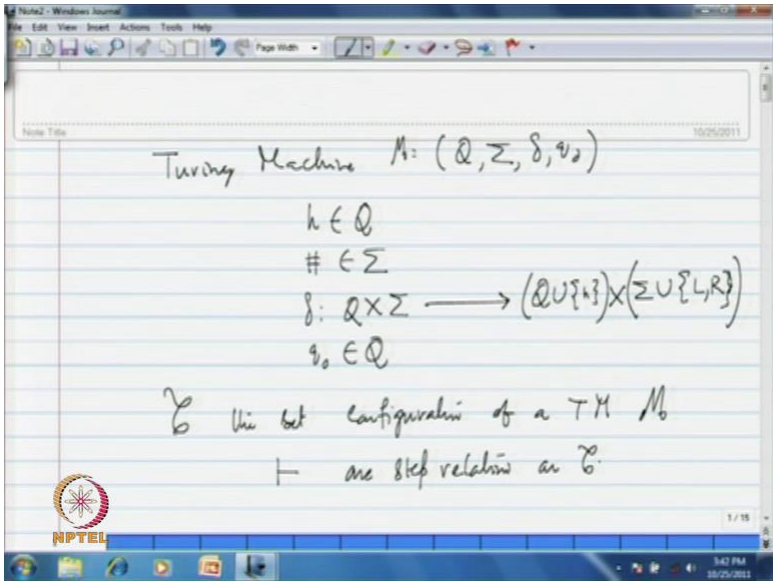
Formal Languages and Automata Theory
Prof. Dr .K.V. Krishna
Department of Mathematics
Indian Institute of Technology, Guwahati

Module - 11
Turing Machines
Lecture - 2
Turing Computable Functions

In my previous lecture, I introduced the concept called turing machine. As we see several sophisticated or more complicated languages than regular or contextual languages for that purpose, we have introduced. And at that point of time, I have also mentioned that, this has both the features of language acceptor, as well as the features of more and mille type of machine giving output also.

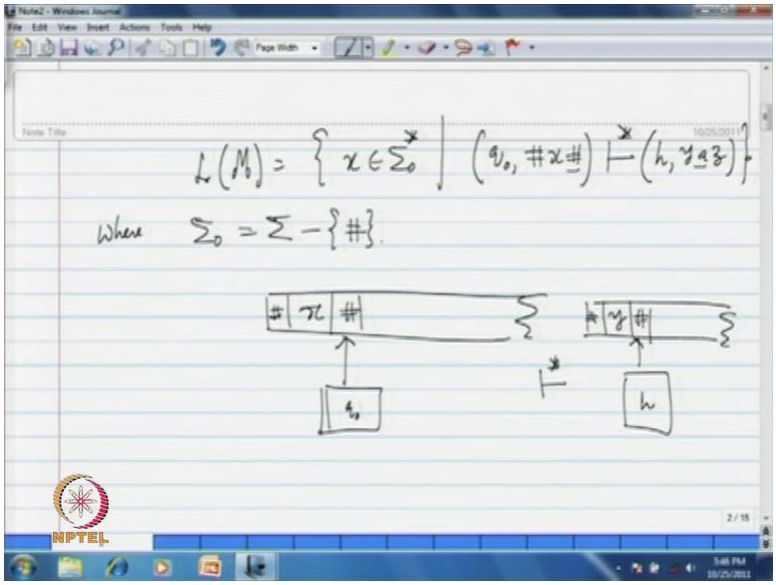
(Refer Slide Time: 01:01)



First let me recollect, what we have discussed in the previous class. I have introduced turing machine as a quadruple that is written Q, sigma, delta, q naught; where q is a finite set, sigma is a finite set, delta transition map and q naught, the initial state of this machine. And q set of states in which h is not part of that the halting state, that we are using. And we take this special symbol to use blank, that is an element of sigma and we have taken this, a total function the transition map that is from Q cross sigma, uninary straightened symbol we assign, so called the next state can be halting state also.

And that is either you print a symbol or you move to left or right. This is how we have defined and q naught is an element of q called the initial state. This is how, I have introduced the notion of turing machine as a quadruple and I have introduce the concept of computation there, like in case of other automator. So, one step relation is introduced and that is through each transition, if you apply a transition on a ((Refer Time 02:46)) symbol and a particular configuration. So, this one step relation and configurations are introduced, if I write c, this set of configurations of a turing machine and I have discussed about this one step relation as usual a binary relation on c.

(Refer Slide Time: 03:29)



And then we talked about the reflexive transitive closure of that and given a turing machine M, a language accepted by a turing machine M is set of all those strings x in sigma naught star such that. If you give in this format infinitely many step, you are getting a halting configuration, that say y a z. This is how, we have discussed where, sigma naught is sigma minus blank, a blank symbol. And I have constructed some turing machines to accept particular language.

We have discussed and how to construct those turing machines and all that. And, I gave some of some languages, some popular languages to known to you earlier, to construct a turing machine. Now, we will introduce the concept of giving output also by the turing machine. Here, I only thing is, I fix some convention and what conditions we say for a particular input x, the output is y, so that kind of convention, I have to introduce. Again,

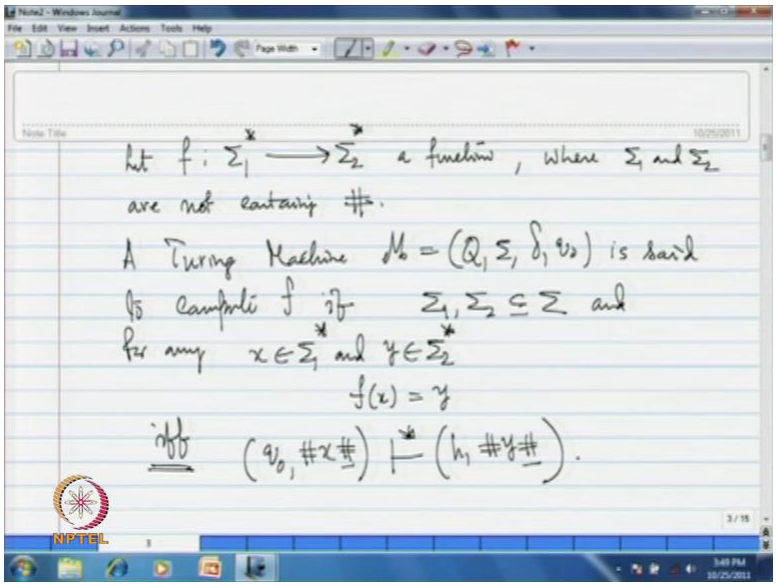
I give the input as earlier on the left justified right side infinite tape we have, this is the turing machine, I let me call it as, that the standard turing machine under discussion.

I give the input as earlier, say x is given as input and on the right side this blank, the reading and writing and we keep, this is the instantaneous description or configuration we call, this is how we start. And after finitely many steps, we will, whatever the output that you are expecting y, we should leave with this format and again the reading and writing at is here. By fixing this convention, we get certain advantages that we will discuss and you will realize after you know, one or two lectures that why we are fixing this kind of convention.

Because, we see lot of variance in the literature, the various books when you are looking at. You can see that, if you are expecting y as output, you may print y, in place of x itself, because by erasing x or whatever ultimately on the tape, if you are left with y, then you are happy. So, it is just a convention that the output how do you expect on the tape, because unlike in case of mille machines that we have discuss, there you have a separate tape for output and given an input on the input tape, the output is anyway shown up, the required output we shown up on the output tape.

Here, both input and output we have own for both of them, we have only one tape. Thus, we can have certain convention to report the output. Here, for my lectures, I fix this convention that the input will be given as usual, that we are giving in case of language acceptance, but the output I expect. Again, in the format that first cell should be blank and what are the outputs expected, that should be printed say y and then on the right blank, I will be halting with the halting state. In the beginning, we are with the initial state say q naught and you are, when you are halting, you will be halting with the halting state like this. So, this is how we expect the output.

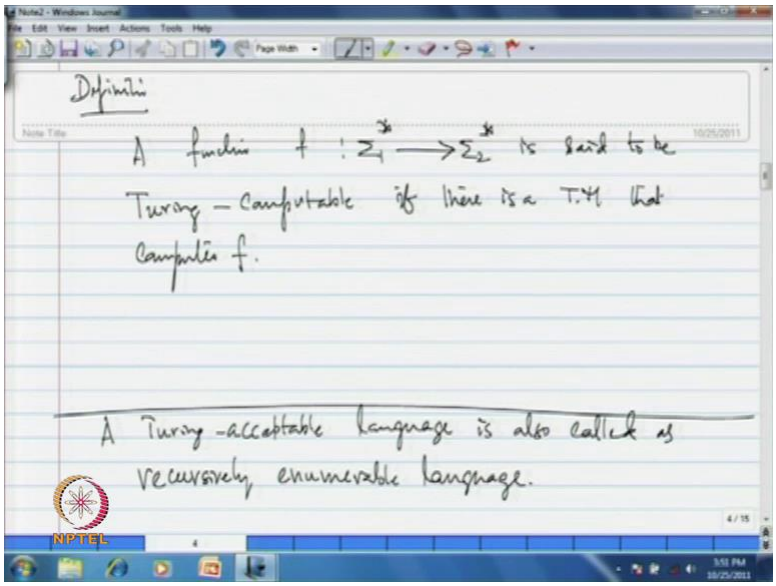
(Refer Slide Time: 07:30)



Now, let me formally give this definition. Let, f from sigma 1 star to sigma 2 star, a function where sigma 1 and sigma 2, two alphabet are not containing the special symbol blank. You take a function we say, a turing machine say M Q, sigma, delta, q naught is said to compute the function f, if this two alphabets sigma 1, sigma 2 are subsets of sigma and for any x in sigma 1 star and y in sigma 2 star, if y is image of x under f.

Then, it give x as input and you expect y as output, as per our convention and conversely. That is with the initial configuration, so if you start with something and by the time it halts and by the time it halts by leaving some output y. Then, that has be image under the function f, that is the meaning, if and only if. Look here, that x is, if y is image of x under f, then y has to come as output and whatever the output for any string, you are getting that has to be the image. So, that is why, we are writing if and only if. So, under these conditions we say the function f is computed by the turing machine n.

(Refer Slide Time: 10:25)

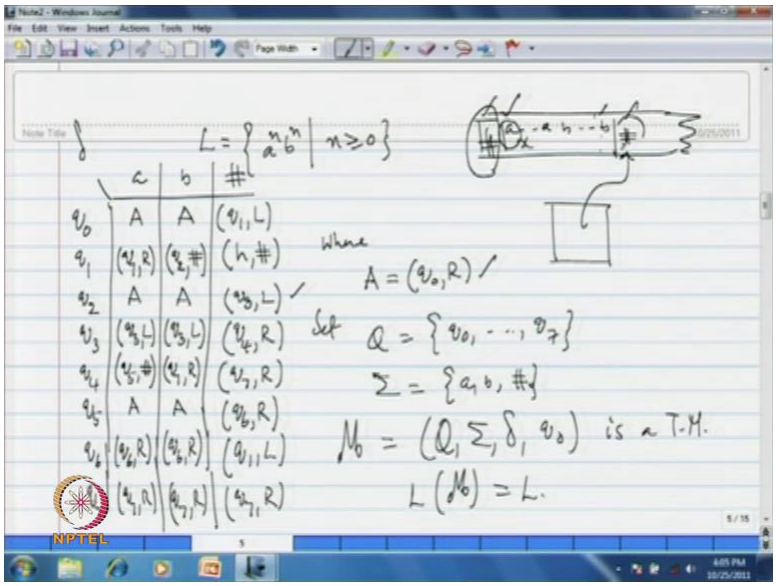


Now, a function f say sigma 1 star to sigma 2 star is said to be turing computable or simply computable computable. If there is a turing machine that computes f this, another definition. So, when do we say a two, a function is computable. Now, we have mentioned, when we say a turing machine computes f. So, this definition we have, now. So, to compute a function we have to construct a turing machine with this convention. Then that particular function we may call it is a computable function or more precisely turing computable function, because we are constructing a turing machine to compute that function.

So, this is another notion that, I talk about other than the language acceptance with respect to turing machines. And let me just point out, that the languages that are accepted by turing machines or turing acceptable language. I may say or I may also call, one may say that has recursively enumerable languages. So, let me just point out this, a turing acceptable language is also called is also called as recursively enumerable language.

Now, we see that we construct turing machines as of now, for two purposes; one, turing acceptable languages or recursively enumerable languages or for turing computable functions. Now, let me continue with few more examples, so that you can get familiar with the notion of turing machine or the constructing turing machine for certain purpose.

(Refer Slide Time: 13:13)



So, let me just take this your home work example, that is, a power n, b power n such that, n greater than or equal to 0, this I have asked due to construct a turing machine. What type of checking that we conduct the turing machine, because in, for this language when we have constructed push down automaton, you are reading of course, there from left to right? Because, there is nothing like moving left and right, we have only one direction moving that is, with the usual convention that moving from left to right side. As long as, you are getting is, you are putting a (s) in to stack, that is how we are remembering.

And once you start getting b (s), you are matching with the number of a(s), that are already in the stack, if there matching then you are accepting otherwise, you are rejecting there in case of push down automaton. Here, we do not have any other memory device, memory place here, like stack. Here, we have we can go backend forth on the tape and that is how, we will actually do the matching, yes. You see that if, a power n, b power n, so certain number of a (s) followed by certain number of b (s) are given. You simply go backend forth and keep matching with the symbols.

And, because there is nothing like counting and remembering, here, the things are given, it is something like, we just match them and understand, whether it is at the form, a power n, b power n or not . So, if the x is, some x is given as input, the first cell anyway as usual blank. If x is given as input; that means, say for example, certain number of a(s)

and followed by certain number of b (s), that is the required thing to be accepted and reading at and writing at is here.

You start with the initial state say for example, q naught and a take a left move just, if this is empty string that should be accepted, because when n equal to 0, a power n, b power n, this string is epsilon, that is the empty string, then it should be accepted. So, just to distinguish this blank, this blank and this blank, I will change the state. So, when I go to left side, if I get the blank in the beginning or in the beginning, then I will simply accept it, when you get b that is when the process starts, if you get a, clearly the string is in out of the form, a power n, b power n.

Therefore, we can put it in infinite loop or we can make it, you can make the machine hang. So, this first checking condition will be, if I get b what I have to do, that I have to pursue, I may do this, I will make blank cell here, I will go till this end and cross check, take a right move and cross check whether corresponding to this b, whether there is an a. If there is an a, then I will mark it, maybe I will make it blank, then I will go to right, till the blank, then I will take a left move and see, whether there is a blank and if there is blank then corresponded to this a.

So, there is only a, b as input, if there is another b, then I continue this process as earlier, if there is an a, again the input is not in the required form to accept. So, that is how, this loop I can continue and eventually, if it is in the required form, I have to accept in the halting state time. So, let me just, give you the transition map with this logic. Let me, start with the symbols of course, you are allowed to take more and more symbols in sigma.

Now, let me start with, in the initial state, I am reading blank, I will take a left move by changing the state to q 1; q 1 indicates that, I have started reading the input. If that is blank, then we can simply accept, so halt by say, let me print, there itself blank and halt. We can do whatever, you can go to, you can take a right move and halt, but we should ensure that we should not take a left move here, if you take a left move, it hangs. So, I is put it in halting state may blank, and there is another possibility that you may get a b, a positively, I mean.

So, in which case, I understand that by changing state say q 2 and I will make it blank. So, when I am in q 2, I know, that I am reading blank. So, I am not going to read any a or

b. So, this is the cell I require some information. So, I will change, say for example, to q 3 and take a left move, so that means, I have just mark b and I have to go till left end and I have to see, whether there is an a in the at the in the beginning.

So, this q 3 is indicating that, I have just mark b and I am going to cross check, whether there is an a. So, in q 3, I continue till the left end. So, that will be recognized by blank. So, if I get a(s) or b (s) in between on the way, I will keep going to left. So, that is q 3 left, even if I get b in q 3, I can continue. If I am getting blank, now I recognized, that is how this blank is required. If there is no such special marking here on the tape, we will go and hang, because what to cross check, because input is, if it is in the, from the beginning.

So, that is how, I hope now you understand that, why we require some special symbol in the beginning, when I am going, when I start reading the input from right and going to left. So, we required a special symbol in the beginning on the tape that is how, we have introduce this. So, once you receive blank, then I can take a right move and let me indicate that by changing a state, so a q 3 say, let me go to q 4 and take a right move. In q 4, what is expected, in q 4, we expect that, there has to be a.

So, I will look for that, positively suppose, there is an a, then I will change the state to q 5 and print and mark it that. So, we erase that by printing blank there. So, in q 5 of course, I know that, I will be reading the blank only, because just we have printed blank. So, let me change the state, so say for example, q 6 and move to right. And in q 6, now I have to go all the way till the end, till right end. So, if I get a (s) or b (s), I will continue on the tape by continuing on the same states, say q 6 or till I reach to the blank.

Once I reach to the right end, the blank you know that, in the beginning when I am in the initial state q naught, I take a left move by moving to the state q 1, so here, the same thing we can do. So, that it will continue to the loop. So, I will change to q 1 and take a left move. Again, now you see, if there is another b and corresponding to which, if there is a another a in the beginning of the tape, as per this whatever we have defined in q 1, you are reading b, change to state q 2. And in q 2, you are reading blank we know; because we have just a printed blank, no other symbol is possible.

So, q 3 we are changing to the state q 3 and moving to left. And in q 3, you keep continuing to the left and till you reach blank, because in the beginning we have the first

a, we have erased. So, there is a blank there. So, till that point, it will go and take a right move there, and positively we are expecting an a there. So, in which case, we are changing the state to q 5. And in q 5 of course, since we have printed just blank, we are reading blank. So, that change to state q 6 and in q 6, you can continue till right end.

So, use these loop continues, if we have say for example, three a(s) and three b(s), the loop will continue for three times. If you have say for example, four a ‘s followed by four b ‘s, the loop continuous for four times and so on. Now, let us look at other part. Now, for example, in the beginning here, if you get a, so this is the cell, that we have to look at. In this case, we put say for example, in q 7 and ask to move to right, because this is not desired, in the state q 1, I am not expecting to read a, for the desired in string in the language.

So, I will put it in the state say for example, q 7. And, in q 7 what I will do, I will keep moving to right, so that, the machine goes to that infinite and to the, it will done on in the infinite tape, say I will continue in q 7 and keep going to right. So, q 7 keep going to right q 7 keep going to right. So, this takes the machine to the infinite tape and it never halts, because there are you know, that is the right side infinite tape. So, in q 1, I am not expecting to root a. So, if I am reaching that, then this what, we are doing. In q 2, we have printed blank and therefore, in q 2 a and b, you would not get at all.

So, since delta is a total function, we have to define it as a total function, because for every state and the input symbol, we have to give definition. So, you give something arbitrarily, let me use the symbol a, to say that arbitrarily we define, let me give something later this thing. Similarly, in the initial state as per our convention, we start with the blanks cell, that is on the right side of the input. So, a and b are also not expected, because we are not continuing in q naught, once you come on to the input and changing the state for q 1. So, these two are arbitrarily I will define something.

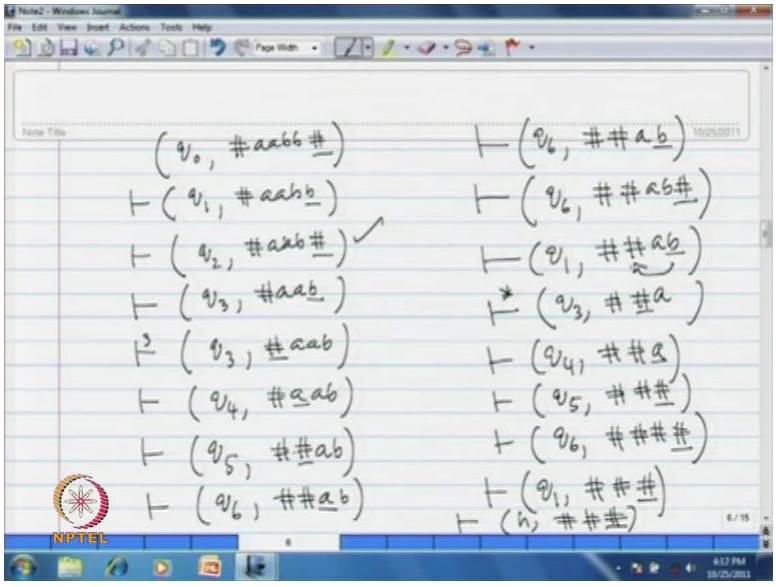
Now, look at in case of this q 4. So, what is the meaning of q 4, that I reached to the left end and took a right turn, if I get a, then I am; if I get b then, this is a mismatch for the pattern. So, now I can put it in q 7 and asked to go to right. Similarly, I have, if I have consumed a b, if I have noted a b, corresponding to that, if there is no a, if there is only blank. Then, also I will put it in the infinite loop that is how, I can do. And in q 5, q 5 means currently I am reading blank, I have just by, when you are changing to the state q

5, we have printed blank there. So, we will be reading blank only, we are not going to get a or b or any other symbol.

So, this two places, again you can define anything arbitrarily. So, for this A, something since we have to gives some state component you, let me put say q naught and something may be asking it to go to right or whatever. Because, the situation would not arise, wherever the cell I have written A, that situation would not arise. But, since by definition, delta is a total function for every state and a symbol, I should have some definition. So, let me say define, if this state component q naught and something I have to define, I am just saying R, so this is how, I declare.

And, with this definition, if you set the turing machine, now the state set is so where A is this. Now, set Q to be the states, q naught to I have till q 7, sigma is a, b blank, delta as defined here, we consider. M, this Q, sigma, delta is defined here, and q naught is a turing machine, such that the language accepted by M is the one, we have asked. What you do, you just do some computations on it and realize that, these precisely accept this language.

(Refer Slide Time: 26:33)



Let me just do some computations and observe this. So, in the initial state, if you are given say a a b b is the initial configuration. As per our definition, will go to q 1, blank a a b b and in the next step, it will print blank at that place, a a b this is easy situation. And, in q 2 of course, I read blank then I change to q 3 and go to left, having q 3 a keep

moving to this, till left end of course, here one step, I come on to this a and then next step, next a and come on blank. So, in three steps, I will be, let me indicate that, in three steps, I will get this kind this configuration and when I have blank, in q 3, I will change it to q 4 and go to right.

So, now the state is q 4 and I make a right move, so this is the configuration. And, in q 4, if I getting A, I will change it to q 5 and make it blank, q 5 and this is made blank, that is the situation. And in q 5 of course, I will be reading blank. So, I change it to q 6 and go to right, q 6 and we go to right; that means, the current cell is this. And in q 6, I continue to move to a right, that is how, it is defined. So, q 6 while leading a or b, we move to right. So, this is blank, blank a b, this is one step. In another step, also we will move, that is q 6, blank blank a b, now I go to this, so I will get blank here.

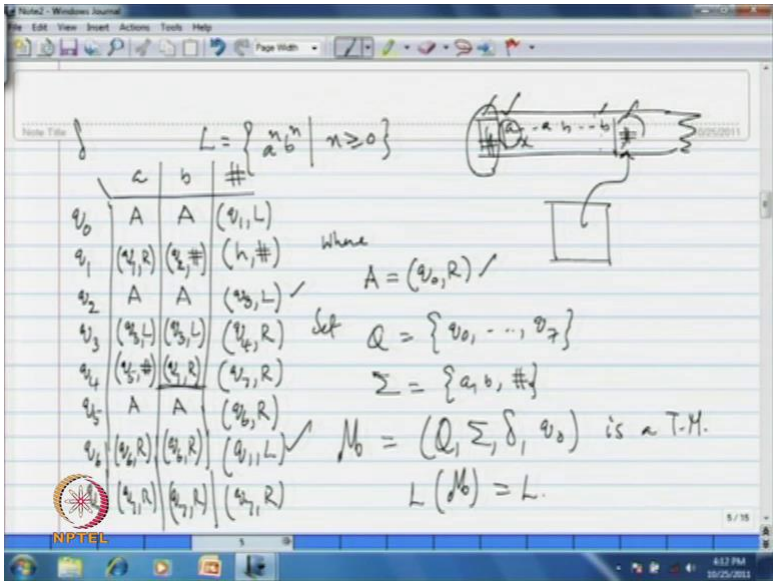
Now, in q 6 when I receive blank, then I change to q 1 and go to left that is the idea here. So, I will come to q 1 and take a left move. Now, as earlier when I read blank, it will print, when I read b, it will print blank there, change in to q 2. And, in q 2, I am reading blank, I will change it to, I will change it to q 3 and keep moving to left, till I go to this blank. So, here let me just, after finitely many steps of course, here making here, it is blank that is one and in q 3, I will continue to this position and at this blank… So, this indicates that I have reached it to this; I do not have b now, because we made it blank there.

Now, in q 3 again, I take a right move by changing to q 4 and now the current situation is this. In q 4, when I am reading a, we convert it to blank. So, that is q5, this situation. Now, in q 5, I am reading blank, it will change to q 6 and take a right move in q 6, it will take a right move, so you get this configuration. And in q 6, you take when you are getting blank with it will take a left move. So, that is now q 1, blank blank blank, this is the, because in q 6, when I am reading blank, the configuration, thus, there definition is will change to the state q 1 and take a left move, this is the transition.

So, by applying the transition, I get this. In q 1, when I have blank, then it halts by printing blank there. So, now in the next step, we go to halting state by printing blank in a third cell of course, this is the final configuration. So, with this computation you see, that a a b b is accepted and if you consider a a a b b b. So, a cube, b cube then the same

thing you can see that, this will continue, this loop will continue and finally, you will come to the halting state.

(Refer Slide Time: 31:57)



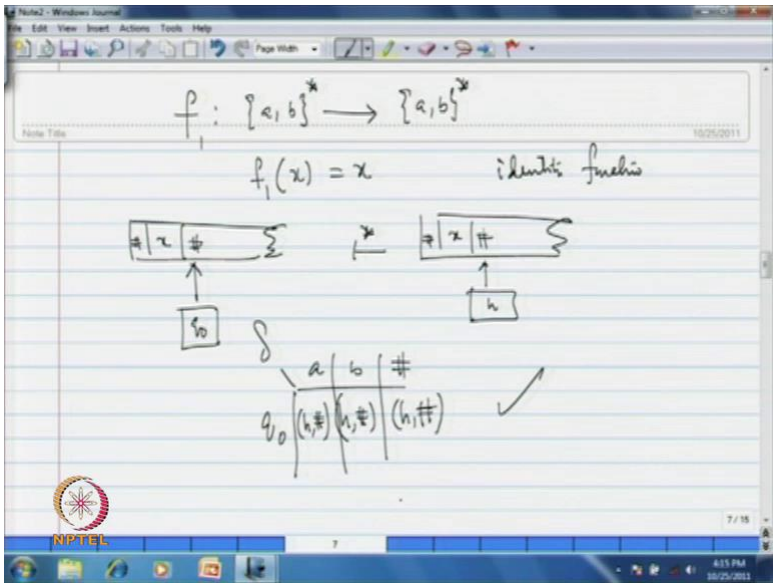
For example, if there is a string ending with a, you as per the definition here. In q 1, if you are getting a, I am going to q 7, then thus it will go to the infinite loop of going to right, keep going to right on the infinite tape or. For example, you have a b, but corresponding to you if which is there is no a, I will come to this situation, that in q 4, I am cross checking whether there is a. Otherwise I am putting in this and we are give defining this transition, that is q 7, R and thus it is going to again on the right on the infinite tape, we keep a, it will move on the infinite tape.

So, the string will not be accepted and what are all the other patterns, which is not of the form a power n and b power n. Because, this is the total function here, what we have defined delta is a total function. So, you give any situation of the input, we have defined something here. And, you see, what are the possibilities, that we have defined and then through which, we can argue that because there only finitely many transitions. And, the transitions are having certain pattern, because we have followed certain logic.

So, you can apply the pattern and prove that these transition will give, will this transition will accept the language, a power n b power n and greater than equal to 0. So, you look at the pattern of input and argue on and we can of course, prove that, this language actually the turing machine is precisely L. So, now you can of course, do this kind of

competition, just to realize that, if some other string is given as input, what is happening. Now, as I have introduced the notion of turing computable function. Let me now, give you a function and construct a turing machine computes that particular function.

(Refer Slide Time: 34:05)



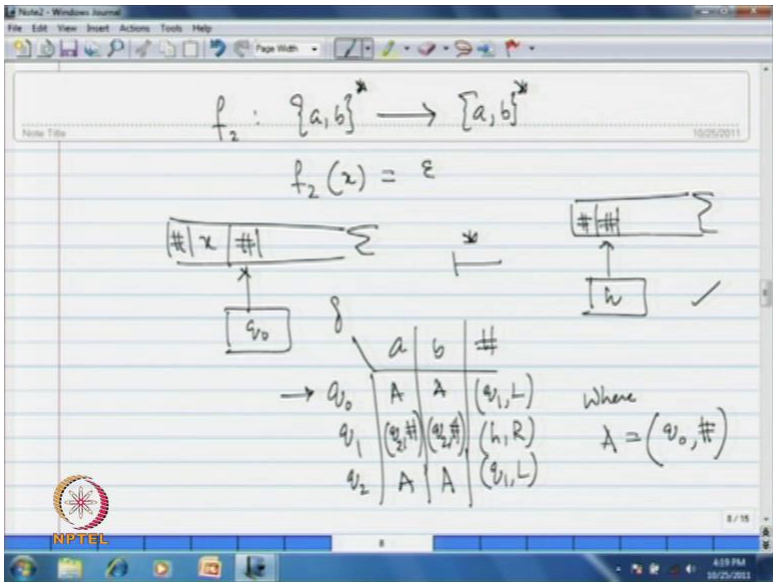
A simple function here, that is let me take f 1 from a b star to, I will call it as given a b star such that, f 1 of any x is equal to x, the identity function, so called. So, whatever is the input that you give, you have to do get the same output. So, here in this case, so the input is expected this way, as we have mentioned, but originally it is in q naught and what I have to give, after finitely many steps, I should leave this with by the halting state. We can quickly guess what sort of machines should be that or what type of transitions that we have to define there.

Very simple, you take one initial state and then immediately you halt it, because, what as the input, you want to leave, the same, as output. So, here I need not do anything here and we can pursue the job very quickly. So, the turing machine very quickly, one can easily say that, because the alphabet is this. So, the transition with one state, I can give and you define anything arbitrarily. So, I have only one state. So, I can yes, whatever that you are reading of course, I will read blank.

So, there I have to halt we know, but I should not take here, the chance of moving, I have to print blank here. Because, so that the reading and writing at their only. Here, also let me, just say anything arbitrarily you define here, that does not matter. We are not going

to encounter in the initial state q naught a or b, we are not moving the head. So, this transitions is there turing machine with this transitions precisely, compute this function as you understand quickly from the transitions defined here. As a very trivial turing computable function you see.

(Refer Slide Time: 36:23)



Let me now give this. Little non trivial, but of course, you realize that, this is also a trivial function u a b star to, let me say for the sake a b star only. And, now f 2, I define for all x, I will send it to the empty string. So, here what we have to do, whatever is on the tape, we have to erase it and leave that tape. Because, the epsilon when we are putting this as input for example, the obviously, the format is as per our convention, the first cell is blank. And, here I have put x and then on the right side of that particular input, I have to be there.

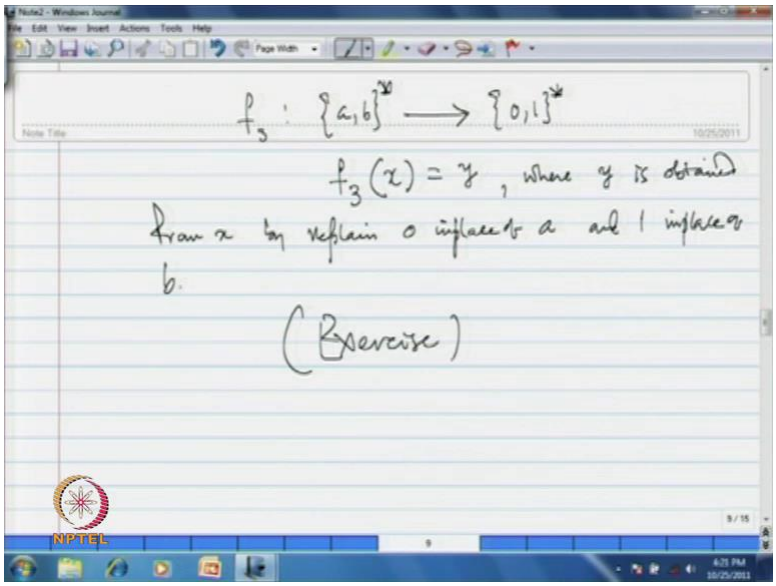
So, epsilon will be indicated by this, this is what is, epsilon on the tape will be indicated by this, as input of course, of course, as output here we have to see, so that means, if you start with the tape, of this format we starting here. So, initial state q naught after finitely many steps will come to the halting state by having you know, having of this configuration, how do we defined this. You just take a left move as long as you are getting a's or b's keep erasing it, once you get a blank, you just take a right move and simply halt, this is the logic, I will define it here.

So, the delta transition function of the turing machine, I am going to construct take say in q naught, just to distinguish that I have started the process. So, I will change it to q 1 and take a left move and in q 1, if I get a, say I will go to q 2 make it blank or even if I get b will go to q 2 make it blank. In q 2 of course, I am reading blank, so I will change it to q 1 and take a left move or I can of course, write in instead of writing here, the transitions q 2, I can say q naught and does not matter.

Because, we can reduce the number of states there, I am not worried about to how many states that I am using, at this point of time. So, I will simply say that, I am changing to the state q 2 and in q 2, I am reading blank. So, it will again to pursue this job, I change it to q 1 and take a left move. In q 1, when I get again a, it changes to q 2 and print blank and b prints blank. In q 1, if I am reaching to this state, I am in the situation, having blank, then I take a right move and halt, so that, I have erased everything. In q 2, since I am reading blank, I have only this possibility, and similarly, in the beginning, I do not have to read any non empty cell, only blank cell.

So, here here arbitrarily you define anything, because we have to define something, as delta is the total function in the definition, say here, a can be something. So, with this transition function, whatever the turing machine I construct, because the states are clear. There are three states, symbols are there and the delta I have define, declare q naught is initial state. Now, this turing machine clearly pursues the job, that as desired, to show that the function f 2 is a turing computable function. This machine simply erase as the tape and leaves the blank tape, not blank tape essentially, the as epsilon as output. So, these are little understandable very quickly and little trivial.

(Refer Slide Time: 40:29)



One more for the sake of understanding this, let me consider for that the change of in the situation. So, let me use 0, 1, star. And, what do I do, f 3 of x is y, where y is obtained from x by replacing 0, in place of say a and 1 in place of b, wherever you have b, you place it with 1, wherever you have a, you place it to 0. I hope, that is very simple phenomena and you can understand this function, how this is defined, for epsilon the out, the image is epsilon. So, what essentially you have to do, you keep going to left, as long as, you are getting a, whenever you get a, you print it print 0 there.

Then, you are getting 1, you print, whenever you are getting b, you print 1 there, till you go to the end of the tape left end that is indicated by blank. Then, you take a right move and keep going to right move, till a reaching to the n. So, take this an exercise and realize that this is a turing computable function. Now, the point is you look here some of this, because so far if we, I have handle just very simple example, say that turing computable function or turing acceptable languages. In just in case of, a power n, b power n itself, I have used about seven states and any other language.

Now, you can try for, all those palindromes you get by the, in a similar way you can try or I have ask to construct a turing machine to accept the language say, a power n, b power n, c power n, greater than or equal to 0. So, you require one more sort of check, it is not just reading after b, we have check for a, if it is a power n, b power n, c power n. Then, you have to see that corresponding to each symbol, rather there is in this respect to

pattern or not, those things we have to cross check. And, as long as, the symbols are increasing, or the checks, the loop, whatever, that we define, we are choosing certain new states and that is, how we are creating the memory.

And for simple examples, we see that, you here, you will require several states. Sometimes, you know for very simple examples that we have already constructed for regular languages or contextual language type of things themselves. You may have to require some twenty states, thirty states like that. And, that transition function you know looks very complicated, because you have to go through carefully, the transitions are carefully defined and the places have everything.

Now, if you look at some sophisticated, more complicated languages are computable functions. The type of turing machines that you would, you are going to construct will be more and more ((Refer Time 44:09)) to see. You would construct and you may verify, you can always prove that, whatever that transition functions, that you have constructed, does the job, the result job, but the point is, if there are, for example, multiplication, addition and this kind of things you know, you can do it using computer.

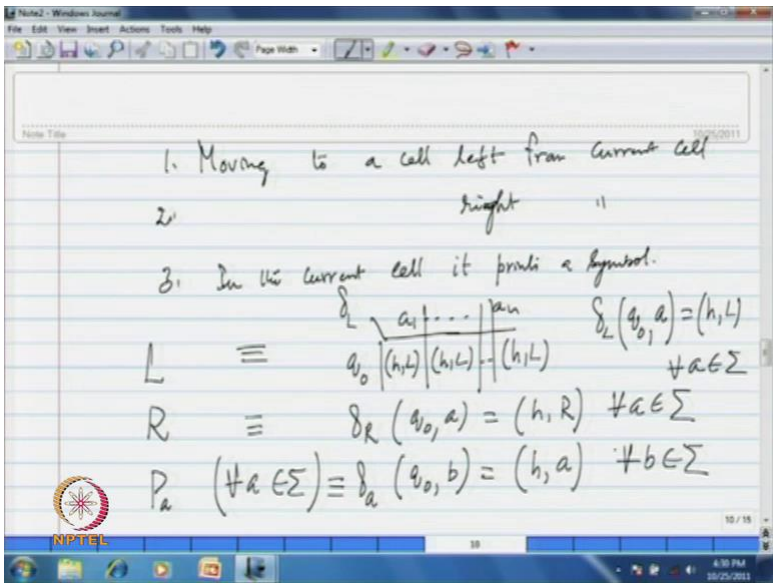
So, this can, as we have mentioned that, this turing machine is essentially taking care of the things that you can handle using usual computer. There are several things that you see, that you can do it using your usual computer. So, for certain simple phenomenon, if you have to a require, say for example, hundred states, two hundred states and one lakh are of that order, then it will be very difficult to really write and see. So, in this direction, I will introduce a short hand and as a notation, short notation while constructing a turing machine, we will follow that notation and try to decrease this complexity in seen or in understanding the construction of that particular turing machine.

First, in the direction, first let us first realize, what are the basic turing machines possible? And, first… And, what are the turing machines that we construct and those turing machines we further use to construct some of the complicated machines, wherever it is required. So, this is the logic, that we follow to construct, to introduce a notation or short hand for constructing a turing machine. In the direction, first we realize, what are the basic turing machines and some of the basic turing machines, which you may quite often you use, I may give some fixed symbols for that.

And those symbols, whenever you recall, we assume that, this turing machine that you have already constructed and now, I can make use of that. Like, in your programming, you would have written certain modules, certain procedures and in bigger a program and whenever you require, you can of course, call it and continue in this. But, only thing is here we have to take care that, whenever you are connecting to a particular turing machine, the pattern of the input is essentially as required to the machine that you have constructed earlier and whatever that the output, that you are getting or whatever it is accepting or whatever.

So, the time it is actually halting, where we are actually leaving the tape, that is very important. And, now I can make this point clear, that why we have to follow a fixed convention here, because starting here and ending here. That kind of convention, what we are following, so that in this particular call, for this particular cause, this is, this convention is useful.

(Refer Slide Time: 47:10)



Now, let me start with, looking at basic turing machines and then we will try to see that, how to construct certain complicated turing machines with using this easy notation. One can quickly realize that turing machine, the basic operations it is doing as it is moving of course, it is reading and writing here. Moving to a cell left, from the current cell or it is moving to a cell right from the current cell or in the current cell on the tape, it would

print some symbol. These are the three possible transitions, current cell, it prints a symbol.

So, these are the three things, the basic things that the turing machine is doing. Now, if I wanted turing machine, wherever it is currently, it has to just move one cell left and halt. Let me call the turing machine with this L, this is the name of the turing machine and similarly, if I write R is the turing machine that takes a right move from the current cell and halts. And, for all a in sigma, I write P a, the turing machine that prints a in the current cell and halts. Defining these turing machines is very easy, is not a big deal, because for L you can construct the turing machine like this, this transition.

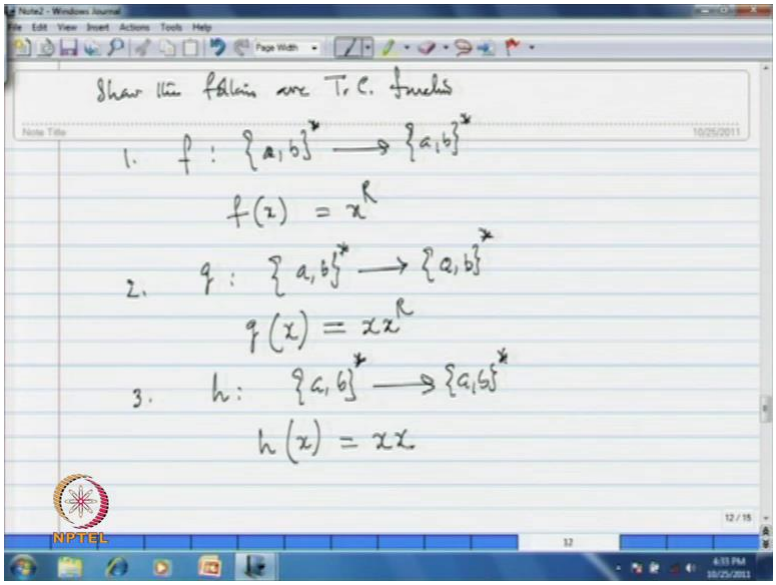
So, let me call it as delta L, what are the inputs symbols let me, a 1 a 2 a n say, what are all the symbols that you have. I take one state say initials, the initial state q naught, wherever I am, I will just halt by taking a left move that is how you can define and so on. So, that is the transition function delta L is defined such a way that, this q naught at any symbol a, this is halt by taking a left move for all a in sigma, this is the turing machine. Similarly, one can look for delta R that you can define, take one states that is the initial state for all symbols, it define it as halt by taking a right move for all a in sigma.

And now, if I write here, delta a here, I take one state, that is the initial state, what are the symbol currently I am reading let me use now, say b, a variable here. I halt by printing a in the current cell for all b in sigma. So, if you define things like this, this turing machines precisely, they do whatever that is required as mentioned here. The turing machine L, whatever it is the current cell from which, it will take one left move and halts. And, the turing machine R that takes right move and halts from the current cell. P a, the turing machine which prints a in the current cell and simply halts, whatever it is reading in the current cell.

So, these are the basic operations are, this is the basic type of transitions that we are realizing in a turing machine. So, these things are now, I have used, L R P a for this basic turing machines. So, using these basic turing machines, how do construct certain complicated turing machines and what is the formal way of constructing, then everything, that we will discuss in next class. Now, since we have discuss how, what is the turing acceptable language, what is the turing computable function, how to construct

a turing machine. Using a straight phenomenon, you construct certain turing machines and see that, how many states actually; you are requiring and all those things.

(Refer Slide Time: 51:54)



So, let me give certain problems. So, f from say, a b star to a b star, you define ((Refer Time 52:07)) say f of x is equal to x power R, reversal of the string and prove that, this is turing computable functions. So, we receive this or it may take say g, a b star, example 0 1 star, g of x take any string that you create, let me put, so a b star only, it is put x x power R, say h, a b star to a b star, x x of course, will you few more examples.

And, these are the very simple in this line up and you see that, these are for the simple examples, how many states that you have requiring and how the notion of giving a short notation to construct a turing machine is useful, that you will be realize. So, construct turing machines to show that these functions are truly computable and certain similar functions, that you can take and construct turing machines. We will discuss construct giving short notation in the next class.