Computational Complexity Theory
Prof. Raghunath Tewari
Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur

Lecture -38
Introduction

Hello. So, let us get started. So, what we will do today will just finish the parity lower bound proof and will look at some extensions of this result. And then I plan to discuss some bit of communication complexity. So, by itself communication complexity is a very vast area and probably it needs a course by itself to completely understand the techniques and the power of communication complexity.

But we will just give a brief overlook as to what model is and what kind of problems we expect to get solved using that model. So, what we had last time were these two lemmas so we had that suppose if we have an AC 0 circuit, so lemma 1 said that if we have an AC 0 circuit.

(Refer Slide Time: 01:23)

So, it can be approximated well in some sense, in the sense that it can be approximated by a proper polynomial p such that degree of p is, what was the degree that we had? 2t to the power d so, this was for all t and p matched the value of this AC 0 circuit on atleast size of C times 2 to the power n minus t inputs. So, this was the first lemma that we had and the second lemma was so this was lemma 1 and the second lemma was that if there is a root n degree polynomial if there

is a root n degree proper polynomial that approximates parity.

Then it cannot agree with parity on more than, what was the bound that we had? 9 10th of two to the power n. So, p agrees with parity where degree of p is less than root n. So, this is what we had so now it is easy to combine these two lemmas. So, what we do is we just set t equal to half of n to the power 1 over 2d. We never had such a constraint no for any t greater than 0, there are two to the power n inputs.

So, if that is true, but all we are saying is that the number of inputs, so what is this? This is basically the C agrees with p on so many inputs on atleast so many inputs. So, of course I mean C cannot agree with p on more than two to the power n inputs because that is the upper bound. But this is what all we are saying is a this is just a lower bound. So, t can be any number any positive integer. So, if we set t to be equal to this then what do we get here, we get that, so let D be the set of inputs on which C agrees with p.

Then what we have is that cardinality of D is greater than size of C times two to the power n minus half of n to the power 1 over 2d and degree of p is less than or equal to justify plug-in this number here, we get that this is less than or equal to n to the power half in lemma 1. And from lemma 2 what we have is that if you do have such a thing we have that this is less than or equal to 9 10th of 2 to the power n.

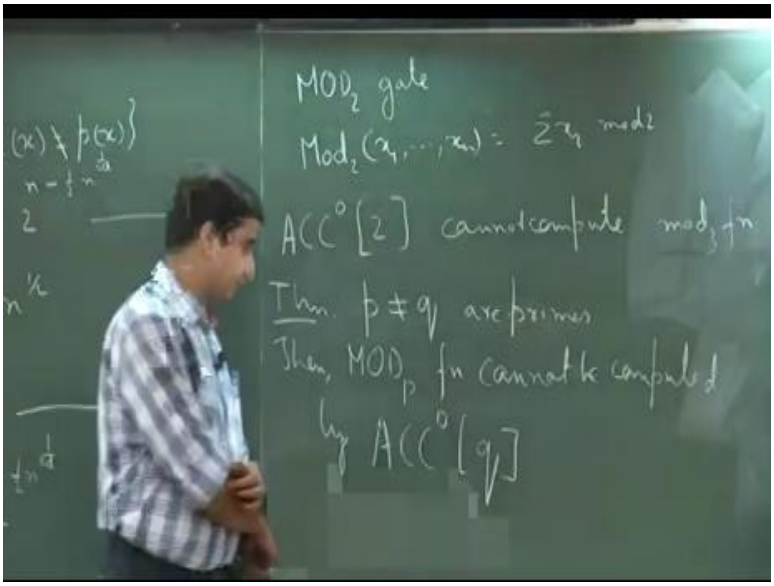
If you just combine these two equations what we get is size of C, so let me make a mistake. Does D is the set of points where the two things agree. So, lemma 1 says that if we have a AC 0 circuit then it agrees on atleast so many points and lemma 2 says that p agrees with parity on at most so many inputs. In the second condition, what is the lemma 1? Differs from C, so p agrees with P on 2 to the power n minus 1.

So, C differs with p on at most so many inputs and here what we have is? So, let D be the set of points where C and p differ then we have that D is less than or equal to this quantity and again, let us take the same things. So, P differs with parity on at least 1 10th fraction. So, now, I think it is correct. So, this is what actually I wanted. So, now I can just club these two together and we

have that size of C is greater than or equal to whatever constant it is 1 10th of 2 to the power n, 2 to the power n cancels of.

So, what we have is 2 to the power half n to the power 1 over 2t. And this is what theorem 1 or the first theorem whatever it was had claimed. So, is this clear to everybody? So, I was just confusing with the number of points on which it is same with what was actually stated in that lemma. So, that is the proof so now the question is that what we have is that AC 0 circuits are not able to compute a function, which is as simple as the parity function? So, what if I append an AC 0 circuit with gates which does compute the parity function for us?

(Refer Slide Time: 10:54)



So, let us say that I define a mod 2 gate as 1 that just outputs the following values. So, if the mod 2 gate is given inputs x 1 to x n it will just output sigma x i mod 2. And we can also generalize this. So, suppose if I append an AC 0 circuit that what I mean, is that together with AND OR NOT gates it can also have these mod 2 kind of gates. So, then what kind of power does these circuits have? So, there is a name to this also.

So, this is known as ACC 0 and in square brackets this 2 means that it has mod 2 gates. So, then again what can be shown just by an extension of this argument is that this kind of circuits cannot compute the mod 3 function. So, mod 3 function is that given x 1 to x n I will output 0 if the sum is divisible by 3 and I will output a 1 otherwise. So, the sum is not divisible by 3 I will output a 1

and this can actually easily be sure. In fact, the proof that is given in your book is actually for this particular statement.

So, what we had showed was that there is a proper polynomial which can approximate the NOT and the OR gates. But you can also show that there is another proper polynomial which can approximate a mod 2 gate very easily. So, therefore there is a proper polynomial which approximates a circuit of this form. And then on the other hand you show that now if I try to compute mod 3, that is not possible by such a kind of circuits. So, and this entire thing can be generalized to the following.

So, suppose p and q are distinct primes so they are not the same. Then the mod p function cannot be computed by an ACC 0 circuit with mod q gates. So, this additional C just stands for counter. So, it is a constant depth, polynomial size unbounded fan in circuit together with mod q gates. So, that is what it stands for. So, this is the general theorem, which actually (())((14:48)) proof. So, you are saying that you have one ACC 0 circuit, which has a all the primes up to some k.

So, that is the thing, so suppose if you have ACC 0 circuit for x forget about k, suppose it has a two kinds of gates suppose it has a mod two and mod three gates. So, suppose if we allow it something like this;

(Refer Slide Time: 15:24)

So, just denoted by ACC 0 and let us say it has mod two gates and it has mod three gates. So, what is the power of this circuit, so this was an open question which came up immediately after this result. So, this was for a single prime and people started asking that can I show that certain functions are not possible. So, people fail I mean, it was not possible to show up some easy functions cannot be computed by the circuit, so they tried to see that can I show some p complete problem that is not being computed by this circuit.

So, that was also not possible, people fail even for NP, even if you can NP complete problem they are not able to show that NP complete problem was not being computed by such a circuit. And this went all the way up to NEXP. So, it was an open question whether there exist a language in NEXP. NEXP is a very powerful class such that L does not have an ACC 0 circuit of this form. So, this was the open question.

Does there exist a language in NEXP? And after this result so this was in the late 80s after this result up to last year, in fact this was an open problem. And it was just last year by a guy named Ryan Williams that it was showed that there is a language in NEXP which does not have a ACC 0 circuits. So, now this problem is solved and it is a very recent result. So, you can see that a very simple addition, a very simple extension of what we have been seeing so far can make the computational model, quite powerful.

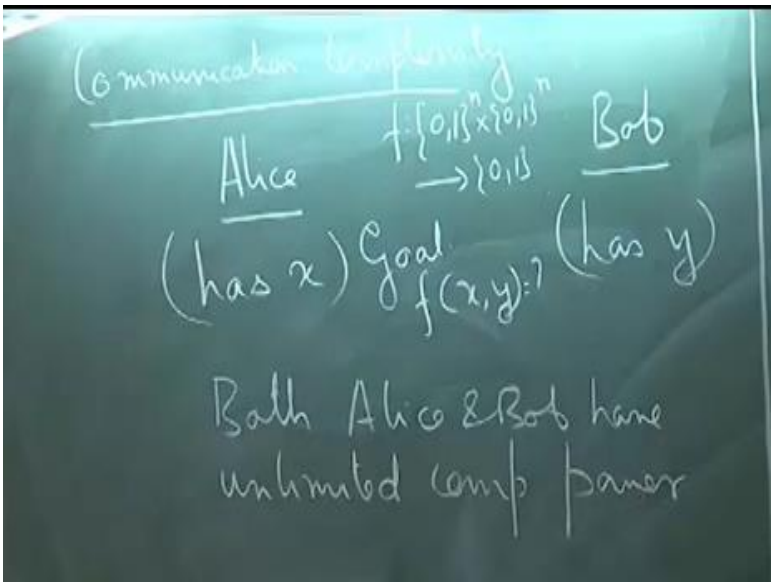
This is actually same as having mod six gates that can be shown. Suppose if you have mod 2 and mod 3, you can actually compute mod 6. So, ACC 0 6 can be computed using this whether this can be computed using ACC 0 6 is I think it should be but of course it can compute 12 also. But maybe this is a good question to think about so is ACC 0 6 equal ACC 0 2, 3. So, just think about this. So, one direction is immediately clear the other direction also should be possible.

So, immediately you cannot infer but then can you cleverly manipulate your input bits or can you do some clever construction so that you are able to get this information from this circuit. So, that is that. No 0, 1. So, if it for example, if I consider ACC 0 6 if the sum is divisible by 6, it will output 0 otherwise 1. Equally correct so if you have 2, 3 you can compute any linear combination of 2 and 3.

So, for 12 what you can do is so you can do it for six, and then for 12 you again, check it with another mod to get if it should be divisible by 6. Now if you are gonna do it any p to the power k can be done in mod p, but then how do you prove that any mod p to the power k can be done in mod p. So, think about it, I mean I do not want to go into these details. Now but I think it is a nice exercise to think these things.

So, as you said, I mean, if you have a, ACC 0 circuit with let say p gates for some prime p then you can compute the function mod p to the power k for any k using this kind of thing. So, this is a more general thing to show and once you show this then you can easily show that 12, 18 all these things are possible. 12 is nothing but 2 square and 3.

(Refer Slide Time: 23:05)



So, let us look at communication complexity. So, what is the model and what are we trying to do here? So, the model in a communication complexity setting is very close to this interactive proof model that we saw. But well, I should not say very close but it is kind of similar to this interactive proof model. So, there are two parties, so there is a person let us say named Alice, there is somebody called Bob.

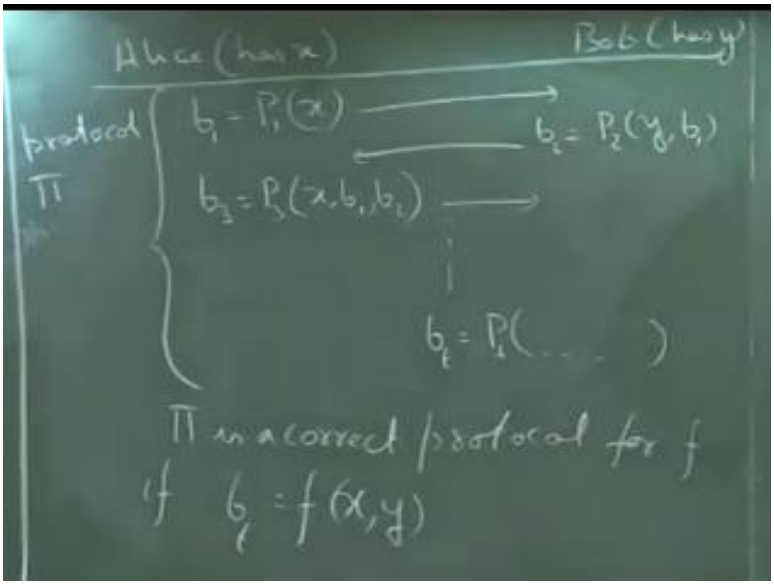
So, this is the traditional names that is a communication complexity people use. So, the thing is that so both these guys, they want to compute a function f. Together they want to compute a

function f so f is a function from 0, 1 to the power n cross 0, 1 to the power n to 0, 1. So, in other words f takes 2 strings to n bit strings and it outputs, a single output bit. And what we assume is that Alice has the first string so Alice has let say x and Bob has the second string.

And as I said, the goal is to compute f of x, y. So, what is f of x, y? So, the entire input is not presented to both the parties. The first party gets one of the input and the second party gets the other half. So, this is the main difference between the interactive proof model and the communication complexity model. When interactive proof model the input was visible to both the prover and the verify.

So, now the question is that how many bits needs to be communicated between these two parties? So, that finally they are able to answer what f of x, y is; and we always assume that both these parties have unlimited computational powers. So, both Alice and Bob have unlimited computational power. So, let us look at some basic definitions first.

(Refer Slide Time: 26:27)



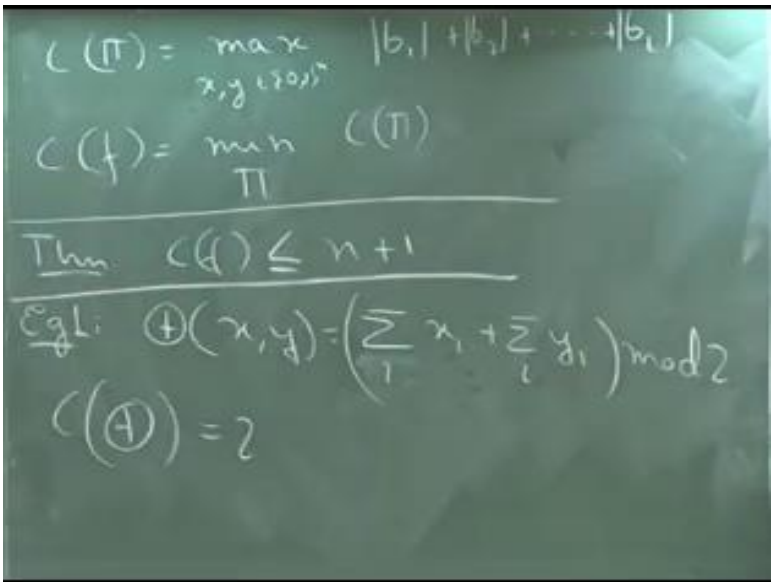
So, let us say we have Alice and Bob so Alice has x and Bob has y. So, what Alice does is? So, we define a protocol in the following manner. So, Alice compute the first bit b 1 which is applying some function P 1 on to the string that he has and he communicates this bit to Bob. So, now Bob can do the following. So, Bob applies some function P 2 on his own string y together with the bit that Alice gave him and let us say he gets some bit b 2.

So, I am just assuming here that these are single bits but I mean they can be strings. So, it is possible that instead of one bit they compute entire string. So, but let us just keep to this simple model and communicates this back to Alice and then Alice computes b 3 which is some P 3 applied on x is a initially computed bit b 1 and the bit that Bob gave him and this gets sent to Bob and so on.

So, finally a P t is I mean, whatever it should be. I mean if t is odd then P t will be x, b 1 through b t minus 1 and if t is even then it will be y comma b 1 to b t minus 1, whatever it is. And the protocol actually ends I mean, it is a correct protocol. If let us say that we denote this by pi, we call this our protocol. So, we say that pi is a correct protocol for the function if the last bit that whoever it is communicates, whether it is Alice or Bob is equal to the value of whatever the function intended to compute.

So, I do not this a bit incorrectly, so b t is P t of whatever it is, if b t is equal to f of x, y. So, they do serve a certain communications and then the final bit is a what the answer should be. So, now let us see what is the complexity of this protocol. So, what do you expect the complexity of this protocol to be?

(Refer Slide Time: 30:08)



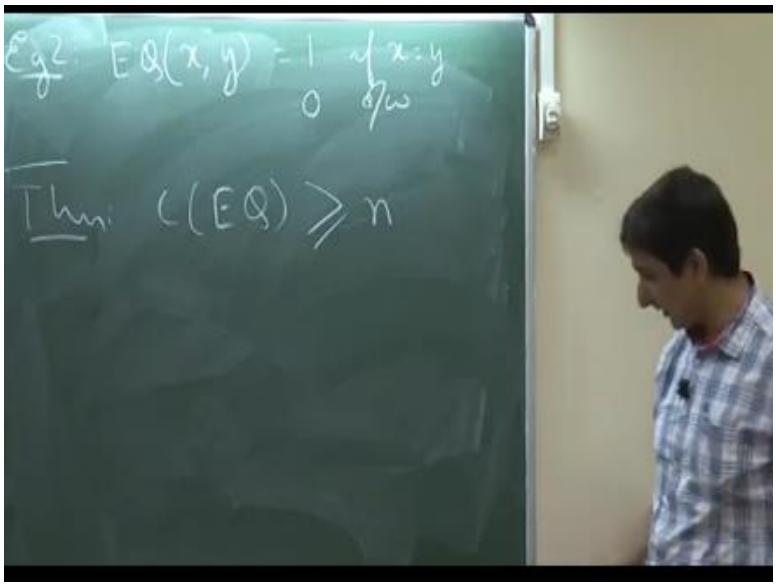
So, we define the complexity of this protocol to be the maximum over all choices of x, y of the

length of these strings b 1, b 2 so on till b t. So, for a protocol this is what its complexity is and the complexity of a function what do you expect this to. So, why do we take the max it is because I mean it should be I mean we are looking at the worst case complexity. So, overall possible strains, whatever is the maximum, number of bits that gets transmitted.

So, now what do you expect the complexity of a function to be the communication complexity? How do you define it? So, it is the minimum over all protocols pi, C of pi such that pi is the valid protocol for f. So, as many of you observed that trivial upper bound is n + 1. Because Alice can send x and Bob sends the answer. So, let us look at some specific examples. We will not get too much time today but let us look at least one example. So, suppose we are looking at the following extension of the parity function.

So, let us say I defined parity of x, y in the obvious manner, so it is parity of x i's I mean, it is a sum of x i's plus the sum over y i's mod 2. So, can you design a clever protocol for this function. So, Alice has x, bob has y and they want to compute this. And then 2 to be precise, so exactly Alice computes this quantity is sends that bit to Bob and then just bobs computes this quantity and takes the x or with whatever Alice had sent and outputs that. So, this parity function has a communication complexity 2. So, let us look at another function.

(Refer Slide Time: 33:30)



So, the second example is the most a more interesting example, this is the equality function. So,

we say that this function evaluates to 1 if x is equal to y and 0 otherwise. So, now what is the complexity of this function? The communication complexity. So, what can be shown by simple counting techniques is that so we will not discuss them today. But let me state the theorem is that the communication complexity of this function is lower bounded by n.

So, no matter what protocol you choose. Now, this is not obvious actually. I mean, it seems obvious but it is not so obvious. Because what we are claiming here is that no matter what protocol we choose, we cannot do better than n. So, we need to transmit at least n bits back and forth. Not really I mean, we do not need any compressibility kind of thing over. I mean, you can just and there are many ways to prove this and if you just do some clever counting argument, you will be able to show this and that is what we will do next time.

But what I am trying to emphasize is that when we want to claim a lower bound that basically should be over all possible protocols. And that is something which is I mean, even it seems obvious it needs a proper proof. So, let us stop at this point we will discuss the proof next time.