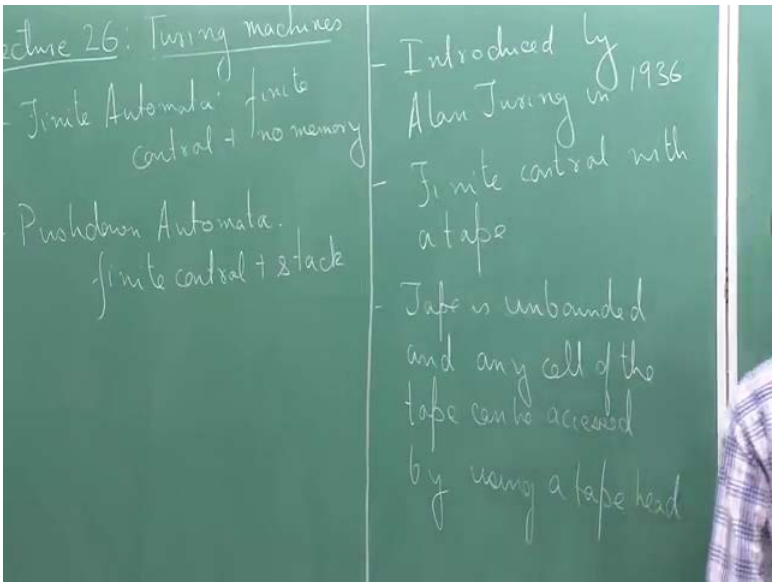
Theory of Computation
Prof. Raghunath Tewari
Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur

Lecture – 26
Turing Machine

Welcome to the 26th lecture of this course. Today, we are going to introduce this new computational model known as Turing Machines.

(Refer Slide Time: 00:17)



So, before I talk about Turing machines, let us do a quick recap of whatever we have seen so far. So, in terms of automata, we have seen finite automata. So, in finite automata, we had a finite control and no memory. Then we introduced pushdown automata. So, in pushdown automata, we had again finite control plus memory in the form of a stack. So, by finite control, essentially we mean that the set of states is finite in nature.

So, in finite automata both deterministic and nondeterministic, we do not have any memory in that model, but in the case of pushdown automata we have an unbounded stack as our memory. So, there is no bound on the length of the stack that can be used; the only restriction is that it is a stack; therefore, we can only access the top most element at any given point of time.

Now the question is that how can we extend this. So, instead of a stack what if we consider other data structure. A Turing machine is essentially an automaton, where we have finite control plus memory in the form of a tape instead of a stack. So, what is a tape, a tape is again an infinite data structure, but in the case of a tape, we can access any cell at any given point of time.

So, a tape has a tape head which is pointing to some cell of that tape; and we can move the tape head left or right whichever way we want, and access whichever cell we want to. We do not have to erase contents of the tape to access a certain cell as we had to do in the case of a stack, so that is essentially during what a Turing machine is. We will see more formally what is the definition how we define a Turing machine.

But before that let me give a brief historical motivation as to how Turing machines were invented or how they were discovered. So, a Turing machine somewhat contrary to that what the name suggests, it is not a mechanical machine, it is not a machine in the sense that we colloquially use it. It is a mathematical object, it is basically an automaton, it is a mathematical object which is modeled to simulate computation. So, any computation or computation can be modeled in a Turing machine in a similar way that it had been done in the case of finite automaton and pushdown automata.

So, Turing machine was introduced by Alan Turing in 1936. The historically it is very interesting how Turing machines were actually discovered. So, what Alan Turing was trying to do was, Alan Turing was trying to understand the limits of computation. So, are there any computational problems, which cannot be computed in the sense of the word? So, there were, so 1936, we did not have computers, computers that we see today. So, but the definition of computation was sort of clear, I mean any process that can be used to obtain an answer from a given input using a step-by-step process.

For example, if I want to multiply two numbers, I can always do that on pen and paper, I can always device an algorithm that would tell me how to go ahead multiplying two numbers in a step-by-step manner to obtain the product of the two numbers. Similarly, for other problems also, there are methods that you can apply in order to get the solution from the input in a step-by-step manner. The question that Alan Turing or even mathematician before Alan Turing were interested in its that is there any problem is there any computational problem where you cannot actually obtain a solution by going in a

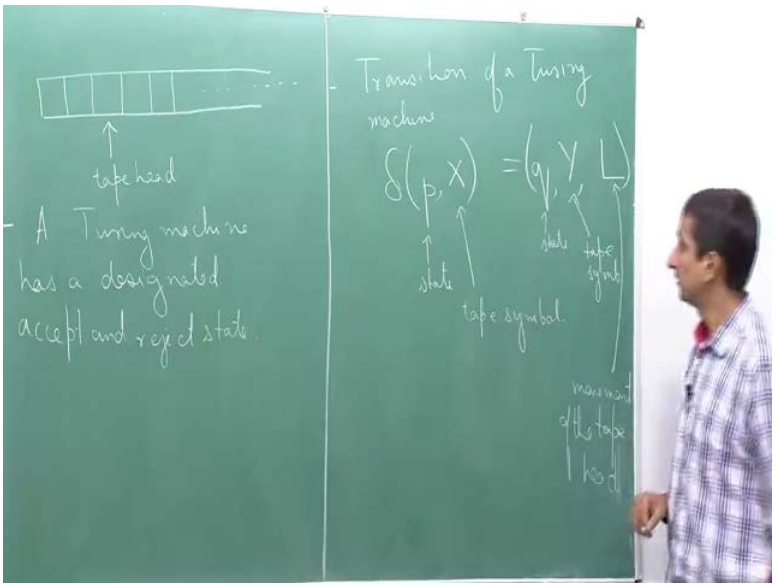
step-by-step manner. And to actually prove this, it is essential to give a mathematical modeling of what computation is.

So, informally we understand what we mean to solve a problem in a step-by-step manner to device an algorithm, but in a generic manner, how do we define such computation or how do we define algorithm. So, to answer this question, Turing actually came up with this model which today we call as Turing machine, and he showed that there are problems, there are computational problems which cannot actually be solved using the Turing machine.

And what he proposed as a hypothesis is that any problem that can actually be solved even by let say the most powerful computer can also be solved by a Turing machine. So, there were no computers at that time. So, what he essentially said what is popularly today known as the church during the hypothesis is that any problem any for any computational problem that can be computed also has an algorithm using Turing machines.

So, it is still a hypothesis, there is no formal proof of this statement, but what interesting the thing that Turing showed that there are problems, there are problems that can be modeled as computational problems, which cannot be solved using Turing machine. So, we will look at these questions, and we will actually go into the limits of Turing machines in a few lectures, but today our goal is to understand how Turing machines are defined, and why are they more powerful than pushdown automata. So, as I said that a Turing machine is essentially has a finite control with a tape. The tape is unbounded and any cell of the tape can be accessed by using a tape head.

(Refer Slide Time: 09:41)



Let us look at a picture. So, essentially a tape is unbounded in nature. So, it is a more precisely, it is semi unbounded. So, in the left side, there is a bound; so you cannot move to the left of this boundary on the left, but on the right side, it is unbounded. So, this is how you can picture it. And it has cells; so every cell has is capable of containing one symbol. And there is a tape head, so this is the tape head, which is capable of pointing to any cell of the tape; and whichever cell the tape head points to you can read the contents of that cell only.

Now suppose if you want to read the content of this cell, then in two moves first the tape head has to move from this current cell to the one to its right; and then again in the next move, it has to move one more step to the right, and then you can read this. So, in every step basically you can move the tape head either to the left or to the right, and you can read the contents of the cell to which the head is currently pointing to, and you can also modify the content. You can modify to you can basically change whatever symbol is contained in that tape cell to anything that you want. So, this how a Turing machine works.

So, essentially the tape corresponds to the memory of the Turing machine. So, if you want to think of this as a computer, so the finite control is essentially the CPU of the Turing machine of the computer. The finite control is able to decide where your tape head should move to, what contents it should be replaced by. So, suppose if you are

running an algorithm, a finite control is essentially the steps of the algorithm, what essentially you need to do at every step, and the tape is essentially the memory that your algorithm uses, so whatever the content that you want to save in order to compute something.

Now a Turing machine has a designated accept and reject state. So, this is again something that is different from a pushdown automata or finite automata. So, in the case of a finite automaton or let say pushdown automaton, there was an input tape where the input was written. But in the case of a Turing machine, we will assume that the input itself is written on the on the tape of the Turing machine, so that we will assume that a Turing machine has only one tape which is used as work space as whereas to store the input.

And we will assume that the input is written to the left - left hand side of the tape. So, whatever input is it is written on the left hand side, and then you can actually erase the input also, and you can basically go back and forth on the tape you can go forward and then again come back read some bit, again go forward and so on. And the way an input is accepted or rejected is basically by means of - accept and reject state.

In the case of pushdown automata, we did not have a reject state. So, we said that an input is accepted, if the Turing machine is or if the pushdown automaton is on a accept state when the input is fully read; and otherwise it is rejected. In the case of a Turing machine, as soon as the computation of the Turing machine enters the accept state or the reject state, we halt the computation. And if it enters the accept state, we accept the input; if it enters the reject state, we reject the input; it is not dependent on whether the input is read or not. The input can be read as many times as necessary that is point number one.

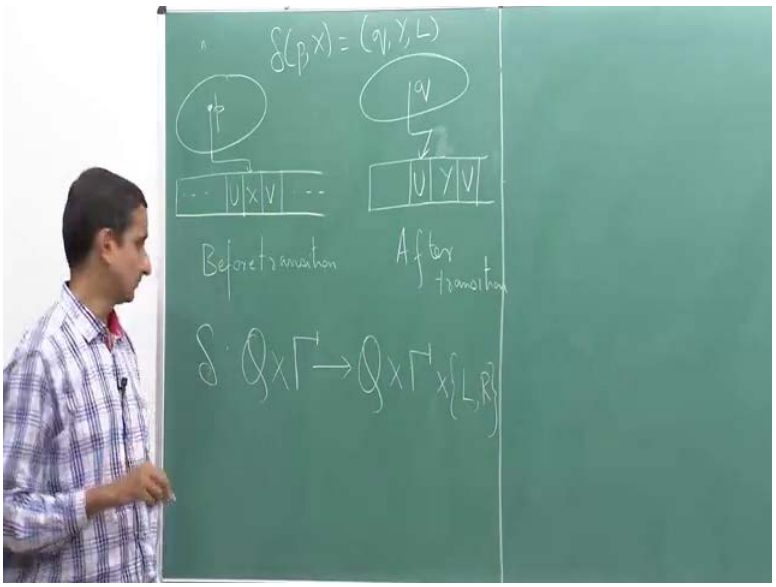
And point number 2 is that because of this condition that we only halt when we enter accept or the reject state, it can very well happen that you design a Turing machine, which never enters accept or the reject state. In that case, we say that the Turing machine essentially goes into infinite loop it never halts on that particular input. So, you can for example, you can write programs in C, which never halt right, you can write loops that run forever. So, similar thing can actually happen with a Turing machine also.

Now how does the transition look like? So, informally or I want to describe so transition of a Turing machine essentially looks like the following. So, we have a transition

function delta, which takes two things a state p and a tape symbol X. So, this is a state, this is a tape symbol. And it produces a triplet q, Y and let say L. So, what are these 3 things? So, on reading state p or on from state p on reading the tape symbol X, this transition means that the machine goes to state q, it replaces X with the symbol Y, so this is also state.

This is another tape symbol and the head moves to the, so this is the movement of the tape head. So, after reading the symbol X, the machine replaces X with Y; and the head moves one cell to the left. So, instead of L, you can also have R, which would mean that the head moves one cell to the right. So, I had said earlier that the head only moves either one cell to the left or one cell to the right. So, this last parameter, this last element of the tuple essentially represents that fact.

(Refer Slide Time: 17:45)

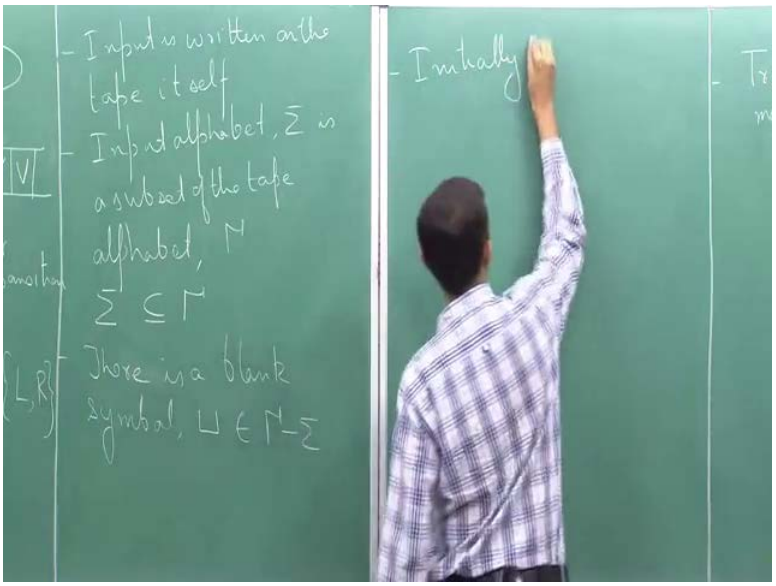


So, pictorially if I want to represent this, so let say that I have my finite control the machine is at state p. And here is my tape where there is a cell with X written on it. And maybe there is Let say the left cell has U written on it, right cell has some symbol V written on it and the remaining tape contains some other content. So, this is before the transition. So, after the transition what happens? So, if I look at the transition delta p X going to q Y L what happens after the transition. So, after the transition it moves to state q on my tape, I replace X with Y and sorry the tape head moves left.

The tape head will basically point to U now. So, this is basically after the transition. So, if I denote the transition as delta p, X equal to q, Y, L. So, on reading X, it replaces X with Y; it goes from state p to state q and the tape head it moves from the cell which it is currently pointing to the cell left of it. So, this is each transition.

Therefore, if I want to formalize this is what does the transition function I mean how to define the transition function. The transition function delta is essentially a function which takes two arguments. So, first it takes a state, and then it takes a tape symbol. Let us represent tape symbol with gamma, then it produces a triplet. So, it produces a state, a symbol Y, and it gives us a bit which tells us whether to go left or right, so cross two bits either left or right. So, this is the transition function of a Turing machine.

(Refer Slide Time: 21:11)

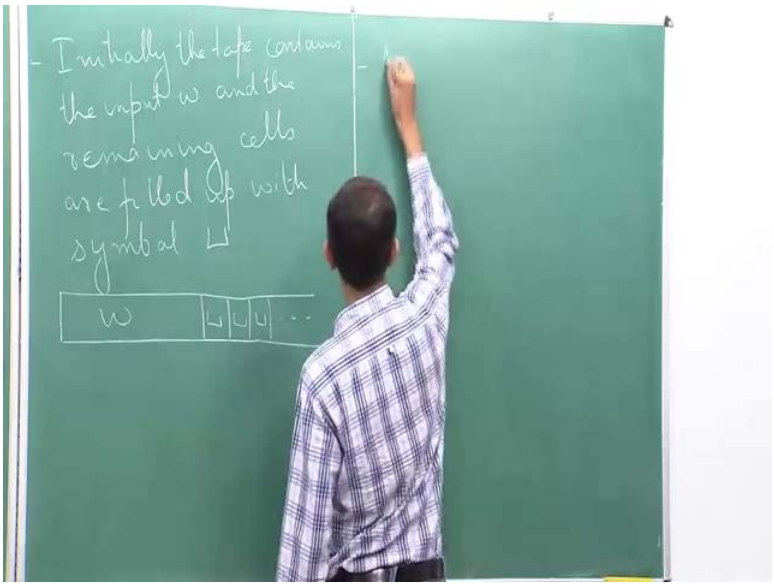


As I said the input is written on the tape itself. So, it is written on the tape, I mean actually you actually begin the computation with only the input written on the tape. And there is a special symbol. So, before I say that, so we will assume that, so because the input is written on the tape itself, what we will assume is that the input alphabet sigma is a subset of the tape alphabet gamma.

So, basically sigma is a subset of gamma and in fact, this is a proper subset, because we will assume that so there is a blank symbol. So, this is represented as open box like this, which belongs to the tape alphabet gamma, but not the input alphabet sigma. So, there is a blank symbol gamma which belongs to sorry there is a blank symbol which belongs to

the tape alphabet gamma, but it does not belong to sigma. So, this is just to distinguish the input from the remaining contents of the cell.

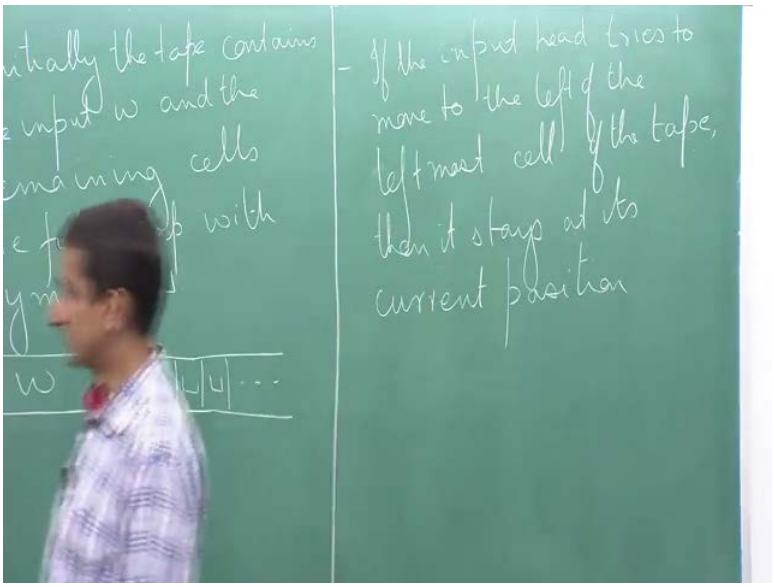
(Refer Slide Time: 23:40)



At the very beginning, what we will assume is that, so in the beginning we will assume that, so initially the tape contains the input w, and the remaining cells are filled up with the symbol. So, initially basically we have the tape, which contains w till let say some point; and the remaining cells are filled up with blank. This is the initial configuration.

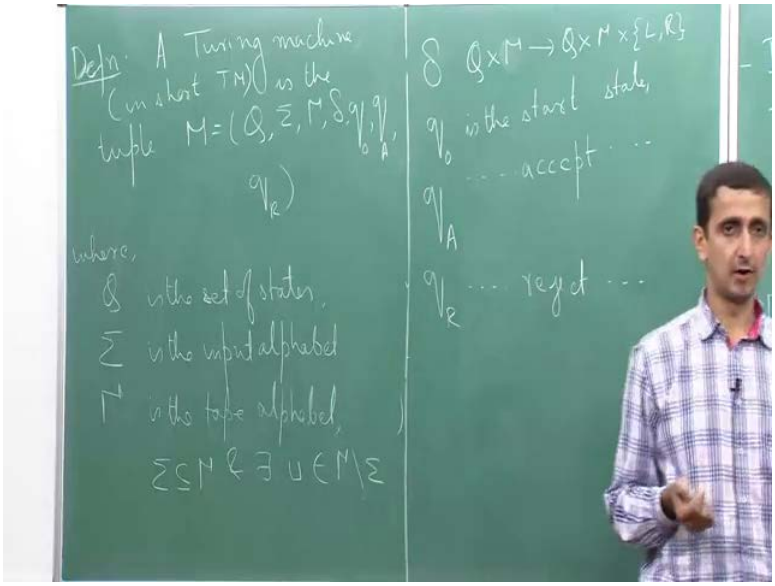
So, one more point, so because we have this assumption that because if we the way we define a tape, the tape is bounded on one side. So, when we look at the tape, if I look at the left most cell of the tape, I cannot go beyond the left most cell I cannot go to the left of the left most cell.

(Refer Slide Time: 25:32)



Therefore, when I define the transition function I need to have this additional point that if the input head tries to move to the left of the left most cell of the tape, then it stays at its current position. So, if the input cell actually tries to move to the left, then it will not make that move and basically just stay at it is current position. So, this is what we need, this we need to make this assumption about the transition function in order for it to be a complete function.

(Refer Slide Time: 27:02)



Now, that we have all these points, now we can give the formal definition of what a Turing machine is. So, a Turing machine in short TM is the tuple let say M equals Q, sigma, gamma, delta, q naught, this is the start state, and we have two other states q A comma q R. Where of course, Q is the set of states sigma is the input alphabet; gamma is the tape alphabet, where sigma is a subset of gamma, and there exists a blank symbol, which is contained in gamma, but not in sigma. So, this is the set minus sign. Delta is a map from Q cross gamma to Q cross gamma cross L comma R; q naught is the start state; q A is the accept state, and q R is the reject state. So, there is a unique accept state and a unique reject state, where the computation halts.

So, I will stop here. And we will continue with the description of we will continue with more definitions about a Turing machine and some examples in our next lecture.