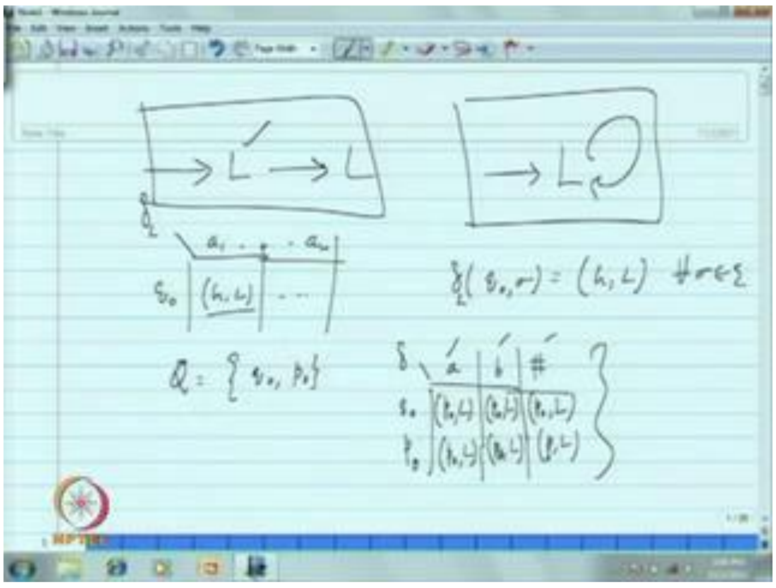
Formal Languages and Automata Theory
Asst. Prof. Dr. K. V. Krishna
Department of Mathematics
Indian Institute of Technology, Guwahati

Module - 11
Turing Machines
Lecture - 4
Multi Input

Yes, so far I have introduced the concept of Turing machines, Turing acceptable languages, turing computable functions and for combination of Turing machines for combining small Turing machines to construct a complex or larger Turing machines. And we introduce the concept of machine schema and a machine schema represents a Turing machine which is a composite Turing machine based on the some of the smaller Turing machines. So, this we have I have introduced and in the previous lecture I asked you to understand that when we are taking two components of same Turing machine and if you are combining in a particular way.

And if the state sets states are common some states are common what kind of problem that you encounter. So, this I ask you to look at and see what is what will happen when in such a situation and why we are maintaining the state sets or disjoint. So, let me give, so more some information on that looking at that exercise and then we will further discuss on some of the issues related to this complex Turing machines.

(Refer Slide Time: 01:43)


$\rightarrow L' \rightarrow L$
$\rightarrow L$
$Q = \{s_0, p_0\}$
$\delta(s_0, \sigma) = (h, L)\ \forall \sigma \in \Sigma$

In this example when I said if you combine L with L and if you consider because you known this L is the Turing machine whose transition map as we have whatever the input letters you have. Whatever is the input letters a 1 a 2 an and in all those input letters you can simply take a left move and halt this what we define. So, that is delta of q naught sigma is halt by moving to left for all sigma n sigma. Now, when we are combining like this if you consider here also q naught and here also q naught, what is the situation and how this composite machine will be. So, in case I take this and now if I combine see the new this composite Turing machine the states as per our this thing this is q naught will be there and one p naught you will be choosing.

So, these are the states and sigma as original and delta we do it like this as per the definition because the new delta, let me call this is delta L and the delta what we define here you have q naught. Let me assume for the time being a b blank or the input letters and see what will happen. As per the original this thing where it is halting I have to put it in p naught and then I take a left move and here also I will put in p naught and take a left move and I put in p naught and take a left move.

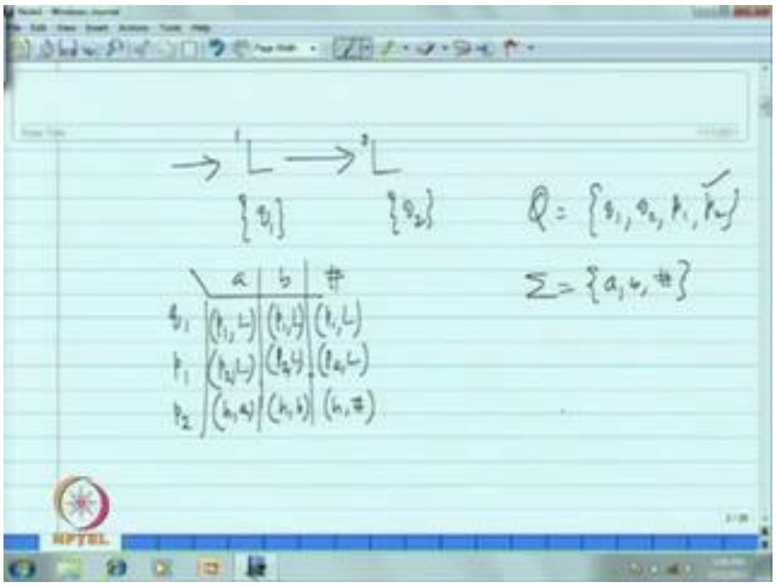
So, this is what when this what is the first performance whenever it is halting the first component we have to do that and in p naught whatever that machine is doing that is what we have to do. So, that is we have to do, so in L now that is going to q naught. So, what are the performance of that q naught that we have to do.

So, in p naught in p naught what you have to do whatever that this is doing, but as per the definition again here again here. That is because it is going to h L there is only one you have consider, so again this will be p naught L because it is going to halt in configuration this is p naught L the p naught L. So, what we happen with this what will happen with this kind of transitions it will take a left move in the beginning. And it is connected to p naught and in p naught if you get any of these symbols again you are going to left and you are continuing in p naught that is what we are doing and there is no situation that you are getting halting.

So, essentially this is when I am taking the same state set the situation is essentially this not this not as desired. Because here essentially you require only two left move sit is behaving like the Turing machine which always takes left moves on the input and of course eventually this will hang.

This is not the desired construction, so for this for this is not the way that if you consider same state set.

(Refer Slide Time: 05:19)



$$\rightarrow {}^{1}L \rightarrow {}^{2}L$$

$\{q_1\}$ $\{q_2\}$

$Q = \{q_1, q_2, p_1, p_2\}$

$\Sigma = \{a, b, \#\}$

	a	b	#
q_1	(p_1, L)	(p_1, L)	(p_1, L)
p_1	(p_2, L)	(p_2, L)	(p_2, L)
p_2	(h, a)	(h, b)	$(h, \#)$

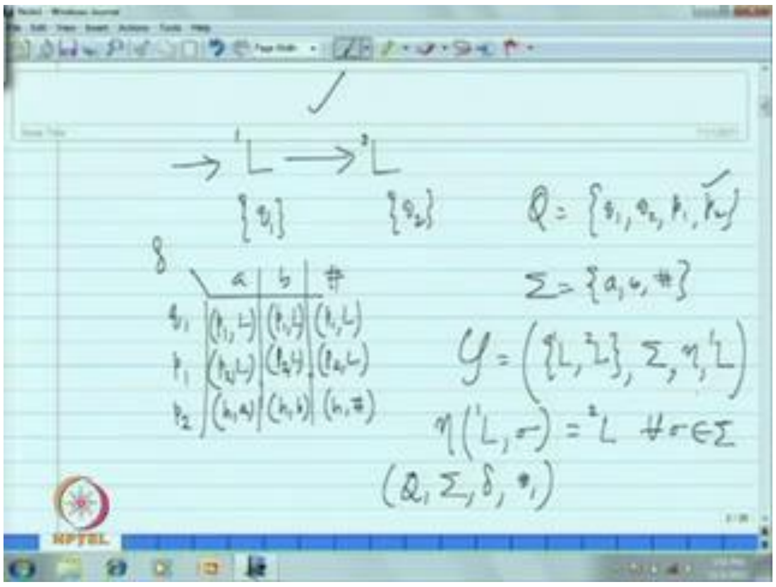
This is not the way that we have to do, so in which case what you do for L L you are taking two copies left machine. Let me denote this is number one and the second copy in which case for example if you take this is say q1 and for this you make q2 and. Now, you see that as when we are defining this q1 is there in the definition what are the symbols? let me assume a b blank for the time being. So, as it is halting you will connect you will put it in say in the first machine that is let me call p1 is a state and take a left move. And here in b again that I am putting in p1 and left move t1 and left move and for p1 how do we define that is p2.

Corresponding to this I am taking nozzle this state says a the state set, now is corresponding to this what are the state sets of this q1 q2 corresponding to the first one, let me take p1 for this p2. So, this p 2 L and this is p 2 L, this is p 2 L because in the second machine the definition is again it is connected to initial state and in the initial state of the second machine. This L2 the q2 in the that is the initial state in q2 when you put any symbol it is going to the left the definition is the same and this is what the left machine. So, I will again have to put because it is a halt it is going to the halting state I have to put it in the auxiliary state of the second machine that is q2 here that is p2 here, so I will be putting in p2.

Now, you cross check for this machine there is no definition of eta and when you come to p 2it will simply because as there is no definition whatever is the current symbol that we print the same symbol and go to the halting state. So, that is the construction halting state blank, so in general here I have just resumed a b blank or the input symbols, but if you take any alphabet. So, we essentially copy the same and you look at, now the difference between taking the same state set because when I am calling L because for the short notation I have just in the beginning I said L connected to L there are two copies of L.

But, you have to maintain a different states and the when you are taking a particular machine if you have using it for several times you take copies several copies of that machine that means essentially the state set you take different labels of the states. So, that you can maintain the disjoint states at the components on you are combining then it pursue it pursues the property where as desired otherwise. There is a problem that you know you may construct a wrong Turing machine with the machine schema. So, here the machine schema I hope it is clear to you that here you have two machines in this machine schema when you are looking it.

(Refer Slide Time: 08:32)



$$\rightarrow {}^1L \longrightarrow {}^2L$$

$\{q_1\}$ $\{q_2\}$

$Q = \{q_1, q_2, p_1, p_2\}$

δ

	a	b	#
q_1	(p_1, L)	(p_1, L)	(p_1, L)
p_1	(p_2, L)	(p_2, L)	(p_2, L)
p_2	(h, a)	(h, b)	$(h, \#)$

$\Sigma = \{a, b, \#\}$

$\mathcal{Y} = (\{{}^1L, {}^2L\}, \Sigma, \eta, {}^1L)$

$\eta({}^1L, \sigma) = {}^2L \quad \forall \sigma \in \Sigma$

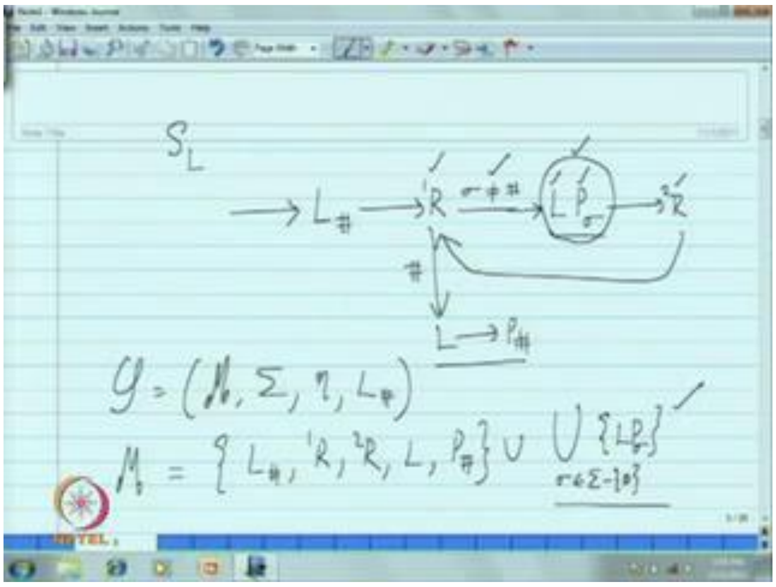
(Q, Σ, δ, q_1)

Because this Turing machine actually represented by the machine schema here the Turing machines are L1 L2 there are two copies of left machine and the sigma the common alphabet for all that is what is here the eta the partial pressure map. We will now indicate and this machine is the initial machine for this machine schema where eta of L1 for all symbols for all sigma is connected to the second left machine this is for all sigma in sigma and after L2 there is no definition.

So, this is the partial function for only the machine L1 the L1 I mean here the left the left machine the copy the first copy here. So, this what is the definition of eta and with this machine schema we have constructed this machine and this machine precisely takes only two left moves on the input and the corresponding transition map is this. Because this machine schema represents the Turing machine whose state set is q and of course alphabet is sigma and delta as indicated in the table. And the initial state here is the initial state of the first machine that is q1.

So, this is the Turing machine this is therefore Turing machine that is represented by this machine schema. Now, let me give little complex example is a very simple example that I have indicated this I hope clearly distinguishes that when you are combined when you are taking a particular Turing machine. Several copies if you do not distinguish the states disjoint state set if you do not maintain what kind of problem will come that I hope it is through this example I hope you understand that.

(Refer Slide Time: 10:33)



S_L

$$\begin{tikzcd} {} \arrow[r] & L_{\#} \arrow[r] & {}^1R \arrow[r, "\sigma \neq \#"] \arrow[d, "\#"'] & L P_\sigma \arrow[r] & {}^2R \arrow[ll, bend left] \\ & & L \arrow[r] & P_{\#} & \end{tikzcd}$$

$\mathcal{G} = (\mathcal{M}, \Sigma, \eta, L_{\#})$

$\mathcal{M} = \{L_{\#}, {}^1R, {}^2R, L, P_{\#}\} \cup \bigcup_{\sigma \in \Sigma - \{\#\}} \{L P_\sigma\}$

Now, let me recall the example that left shift machine I have constructed the previous lecture that is because we have to move all the things all the things input one to its left. So, remember that we have constructed like this blank it takes a right move and then sigma not equal to blank whatever is there you just take a left move and then print that sigma.

And then take a right move and then take a right move and, now you take on more right move whenever you are getting blank you just take a left move and print blank there. This is how we have constructed and you see here there is a machine R here once and here is R and here is a left machine here is a left machine that you can clearly see that we machine L is here L is also here. Now, here is one typical situation that when your reading sigma here under the sigma what are the sigma that you are reading at this place. That this thing that you have to print after this much situation like you have to take you have compose with these Turing machine.

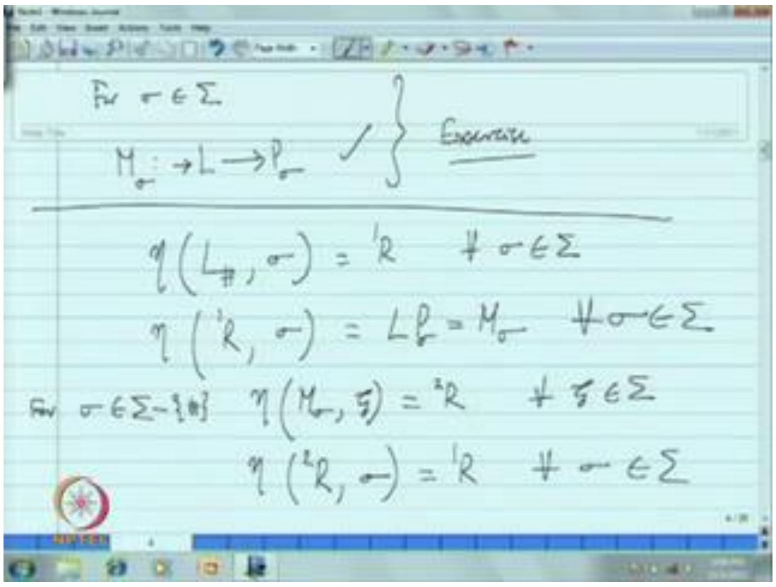
And then you have to remember what sigma is taken over and you have to print that here. So, one has to understand that under the sigma this component need to be constructed. So, this component let us construct it separately and call that machine one different machine schema and this composite machine has to be called depending on that sigma. So, thus here and of course here there are R here is 1 R this is another R and of course L hash you have only one and there is L and this I am anyway combining. So, thus you see what are the components here.

Let me call this is one right machine and this is the second copy of the right machine, thus in the machine schema the machines the basic machines if we call that is S. This is M the common alphabet whatever that you have sigma eta will define and the first machine here. As you can see that L hash and where this M the set of machines having L hash as one component here and you have machine R1 let me write and similarly, you require another copy this R2 you call. And now here you can see something like because this L and anyway I am going to combine this.

So, one left machine you would require and printing blank this machine you would require this as indicated here union for all non blank symbol you require this machine. That is union the machine L compose with machine l compose with p sigma this sigma minus blank in fact these two together this two together and I can represent this at once anyway does not matter. So, these are the machines that we are seeing whatever the machines here I am writing first in this particular place these.

Things you have to first declare that means L p sigma this is the machine schema for that machine that let me write this as M sigma and you can also combine for sigma in sigma and the you can call that also here there is no problem. So, for sigma in sigma in which case I may just represent it like this L p blank. So, I do not require this here in that case here for all sigma and sigma, so let us represent it this way.

(Refer Slide Time: 15:12)



For $\sigma \in \Sigma$

$M_\sigma : \rightarrow L \rightarrow P_\sigma$ ✓ $\Big\}$ Exercise

$\eta(L_\#, \sigma) = {}^1R \quad \forall \sigma \in \Sigma$

$\eta({}^1R, \sigma) = LP_\sigma = M_\sigma \quad \forall \sigma \in \Sigma$

For $\sigma \in \Sigma - \{\#\}$ $\quad \eta(M_\sigma, \xi) = {}^2R \quad \forall \xi \in \Sigma$

$\eta({}^2R, \sigma) = {}^1R \quad \forall \sigma \in \Sigma$

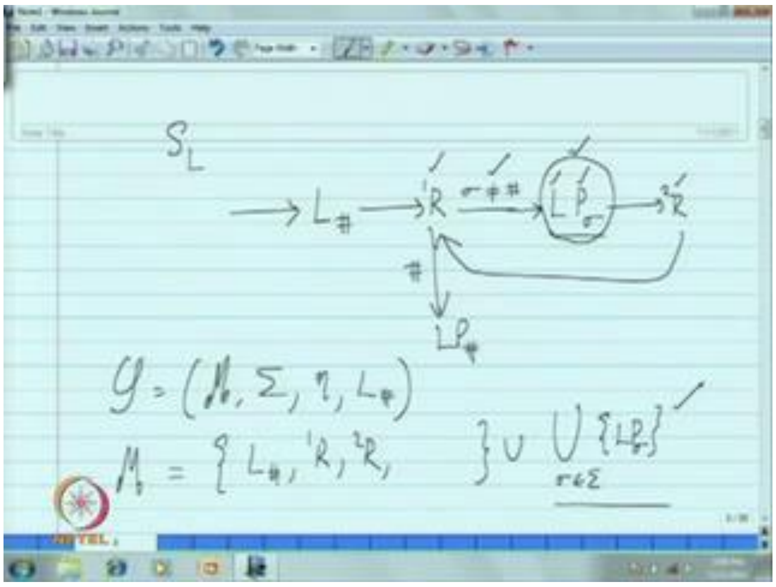
And now see that how to write this machine schema is clear to you because there is one machine L and for fixed sigma you have to take p sigma and combine that. So, how many such machines that you have to declare for all for each symbol of sigma you have one such machine. So, you take this an exercise to write the machine schema for this write the machine schema for this, now those machines are now available in this particular list this is a finite set of Turing machines.

Now, for this for this machine schema the eta, now will define like this the L hash for all symbols it should be connected to this one R1. So, the first machine the machine schema has to be like this sigma is connected to the first for all sigma in sigma. And now you look at here for R1 for non blank symbols I am connecting to this for blank symbol I am connecting to this whatever it is for all sigma this will be connected to p sigma. So, I declare it like this, so eta this R1 whatever the sigma that you are getting whatever sigma that you are getting you will connect to that L p sigma this is we named it as M sigma.

So, this is what is M sigma that you have constructed separately, so you connect it to that machine because this for given sigma for sigma. And sigma this how of course this for all sigma and sigma one another is same for all sigma and sigma we have to connect to M sigma this is the situation. Now, n sigma for non blanking that will be connected to this, but not for blank.

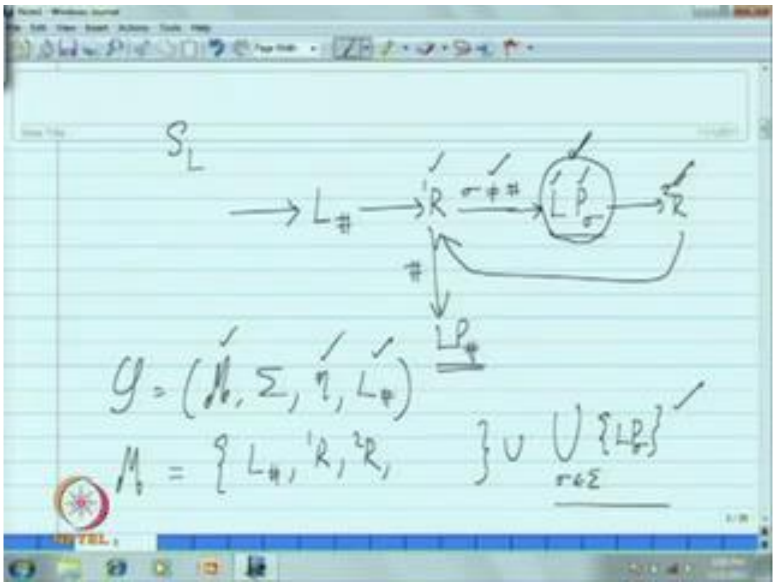
So, here for sigma in sigma minus blank this eta will be defined for M sigma for that n sigma only M sigma for all symbols, now let use some other say zeta is. So, for non blank symbols this is connected to the second copy of R second copy of R this is for all this zeta n sigma because this is connected unconditionally this is connected to this unconditionally.

(Refer Slide Time: 17:56)


S_L
$\to L_\# \to {}^1R \xrightarrow{\sigma \neq \#} LP_\sigma \to {}^2R$
$\#$
$LP_\#$
$\mathcal{G} = (M, \Sigma, \eta, L_\#)$
$M = \{ L_\#, {}^1R, {}^2R, \} \cup \bigcup_{\sigma \in \Sigma} \{LP_\sigma\}$

So, but this is connected only when the sigma is non blank that how is indicated here for non blank symbols we are connecting it this way. Now, this case let me indicate this here for all sigma and sigma earlier. Now, this R the second copy of R will be connected to first copy of R unconditionally I mean for all symbols essentially. So, that is another definition we require the second copy of R is connected to the first copy R for all symbols for all sigma and sigma and now there is no connection from this. So, this is a this is undefined here, now the set of machines is declared and the initial machine is declared here.

(Refer Slide Time: 19:03)



S_L
$\rightarrow L_\# \rightarrow {}^1R \xrightarrow{\sigma \neq \#} L P_\sigma \rightarrow {}^2R$
$\#$
$LP_\#$
$\mathcal{G} = (\mathcal{M}, \Sigma, \eta, L_\#)$
$\mathcal{M} = \{ L_\#, {}^1R, {}^2R, \} \cup \bigcup_{\sigma \in \Sigma} \{LP_\sigma\}$

And sigma is the common alphabet as indicated earlier, now this eta is as defined here this is the partial map eta given here. Now, this machine schema this machine schema precisely represents here you have to here you have two things to note one is taking R2 copies this R1 R2 and because we are carrying some information on this particular edge. This is a particular place what we have to do we have to first construct this composite machine and here you may call that as M sigma what are the sigma non blank thing that you are carrying that M sigma will be connected to this. So, there is all the Turing machines here under consideration.

So, this is essentially these are all M sigma, now how many Turing machines are there in this M there are one two three and here mod sigma. So, essentially in the size of M is size of M is mod sigma plus 3 this many Turing machines are here. And appropriately depending on the sigma that you are getting it here you see that you may see that here there one two three four only five machines are there as a basic machine components here. But, there are not actually five this is depending on sigma that we whatever that that is connected to here this is represented in this variable.

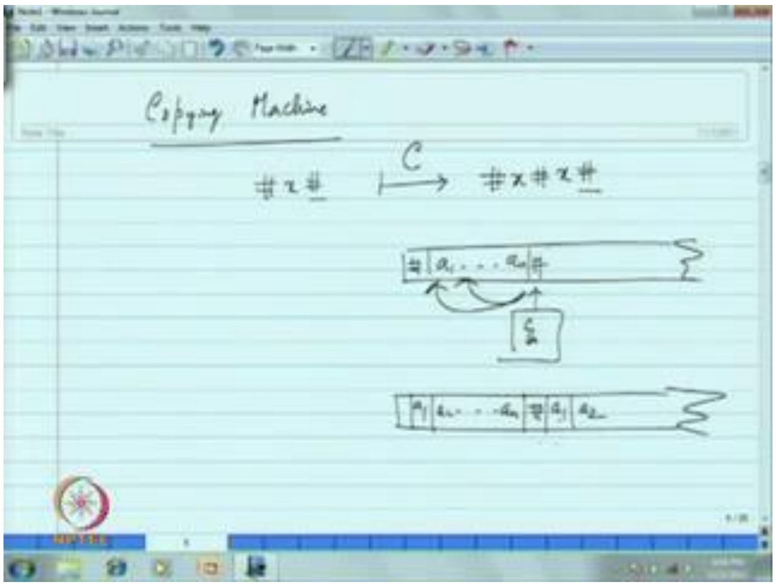
So, the Turing machines in this machine schema is number of symbols of sigma plus 3 these three machines are there. So, this is the machine schema of the left shift machine and in short notation we are representing it that way the way that I have constructed earlier likewise. What are the Turing machines we give we have to be very careful to look at the correspondence with the proper definition the basic definition the.

In terms of the state because the diagram intuitively everything will be clear and construction is very easy very compact and very nice representation that we have. But, the problem is when if you want to actual refer to the definition and what are the respective states if you want to write you have to be very careful with this machine schema. Because the components essentially we have to maintain the disjoint set of states and you have to be more and more careful how to represent this kind of situation.

This kind of situation composite machine which is called, now you can now treat this is one of the building block and accordingly you have to declare the machine schema. Otherwise you know if you separate it here in this present machine schema it will be very difficult for you to represent it here.

It is in fact not possible because this sigma the information of sigma on this particular at this particular place has to be carried. And therefore this has to be combined prior to declaring the machine schema of this SL. Now, let me give one more example their sigma information may be quite.

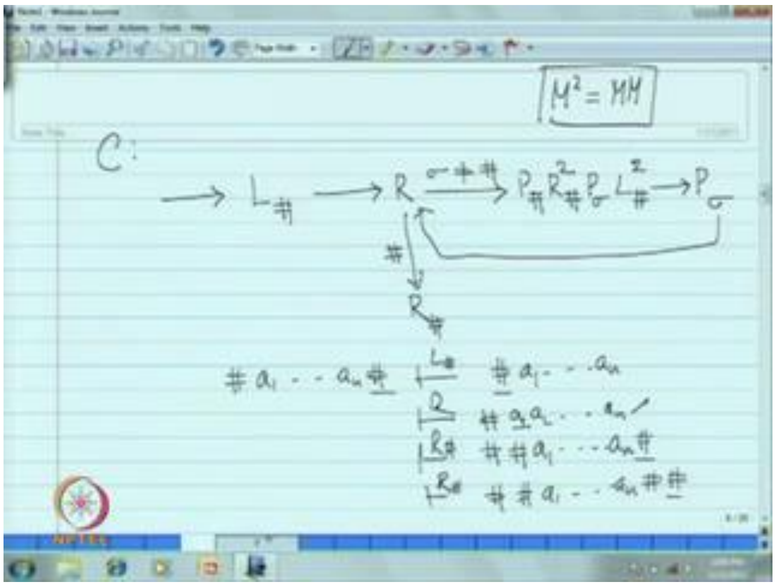
(Refer Slide Time: 22:22)


Copying Machine
$\#x\underline{\#} \overset{C}{\longmapsto} \#x\#x\underline{\#}$
$\# \; a_1 \ldots a_n \; \#$
q
$\# \; a_1 \ldots a_n \; \# \; a_1 \; a_2$

So, this is so called Copying machine, so this Copying machine how it performs you give any string x as input is the format that we are following we want this kind of situation output the same x separate it by one blank will be declared. So, that the copy is shown very clearly, so this copying machine if I write the machine C how do we write because here the logic will be. So, the input may be given like this say a 1 a n this is where your starting, so finally you should be left with this kind of tape. So, the logic one may look at like this because at this first place I have to have this a 1, so that means I have to go all the way till this place carry this information and copy it here.

And then you go to the second position from here all the way till a 2 and copy that and to a position like this. So, this a 1 a 2 a n then there are there you from this position you go to this place you can understand that till go till blank and take a right move. And what are the first symbol go till end of tape and print it there and so on you keep doing this till you reach till this blank. So, this is the logic one can easily follow to construct such a Turing machine.

(Refer Slide Time: 24:08)



$\boxed{M^2 = MM}$

C:

$$\begin{tikzcd}
{} \arrow[r] & L_{\#} \arrow[r] & R \arrow[r, "\sigma \neq \#"] \arrow[d, "\#"'] & P_{\#} R^2_{\#} P_{\sigma} L^2_{\#} \arrow[r] & P_{\sigma} \arrow[ll, bend left] \\
 & & R_{\#} & &
\end{tikzcd}$$

$\# a_1 \cdots a_n \underline{\#} \vdash^{L_\#} \underline{\#} a_1 \cdots a_n$

$\vdash^{R} \# \underline{a_1} a_2 \cdots a_n$

$\vdash^{R_\#} \# \# a_1 \cdots a_n \underline{\#}$

$\vdash^{R_\#} \# \# a_1 \cdots a_n \# \underline{\#}$

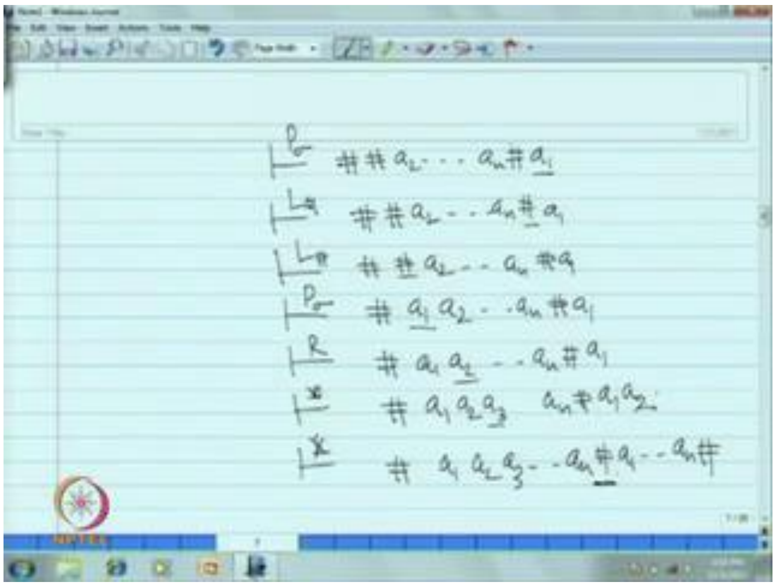
So, in the beginning L hash go till end of the tape left end take a right move. Now, what are the symbol that you are reading if it is not blank then let me print blank there and then I will take R hash two times the indicating R hash square I mean this I am applying two times.

So, the abbreviation M the Turing machine, this abbreviation I am following M square is M the Turing machine, so you can take right hash moves there you print whatever that you are carrying p sigma and then you go L hash square and at that place you print that same symbol that same symbol back and then take a right move. Now, you see how complex is when you are receiving blank that means after finishing the input given to you then you just take R hash.

Then you can halt look here this is copying machine as desired because x the input whenever it is given say a 1 a 2 an what it does in the beginning it comes of course after finitely many steps you see that with the L has a 1 a 2 an. This is how the situation is and then it takes a right move it takes right move that means you will here and as this is not blank it prints blank here it prints blank here and then it takes two R hash machines R hash square.

So, that means it comes to this end and then one more R hash. So, next blank it will look for that means at first R hash it will come here and with another R hash, let me probably write here the what are machines that we are using instead of saying the their finitely many number of steps. So, here L hash is the applied and here R is applied and here R hash is applied then here also R hash is applied. So, that the situation is as follows a 1 a 2 an, so next blank on the right. So, that is immediately there.

(Refer Slide Time: 27:10)



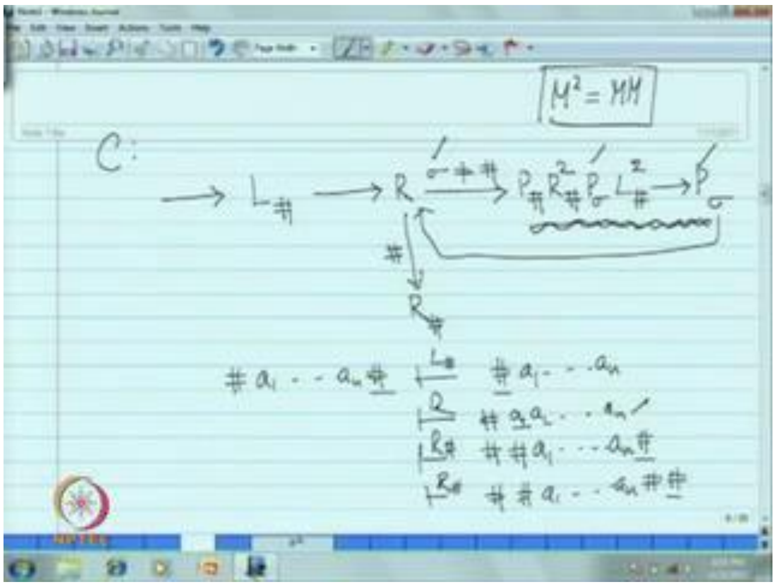
$$\vdash^{P_\sigma} \# \# a_2 \cdots a_n \# \underline{a_1}$$
$$\vdash^{L_\#} \# \# a_2 \cdots a_n \underline{\#} a_1$$
$$\vdash^{L_\#} \# \underline{\#} a_2 \cdots a_n \# a_1$$
$$\vdash^{P_\sigma} \# \underline{a_1} a_2 \cdots a_n \# a_1$$
$$\vdash^{R} \# a_1 \underline{a_2} \cdots a_n \# a_1$$
$$\vdash^{*} \# a_1 a_2 \underline{a_3} \quad a_n \# a_1 a_2$$
$$\vdash^{*} \# a_1 a_2 a_3 \cdots a_n \underline{\#} a_1 \cdots a_n \#$$

So, at this place it will print that sigma what are the sigma is carried a 1 is carried and therefore here what will happen. So, when you are applying that p sigma there, so you have things like this a 1 a 2 a n blank at this place you are there you print a 1. Now, you look at the machine it takes two L hash. So, first L hash comes here, so apply L hash and you will be at this place this is the situation apply one more L hash that machine. So, you will come to this blank this is a 1 is printed, now at this place you print that sigma back. So, whatever that still it is carrying that a 1 that is the information is maintained.

So, that is now a 1 here a 2 and so on a n a 1. Now, look at I connect I go to right and connect to the same loop from this p sigma. So, that means now I take one right move now it is the it is now with a 2. Now, a 2 will at this place you print blank you get print blank R hash square p sigma and L hash square again p sigma. So, that means what will happen after finitely many steps here you are of course here printing blank and you are going till first blank and the next blank is here at the end that is here. So, here it will print a 2 it will come back here to this blank position and it will type this a 2 back and it will move to the position. And now a3 will be copied end and so on till you receive this till you come to this particular blank .

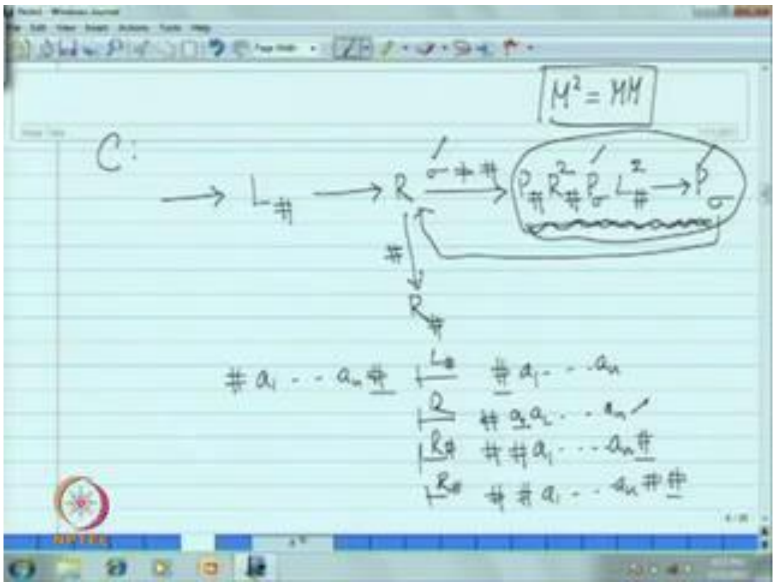
Then once you come to this blank it will simply take R hash and it will go to the right end. So, thus the output finally will be, so after print typing an it will now the here is blank. So, it will go to the right end that is R hash this is how the input is perused to get x x what are the x input given, so x will be given as output to this machine.

(Refer Slide Time: 29:50)


$M^2 = MM$
C:
$L_\#$
R
$\sigma \neq \#$
$P_\# R^2_\# P_\sigma L^2_\# \to P_\sigma$
$\#$
$R_\#$
$\# a_1 \cdots a_n \#$
$L_\#$ $\# a_1 \cdots a_n$
R $\# a_1 \cdots a_n$
$R_\#$ $\# \# a_1 \cdots a_n \#$
$R_\#$ $\# \# a_1 \cdots a_n \# \#$

Now, look at the construction the composite machine that here we have for copying. This particular black what are the block I am underlining here this particular block you see what are sigma that I am carrying here in this composite machine in this composite machine I have to have I have to use same information here. And I have to use that information again here, so that means here.

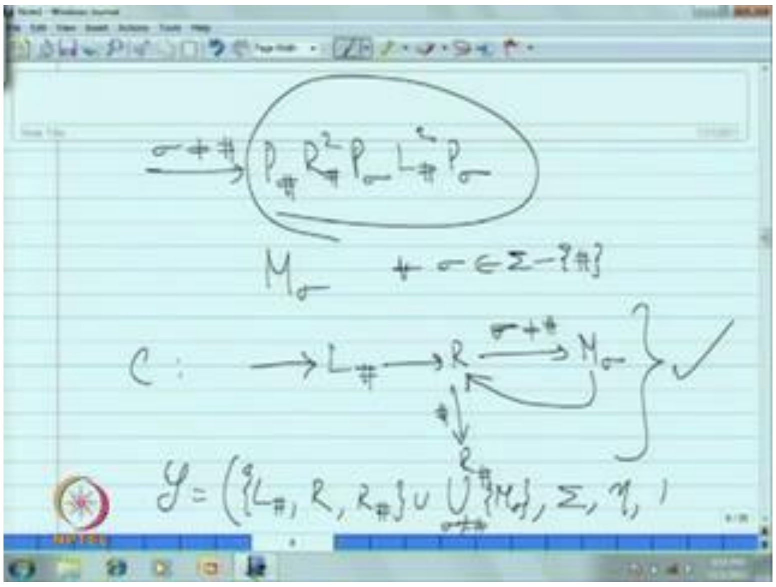
(Refer Slide Time: 30:21)



$M^2 = MM$
C:
$\rightarrow L_\# \rightarrow R \xrightarrow{\sigma \neq \#} P_\# R_\#^2 P_\sigma L_\#^2 \rightarrow P_\sigma$
$\# \downarrow R_\#$
$\# a_1 \cdots a_n \# \vdash^{L_\#} \# a_1 \cdots a_n$
$\vdash^{R} \# a_1 a_1 \cdots a_n$
$\vdash^{R_\#} \# \# a_1 \cdots a_n \#$
$\vdash^{R_\#} \# \# a_1 \cdots a_n \# \#$

One may understand that this printing blank taking R hash square again you have to print sigma then L hash square and then print sigma.

(Refer Slide Time: 30:33)

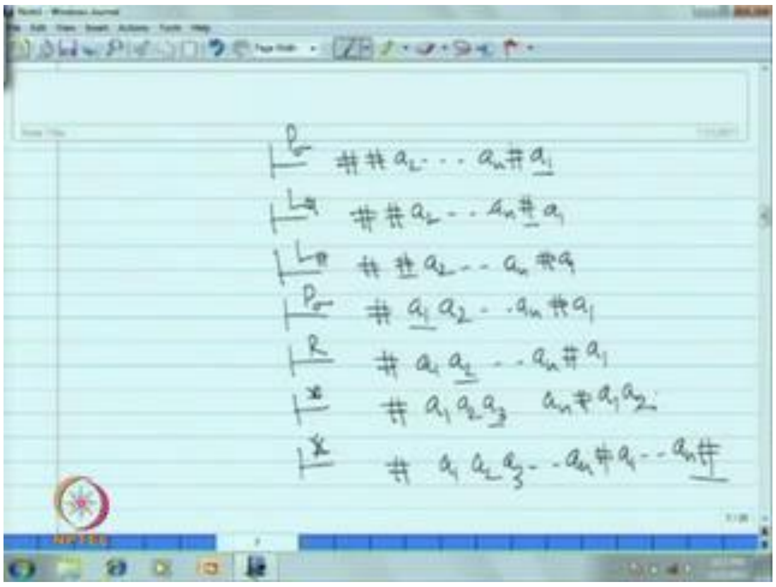


$\xrightarrow{\sigma \neq \#} P_\# R_\#^2 P_\sigma L_\#^2 P_\sigma$
$M_\sigma \quad \forall \sigma \in \Sigma - \{\#\}$
C: $\rightarrow L_\# \rightarrow R \xrightarrow{\sigma \neq \#} M_\sigma$
$\# \downarrow R_\#$
$\mathcal{G} = (\{L_\#, R, R_\#\} \cup \bigcup_{\sigma \neq \#} \{M_\sigma\}, \Sigma, \eta,$

So, the sigma whenever I am carrying whatever the non blank information that I am carrying in this particular machine I have to use it in this I have to use this sigma has to be used in this particular at this particular place and again coming back and printing. So, this entire component first you have to construct separately for this sigma non blank you have to declare that machine and this machine will to be constructed. So, once you construct this machine and if you call that as M sigma that M sigma can be used as a component of this construction.

And accordingly you can you can declare the machine schema of the Copying machine to thus what I can say the copying machine in which will be precisely for all sigma in sigma minus blank you construct that machine first.

(Refer Slide Time: 31:25)



$$\vdash^{P_\sigma} \# \# a_2 \dots a_n \# \underline{a_1}$$

$$\vdash^{L_\#} \# \# a_2 \dots a_n \underline{\#} a_1$$

$$\vdash^{L_\#} \# \underline{\#} a_2 \dots a_n \# a_1$$

$$\vdash^{P_\sigma} \# \underline{a_1} a_2 \dots a_n \# a_1$$

$$\vdash^{R} \# a_1 \underline{a_2} \dots a_n \# a_1$$

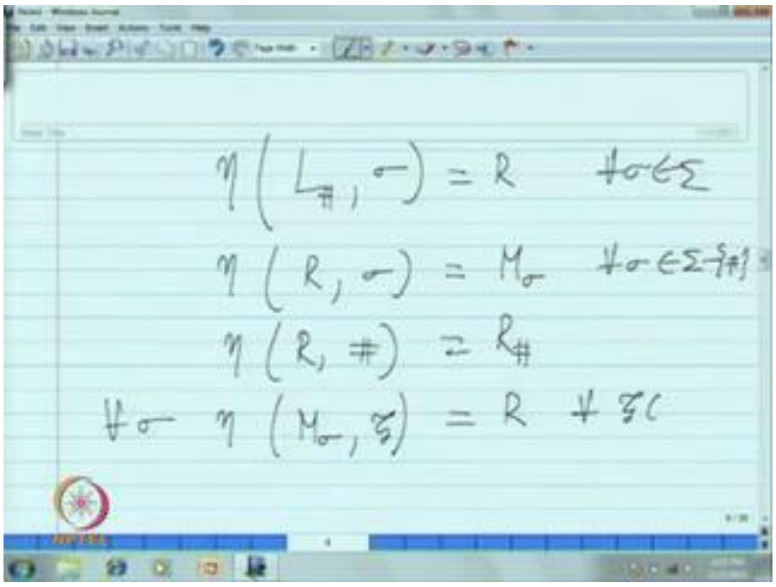
$$\vdash^{*} \# a_1 a_2 \underline{a_3} \quad a_n \# a_1 a_2$$

$$\vdash^{*} \# a_1 a_2 a_3 \dots a_n \# a_1 \dots a_n \underline{\#}$$

Then the copying machine when you are looking for the machine schema you have to construct it this way that is R sigma non blank M sigma and then. So, take a right move and sigma if it is not this then you connect to M sigma and M sigma will be connected back to R whenever you are receiving blank you are taking R hash. So, this is the simplest look how elegant is this machine copying machine, so with this you know notation the copying machine can be seen this is an very elegant manner. And now only thing is you have to understand that M sigma you have declare first and then you see there are one two three four components. And in this machine schema you can you can declare these are the machines and this L hash this one machine R hash union this M sigma.

For sigma not blank say for sigma not blank sigma not equal to blank let me write like that in a short way sigma not equal to blank for all those. So, these are the Turing machines and what are the common alphabet you have and eta essentially this definitions we have to have and the initial machine is L hash this is the machine schema.

(Refer Slide Time: 33:03)



$$\eta(L_{\#}, \sigma) = R \quad \forall \sigma \in \Sigma$$

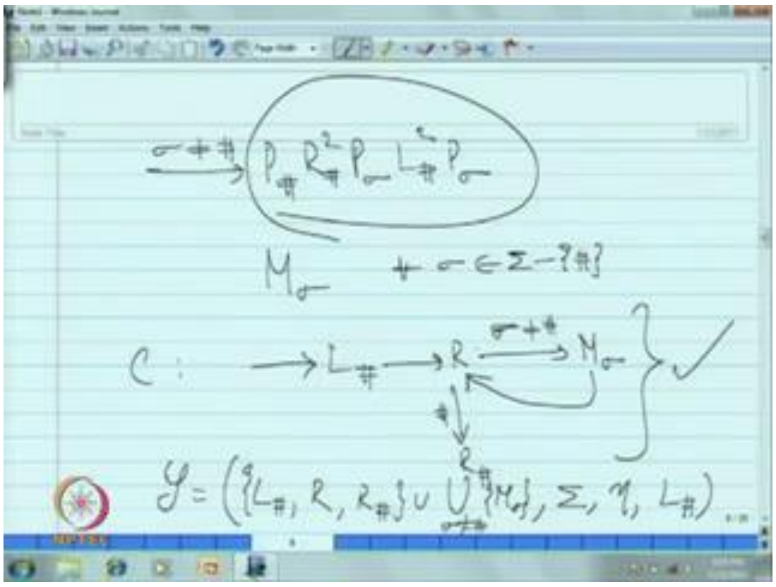
$$\eta(R, \sigma) = M_{\sigma} \quad \forall \sigma \in \Sigma - \{\#\}$$

$$\eta(R, \#) = R_{\#}$$

$$\forall \sigma \quad \eta(M_{\sigma}, \zeta) = R \quad \forall \zeta \in$$

And here the eta is defined L hash for all symbols sigma will be connected to R for all sigma and sigma and eta R. Now, you see there are two situations that for non blank you are connecting to M sigma for blank you are connecting to. So, R for sigma this will connected to M sigma for all sigma in sigma minus blank and the definition for blank is R hash this how we are connecting and now this M sigma will be connected to for all sigma where ever it is there M sigma for all symbols. Let me you say zeta is connected to is connected to this R for all this zeta and sigma.

(Refer Slide Time: 34:07)



$$\xrightarrow{\sigma \neq \#} \left(P_{\#} R_{\#}^{2} P_{\sigma} L_{\#}^{2} P_{\sigma} \right)$$

$$M_{\sigma} \quad \forall \sigma \in \Sigma - \{\#\}$$

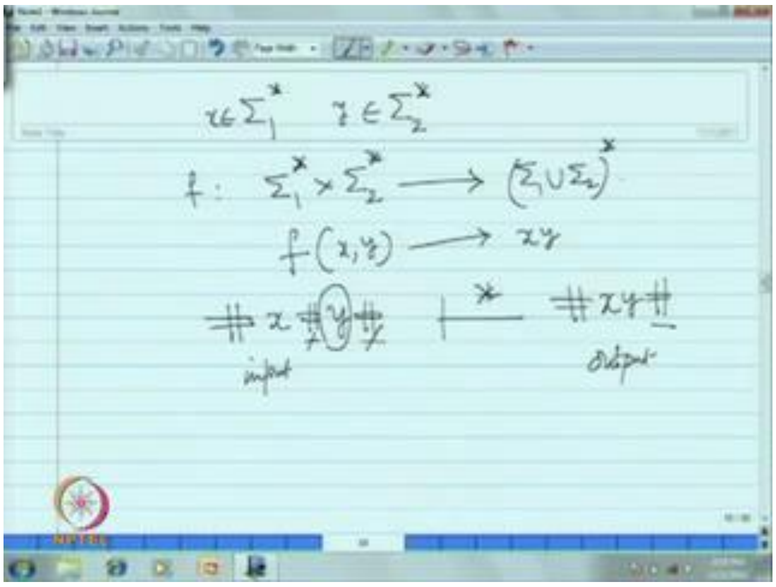
$$\begin{tikzcd} C : \arrow[r] & L_{\#} \arrow[r] & R \arrow[r, "\sigma \neq \#"] \arrow[d, "\#"] & M_{\sigma} \arrow[l, bend left] \\ & & R_{\#} & \end{tikzcd}$$

$$\mathcal{S} = \left(\{L_{\#}, R, R_{\#}\} \cup \bigcup_{\sigma \neq \#} \{M_{\sigma}\}, \Sigma, \eta, L_{\#} \right)$$

So, that is how the construction is now you look at how actually we have to represent the machine schema and see how to maintain those things and to represent that once you declare these things and for each component you have the states that you that are known to you. And when I am combining this four machines you take four auxiliary states and as per the definition of the Turing machine composite Turing machine which is represented by the machine schema as I have given the definition you can write the state transition table.

And you see this is actually matching with the original definition given to you. But, using this notation you can construct a very elegant complex Turing machine in a elegant manner. Now, in order to look at more and more complex works that we can pursue with Turing machines, let me first introduce one more notion that essentially computing with multiple inputs in which case for.

(Refer Slide Time: 35:19)



$x \in \Sigma_1^*$ $\quad y \in \Sigma_2^*$

$$f : \Sigma_1^* \times \Sigma_2^* \longrightarrow (\Sigma_1 \cup \Sigma_2)^*$$

$$f(x,y) \longrightarrow xy$$

$$\# x \# y \# \quad \vdash^* \quad \# xy \#$$

input $\qquad$ output

Let me first give an example where you have one string that you are taking from a sigma 1 star and I am taking another string y from say sigma 2 star and the function what I want to pursue is I take two strings. So, then the function will be like this and now let call sigma 1 union sigma 2 star, here how it is define you take two strings I wanted to simply I wanted to simply concatenate them I want to simply concatenate them. So, given two stings concatenating this is a very basic operation that we can easily pursue using your usual computer. Now, you see in order to capture that notion first we their question is essentially whether it is Turing machine computable function, now in which case the function will be it is not just taking one input it will take two inputs.

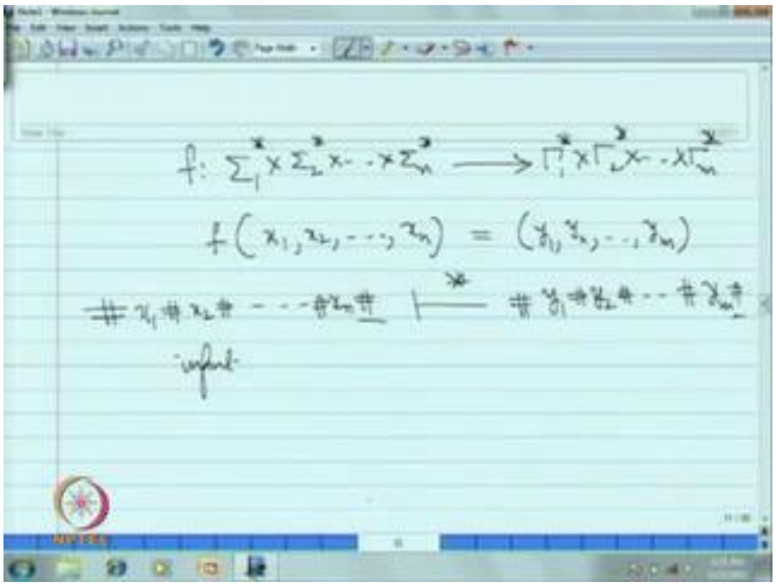
So, I am representing this a function from two sets that is sigma 1 and cross sigma 1 star cross sigma 2 star to sigma 1 union sigma two star because what are the symbols of sigma 1 sigma 2 may occur here as you see. So, this is the function we see this is Turing computable function in which case my notation you will be giving input in this format the.. This is the initial this thing after finitely many steps I should be left with this kind of situation that is the expectation using this format this input format and this is the output format.

So, when two strings are given this is how I will take when three strings are given then I will separate because this is the blank symbol is a special symbol that special symbol I will put in between. And you see that this the what whenever we are taking the input we are not allowing blanks as you have noted earlier, now this blank can be used as special symbols if you already using somehow or other in the input.

The blank also in which situation you can introduce some other special symbol because essentially we should distinguish when I am saying I am giving two strings as input. We should you should able to distinguish like what is the first string what is the second string and the input tape. So, that way I am I do not require because I have reserved this blank symbol as a special symbol I will just use it. So, this blank in the beginning and the second blank because till between these two blanks. This block whatever is that is the first input I can say and between the second blocks of blanks I was I will say this is a second input that is how if you want three strings as input.

Then I will separate with one more blank and I will give say for example x y z i can give. So, this is the this is how we give the input and we expect the output as here I mention one string if again if you want to have two strings as output you can maintain such a format and so on by separating blanks. So, let me fix the convention that the input if n number of inputs are given then they will be separated by the blanks and if m number of outputs are expected they will be separated by blank.

(Refer Slide Time: 38:43)



$$f : \Sigma_1^* \times \Sigma_2^* \times \cdots \times \Sigma_n^* \longrightarrow \Gamma_1^* \times \Gamma_2^* \times \cdots \times \Gamma_m^*$$

$$f(x_1, x_2, \ldots, x_n) = (y_1, y_2, \ldots, y_m)$$

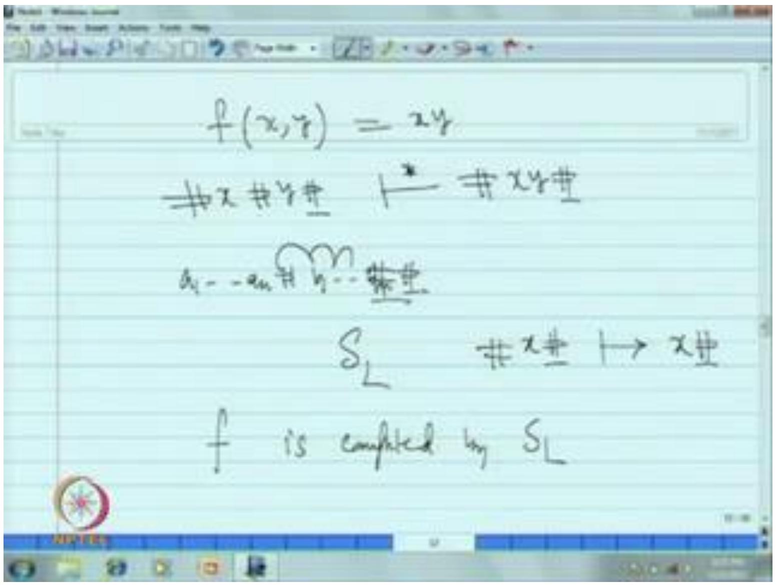
$$\#x_1\#x_2\# \cdots \#x_n\# \overset{*}{\vdash} \#y_1\#y_2\# \cdots \#y_m\#$$

input

So, in general if a function is considered that say for example sigma 1 star cross sigma 2 star cross and so on cross sigma n star. Consider a function take into n inputs and for example gamma 1 star cross gamma 2 star and so on cross gamma m star. Some alphabet sigma a s and gamma a s are alphabets m number of n number of inputs are taken say x 1 x 2 and x n the input taken which is assigning to say for example y 1 y 2 y m.

This is the if this is the thing then the input will be of this x 1 blank x 2 blank and so on blank x n. So, at this place you are reading and writing head initially the input and after finitely many steps the machine should hard with the output y 1 y 2 and so on y m. This is the expectation, now as per this format we say a function f is Turing compatible if there is a Turing machine computes this function, now I will say a function. So, this function f is computable if there is a Turing machine computing this function.

(Refer Slide Time: 40:15)



$f(x, y) = xy$

$\# x \# y \underline{\#} \vdash^* \# xy \underline{\#}$

$a_1 \cdots a_n \# b_1 \cdots \underline{\#}$

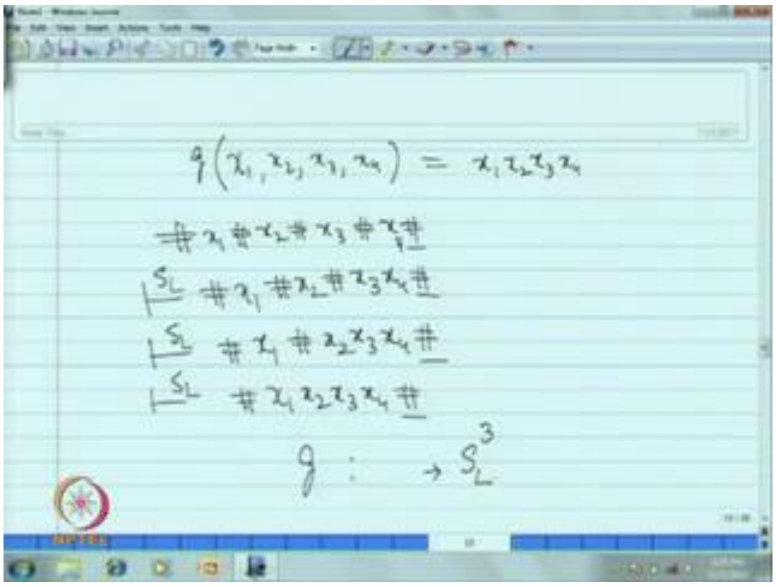
$S_L \quad \# x \underline{\#} \mapsto x \underline{\#}$

f is computed by S_L

Now, the function what I have mentioned in the just earlier with two inputs if you want to concatenate them. Now, you see the input x y this is how it is given the expectation is after finitely many step the machine should halt by concatenating these two in this way can you look at that. How can I pursue this essentially from the original position of the input we just go to the this first blank to its left and keep copying the symbols to left. So, that y can be shifted one cell to its left and the n at the end is the cell that you can make it blank. So, when I say a 1 a 2 an is x and b1 b2 b m is y from here you go here move b 1 then move b 2 move b 3 and so on this b m on.

Say it is moved then you can make it blank here and you stop here. So, this process I need not now pursue it separately what are the machine as cell the left shift machine what it does you give any strings with one blank in the beginning. So, it simply shifts this x 1 cell to its left this is the operation of S L, now this function f is computed by S L precisely because S L if you just connect then automatically from here it will shift y and then the resultant string is x y. Because your halt at end of y and you see thus f is a Turing computable function computed by S L S L the left shift machine looks very simple, but you see such a thing that it will now if you want to come concatenate. Say for example four strings are given then you can use this S L for that many times and see that it is.

(Refer Slide Time: 42:29)



$$g(x_1, x_2, x_3, x_4) = x_1x_2x_3x_4$$

$$\#x_1\#x_2\#x_3\#x_4\underline{\#}$$

$$\vdash^{S_L} \#x_1\#x_2\#x_3x_4\underline{\#}$$

$$\vdash^{S_L} \#x_1\#x_2x_3x_4\underline{\#}$$

$$\vdash^{S_L} \#x_1x_2x_3x_4\underline{\#}$$

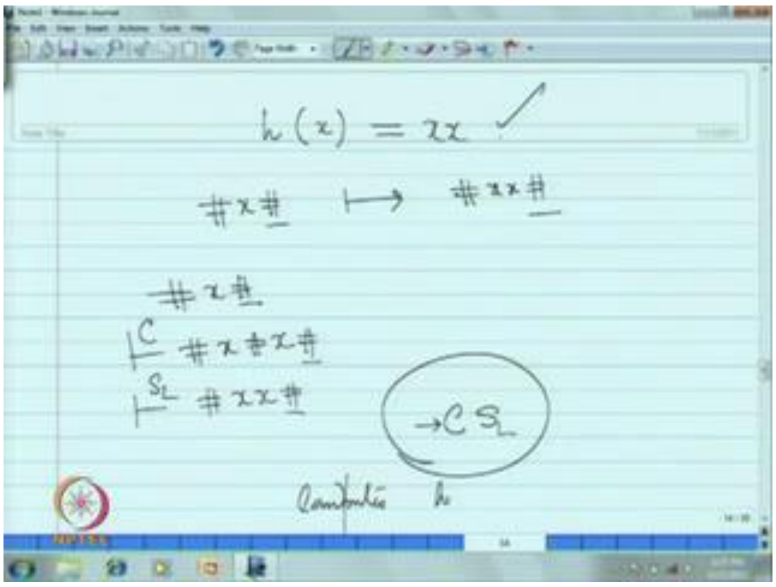
$$g : \rightarrow S_L^3$$

Now, let me look at that example the example x 1 x 2 x 3 x 4 say four components are there, now g is a function this is appropriately from four that sigma e star. You can write now concatenating these if you want to pursue what is the Turing machine. Now, the input is like this x 1 x 2 x 3 x 4 this is the input now I apply left shift machine once. Then what will happen that x 1 x 2 this x 3 x 4 will be combined this place once I apply left shift machine this x 4 will be shifted one cell to its left and this is the resultant string. Now, this is one single string x 3 x 4 is a single string say some y, now once again you apply left shift machine then what will happen that is x 1 this x 2 x 3 x 4 this is the situation.

Now, because these two strings will be concatenated and now once again if I apply left shift machine the resultant string is x 1 x 2 x 3 x 4 you see how the left shift machine if you use it for three times. So, that means to pursue this g this can be computed by S L cube that is all this is the machine. So, S L you compose with S L and compose it with S L remember these three S L s you have to maintain three different copies if compose S L with same copy again it will now pursue in different manner.

That is it will keep trying to shift infinitely many times and you never know what will happen at the situation infinitely many times it has to pursue. Then it will try keep shifting that it will it will hang. So, when I am writing S L cube here precisely three copies of S Ls you take and concatenate them I mean compose them one to another all the three copies. So, this S L is helping you to concatenate this strings, now if because you know one of the previous exercises I gave you one function

(Refer Slide Time: 44:54)

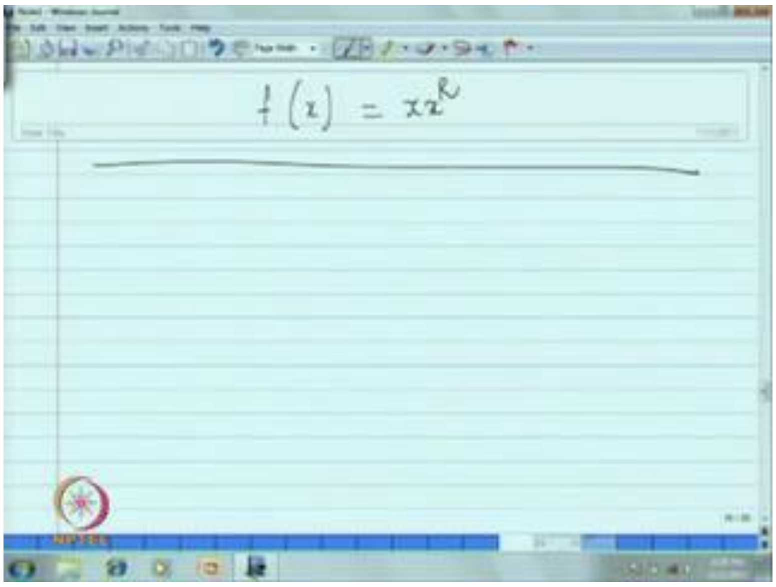


$h(x) = xx$ ✓
$\#x\underline{\#} \longmapsto \#xx\underline{\#}$
$\#x\underline{\#}$
$\vdash^{C} \#x\#x\underline{\#}$
$\vdash^{S_L} \#xx\underline{\#}$
$\rightarrow C\, S_L$
Concatenation h

Let me write one example here that is say x the input string I ask you to give xx for this case what you do. So, that means the input will be given this format and the output is expected this way this is what we need now you can quickly understand that what are the. So, far machines we have construct what are machines, so far we have constructed first I will start with this input like this and first applying copying machine just we have declared this copying machine will prepare a copy of x.

And halt at this position if you if you compose this with S L, now the resultant string is this will move one cell to its left. So, it will become xx, so that means to pursue this to pursue this what we have to do you just concate compose C with S L and of course the starting machine here is C. So, this Turing machine computes this Turing machine computes h the function h as h of x is xx that you can pursue using this two dimension.

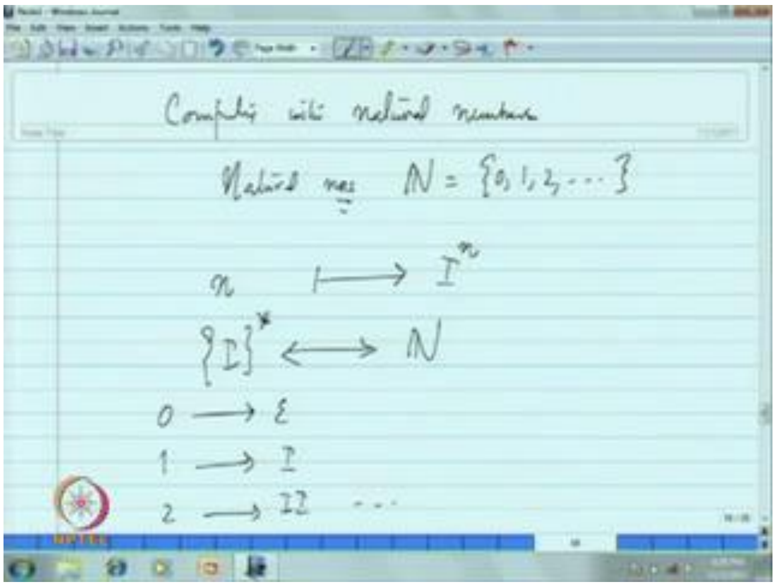
(Refer Slide Time: 46:24)



$f(x) = xx^R$

Now, likewise as we have ask earlier I think this function also asked to xx power R. Now, you can try that whatever the x given to you have to first prepare the reversal and once you left shift machine on and that those two strings will be concatenated. Now, you have to construct a machine first that it has to give x power on R next to the given input without erasing that and then you can concatenate them like wise other problems that we can look into…

(Refer Slide Time: 47:03)


Computing with natural numbers

Natural nos $\mathbb{N} = \{0, 1, 2, \dots\}$

$n \longmapsto I^n$

$\{I\}^* \longleftrightarrow \mathbb{N}$

$0 \longrightarrow \varepsilon$

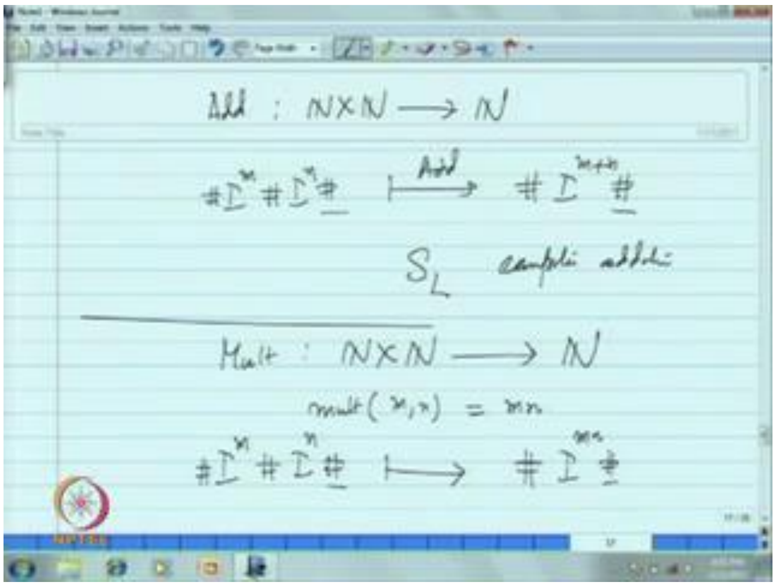
$1 \longrightarrow I$

$2 \longrightarrow II \quad \dots$

And now let me introduce the notion of computing with natural numbers natural numbers. Let me write this is set an I include 0 1 2 and so on, now how how do because we have discussed computing with strings. Now, when I am talking about computing with numbers, now I have to represent this numbers in a as a string because Turing machine. We have understood that we are computing over strings, now the representation here is any natural number you give me that natural number we represent in the numeric format were with one symbol as I power n.

So that means this natural numbers will now consider the alphabet singleton I and we take this correspondence with natural numbers. So, zero with epsilon if it is 1 we are using 1 I if it is 2 will print 2 I s. So, this is the unary representation a 3I s and so on. So, that is how we represent and when I want to pursue certain basic you know computation with natural numbers. That we can demonstrate and see that how to how to compose basic Turing machines in order to compute the functions on natural numbers.

(Refer Slide Time: 48:26)



$$Add : N \times N \longrightarrow N$$

$$\# I^m \# I^n \underline{\#} \xmapsto{Add} \# I^{m+n} \underline{\#}$$

S_L compute addition

$$Mult : N \times N \longrightarrow N$$

$$mult(m,n) = mn$$

$$\# I^m \# I^n \underline{\#} \longmapsto \# I^{mn} \underline{\#}$$

Let me start with very basic function then after wards say for example addition this is a function from N cross to N to you know that given two natural numbers add to numbers, so the input I will take say I power m blank I power n. So, what should be the result addition function this addition what I would require is I power m plus n as output. Now, you can quickly answer that what is the Turing machine that performs addition because only thing is input is different here natural numbers I say you can quickly see that if just shift the second block then it will quickly give you the addition. So, that means S L is sometimes that we have observed that to concatenate two strings I am using S L.

Now, you see this S L the same Turing machine if the input is natural numbers in this format it performs addition. So, this will be this computes the addition function and natural numbers S L computes addition on natural numbers. Now, one can try multiplication of course this is again two natural numbers you give as input and this multiplication is m n is m multiplied by n that means the input if you are taking like this your you have to give output this way. So, you can think of this and not only this multiplication may be you know the quotient on the integral part of two numbers are may be subtracting.

But, of course when I am saying that function from natural numbers to natural numbers we can perform m on us that means when from a bigger number when we are subtracting you will subtract if you are if you want to subtract from a smaller number then what you have to you will result 0. So, this kind of m on us function are many other arithmetic operations and natural numbers may be the computable functions that you can represent through Turing machines there the input will be taken this way.

So, if it is having one number as input and asking you to per pursue something then you will you will you will take just one block of I s if two natural numbers are inputs. Then as for addition and multiplication we are taking it will be like this and we are taking three numbers as input then again as per the formats as three strings you have to take the input and so on. Now, you can try for quotient and other simple basic arithmetic operations that you can you normally do with your computer you can try constructing a Turing machine and some more examples in this direction we will discuss in the next lecture.