see is a cross so this cross is written back as cross and it moves to the right scanning a.

So this completes one iteration and therefore you see what is happening in that iteration? What is happening is for each a it finds a corresponding b and a corresponding c. Now this iteration if this successfully happens for all a’s that is for each a it can find a particular b and a particular c corresponding b and a corresponding c and at the end no more b’s or no more c’s are left in that case of course number of a’s same as number of b's same as number of c’s so the Turing machine will be able to decide that indeed the block of a’s and block of b’s and block of c’s that we had to begin with they have equal number of symbols.

To before we end we end this class here but before we end I would like to make a few points some of you already might be wondering that is this a kind of restriction that we are placing that whenever we are entering in a state we always move in the same direction in this kind of diagram that is what is happening but in general of course that this one particular state you can enter by either moving to the left or moving to the right.

In that case how can I take care of such a state in this kind of diagram? Indeed you cannot, however the point is that if you had a Turing machine where a particular state is entered both by moving to the left as well as by moving to the right then you can have an equivalent Turing machine equivalent in the sense it will do the same job really where each state when you enter will be like this that is every state you enter on only one direction.

So we lose no generality by considering these kinds of diagrams as the diagrams for our Turing machines and this is a simple exercise to show a little later you yourself I am sure you will be able to solve that exercise that given a Turing machine which has a state or a number of states such that a you know this state you move into sometimes by moving to the left, sometimes by moving to the right from such a Turing machine you should be able to obtain another Turing machine which will do essentially the same thing by having states with this restriction, so that is the point number 1.

So our point number 1 is our diagram for Turing machines is general enough. And second thing we have not fully completed this example but I think you are already convinced that indeed you should be able to recognize fully you know by doing a few more additions etcetera about the limit cases that is what happens when you know we have not said when the halting is happening and all that in this diagram but all that we can take care of we have not said that halting by saying yes the string is indeed like number of a’s is equal to number of b’s is equal to number of c’s but if that is not the case what happens so all that we can incorporate that not too difficult here but already if the Turing machine can indeed solve this example successfully then we already know that it can do something which no PDA could because we know that this language is not a CFA.

So therefore at least right now you are getting to believe that Turing machines can do things more than what PDA's ever could, can I do everything that a PDA could? Yes, that is not too difficult but right now let us not get into those details before if kind of appreciate (())(69:06) the definition of Turing machine we are some of our clear about working with Turing machines and then as the course proceeds I think you will be convinced that indeed Turing machines can code all algorithms.

One more point before we end remember that this is for example one Turing machine as of now there may be some more states later on we will add but right now it has a number of states, number of transitions we get different Turing machines by of course having different or set of states maybe symbols maybe transitions what we are finally interested in what all things the class of Turing machines can perform.