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

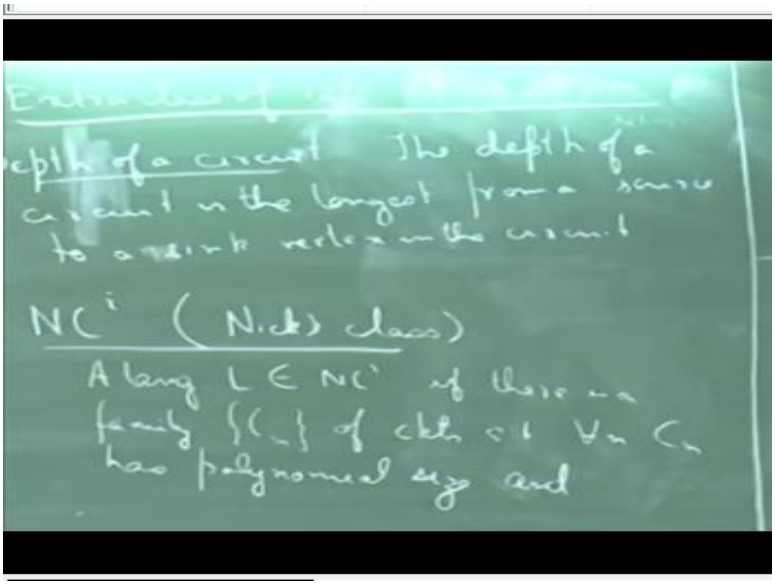
Lecture-16
Bounded Depth Circuit Classes

So, last class we saw the Karp-Lipton theorem and so, what was the essential idea of that theorem. The essential idea was of that theorem was to guess that circuit. So, firstly you show that if you have a polynomial time computable function F for any such polynomial time computable function F you can build a family of circuits which will guess a satisfying assignment of that circuit.

Suppose you have a polynomial time computable function F on two variables x and y, you can guess you can build a family of circuits that will compute a y if such a y exists. So that was the thing and then we saw that we can switch the quantifiers in the following way that we can guess that circuit beforehand. And then for all wise I can just verify whether that function together with that circuit evaluates to 1.

So, that was the main idea to be able to guess a circuit. So, let us look at a couple of things. So, as I had said earlier, we will look at other restricted forms of circuit where we restrict the depth of a circuit.

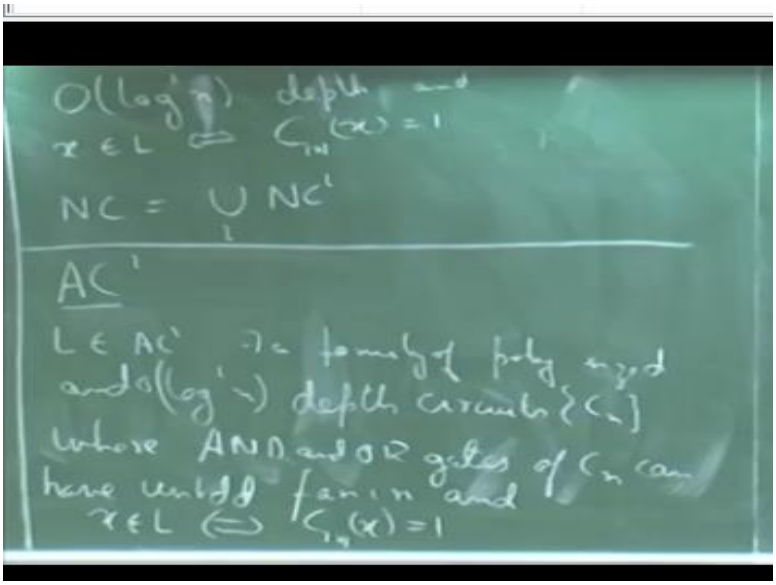
(Refer Slide Time: 01:51)



So, let us first define what we mean by so the depth of a circuit is the longest path from a source to a sink vertex. So, it is the usual graph theoretic notion. So, now based on this depth, we define these class of circuit families. So, let us first define NC i. So, just as an aside, so this NC stands for next class. So, Nick Pittenger was a complexity theorist. And he did a lot of work in this area of parallel complexity and he introduced this notion of parallel computation.

Well, I mean, he did not exactly introduce but he proved some nice results about parallel computation. So, this class is dedicated to him. So, NC i is defined as follows. So, we say that a language L belongs to this class, NC i if there is a family's C n of circuits such that for all n C n has polynomial size.

(Refer Slide Time: 04:58)



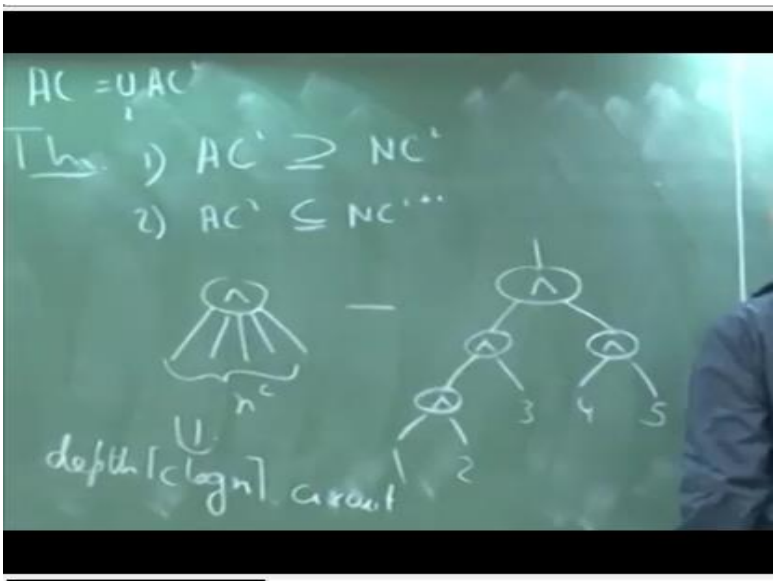
And order of log to the power i d of n and depth and x belongs to L if and only if C given x, so, that proper C evaluates to 1. So, we are just looking at circuit families which still have polynomial size but now we are restricting the depth to be poly log, we cannot have arbitrary depth. And in fact this exponent i in the definition of NC i corresponding to which level of the poly log hierarchy we are keeping this family N and NC is just the union of all these NC is.

So, NC 1 is basically those circuits that have polynomial size and log depth and NC 2 will have log square depth and so on. So, one crucial thing that you must note here is that, so, when we talk about circuits and we talk about the class NC, so, here we require these circuits to have bounded fan-in. So, if you recall we did define circuits which have bounded fanin.

So, that is why I explicitly did not mention it here. But then I also mentioned it that you can consider circuits which have unbounded fanon in particular for OR and AND gates. If you have an OR gate with unbounded fanin let us say it has fanin 10 or something that will evaluate to one if at least one of its input bits are 1 and a NAND gate will evaluate to 1 if all the input bits are 1.

So, that is the usual generalization of fanin of a circuit, but for the case of NC we are only looking at circuits which has bounded fanin. So, that is crucial and we will see very soon why that is crucial. So, similarly we can define another class AC. So, AC i is defined as class of all languages L such that there exists a family of polynomial sized and order log to the part i n depth circuits CN, where AND and OR gates can have unbounded fanin basically of CN can have unbounded fanin and of course, x belongs to L if and only if C of x evaluates to 1. So, here we are just again extending the family of circuits to have unbounded fanin.

(Refer Slide Time: 09:39)



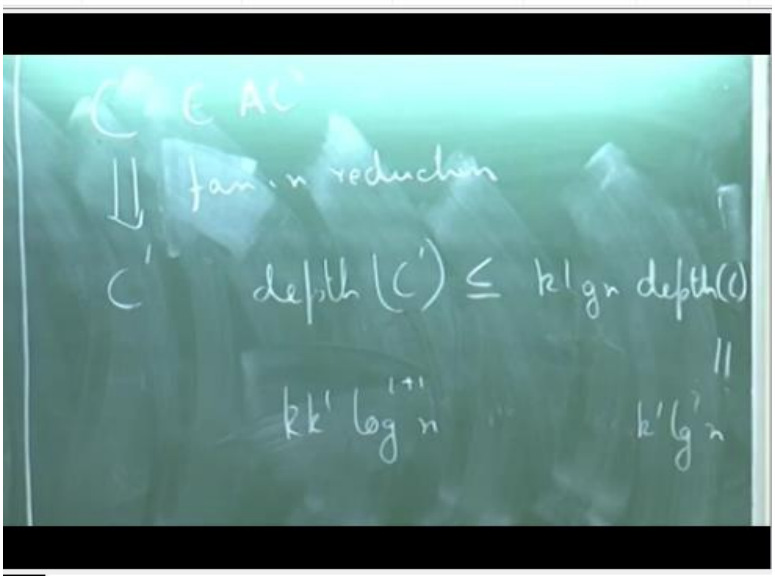
And similarly, AC is the union of all AC I's. So, what can we say about relations between these classes? What can we say about NC i and AC i. Let us say for example and by definition so the first thing is AC i contains NC i. So, if we look at different levels let us say if you look at 2 consecutive levels i and i + 1 can we say something for example, can we say something about AC i and NC i + 1.

So, the as is actually in i + 1 itself, actually AC i is contained in NC I + 1 and the reason is precisely what he said but you can generalize it again. So, suppose you have a gate. Suppose

you have a let us look at an example you have an AND gate which has fanin phi, let us see. So, you can simulate this same gate using a fanin 2 circuit, which has the ceiling of log of phi depth. So, it just does the following. You have 2 AND gates now. And then again, for this, you have 2. So, you feed the first input here, the second input here, third here, fourth here and fifth here.

And then if you consider the output at this node, you will see that this is just the AND of all these five bits. Okay? So, the idea is that no matter how much large fanin you have, so let us see even if you have polynomially, large fanin something like n to the power C, this can be simulated by a depth C log n to be precise a ceiling of C log n sized circuit Yes. So, that is why so, suppose you have an AC i circuit. So, suppose you have a circuit C, which belongs to AC i.

(Refer Slide Time: 12:56)



What I am claiming is that for every gate of that circuit, I can simulate it by a circuit, which has C log n depth or some order login depth. So, do this for every possible gate in that circuit. So, what is the increase in the depth of the original circuit C. So, you do this fanin reduction step, let us see, you get a circuit C prime. So, the claim is that the depth of C prime will be at most some k log n times depth of the original circuit, it will never increase by more than this factor.

So, if the original depth was C, now, I get a circuit which has depth, I mean now I get a circuit C prime, if you look at depth of C prime, that can be at most as large as log n times the depth of C. So, suppose C was in AC i what did it mean? So, if C is in AC i, what it means is

that there exists so, the depth of C is equal to some k prime log to the power i n. Now, we are getting another circuit no sorry, this is not C prime that was C.

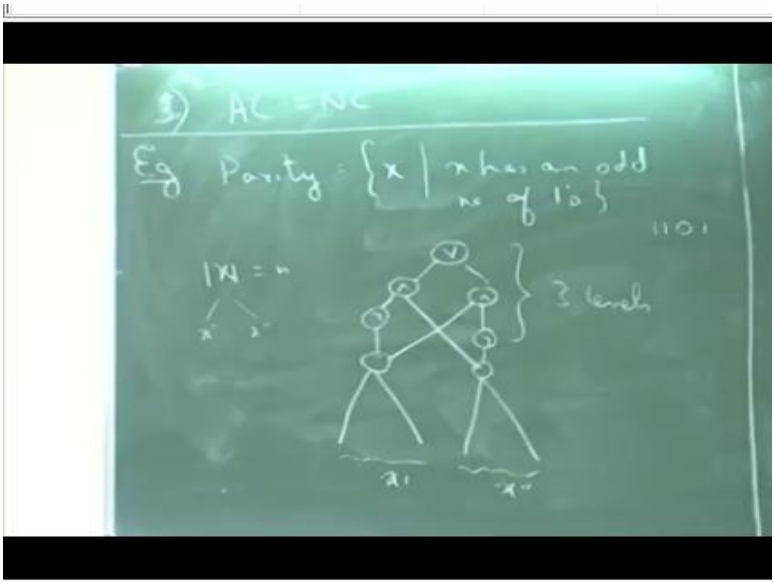
The depth of C was some k prime times log to the power i n. But now you are getting another C prime which is at most increasing that by a multiplicative factor of k log n. So, the depth of C prime would be nothing but at most k times k prime times log to the power i + 1 n, so that is what. So, at every vertex, you are not increasing it by factor of more than log n. So, therefore, in totality, it would only increase by a multiplicative factor of one extra login, and that is fine with us.

So, that is the thing. So, you can do this for both AND and OR gates. So, that will show that AC i is in NC i + 1, is this clear to everybody? A C is, it is alternating class, but, it is not named after person. So, alternating refers to the fact that so see now since you have unbounded fanin you can think of this so the levels of this circuit as alternating between AND and OR gates.

So, suppose you have many OR or you have many AND gates in various levels, you can just clump them all together into one AND gate or one OR gate and then the next level will be another AND or OR gate and then it will alternate. So, if you have bounded fanin such an alternation is not necessarily possible. But since you are allowing unbounded fanin, you can always think them.

So, if you take any path from the source to the sink, the gates would alternate, that is the reason. So, the third part of the theorem is because this is true. So, therefore, you can say that AC is same as NC, because this is true.

(Refer Slide Time: 17:12)



So, let us look at an example. Let us look at an example problem that is NC 1. So, this language is called parity. So, parity is the set of all strings x such that x has an odd number of 1's, can you design an NC 1 circuit for parity then C 1 would mean that it has to have bounded fanin and log depth AC 1. So, what you can do is just an another way of just stating what you guys said.

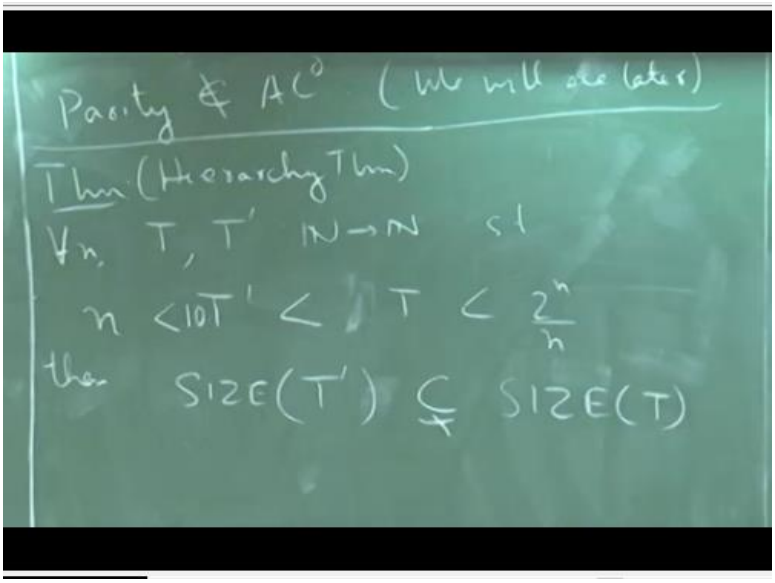
So, let us say you are given an input x which has length n. So, construct a circuit for x as follows. So, divide x into 2 parts. So, let us say you have the first half of x, let me call it x prime and the second half of x is x double prime. So, now suppose you have a circuit for x prime. And you already have constructed a circuit for x double prime that let us assume correctly computes parity.

That is if this has an odd number of 1's, this will output 1 and if this has an odd number of 1's, then this circuit will output 1. How can you combine these 2 answers to get the correct answer for the entire input? So, it is just basically taking the czar of these 2. So, what you can do is you take a NOT get here and then NOT get here, then you take AND here and then AND here, feed this and this into this AND gate.

And this input and this input into this AND gate and then have an OR gate on top. So, that total increase in level that you have made from going from x by 2 to x is some 3 steps some 3 levels. Look at the definition of this language. It is not whether all the 1's are I mean it is not where asking whether x has an odd length. It is saying that x has an odd number of 1's in it**.**

For example, this string would be in the language. So, if you are taking the AND of all these inputs, you would get a 0 here. But ideally, you should get 1. And it is actually what we will prove later on, so we would not prove it now.

(Refer Slide Time: 20:26)



Maybe sometime towards the end is that parity does not belong to AC 0. It is actually, that is one of the most well known lower bound results in complexity. It is quite complicated, but that is true. So, you cannot have a constant depth circuit even if you allow it unbounded number of inputs, you cannot have a constant depth circuits, which computes parity. So, that is quite a strong result. So, we will show this later on. That is not known. It is believed to be proper, but it is not known. So, let us look at a interesting theorem.

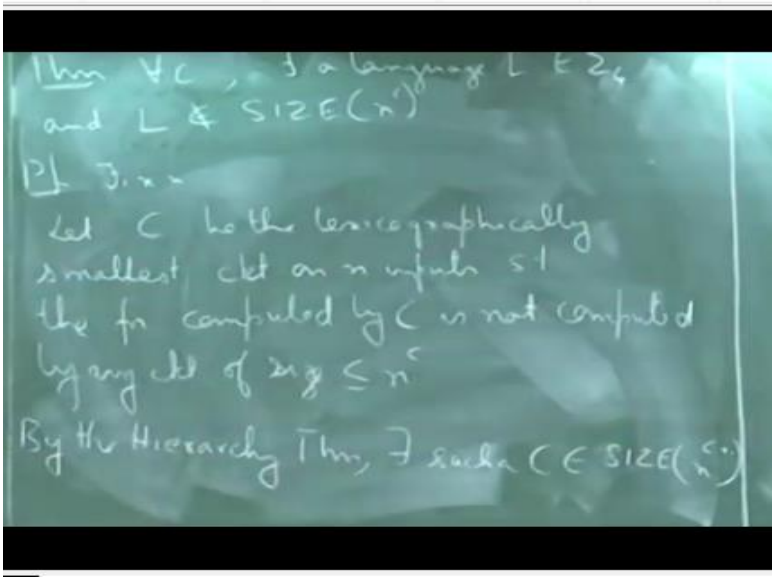
So, first we will just state a theorem which one? for i = 0 yes. As I said, but if you look at the individual levels of these hierarchies, is NC i contained in NC i + 1 are not properly of course, it is contained in it, but is it a proper containment or not? So, these questions are not known. So, what state here is a hierarchy theorem for circuits, but I would not prove it. So, for all N, if you look at functions, T and T prime going from N to N, such that they are at least as large as T prime and T is some 10 times T.

And they are upper bounded by 2 to the power n by n, then size of prime is properly contained in size of T. T is bigger than 10 times the value of T prime. Of course, the other way around is always true. So, if you have 2 functions T and T prime which are lower and upper bounded by n and 2 to the power n by n respectively, such that T prime is smaller than

T by 10. Then, there exists a language in size of T, which does not have T prime size circuit family.

And I would not give this proof because this is very similar to Shannon's theorem that we saw last time, you just have a counting, or you just do a counting argument, see count the number of circuits here and there and show that there is always some circuit family which belongs to this class, but not to this class. So, it is given in your book if you want, you can look at it. It is not difficult. So, what I will prove next for the remaining part of this class is.

(Refer Slide Time: 24:52)



I will prove this theorem that so for all constants, C there exists a language L in sigma 4 and L does not belong to SIZE of n to the power c. So, there exists a language which does not have a circuit of SIZE n to the power c for I mean no matter what c you pick will always be a language, but it is still contained in level of the polynomial hierarchy in particular in sigma 4.

So, one thing before I proceed to the proof of this, this is not saying that P by poly is properly contained in sigma 4. So, you should not confuse it with that notion, what it is saying is that for every c you can construct a language which has this property. It is not saying that there is some language here which is not contained in size of n to the power c for all possible c's, I mean it is the same as saying something like this.

I mean, if you look at the time hierarchy theorem that we discussed, so, that said that there is a language in some D time T to the power n square which is not in D time T to the power n or sorry T of n. So, that does not say that there is some language in some level of P which is

now what do I mean to say so, what I mean is that so, that does not give us a proof that n P is not equal to P.

Because if there is some language in some level of P of course, that is also in n P but then using the time hierarchy theorem, you can conclude that it is not in a lower level of P. But that does not prove that n P is separate from P. So, in the same way this does not show that P by poly is separate from sigma 4, it only shows that for every constant there is a language in sigma 4 which does not belong to size of n to the power c.

So, how do we prove this. So, let us C be the lexicographically smallest circuit on n inputs, such that the function computed by C is not computed by any circuit of size less than or equal to n to the power C. So, let us see what we mean here and more particularly, actually we will be looking for a C which would have size n to the power C + 1. So, let us see.

So, what I am claiming here is that C is a circuit which is lexicographically the smallest such circuit. So, you can always encode a circuit in a certain way. And if you look at encodings of all circuit, you can form a lexicographical ordering between them. So C is the lexicographically smallest circuit on n inputs such that whatever function C computes that is not computed by any circuit having size, at most n to the power C.

So, there can be many circuits, C which are different from any circuit of size at most n to the power C. But I am picking the lexicographically smallest such circuit. And by the hierarchy theorem, what we can claim is there exist such as C in size n to the power C + 1, because here I am saying that just within a constant multiple of a given time function, there exists languages, which is not contained in the smaller function sized circuit.

But here, I am allowing actually a multiplicative factor of n. So, of course, there will be a circuit which will satisfy this property. No, no, it is not taking n as input. So, when I said fix n, I mean that I am only dealing with all circuits, which have a fixed number of input gates. So, I am looking at all the inputs, which have length n. No, there only be finite many circuits, because every circuit is basically a graph.

So, if you fix an input gates, there will be some exponentially many graphs. So, is this clear to everybody because this is sort of the basic of this theorem. Here, by the hierarchy theorem.

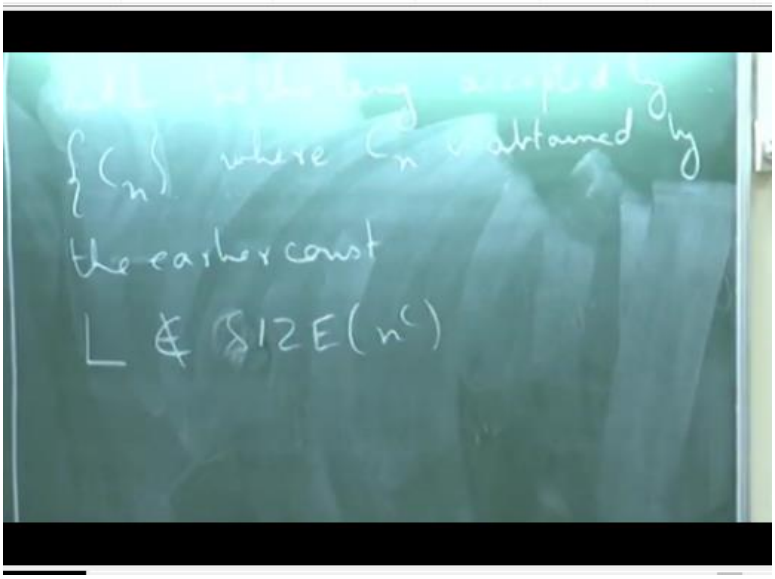
So, what is the hierarchy theorem saying us, suppose we have a function P. So here I am taking my T to be n to the power C + 1. And I am picking my T prime to be n to the power C, of course, that satisfies the hierarchy theorem, because T is strictly greater than 10 times T prime for large enough AND.

What the hierarchy theorem says is that for such cases, there exists a language here. So, when we say that there exists a language here, it means that there exists a circuit family, which accepts that language, such that that does not exist any circuit family here, which will accept the same language. So, that is what it means to say it is a proper containment. So, now what I am saying is that, if we look at C to be a circuit, which has this size n to the power C + 1.

The way I am defining C, the function that is being computed by C cannot be computed by any circuit, which has length at most n to the power C. Sorry, SIZE at most n to the power C**.** And there can be many sub circuits. So, the hierarchy theorem guarantees that there exists some circuit, but there can be many sub circuits. All I am saying here is that I am picking the lexicographically smallest sub circuit, that is all.

That will come later. So, that we have not seen. So, all I am saying is that we are guaranteed of the existence of such a circuit by the hierarchy theorem. Any other questions? So, what can we say about the language computed by this family C n?

(Refer Slide Time: 34:39)

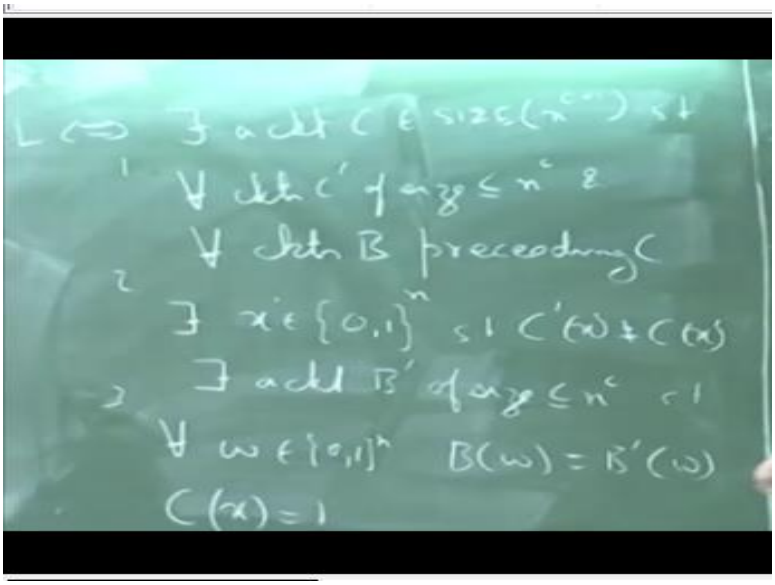


So, now if we look at C n. So, C n is the, let us say the family of such circuits. So, here we had fixed one, I mean for all choices of n we can get a circuit like this and that will yield a

circuit family C n. So, what can we say about the language accepted by C n. So, let L be the language accepted by C n, where C n is obtained by the earlier construction.

What can we say about this language L or in particular can we give some lower bound on this language that this will not be in someplace exactly from the very definition I mean C is those circuits such that the function computed by them cannot be computed by any circuit of size n to the power C. So, just by our construction L does not belong to size n to the power C. So, we have shown one part of the theorem. Now, we need to show that L is indeed sigma 4 and we will be done. So, how to show L is in sigma 4 we just need to give a properly computes every input and it can have at most 3 alternations.

(Refer Slide Time: 37:06)



So, all x of a particular size let us say when do I say that x is in L. So, I say that x is in L if and only if there exists a circuit C of SIZE n to the power C, well just write it here C n SIZE how much n to the power C + 1, such that for all circuits C prime of size at most n to the power C and for all circuits B proceeding C in the lexicographical ordering, there exists some input we call it x prime.

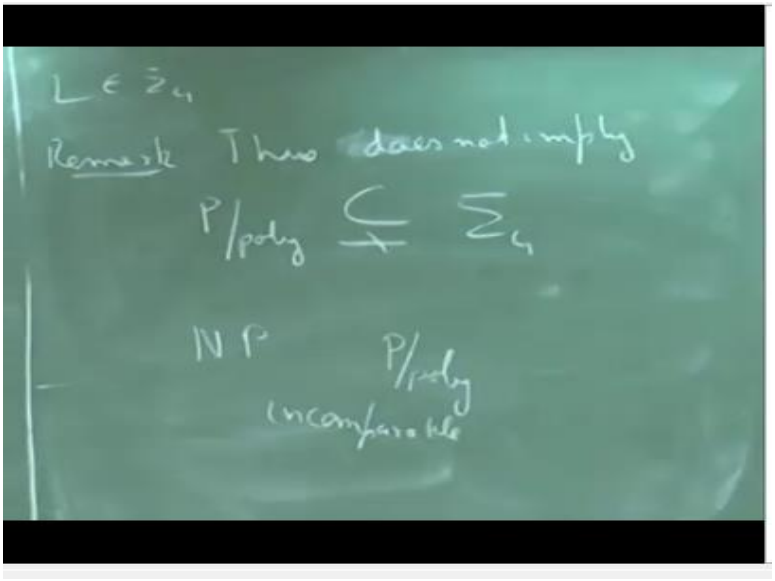
So, basically we need to so I will just tell you what we are doing here. So, what do we need to check to show that an x is properly computed, to show that x belongs to the language we have to show that well x is computed by C, x is not computed by any circuit that has size at most n to the power C also for any circuit that precedes C in the lexicographical ordering, x cannot be computed by that. So, there will be some string on which they would differ.

So, these are the things that we have to check in order to show that L is computed by this circuit family. So, there exists an x prime such that C prime of x is not equal to C of x and for all circuits B preceding C there exist a circuit B B prime of SIZE at most n to the power C, such that for all w n 0 1 to the power n B of w would be the same as B prime of w. And C given x evaluates to 1.

So, we have one alternation here, we have a second alternation here and we have a third alternation here. So, let us just carefully check whether this correctly computes the language for us or not. So, there exist some circuit having size n to the power C, such that if you take any circuit of size n to the power C I will use this existential quantifier to come up with an x prime such that C and C prime differ on that particular x prime.

So, this should have been x prime. Also for all circuits B that preceded C there would be some circuit B prime having size at most n to the power C, such that they would compute the same set of strings. So, here I am basically checking whether C is the lexicographically first such circuit differs from any circuit of type at most n to the power C and then C x has to be equal to 1.

(Refer Slide Time: 42:27)



So, this shows that L belongs to sigma 4, again let me empathize because this is very important, so we are only doing this for a fixed C. So, this does not imply that P by poly is properly contained in sigma 4 because that is again not believe to be true, we are only showing that there is for every C there is some language which does not belong here. So,

these 2 classes are incompatible, I mean sigma 4 is still at a higher level we cannot even compare n P and P by poly for that matter.

So, even these 2 classes are incompatible, any questions? Yes because otherwise you do not know, I mean may be the L that you have is also accepted by some circuit that is by some circuit which is lexicographically prior to C. So, here we are checking that it is not accepted by any circuit of SIZE n to the power C that part is ok, but we also need to check the other thing, that it is not accepted by any circuit which is lexicographically low as well.

Because there can be many such C, that is the point I mean the reason why you need to put that constant is that there can be many such C's. So, which C do you pick? If I pick the lexicographically smallest C and the question is how do you check that and to check that we are introducing may be couple of more alternations.