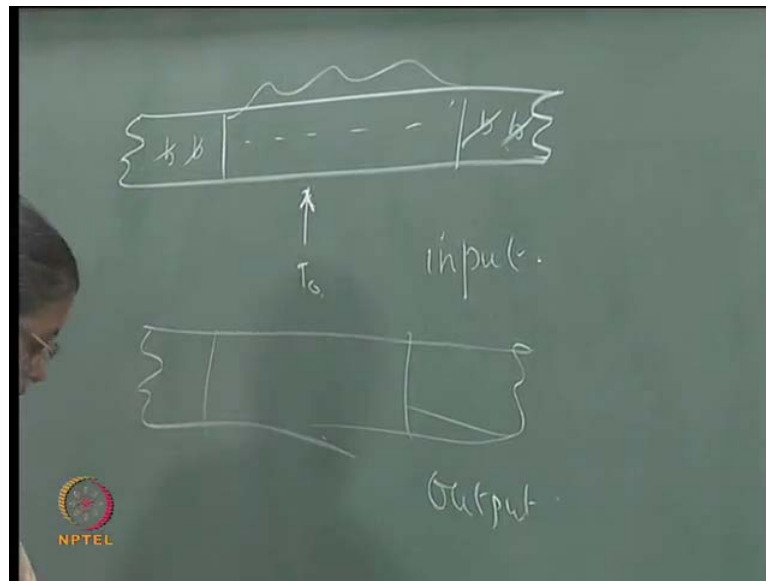


Theory of Computation
Prof. Kamala Krithivasan
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Lecture No. # 27
Turing Machines

(Refer Slide Time: 00:16)



So, today we shall consider some more examples of a Turing machine. A Turing machine has an infinity tape and it has some nonblank portion, rest of the tape is blank. I use this symbol for blank, the last lecture we considered them as an input output device today also first problem, we will consider it as an input output device. So, when you start the machine in an initial state, say q_{naught} , you have to specify the head position. This is called the head of the Turing machine, head of the tape, there is some nonblank portion that is the input after making some moves, the machine halts. And, when it halts whatever is the nonblank portion that is the output. We consider two examples in the last lecture, one was to check whether a binary string has odd parity or even parity number of one's is even or odd.

Another one is whether given string of parenthesis is well formed or not. Today we shall consider an example where a unary number will be converted into a binary number.

(Refer Slide Time: 01:45)



So, the input is given like this, this is the tape.

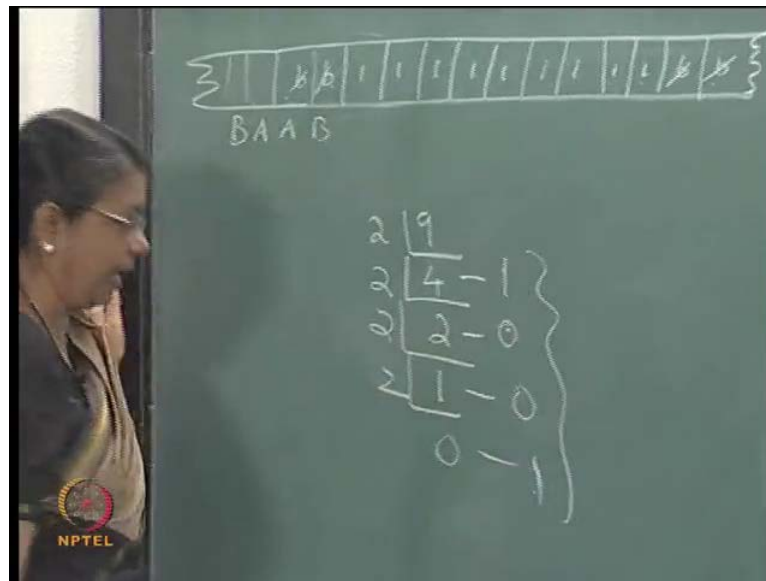
(No audio from 1:45 to 2:01)

Suppose I have 1 1 1, I have nine 1's, is a number nine in unary, it is (()) with some a or you can just have blank also, does not, blank itself is ok. So, the input is the number nine in this example I am , any n number n will be given as a unary number 3 4 5 6 7 8 9, 9 what is the binary equivalent to 9?

(())

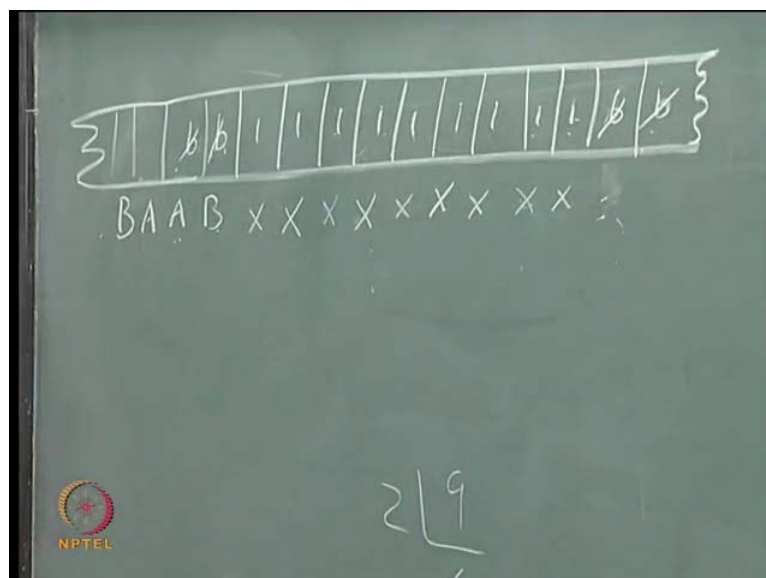
1 0 0 1, I want to get this, but I should not mix up with this one, and this one. So, rather I will print it as B A A B, where B has the representation 1, A represents 0. So, at the end I should have B A A B printed here, in this portion. This 1's, you can change them to x's at that time. So, when the machine halts, whatever you have here, you must to interpret the B as A 1 and the as A 0, and this should give the binary representation of this unary number. How will you design the machine? How will you design the Turing machine?

(Refer Slide Time: 04:11)



Now, how do you convert a binary number to a unary number? Suppose, I have 9 you divide by 2, quotient is 4 remainder is 1, again divide by 2, quotient 2 remainder 0, then divide by 2, quotient is 1 remainder is 0, divide by 2 1 0 1 1, this is, you know. So, the representation is 1 0 0 1, this you should get. So, what you do is? Informal description we can very easily give, but it is also necessary for us to give the formal definition that is the delta mappings informally.

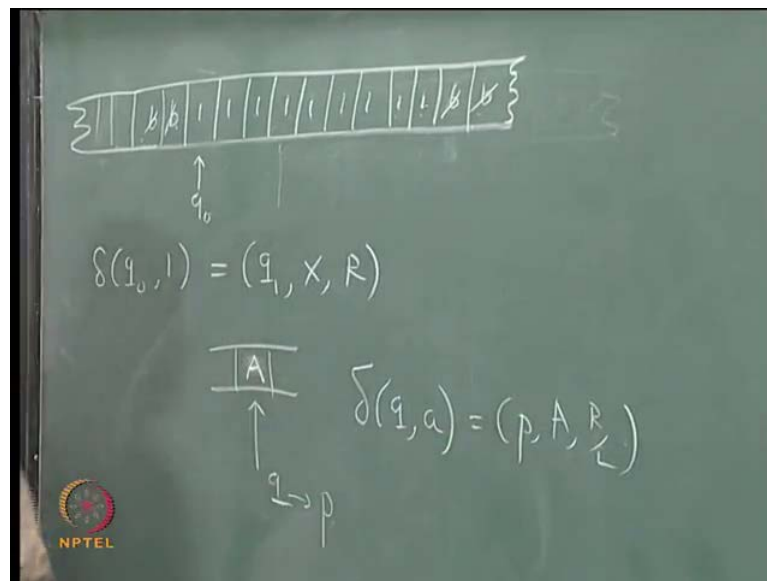
(Refer Slide Time: 04:56)



You can very easily describe this, change the alternate 1's into x's and while doing switch between q_{naught} and q_1 two states, q_{naught} q_1 q_{naught} q_1 , like that you switch. So, if you are in q_{naught} ; that means, you are, you have seen an odd number of, even number of 1's. If you are in q_1 ; that means, you have seen in odd number of 1. So, when you reach here, in q_{naught} ; that means, the remainder is 0, if you reach here in q_1 ; that means, the remainder is 1 and divided by 2. Go back and depending upon what is the remainder, print B or A. Then now I have 9 divided by 2 I have 4. Now, these have been changed to x's I have 1 2 3 4, now the quotient is there, as 1's, change alternate 1's change this into x, leave this, change this into x, leave this out.

Then, when you reach here you will realize by going in stage q_1 . Not that you have seen in even number of 1's the remainder is 0. So, go back and print a 0. Again do the same things, change this into x, leave this out, go print A, then again do the same thing, leave this and change this, and go and print B. Now, when you go you do not see a 1 at all, you halt. So, at this state the output is given here.

(Refer Slide Time: 07:10)

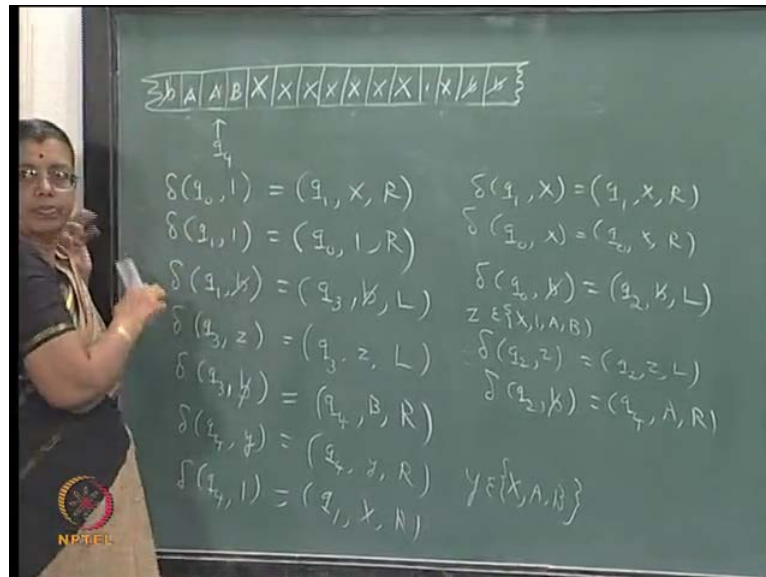


So, the input is a string of 1's is it not? Input is a string of 1's. It represents the unary number. So, you start in state q_{naught} , delta of the mappings will be written like this. Wait I will do, I will write here delta of q_{naught} 1 is q_1 x R, the mapping depends upon the state and the symbol read. Depending upon the state and the symbol read, see when you have a symbol a and you are reading this in state q. The mapping will be given by

delta of q a is equal to some P A R or L, that is the state will change to p from q. It will change to p over this a, this symbol will be printed and the move will be tape head will move here, if it is R, it will move here if it is l. In fact, in some books, you also use stationary, remaining stationary also is used, but that is not many books you will say only L and R.

The reason is remaining stationary, we can achieve by two moves moving left and right. So, usually only L and R are taken into account. So, in q naught you see a 1, change this into an x and move right in state q 1, now alternate once you have to leave out, that is you should not change it into a x.

(Refer Slide Time: 09:04)



So, delta of q 1 1 is, but the states should change, see Britain's, alternating between q naught and q 1. You are remembering whether you have come across odd number of 1's are even number of 1's. So, state goes to q naught, but this 1 you do not do anything with that and you are moving. So, next instance it will be q naught and 1, what is a move for q naught and 1? q naught 1 is q 1 x r. So, change this into x and move in q 1, in q 1 when you see a 1, you leave with the 1 out, but change the state to q naught. So, next instance, you will go here in state q naught and in q naught, if you see a 1, you go to q 1 print x and move right. So, print x and move right in q 1, in q 1 if you see a 1 go to q naught, but do not do anything with the 1. So, you will move here in q naught, q naught 1 is changes into x and move her in q 1 q 1 1 is just move right in q naught, q naught 1 is q 1 x R. So,

this will be change in to x and you are here in state q 1. So, the first blank symbol, if you see in q naught, it means you have come across even number of 1's. If you see the first blank symbol in case state q 1; that means, you have come across odd number of 1's. So, when divided by to the last digit will be 1, mean the remainder 1. So, that is in the binary representation the last digit will be a 1. We have the assume that B stands for 1 and a stands for 0, now it has move left, come across all this x's etcetera until it reaches the B. So, delta of q 1 blank is q 3, then blank it does not do anything with a blank symbol, just blank starts moving left. So, in set q 3 its starts moving left until it sees a blank symbol. So, delta of q 3 here, it can be anything except blank. So, I will put it as some z equal to q 3 z L. Now, z can be it can be x, it can be 1, it can be a, it can be anything, it can be except blank. Because after sometime, you will also have a b's, here it will move through all this and in q 3, it when it sees a blank, it goes to a state q 4. And, what will it print, what will it print B or A? q 1 means, it has come across odd number of 1's; that means, which, what should print B? (()) louder please.

B.

B and in q 4 its starts moving. So, in q 3 it has moved completely over this, and here it prints B and moves. In q 4, in q 4 see the x's, it has to cross some excess in q 4, it will move. And, when this is the first 1, it will change, it starts changes the second, the second step will, and second pass will be started. So, delta of q 4 x are, not even x, it again, I will put it as some y will be q 4 y R. But when it is, what is y? y could be x or A or B, now delta of q 4 1, it has to start the process all over again. So, when it comes here in state q 4, when it sees a 1, it will change this into a x and go to q, q 1 x R. Now, in q 1 it has to pass through the x's, delta of q 1 x is equal to q 1 x r, but when it sees a 1, it goes to q naught.

So, it goes to q naught, in q naught, if it sees x also, q naught x is q naught x R, we will just move. So, here when it comes, it will change this into x, move here in q 1, then move here in q 1, then move here in q naught. Is this clear? It changes alternate 1's into x's and also switches the state from q naught to q 1 and so on. When it passes over the x's, it is just moves. Whatever state it is? Eight q 1 or q naught it just moves over. So, ultimately when it reaches the, here it is q naught. So, it will reach the blank symbol in state q naught; that means, it has come across even number of 1's. So, in q naught, it sees

a blank. So, delta of q naught blank, now what state it has to go? Not q_3 q_3 , if it moves left, it prints B, you should have a state for printing a. So, it goes to say q_2 blank L.

So, here it moves here in state q_2 and q_2 , it just whatever symbol is there it moves left until it reaches the blank. So, similar to this move $q_2 z$, z can be anything, I will write z here, z belongs to x . What is that x_1 A B? Now, instead of q_3 , if you have q_2 $q_2 z$ is $q_2 z$ l. So, you have reach the blank symbol in q_2 , now, what should do you print A or B?

A

A So, delta of q_2 blank is q_4 A. So, you print A and move in q_4 , in q_4 you move until you see a 1, in q_4 , you see a 1. When you see a 1, you change that into x and move to q_1 . So, you change this into a x and move right in q_1 you go up to this just move and when in q_1 you see A 1, you go to q naught, do not do anything with A 1 move. So, you go here and q naught, then you move in q naught, now in q naught you see a blank symbol. What does that mean? You have come across even number of 1's. So, the remainder is 0. So, you go to q_2 naught blank is q_2 blank L. So, you start moving left in q_2 , you move left, until to this blank, and in q_2 blank, you have to print a and move.

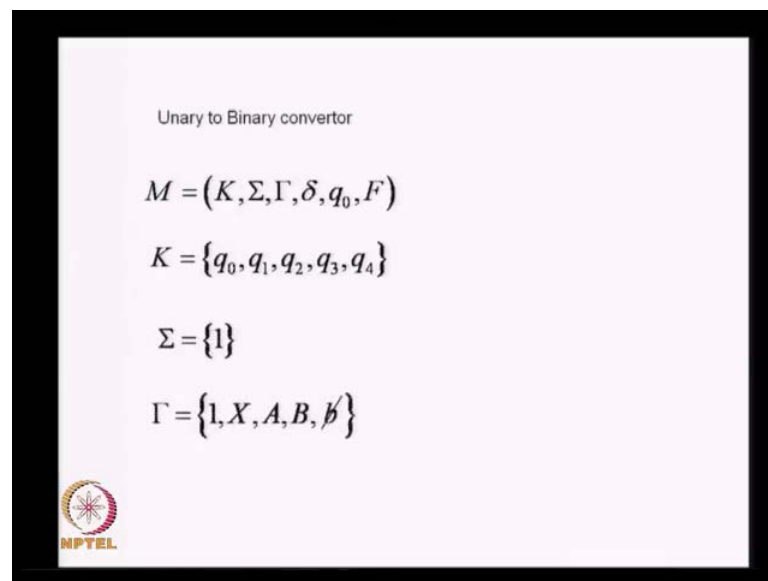
So, print A here and start moving right in q_4 , in q_4 you start moving until you see a 1. When you see a 1 you go to q_1 make it x and move. So, make this into a x and move in straight q_0 , q_1 q_1 . So, in q_1 you see the blank symbol, you are remove in state q_1 and q_1 blank is you have to move left in state q_3 . Say in state q_3 , you move left until you reach this blank, in that case you print B and start moving in q_4 . In q_4 , when you see the first 1, you changes into a x and move to q_1 . Now there are no more 1's left. So, you will be moving in q_4 throughout and reach the blank symbol in q_4 . So, when you have q_4 and a blank halt, blank again you do not do anything in the blank. You go the halting state and halt, usually after the work is done; you go to a halting state.

And that is denoted as h, you do not $(())$ usually after that is over the machine halts, after the computation is over the machine halts. So, now, all the 1's have been changed into x and you have the binary representation of the number, this is a unary to binary. So, by practice, you should be able to write these moves. So, will take, I will give some assignment of with number problems. So, now if you suppose I have 6, how will it work? It will first, we have 6 then it will change the first 1 in to x , leave this out changes this, leave this out. When it goes to the blank symbol, it realizes, it has come across even

number of 1's. So, comes back in state q_2 and prints A, then next step it will change this into x leave this out, changes into x.


That means, it has come across odd number of 1's, in q_1 , it will reach then q_1 come back in q_3 and print B. Again go here, now change this into x, it will reach here in state q_1 , come back in q_3 and **(C)** B.

(Refer Slide Time: 23:59)



Unary to Binary convertor


$$M = (K, \Sigma, \Gamma, \delta, q_0, F)$$
$$K = \{q_0, q_1, q_2, q_3, q_4\}$$
$$\Sigma = \{1\}$$
$$\Gamma = \{1, X, A, B, \emptyset\}$$



This is the way it works, thus we have seen that we can have a Turing machine which converts a unary number to a binary number. And, it has got 5 states, q_0 q_1 q_4 are right moving and q_2 and q_3 are left moving states.

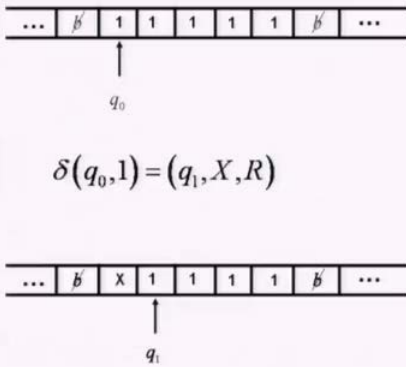

(Refer Slide Time: 24:16)

mappings

$$\delta(q_0, 1) = (q_1, X, R)$$
$$\delta(q_1, 1) = (q_0, 1, R)$$
$$\delta(q_0, b) = (q_2, b, L)$$
$$\delta(q_1, b) = (q_3, b, L)$$


These are the mappings, this we have seen just now. Let us see how the machine works on another tape having five 1's. This number 5 you has to be written in binary in this place.

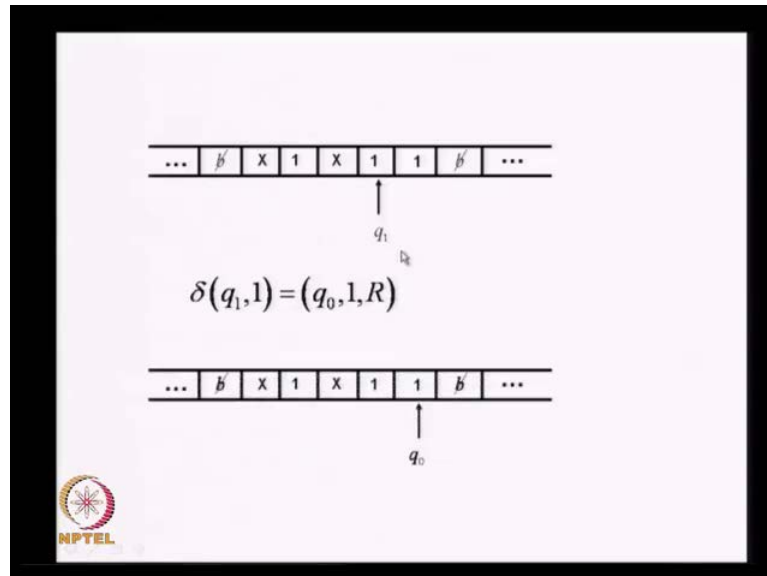
(Refer Slide Time: 24:20)


$$\delta(q_0, 1) = (q_1, X, R)$$


So, initially the machine is reading this 1, in state q_0 and uses this mapping $q_0 1$ is equal to $q_1 X R$ and goes to the right printing a X on the cell it read. Now, from this situation, it leads out the next 1 and goes to the next cell just reads it and moves

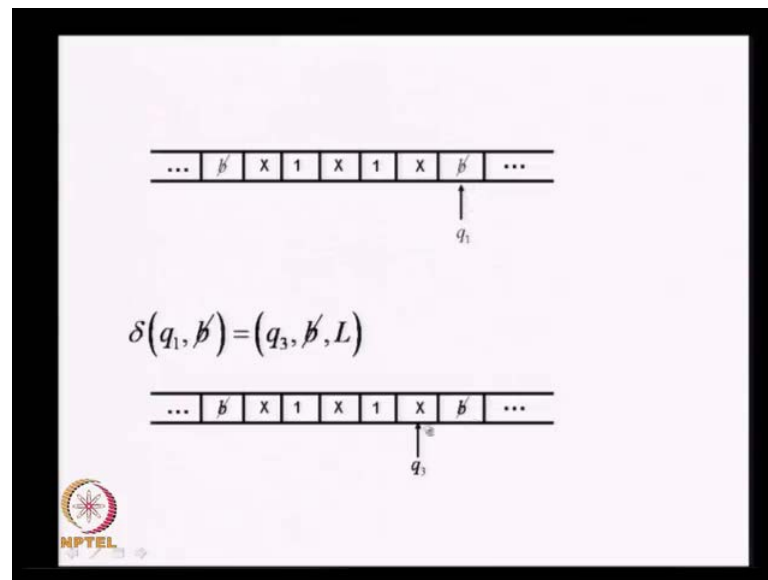
right using this mapping. Now, when q naught again, if we sees a 1 it changes that into a x using this mapping and moves right.

(Refer Slide Time: 25:19)



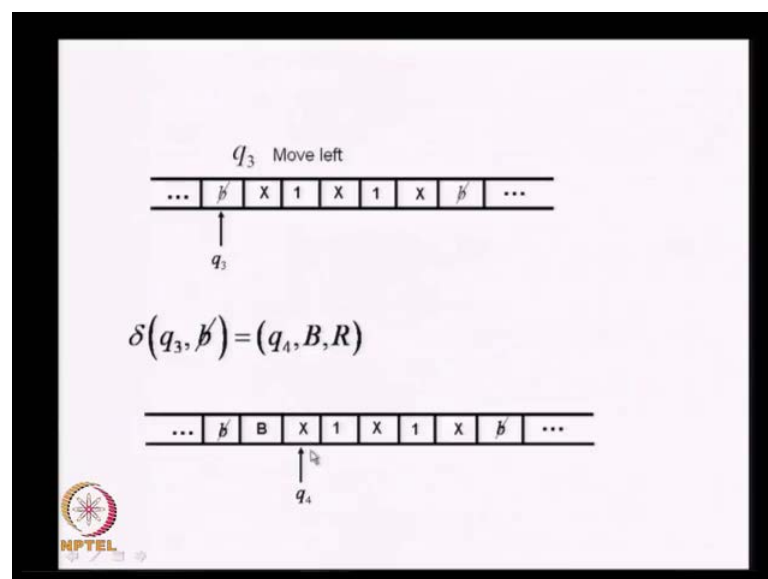
So, it changes alternate 1's into x's and leaving the other once as they are, now in q_1 it sees a 1, it moves right in state q_1 naught. But does not change the 1, this is the last symbol it reaches the 1; it reads it in q_1 naught. And, using the mapping q_1 naught 1 delta of q_1 naught 1 is equal to q_1 x R, it changes that into x, and moves right. Now, it sees the blank in state q_1 ; that means, it has encountered an odd number of 1's. So, it moves back and then it has to print B here, that is B denotes capital B denotes 1 and a denotes 0 here. So, it has to move left and print a B in this position.

(Refer Slide Time: 26:07)



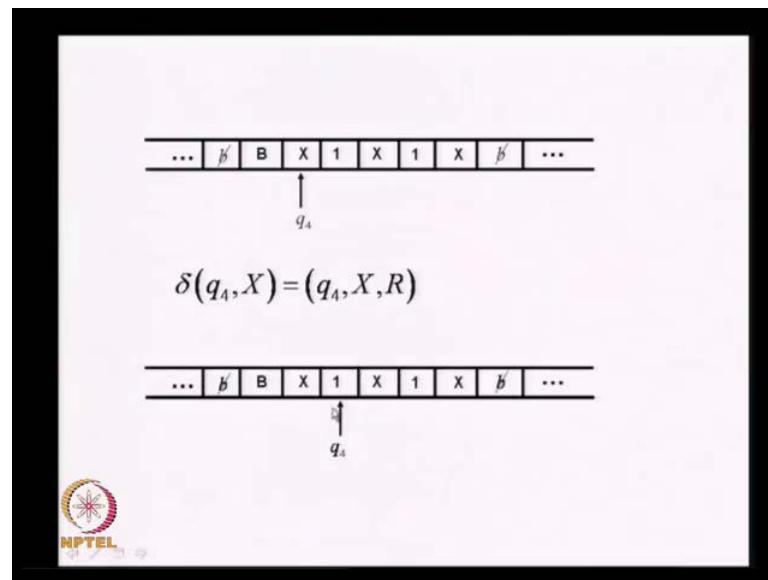
So, in q_1 B, it moves left and it uses this mapping q_1 blank is equal to q_3 blank L and moves left in straight q_3 , until it reaches here. And in this position, it reads the blank symbol in q_3 and (()).

(Refer Slide Time: 26:26)



Uses this mapping, delta of q_3 blank is equal to q_4 blank R prints the capital B here and moves right in straight q_4 . q_4 , it moves right, it is a right moving state, it moves right in state q_4 till it sees a 1. So, this x does not change and it just moves right.

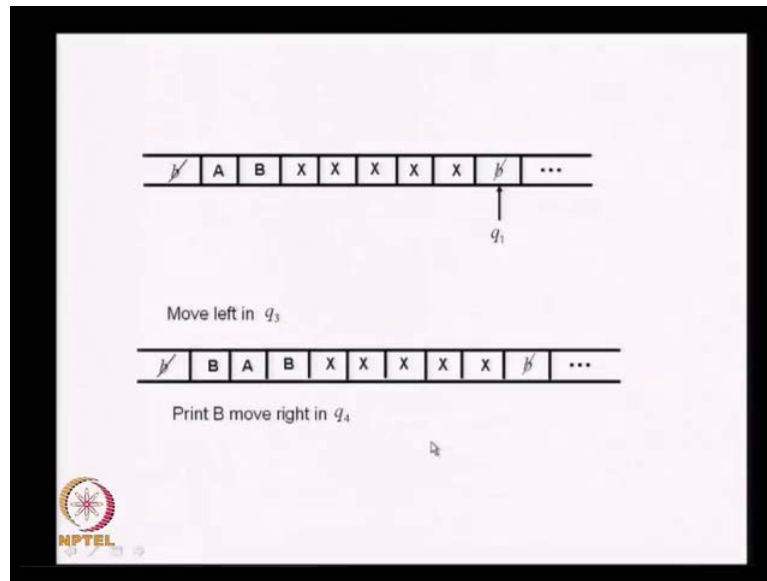
(Refer Slide Time: 27:00)



So, in q_4 it reads a x , it just moves right, going to the next symbol 1 . Now in q_4 it sees a 1 this is a situation and it uses this mapping $q_4 1$ is equal to $q_1 x R$ and moves right. Here again there are two 1 's, it changes alternate one's into x 's. So, this 1 it changes into x , but this 1 leaves out moves right, uses this mapping moves right but does not change this 1 . And again moves right, and in this position it reaches x in state q_{naught} and δ of $q_{naught} x$ is equal to $q_{naught} x R$, uses this move and goes to the blank. Now, in this position it reads the blank in state q_{naught} ; that means, it has encountered an even number of 1 's. So, it moves left in state q_2 and when it reaches the blank cell here, it prints a denoting zero.

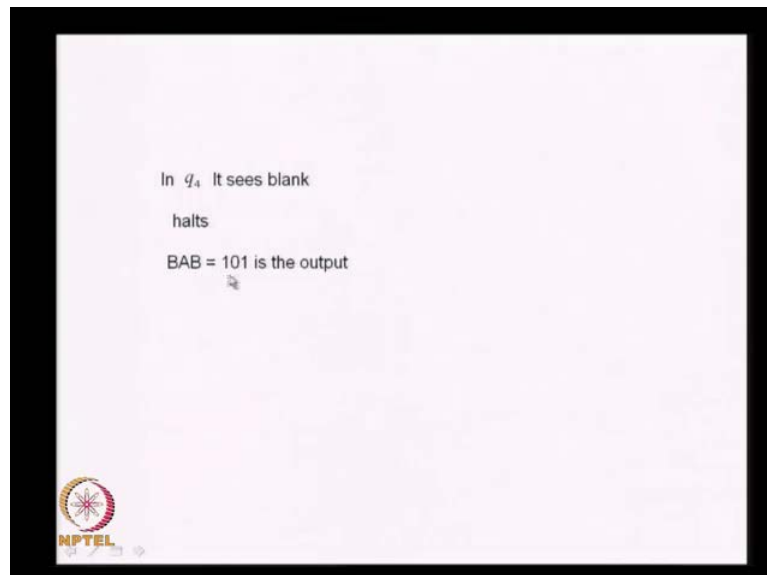
So, in this position, it just starts moving left and the state is q_2 now, it moves and at this position it prints A and moves right. In q_4 , it just moves right and when it encounters the first one, it changes that into a x by this move, δ of $q_4 1$ is equal to $q_1 x R$ and goes to the next cell. It reaches the blank in state q_1 again; that means, it has encountered an odd number of 1 's. And, starts moving the left, in state q_3 , it moves left in state q_3 until it reaches this blank, it prints B and starts moving right in q_4 .

(Refer Slide Time: 29:07)



Now, we have the binary representation of five here B A B and in q_4 it moves right looking for the next one to change, but it does not encounter 1, there is no 1 here all the one's have been changed to x. So, it reaches the blank symbol in q_4 ; that means, it has converted all the 1's into x's and the process is over.

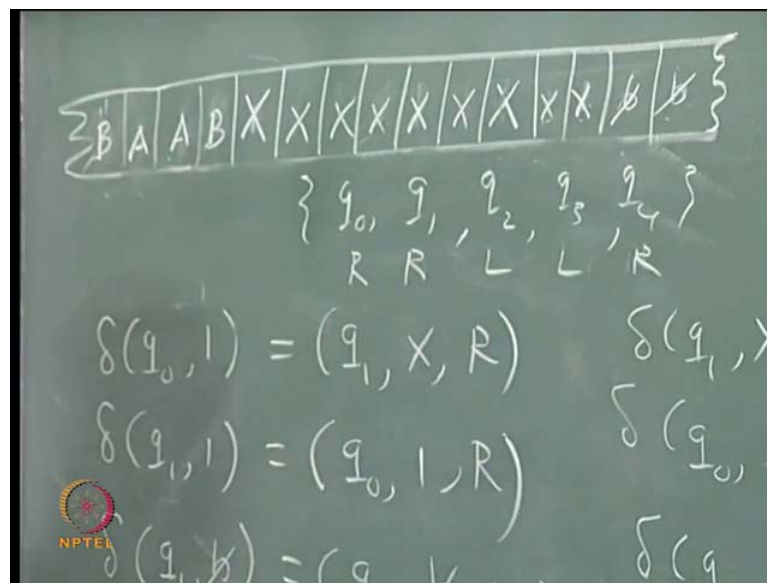
(Refer Slide Time: 29:48)



So, it halts at this position, in q_4 when it sees a blank it halts, at that time the output is BAB of course, followed by some x's. But the output you are expected to have is in terms of B A and B, B standing for 1, A standing for 0, and it gives the binary number

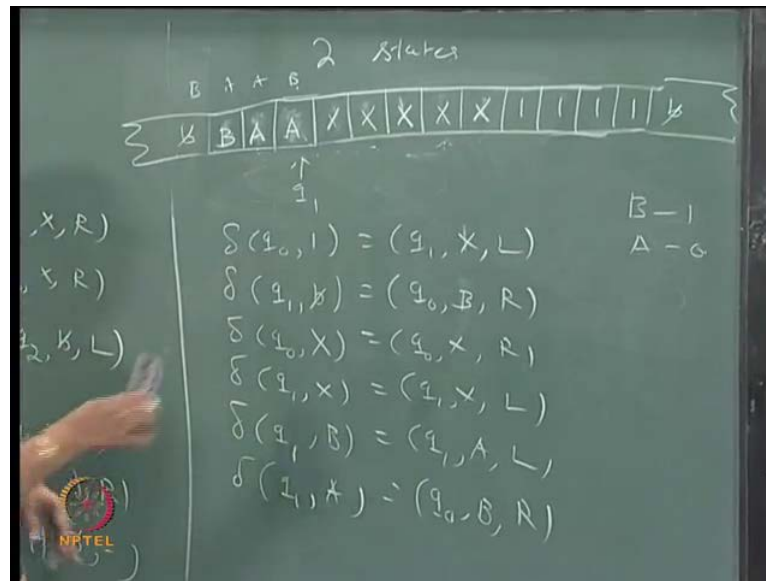
five. So, this is the way the unary to binary convert works. Now, the same problem, same input, same output in the same position, same work. I mean, I would not say same work the input is same, the output is also B and A and same position. Do it with two states, now, how many states we have used here? q_{naught} q_1 q_2 q_3 q_4 5 states, I have used and you must realized that q_{naught} is the right moving state, q_1 is a right moving state, q_2 and q_3 are left moving state. When you go to q_2 move left, when you go to q_3 it start move left ,when you go to q_{naught} you start moving, q_1 is right moving, q_4 is also right moving .

(Refer Slide Time: 31:05)



So, out of the 4 5 states, q_{naught} q_1 q_2 q_3 q_4 , this is right moving, this is right moving, this is left moving, this is the right moving, right, is it not, so?

(Refer Slide Time: 31:30)



Now, the achieve the same effect with two sticks, same input output, do it with two sticks, the number nine is given or maybe blank.

(())

(No audio from 31:03 to 32:24)

Ultimately I want B A B here, the machine starts here then state q naught. So, delta of q naught 1 is q 1 x L. You are going to have this number by using it as a binary counter initially will have one, then increase it to two, three and so on. So, first when it sees a 1 it changes that into a x, moves right in q 1, in q 1, now it has seen. So, it has to print a 1 that is A. So, delta of q 1 blank is q naught A right. So, it prints A, and I am sorry it prints B sorry prints B, B is 1. So, it prints B and moves right, B is 1 and A is 0, B is 1 A is 0. Now, you have two states. So, one must be a right moving state another must be a left moving state. So, in q naught, when you see x, you just move right, then when you see a 1 in q naught changes this into x, and start moving left in q 1. So, in q 1, if you see a x just move left.

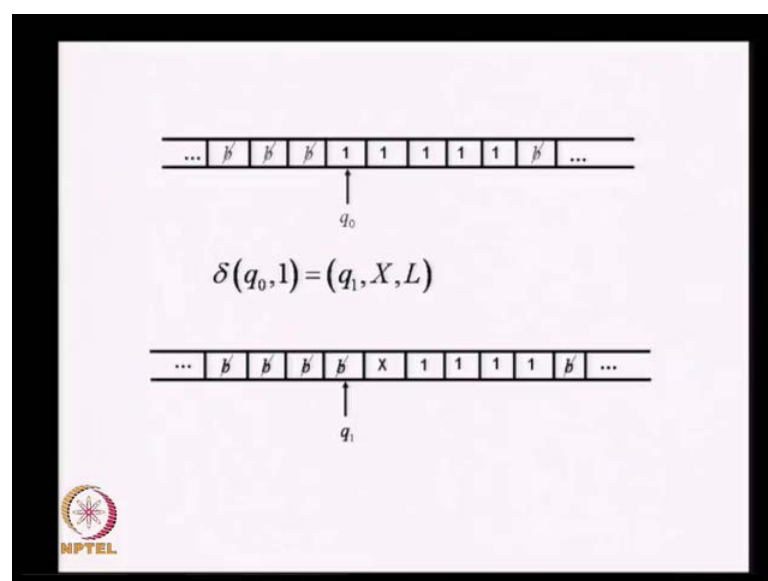
Now, in q 1, when you see, now you have change two of them into x's, now we should have the number two that is B and A, you must have. So, when in q 1, you see a B changes into A and move left changes this into A and move left in q 1. When you see a blank print B and move right q naught. So, now, you print a B and start moving right in q

naught. So, now, you have change two of them into x's and the number 2 is appearing here in binary. So, move like this, in q naught, you changes into a x, come back in q 1. In q 1 when you see a number 3 should appear here, that is B B should appear here. So, A will be changed into B q naught B .you will start moving in q naught in q naught you move over the x's and come here q naught 1 is q 1 x.

So, you change this into x move then in q 1 when you see B, change this into a A, you a move left again, when you see B change this into A move left, when you see a blank print B and so on. Now four of the once you have converted into x and you are having the number four, now do the same thing, change this into x and come back here. You will be seeing A, then you go to q naught print B, move right, change this into x, in q naught you move right, in q 1 you move left, q 1 you come here. When you see B change this into A move left, then when you see A change this into B. Now, I have numbers six 11 0, I have B B A'S 11 0. Now, again move here, change this into x, come back in state q 1, change this A into B, seven I have.

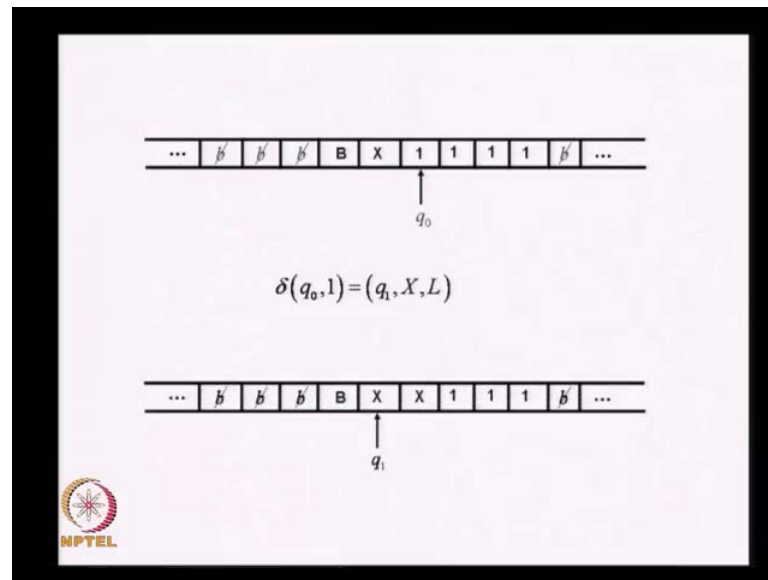
Now, go changes into x and when you come back changes B into A, move left changes B into A changes B into A, then when you see the blank print B, again go, change this, come back, changes into B. Now, in q naught if you are suppose to see A 1 delta of q naught blank, you halt.

(Refer Slide Time: 38:14)



Let us see how this works on the unary number 5? we have this situation five 1's and it makes use of the move delta of $q_{naught} 1$ is $q_1 x L$, q_{naught} is a right moving state, and q_1 is a left moving state, using this move, it becomes like this. Now in q_1 , when it sees a blank, it prints 1, that is B.

(Refer Slide Time: 38:41)



And moves right, it uses this mapping prints B, and moves right in state, q_{naught} , q_{naught} is a right moving state, it moves over the x's. And, when it first sees the 1 in this situation, it converts that into x and starts moving left. Now 1, 1 has been changed into x and you have the number 1 here. B denotes 1. When it change the 1 into x, you have to move to left and have the number 2 here that is B A you must have. Let us see how it happens? in $q_{naught} 1$, it changes the 1 into x and moves left, in q_1 it sees x again moves left using this mapping. And, when it sees the capital B in state q_1 , it changes this into A using this mapping and moves left. So, it prints a B here, using this move, prints B and starts moving right.

In state q_{naught} , it goes over everything and changes the next 1 into x and starts moving left, it proceeds like this. At this situation it has converted four 1's into x. And, you have the number four, here the tape head points to the last one, it is seeing the 1 in state q_{naught} , it changes that into x. And start moving left in state q_1 , now, you have the number four, here when it sees the in state q_1 , it changes that into a B and starts moving

right. It gives this, move delta of q 1 is equal to q 1 B R changes that into B and starts moving right.

Now, in q naught it looks for one, but it does not get 1, it keeps on moving to the right until it sees the blank, in q naught at this stage, it should be understood that it has converted all the 1's into x's and so it halts. So, the number five is again here, the output the binary number for 5 is this, the unary number has been converted into the binary number. Note that in this two examples which converted the unary number to a binary number. The same problem is treated, but you are achieving the result in two different ways, which is advantages, which one is advantageous to use, this one or the earlier one?

(())

The earlier one, the number of steps will be less, even though you use 5 states, the number of passes you make will be less. Here the number of passes each one you have to move right and move right and so on. So, number of steps will be more here, that also matters. So, these are some examples of having the Turing machine as a input output device. So, you have now consider the unary to binary converter, construct the binary to unary converter. You can try that or binary to decimal converter. Whatever it is? Something you can have and you can multiply two numbers, you can add two numbers, all those things can be done. In fact, whatever you can do with any, for whatever you can write a program, you can do for this.

But you must also realize that finite state automaton you can use for adding two numbers, arbitrarily large binary numbers, you can use a finite state automaton. But you cannot use the finite state automaton to multiply to arbitrarily large binary numbers. Why? Because it has finite amount of memory, for example, suppose we can have a finite state automaton with n states for doing the multiplication. Then consider some 2 power n plus 1 or something like that it will be followed by 1 followed by some zeros 1 followed by zeros, this will be the set of input you have to remember the partial (()) that is not possible. So, at the end, you when you after getting 0 0 0 0 like that, it will get into a pattern. And, when it reads 1 1, it just cannot change and do something.

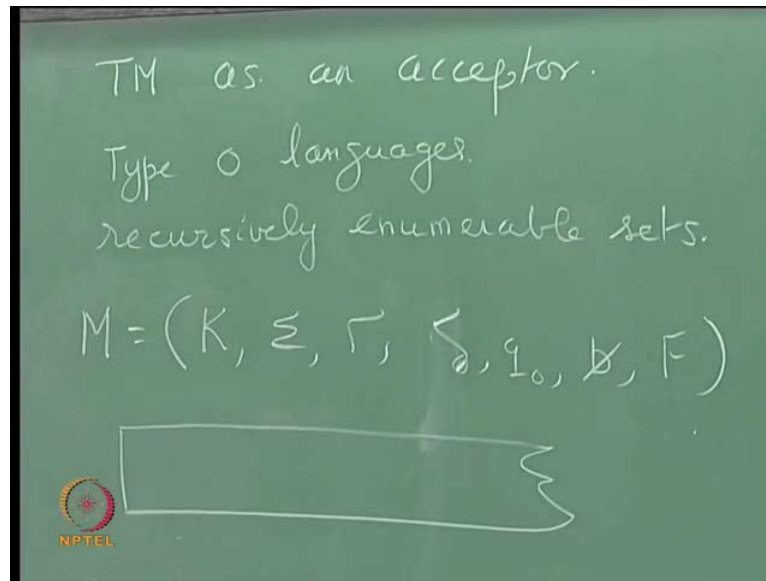
The finite state automaton has only a finite amount of memory. But the whole computer system itself, we can look at it as finite state automaton even though, what it can do is, see the number of each, count the number of bits in the memory, number of bits in the

registers. So, each bit can be a 0 or 1. So, I will just put it like this, the main memory there are p bits. So, many bits in multiply by 8 or something like that and registers. You have some q bits, then totally you have p plus q bits each one can be 0 or 1. So, there will be the whole system, when you say it is in a state, there will be two power p plus q states, it is a very large number, can look at it as a finite state automaton. But that number of states is extremely large and when something takes place, it changes from one state to another.

And, what sort of a program will have the maximum possible running time on such computer? just for curiosity, something which uses the main memory as a counter, everything will be 0 0 0 0 0 0 0 1 then increment by one, get two, three like that as a huge accumulator, if you like consider the whole memory, you will achieve all states. Is it not? The maximum possible running time we can have by achieving all states. So, and that sort of a thing will happen only when you look at it as a whole memory, as a counter keep 0 0, then increment by 1 and so on. If you write a program that will set of that will have the, it will go through all the states if you look at it as a finite state automaton.

Otherwise anything will repeat without, see you may have a program, it gets into an infinite loop or something like that, but then it will get into loop in such a way that it goes back to the original state, it will not pass through all the states. When (()) when you wanted to pass through maximum number of states possible you have to do like this. Consider it as a gigantic accumulator and keep on having them on this.

(Refer Slide Time: 46:52)

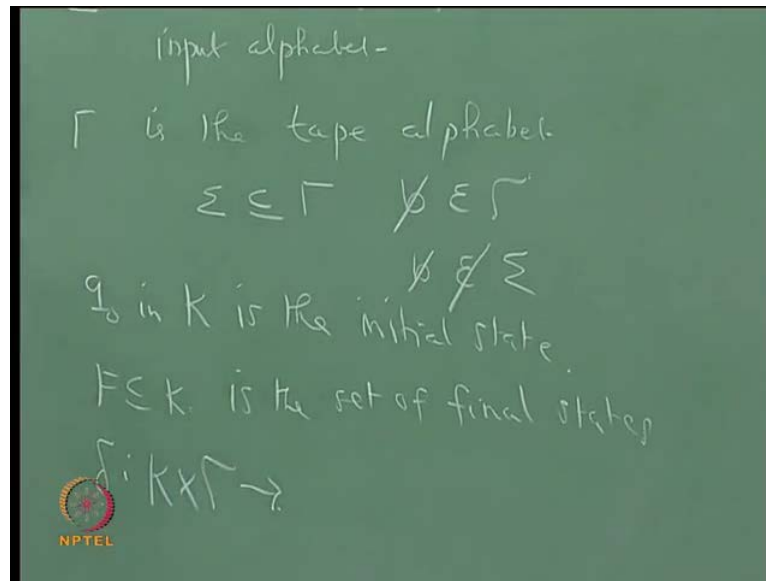


Now, Turing machine as an acceptor, we look at the Turing machine as an acceptor, it accepts some strings. Type 0 languages or recursively enumerable sets are accepted by Turing machines. It accepts type 0 languages; they are also called recursively enumerable set.

(No Audio From: 47:25 to 47:41)

Now, when you define a Turing machine as an acceptor k sigma gamma q naught delta q naught, sometimes this blank will not be denoted in the seven I am taking it has a seven tuple, sometimes it will also denoted as a six tuple in some books we denoted like this. Now, when we look at it as an acceptor the machine is taken like the tape is left end we are fixing now, it is infinite in one direction only now, because later on we will prove that two way infinite is equivalent to one way infinite tape. They are actually equivalent models, but when we look at it as an acceptor we fix the left end, right end it is infinite. So, initially the input is given here followed by blanks, this is a input. The machine starts the initial state reading the left most symbol, here the where the machines starts is fixed. When you look at it as an input output device, where it starts you have to specify, it is not necessary it should start the left, a left most nonblank symbol or something like. You can start anywhere that you have to specify. But here the machine starts on the left most symbol and it keeps on moving in the same manner until it reaches a final state. When it reaches a final state without loss of generality, you can assume that it halts.

(Refer Slide Time: 50:01)



So, when reaches a final state it will stop and accept the input. Here the thing is k is a finite set of states, σ is the set of input symbols that is a input alphabet, γ is the tape alphabet, σ is contained in γ , blank belongs to γ , but blank does not belong to σ . Blank symbol is denoted like this, it is a symbol of γ , but it is not a symbol of σ . δ we have to specify, $q_0 \in K$ is the initial state, $F \subseteq K$ is the set of final states, δ is a mapping from $K \times \Sigma \rightarrow L R$, this is deterministic, please note that this is a deterministic version we are considering .We could also consider nondeterministic version, that we shall see later.

Next, we have to define what is an I d and what is the relationship between I d's and so on and what is the string accepted, how do you define the language accepted and so on. So, we shall continue in the next class.