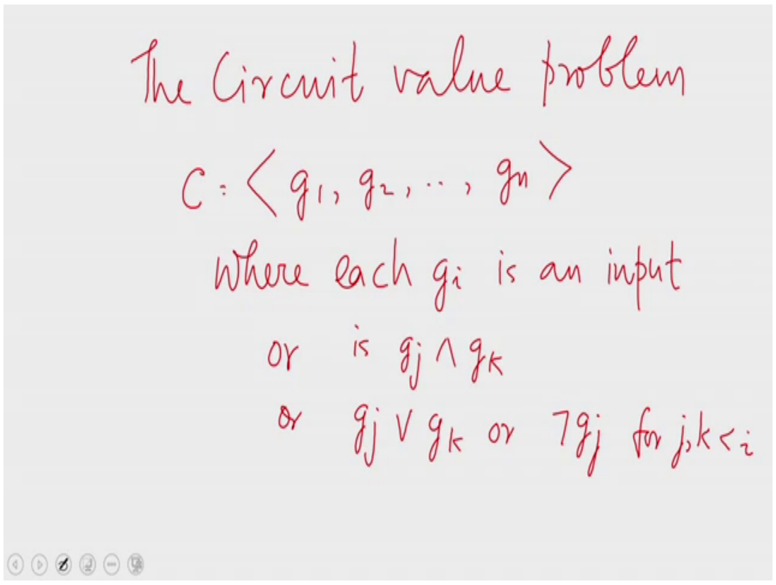
Parallel Algorithms
Prof. Sajith Gopalan
Department of Computer Science & Engineering
Indian Institute of Technology, Guwahati

Lecture – 35
Circuit Value Problem is P-complete for NC-reductions

Welcome to the 35th lecture of the MOOC on Parallel Algorithms. Today we shall see the first P complete problem for NC reductions this is the Circuit Value Problem.

(Refer Slide Time: 00:45)



The Circuit value problem

$C : \langle g_1, g_2, \ldots, g_n \rangle$

where each g_i is an input

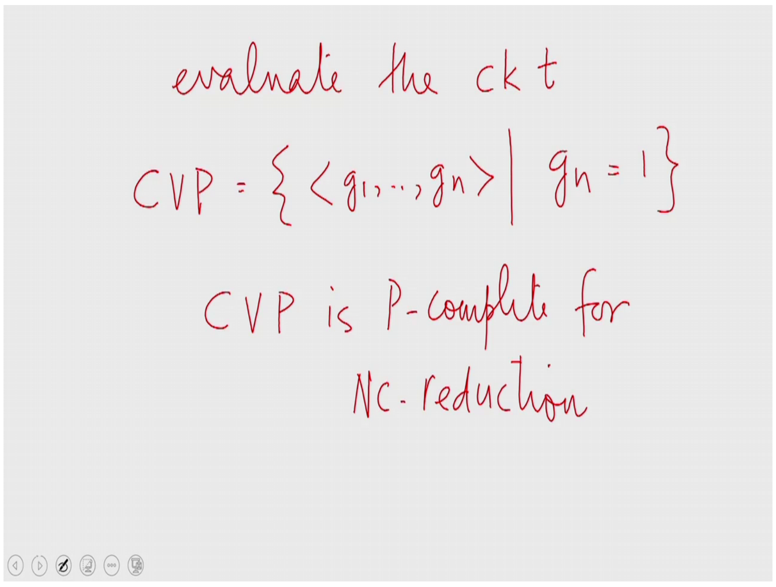
or is $g_j \wedge g_k$

or $g_j \vee g_k$ or $\neg g_j$ for $j,k < i$

So, let us begin with the definition of the Circuit Value Problem. The circuit value problem takes an instance of a circuit, where each g i is an input or is g j and g k the end of two previous gates or g j or g k or negation of g j for j k less than i.

So, an order tuple of this form is called an instance of the circuit value problem. What is required in the problem is to evaluate the circuit.

(Refer Slide Time: 01:38).

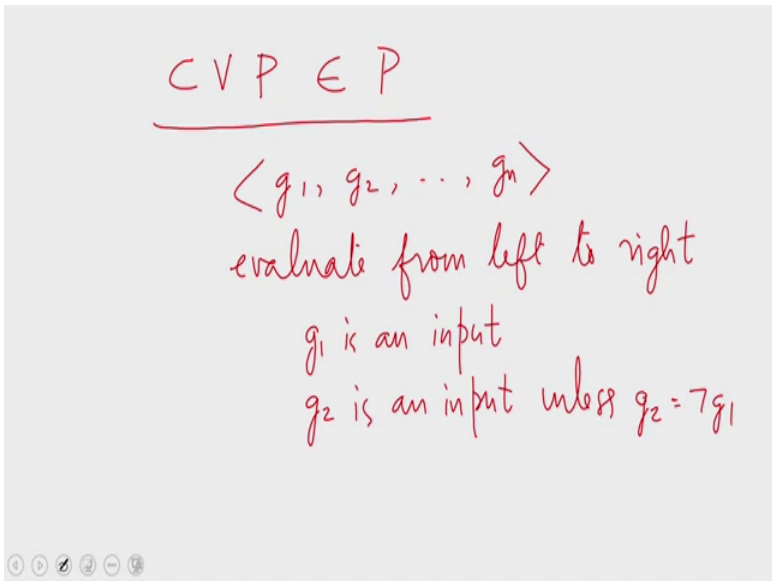

evaluate the ckt

$CVP = \{ \langle g_1, \ldots, g_n \rangle \mid g_n = 1 \}$

CVP is P-complete for NC-reduction

So, we define the circuit value problem as the set of all ordered pairs g 1 through g n such that g n evaluates to 1. We shall show that CVP is P complete for NC reductions, from the definition of P completeness for NC reductions we know that a language is P complete for NC reductions, if it belongs to P and then every language in P, NC reduces to this language.

(Refer Slide Time: 02:24)


$\underline{CVP \in P}$

$\langle g_1, g_2, \ldots, g_n \rangle$

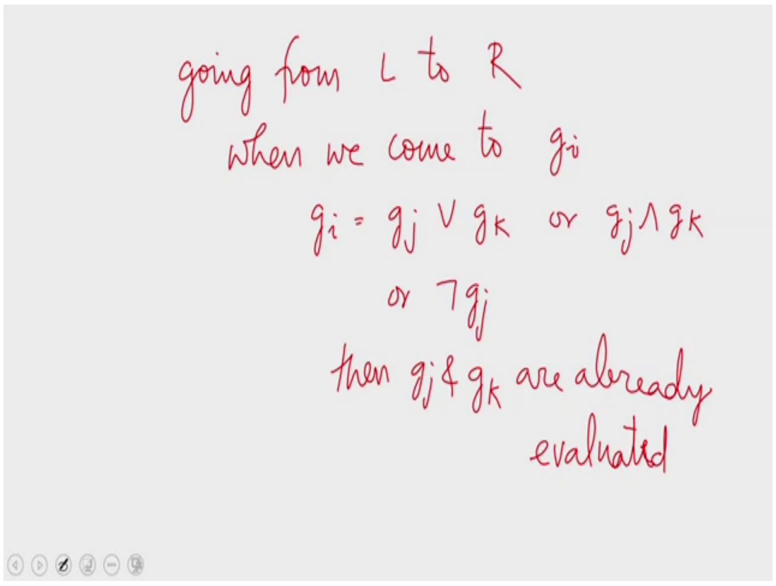
evaluate from left to right

g_1 is an input

g_2 is an input unless $g_2 = \neg g_1$

So, for C V P to be P complete for NC reductions we must first show that C V P belongs to P, but this is easy to show. When we are given a circuit of this form where each g i is either an input or an AND OR or negation of previous gates, then you can evaluate this circuit going left to right. Here, g 1 is necessarily an input, g 2 is an input if it is not a negation of g 1 unless g 2 is the negation of g 1.

(Refer Slide Time: 03:11)



going from L to R

when we come to g_i

$g_i = g_j \vee g_K$ or $g_j \wedge g_K$

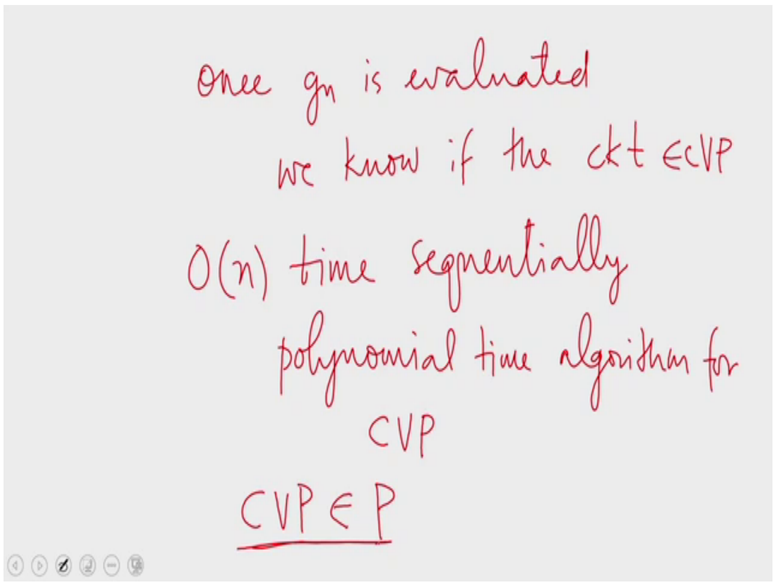
or $\neg g_j$

then g_j & g_K are already evaluated

So, this will form the basis and then going from left to right, when we come to i that is when we look at gate g i. If g i is g j or g k or g j or g j and g k or negation of g j, then g j and g k where, j and k are less than i are already evaluated because we are going from the left to the right. Therefore, g i can be evaluated, the only other possibility is the g i is an input itself in which case g i is readily available.

So, either way when we move from the left to the right the gates can be evaluated one at a time. So, when we reach the right end g n would be evaluated.

(Refer Slide Time: 04:05)



Once g_n is evaluated
we know if the ckt $\in$ CVP

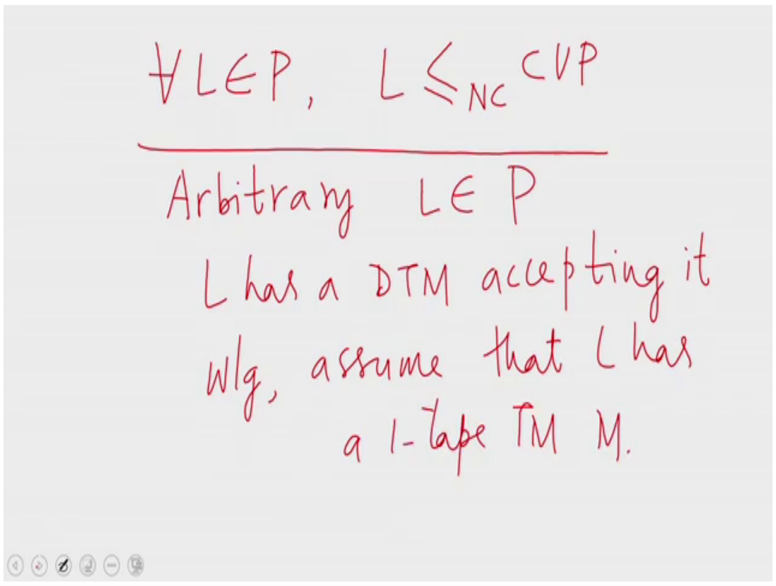
$O(n)$ time sequentially
polynomial time algorithm for
CVP

$\underline{\text{CVP} \in P}$

So, all that is necessary is to evaluate g n once g n is evaluated, we know if the circuit belongs to cvt or not. If the circuit belongs to CVP then the circuit evaluates to 1, otherwise the circuit evaluates to 0.

So, the evaluation of the circuit can be done in order of n time sequentially, which means there is a polynomial time algorithm for CVP. In other words, CVP belongs to P. So, the first requirement is met, for CVP to be P complete for NC reductions it has to be in P first of all that is satisfied.

(Refer Slide Time: 04:53)



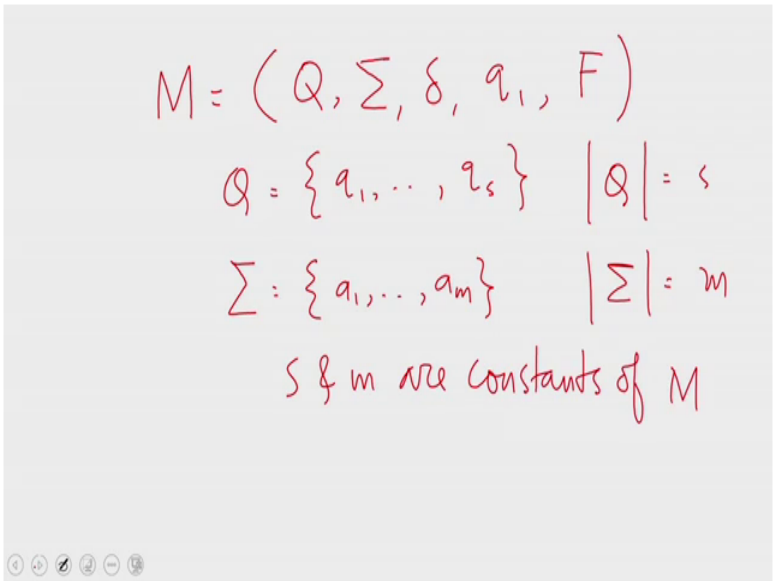
$\forall L \in P, \; L \leq_{NC} \text{CVP}$

Arbitrary $L \in P$
L has a DTM accepting it
wlg, assume that L has
a 1-tape TM M.

So, now what remains to show is that, for every language L belonging to P, L NC reduces to C V P. But then how do we show that? Let us say we take an arbitrary language L in P.

So, L is an arbitrary language in P, but then P is the class of all languages that are solvable in polynomial time use in a deterministic turing machine. Therefore, we can assume that L has a deterministic turing machine, without loss of generality we can assume that this machine is a 1 tape turing machine. Because always a multi tape turing machine can be simulated on a 1 tape turing machine. So, let us assume that, L has a 1 tape turing machine.

(Refer Slide Time: 05:39)



$M = (Q, \Sigma, \delta, q_1, F)$

$Q = \{q_1, \ldots, q_s\}$ $|Q| = s$

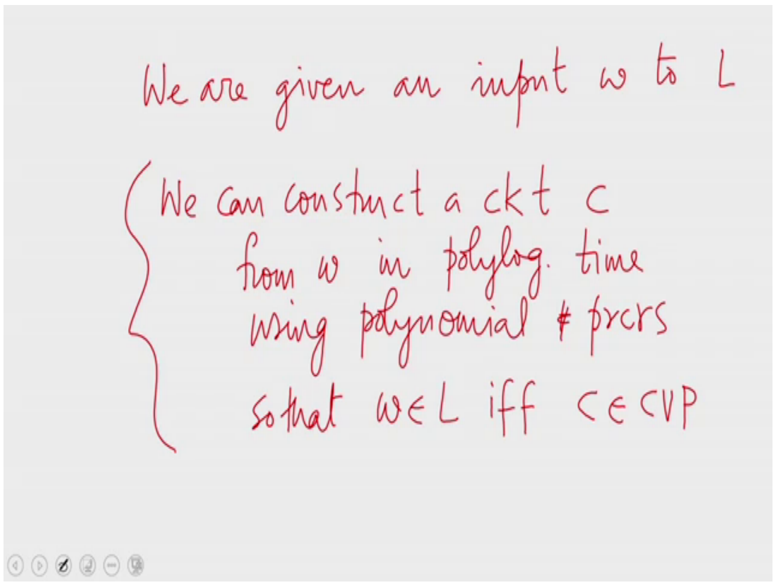
$\Sigma = \{a_1, \ldots, a_m\}$ $|\Sigma| = m$

s & m are constants of M

So, let us say M is made up of these tuples Q is the set of states, sigma is the alphabet, delta is the transition table, q 1 is the start state and f is the set of final states. Let us say the set Q is made up of s states.

The state space has a size of s let us say and let us say sigma is a 1 through a m this mod sigma is m. So, s and m are constants of the machine.

(Refer Slide Time: 06:28)



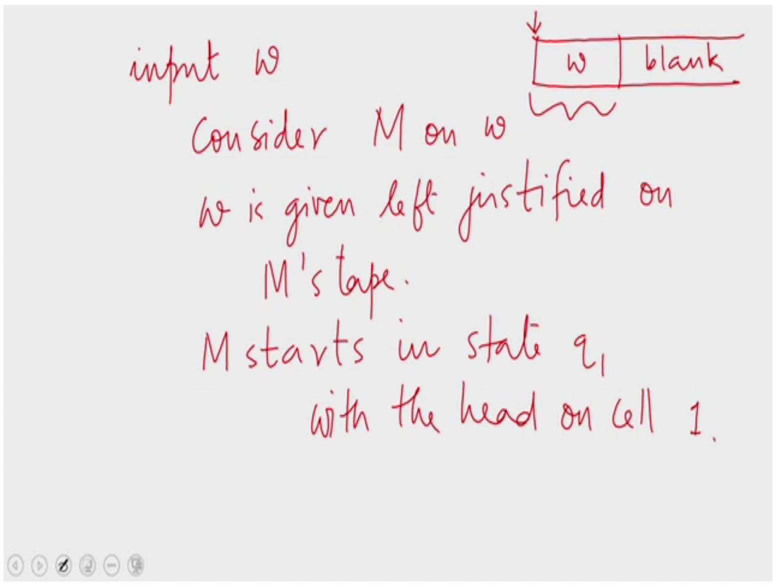
We are given an input w to L

We can construct a ckt C from w in polylog. time using polynomial # prcrs so that $w \in L$ iff $C \in CVP$

And then let us say we are given an input W. We can construct a circuit, from W in poly logarithmic time, using polynomial number of processes so that W will belong to L. if it only if C belongs to CVP. So, what we are going to show is that given an arbitrary language L and an arbitrary input W to L, we can construct a circuit C from W in poly logarithmic time using a polynomial number of processors.

So, that W will belong to, if and only if C will belong to CVP. Therefore, if we have an algorithm for checking membership in CVP, then that algorithm can be use for checking membership in L of W. So, this is what we need to show, to establish that CVP is P complete for NC reductions. So, how do we do this?

(Refer Slide Time: 07:35)



input w

Consider M on w

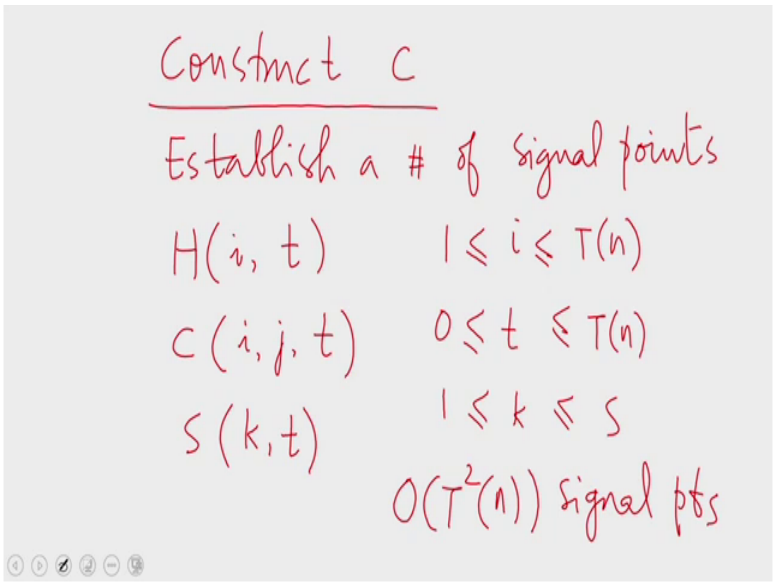
w is given left justified on M's tape.

M starts in state q_1 with the head on cell 1.

So, we have the input w; consider the execution of this machine on w. So, let us assume that w is, given left justified on M’s tape. So, w is the input to M and that is given left justified in the tape. If this is the tape then w is given at the left end, the rest of it is we as assumed to be blank. The rest of the tape is assumed to be blank. So, the input is given at the left end of the tape and we assume that the machine starts at the leftmost cell.

So, M starts in state q 1, with the head on cell 1. We assume that the cells are numbered consecutively from the left end so the leftmost cell is number 1.

(Refer Slide Time: 08:31)



Construct C

Establish a # of signal points

$H(i, t)$ $\quad 1 \leq i \leq T(n)$

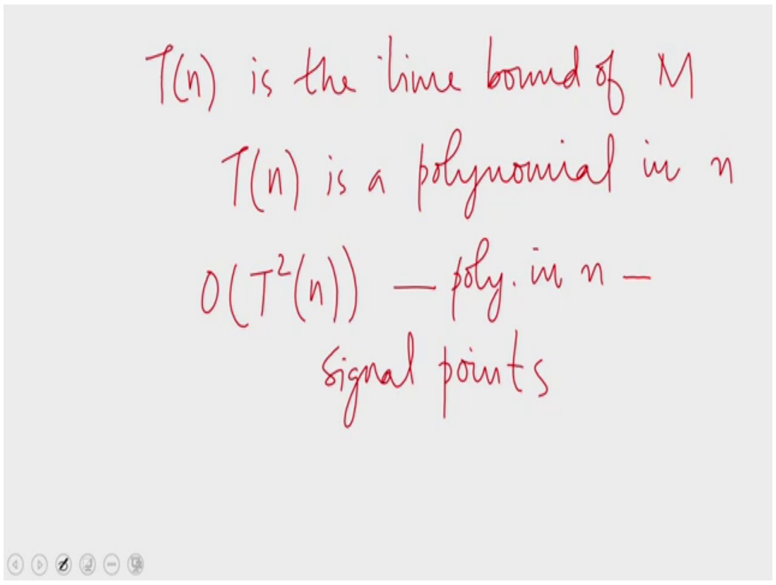
$C(i, j, t)$ $\quad 0 \leq t \leq T(n)$

$S(k, t)$ $\quad 1 \leq k \leq S$

$O(T^2(n))$ signal pts

Now, we are going to construct a circuit C; to construct C, we will establish a number of signal points. These signal points are denoted like this; we have signal points of the form H i t, we have signal points of the form C i j t, and we have signal points of the form S k t. Here; i varies from 1 to T of n, t varies from 0 to T of n, and k varies from 1 to S.

(Refer Slide Time: 09:23)



T(n) is the time bound of M

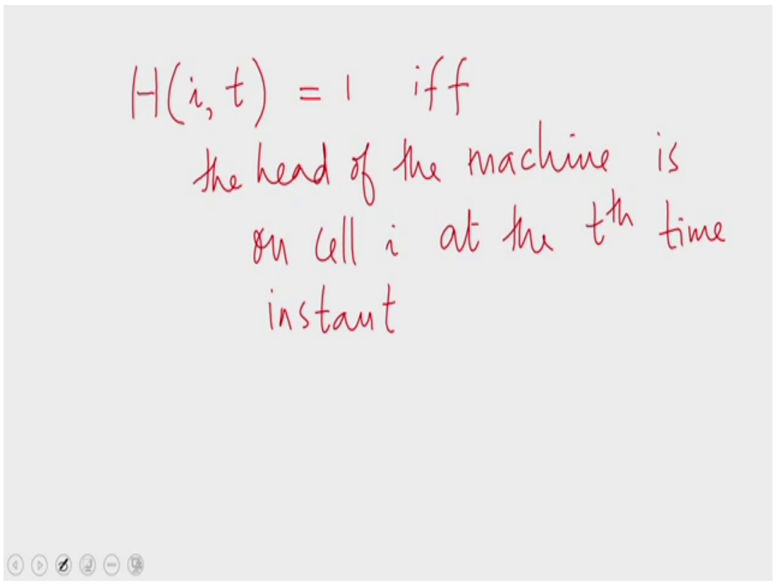
T(n) is a polynomial in n

$O(T^2(n))$ — poly. in n —

signal points

So, you can see that there are order of T square such signal points, T square of n signal points, where T of n is the time bound of the machine. That is when M is given an input of length n, we have a guarantee that M will come to a halt in T of n steps. And we also know that T of n is a polynomial in n that is because M is a polynomial time bounded machine. So, we have established order of T squared of n which is also a polynomial in n signal points.

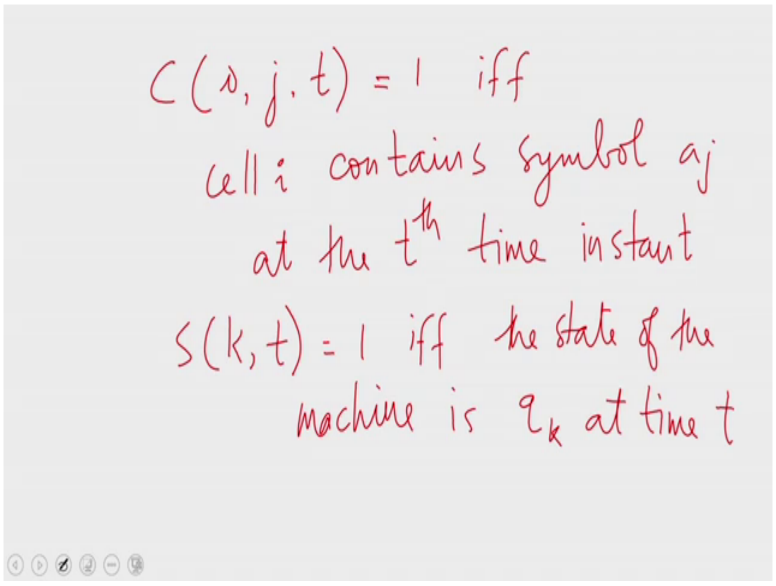
(Refer Slide Time: 10:00)



$H(i, t) = 1$ iff
the head of the machine is
on cell i at the t^{th} time
instant

So, let us see what these signal points mean. H of i t is going to be 1, if and only if the head of the machine is on cell i, at the tth time instant. H of i t is going to be 1, if and only if the head of the machine is going to be on cell i at the tth time instant. Similarly C i j is going to be 1 if and only if cell i contains simple a j at the tth time instant.

(Refer Slide Time: 10:29)

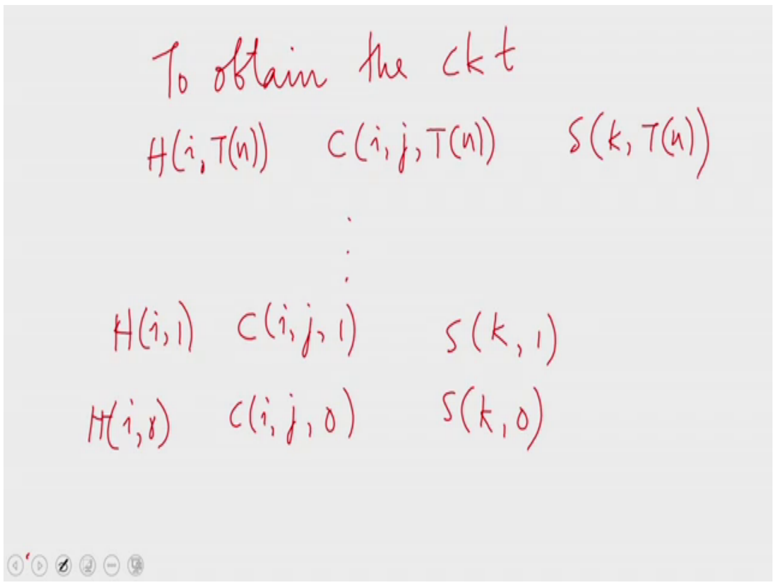


$C(i, j, t) = 1$ iff
cell i contains symbol a_j
at the t^{th} time instant

$S(k, t) = 1$ iff the state of the
machine is q_k at time t

So, now you can imagine what S k t is going to be. This is going to be 1 if and only if the state of the machine is q k at time t. So, we have order of T squared of n signal points of the sort and we are going to connect these signal points using gates to obtain a circuit.

(Refer Slide Time: 11:00)



To obtain the ckt

$H(i, T(n))$ $C(i, j, T(n))$ $S(k, T(n))$

$\vdots$

$H(i, 1)$ $C(i, j, 1)$ $S(k, 1)$

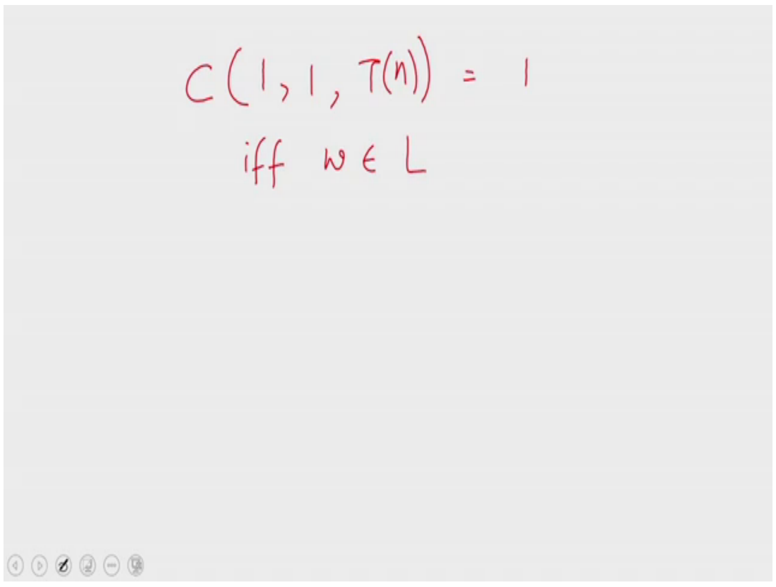
$H(i, 0)$ $C(i, j, 0)$ $S(k, 0)$

To obtain the circuit at the lowest level we have H i 0's, C i j 0's and S k 0's. So, these are the signal points corresponding to time 0 and then at the next level we have H i 1's, C i j 1's and S k 1's. And going all the way we come to H i, T n the algorithm is going to halt within T of n steps. So, we do not have to consider time that is beyond T of n.

So, we have now established signal points of the sort, now all that is necessary is to connect up these signal points using logic gates. And then we will have the entire circuit laid out. And now what is the circuit that we want? We want the circuit which will evaluate to 1, if and only if the string is accepted by the language. Now, let us assume a convention that, when the turing machine halts, if it is going to accept the input it will write 1 in the first cell. If it is going to reject the string it is going to write a 0 in the first cell.

So, all we have to do is to check whether the cell, the leftmost cell is going to contain a 1 at the time the algorithm halts. Which is at time T of n cell 1 should contain a 1.

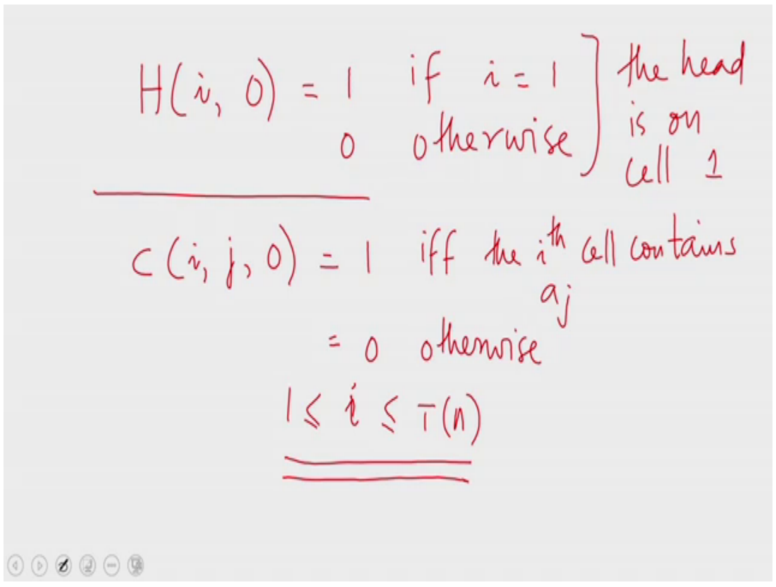
(Refer Slide Time: 12:31)



$$C(1, 1, T(n)) = 1$$
$$\text{iff } w \in L$$

This is precisely the case where C 1, which is cell 1 contains a 1. So, a 1 will be denoted using 1 again, T of n happens to be 1. This is if and only if W belongs to L. So, if you manage to connect up the signal points in such a way that the signals all will satisfy the intended meaning then we only have to check whether C 1, 1 T of n is 1 ok. Now, let us see how the circuit can be laid out.

(Refer Slide Time: 13:10)



$$H(i, 0) = \left.\begin{matrix} 1 & \text{if } i = 1 \\ 0 & \text{otherwise} \end{matrix}\right] \text{the head is on cell 1}$$

$$C(i, j, 0) = 1 \text{ iff the } i\text{th cell contains } a_j$$
$$= 0 \text{ otherwise}$$
$$1 \leq i \leq T(n)$$

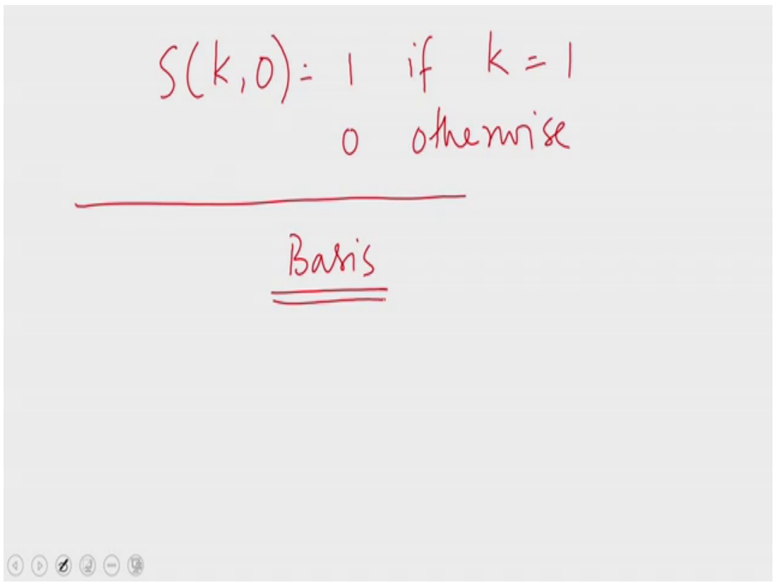
So, first of all we have to define, the inputs to the circuit. So, H i 0 is 1, if i equal to 1, 0 otherwise. What it means is that at time 0 that is when the machine starts up, the head is it the leftmost cell with the leftmost cell is numbered 1 therefore, H of i 0 is 1, if i equal to 1 and 0 otherwise. So, it says that H of 1 0 is true, and H of i 0 is false, for every i other than 1.

In other words the head is on cell 1. So, if we set H i 0 to 1, if and only if i equal to 1 then we have the intended purpose. The H 0 values are all appropriately set. Then let us consider C i j 0's, that is we want to assert that at the beginning of the machine cell i contains symbol a j, but then at the beginning of the machine which cell contains what is decided by the input therefore, cell i will contain a j precisely if the ith symbol of the input happens to be a j.

So, this is if and only if the ith cell contains a j, this will be 0 otherwise. So, you remember that here i varies from 1 to T of n, that is because a turing machine which is T of n time bounded, cannot use any more space than T of n. The machine in the worst case will be using 1 cell in, 1 new cell in every single step. Therefore, if the machine halts in steps it will use at most T of n cells, and we are accounting for all these T of n cells. Initially the input is written left justified on this tape the input has a length of n, if n happens to be less than T of n, then there will be some extra space that is use by the machine.

So, initially the machine will mark out these extra space, with blanks blank symbols and the input will be kept in the first n symbols. So, we have initialized C i j values for such T of n i values.

(Refer Slide Time: 15:39)



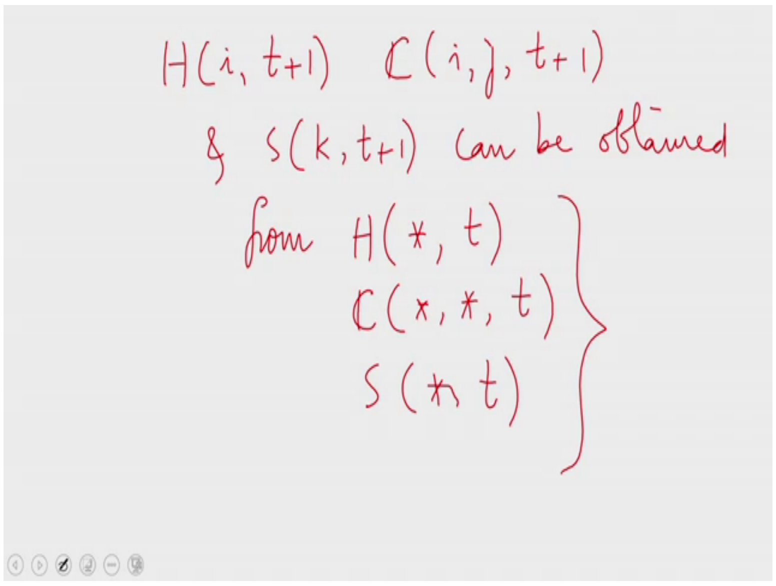
$S(k,0) = 1$ if $k = 1$

0 otherwise

Basis

So, C i j values are also appropriately set for time 0. And then S k 0 will be 1 if and only if k equal to 1 that is the state is supposed to be q 1when the machine starts up, it is 0 otherwise. So, these settings will form the basis and the settings are all appropriate the H i values C i j values and S k values at time 0 all represent the position of the head, the contents of the cell and the state at the beginning of the execution. So, this will form the basis.

(Refer Slide Time: 16:14)



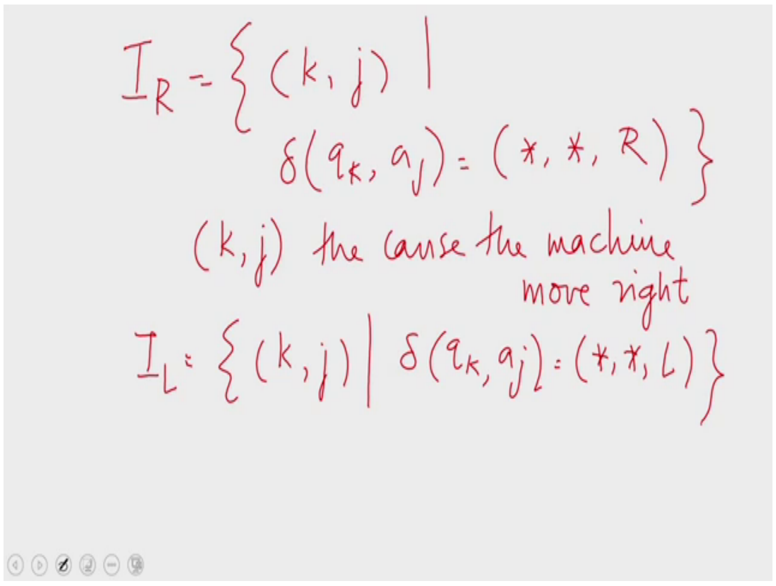
$H(i, t+1)$ $C(i, j, t+1)$

& $S(k, t+1)$ can be obtained

from $\left.\begin{array}{l} H(*, t) \\ C(*, *, t) \\ S(*, t) \end{array}\right\}$

Now, let us see how H i t plus 1, C i j t plus 1 and S k t plus 1, can be obtained from the H values at time t, the C values at time t and the S values at time t. So, using these values we will be able to establish the t plus 1 values for H C and S.

(Refer Slide Time: 16:51)



$$I_R = \{ (k, j) \mid \delta(q_k, a_j) = (*, *, R) \}$$

(k, j) the cause the machine move right

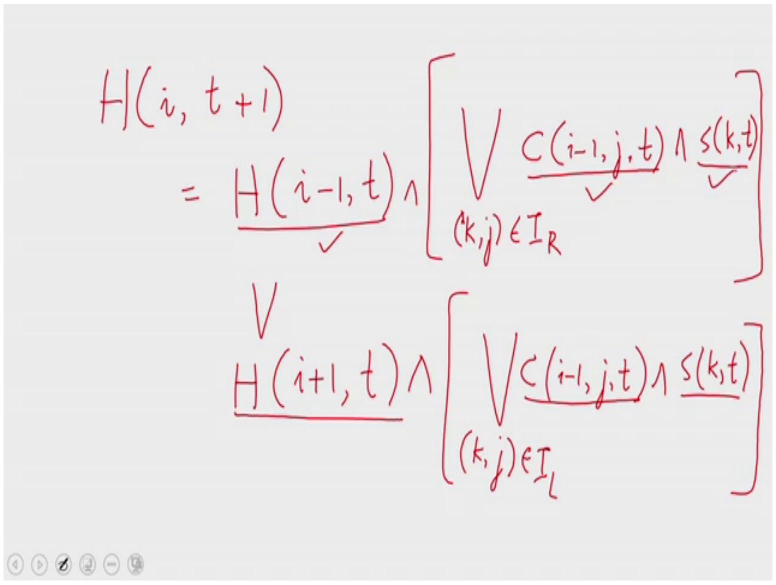
$$I_L = \{ (k, j) \mid \delta(q_k, a_j) = (*, *, L) \}$$

So, first let us consider H i t plus 1. For this let me define I R as this set. I R is defined as all ordered pairs k j. So, that delta of q k a j is a tuple of this form. Here the star denotes a wild card, it does not matter what it is.

So, I R is defined to be the set of all ordered pairs k j so that when the state of the machine is q k and the symbols can this a j. The machine has a transition of the sort, it will we do not care which state it enters and what is the new symbol that it is written, but we insist that the head should be moving right. So, these are essentially all k j ordered pairs which will cause the machine to have a right moment.

So, these are the k j pairs or state symbol pairs that cause the machine to move right, there is a read write head to move right. So, the machine has a rightward moment, when 1 such a ordered pair happens. Similarly let us define I L as the set of all k j pairs that cause left movements. So, here delta of q k a j must be a tuple of this form.

(Refer Slide Time: 18:13)



$$H(i, t+1) = \underset{\checkmark}{\underline{H(i-1, t)}} \wedge \left[\bigvee_{(k,j) \in I_R} \underset{\checkmark}{\underline{C(i-1, j, t)}} \wedge \underset{\checkmark}{\underline{S(k,t)}} \right]$$

$$\vee \; \underline{H(i+1, t)} \wedge \left[\bigvee_{(k,j) \in I_L} \underline{C(i-1, j, t)} \wedge \underline{S(k,t)} \right]$$

So, with these definitions, we can now define H of i t plus 1 that is why we want to assert that at time t plus 1 the head of the read write head; the read write head is going to be on cell i.

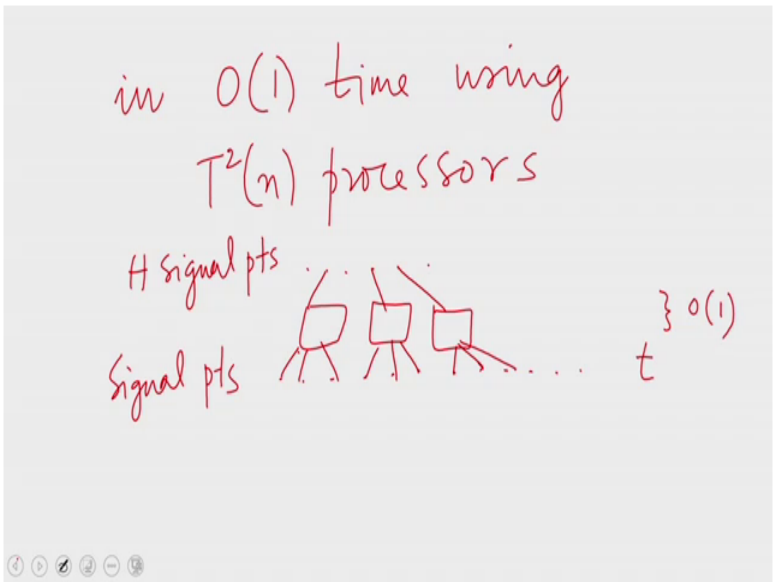
Now, this is possible, if either in the previous time instant the head was at position i minus 1. And the machine had a rightward moment. So, let us see what it says, this term asserts that at time t the head was on the i minus first cell, not only that for some k j belonging to i R, that is for some state symbol pair which causes the machine to move right. The i minus first cell contained a j, in the previous time instant and also the state of the machine was q k in the previous time instant.

So, in the previous time, time instant we had the composition of q k and a j, the state was q k and the cells can contained a j. Now this pair is a right movement inducing pair therefore, the machine would have moved right. So, what we are asserting is that in the previous instant the head was on the i minus first cell and that i minus first cell contain some a j and the state happened to be some q k. And this q k and a j were so, that they would have induced a right movement in the machine. So, the machine would have moved to the next cell which is the ith cell. That is how the head came to be in cell i in the t plus first step.

Now, this is one possibility, that is the head could have come to the ith cell through a right moment, but then the head could have come to the ith cell through a left moment as well. So, that is the other possibility. So, either this could have happened or it could be that in the previous instant, the head was on the i plus first cell and there existed some left movement causing k j pair. So, that this was the k j pair that was prevalent in the tth time instance, if either of this happens then the head would end up being in the ith position in step number T.

Now, if you have these signal points we know H i minus 1 t, H i plus 1 t and let us say we know C i minus 1 j t, S k t, C i minus 1 j t and S k t we have all these signal points for every k j belonging to I R and for every k j belonging to I L. Then we can construct a Boolean Circuit for H i t plus 1 using AND, OR and NOT gates. In fact, we do not need a NOT gate here, we need only and AND, OR gates.

(Refer Slide Time: 21:14)

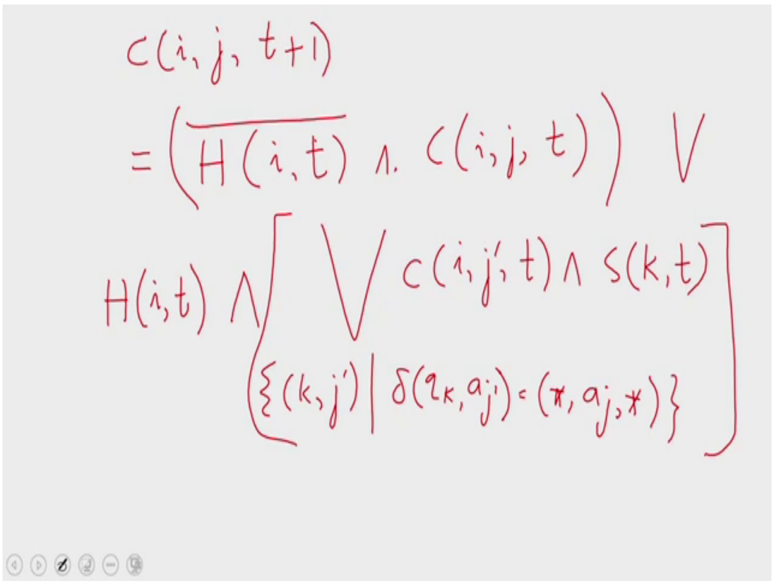


But then such a circuit can be obtained, in order of 1 time using T squared of n processors. So, what we say is that from the signal points at level t. So, these signal points involved the H signals, C signals and the S signals from all of these we can obtain the H signal points by appropriately setting up circuits from the tth level to the t plus first level.

Each box we will contain a set of and or gates and the depth of the circuit is going to be order 1, as you can see this circuit this Boolean expression has only a constant depth. It is a disjunction where each disjunct is a where each term is a conjunct and so on. So, there is a constant depth to this Boolean circuit. So, the Boolean circuit can be established in order 1 time, using T squared of n processes that is, drawing off a setting up of this circuit will require only order of 1 time using T squared of n processors.

So, we know that all the H signals at level t plus 1 can be connected up to the signal points at the previous level using Boolean circuits in order 1 time. T squared of n is a polynomial because T of n is a polynomial. So, we know that this can be established in NC.

(Refer Slide Time: 22:46)



$$C(i, j, t+1) = \left(\overline{H(i,t)} \wedge C(i,j,t)\right) \vee H(i,t) \wedge \left[\bigvee_{\{(k,j') \mid \delta(q_k, a_{j'}) = (*, a_j, *)\}} C(i,j',t) \wedge S(k,t) \right]$$

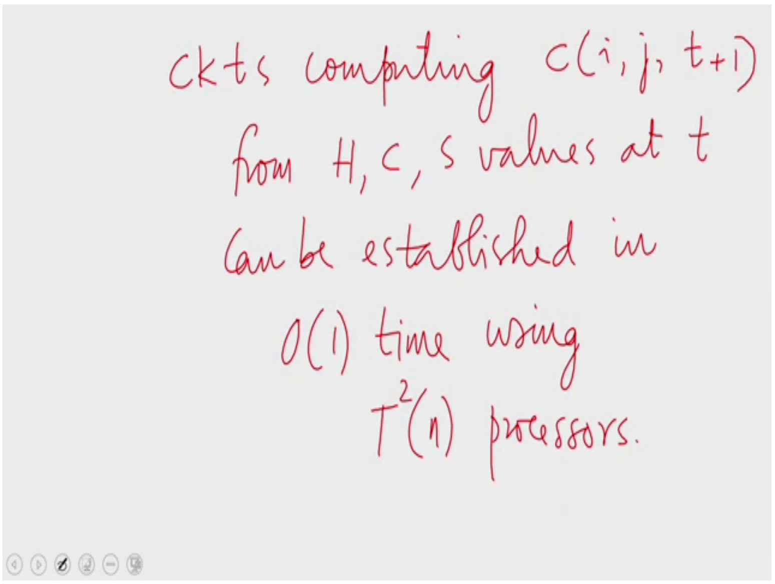
Now, let us come to the C i j t plus 1 signals. C i j t plus 1, wants to assert that cell i contains symbol j at time t plus 1.

Now, this is possible if the cell that the machine scanned in the previous step that is in the tth step was not i and cell i contained symbol j. What it means so the head was elsewhere. The head was not in the vicinity the head the head was not on cell i. If the head was not on cell i, the machine would not have been able to change the content of cell i in step t. And if cell i contained a j in step t and the head was elsewhere, then the content of cell i would continue to be symbol a j in the t plus first step as well.

So, this is one possibility this is one way, how cell i could contain symbol a j in the t plus first step. Another way is for the machine to actually change the cell to get a j in cell i in step t plus 1. That is possible only if the head was on the ith cell in the previous step and we take a disjunction over all ordered pairs k j prime, of C i j prime t and S k t. Thus we want to say that there x is an ordered pair k j prime so, that q k, a j prime is an a j writing move, that is if the state the previous state happened to be q k and this scanned cell contained a j in that case then the machine would have written a j in the cells scanned.

So, such a q k, such a q k a j prime was in contention in the previous step, in that case if the cell scanned was i and the cell contained a j prime and the state was q k in the previous step that is time t then the machine would have rewritten the content of cell i to a j. Because that was a j writing a j writing step so, this is how we can synthesize C i j t plus 1. So, you can see that it uses only H C and S signals of time t, it uses negation AND and OR, and the depth of this Boolean expression is only a constant.

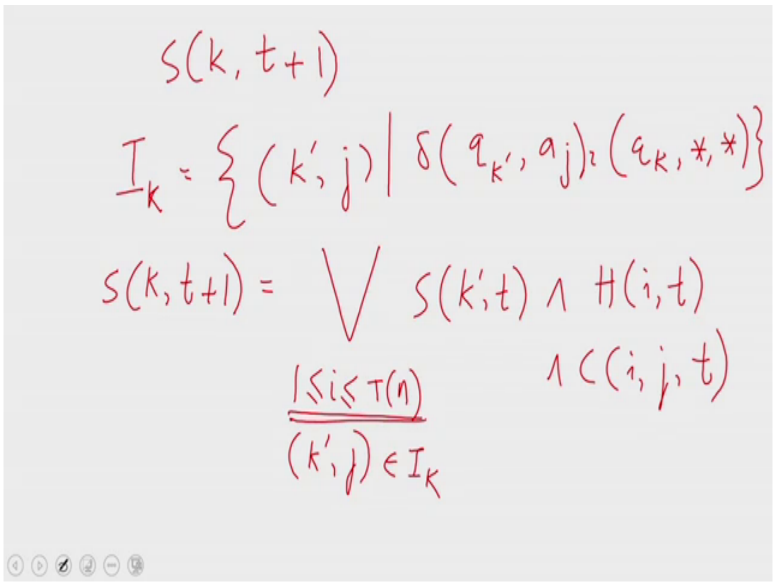
(Refer Slide Time: 25:51)



Ckts computing $c(i, j, t+1)$ from H, C, S values at t can be established in $O(1)$ time using $T^2(n)$ processors.

Therefore we see that C i j t plus 1 can also be that is circuits computing C i j t plus 1 from H C and S values at t can be established in order 1 time, using T squared of n processors on any pair.

(Refer Slide Time: 26:27)



$$S(k, t+1)$$

$$I_k = \{(k', j) \mid \delta(q_{k'}, a_j) = (q_k, *, *)\}$$

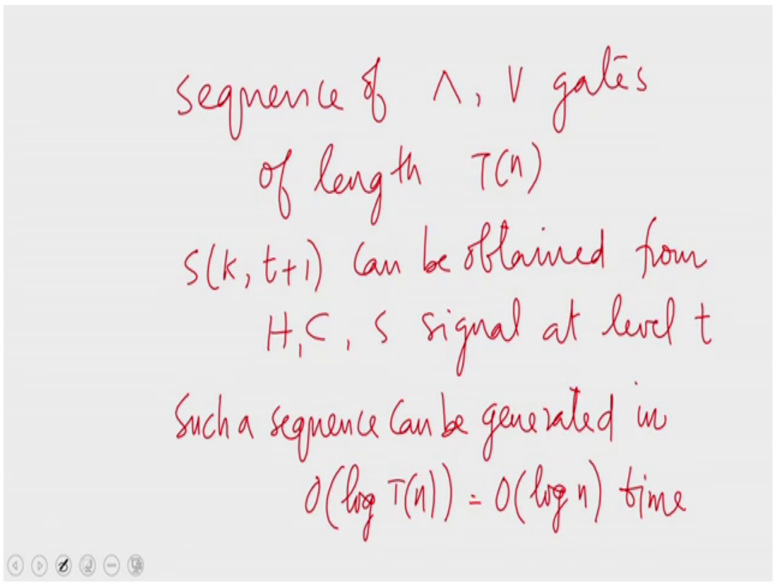
$$S(k, t+1) = \bigvee_{\substack{1 \leq i \leq T(n) \\ (k', j) \in I_k}} S(k', t) \wedge H(i, t) \wedge C(i, j, t)$$

Now, finally, we come to signals of the form S k t plus 1. For this we need to define a set I k. I k is defined as the set of all ordered pairs k prime j. So, that delta of q k prime a j happens to be q k with wild cards. So, I k is the set of all state symbol pairs q k prime a j so, that when such an ordered pair happens, the machine switches into state q k. So, these are ordered pairs which induce state q k.

Now, with I k we can write the expression for S k t plus 1, we want to say that the machine is in state q k at time t plus 1, this is possible if so what does it say? There access an I ranging from 1 to T of n, that is there access some cell on the tape the tape is going to use T of n cells. So, there is some cell on the tape so that, there is a q k switch inducing pair k prime j so that the state was q k prime in the previous step the head was on the ith cell the so, the head was on some cell and this cell contained a j. Where a j is the other half in this q k inducing pair, if that is the case then the state of the machine should be q k in step t plus 1 as indeed would be the case.

If there is some cell i which contained a j in the previous step and in the previous step the state had been k prime and this k prime a j k prime j pair happens to be a q k state inducing pair. In that case then in step t plus 1 the state has to be q k. Now, this is a much larger Boolean expression than the previous one because there are many more disjunctions here.

(Refer Slide Time: 28:37)



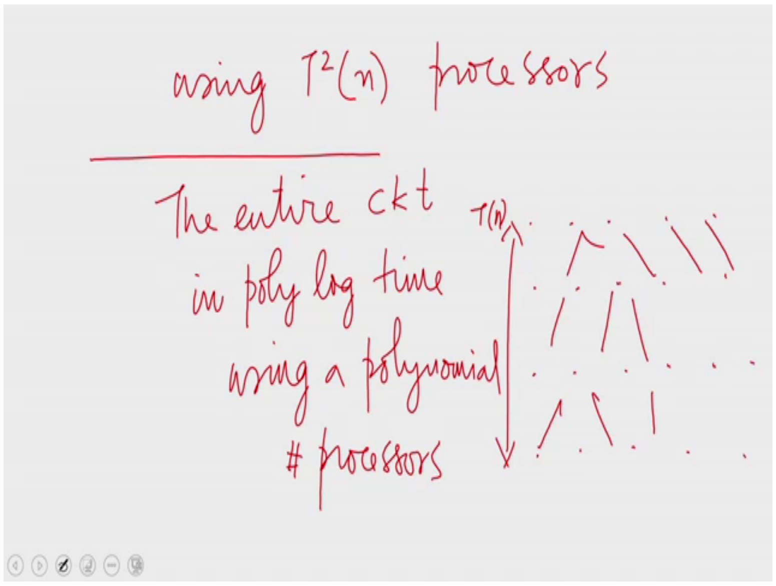
sequence of $\wedge, \vee$ gates of length $T(n)$

$S(k, t+1)$ can be obtained from H, C, S signal at level t

Such a sequence can be generated in $O(\log T(n)) = O(\log n)$ time

But still using a sequence of a AND OR gates of length T of n, S k t plus 1 can be obtained from H C and S signals at level k at level t. And such a sequence can be generated, in order of log of T n which is order of log n time using T of n processes.

(Refer Slide Time: 29:14)

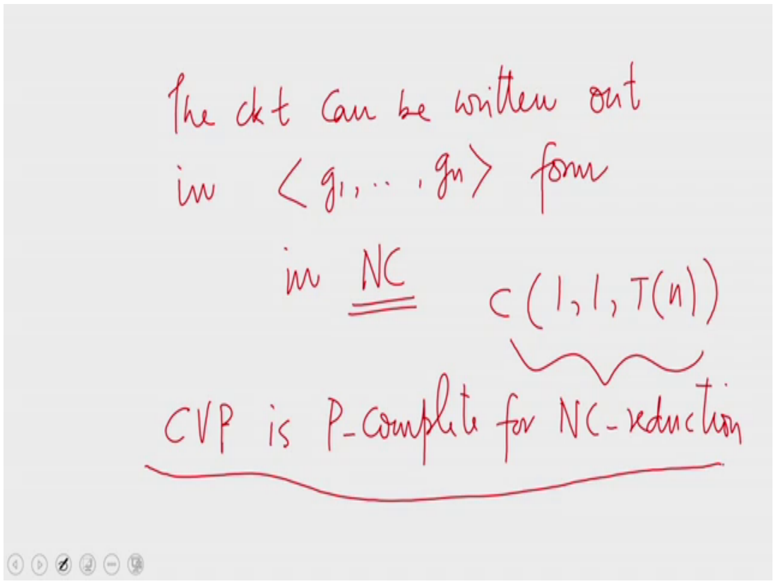


using $T^2(n)$ processors

The entire ckt in poly log time using a polynomial # processors

Using T square of n processes, what it means is that the entire circuit, the circuit with gates between the levels. So, we have T of n such levels, for times varying from 0 to T of n there are T of n plus 1 such levels. These levels can be interconnected with a separate set of processors for each adjacent pairs of levels, in poly log time using a polynomial number of processors. So, the circuit itself can be drawn up.

(Refer Slide Time: 29:50)



The ckt can be written out in $\langle g_1, \ldots, g_n \rangle$ form

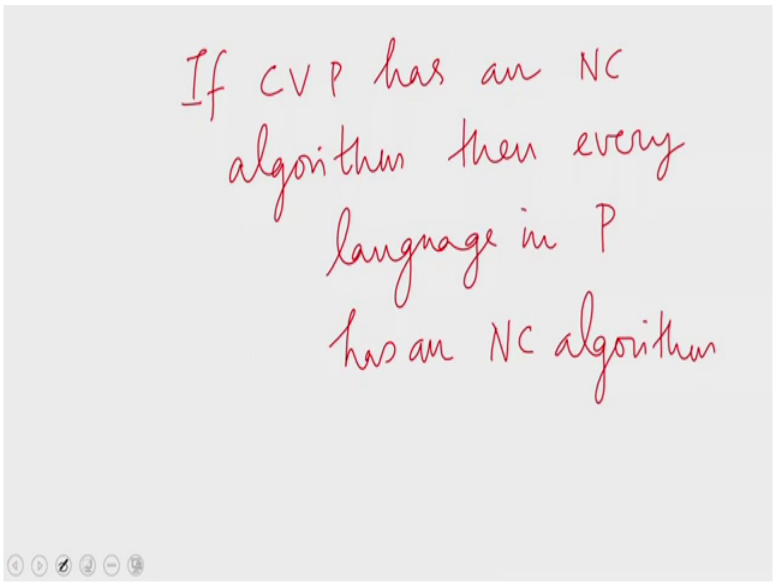
in $\underline{\underline{NC}}$ $\qquad \underbrace{C(1,1,T(n))}$

$\underline{\text{CVP is P-complete for NC-reduction}}$

The circuit can be written out visualizing the circuit and writing it out in this form, are essentially the same as you can figure out, in NC. That is in poly logarithmic time using a polynomial number of processors which means when we are given an input W which is a potential input for the language L, using the transition table of the turing machine for this language L we can write out a circuit so that the circuit will evaluate 1, that is there is 1 signal point in this circuit, which will evaluate 1 if and only if W belongs to L which is that signal point it is C 1,1 T of n. That is, when the machine comes to a halt the T of n step, the first cell should contain a 1, a 1 let us assume is symbol 1.

So, in that case W should belong to L and only in that case will W belong to L. So, evaluating this circuit can essentially check the membership of L. So, this establishes that C V P is P complete for NC reductions. Whereas, we have just now shown that any arbitrary language L in P is NC reducible to C V P. C V P is at least as hard as all these languages. In addition C V P also belongs to P. So, C V P is P complete for NC reductions. So, there is indeed a language which is P complete for NC reductions.

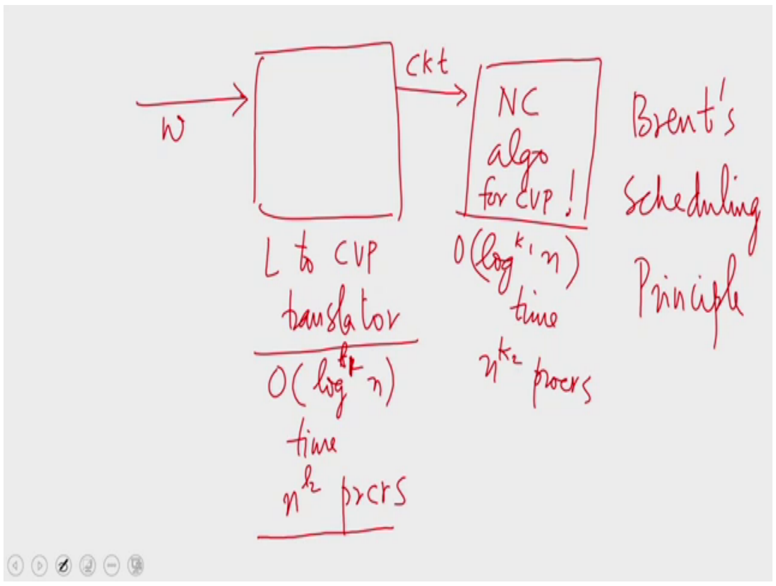
(Refer Slide Time: 31:26)



If CVP has an NC algorithm then every language in P has an NC algorithm

But then what it means is that if C V P has an NC algorithm, then every language in P has an NC algorithm. How do we show this?

(Refer Slide Time: 31:42)

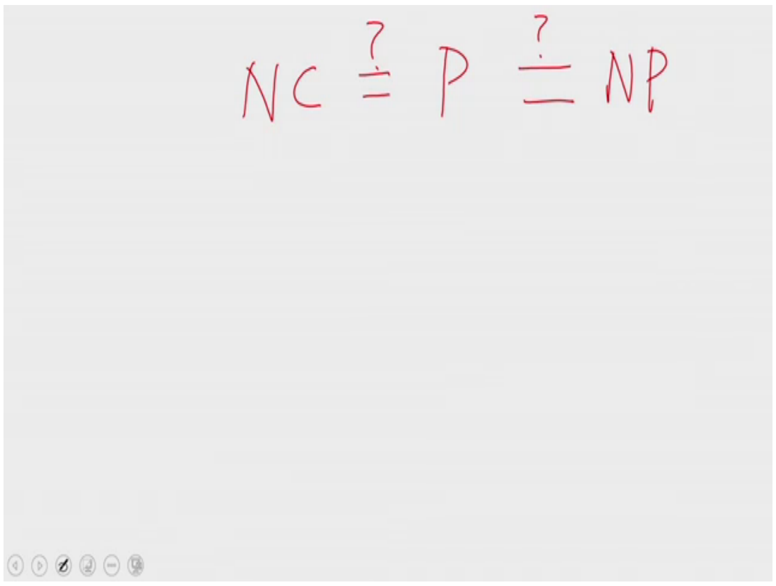


Suppose this is the hypothesized NC algo for C V P of course, no such algorithm is known. Suppose this runs in order of log n power k 1 time, using n power k 2 processors. And then let us consider an L to C V P translator which we have seen just now. This will take an input a potential input to L and will produce a circuit.

Suppose this translator, since this is an NC translator this should run in log n power k 2 time or log l power l, l 1 time using n power l 2 processors, then by the juxtaposition of these 2 algorithms. We will have an NC algorithm when we juxtapose them we have to use Brent's scheduling principle. Each of these individual components of the algorithm are poly logarithmic time algorithms and each has a polynomial cost. So, the total cost is polynomial as well and then the larger of these 2 time bounds can be used for finding the number of processes that should be used in Brent's scheduling principle.

Such an execution of this juxtaposition will be an NC execution. Therefore, we have established that if CVP has an NC algorithm, then every language in P has an NC algorithm. And this is assumed to be unlikely today, because even after several investigations we still have not been able to find a P algorithm for any of the find an NC algorithm for any of the P complete languages for NC reductions. And nor have we been able to show that NC is not equal to P.

(Refer Slide Time: 33:39)



$$NC \overset{?}{=} P \overset{?}{=} NP$$

So, this question remains an opened one, exactly the way same way as this question remained, we do not know whether P equal to NP. Similarly we do not know whether NC equal to P. In the next class we shall see, some reductions from some commonly encountered problems like; a depth first search etcetera to the circuit value problem and that will establish that some of the very common very easy problem the easy to solve sequentially problems are hard for parallelization.

For example, depth first search has a very easy sequential algorithm, but it does not have an efficient parallel algorithm as of now and it is rather unlikely to have an efficient parallel algorithm. Since, we do not know whether NC equal to P, if NC is not equal to P, then DFS will not have a poly logarithmic time polynomial processes solution. So, this we shall see in the next class. So, that is it from this lecture hope to see you in the next.

Thank you.