

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

Lecture - 36
Circuit Hierarchy Theorem

(Refer Slide Time: 00:15)

$S(n) = 2S(n-1) + 4$ & Base Case $S(1) = 1$.

This solves to $S(n) = \frac{5}{2} 2^n - 4 = O(2^n)$

Theorem (Lupanov): $L \in \text{SIZE}(O(2^n/n))$ for any L . \rightarrow Somewhat tight!

Hello and welcome to lecture 36 of the course computational complexity. In the previous lecture we saw that every language has a circuit of size 2^n and order 2^n . And then we saw that there is a better bound order $2^n/n$.

(Refer Slide Time: 00:33)



Lecture 36 Circuit Hierarchy Theorem

Theorem (Shannon): There are languages L
such that $L \notin SIZE(\frac{2^n}{10n})$.

Proof:

* How many circuits are there of size s



And at that point I mentioned that this is rather tight and we will see the proof. Why it is tight? So, the reason is there is a theorem by Shannon that says that there are languages that require at least 2^n divided by n size circuits. So, the state the formal statement that I have written here is that there are languages that require that cannot be computed in circuits of size 2^n divided by 10 times n . so, the denominator has an extra 10.

So, this 10 is somewhat arbitrary we will see why and we will lead we will prove this and then we will lead on to something at hierarchy theorem for circuits so we saw time hierarchy and space hierarchy. So, we saw Turing machines or if f is a function g is another function in terms of it and if f is sufficiently smaller than g then there is a language that cannot be computed in f but can be computed in g time.

Similarly, when f is sufficiently smaller than g then there is a language that cannot be computed in f space but can be computed in g space. This is what we saw, in exactly the same spirit we will see the circuit hierarchy theorem where we say that f and g are appropriately chosen function. So, that f is rather small and g is f is somewhat smaller as compared to g and also satisfying some other bounds.

So, you will see why this is important in that case there is a language that can be computed by g but cannot be coupled by f . So, this is what we will do in this lecture. So, first we will see Shannon's theorem.

(Refer Slide Time: 02:28)

* How many circuits are there of size s that use only \wedge , \vee and \neg gates?

→ For each gate we need to specify

→ What type gate? \wedge, \vee, \neg $O(1)$

→ For a specific gate, say gate i , where are the inputs coming from?

$n_1, \dots, n_k, \dots, n_m$

Why there are languages such that that cannot be computed in 2 power and divided by 10 in size circuits. And this is an extremely simple kind of argument all he does is some counting. So, let us see first as a first step let us see, how many circuits are there of a certain size? And then we will see how many functions are there of a certain number of a certain for a certain n ? And then we will say that if the size is small enough or sorry size is small enough then that size is not enough to even enumerate all the functions.

So, if under size is big you cannot come you cannot possibly compute all the functions. So, let us see the full proof. So, first let us see how many circuits are there of a certain size. So, to make it simple we will just restrict ourselves the De Morgan bases and gate or gate and not gates. So, it is easier to see to bound than the number of descriptions of a certain circuit and then use that to let us say we if you say that all the descriptions are of size 10 or 100 bits long.

Then we know that there cannot be more than 2 power 100 circuits that is what we will do. So, how do we describe a circuit? For each gate in the circuit, we need to specify two things one is what type gate it is. So, that is one of AND gate OR gate or NOT gate, these are the three things.

And this requires let us say constant bits maybe two bits are enough there are three possibilities or 2 bits are enough.

And give for a given gate let us say for a specific gate say gate i , which are the inputs or where are the inputs; coming from, I think I might need a bit more space. That is enough. So, let us see how this this works so given a gate we already said we will restrict to De Morgan basis and in fact the theorem statement says no theorem statement does not specify. But we will restrict ourselves to AND gate OR gate of fan 2.

(Refer Slide Time: 05:51)

where are the inputs coming from?

\wedge * left child, right child

$\log_2(n+s)$ $\log_2(n+s)$

To specify the gate: $2 + 2 \log_2(n+s)$

$\leq 2 + 2 \log_2 2s$

$= 2 + 2 + 2 \log_2 s$

$\leq 3 \log_2 s$

To specify s gates, we need $3s \log_2 s$ bits.

This requires $3s \log_2 s$ bits.

$\dots \rightarrow 3s \log_2 s \dots$



So, we need to specify left side where the left side comes from and we need to specify where the right side comes from. So, if an AND gate it has a left side and side we need to specify where these come from. So, there are we have already said that we are talking about circuits of a certain size s . So, there are s gates so it could be from any of these s gates there are also these n inputs so it could be from these n inputs also. So, the left child could be one of the $n + s$ possibilities.

And right child also could be one of these $n + s$ possibilities. So, to do in order to describe this we need $\log n$ plus bits to describe left child as well as a child because there are $n + s$ possible potential places from where you can choose the input. So, now to specify the circuit how to completely specify the gate we require so first of all let us say 3 bits constant number of bits let us say that is a constant. That may or maybe we can say 3 bits or maybe it is 2 is enough.

Let us say 2 bits to specify the gate type and plus this $2 \log n + s$ to specify the inputs. Now two $\log n + s$ so one way to bound n so the size of the circuit does not will at least be the length so n is upper bounded by the size. So, you could write it as $2 \log 2 + 2 \log 2 s$ and this $2 s$ can be this 2 can be again split into. This is equal to 2 plus $2 + 2 \log s \log 2$ is 1 and $2 \log 2$ is 2 . So, but now we have a constant 4 outside but then $4 + 2 \log s$ we could just upper bounded by $3 \log s$. This is to specify one gate to specify all the gates just to specify s gates we need $3 s \log s$ bits.

(Refer Slide Time: 09:02)

The slide contains handwritten notes in red ink. At the top right is the NPTEL logo. The notes are as follows:

- This means there are $\leq 2^{3s \log s}$ circuits.
- + How many functions are there
- $f: \{0,1\}^n \rightarrow \{0,1\}$? Ans: 2^{2^n}
- We will choose s so that $3s \log s < 2^n$
- If $s = \frac{2^n}{10n}$, no. of circuits is
- $\leq \frac{2^{2^n}}{10n} \cdot \log\left(\frac{2^n}{10n}\right) \cdot \frac{2^n}{3} \approx 2^n$

In the bottom right corner, there is a small video inset showing a man with glasses and a pink shirt speaking.

And once we have that the description of the circuit is $3s \log s$ and how many such circuits can be there, it is just 2 power that as I said already so it is 2 power $3s \log s$ circuits are there of size s . Now how many functions are there of a from $0, 1$ power n to $0, 1$? So, how many input combinations are there? The number of input combinations; are the 2 power n input combinations. And for each of these input combinations it could be one of two possible choices 0 or 1 .

So, this is the answer is 2 power 2 power n so you this is the answer this is the number of functions. So, for a certain sized circuit there are only 2 power $3 s \log s$ circuits of a certain size s but for a given n there are 2 power 2 power n functions. So, what we have to do is choose s in such a way that $3s \log s$ is smaller than 2 or so we need to choose s , we will choose s so that $3s \log s$ will be less than 2 power n and this is the key thing.

This is going to be the key thing $3s \log s$ is less than or equal to or strictly less than 2^n which means that for that size there are only this many circuits so it cannot possibly compute all the functions of from $0, 1$ power n to $0, 1$.

(Refer Slide Time: 11:15)

NPTEL

If $s = \frac{2^n}{10^n}$, no. of circuits is

$$\leq 2^{\frac{3 \cdot \frac{2^n}{10^n} \cdot \log(\frac{2^n}{10^n})}{1}} \leq 2^{\frac{2^n}{3}} < 2^n$$

So no. of circuits $<$ No. of functions.

Note: A randomly chosen function is

So, if you choose s to be 2^n divided by 10^n then if you compute $3s \log s$ it just happens to be 3 times 2^n divided by 10^n times \log of this. But \log of this can be upper bounded by \log of 2^n so it is just n so the n in and n in the 2^n and in the 10^n cancels so you get something like 2^n divided by one step if I want to add it is 3 times 2^n divided by 10^n times $\log 2^n$ divided by 10^n this is less than or equal to.

So, \log to power n I replace it by n so 3 times 2^n divided by 10^n and this is strictly less than 2^n and divide by 3 . Again, this I think this is a straightforward calculation what times I am erasing it. So, but anyway the point is at 2^n and divided by 3 this exponent is actually less than this exponent 2^n . So, and in fact 2^n divided by 3 is like one third of 2^n power n but once there in the exponent of 2 , 2 times 2^n divided by 3 is much smaller than 2 , 2^n power n .

So, this is much smaller so number of circuits of this size 2^n and divided by 10^n is ways smaller than the number of functions which means there is at least some function that is not

captured by a circuit of the size it cannot be the case that all the functions are captured because each function requires a separate circuit. So, that completes the proof that there are functions that require more than 2^n divided by 10 in sized circuits. So, that is the proof.

However, there is one interesting point here or 2 interesting points rather the first thing is that 2^n is way bigger than 2^n divided by 3. So, it is much bigger so just to compare if you want to take 2^3 is 8 and 2^9 is 512. So, even at the small scale it is so big or 2^{10} is 1024 and 2^{30} will be so 1024 is like a thousand to 2^{30} will be like a billion 10^9 .

So, it is you can see how big these are which means the number of functions; are way more than the number of circuits.



(Refer Slide Time: 14:13)

So no. of circuits < No. of functions.

Note: A randomly chosen function is likely to be hard.

Even then, very hard to find an explicit function that is hard. Best known is a function that requires $\Omega(n)$ size.

Theorem: For any f, g such that



Which means if you pick; a function at random so by random, I mean you could do any reasonable process one possibility is to look at all the inputs and flip them 0, 1 based on a random choice. If you pick a function at random it is very unlikely that it will have a small circuit by small, I mean by this side this size $3s \log s$ or 2^n sorry this size 2^n divided by 10 because more the number of circuits of this size is very small.

So, the number of functions that have a circuit of this size will also be small. So, a random function that you choose is going to be very likely to require a circuit of a bigger size. However, so you may think that most of the function if you pick a random function, it will require a lot it will require more size than 2^n divided by 10^n . But this is true but still there is this problem most of the natural functions that we can think of actually require only a small circuit smaller than this 2^n divided by 10^n .

In fact, people have tried to construct circuits that functions that require big circuits and people have been incredibly unsuccessful in this. So, again this proof is some kind of a not an explicit proof it just says it just counts the number of functions number of circuits and produces and says that there has to be a function that requires a lot circuit. But it does not tell you which function it is and people have tried to compute which function it is.

But without much success, in fact forget to power in size circuit or 2^n and divided by 10^n even n^2 even for n^2 . We do not know which function requires an n^2 size circuit. In fact, the best the best known explicit function let us say for if you just allow let us say AND gate OR gate and NOT gate of fan in 2 is of size roughly $4n$ or requires size $5n$. so that is just 5 times n so forget exponentials is we do not even know quadratic size.

If we do not even know linear beyond $5n$ let us say $10n$ we do not have we do not have a function that requires $10n$ size. So, we have this interesting situation where almost all functions are hard to compute but if I ask you produce one function that is hard to compute it is people have not been able to do that so that is the Shannon's theorem.

(Refer Slide Time: 17:28)



Theorem: For any f, g such that
 $n \leq 25n f(n) < g(n) \leq \frac{5}{2} 2^n$

$$100 2^n$$

$$4 \frac{2^n}{n}$$

we have $SIZE(f(n)) \not\subseteq SIZE(g(n))$

Proof: We need to make use of the results that we have already seen. We know that

* there are functions $h: \{0,1\}^n \rightarrow \{0,1\}$ that cannot be computed in $SIZE(\frac{2^n}{10e})$



And from there we will go to this hierarchy theorem. So, what I have stated here is if 25 times n times f n is strictly less than g n then there is a function. So, f n and g n are we talking about size their size bounds they are not the functions themselves their size their functions they denote the size of the circuit. In that case there is a function that cannot be computed in f n size but can be computed in g n size that is the theorem so it is a hierarchy theorem.

But interestingly both the hierarchy theorems that we saw so far time hierarchical space hierarchy use diagonalization. Whereas it is not so easy to or it is or rather it is not clear how you would apply diagonalization to the circuits. Because in the case of time hierarchy space hierarchy we have one we have we could list down Turing machines and so on. But here we cannot list down all the circuits because it is a circuit family itself has infinitely many circuits.

So, it is not clear how to apply diagonalization so what we will do is? We will not so whatever the ingredients that we required we have already have we have already shown. We will just use those ingredients. So, we show we saw that there is every language has an order 2 power 1 size circuit or rather 5 by 2 times 2 power 1 size circuit I am referring to this result. And we also saw that there are languages that that require more size than 2 power 1 divided by 10 1 size circuits.

So, we just use these will be our ingredients. And why do we have this upper and lower bound which so we certainly did not have an upper bound in the case of time hierarchy or space

hierarchy. This is because we need an upper bound because we know that all functions are computable in or all functions have a $5 \times 2^{2^n}$ power and size circuit. So, if you did not have an upper bound and if you took something like 100×2^n power $g(n)$ to be 100×2^n power n .

And $f(n)$ to be you divide 25 divided by n so you got something like $4 \times 2^{2^n}$ divided by n and this happens to be bigger than $5 \times 2^{2^n}$ power n . So, both g and f even f will be enough to decide all the functions. So, all the languages, so that is why it is important to have an upper bound here g should be sufficiently smaller than the number at by which you can compute all the functions.

If g is much bigger than that then even f will be bigger than that. So, we need an upper bound here and lower bound its n is like a trivial lower bound

(Refer Slide Time: 20:40)

The slide contains the following handwritten notes:

- that cannot be computed in $\frac{2^n}{10n}$
- Every function $h: \{0,1\}^n \rightarrow \{0,1\}$ can be computed in $\text{SIZE}\left(\frac{5}{2} 2^n\right)$
- We choose l such that $f(n) \approx \frac{2^l}{10l}$
- Choose an h & $\text{SIZE}(f(n)) \approx \text{SIZE}\left(\frac{2^l}{l}\right)$
- Now left $h: \{0,1\}^l \rightarrow \{0,1\}$

Red handwritten notes on the right side of the slide:

$$g(n) > 25n f(n) \\ > 25n \frac{2^l}{10l} \\ = \frac{5}{2} 2^l$$

The NPTEL logo is visible in the top right corner of the slide. A small video inset in the bottom right corner shows a man speaking.

What have we shown seen? So, far we have seen that there are functions that cannot be computed in 2^n divided by 10 size that cannot be computed into 2^n divided by 10 size. So, and every function can be computed in $5 \times 2^{2^n}$ divided by 2^n multiplied by 2^n size. So, basically it is just this is the factor 5×2 in the numerator 10 in the denominator if you multiply 10 with 5×2 you get 25 and that 25 is what we have in the statement here.

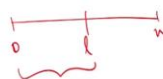
And this l in the denominator shows up as n in the numerator with f_n so that is how you get the 25^{th} n . So, we choose an l such that f_n will be roughly 2^l divided by 10^l . So, this is to power l so f_n will be roughly 2^l divided by 10^l that is all that we do. So, g will be so because of this condition g will have to be at least $5 \times 2 \times 2^n$. So, why is that? So, g_n is at least 25^n times f_n and we will choose n in such a way that n will be bigger than n .

So, this means it is at least $25^l \times 2^l$ divided by 10^l so l cancels so which is equal to 25 divided by 10 is $5 \times 2 \times 2^l$. So, g_n is bigger than 2^l , so which means that g_n will be able to will is enough to compute all functions from $0, 1^l$ to $0, 1, 0, 1$ g is sufficiently big 2^l capture all functions and f is certainly we know that there are many functions from $0, 1^l$ to $0, 1$ that cannot be computed f size.

So, now choose such a function h , that is computable or that is decidable in using a circuit of size that is not decidable with a circuit family or circuit family of size 2^l divided by l . And this h will automatically be decidable in g this h maybe I will write that h will also be in size g of n because of what we said here above. So, we defined ψ_h to be so we have chosen h to be from $0, 1^l$ to $0, 1$.

(Refer Slide Time: 24:00)

\downarrow Choose an h & $SIZE(h(n)) \approx SIZE(\frac{2^l}{l})$
 Now lift $h: \{0,1\}^l \rightarrow \{0,1\}$ to $h \in SIZE(g(n))$
 $h': \{0,1\}^n \rightarrow \{0,1\}$ by
 $h'(x) = h(\text{first } l \text{ bits of } x)$



Now all we need to do is to lift it to the 0, 1 power n domain so we will call it h prime which is from 0 1 power n. So, how do we do that all we need to do is define h prime of x to be so h prime is an on acts on an n bit input and we define it to be h computed on the first l bits of x first l bits. So, h prime is nothing but you look at the first l bits so this is 0 this is 1 to n or 0 to n. So, let us say this is l so h of h of this quantity is the h prime of the entire string.

So, now what happens is that h prime is not we cannot we do not have a circuit of size f n for h prime but we have a circuit of size g n. So, that is all that we need so that is what that is all that we wanted to show.

(Refer Slide Time: 25:11)

The slide contains the following handwritten text:

Theorem: For any f, g such that

$$n \leq 2^{\Omega n} f(n) < g(n) \leq \frac{5}{2} 2^n$$

we have $SIZE(f(n)) \not\subseteq SIZE(g(n))$

If we use Lupurnov instead, we could get a better result

$$n \leq 10 f(n) < g(n) < \frac{2^n}{n}$$

Proof: We need to make use of the results that we have already seen. We know that

The NPTEL logo is visible in the top right corner of the slide. A small video inset in the bottom right shows a man in a red shirt speaking.

In fact, if we used so this for this proof, we use that every function can be computed in size 5 by 2 multiplied by 2 power l which is what we showed in the previous lecture. This bound but however I said that there is a Lupurnov bound which says that every function has a size 2 power n divided by n size circuit. So, if we use that then we could get a statement if we use Lupurnov instead of the other bound we could get a better result.

So, n can be less than so, there the difference will only be a factor 10 because this will be 2 power l divided by 10 l the functions that cannot be computed in size and this will be just 2 power divided by l. So, this just it will just be a factor of 10 difference. So, strictly less than g n

and strictly less than 2^n divided by n . So, if you use, we will get a better result actually but then that is fine so in any way the point is that if you have more bigger size circuits.

You could compute more functions and that is the main point that I want to convey here. So, what we saw in this lecture was basically two results one was the Shannon's theorem which says that there are languages which require 2^n and divided by n size circuits or even more than that. So, this together with the Lupinus theorem shows an almost tight situation so every language requires every language can be computed by 2^n divided by n size circuit.

But 2^n divided by 10^n size circuit is not quite enough. So, it is only a constant factor of 10 and which was just by accounting argument Shannon's theorem that then you just counted the number of functions and we just counted the number of circuits. And we saw that when this size was this 2^n divided by 10^n then we could not the number of functions number of circuits itself is smaller than the number of functions.

And we mentioned the randomly chosen function in any so that in fact the number of functions that cannot be computed in this size are the majority or an overwhelming majority. So, most of the functions cannot be computed in this size. But it has been an embarrassing situation most of the functions almost all of the functions that we know are can be computed in simple easy small size circuits.

And we do not know of an explicit function that requires this much size and far from it in fact and then we saw the hierarchy theorem which used the both the Lyapunov and the Shannon's theorems and yes, I think with that we will conclude this lecture. Thank you.