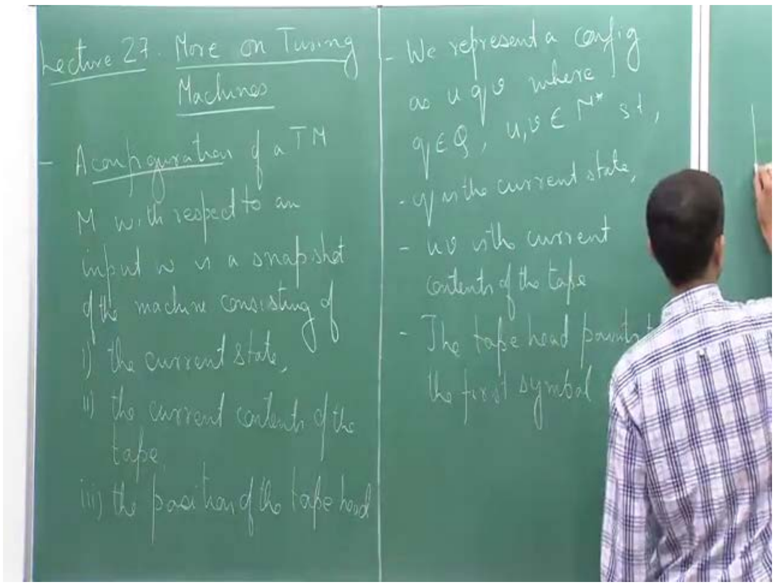
Theory of Computation
Prof. Raghunath Tewari
Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur

Lecture – 27
More on Turing Machine

(Refer Slide Time: 00:51)



Lecture 27. More on Turing Machines

- A configuration of a TM M with respect to an input w is a snapshot of the machine consisting of
  i) the current state,
  ii) the current contents of the tape
  iii) the position of the tape head

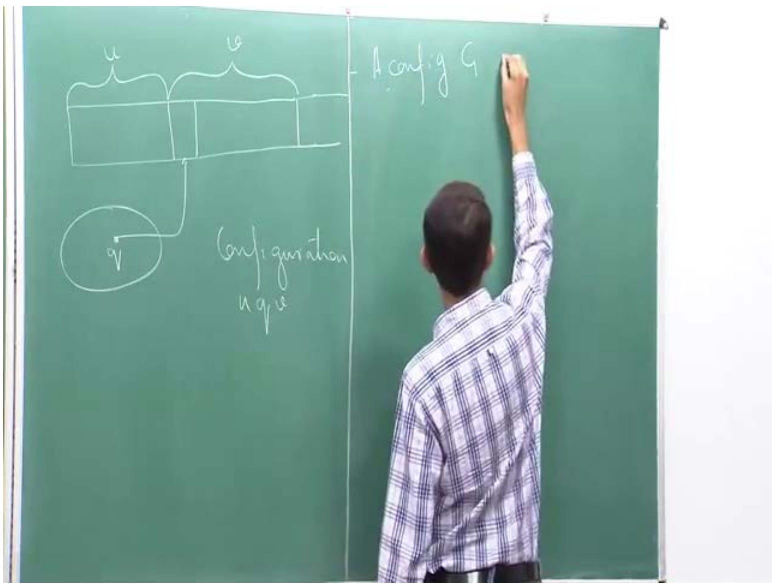
- We represent a config as $u q v$ where $q \in Q$, $u, v \in \Gamma^*$ s.t.,
  - q is the current state,
  - uv is the current contents of the tape
  - The tape head points to the first symbol

Welcome to the 27 lecture of this course. So, today we are going to look at more definitions involving Turing machine, so for example, what does it mean for a Turing machine to accept an input reject an input and so on. And we are also going to see an example of language accepted by a Turing machine. So, the first thing that we will see is the concept of a configuration of a Turing machine. So, a configuration of a Turing machine M with respect to an input w, so the configuration is always defined for a Turing machine with respect to some input.

So, basically it will be changes with different input is a snapshot of the machine consisting of, it consists of three things the current state, the tape contents or the current contents of the tape, and thirdly the position of the tape head. So, essentially what a configuration means is that, so during the computation of the Turing machine on some input w, if I just freeze the computation at some instance, if I just freeze the computation, in other words if I take a picture of the computation at any given point of time during the computation, what are the objects necessary to describe that snapshot or what are the objects necessary to describe that particular moment of computation.

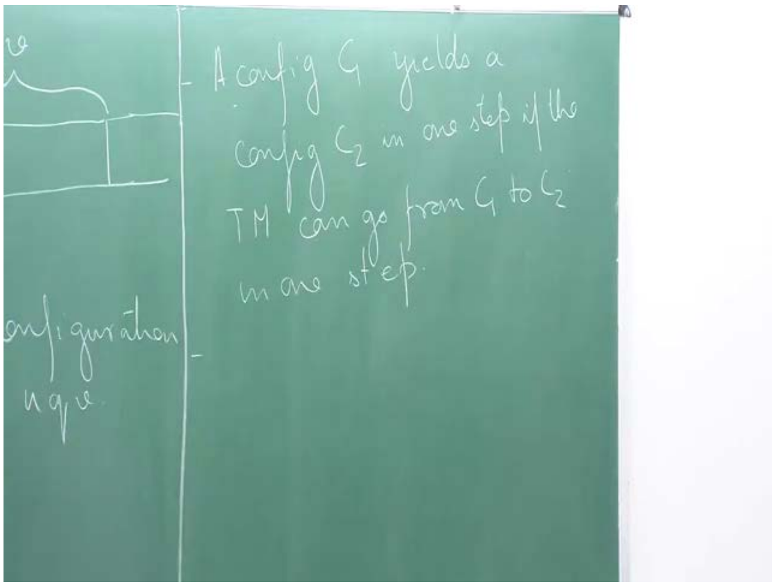
So, the Turing machine at any at that point is in some state, so the current state is necessary it can be in any state, but tape can have any contents at that moment. So, the current content of that tape is necessary. And the third thing that is necessary is the position of the tape head, because the head can be pointing to any arbitrary or any position or to any cell of the tape at that given point. So, formally we use the following notation. So, we represent a configuration as u, q, v, where q is a state, u and v are strings over gamma. Such that q is the current state, the string u, v is the current contents of the tape, and the tape head points to the first symbol of v.

(Refer Slide Time: 06:07)



So, let me explain this pictorially. So, suppose at a given point, this is my tape. I have my finite control over here. And let us say that I have the currently the computation is at some state q, and it is pointing to some cell over here. So, how do I represent this? So, let say that this is what the tape currently it contains. So, I divide the current contents of the tape into two parts; one part is whatever is contained to the left of where the input head is pointing, so basically this portion. So, this portion I will call as u. And the second is whatever is contained to the right of u so, basically including the current tape cell and whatever beyond it, so that I will call as v. So, the string that is to the right of u, until the end of the contents of the tape. So, this is my string v. So, essentially the string w that is currently there on the tape is concatenation of u, v and of course, q is the current state.

(Refer Slide Time: 08:18)



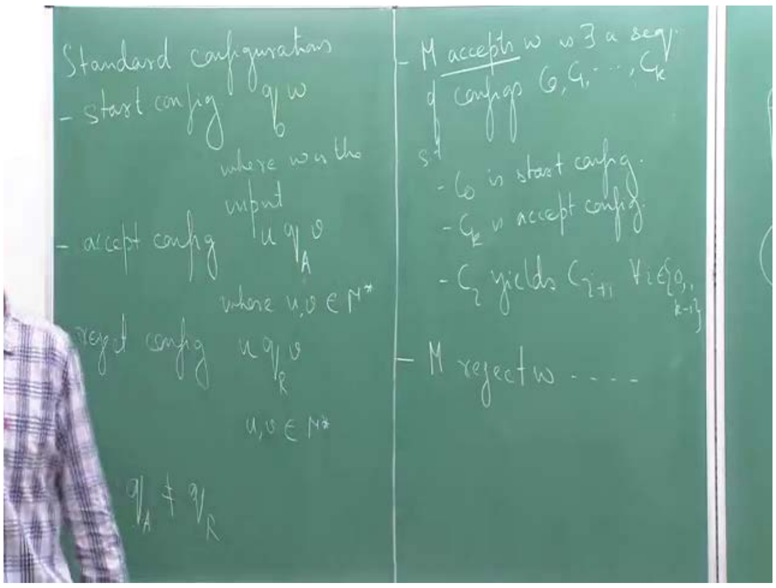
v

onfiguration $u q v$.

- A config C_1 yields a config C_2 in one step if the TM can go from C_1 to C_2 in one step.

So, this is what is meant by the configuration u, q, v. Now, what does it, how do we define a single step? So, we say that a configuration C 1 yields a configuration C 2 in one step if the Turing machine can go from C 1 to C 2 in one step. So, this is similar to actually pushdown automata. So, this is defining one step. So, how do you go from one configuration to another configuration?

(Refer Slide Time: 09:43)



Standard configurations

- start config $q_0 w$ where w is the input
- accept config $u q_A v$ where $u, v \in \Gamma^*$
- reject config $u q_R v$ $u, v \in \Gamma^*$

$q_A \neq q_R$

- M accepts w is $\exists$ a seq. of configs $C_0, C_1, \cdots, C_k$ s.t
  - C_0 is start config.
  - C_k is accept config.
  - C_i yields C_{i+1} $\forall i \in \{0, \ldots, k-1\}$
- M reject w . . . .

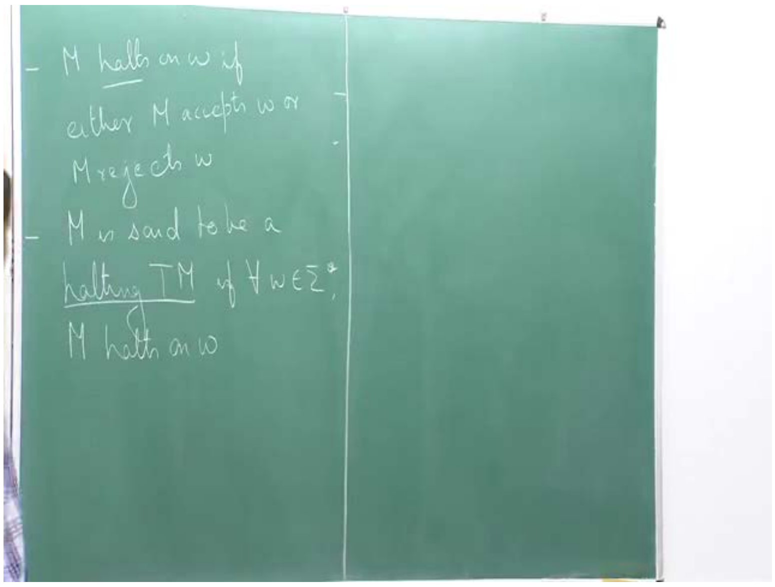
Similarly, we define… So, now, let me to define how to accept or how to reject a string, let me just talk about some of the standard configurations. So, let me do it here. So, we have the start configuration. So, how does the start configuration look like? So, the start configuration will essentially look like q 0 w, where w is the input. So, the reason is because at the beginning, so if I go back to this drawing, so at the beginning, the input head is pointing to the first cell of the tape, and the tape just contains the input at the beginning. So, v is what the input is, u is the empty string, because there is nothing to the left of input tape, and the current state is of course, q 0, so that is why it is q 0 w.

So, now by similar argument what is the accept configuration going to be. So, they accept configuration will be some string u followed by the accept state, and some string v, where u comma v are some two arbitrary strings. They need not have any relation with the input; they can be totally modified thing. The only thing that is important is that the current state should be the accept state. So, the moment the Turing machine enters the accept state, the input is accepted. Similarly, for reject, the reject configuration is u q R v where once again u comma v belongs to gamma star.

So, another point that I want make at this stage is that it is necessary here that the states q A and the states q R are not the same. You cannot have just one state for accept and reject because then if the Turing machine is enters this state, we do not know whether to accept or to reject, it is necessary that they are separate states. So, we say that M accepts w, if there exists a sequence of configurations C 0, C 1 up to C k such that C 0 is start configuration, C k is accept configuration, and C i yields C i plus 1, for all i in 0 up to k minus 1 . So, from C i, I should be able to go C i plus 1 in one step. So, this is the definition of acceptance.

Similarly, we can define a, what does it mean to say that M rejects w. So, I am not writing the entire definition. So, everything stays the same, but only difference is that instead of accept configuration, now C k must be a reject configuration. So, this is a fundamental difference between Turing machines and other automata that we have seen earlier. So, in other automata whether finite or pushdown, we said that if a string is not accepted, then it is rejected by default, but in the case of Turing machine that need not be the case. So, for every string, there are three possibilities. So, possibility one is that it is accepted possibility two is that it is rejected. And then there is a third possibility that it never enters either an accept configuration or the reject configuration which means that it loops forever it goes into an infinite loop.

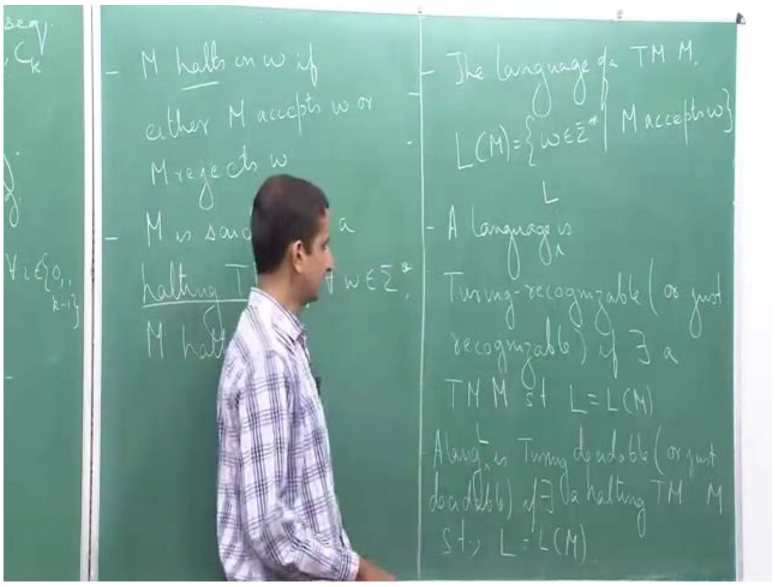
(Refer Slide Time: 15:22)



- M halts on w if either M accepts w or M rejects w
- M is said to be a halting TM if $\forall w \in \Sigma^*$, M halts on w

So, we say that M halts on w if either M accepts w or M rejects w. M is said to be a halting Turing machine if for all w in sigma star, M halts on w. So, basically we say that a Turing machine is a halting Turing machine, if it always either accepts a string or it rejects a string, it does not loop on a string forever.

(Refer Slide Time: 16:53)



- M halts on w if either M accepts w or M rejects w
- M is said ... a halting T... $\forall w \in \Sigma^*$, M halt...

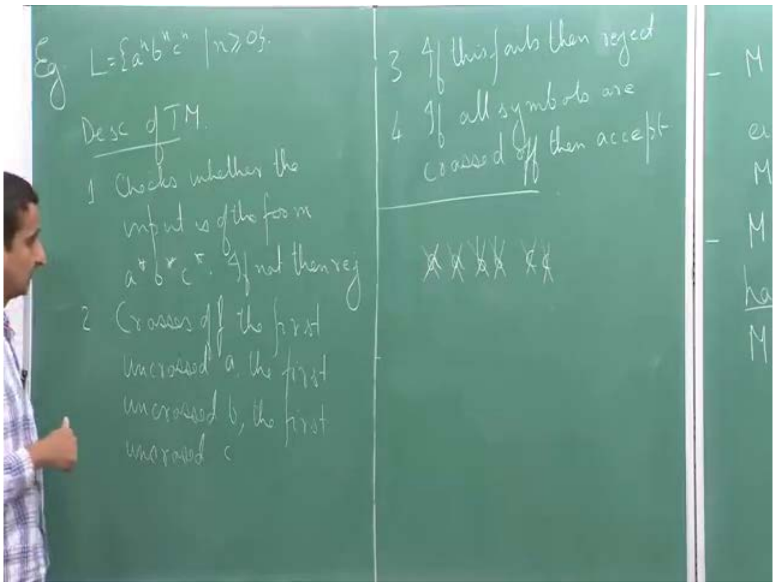
- The language of a TM M.
  $L(M) = \{w \in \Sigma^* \mid M \text{ accepts } w\}$
- A language L is Turing-recognizable (or just recognizable) if $\exists$ a TM M s.t. $L = L(M)$
- A lang L is Turing decidable (or just decidable) if $\exists$ a halting TM M s.t. $L = L(M)$

So, few more definitions, so we define the language of a Turing machine, the language of a Turing machine M comma l of M is defined as the set of strings w such that M accepts w. So, we say that a language is Turing recognizable or may be in short just recognizable, if there exist a Turing machine M such that L equals L of M. So, similarly we say that a language is Turing decidable or just decidable, if there exist a halting Turing machine M such that L equals L of M. So, let us just go through these three definitions. So, first of all the language of a Turing machine is always defined as the set of strings that it accepts. Now, we say that a language is Turing recognizable. So, we say that language, so I am sorry I should have said a language L is Turing recognizable if there is a Turing machine M such that l is equal to l of M.

So, what this essentially means that for every string that is in the language the Turing machine accepts it; and whatever the Turing machine accepts in the language, but if there is a string that is outside the language, either the Turing machine can reject it or the Turing machine can go into an infinite loop on that string. So, first strings that are in the language, there is only one possibility the machine will accept. For strings that are outside the language there are two possibilities; either, the machine rejects or it goes into a infinite loop. So, that is the thing and Turing recognizable in short is also called recognizable and some texts also call such languages as recursively enumerable languages.

The second type the second class is the class of the decidable languages, so a language L is said to be decidable or just sorry Turing decidable or just decidable if there exist a halting Turing machine M such that l is equal to l of M. So, what this essentially means is that for every string that is inside the language, the machine accepts; and for every string that is outside the language, the machine rejects because the machine has to halt on every input. So, this is a important point.

(Refer Slide Time: 21:56)


Eg. $L = \{a^n b^n c^n \mid n \geq 0\}$.

Desc of TM.

1 Checks whether the input is of the form $a^* b^* c^*$. If not then rej

2 Crosses off the first uncrossed a, the first uncrossed b, the first uncrossed c

3 If this fails then reject

4 If all symbols are crossed off then accept

$\cancel{a}\cancel{a}\cancel{b}\cancel{b}\cancel{c}\cancel{c}$
M
M
M
M

So, now let us look at an example. In fact, for our example, we will choose a non-context free language. So, let us consider the language L which is equal to a to the power n b to the power n c to the power n. How do we accept this input? So, I will instead of constructing that transition function in a formal manner, let me give the intuition or a more slightly more high-level description of how this language is accepted. So, first what the machine does is that it, so description of Turing machine. So, first what it does is it checks whether the input is of the form a star b star C star, so it just makes a pass of the input. So, it goes forward checking whether it is a collection of a’s followed by a collection b’s followed a by collection of c’s and then it again come back. If it is not so, then it will reject; and if it is so then it comes back to the input. So, if not then reject.

So, next what it does is that it crosses off the first uncrossed a, the first uncrossed b and then the first uncrossed c. So, it basically keeps on scanning the input in each round what it does that it crosses off the first uncrossed a, then the first uncrossed b, then the first uncrossed c and again comes back to the first uncrossed a, and then it again comes back. So, if this fails then reject if all symbols are crossed off then accept. So, if in any round you try to cross off symbol and it fails then you reject. So, may be you crossed of an ‘a’, but you could not crossed off ‘b’ you reject; may be and the same with c. But if in the end you end up crossing all the a’s, b’s and c’s then you will accept. So, what I mean by is this is the following. So, let say that initially we have on the tape a a, b b, c c. So, how does the algorithm work? So, first it starts from the left it crosses off the first a, it crosses off the first uncrossed b, it crosses off the first uncrossed c, and then again comes back. Again it makes pass of the input, it crosses off the first uncrossed a, it crosses the first uncrossed b, and the first uncrossed c and again it come back. So, now all the symbols are crossed off, hence it accepts.

if on the other hand if we had let say a different the number of a’s was different from the number of b’s or number of b’s was different from the number of c’s, some symbol would be left out in which case we would have rejecting. So, this is how the algorithm works. So, one more point about this Turing machine is that what does it mean to say that symbol is crossed off. So, what it means essentially is that see in our tape alphabet we can actually have more symbols. So, originally our input symbol consists of a, b, c. Now, I can add three more symbols on my tape alphabet a crossed off version of a, a crossed off version off b, and a crossed off version of c. So, whenever I am crossing off an ‘a’, it is essentially replacing a with a crossed off version of a and the same with b’s and c’s. So, this is something that can be done. So, I will stop here today, we will again continue next time.