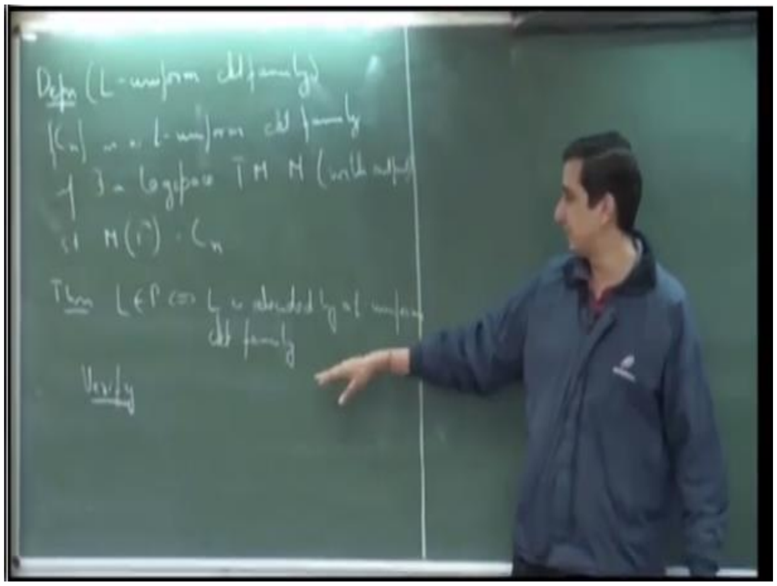
Computation Complexity Theory

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

Lecture-15
Shannon Theorem and Karp-Lipton-Sipser Theorem

So, yesterday the last thing that we saw was this P-uniform circuit families. And we also saw that if we have a language which is accepted by a P-uniform circuit family, it is exactly the same set of languages that are n P in the class P**.**

(Refer Slide Time: 00:37)


Defn (L-uniform ckt family)
{C_n} is a L-uniform ckt family
if ∃ a logspace TM M (with output)
s.t. M(1^n) = C_n
Thm L ∈ P ⟺ L is decided by a L uniform ckt family
Verify

So, similarly we can also define an analogous notion of log space uniform circuits**.** So, they are known as L uniform circuit families, and the definition is basically the same it is only that. Now instead of restricting the turing machine to run in polynomial time, you just impose a log space restriction on it is work time. So, C n is a L uniform circuit family, if there exist a log space turing machine M**,** so with output**.**

So, we are looking at turing machines which have an output tape also, because I want to write out the circuit finally, I want to print the circuit such that M on input 1 to the power n outputs the circuit C of n. And as it turns out that even, so although this is a restricted family of circuits. Because restricted in the sense that it is a subset of P-uniform circuits because of course if circuit can be generated by a log space machine, the same machine also we know runs in polynomial time.

So, we know that L is contained in P, so therefore this is a smaller family, yes, but nevertheless the following result still holds. That L is in P if and only if L is decided by a L uniform circuit family. So, yesterday we saw this theorem that the same theorem holds for P-uniform circuit families but it also holds for this stricter notion. And the reason is that so one direction is easy of course because L uniform is a subset of P-uniform.

But for the other direction what needs to be noted is that, again the same reduction that we saw yesterday, where we showed that P is contained in P slash poly. If you just observe that reaction that reduction carefully it is actually a log space computable reduction. So, at any point of time you are not storing more than logarithmic number of bits in your work day.

So, that is the observation one needs to make to prove this theorem so I will just leave that thing you can verify that yourself but that is the essential idea any questions? Karp even polynomial time Karp reductions. There are no such implications as such basically what this means is that I mean if I impose a certain kind of uniformity constraint on a circuit family, all I am saying is that then the class of languages that can be decided by those family of circuits is not more than P, so that is only thing.

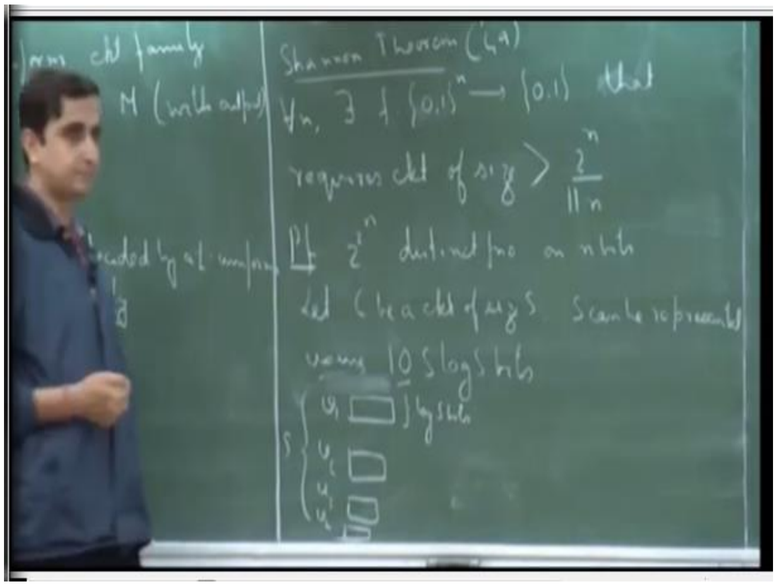
And what I am saying here is even if you look at a stricter notion of uniformity that is log space uniformity which is even more stricter than polynomial time uniformity. Even then I am claiming that the class accepted is exactly the set of languages in T. So, it is there is no, so I do not know what you mean by connection with Karp and log space reduction. No, but here we are not looking at reductions as such we are looking at machine which produces.

And yes, if you are looking at any circuit family that is being produced by these kinds of machines, then yes they yield the same family. But now there can be other interesting questions. So, suppose instead of looking at any circuit family if I look at let us say special kind of circuit families. Let us say circuits which have a small depth**,** so till now we have not talked about depth at any point.

I mean we did not put any restriction on what depth a circuit can have the only restriction that we talked about was that of size. But if you look at circuits having certain depth then we later see that log space uniformity is actually a stronger constraint than P-uniformity. So, we will probably see that in sometime later. But as far as general circuits are considered, general in the sense that circuits which only can be generated by an L uniform I mean circuit families which are either L uniform or P-uniform, there it does not make a difference**,** any other questions?

So, let us look at a lower bound result something which I briefly mentioned yesterday actually, but let us look at the proof of it.

(Refer Slide Time: 07:50)


Shannon Theorem ('49)
∀n, ∃ f {0,1}^n → {0,1} that
requires ckt of size > 2^n/11n
Pf 2^{2^n} distinct fns on n bits
Let C be a ckt of size S. S can be represented
using 10 S log S bits
log S bits
M (with output)

So, a lower bound result regarding circuits**.** So, this is not very difficult to prove**,** let me state the theorem first. So, this was proofed in 1949 by this guy called Shannon and it is historically named after his name. So, what he proved is that for all n there exists a function f that takes n bits as input and outputs either 0 or 1. That requires circuit of size at least as large as 2 to the power n by what did I have some 11 n**.**

So, this number is not very crucial I mean we will just see that it is because of certain back calculation that will end up with this number. But the point is that there exist certain functions for which I mean there exist certain function on n variables which requires circuits that have at least exponential size, so that is the main idea behind this theorem**.** So, I mean there are functions which are very hard to compute in some sense.

I mean you cannot have any polynomial time sorry any polynomial sized circuit for such function**.** So, and the proof is basically just a pigeonhole principle kind of argument. So, let us quickly run through it, so how many functions are there on n variables on I am sorry on n bits? 2 to the power 2 to the power n**.** So, there are 2 to the power 2 to the power n distinct functions on n bits**.**

So, now let C be a circuit of size S**,** let us say we are looking at all circuits or like we are looking at a particular circuit of size S. How many bits do we need to represent this circuit? What is a good upper bound on the number of bits that we need to represent a circuit? So, for that let us look at a way of representing a circuit, so one way of representing a circuit is of course by using a an adjacency matrix.

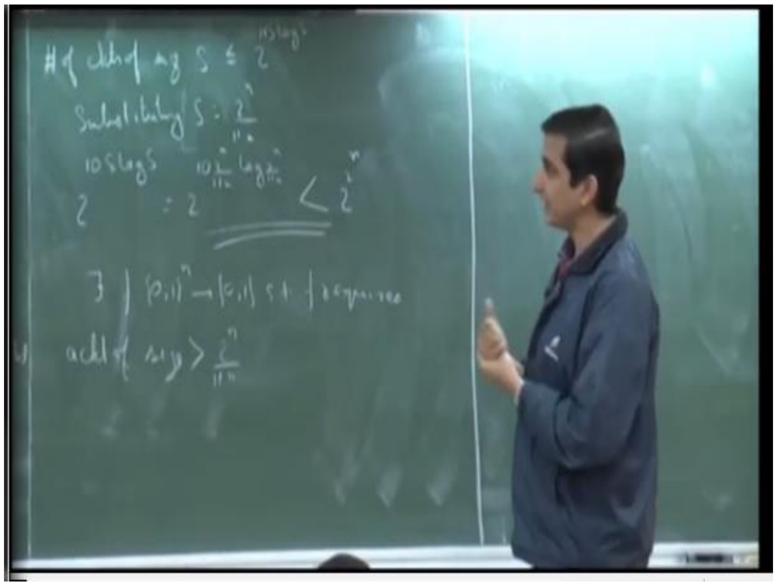
Because a circuit is nothing but a dag, so I can use but that is kind of too big representation. Because not only is a circuit a dag it is a dag which has a constant degree**.** So, every vertex has in degree 2, so I can represent a circuit using an adjacency list where each list has no more than 2 elements**.** So, there are S many vertices and each corresponding to each vertex there is a list of size utmost 2.

So, I can say something like, so C is a circuit of size S and S can be represented using sum let us say 10 S log S bits**.** So, this S comes from the fact that the number of vertices in this graph is S. And we need log S bits to represent one of it is incoming edges, so there are constantly many of them, so I am just being liberal enough and taking a large enough constant**.** Because maybe you need to also remember which is a I mean the labels on the vertices as well.

So, you need 1 or 2 more bits for that**.** So, let us look at this, so one way of representing a circuit is by using a adjacency list representation**.** So, let us say the circuit has 4 nodes 4 vertices**.** So, I have a vertex v 1, I have v 2, v 3 and v 4 and each of these lists have size 2**.** So, there are S many such vertices on my adjacency list representation.

And each such list has constantly many elements and to represent to one element you need log S bits**,** so that is why it is some 2 times log S plus maybe some other constants that is why I took an upper bound of 10**.** So, this 10 is not very important, the point is that it is some constant times S log S. So, there are S many things over here and to represent this you need good place. So, each of these needs log S bits some order log S**,** so is this clear? So, now how many such circuits are there of sizes.

(Refer Slide Time: 14:47)



# of ckts of size S $\leq 2^{10 S \log S}$
Substituting $S = \frac{2^n}{11n}$
$2^{10 S \log S} = 2^{10 \frac{2^n}{11n} \log \frac{2^n}{11n}} < 2^{2^n}$
$\exists f : \{0,1\}^n \to \{0,1\}$ s.t. f requires
ckt of size $> \frac{2^n}{11n}$

So, the number of circuits of size S is equal to or is 2 to the power 10 S log S bits**.** Because each such circuit can be represented by a string of so many bits, and this is the total number of distinct such strings. So, now let us see where we get a contradiction, so we have 2 to the power 2 to the power n distinct function on n bits.

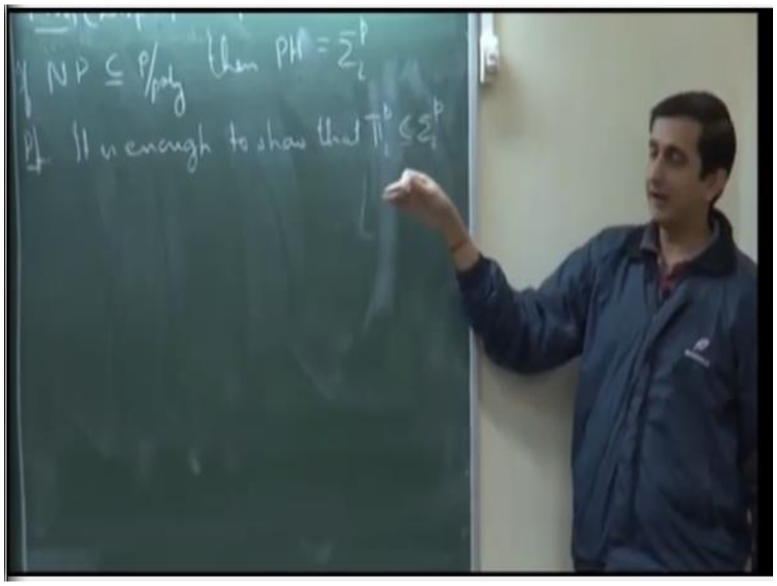
So, therefore, so if I substitute the value of S, so substituting S equals the size that we picked 2 to the power n by 11 n, 2 to the power 10 S log S is equal to 2 to the power 10, 2 to the power n by

11 n times log of 2 to the power n by 11 n**.** And if you work this out, this is not very difficult, if you work this out you will find that this is strictly smaller than 2 to the power 2 to the power n**.**

So, what this implies is therefore there are functions, so there is a function g from such that f requires a circuit of size strictly greater than 2 to the power n by 11 n. So, guys please check this inequality**,** because I just worked it out but might be a mistake. But again the main point here is that the size is, I mean there are functions for which we need exponential sized circuits**,** so that is a lower bound result.

So, which means that there are difficult functions as well. So, now let us come back to the other question that we posed at the very beginning. That so we saw that P is contained in P by poly, so what happens if n P is contained in P slash poly as well**.**

(Refer Slide Time: 17:58)


If NP ⊆ P/poly then PH = Σ_2^p
Pf. It is enough to show that Π_2^p ⊆ Σ_2^p

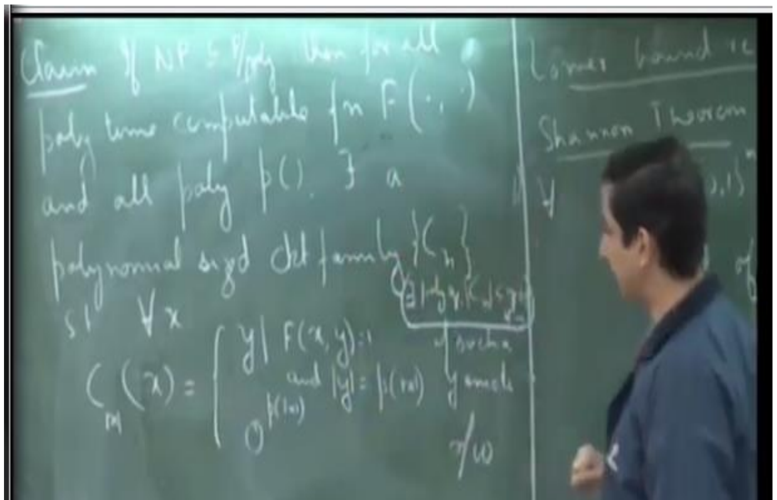
Yes 11 is just because I want, so if you just work this out I mean there are much better optimal bounds known here**,** there are even larger sizes for which the statement is proven. So, I mean I was just it basically all depends on what is the representation of the circuit that you are choosing, how many bits you are using to represent that**.**

So, those things are taken care I mean those things are taken into account, and that does give a better representation which uses lesser number of bits and which does give more optimal bounds than this. But the fundamental point is that whatever optimal bound you get it is still exponential. So, let us look at this theorem this is quite interesting as well. So, this was first proven by Karp and Lipton in 1980 and then subsequently it was actually improved by Sipser I think in 81 or 82.

So, we can credit all these three guys for this theorem. So, what they showed was if NP is contained in P slash poly, then the polynomial hierarchy collapses to the second level. So, actually what Karp and Lipton showed in 1980 was that if NP is contained in P slash poly then the hierarchy collapses to the third level, that is sigma 3 P. And then subsequently Sipser improved it to sigma 2 P.

So, let us see how we can prove this, so what is sufficient to prove in this case is. Because if pi 2 P is subset of sigma 2 P, then we as we have seen earlier what it will mean is that sigma 2 P and pi 2 P are the same classes. And if they are the same then it would imply that the hierarchy collapses to that level. So, we have seen this when we were looking at polynomial hierarchy.

(Refer Slide Time: 20:51)


Claim If NP ⊆ P/poly then for all
poly time computable fn F(·,·)
and all poly p(). ∃ a
polynomial sized ckt family {C_n}
s.t. ∀x
C_|x|(x) = y | F(x,y)=1 and |y| = p(|x|) if such a y exists
0^{p(|x|)} o/w
Lower bound
Shannon Theorem

So, let us look at a claim before proceeding to the actual proof. If NP is contained in P slash poly then for all polynomial time computable function F. So, let us say we are looking at functions which take 2 arguments for all polynomial time computable functions F and all polynomial P there exist a polynomial sized circuit family C n. Such that for all x if you run the circuit C mod x whatever is the length of x, if you look at that particular circuit.

And you run the input x on that circuit, so that will give a string y such that F of x, y = 1 and cardinality of y is less than well I can say equal also it does not matter. Let us just keep this equal cardinality of y = P of x. If such a y exists and otherwise what this circuit will output is a string of so many zeros. So, in the first case it either outputs a string y of length P x which makes this function evaluates to 1.

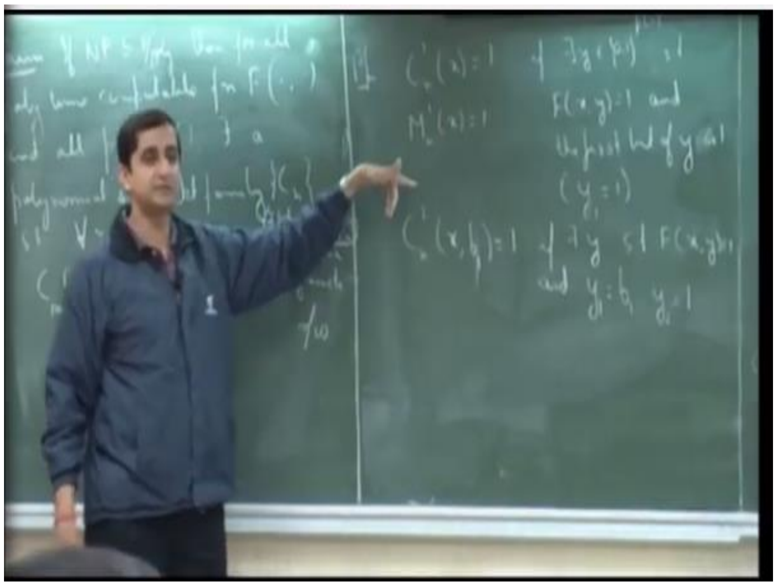
And if such a y does not exist then it outputs say string of so many zeros otherwise**.** So, let us understand what is happening here. So, under this hypothesis that NP is contained in P slash poly, that is NP has polynomial sized circuits. If you take any polynomial time computable function F on 2 arguments and you take any polynomial P. Then there is a polynomial sized circuit family**.**

So, maybe I can write this as there exists some polynomial q such that C of n has size less than q of n for all n**.** So, this is just another way of saying that there is a polynomial sized circuit family which has the property that given any x if you run the appropriate circuit on that input x, it will either output a y such that f of x, y = 1 and y has length P of x, if such a y exist.

And if such a y does not exist then it will output a string of P of x many zeros. Input to this is x yeah, so maybe I will let me clarify one more thing here. So, this is not the kind of circuits that we defined yesterday or the day before yesterday**.** So, here we are looking at circuits which actually have a output which is longer than 1**,** so we are outputting a string in particular, we are not outputting just one bit.

And that we said yesterday that it is very easy to generalize, I mean we can just look at circuits which have a more than one sink**.** And each such thing corresponds to one bit of the output string. Any y, any it will just output some y, so there can be many y such that y has length P of x and f of x, y = 1, in which case it will just output 1 such y. So, this is not very difficult let us see how this happens**.**

(Refer Slide Time: 26:56)



So, before proceeding to the formal proof let us look at a picture to motivate how the proof proceeds. So, what we will do is we will construct for each n, for each n we will construct P of n many circuits**.** So, what each such circuit will do is I will have a circuit C n of 1**,** so C n of 1 on x will output 1 if there exist a y of length P of n. Such that F of x, y = 1 and the first bit of y is 1**.** So, I will just refer to this as y 1 = 1**,** so this particular circuit given an x it will check if there exists a y of that particular length P of n, such that F on x, y = 1 and the first bit of y is 1**.**

So, what can we say about this circuit C n 1, so let us not go that far I mean we have not talked, but what can we say about the complexity of this circuit? The point is that, so if you look carefully at the definition of what is happening this circuit is actually realizable in NP or this circuit can actually be computed by an NP machine and that is quite simple. So, what that NP machine does is given an input x it guesses a y which has that particular length.

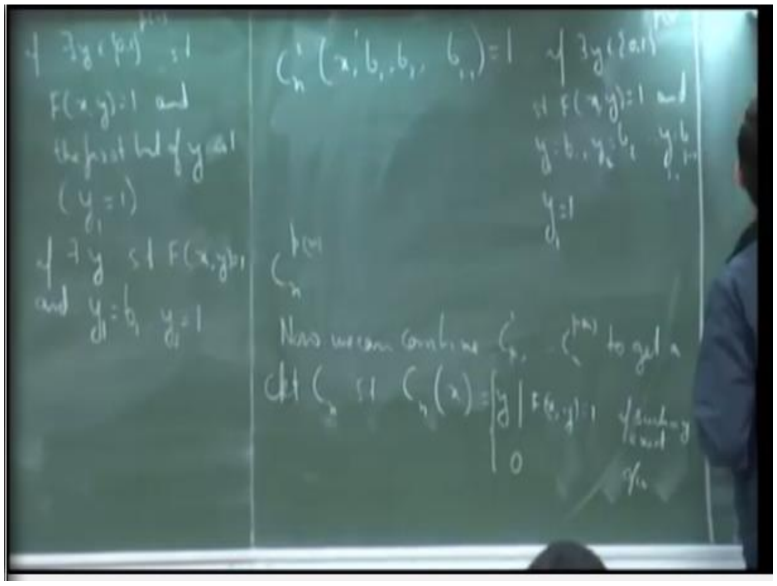
And then it substitutes x and y, so x is already given, so it substitute that x and y on to the function F, and since we are assuming F to be a polynomial time computable function**.** So, it just evaluate what is the value of x and y I mean what is the value of that function on given x and y. And it also checks whether the first bit of y is 1 that is also easily checkable. And if all these conditions are met it will output 1 otherwise 0**.**

So, it is very easy to see that this is checkable in NP, so is this clear to everybody. No so this has currently this has nothing to do with circuit, but we will come to that. All I am saying is that if you have such a problem where you want to design a circuit which has this property, this problem can be decided by an NP machine that is all I am claiming**.** And the reason why I am claiming that is basically because of the hypothesis. If there is an NP machine which can do this, then there is some circuit let us call it C n 1 prime which has polynomial size and which can also do the same thing.

So, that is the reason why I want to come with an NP machine for this**,** because I want to apply the hypothesis**.** So, now we will construct a sequence of such circuits, so similarly C n 2 given x will output 1**,** so this is not quite complete. So, C n 2 of x, so it takes x and it also takes the bit y 1, so this outputs 1 if there exists such a y such that F of x, y = 1 and y 1 is equal to here we should not use this same here.

So, it is let us say it is given a bit b 1, so y 1 = b 1 and y 2 = 1**.** So, now I am claiming that suppose you are given a string x and a single bit b 1. Again checking whether there exist a y which has length P of n such that F of x, y = 1 and the first bit of y 1 = b 1, and the second bit is equal to 1, again this can be checkable in NP.

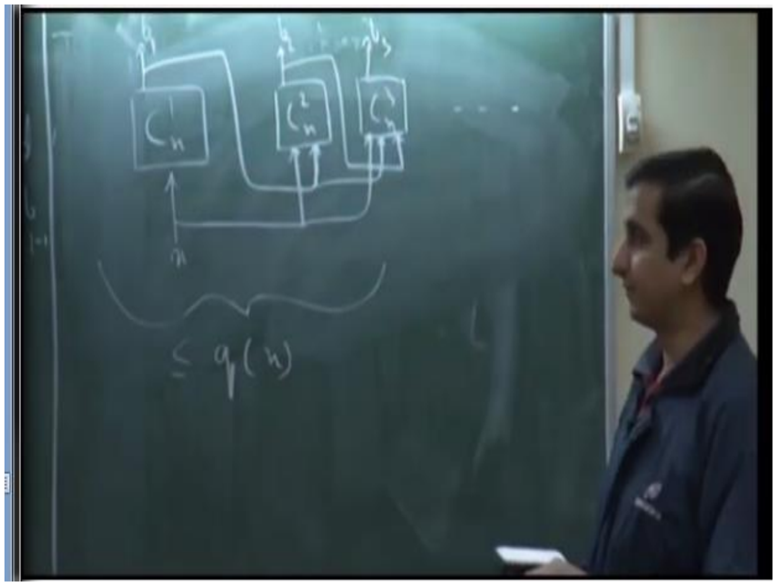
(Refer Slide Time: 32:25)


if ∃y ∈ {0,1}^{p(n)} s.t
F(x, y) = 1 and
the first bit of y is 1
(y_1 = 1)
if ∃y s.t F(x, y) = 1
and y_1 = b_1 y_2 = 1
C_n^i (x, b_1, b_2, ... b_{i-1}) = 1 if ∃y ∈ {0,1}^{p(n)}
s.t F(x, y) = 1 and
y_1 = b_1, y_2 = b_2 ... y_{i-1} = b_{i-1}
y_i = 1
C_n^{p(n)}
Now we can combine C_n^1 ... C_n^{p(n)} to get a
ckt C_n s.t C_n(x) = y | F(x, y) = 1 if such a y exist
0 o/w

So, now we can generalize this and define the circuit C n i that given x and the bits b 1, b 2 up to b i - 1 will output 1, if there exist a y of size P n. Such that F of x, y = 1 and y 1 is b 1, y 2 is b 2, y i - 1 is b i - 1 and y i is 1**.** So, if you have such an input again I am claiming that this circuit can be realized by an NP machine**,** with just guesses a y, it verifies whether the first i - 1 bits are equal to these given i - 1 bits, and the i th bit is 1 or not. So, similarly we do this and we get the circuit C n P of n**.** So, now we can combine C 1 sorry C n 1 up to C n P of n to get a circuit let us call it C of n such that C of n on x is equal to and it is 0 otherwise.

So, if such a y exists and it is 0 otherwise**,** so basically what we are doing is so as I said that each of these computations can be realized by an NP machine. So, therefore by our assumption there is some circuit for each of these computation, so now I can club all those circuits together**.** So, if you want to look at a pictorial view what is happening is, so suppose you have this circuit C n 1**.**

(Refer Slide Time: 35:45)



So, this is taking x as input and it outputs a bit b 1, so now I have C n 2, so I also feed x to this and along with that I also feed this bit b 2 this circuit b 1 onto this circuit. So, this will give me another bit b 2, now I have the third circuit. So, this gets x this gets b 1 and it gets also b 2**,** and this will give b 3 and so on. And now this entire circuit again this has size some polynomial in n, so is it clear how this is working**.**

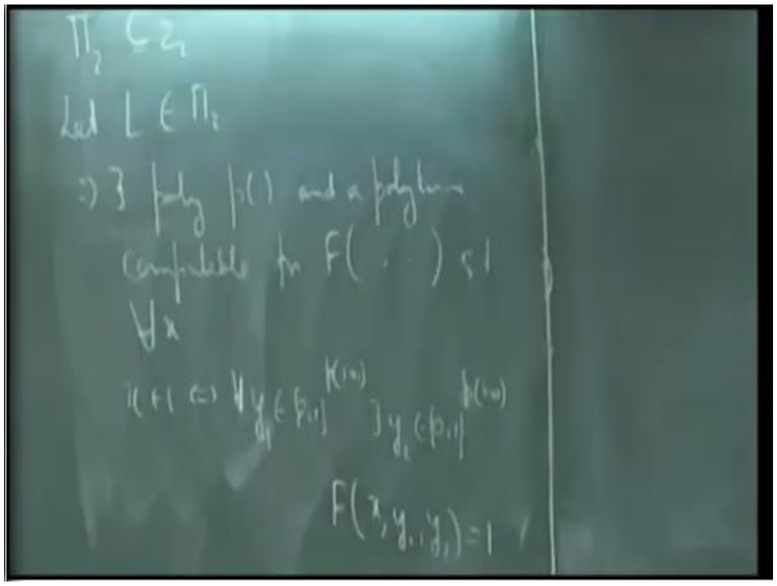
So, let us look at just one of these predicates, so suppose given an x, I want this circuit to output 1 if this happens. So, first of all the first claim is that this predicate can be computed by an NP machine. So, there is some NP machine let us call it M n of 1 that given x outputs 1 when this happens, is that clear. Because F is a polynomial time computable function, so the machine just guesses y and verifies whether F of x y is 1 and the first bit of y is 1.

So, now since NP is contained in P slash poly for such a machine there is a circuit which does that same thing, there is a family of circuit which is doing basically that same thing. So, I am just so that is the reason why I said that there is a circuit C n 1 which can compute this predicate, a polynomial sized circuit. So, if you are more comfortable, so maybe I skipped a step here but if you are more comfortable.

One way you can think of this is first think of this as a NP predicate that is being computed, and then think of there being a corresponding circuit a polynomial size circuit which exist for that NP predicate because of our hypothesis. So, now again the same argument holds for the second circuit, the third circuit and so on. So, this circuit I am claiming now has some polynomial size and in particular this has size some q of n q of that is what I had claimed.

So, I do not know if I have enough time to complete the rest, but let us see. So, now let us come back to our theorem it is not much left.

(Refer Slide Time: 39:13)



$\Pi_2 \subseteq \Sigma_2$

Let $L \in \Pi_2$

$\Rightarrow \exists$ poly $p()$ and a poly time computable fn $F(\cdot,\cdot,\cdot)$ s.t.

$\forall x$

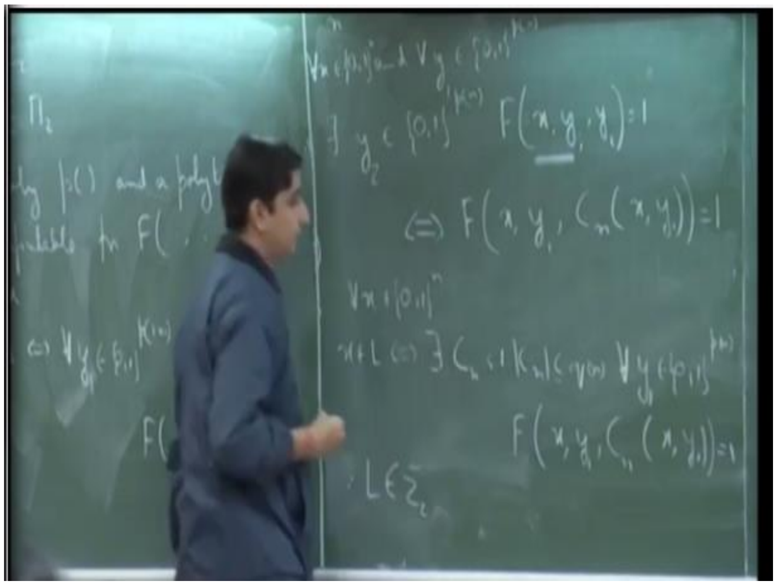
$x \in L \iff \forall y_1 \in \{0,1\}^{p(|x|)} \; \exists y_2 \in \{0,1\}^{p(|x|)}$

$F(x, y_1, y_2) = 1$

So, now we want to show that pi 2 is contained in sigma 2**.** So, which means that what we want to show, so if we have a language L in pi 2. So, this implies that there exist a sum polynomial P and a poly time computable function F on three arguments. Such that for all x, x is in L if and only if what is the definition of language being in pi 2. So, for all y 1 let us say having size P of mod of x, there exists a y 2 and F given x, y 1 and y 2 evaluates to 1.

This is the definition of pi 2**.** So, now how can we use our earlier claim, so what did we claim earlier that for all functions F and for all polynomials there exist a family of polynomial sized circuits. Such that circuit given x will output a basically certificate y, which makes that function evaluate to 1**.**

(Refer Slide Time: 41:41)



So, now what we can say is for all x and for all such y 1s**,** so let us say we are looking at some particular n**.** So, we are looking at some fixed length n, so we are looking at all x's of size of psi 0 1 to the power n and all y 1 of psi P of n there exist a y 2 of length 0, 1 to the power P of n such that F of x, y 1 comma y 2 = 1. So, this is equivalent to saying that F of x, y 1, C of n given x, y 1 evaluates to 1**.**

So, all I am saying is that so let us see what is happening here. So, we have a function F and we have a polynomial P**,** such that we say x is in L if this predicate gets satisfied. So, now from our earlier claim if this happens then we have a family of circuits which gives us a y which satisfies that predicate**.** So, just think of this as being the x in our earlier claim**,** this x and y 1 together as being the x in our earlier claim**.**

So, for all x here there exist**,** so therefore again for all x in 0, 1 to the power n, I can just restate this as follows. So, I can say that x will belong to my language L if and only if there exist a circuit sum such circuit such that C n has psi sum length sum C n S i is some polynomial in n and for all y 1 F given x, y 1, C n of x, y 1 evaluates to 1**.** So, now you see that this is basically a **a** sigma 2 predicate now.

Because now we have switched these quantifiers**,** so instead of having a for all in front we have a there exist. Essentially what we are doing here is that we are guessing that circuit apriori**.** And then we are checking that for all y 1 is this predicate getting satisfied. And this holds true for all n, so therefore L is in sigma 2, so that completes our proof. So, we are out of time, so we will stop here.