

The Natural Proofs Barrier and $P \stackrel{?}{=} NP$

March 13, 2019

1 Introduction

For decades the question of $P \stackrel{?}{=} NP$ has occupied a central role in the world of theoretical computer science and is such a well known problem that the reader surely knows both its statement as well as its significance. Although the vast majority of the computer science community believes $P \neq NP$ and many have dedicated tremendous efforts into proving lower bounds, we are not yet in sight of a resolution to this question. Therefore it is only natural to ask why it is so difficult to separate P from NP .

One natural answer to this question would be to show that certain commonly used proof techniques are not strong enough to imply the result. There are a few results of this variety. Most notably Baker, Gill, and Solovay proved in 1975 that “relativizable” proofs fall short, and Razborov and Rudich proved in 1994 that “natural” proofs also are insufficient given certain commonly held assumptions [BGS75; RR94]. The focus of this paper is on the latter, but the interested reader is encouraged to also read about the former, for instance in section 9.2 of Sipser’s textbook [Sip96], section 6.1.2 of [Aar16], or the original paper [BGS75].

The Razborov and Rudich result pertains to lower bounds of circuit complexity of certain problems. Circuit complexity was an area of great progress at the time, and many believed that it would lead to an ultimate proof for $P \neq NP$. Several groups were able to use combinatorial arguments to prove quite strong lower bounds for restricted classes of circuits, and it seemed only a matter of time before the techniques used in these results were slowly refined, building up stronger and stronger results until ultimately it was demonstrated that some NP -complete problem was not in $P/poly$. However, Razborov and Rudich defined their concept of a natural proof in such a way that it encompassed all of the circuit lower bound proofs known to date, and so their result presented a significant obstacle for this path towards $P \neq NP$ and reshaped the direction of work on the problem.

The argument behind the result follows loosely along the following lines. Define a *natural* property of a function as one that both applies to a random function with noticeably high probability and is able to be computed efficiently. Then any proof that relies on picking such a property that applies to some NP functions and shows that no function in $P/poly$ can have this property runs into the following issue. For any pseudo-random function in $P/poly$, the property will never apply (by choice of the property). However, it *will* apply to a truly random function with some detectable probability (again by assumption). This fact can be used to break pseudo-random generators, which is generally believed to be impossible. If

you define a natural proof as one that relies on a natural property, then it follows that no natural proof can separate P from NP .

At this point, all of these ideas are quite informal, and critically the argument is not interesting unless the idea of a natural proof is actually useful in regards to common proof techniques. Indeed, much of the Razborov and Rudich paper is dedicated to arguing that their definition of a natural proof does a good job at describing existing proof techniques of the time. We touch on this more in section 2 where we formalize and explore further the idea of natural proofs. In section 3, we introduce pseudo-random generators and pseudo-random function generators as well as proving a relation between the hardness of the two. In section 4, we put these pieces together to ultimately prove the following result.

Theorem 1. *There is no natural proof of super-polynomial circuit size bounds unless $H(\phi) \leq 2^{k^{o(1)}}$ for every pseudo-random generator $\phi : \{0, 1\}^k \rightarrow \{0, 1\}^{2k}$ in $P/poly$.*

In a way, this result is somewhat ironic. The reason that $P =?NP$ is so interesting and that so much effort has been spent attempting to show $P \neq NP$ is that there are certain useful every-day problems that seem to be very difficult to solve, but we have no way of showing this. This theorem shows that if we use natural proofs to conclude that some of these functions are hard to compute, we simultaneously give an algorithm for solving other “hard” problems. Indeed, in the same paper Razborov and Rudich demonstrate another unconditional result about the difficulty of proving bounds on the factoring or discrete logarithm problems. In the course of this proof, they show that any proof that proved stronger and stronger lower bounds for either of these problems would simultaneously give a more and more efficient algorithm for solving them.

2 Natural Proofs

To begin with, we define the notion of a natural combinatorial property, which is the center of the result. In order to avoid confusion, we first make precise what we mean by a combinatorial property

Definition 2. *Let F_n be the set of all boolean functions that take inputs of length n . A combinatorial property of boolean functions is a set $\{C_n \subset F_n : n \in \omega\}$*

We say that a boolean function f_n has a property C_n precisely when $f_n \in C_n$ or to use different notation $C_n(f_n) = 1$.

Definition 3. *A combinatorial property C_n is natural if there exists some $C_n^* \subset C_n$ such that the following two conditions hold*

- (i) **Constructivity:** *The predicate $f_n \stackrel{?}{\in} C_n^*$ is in P . That is C_n^* is computable in polynomial time with respect to the length of its truth table (2^n).*
- (ii) **Largeness:** *$|C_n^*| \geq 2^{-O(n)} \cdot |F_n|$ where F_n is the set of all boolean functions on inputs of length n .*

Intuitively, constructivity means that the property is not overly complicated. The fact that it can be checked efficiently whether or not a function has a certain natural property is critical in the ultimate proof of theorem 1. Notice that in the definition, we view boolean functions as strings of length 2^n that are simply truth tables. This is a useful perspective that we will use throughout. Implicit in constructivity is the idea that we can view a combinatorial property as a function, in particular the function that outputs 1 for every truth table that the property describes and 0 otherwise. Accordingly, we may view a natural property as a specific type of function in P . Largeness simply means that a random function possesses the property with non-negligible probability.

Finally, we define the notion of usefulness.

Definition 4. *A combinatorial property C_n is useful against $P/poly$ if for any sequence of functions $f = (f_1, f_2, \dots, f_n, \dots)$ such that $f_n \in C_n$, the circuit size of f is super-polynomial, that is for any fixed $k \in \mathbb{R}$, there exists an $n \in \mathbb{N}$ such that the circuit size of f_n is greater than n^k .*

Usefulness, true to its name, represents whether or not a property is potentially useful for separating $P/poly$ from NP . Clearly if a property contains functions in $P/poly$, then it will not be helpful when trying to prove that some functions are not in $P/poly$.

With these definitions in mind, we can now discuss natural proofs. We define a natural proof to be any proof that relies on a natural combinatorial property. This definition is quite loose and informal (as it was in Razborov and Rudich), but this does not effect the precision of the final result. In particular, Theorem 1 is best read more along the lines of “there does not exist a natural property that can separate $P/poly$ from NP unless...”. We can now start to put together an idea of what a natural proof might look like. In order to prove that some function $\{g_n\}$ does not have polynomial size circuits, you might choose some property C_n of g_n and attempt to prove that all functions f_n with this property (including g_n) do not have small circuits. That is, you show that C_n is useful. If C_n is easily computable and there are a significant number of f_n sharing this property, then the proof is a natural proof.

The important part of these definitions is that nearly all of the circuit lower bound proofs follow this template. In their paper, Razborov and Rudich demonstrate six different examples of how lower bound proofs for circuits can be shown to be natural. The process of showing this is often not easy, and so the technical details are omitted here as the goal is to study the barrier to $P \neq NP$ rather than the mechanics of known lower bound results. However, it is notable that not only does the definition of a natural proof include practically all circuit lower bound results, but also it is quite hard to construct useful seeming circuit lower bound proof that is not natural (for more analysis of such attempts, the reader is referred to [Aar16] section 6.2 and 6.6 or [Gow09]).

3 Pseudo-Random Generators

The proof of the natural proofs barrier relies heavily on the use of pseud-random generators as well as assumptions about their strength, so before getting to the proof itself we touch on these matters. A pseudo-random generator is a function that takes in k bits of input and deterministically expands this to an output of n bits that “looks random” for some $n > k$. In

particular, we would like for there to be no efficient algorithm that can tell the difference between the two. This inspires the following definition

Definition 5. Let $\phi : \{0, 1\}^k \rightarrow \{0, 1\}^n$ for $n > k$. Let $f : \{0, 1\}^n \rightarrow \{0, 1\}$ be a boolean function. Let x be uniformly random from $\{0, 1\}^n$ and y uniformly random from $\{0, 1\}^k$. We say that ϕ is a M -hard pseudo-random generator if

$$|\Pr[f(x) = 1] - \Pr[f(\phi(y)) = 1]| \leq \frac{1}{M}$$

for all functions f that are computable by a circuit of size at most M . We use the notation $H(\phi)$ to denote the hardness of ϕ , that is $H(\phi) = M$.

Notice that in this definition we use M in two places instead of having a separate parameter for each. This is because these individual parameters can be manipulated by repeating the experiment, and so up to a polynomial factor these are the only interesting values.

The assumption that the natural proofs barrier rests on is that there exist 2^{k^ϵ} -hard pseudo-random generators that produce a $2k$ bit output from a k bit input. This is a central assumption to many cryptographic applications and is widely conjectured to be true. Indeed, it is conjectured that there are pseudo-random functions that require $2^{p(n)}$ time to distinguish from truly random functions for any arbitrary $p(n)$ [Aar16]. Furthermore, the natural proofs barrier does not rely on the fact that a specific type of pseudo-random generator that is known of today can not be broken in polynomial time, but rather that there exists *any* pseudo-random generator that takes super-polynomial time to break: a much weaker assumption.

3.1 Pseudo-Random Function Generators

Recall that the proof given by Razborov and Rudich relies on using a natural property to distinguish a pseudo-random function from a truly random function rather than performing the comparison directly on strings. Therefore, we must introduce the concept of a pseudo-random function and demonstrate a construction of pseudo-random functions that will be useful for later analysis. Intuitively, what we are trying to do is to use a pseudo-random generator to create a pseudo-random function generator such that if we are able to tell the difference between a truly random function and a pseudo-random function then we could tell the difference between an element generated by our pseudo-random generator and a uniformly random element. The construction that we will consider was first proposed and analyzed in [GGM86], and the presentation of it here has much of its content based off of [Gow13] and [RR94].

Suppose we have some pseudo-random generator $\phi : \{0, 1\}^k \rightarrow \{0, 1\}^{2k}$. We can split the output of ϕ in half and think of ϕ as a pair of functions (ϕ_0, ϕ_1) where $\phi_0(y)$ is the output of the first k bits of $\phi(y)$ and $\phi_1(y)$ is the remaining k bits of $\phi(y)$. We can now define a random function $\phi_x : \{0, 1\}^k \rightarrow \{0, 1\}^k$ by taking compositions of the functions ϕ_0 and ϕ_1 based on the digits of x . Specifically, let $\phi_x = \phi_{x_n} \circ \phi_{x_{n-1}} \circ \dots \circ \phi_{x_1}$. For example, if $x = (1, 0, 0)$, then ϕ_x is ϕ_1 composed with ϕ_0 composed with ϕ_0 . Since we care about decision problems, we are more interested in having a function ψ with a binary output, which we can get by taking the first digit of the output of ϕ_x . Notice that ψ takes two inputs, $x \in \{0, 1\}^n$ and

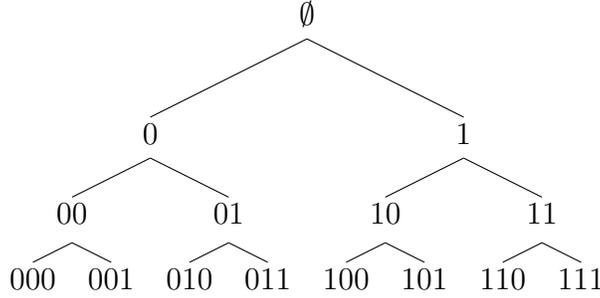


Figure 1: A depiction of T for $n = 3$

$y \in \{0, 1\}^k$. When describing the construction, we thought of ψ as a function of y that was determined by some other parameter x . From now on it will be more useful to think of $\psi : \{0, 1\}^n \rightarrow \{0, 1\}$ as a random function of x that is indexed by y . We use the notation ψ_y to denote this function. This will be convenient as we have been supposing that we have access to k bits of randomness, and so we can view ψ_y as a pseudo-random function that takes input x . However, recall that the goal is to make statements about the $P =?NP$ problem which does not have anything to do with randomness itself. Therefore it will be useful to view ψ_y as a *distribution* over functions. Ultimately, we will look at specific functions and try to determine if they were drawn from a uniform distribution or the distribution ψ_y .

There are two important properties of the distribution of functions ψ . First of all, if $k = \text{poly}(n)$, then ψ_y can be computed by polynomial size circuits since we take a composition of n functions that are computable in time polynomial in k . Second, the distribution of ψ_y is now closely tied to that of ϕ in that in a certain sense if ϕ “looks random” then so does ψ_y . Indeed, we claim that this construction of ψ_y is sufficiently “random looking” that if we can tell ψ_y from a uniformly random function (i.e. a uniformly random truth-table of length 2^n), then we will be able to break the underlying pseudo-random generator. This is formalized in the following lemma.

Lemma 6. *Let $f : \{0, 1\}^n \rightarrow \{0, 1\}$ be a uniformly random function, $g : \{0, 1\}^n$ be a function drawn from the distribution ψ_y , and $y \in \{0, 1\}^k$ be a uniformly random string. If there exists a polynomial time computable function A such that*

$$|Pr[A(f) = 1] - Pr[A(g) = 1]| \geq 2^{-O(n)}$$

then $H(\phi) < 2^{k^\epsilon}$ for any choice of ϵ .

Proof. Consider the set of all binary strings of length less than or equal to n . We use them to build a tree T such that the empty string is at the root of the tree and for each vertex in the tree representing string σ , its children are the strings $\sigma 0$ and $\sigma 1$. An example of such a T can be found in figure 1. We construct a sequence of 2^n subtrees of T which we call T_1, \dots, T_{2^n} in the following way. Let T_1 be the root of T , that is the tree containing a single vertex that is the empty string. For each T_i , construct T_{i+1} by taking T_i , arbitrarily choosing a leaf that has depth strictly less than n (i.e. has children in T), and letting T_i be the subtree containing T_i and the two children selected. An example of one such sequence is shown in

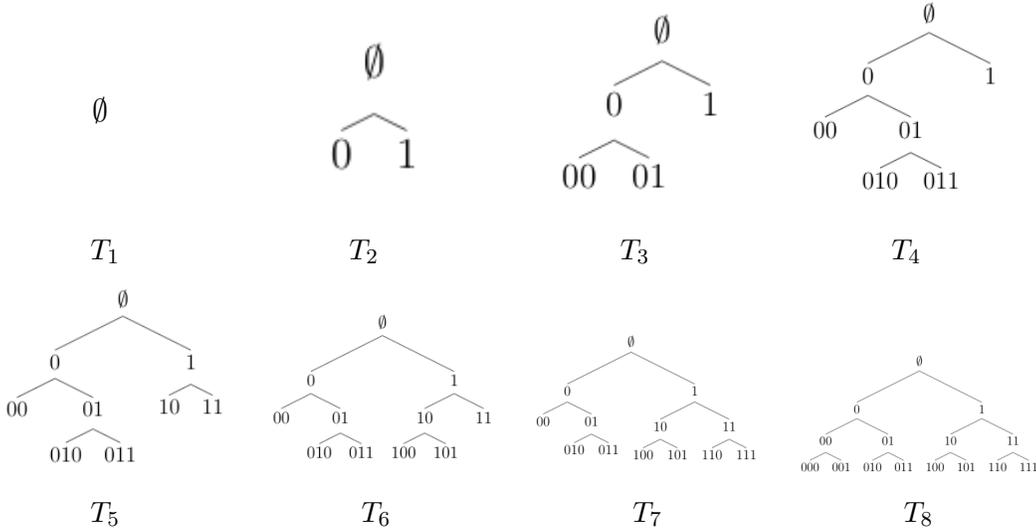


Figure 2: An example of one possible sequence $\{T_i\}$ for $n = 3$

figure 2. Notice that since there are $2^{n+1} - 1$ strings of length less than or equal to n , there are 2^n trees in our sequence, and accordingly, $T_{2^n} = T$.

From each T_i , we can create a probability distribution over binary functions with inputs of length n in a way similar to what was described above. Specifically for $x \in \{0, 1\}^n$ let m be the largest possible integer such that x_1, x_2, \dots, x_m is a vertex in T_i . For example, in the sequence depicted in figure 2, if $x = 100$ and we are considering T_3 , then $m = 1$ as 1 is in T_3 but 10 is not. Similarly, for T_6 we have $m = 3$ since all of x appears in T_6 . Take y drawn uniformly at random from $\{0, 1\}^k$ and take the first digit of the composition $\phi_n \circ \dots \circ \phi_{m+1}$. If $m = n$, then we simply take the first digit of y .

If $i = 2^n$, then every string of length at most n is included in T_i , including x itself. Therefore, $m = n$ and the resulting distribution of functions is just the uniformly random distribution since we just take the first digit of y . Similarly, if $i = 1$, then we are taking the first digit of the composition $\phi_{x_n} \circ \dots \circ \phi_{x_1}$, which is just ψ_y . Recall that by assumption there exists some function A such that

$$|Pr[A(f) = 1] - Pr[A(g) = 1]| \geq 2^{-O(n)}$$

for f uniformly random and g drawn from ψ_y . From now on, we will use the notation P_i to denote the probability that $A(f) = 1$ for f drawn from the distribution defined by T_i . With this notation, we can re-write the above bound as

$$|P_1 - P_{2^n}| \geq 2^{-O(n)}.$$

Therefore there must exist some i such that

$$|P_i - P_{i+1}| \geq 2^{-O(n)}.$$

The only thing that differs between T_i and T_{i+1} are the two leaves that are added. Specifically, there exists some σ which is a leaf of T_i , but such that $\sigma 0, \sigma 1 \in T_{i+1}$. We would like to

isolate the effects of this difference, so we condition on the random values that depend on all sequences other than σ , $\sigma 0$, and $\sigma 1$. Again, by an averaging argument there must exist choices of these random values such that we still have

$$|P_i - P_{i+1}| \geq 2^{-O(n)}.$$

This now gives us a method for breaking the underlying pseudo-random generator ϕ . Let $z \in \{0, 1\}^k$. We would like to decide whether z is uniformly random or is $\phi(y)$ for some y chosen uniformly at random. Based on the definitions of ϕ_0 and ϕ_1 this is the same as asking if z is of the form $\phi_0(y), \phi_1(y)$. Define $f : \{0, 1\}^n \rightarrow \{0, 1\}$ as follows. On input x , let τ be the maximal initial segment of x that is a vertex in T_i and let r be the length of τ . If $\tau \neq \sigma$, then take the first digit of the composition $\phi_{x_n} \circ \dots \circ \phi_{x_{r+1}}(y(\tau))$ where $y(\tau)$ are the random values that we conditioned on. Otherwise if $\tau = \sigma$, apply $\phi_{x_n} \circ \dots \circ \phi_{x_{r+2}}$ to the left half of z if the next digit of x is 0 and to the right half of z if the next digit of z is 1. Notice that if $z = \phi_0(y)\phi_1(y)$, then this is the same as taking the first digit of the composition $\phi_{x_n} \circ \dots \circ \phi_{x_{r+1}}$, and therefore we have a function drawn from the distribution defined by T_i conditioned on the appropriate quantities. However, if z is uniformly random then f is drawn from the distribution defined by T_{i+1} . Since as demonstrated above we can tell the difference between these in time $2^{O(n)}$ (i.e. polynomial in the length of the truth table), then the hardness of ϕ is at most $2^{O(n)}$. \square

4 Natural Proofs Barrier

With lemma 6 in hand, we now have the bulk of the technical work done to prove Razborov and Rudich's result. Indeed, much of what remains is given for free by the definition of a natural proof. We now complete the proof of theorem 1.

Theorem 1. *There is no natural proof of super-polynomial circuit size bounds unless $H(\phi) \leq 2^{k^{o(1)}}$ for every pseudo-random generator $\phi : \{0, 1\}^k \rightarrow \{0, 1\}^{2k}$ in $P/poly$.*

Proof. Suppose for the sake of contradiction that such a natural proof exists and let C be the natural property used in the proof. There must be some $C^* \subset C$ such that C^* is large and constructive, so without loss of generality we assume $C = C^*$. Then by assumption C is computable in polynomial time and applies to a random function with probability at least $2^{-O(n)}$. Additionally, for C to be useful against $P/poly$, C must not apply to any function in $P/poly$. In particular, C does not apply to any function drawn from the distribution ψ_y . Therefore viewing C as an algorithm we have

$$|Pr[C(f) = 1] - Pr[C(g) = 1]| \geq 2^{-O(n)}$$

for f uniformly random and g drawn from ψ_y . Therefore, by lemma 6 $H(\phi) < 2^{k^\epsilon}$ for any choice of ϵ , and the result follows. \square

To a researcher with the goal of proving $P \neq NP$ this result could seem somewhat discouraging. Indeed many were, and they turned to other areas instead. However, I conclude here in the same way as Razborov and Rudich did by highlighting a more optimistic view.

Just as the proof might be thought of as a barrier, it can also be thought of as a significant step of progress. Without the publication of this result, researchers might have wasted years and years of effort attempting to prove $P \neq NP$ in a way that was never going to succeed from the start. It is perhaps less appropriate to think of this result as a wall blocking the way than it is to think of it as a guiding force that will help the community collectively acquire a greater understanding of the problem.

References

- [Aar16] Scott Aaronson. *P =?NP*. <https://www.scottaaronson.com/papers/pnp.pdf>. 2016.
- [BGS75] Theodore Baker, John Gill, and Robert Solovay. “Relativizations of the $P =?NP$ Question”. In: *SIAM* 4.4 (1975).
- [GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. “How to Construct Random Functions”. In: *J. ACM* 33.4 (Aug. 1986), pp. 792–807. ISSN: 0004-5411. DOI: 10.1145/6490.6503. URL: <http://doi.acm.org/10.1145/6490.6503>.
- [Gow09] Timothy Gowers. *A conversation about complexity lower bounds*. <https://gowers.wordpress.com/2009/09/22/a-conversation-about-complexity-lower-bounds/>. Blog. 2009.
- [Gow13] Timothy Gowers. *Razborov and Rudich’s natural proofs argument*. <https://gowers.wordpress.com/2013/10/07/razborov-and-rudichs-natural-proofs-argument/>. Blog. 2013.
- [RR94] Alexander A. Razborov and Steven Rudich. “Natural Proofs”. In: *Proceedings of the Twenty-sixth Annual ACM Symposium on Theory of Computing*. STOC ’94. Montreal, Quebec, Canada: ACM, 1994, pp. 204–213. ISBN: 0-89791-663-8. DOI: 10.1145/195058.195134. URL: <http://doi.acm.org/10.1145/195058.195134>.
- [Sip96] Michael Sipser. *Introduction to the Theory of Computation*. 1st. International Thomson Publishing, 1996. ISBN: 053494728X.