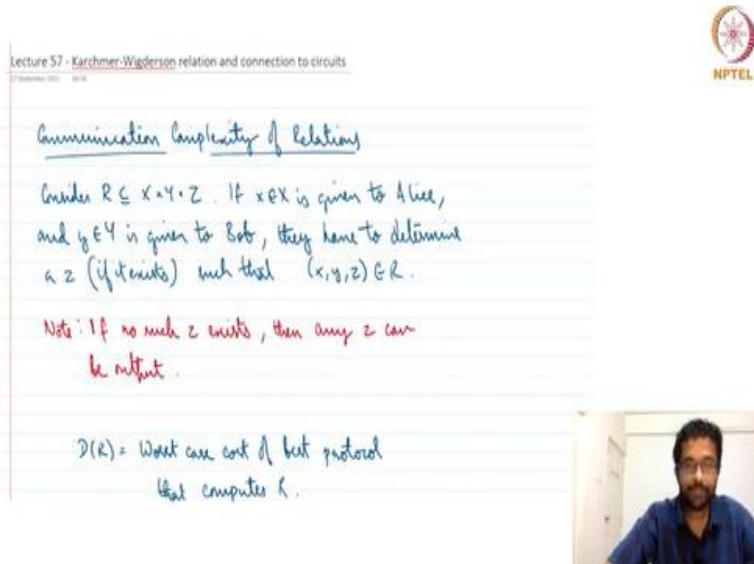



**Computational Complexity**  
**Prof. Subrahmanyam Kalyanasundaram**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Hyderabad**

**Lecture - 57**  
**Communication Complexity of Relations**

(Refer Slide Time: 00:15)



Lecture 57 - Karchmer-Wigderson relation and connection to circuits




Communication Complexity of Relations

Consider  $R \subseteq X \times Y \times Z$ . If  $x \in X$  is given to Alice, and  $y \in Y$  is given to Bob, they have to determine a  $z$  (if it exists) such that  $(x, y, z) \in R$ .

Note: If no such  $z$  exists, then any  $z$  can be output.

$D(R)$  = Worst case cost of best protocol that computes  $R$ .



Hello and welcome to lecture 57 of the course on computational complexity. So, in the past couple of lectures we saw the topic of communication complexity. In this lecture and the next we will see an application of communication complexity into a circuit complexity. So, we will see how communication complexity in a certain setting will imply lower bounds for circuit complexity. So, in order to get started let me define communication complexity of relations.

So, far we have seen communication complexity of functions where there is a function  $f$  that is to be jointly computed by Alice and Bob. But here we are going to talk about the relations, so suppose there is a relation  $R$  that is a subset of  $X \times Y \times Z$ . So, think of  $X$  and  $Y$  as the same  $X$  and  $Y$ , that  $X$  goes to Alice and  $Y$  goes to Bob and  $Z$  being the output. So, in the case of a function for every  $X$  and  $Y$ , there is a unique  $Z$ .

But in the case of a relation that could be multiple  $Z$  or no  $Z$  as well. So, the goal is  $X$  is given to Alice,  $Y$  is given to Bob and then they have to determine  $Z$  such that  $x, y, z$  is in the relation. So,

if this relation is a function, it is just like computing a function, just that now we have a slightly more generalized setting. There could be multiple  $z$  that satisfy the condition. And there could also be no such  $z$  in that case we do not care what the output is.

So, if it was a function then  $z$  would be unique for a specific  $x$  and  $y$ . But since it is not a function there could be multiple such  $z$ .

**(Refer Slide Time: 02:09)**

The slide contains handwritten notes on a whiteboard background. At the top right is the NPTEL logo. The main text defines a relation  $U \subseteq \{0,1\}^n \times \{0,1\}^n \times \{1,2,\dots,n\}$  and  $U = \{(x,y,i) \mid x_i \neq y_i\}$ . Below this, it notes  $D(U) \leq n + \log n$  and describes a protocol where Alice sends  $x$  and Bob sends  $z \in \{1,2,\dots,n\}$ . A truth table for  $n=2$  is shown with columns for  $x_1, x_2, y_1, y_2$  and rows for  $(x,y)$  pairs. The table entries are: (0,0) has  $z=1$ ; (0,1) has  $z=2$ ; (1,0) has  $z=2$ ; (1,1) has  $z=1$ . Below the table, it states a claim  $D(U) \geq n-1$  and a reduction to a game: given a protocol for  $U$ , Alice and Bob play a game.

And deterministic communication complexity of the relation is the worst case cost of the best protocol that computes this relation. So, that is denoted as usual by the symbol  $D R$  and one example is the so-called universal relation. So, universal relation is  $x$  is the set of all  $n$  bit vectors,  $y$  is a set of all  $n$  bit vectors and they have to output  $z$  which is like a number from 1 to  $n$  and the number that they have to output is if Alice and Bob are given two strings  $x$  and  $y$  such that  $x$  is not equal to  $y$ .

Then they have to output an index  $i$  for which the  $i$ th bit of  $x$  is not equal to the  $i$ th bit of  $y$ . So, they have to output  $i$  such that  $x_i$  is not equal to  $y_i$ , this is the objective. Of course, when Alice and Bob are given the same  $x$  is equal to  $y$  then we do not care what the output. So, just like we had a matrix for the functions where we computed functions, we could have a matrix for the relations.

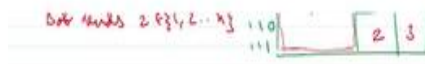
So, here we have a 8 by 8 matrix so the rows and columns are indexed with 3 bit numbers and possible responses for the universal relation. So, if you see the bottom left corner, the square that I am highlighting now is for all the 4 by 4 16 values here. The first bit of Alice is equal to 1 and the first bit of Bob is equal to 0, so since they differ in the first bit, they could answer could be 1. And similarly in the top right corner here for all the sixteen entries, Alice's first bit is 0 and Bob's first bit is 1.

So, since they differ in the first bit, they could output 1 here and if you look at this square for instance, the small 2 by 2 square Alice's second bit is 1 and Bob's second bit is 0. So, the output two as the answer because they differ in the second bit and so on; and you can verify the rest of the entries here. So, this is a picture of the matrix corresponding to  $R$ , the relation of the universe not  $R$ , the  $u$ .

The universal relation that they can possibly in output and I have actually it is already separated into monochromatic rectangles. So, this is just to see how a protocol, so from this you can possibly see how they may execute a protocol and one observation is that for the universal relation there is always an  $n + \log n$  bit protocol. So, Alice can send the entire  $x$ , so Alice send  $x$  and Bob sends  $z$  from 1 to  $n$ , so  $x$  requires  $n$  bits and  $z$  requires  $\log n$  bits.

So, that is why this is  $n$  plus  $\log n$  protocol and this is a very straight forward protocol. This will work for any relation not just this relation as long as, so instead of  $\log n$  it should be  $\log$  of size of  $z$ .

**(Refer Slide Time: 05:55)**



Claim:  $D(U) \geq n-1$   
 → Complement of EQ  
 Reduction to NE: Given protocol for U, Alice & Bob can compute i using  $D(U)$  bits. Then Alice & Bob send  $x_i$  &  $y_i$  to each other.

$$n+1 \leq D(NE) \leq D(U)+2$$

$$\Rightarrow D(U) \geq n-1$$

Recall: Circuits: Depth of a circuit C is the



But another thing to note is that there is a lower bound for this universal relation of  $n - 1$ . Why is this? This is because you can compute the function not equal to, so this is a complement of equality. It is just the opposite of equality, so given a protocol for universal function we will show how to compute equal and not equal to, so not equal to and equal to should have the same complexity because it is a slipping of the answer.

So, if they know that the function is not equal to then they know that it is equal to also. Perhaps it is easier if I just say if I do not say not equal to I just say equality. Now maybe not equal to is necessary, complement of equality. So, we will see how we can determine not equal to from the universal relation. So, suppose we have a protocol for universal relation that means if  $x$  and  $y$  differ.

Then Alice and Bob can identify using the protocol for universal relation which bit they differ on that is the universal relation. So, given  $x$  and  $y$  using  $D$  bits, if the assumption is that there is a protocol for universal relation, so using  $D$  bits Alice and Bob can execute a protocol and identify  $i$  the bit where they differ, the index where they differ. And now that they know the index where they differ or they potentially differ, they can send  $x_i$ .

Alice can send  $x_i$  to Bob and Bob can send  $y_i$  to Alice to each other. So, now that they know each others  $i$ th index. Alice knows  $y_i$ , Bob knows  $x_i$  and then they can compare whether their

Alice can check whether  $x_i = y_i$ , Bob can also check whether  $x_i = y_i$  and then that is enough. Both of them know that  $x_i$  is equal to  $y_i$  or  $x_i$  is not equal to  $y_i$ , if  $x_i$  is not equal to  $y_i$  then  $x$  and  $y$  are not equal, if  $x_i$  is equal to  $y_i$ .

Then well they are equal because the whole thing worked on the assumption that they were not equal and that is why we are getting the error. So, what is the complexity of this protocol? The complexity of this protocol is the complexity of the protocol for executing universal rotation or computing universal relation  $D_u$  and after which this end  $x_i$ , Alice sends  $x_i$ , Bob sends  $y_i$ . So, two more bits after  $D_u$ .

So, this is the complexity of this protocol and this is one protocol for not equal to. So, deterministic complexity of not equal to is at most  $D_u + 2$ . But at the same time using lower bounds for equality, we can also say that the deterministic complexity for not equal to is at least  $n + 1$ . So, we know that deterministic complexity is not equal to at least  $n + 1$  and at most  $D_u + 2$  and this gives a lower bound for  $D_u$  as  $n - 1$ .

So, using the universal addition you can compute not equal to and so that gives you this lower bound of  $n - 1$ , where the upper bound, the trivial upper bound is  $n + \log n$ . In fact, this has been improved later, I think  $n + 1$  by Tardo and Swick, but anyway I am not getting into that.

**(Refer Slide Time: 09:55)**


$\therefore D_u \geq n - 1$


Recall: Circuits - Depth of a circuit  $C$  is the length of the longest <sup>path</sup> from the o/p (root) to an input (leaf). Denoted  $d(C)$ .


Depth of a function  $f$ ,  $d(f) = \min_{C: \text{Computes } f} d(C)$ .

Karchmer-Wigderson relation: Given a Boolean

$f: \{0, 1\}^n \rightarrow \{0, 1\}$   $n \geq 1$   $0 \leq v \leq n-1$







So, this is just an example of what is a communication complexity of relations. So, in this lecture we will see how communication complexity of relations, how using that we can get a circuit lower bound. So, lower bound for the depth of circuits not the size, so far, I think we have seen size based lower bounds but this will be a depth lower bound. So, you may recall that a circuit is a tree, so in this lecture we will assume a binary tree.

And the depth of a circuit is a length of the longest path from the output which is the root of the circuit to an input. So, if you have some something like this, this is the depth of the circuit, suppose this is the longest path, this is the depth of the circuit and this is denoted  $d$  of  $C$  the depth of the circuit. And the depth of a function  $d$  of  $f$  is this depth of the smallest circuit that computes this function. So,  $d$  of  $C$  for the smallest which for the  $C$  that minimizes this.

**(Refer Slide Time: 11:22)**

Karchmer-Wigderson relation: Given a Boolean function  $f: \{0,1\}^n \rightarrow \{0,1\}$ , let  $X = f^{-1}(1)$  and  $Y = f^{-1}(0)$ . Let  $R_f \subseteq X \times Y \times \{1,2,\dots,n\}$  be the set of all  $(x,y,i)$  such that  $x_i \neq y_i$ . So given  $x$  to Alice,  $y$  to Bob, they need to determine  $i$  such that  $x_i \neq y_i$ .

Check:  
 $x, y \in \{0,1\}^n$   
 $X \cap Y = \emptyset$   
 $X \cup Y = \{0,1\}^n$

$f(x) = 1$   
 $f(y) = 0$

100 000 001 010  
 101 3  
 $f(x,y) = 3$



So, depth of a function is the depth of the smallest circuit that computes that function, so we have seen circuits we know what this is. So, now let us define the crucial relation that will help connect communication complexity of relations and depth of circuits. So, what we will do is given a function, a Boolean function; normal Boolean function which is on  $n$  bits. So, this function on  $n$  bits we will define what is called a Karchmer Wigderson relation.

So, this is because it was first seen in a paper by Karchmer and Wigderson. So, what is the relation? The relation is this, so you separate the domain of  $f$  into 0 inputs and 1 inputs, so let  $x$

be all the one inputs meaning all the inputs, all the 0, 1, so 0, 1 power n inputs which give the answer 1 and y be the all the 0 inputs. So, both just for sanity check I just say that, both x and y are in fact subset of 0 1 power n.

In fact, they are not just subset, they are disjoint because for a specific string either it is f of x is 0 or f of x is y and every small x is either in capital X or capital Y. So, it is a partition, so in fact this is true X intersection Y is the empty set and X union Y is 0, 1 power n, all this can be seen. So, you can check all this, this is just to make sure that you understood this correctly. Now we define the Karchmer Wigderson relation also called R f.

So, it is like the universal relation but slightly different. So, the universal relation we allowed it is 0, 1 power n cross 0, 1 power n cross 1, 2, 3 up to n, but here the Alice input or the X is just the one input Y is just the 0 input. So, suppose Alice got a string small x which is a one input, so Alice got x such that f of x = 1 and Bob got y such that f of y = 0. Obviously, x is not equal to y because x is 1 input, y is a 0 input.

And then they have to output an index where x i is not equal to y i. So, just again to add f of x = 1, f of y = 0, this is what we can infer from this and they need to together determine i is such that x i is not equal to y i.

**(Refer Slide Time: 14:31)**

Handwritten notes on a slide illustrating a function  $f(z)$  and its decision tree. The function is defined as  $f(z) = [z_1 \wedge (z_2 \vee z_3)] \vee (z_2 \wedge z_3)$ . The decision tree starts with a root node  $V$ , which branches into two nodes  $A$ . Node  $A$  on the left branches into  $z_1$  and  $V$ . Node  $V$  branches into  $z_2$  and  $z_3$ . Node  $A$  on the right branches into  $z_2$  and  $z_3$ .

A truth table is shown with columns for inputs  $z_1, z_2, z_3$  and output  $f(z)$ . The rows are labeled with binary strings: 101, 111, 110, 011, 100, 000, 001, 010. The output is 1 for the first three rows and 0 for the last four rows.

Inputs are listed as:
 

- 1-inputs: 011, 111, 110, 101
- 0-inputs: 100, 000, 001, 010



So, just an example, so perhaps just to see this better, so let us say  $f$  of  $z$  is equal to this,  $f$  of  $z$  maybe I will just say I just say  $z$  is equal to this. So,  $z_1$  and  $z_2$  or  $z_3$  are the whole  $R$  of  $z_2$  or  $z_3$ , maybe I will just rearrange these terms so that it is in one line, maybe this is okay. So, here it is also drawn in a circuit fashion  $z_1, z_2, z_3$ , so  $z_1, z_2$  and  $z_3$  appear twice here but that is okay. Since we are computing the same function.

There is an  $R$  and the left-hand side of the or the left sub tree computes  $z_1$  and  $z_2$  or  $z_3$  and the right subtree computes  $z_2$  and  $z_3$ . So, now let us see which inputs give out what, so which are the one inputs here. So, for a one input either  $z_2$  and  $z_3$  should be 1, the second part should be one which means anything that ends in 1 1, so 0 1 1 and 1 1 1 are one inputs and which are one inputs.

So, one inputs are 0 1 1, 1 1 1 and if  $z_2$  and  $z_3$  are not both one, this is AND this is an OR here. So, even the first term could be 1, so the first term is  $z_1$  and  $z_2$  or  $z_3$ , so either  $z_1$  has to be 1 and either  $z_2$  or  $z_3$  has to be 1. So, you could have 1 1 0 or 1 0 1 or 1 1 1 which is already written, so these are the four one inputs, so you can verify that these are the four one inputs. And I am writing them as the columns of this matrix over here.

So, if you just see I have written them down as the columns of this matrix. The rows are indexed by these four values which are the 1 input and everything else is 0 input and you can indeed verify that the rest are 0 inputs. Verify meaning you can just plug them in the circuit and see that they are all the output is 0 and the columns are indexed by the 0 inputs in the above matrix. So, now notice that it is not a  $2^{\text{power } n}$  by  $2^{\text{power } n}$  matrix.

In this case it is a 4 by 4 matrix but it depends on how the 0 inputs and one inputs are distributed. So,  $2^{\text{power } n}$  here is 8 but, in this case, it is a 4 by 4 but it could also have been a 5 by 3, if 5 cross 3 matrix if there were 5 one inputs and 3 0 inputs. And again, we have the same thing as universal relation, we have to determine an index where  $x$  and  $y$  differ and this is one way to partition it. So, notice that for some entries there are multiple correct answers.



So, for instance; especially if you look at this entry here, anything would be ok here. Because the column is indexed by 0 0 0 and the row is indexed by 1 1 1. So, 1 or 2 or 3 anything would do, but in this case, it turned out to be more convenient to have 1 because the rectangle partitioning is better in this case. So, this is the matrix corresponding to the Karchmer Wigderson relation corresponding to this function.

So, Karchmer Wigderson relation is obtained from a function by classifying it into 0 inputs and 1 input. And the goal is to together collectively identify an index  $i$  such that  $x_i$  is not equal to  $y_i$  and in this case there will be no do not case because all  $x$ 's are different from  $y$ 's. So, there will not be a situation where  $x$  and  $y$  are the same.

**(Refer Slide Time: 19:30)**

Theorem: For any  $f$ ,  $D(R_f) = \text{depth}(f)$ .

Proof: (i)  $D(R_f) \leq \text{depth}(f)$ .

We will construct a protocol from a circuit.

Let  $C$  be a circuit that computes  $f$ . Alice/Bob use the following protocol. They traverse the nodes top down from the root. At every step, they ensure that they are at a node that outputs a function  $g$ , i.e.  $g(x)=1$  and  $g(y)=0$ .

Base Case: At the root,  $f(x)=1$ ,  $f(y)=0$ .

And the theorem the main theorem by Karchmer and Wigderson is that, whatever function we take, the deterministic communication complexity of the Karchmer Wigderson relation which I denote by  $D$  of  $R_f$ , this is equal to the depth of the function. Meaning the; depth of a circuit that can compute the function. The smallest depth circuit will have depth equal to the determinacy communication complexity of  $R_f$ .

So, this seems very, very interesting and somewhat surprising because on one hand we have this relation and then there is communication between Alice and Bob. On the other hand, we have the circuit computation model and then just computing of a certain function and one would think

how on earth would the; communication complexity bound translate into a circuit complexity bound. But in fact, it does.

So, let us see the proof why this is the case? So, we prove it in two parts; first we show that the depth of the deterministic communication complexity is at most the depth, this is part one. Deterministic complexity of the relation is at most the depth of the function. And the second part we show that the depth of the function is at most deterministic communication complexity of the relation.

So, we prove two inequalities and that is how we get the equality about, so how do we get the first inequality? So, what we do is to given a circuit for the function. So, we start from a circuit, from that we will build a protocol for the relation. And as long as we started with the circuit for the function  $f$ , we will ensure that we get a protocol for the relation  $R f$ . And we will get the protocol in such a way that the complexity of the protocol is equal to the depth of the circuit.

So, I could have chosen any circuit that computes  $f$ , so I could choose the circuit of the smallest depth that computes  $f$ . So, I could choose a circuit of depth  $D f$  which means I get a protocol of complexity  $D f$ . So, this is an upper bound, this is one protocol of complexity  $D f$ , hence the deterministic communication complexity is upper bound by the complexity of this protocol which is  $D f$ .

So, I hope the high-level picture is clear, now all that we need to see is given a circuit that computes  $f$ , how do we get a protocol for the communication complexity of  $R f$ ?

**(Refer Slide Time: 22:32)**

Theorem: For any  $f$ ,  $D(R_f) = \text{depth}(f)$ .

Proof: (i)  $D(R_f) \leq \text{depth}(f)$ .

We will construct a protocol from a circuit.

Let  $C$  be a circuit that computes  $f$ . Alice/Bob use the following protocol. They traverse the nodes top down from the root. At every step, they ensure that they are at a node that computes

a function  $g$ , s.t.  $g(x)=1$  and  $g(y)=0$ .

Base case: At the root,  $f(x)=1$ ,  $f(y)=0$ .



So, what we do, so maybe I will explain here, what we do is I have just drawn the same circuit copy pasted from above. What we do here is, from the circuit we will see how to modify it into the protocol. So, the circuit will look something like this and we will transform the circuit into a protocol tree that is the idea. So, let us traverse from the top down and so at the top this entire circuit computes the top gate computes the function  $f$ , this computes the function  $f$ .

So, what do we have here? We know that  $f$  of  $x$ , that  $x$  is Alice's input is 1 and  $f$  of  $y$  is 0. And we will make sure that at every stage we have every node, so this particular node, let us say it computes  $g$ , this particular node the one left side of the root. We will ensure that  $g$  computes a function such that  $g$  of  $x$  is 1 and  $g$  of  $y$  is 0 and so does every node in the circuit. So, we will make sure that at every node in the circuit satisfies or computes a function such that  $g$  of  $x = 1$  and  $g$  of  $y$  is 0.

So, let us see how this happens, so at the base case is the root, at the root we already know that because of the problem statement  $f$  of  $x = 1$  and  $f$  of  $y = 0$  where  $x$  is Alice's input  $y$  is Bob's input.

**(Refer Slide Time: 24:18)**



Base Case: At the root,  $f(x)=1, f(y)=0$ .

— If the current node is an OR gate, then Alice speaks. Let the gate compute  $f \vee f_1$ . We know  $f_0(x) \vee f_1(x)=1$ . So either  $f_0(x)=1$  or  $f_1(x)=1$ . But  $f_0(y) \vee f_1(y)=0$ , and hence  $f_0(y)=f_1(y)=0$ .

So for either  $f_0$  or  $f_1$ , we have  $f_i$  s.t.  $f_i(x)=1, f_i(y)=0$ . Alice sends this  $i$ .

— If the current gate is an AND gate, then similarly there is an  $f$  such that



So, now there are two cases; so, we will go down and by induction we will do it, so we are at a certain node and we want to say something about the children nodes. So, there are two children, it is a protocol of a circuit of fans in 2, so we want to say something about the children. We want to say that the children gate also satisfies this particular statement, this is the inductive claim. At every step there we ensure that we are at a node that computes a function  $g$  such that  $g$  of  $x = 1$  and  $g$  of  $y = 0$ .

Suppose the current node is an OR gate, so in this case the root node is an OR gate. So, we will transform it into something where Alice speaks, so we will say Alice speaks here OR gate is where Alice speaks and gate will become places where Bob speaks. So, I am not getting there but I am just saying, now what would Alice reply? Now suppose  $f$ , this is an OR gate so let us say the corresponding functions of the two children are  $f_0$  and  $f_1$  suppose this is  $f_0$  and this is  $f_1$ .

And we know that  $f$  of  $x = 1$  and  $f$  of  $y = 0$ . So,  $f$  of  $x = 1$  so I could write that as, so  $f$  is  $f_0$ , so what does this mean? This means that  $f$  is  $f_0$  or  $f_1$ , that is why the top is  $f$ , the children are  $f_0$  and  $f_1$ . So,  $f_0(x)$  or  $f_1(x) = 1$ , so the OR of  $f_0(x)$  and  $f_1(x) = 1$ . This means that either  $f_0(x) = 1$  or  $f_1(x) = 1$  and the next thing is that  $f_0(y)$  and  $f_1(y)$ , the OR of these two things is equal to 0, because  $f$  of  $y$  is 0. Now if OR of 2 things are 0, this means that both of them have to be 0.

So,  $f_0 y = 0$ ,  $f_1 y = 0$  and either  $f_0 x = 1$  or  $f_1 x = 1$  both also could be 1,  $f_0 x$  and  $f_1 x$  both of them could be 1. But we know for sure that at least one of them is a one, so either for  $f_0$  we have that input is held by Alice, either for  $f_0 x$  is equal to so whatever input is held by Alice, let us say that is  $x$  for that  $x$  either  $f_0 x = 1$  or  $f_1 x = 1$ . So, now Alice will say 0 if  $f_0 x$  is 1 and Alice will say one if  $f_1 x$  is 1. If both of them are 1 she could choose any one let us say she says 0.

So, there is an  $i$ , so  $i$  means the subscript of  $f$  such that  $f_i x = 1$  and whichever  $i$  she chooses  $f_i y$  is 0 because  $f_0 y$  and  $f_1 y$  are both 0. So, whatever is that  $i$  Alice sends that  $i$ , so she sends that  $i$ .

**(Refer Slide Time: 28:14)**

So for either  $f_0$  or  $f_1$ , we have  $f_i$  s.t.  
 $f_i(x)=1, f_i(y)=0$ . Alice sends this  $i$ .

— If the current gate is an AND gate, then similarly there is an  $i$  such that  $f_i(x)=1, f_i(y)=0$ . Bob sends that  $i$ .

— When they are at a leaf, the function is  $z_i$  or  $\bar{z}_i$  for some  $i \in \{1, 2, \dots, n\}$ .  
 If it is  $z_i$ , then  $x_i=1$  and  $y_i=0$ .  
 If it is  $\bar{z}_i$ , then  $x_i=0$  and  $y_i=1$ .

$f = f_0 \wedge f_1$   
 $f_0(x) \wedge f_1(x) = 1$   
 $f_0(x) = f_1(x) = 1$   
 $f_0(y) \wedge f_1(y) = 0$   
 $\rightarrow$  one of  $f_0(y)$  or  $f_1(y) = 0$ .

Similarly, if the current gate is an AND gate, Bob peaks, why is that? Just like what we said before we know for sure that if it is an AND gate, then it will be like this. Let us say  $f$  is computed at that gate where  $f$  is  $f_0$  and  $f_1$  and we know that  $f_0 x$  and  $f_1 x = 1$ . So, which means  $f_0 x$  and  $f_1 x$  are both equal to 1 and since the AND of two quantities is 0,  $f_0 y$  and  $f_1 y = 0$ .

This means that one of them  $f_0 y$  or  $f_1 y$  is 0, at least one of them must be 0 otherwise the AND should be 1. So, now Bob knows this because  $x$  is one for both, so Bob will tell which is the  $y$  for which  $f_0 y$  is 0 or  $f_1 y$  is 0. So, which is the  $i$  for which  $f_i y$  is 0, bob sends that  $i$ . So, this is how we transform the circuit into a protocol so all the AND, what we said that Alice will speak and we said what Alice will send for at this node.

So, initially they held  $x$  and  $y$  and at the Alice held  $x$ , Bob held  $y$  such that  $f$  of  $x = 1$  and  $f$  of  $y = 1$  where  $f$  is the function computed at the root node. Then they will move to one of the nodes which satisfies, let us say they move to the left child. So, we know that for the input that they hold  $f$  of  $x = 1$  and  $f$  of  $y = 0$  and then they will keep going down and down and down till they reach a leaf.

So, now the question is here we have a function that is computed by, we have a circuit that computes a function. Now in a protocol they have to output something at the leaf. So, when they reach the end of the leaf that leaves should correspond to something even though they do not really output something? So, what is the entry that the leaf corresponds to?

**(Refer Slide Time: 31:00)**

Handwritten notes on a slide:

- $f(x)=1, f(y)=0$ . Bob sends that 1.
- When they are at a leaf, the function is  $z_i$  or  $\bar{z}_i$  for some  $i \in \{1, 2, \dots, n\}$ .
- If it is  $z_i$ , then  $x_i=1$  and  $y_i=0$ .
- If it is  $\bar{z}_i$ , then  $x_i=0$  and  $y_i=1$ .
- In any case, the bit  $i$  is an answer.
- No. of bits exchanged = depth.
- $D(R_f) \leq d(f)$ .

Additional notes on the right side of the slide:

- $f_0(y) \wedge f_1(y) = 0$
- $\rightarrow$  no  $f_0(y)$  or  $f_1(y) = 0$ .

NPTEL logo is visible in the top right corner of the slide.

So, if they write a leaf, the leaf will be labelled with either something called  $z_i$  or  $\bar{z}_i$  complement where  $i$  is its index, where so basically the input string is  $z$  either the leaf will involve  $z_i$  or  $\bar{z}_i$  complement. So, here it is  $z_1, z_2$  or  $z_3$  there is nothing negated here but if there was a negation it could be there in the circuit here. So, one more point is that what we can do is, if there is a NOT, NOT gate anywhere in the circuit.

We could push that not gate down using De Morgan's laws and we will get a circuit with purely AND and OR gates. Such that the not gates will be only appearing in the input level. So, now if there is a NOT gate here, if I want to have a NOT gate here, I may as well change it to something

like this, I may actually push it down to the input bit itself, so  $z_i$  complement. So, I can do something like this.

So, there will be no NOT gates in the circuit, there will only be inputs that are negated or non-negated. So, when we read a leaf, the leaf either says that  $z_i$  or  $z_i$  complement. Now we need to think in terms of the protocol, what will be output at the protocol level? So, we need to say some  $i$  the index  $i$  for which  $x_i$  and  $y_i$  differ. So, what we know is that because of the induction so far if the leaf is labelled  $z_i$ .

Then that means that the input is such that so the function computed at the leaf is just  $z_i$  which is the symbol of the  $i$ th index. Then we know that  $x_i = 1$  and  $y_i = 0$  if the symbol is  $z_i$  at the leaf. If the symbol at the leaf is at a complement, then we know that  $x_i$  is 0 and  $y_i$  is 1 because that is why  $x_i = 0$  for any string with  $x_i = 0$ ,  $z_i$  complement is 1 and for any string with  $y_i = 1$  the  $y_i$  complement is 0.

So, in either of these cases we will output the bit  $i$ , so we check whether what the leaf of the circuit indicates. So, if it is either  $z_i$  or  $z_i$  complement we will set the leaf of the communication complexity protocol to output  $i$ , the index  $i$ . And that completes it. So, the proof was kind of descriptive, the correctness was kind of included in the procedure, starting from a circuit for the function  $f$  we converted the circuit into a protocol such that this protocol is for the Karchmer Wigderson relation of the function  $f$ .

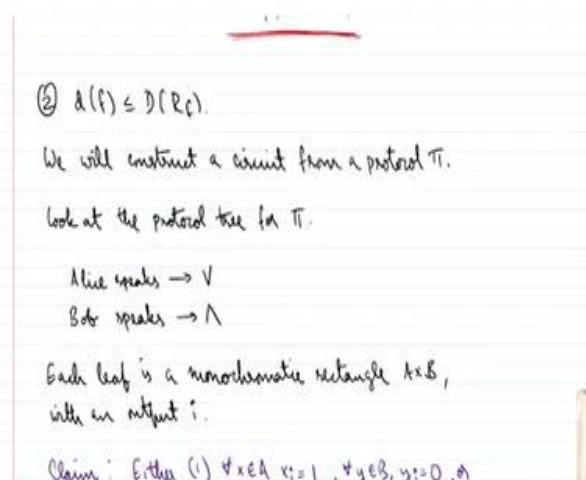
So, this means that the deterministic communication capacity of the relation is at most the depth of the function, so again how many bits are exchanged in this protocol? It is exactly the depth of this tree, the depth of the circuit but if it is this, then you can check that they will exchange at most 3 bits and perhaps you can work out some small examples like this and see whether it checks out correctly.

So, the number of bits exchanged is the depth of the circuit, so the deterministic communication complexity of  $R_f$  is at most the depth of this circuit but the depth of this circuit is, but I could choose a circuit of the smallest depth. So, deterministic communication complexity of  $R_f$  is at

most depth of the function, so that is one direction of the proof where we got a protocol from a circuit.

The other direction is the opposite we are given a protocol for the Karchmer Wigderson relation, we will construct a circuit for the function. So, in fact what we do here is kind of like the reverse of what we did in the first part, so suppose we are given a protocol we will view it as a tree. So, in the first part we replaced OR gates were designated as Alice speaks, Alice to speak and AND gates were designated as places where Bob has to speak. We will do exactly the opposite here.

**(Refer Slide Time: 36:25)**



②  $d(f) \leq D(R_f)$ .

We will construct a circuit from a protocol  $\pi$ .

look at the protocol tree for  $\pi$ .

Alice speaks  $\rightarrow V$   
Bob speaks  $\rightarrow \wedge$

Each leaf is a monochromatic rectangle  $A \times B$ ,  
with an output  $i$ .

Claim: Either (1)  $\forall x \in A, y = 1$  .  $\forall y \in B, y = 0$  .  $\dots$



So, suppose there is a protocol given, we need to convert the protocol into a circuit. So, we have a tree and we have places where Alice is designated to speak, Bob is designated to speak and at the end we have leaves at the bottom where we know the leaves correspond to some output, so that the output we know is something from 1 to n. So, we need to know how to replace this into a circuit, so all the places where Alice is designated to speak, we will replace it with an OR gate. All the places where Bob is designated to speak or replaced with an AND gate.

**(Refer Slide Time: 37:07)**





with an output  $i$ .

Claim: Either (1)  $\forall x \in A, x_i = 1, \forall y \in B, y_i = 0$ , or  
 (2)  $\forall x \in A, x_i = 0, \forall y \in B, y_i = 1$ .

If it is case (1), then set the leaf label as  $z_i$ .  
 If case (2), label leaf  $\bar{z}_i$ . We will show that the circuit outputs  $f$ .

→ We will show that for all nodes of the circuit, the function  $g$  at the node



Now all that remains is what do we do at the bottom all the intermediate nodes are taken care of but the leaf nodes what do they correspond to. In the protocol it is some output  $i$  of the function of the relation, in the circuit it should be some input to the circuit, maybe some  $z_i$  or  $\bar{z}_i$  complement. So, now how do we get that, what should be the labels of the leaves? So, we know that every leaf is a monochromatic rectangle.

This is what we saw earlier when we learned communication complexity. Every leaf is a monochromatic rectangle  $A$  cross  $B$  with which gives, monochromatic means it gives a certain output  $i$ . And we know that if you look at a monochromatic rectangle, so which means all of the leaf outputs  $i$ , so all of this outputs  $i$  something like this. So, this means that the inputs differ in the  $i$ th bit, so this means that the inputs here differ in the  $i$ th bit.

So, the first claim is that it has to be the case that for the entire matrix rectangle  $A$  cross  $B$ , so this is  $A$  cross  $B$ , all of them output  $i$ . So, they differ at the  $i$ th bit but further one more thing is true, the one more thing that is true is that for this entire matrix either  $x_i = 1$  for all this matrix and  $y_i = 0$  for all this matrix or the opposite, it cannot be that some entries have  $x_i = 1, y_i = 0$ , some entries have  $x_i = 0$  and  $y_i = 1$ .

Because if you let us say if this entry has  $x_i = 1, y_i = 0$  and this blue encircled entry has  $x_i = 0, y_i = 1$ . Now consider this purple entry, this will have  $x_i = 1$  and  $y_i = 1$  which means it cannot

output  $i$ . So, what this means is that for the entire matrix either  $x_i = 1$  and  $y = 0$  or  $x_i = 0$  and  $y_i = 1$ , it cannot be a mix match of both. So, entire matrix should be corresponding to this, so if it is the case 1 for the entire matrix  $x = 1$   $y = 0$ , this means that.

So, now this function this leaf can be labelled as  $z_i$  and if it is the other case, the second case 2 then we label the leaf  $z_i$  complement.

**(Refer Slide Time: 40:12)**

circuit computes  $f$ .

→ We will show that for all nodes of the circuit, the function  $g$  at the node satisfies  $g(z)=1, \forall z \in A, g(z)=0, \forall z \in B$ , where  $A \times B$  is the set of inputs that reach the node.

→ At the root, we have the rectangle  $X \times Y$ , where  $X = f^{-1}(1)$  and  $Y = f^{-1}(0)$ . Hence root must compute  $f \Rightarrow$  circuit computes  $f$ .

We do bottom up induction.

So, we have described how to construct the circuit, we have replaced Alice speaking places with OR gate, Bob speaking places with AND gate and we have also explained how to label the leaves. Now all that we need to show all that remains is that this circuit is a valid circuit for the function. So, the main claim is that for all the nodes of the circuit, we will view it as both a circuit as well as the protocol to prove this.

Consider the tree something like this, let us say we consider this node, now let us say the function computed at this node is  $g$  and if you view it as a communication complexity protocol. Let us say the inputs from some matrix  $A$  cross  $B$  reach this node. The claim is that for whatever inputs contained in this  $A$ , the claim is that for whatever inputs  $z$  contained in  $A$ ,  $g$  of  $z = 1$  and for whatever input  $z$  contained in  $B$ ,  $g$  of  $z = 0$ , this is a claim.

So, look at any node in the tree, look at the corresponding set of entries that will reach in that node, in the protocol, in the communication complexity protocol. Suppose it is  $A$  cross  $B$  then for any  $x$  in  $A$ ,  $g$  of  $x = 1$  and for any  $y$  and  $B$ ,  $g$  of  $y = 0$ . And what would this imply? This would imply that whatever function this entire circuit computes, so we have again as of now we do not know what it computes, the claim is that it computes  $f$  we have to establish that it computes  $f$ .

So, at the root, what is the matrix that reaches the root? It is the entire  $X$  cross  $Y$ . So, the matrix that reaches the root is  $X$  cross  $Y$ , whatever function is computed at the root, we know that  $f$  of  $x = 1$  for all the  $x$ 's in  $X$  and  $f$  of  $y = 0$  for all the  $y$ 's in  $Y$  which is exactly the function that we were trying to compute.

So, the root must compute  $f$ , if this red underlined claim is true, so the claim to be proved is I will put this in the green box, this is the claim to be shown. We have built a circuit or we have transformed the protocol into a circuit and now we have to show that this circuit works and I have just described that it is enough to prove this claim.

**(Refer Slide Time: 43:25)**

We do bottom up induction.

- Base case: leaves (input). Values not according to the requirement.
- Consider node  $v$  and children  $v_0$  and  $v_1$ . Let  $f_0$  and  $f_1$  be the functions computed at  $v_0$  and  $v_1$ , respectively. Let  $f'$  be computed at  $v$ .
- WLOG let Alice speak at  $v$ . So  $f' = f_0 \vee f_1$ . Let  $A \times B$  be the rectangle at  $v$ . Alice's bit divides  $A$  into  $A_0$  and  $A_1$ . Inputs from

Diagram: A node  $v$  is shown with children  $v_0$  and  $v_1$ . Above  $v_0$  is a rectangle labeled  $A \times B$  with  $A_0$  above and  $A_1$  below. Below  $v_0$  is  $f_0$  and below  $v_1$  is  $f_1$ . Below  $v$  is  $f' = f_0 \vee f_1$ . Below  $f_0$  is  $y \in B$ . Below  $f_1$  is  $f_0(y) - f_1(y) = 0$ .



We will use a bottom up induction from the leaf to the root, so let us see what happens at the leaf. So, look at here, so this leaf is probably labelled  $z_i$  or  $z$  complement, suppose it is labelled  $z_i$  then we know that by this claim above that whatever functions reaches here it was labelled  $z_i$ .

Because for whatever inputs reached here  $x_i = 1$  and  $y_i = 0$  and if it is  $z_i$  complement whatever inputs reach here  $x_i = 0$  and  $y_i = 1$ .

So, this claim is already true for the leaves, when I say the claim, I mean the claim in the green box is true for the leaves by the way the labels were chosen. So, it is true at the leaves, we set the values correctly as per the requirement. Now consider any node, any intermediate node, let us say  $v$  has children  $v_0$  and  $v_1$  and let us say  $f_0$  and  $f_1$  were computed here. And let  $f$  prime be computed at  $v$ .

Now in the protocol let us say it was an Alice was speaking here, now this means  $f$  prime is nothing but  $f_0$  or  $f_1$  because when Alice was speaking, we replaced it with the OR gate. So, now let  $A$  cross  $B$  be the rectangle at  $v$ , so that let us say the rectangle at  $v$  is  $A$  cross  $B$  and now Alice is speaking here which means some inputs go to the left side and some inputs go to the right side.

So, now what happens is that when some inputs go to the left and some inputs go to the right. What is really happening is that the rectangle is row wise divided into two parts  $A_0$  and  $A_1$ . So, the set  $A$  is divided into  $A_0$  and  $A_1$  and basically this rectangle gets divided horizontally into two parts  $A_0$  cross  $B$  and  $A_1$  cross  $B$ . In the picture I have shown them as contiguous but it may not be contiguous. They may be just combinatorial rectangles; they could be in several parts.

So, inputs from  $A_0$  cross  $B$  reach  $v_0$  and  $A_1$  cross  $B$  reach  $v_1$ . So, we know that for all, we are doing bottom up induction, so we know that the inductive claim is true for  $v_0$  and  $v_1$ . What does it mean? Any input that reaches, that is in  $B$ ,  $f_0$  of any input that is in  $B$ , so let us say  $y$  is in  $B$ , then  $f_0$  of  $y$  and  $f_1$  of  $y$  are both 0, this is the inductive claim. And hence even for  $f$  prime,  $f$  prime is nothing but  $f_0$  or  $f_1$ , even for this  $f$  prime of  $y$  is the OR of 2 0s is 0 so for any  $y$  and  $B$ , this is true and what more do we know?

**(Refer Slide Time: 47:25)**



Let  $A \times B$  be the rectangle at  $v$ . Alice's bit  $f(y) = f_0 \vee f_1 = 0$   
 divides  $A$  into  $A_0$  and  $A_1$ . Inputs from  $x \in A_0, f_0(x) = 1$   
 $A_0 \times B$  reach  $v_0$  and those from  $A_1, v_1$   
 reach  $v_1$ . We know  
 $\forall y \in B, f(y) = f_0 \vee f_1 = 0$   
 $\forall x \in A_0, f_0(x) = 1$   
 $\forall x \in A_1, f_1(x) = 1$  }  $\forall x \in A, (f_0 \vee f_1)(x) = 1$   
 $f'(x) = 1$ .

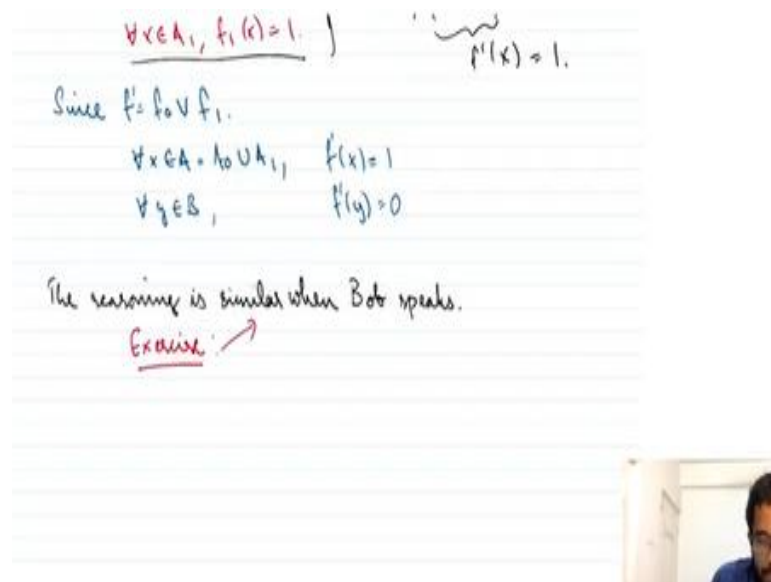
Since  $f' = f_0 \vee f_1$ .  
 $\forall x \in A = A_0 \cup A_1, f'(x) = 1$   
 $\forall y \in B, f'(y) = 0$



We know that for all  $x$  in  $A_0$ ,  $f_0$  of  $x = 1$ . In fact, I think I have written this in the bottom and for all  $x$  and  $A_1$ , I have written it here for all  $x$  and  $A_0$   $f_0$  of  $x$  is 1 for all  $x$  and  $A_1$   $f_1$   $x = 1$ . So, this means that for all  $x$  in so  $x$  and  $A$  means  $x$  is in  $A_0$  or  $A_1$   $f_0$  or  $f_1$  of  $x = 1$  but this is nothing but  $f$  prime of  $x$ , so  $f$  prime of  $x$  also satisfies, so I messed up here  $f$  prime is equal to  $f_0 + f_1$  and OR  $f_1$   $f$  prime of  $x = y$ ,  $f$  prime of  $x = 1$   $f$  prime of  $y = 0$  for all  $x$  and  $A$  and for all  $x$  in  $B$ , for all  $y$  and  $B$ .

So, the inductive claim goes from true for the level  $v$  also, so we are since we are doing bottom up induction. And the same reasoning is true when for the nodes where bob speaks it is just a symmetric argument instead of  $A$  being cut into two sets  $A_0$  and  $A_1$ ,  $B$  will be cut into two sets so it will be a vertical cut rather than a horizontal cut and pretty much what we said will be true in an analogous manner.

**(Refer Slide Time: 49:13)**



So, you can work this out as an exercise. So, what we have shown is that this circuit that we constructed computes the function  $f$ . So, the depth of the function is at most the deterministic complexity of the relation because to start with we could have chosen the protocol that is the best protocol that minimizes the communication. And you can see clearly because then, you can see clearly that the depth of this tree is equal to the complexity of the protocol.

So, we get that the depth of the function is at most the deterministic complexity of the relation, in fact it is actually the reduction is very symmetric, whatever we did in the first part, we are doing the opposite in the second part but it works. So, what we have is that the deterministic communication complexity of the Karchmer Wigderson relation  $R f$  is equal to the depth of the function which means that.

So, the best protocol whatever is the complexity equal to the depth of the best circuit that computes the function. So, this is the Karchmer Wigderson theorem that you can relate to, so given a Boolean function  $f$  you can construct this Karchmer Wigderson relation  $R f$  and such that the communication complexity of this Karchmer Wigderson relation is equal to the depth of the function.

This in itself is very interesting and what we will see in the next lecture is that we will apply this to a specific problem, the computation of whether there  $X$  is a perfect matching in a graph. And

we will derive depth lower bounds for this problem using this Karchmer Wigderson relation. Just to summarize, we defined relation we saw universal relation, we defined the communication complexity of the relation.

Then we saw this Karchmer Wigderson relation and the proof that of the Karchmer Wigderson theorem that the depth of the deterministic complexity of the relation is equal to the depth of the circuit that computes the function. In the next class we will see an application of the central matching, thank you.