CS 40: Computational Complexity

Sair Shaikh

November 18, 2025

Background: [1 slides]

There is an earlier proof that $P \neq NP$ does not relativize, as in there exist oracles A and A' such that in the relativized world you can prove $P \neq NP$ or $P = NP$. Thus, any proof technique that relativizes is not effective against $P \neq NP$.

This paper says you cant use "natural" circuit lower bound arguments to decide $P \neq NP$. Thus, similarly, if your proof "naturalizes", it is fruitless against $P \neq NP$ and you may as well abandon it.

Idea: [1 Slide]

Here's a common way in which you may hope to show $P \neq NP$ using circuit lower bounds.

1. Formulate some measure of "variation" or "scatter" on Boolean functions.
2. Show that polynomial sized circuits can only compute functions of low "variation". This is some combinatorial property C_n.
3. Show that SAT or some other function in NP has high "variation".

This paper shows that if PRGs exist and C_n is "natural" then such a proof cannot work.

Natural Combinatorial Properties: [2 Slides]

A combinatorial property is some subset $\bigcup_n \{C_n \subset F_n\}$. A function f_n is said to have the combinatorial property if $f_n \in C_n$.

The combinatorial property C_n is Γ-**natural** if it contains C_n^* satisfying the following:

1. Constructivity: $f_n \in C_n^*$ can be decided in Γ. Note that the "input sizes" here are 2^n, i.e. f_n as a truth-table.

2. Largeness: $|C_n^*| \geq 2^{-O(n)} * |F_n| = |F_n|/N^k$ where $N = 2^n$. Intuitively, there is a non-negligible chance for a random f_n to have this property.

When we say "natural" without qualification, we mean P/poly-natural.

A property is **useful** against P/poly if every sequence of functions f_n with $f_n \in C_n$ is superpolynomial. That is, for any k, there exists an n, such that f_n has a circuit lower bound $\geq n^k$.

Natural Proofs: [2 slides]

A proof is natural against P/poly if it contains some "natural combinatorial property" that is "useful" against P/poly. This definition is a little vague and you might have to work to show some property used in some proof is "natural". Towards the end of the talk, we will see that most things we can consider are natural.

Main Theorem: Such a property can be used to break any PRG.

Any proof that some function f_n does not have small circuits must either:

1. Use some very specialized property of f_n, one shared by only a negligible fraction of functions.

2. Must define a property so complicated that it is outside the bounds of mathematical experience.

That is, the proof must be unnatural by violating either "largeness" or "constructivity."
The authors claim there are no such examples in the literature, and there are theoretical reasons for why finding these is difficult.

Theorem Idea: [1 Slide]

A natural proof that some function f is not in P/poly must distinguish f from a pseudorandom function in P/poly. This can be converted into an algorithm that can tell f from a pseudorandom function in P/poly. This can be used to break a PRG.

Theorem Statement: [1 Slide]

Define the hardness $H(G_k)$ of a pseudorandom generator $G_k : 0, 1^k \rightarrow 0, 1^{2k}$ as the minimal S for which there exists a circuit of size $\leq S$:

$$Pr[C(G_k(x)) = 1] - Pr[C(y) = 1] \geq 1/S$$

Theorem 1. *There is no P/poly-natural proof against P/poly unless $H(G_k) \leq 2^{k^{o(1)}}$ for every psedorandom generator G_k. In particular, if 2^{n^ϵ}-hard functions exist, then there is no P/poly-natural proof against P/poly.*

Proof: [7 Slides]

[1st Slide]:
Assume for the sake of contradiction such a natural proof exists. Let C_n be the associated large and constructible property useful against P/poly. Let $G : \{0,1\}^k \to \{0,1\}^{2k}$ be a pseudo-random generator. Let $\epsilon > 0$ and $n = \lceil k^\epsilon \rceil$. We will do this in 3-steps:

1. Use G to construct "pseudo-random" boolean function in F_n.
2. Use C_n to provide a statistical test distinguishing random functions in F_n from "psuedo-random" these functions.
3. Convert this statistical test to a statistical test against G.

[2nd Slide]:
Step 1: Use G to construct "pseudo-random" boolean function in F_n.

Let $G_0, G_1 : \{0,1\}^k \to \{0,1\}^k$ be the first and last k bits of G. For $y \in \{0,1\}^n$, let $G_y = G_{y_n} \circ G_{y_{n-1}} \circ \cdots \circ G_{y_0}$. Let $f(x)(y)$ be the first bit of $G_y(x)$. Then, $f(x)$ is a Boolean function.

Visual: For some fixed x, take input y. Send x through G, take the first or second half based on y_0, send this through G, take first or second half using y_1, and so on.

Thus, $f(x)(y)$ is in P, and in particular, computable by poly-sized circuits.

[3rd Slide]:
Step 2: Use C_n to provide a statistical test distinguishing random functions in F_n from "psuedo-random" functions $f(x)$, $x \in \{0,1\}^k$.

As C_n is useful against $P/poly$, $f(x)$ is not in C_n for any fixed $x \in \{0,1\}^k$ and sufficiently large k.
Explanation: For fixed x, $f(x)$ defines a family of functions as you can input any length y. These have circuits upperbounded in size by some polynomial in n (n and k are polynomially related). Thus, we can go far enough to get some instance of $f(x)$ that's not in C_n.

Thus, C_n has empty intersection with $\{f(x) : x \in 0, 1^k\}$. This provides a statistical test:

$$|\,\mathbf{Pr}[C_n(\mathbf{f}_n) = 1] - \mathbf{Pr}[C_n(f(\mathbf{x})) = 1]| \geq 2^{-O(n)}$$

which is computable by circuits of size $2^{O(n)}$.

Note: By largeness, $\mathbf{Pr}[C_n(\mathbf{f}_n)] \geq 2^{-O(n)}$ and the second term is 0. By constructivity, this is computable by circuits of polynomial size in 2^n, i.e. in $2^{O(n)}$.

[4th Slide]:
Step 3: Convert this statistical test to a statistical test against G.

We represent the computation of $f(x)$ for all $y \in \{0,1\}^n$ as the binary tree T, i.e. the split at the ith level represents each successive bit of y, and the leafs represent the outputs. **TODO:** visual here.

[5th Slide]:
We order the internal vertices "children first", i.e. if $v_i \to v_j$ than $j < i$. Thus, we get $v_1, \cdots, v_{2^n-1}$.

Then, we look at all the sub-trees of the form T_i which contain all vertices upto v_i and all leaves. Let $v_i(y)$ be the root of the subtree containing y and $h_{v_i}(y)$ be the height of this subtree. Define $G_{i,y} = G_{y_n} \circ \cdots \circ G_{i,y_{n-h_{v_i}(y)+1}}$. **TODO:** Visual here.

[6th Slide]:
Define random collection $\mathbf{f}_{i,n}$ by setting $\mathbf{f}_{i,n}(y)$ to the first bit of $G_{i,y}(\mathbf{x}_{v_i(y)})$ for $\mathbf{x}_{v_i(y)}$ being uniformly and independently random.

Notice that $\mathbf{f}_{0,n} = \mathbf{f}_n$ and $\mathbf{f}_{2^n-1,n} = f(\mathbf{x})$. Note: $h_{v_0}(y) = 1$ and $h_{v_{2^n-1}}(y) = n$.

So, we get for some i,

$$|\,\mathbf{Pr}[C_i(f_{i+1,n}(x)) = 1] - \mathbf{Pr}[C_i(f_{i,n}(x)) = 1]| \geq \frac{2^{-O(n)}}{2^n} = 2^{-O(n)}$$

[7th Slide]:
Next, fix $\mathbf{x}_v$ for roots v of all subtrees in T_{i+1} except v_{i+1}. Then, we have a statistical test to distinguish $G(\mathbf{x}_{v_{i+1}})$ from $(\mathbf{x}_{v'_i}, \mathbf{x}_{v''_i})$ where v'_i, v''_i are the children of v_i.

Then, $H(G) \leq 2^{O(n)} \leq 2^{O(k^\epsilon)}$. As ϵ was arbitrary, we have broken G.

More general comments: [1 Slide]

More generally, if a complexity class Λ contains a pseudo-random function generator G that are sufficiently secure against Γ – i.e. any algorithm in Γ can only break G with exponentially small probability, then there is no Γ-natural proof against Λ.

Formal complexities measures are large: [3 Slides]

[1st Slide]
Define a formal complexity measure $\mu : F_n \to \mathbb{Z}$ to be an integer-valued function to satisfy the following:

1. $\mu(f) \leq 1$ if f is a x_i or $\neg x_i$.
2. $\mu(f \wedge g) \leq \mu(f) + \mu(g)$
3. $\mu(f \vee g) \leq \mu(f) + \mu(g)$

That is to say, $\mu(f)$ is a lower bound on the formula size for f.

Examples:

1. Circuit complexity.
2. The degree or number of monomials of the polynomial obtained by arithmetization.

[2nd Slide]
Any combinatorial property based on μ already satisfies the "largeness" condition. In particular, we have the following theorem:

Theorem 2. *Let μ be a formal complexity measure on F_n and let $\mu(f) = t$ for some f. Then,*

1. *For at least $\frac{1}{4}$ of all functions $g \in F_n$, $\mu(g) \geq \frac{t}{4}$.*
2. *For any $\epsilon = \epsilon(n)$, for at least $(1 - \epsilon)$ fraction of $g \in F_n$,*

$$\mu(g) \geq \Omega\left(\frac{t}{\left(n + \log\left(\frac{1}{\epsilon}\right)\right)^2}\right) - n$$

If you plug in $\epsilon = 1 - 2^{-O(n)}$, you get that for at least $2^{-O(n)}$ fraction of $g \in F_n$ have complexity $\frac{\alpha t}{n^2}$ for some $\alpha > 0$. Thus, for this to prove $P \neq NP$ the cutoff on what circuits can compute will have to be between $t/\text{poly}(n)$ and t.

[3rd Slide]
Proof of (a):
Let g be a uniformly random function in F_n. Write $f = h \oplus g$ with $h = f \oplus g$. Then,

$$f = (\neg h \wedge g) \vee (h \wedge \neg g)$$

Note that $g, h, \neg g, \neg h$ are all uniformly random. Now imagine $\{g : \mu(g) \leq t/4\}$ at least $\frac{3}{4}$ of all functions. Then, by the union bound,

$$\mathbf{Pr}[\mu(g), \mu(h), \mu(\neg g), \mu(\neg h) < t/4] > 0$$

But then $\mu(f) < t$ for this choice of g. Contradiction.

The general idea for (b) is similar, i.e. write f as a boolean combination of a small number combinations involving random variables, then use union bound and the probabilistic method.

Total: 20 slides