Connections between Razborov's and Rudich's Conjecture

Jiaqi Lu

Linacre College

University of Oxford

A thesis submitted for the degree of the MSc in

Mathematics and Foundations of Computer Science

Trinity 2023

Contents

List of Figures

Abstract

The famous P versus NP question is one of the most central and difficult questions in complexity theory. Several approaches have been tried to tackle this question, including using the method of diagonalization, proving circuit lower bounds or studying proof complexity. Although none of these approaches has given the result yet, complexity theorists have discovered interesting barrier results in using the method of diagonalization and the study of circuit complexity. Unfortunately, no such barrier result has been found in the study of proof complexity. In recent years, Pich and Santhanam tried to give such a barrier result by studying the provability of Rudich's Conjecture.

In this dissertation, we focus on proof complexity with emphasis on Rudich's Conjecture and Razborov's Conjecture.

- **Pseudorandom generator.** We present results and techniques for studying pseudorandom generators, which have strong connections with proof complexity, focusing on stretching pseudorandom generators in the non-deterministic setting.
- **Rudich's Conjecture and Razborov's Conjecture.** We provide an in-detailed explanation and formalization of Rudich's and Razborov's Conjectures. This dissertation encompasses the correlations between Rudich's Conjecture and pseudorandom generators, as well as the tie between Rudich's Conjecture and natural proofs. Additionally, we incorporate the latest finding by Pich and Santhanam, which establishes a connection between Rudich's and Razborov's Conjectures within this thesis.

1. *INTRODUCTION*

Motivation. Informally, the famous P versus NP question asks whether there exists a fast algorithm to solve all the questions that have short and easily verifiable answers. Complexity theorists have explored various approaches to tackle this question. One early approach is to use the method of diagonalization and the ability of Turing machines to simulate other Turing machines. Unfortunately, Baker, Gill and Solovay [BGS75] showed that

there exists oracles A and B such that $\mathsf{P}^A = \mathsf{NP}^A$ and $\mathsf{P}^B \neq \mathsf{NP}^B$.

If a statement remains true when we replace the complexity classes it involves (*e.g.*, P, NP) with their corresponding classes relative to an oracle O, for all possible oracles O, then that statement is called a *relativizing result*. Consequently, whether $\mathsf{P} = \mathsf{NP}$ or $\mathsf{P} \neq \mathsf{NP}$, they are not relativizing results. However, this does not mean diagonalization is completely useless against P versus NP. In fact, Kozen [Koz78] showed that if $\mathsf{P} \neq \mathsf{NP}$ is provable at all, it is provable via diagonalization. The finding in [BGS75] only implies that *nonrelativizing* techniques are needed to approach P versus NP.

Therefore, complexity theorists shifted their attention to exploring other possible approaches. One exciting and fruitful approach is the study of the complexity of Boolean circuits. By employing a similar argument used in the proof of the Cook-Levin theorem [Coo71], we can observe that

All languages in P have polynomial-size circuits.

Based on this result, it becomes evident that if we can identify a specific language in NP that does not have polynomial-size circuits, we can separate NP from P. Such an approach is highly desirable.

A *combinatorial property* of Boolean functions is simply a set of Boolean functions. Consider the following proof strategy:

- Choose a property of Boolean functions.
- Demonstrate that polynomial-size circuits can only compute Boolean functions that lack this property.
- Demonstrate that a Boolean function in NP possesses such a property.

However, Razborov and Rudich [RR94] found no proof strategy like the one mentioned above can succeed. More specifically, assuming the existence of a strong pseudorandom generator, which is a standard assumption in cryptography, no combinatorial property that is "natural" can be used to prove polynomial-size lower bounds on Boolean circuits. Informally, when we refer to a combinatorial property of Boolean functions as "natural", we mean that

- *Constructivity*: It takes polynomial time to determine whether a given Boolean function has this property.
- *Largeness*: A non-negligible fraction of Boolean functions has this property.

Moreover, we say a "natural" combinatorial property is *useful* against a complexity class $\mathfrak{C}$ (*e.g.*, $\mathsf{P/poly}, \mathsf{NP/poly}$) when the Boolean circuits in $\mathfrak{C}$ can only compute Boolean functions that lack the property. Thus, Razborov and Rudich [RR94] have shown that

Assuming the existence of a strong pseudorandom generator, there is no "natural" property that is useful against $\mathsf{P/poly}$.

They showed that the existing lower bounds against weak circuit classes $\mathcal{C}$ were "natural" against $\mathcal{C}$ and proof techniques that can be viewed as "natural" cannot be used to prove strong lower bounds on circuit size against general Boolean circuits. Similar to [BGS75], natural proofs do not imply a dead end for the circuit complexity approach. Complexity theorists always search for techniques to overcome this barrier. One such recent example is hardness magnification [OS18].

Besides the circuit complexity approach, another intriguing approach that has strong connections with logic is proof complexity. In 1979, Cook and Reckhow [CR79] showed that $\mathsf{NP} = \mathsf{coNP}$ if and only if a propositional proof system Q exists, such that for any sequence of tautologies, Q can provide polynomial-size proofs of those tautologies. We refer to such a propositional proof system as *p-bounded*. Then, since we know that the complexity class P is closed under complement, this implies

$$\mathsf{P} = \mathsf{NP} \implies \mathsf{NP} = \mathsf{coNP}.$$

By studying proof complexity, we can separate NP and P if, for any propositional proof system Q, a sequence of hard tautologies exists for which Q does not have polynomial-size proofs. However, up to now, complexity theorists have yet to come up with such hard tautologies or provide a *p-bounded* propositional proof system. Additionally, unlike the first two approaches mentioned above (diagonalization and circuit complexity), in the study of proof complexity, complexity theorists haven't found barriers like [BGS75] and [RR94]. Whether there also exists a barrier in proof complexity remains an intriguing question.

We introduce some notation to more succinctly illustrate the connection between Rudich's Conjecture and Razborov's Conjecture with a potential barrier result in proof complexity. With a bit of abuse of notation, for a propositional proof system

Q and a tautology ϕ, we use $Q \nvdash_E \phi$ to denote that Q cannot efficiently prove ϕ (*i.e.*, does not have polynomial-size proof of ϕ). Given a Boolean function f, $\mathsf{clb}(f)$ is a proportional formula expressing that the Boolean function f does not have polynomial-size circuits.

Then, informally,

Rudich's Conjecture: For every propositional proof system Q, for almost all Boolean functions f, $Q \nvdash_E \mathsf{clb}(f)$.

Therefore, combined with the results in [CR79], the validity of Rudich's Conjecture directly implies $\mathsf{P} \neq \mathsf{NP}$. On the other hand, Razborov's Conjecture, proposed in [Raz15], is much more complicated to illustrate. The hardness assumption of Razborov's Conjecture is the existence of a suitable hard Boolean function. We are more interested in one of its consequences.

One of the consequences of Razborov's Conjecture under the hardness assumption: For a specific propositional proof system F, and for any Boolean function f, $F \nvdash_E \mathsf{clb}(f)$.

We can view the relationship between the consequence of Razborov's Conjecture and Rudich's Conjecture as a tradeoff between the fraction of propositional proof systems and the fraction of Boolean functions.

Purpose and Focus. This dissertation aims to explicitly present results in proof complexity, especially those related to Rudich's and Razborov's Conjectures. We will provide arguments concerning the relationship between natural proofs [RR94] and Rudich's Conjecture, as well as the study of pseudorandom generators, which are often used as assumptions in proof complexity. Finally, we will present the results obtained by Pich and Santhanam [PS19], which presents a result linking Rudich's Conjecture, Razborov's Conjecture and a barrier result for proving proof complexity lower bounds.

We give a brief and informal overview of the contents of this dissertation, along with a literature overview. The precise definitions and proofs will be presented in the subsequent chapters of this dissertation.

From Natural proofs to Rudich's Conjecture. In [RR94], Razborov and Rudich showed that, assuming the existence of a strong pseudorandom generator,

Statement 1: There is no "natural" property that is useful against $\mathsf{P/poly}$.

For a complexity class Γ, we can say a combinatorial property C of Boolean functions is Γ-natural if deciding whether a Boolean function f has this property ($f \overset{?}{\in} C$) is in Γ. In other words, the complexity class in the statement of *constructivity* is no longer required to be restricted in P. Consequently, Razborov and Rudich [RR94] showed that, assuming the existence of a strong pseudorandom generator,

Statement 2: There is no "$\mathsf{P/poly}$-natural" property that is useful against $\mathsf{P/poly}$.

Next, all the proofs eliminated by Statement 2 are proofs that give explicit procedures or algorithms in $\mathsf{P/poly}$, which can be used to decide whether a Boolean function has such a property. It is also worth considering an even stronger statement given in [Rud97].

Statement 3: There is no "$\mathsf{NP/poly}$-natural" property that is useful against $\mathsf{P/poly}$.

Statement 3 is evidently stronger than Statement 2, as it eliminates even more proofs. If a proof relies on a certificate to show whether a Boolean function has such a property, this proof will be eliminated by Statement 3.

Now, let's shift our attention to proof complexity. Since complexity theorists tend to believe in $\mathsf{P} \neq \mathsf{NP}$, the fundamental result sought in proof complexity is

Statement 4: For every propositional proof system Q, there exists a hard tautology ϕ, where $Q \nvdash_E \phi$.

As a potential candidate for a hard tautology, inspired by insights from circuit complexity research, we can consider formulas $\mathsf{clb}(f)$ for Boolean functions f such that $\mathsf{clb}(f)$ is tautological. Subsequently, we can investigate whether $\mathsf{clb}(f)$ is a hard tautology for Q. Hence, we have the following statement.

Statement 5: For every propositional proof system Q, there exists a hard Boolean function f, where $Q \nvdash_E \mathsf{clb}(f)$.

Unfortunately, at this moment, we are still unable to prove the above statement. By considering an even stronger statement and attempting to refute it, we may gain a deeper understanding of the study of proof complexity. This leads us to

Statement 6 & Rudich's Conjecture: For every propositional proof system Q, for almost all of Boolean function f, $Q \nvdash_E \mathsf{clb}(f)$.

We can show that Statement 6, which is Rudich's Conjecture, is equivalent to Statement 3. As a result, we can view Rudich's Conjecture as the intersection point where the two approaches of circuit complexity and proof complexity converge.

Pseudorandom generator and proof complexity. Both [Kra01] and [ABSRW04] have emphasized the importance of pseudorandom generators in studying proof complexity. A pseudorandom generator is considered "strong" if it is computationally hard for subexponential-size circuits to distinguish the bits generated by it from truly random bits. However, when we say a pseudorandom generator is hard for a proof system Q, we mean that Q cannot even efficiently prove the most fundamental property of a pseudorandom generator: a pseudorandom generator is not a onto function.

Complexity theorists try to convert a computationally hard pseudorandom generator to a pseudorandom generator that is hard for a proof system Q. The Nisan-Wigderson generator proposed in [NW94] is considered to be one of the candidates for such a pseudorandom generator.

Our dissertation focuses primarily on the case when a strong pseudorandom generator is the hardness assumption for natural proofs and Rudich's Conjecture. To be more precise, we concentrate on the techniques of "stretching" pseudorandom bits. Given a pseudorandom generator $g : \{0,1\}^n \to \{0,1\}^{n+1}$, if we can build a pseudorandom generator $h : \{0,1\}^n \to \{0,1\}^{N+n}$ for some $N > 1$ based on g, we say we have successfully "stretched" the pseudorandom generator g. [Yao82] and [BM84] showed how to stretch a single pseudorandom bit into many pseudorandom bits. Then, Goldreich, Goldwasser and Micali [GGM86] introduced a technique called the "hybrid argument" to construct a pseudorandom function generator from a pseudorandom generator. Those results suffice for the assumption of natural proofs.

However, for the assumption of Rudich's Conjecture, we require a strong pseudorandom generator that is strong enough against non-deterministic distinguishers. To achieve this, Rudich [Rud97] introduced the concepts of "super-bit" and "demi-bit", which are strong enough against non-deterministic distinguishers. The original techniques in [Yao82] and [GGM86] are sufficient to stretch super-bit but not demi-bit. Recently, by modifying the "hybrid argument", Tzameret and Zhang [TZ23] managed to stretch the demi-bit slightly. However, it remains an open problem whether we can stretch the demi-bit to an arbitrary length.

Newest results on Razborov's and Rudich's Conjectures. By formalizing Rudich's Conjecture itself as a propositional statement, Pich and Santhanam [PS19] investigated whether Rudich's Conjecture has a small-size propositional proof. They showed that Rudich's Conjecture itself does not "admit feasible propositional proofs". In their definitions, this does not necessarily mean that Rudich's Conjecture is the right candidate for a hard tautology. Nonetheless, they still obtained an interesting result: Assuming Rudich's Conjecture, we have at least one of the following consequences:

- for a specific propositional proof system F, and for any Boolean function f, $F \nvdash_E \mathsf{clb}(f)$, which is one of the consequences of Razborov's Conjecture under the hardness assumption.
- for every propositional proof system Q, for a fraction of Boolean functions, $Q \nvdash_E \mathsf{plb}(\mathsf{clb}(f))$.

Here, $\mathsf{plb}(\phi)$ is a propositional formula expressing that there is no polynomial-size proof of tautology ϕ against a specific strong propositional proof system. Therefore, Pich and Santhanam [PS19] demonstrated that Rudich's Conjecture implies that it is hard to prove either the circuit lower bound of a Boolean function or the proof complexity lower bound of a formula, where the proof complexity lower bound of a formula is hard to prove in the sense that it is not easy to prove a significant fraction of such formulas.

Organization. Chapter 2 will start with the necessary definitions of pseudorandom generators and present works and proofs in [Lub96] and [GGM86]. In Chapter 3, we will emphasize circuit complexity, particularly focusing on the results in [RR94]. Starting from Chapter 4, our attention will shift to proof complexity, providing basic definitions of concepts related to proof complexity and presenting general results.

Chapter 5 will give rigorous definitions of Rudich's Conjecture. As there are several different ways to formalize Rudich's Conjecture, we will also show their equivalence in this chapter. Subsequently, we will show how these hardness assumptions imply Rudich's Conjecture.

Chapter 6 will cover the mathematical definition of Razborov's Conjectures along with an explanation. In Chapter 7, our focus will be on the results and proofs in [PS19]. Finally, Chapter 8 will present some further discussion and open problems related to the topics discussed in the dissertation.

Contribution. The contribution of this dissertation is a more systematic treatment of Rudich's Conjecture and Razborov's Conjecture, with an extended discussion of the presented results together with additional details in various places. In Chapter 2, we provide the proof of theorems informally mentioned in [Rud97]. In Chapter 5, we present rigorous proof of the relation between different formalizations of Rudich's Conjecture. In Chapter 6, we provide formal mathematical definitions of Razborov's Conjecture.

Although we will present some basic definitions in the following chapters and make this dissertation self-contained, we assume the reader has some familiarity with complexity theory and logic. Readers can find useful information in [AB09] and [Kra19].

2. *PSEUDORANDOM GENERATOR*

Informally, pseudorandom generators are functions that are "easy to compute" and take a "few" random bits as input, producing "many" pseudorandom bits that are hard to be distinguished from truly random bits by "small-size" circuits. Pseudorandom generators have significant impacts and applications in both cryptography [KL07] and complexity theory [Gol10]. In this chapter, we will present some results on pseudorandom generators with an emphasis on *stretching* the number of their output bits.

2.1. *Standard pseudorandom generators*

First, we provide some necessary definitions in the area of circuit complexity, mostly taken from [AB09].

Definition 2.1. ***(Boolean circuits)****. For every $n, m \in \mathbb{N}$, an n-**input**, m-**output** **Boolean circuit** is a directed acyclic graph with n **sources** and m **sinks**. All non-source vertices are called **gates** and are labelled with one of $\vee$, $\wedge$ or $\neg$ (*i.e.*, the logical operations **OR**, **AND** and **NOT**). The vertices labelled with $\vee$ or $\wedge$ have fan-in (*i.e.*, number of incoming edges) equal to 2 and the vertices labelled with $\neg$ have fan-in 1. The **size** of such Boolean circuit C, denoted by $|C|$, is the number of gates in it. Given an input $x \in \{0,1\}^n$, the output of such Boolean circuit C is denoted by $C(x)$. The **depth** of a circuit is the length of the longest directed path from a source to the sink.*

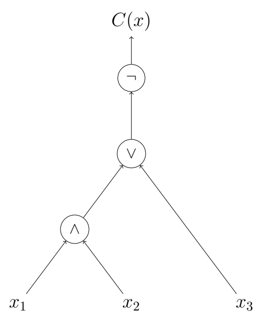

Figure 2.1: Example of a Boolean circuit

In figure 2.1, the Boolean circuit has a size of 3 and inputs $x = x_1x_2x_3$.

Definition 2.2. ***(Circuit families and language recognition)****. Let $T : \mathbb{N} \to \mathbb{N}$ be a function. A **circuit family** is a collection of circuits*[i] *$C = \{C_n\}_{n\in\mathbb{N}}$ where C_n*

[i]Sometimes, we skip "Boolean" when referring to Boolean circuits.

is a circuit with n inputs. A $T(n)$-size circuit family is a circuit family such that for every $n \in \mathbb{N}$, $|C_n| \leq T(n)$.
*We say that a language L is in **SIZE**$(T(n))$ if there exists a $T(n)$-size circuit family $\{C_n\}_{n\in\mathbb{N}}$ such that for every $n \in \mathbb{N}$ and for every $x \in \{0,1\}^n$, $x \in L$ if and only if $C_n(x) = 1$.*
*Analogously, we say that a sequence of functions $f = \{f_n\}_{n\in\mathbb{N}}$, where f_n is a Boolean function with n inputs, is in **SIZE**$(T(n))$(or computable by circuits of size $T(n)$) if there exists $T(n)$-size circuit family $\{C_n\}_{n\in\mathbb{N}}$ such that for every $n \in \mathbb{N}$ and for every $x \in \{0,1\}^n$, $f_n(x) = C_n(x)$.*

Definition 2.3. ***(The class* $\mathsf{P/poly}$*)**. $\mathsf{P/poly}$ is the class of languages that are decidable by polynomial-size circuit families. That is, $\mathsf{P/poly} = \cup_{c\in\mathbb{N}}\mathbf{SIZE}(n^c)$. We say a sequence of functions $f = \{f_n\}_{n\in\mathbb{N}}$ is in $\mathsf{P/poly}$ if there is a polynomial p such that $f \in$ **SIZE**$(p(n))$.*

Now, we give the definition of standard pseudorandom generators. We use $\in_R$ to denote choosing a uniformly random element.

Definition 2.4. ***(Standard hardness and pseudorandom generators (PRG))**. A **pseudorandom generator** (PRG) is a sequence of functions $g = \{g_n : \{0,1\}^n \rightarrow \{0,1\}^{p(n)}\}$ where p is a polynomial such that $g \in \mathsf{P/poly}$ and for any constant $c > 0$, for all large enough n, $p(n) > n$ and for any Boolean circuit C of size at most $p(n)^c$,*

$$|\mathbb{P}_{z\in_R\{0,1\}^{p(n)}}[C(z) = 1] - \mathbb{P}_{x\in_R\{0,1\}^n}[C(g_n(x)) = 1]| < \frac{1}{p(n)^c}.$$

*The **standard hardness** $H(g_n)$ of g_n is the minimal S for which there exists a Boolean circuit C of size at most S such that*

$$|\mathbb{P}_{z\in_R\{0,1\}^{p(n)}}[C(z) = 1] - \mathbb{P}_{x\in_R\{0,1\}^n}[C(g_n(x)) = 1]| \geq \frac{1}{S}$$

From Definition 2.4, a pseudorandom generator is a sequence of functions such that it is hard for polynomial-size Boolean circuits to distinguish pseudorandom strings generated by those functions from truly random strings. It's important to highlight that a pseudorandom generator is formally established as a sequence of functions. Nevertheless, there are instances where, due to a misuse of notation, a single function is also referred to as a pseudorandom generator.

Definition 2.5. ***(SPRG)**. A PRG $g = \{g_n : \{0,1\}^n \rightarrow \{0,1\}^{p(n)}\}$ is a **strong pseudorandom generator** (SPRG) if for some $\epsilon > 0$, for all sufficiently large n, $H(g_n) \geq 2^{n^\epsilon}$.*

Conjecture 1. ***(SPRG conjecture)**. SPRG exists.*

Generally, we believe in the existence of strong pseudorandom generators (SPRG). Assuming the average-case hardness[ii] of some problems in NP, such as factoring and discrete logarithm, it is possible to construct such an SPRG from those problems [Gol08].

2.2. Pseudorandom generators in the non-deterministic setting

The Boolean circuits we considered above are all deterministic circuits. However, in the non-deterministic setting, we deal with non-deterministic circuits as the distinguishers. The difference between non-deterministic circuits and deterministic circuits is analogous to the difference between non-deterministic Turing machines and deterministic Turing machines. Before introducing pseudorandom generators in the non-deterministic setting, we first provide the definition for non-deterministic circuits.

Definition 2.6. *[TZ23]* ***(Non-deterministic circuits)****. A Boolean circuit $C(x, r)$ is a* ***non-deterministic circuit*** *if we distinguish its inputs into two types: standard inputs x and nondeterministic inputs r. C is said to* ***accept*** *an input $\alpha \in \{0,1\}^{|x|}$ to x if and only if there exists an assignment $\beta \in \{0,1\}^{|r|}$ to r such that $C(\alpha, \beta) = 1$ and otherwise it is said to* ***reject*** *α. We denote the size of a non-deterministic circuit C as $|C|$, which is the number of gates in it.*

Similar to the definition of non-deterministic circuits, we can provide the definition of co-non-deterministic circuits.

Definition 2.7. *[TZ23]* ***(Co-non-deterministic circuits)****. A Boolean circuit $C(x, r)$ is a* ***co-non-deterministic circuit*** *if it contains two inputs x, r such that is said to* ***reject*** *an input $\alpha \in \{0,1\}^{|x|}$ to x if and only if there exists an assignment $\beta \in \{0,1\}^{|r|}$ to r such that $C(\alpha, \beta) = 0$ and otherwise it is said to* ***accept*** *α. We denote the size of a co-non-deterministic circuit C as $|C|$, which is the number of gates in it.*

Analogous to Definition 2.2 and 2.3, we define NP/poly and **NSIZE**.

Definition 2.8. ***(Non-deterministic circuit families and language recognition)****. We call $\{C_n\}$ a* ***non-deterministic circuit family*** *if for every n, $C_n(x, r)$ is a non-deterministic circuit with n standard input bits (*i.e.*, $|x| = n$). Let $T : \mathbb{N} \to \mathbb{N}$ be a function. A $T(n)$-size non-deterministic circuit family is a non-deterministic circuit family such that for every $n \in \mathbb{N}$, $|C_n| \leq T(n)$.*
We say that a language L is in ***NSIZE****$(T(n))$ if there exists a $T(n)$-size non-deterministic circuit family $\{C_n\}_{n\in\mathbb{N}}$ such that for every $n \in \mathbb{N}$ and for every $x \in \{0,1\}^n$, $x \in L$ if and only if there exists r such that $C_n(x, r) = 1$.*

[ii]Defined in Chapter 6

*Analogously, we say that a sequence of functions $f = \{f_n\}_{n\in\mathbb{N}}$, where $f_n : \{0,1\}^n \to \{0,1\}^{p(n)}$,is in **NSIZE**$(T(n))$ if there exists $T(n)$-size non-deterministic circuit family $\{C_n\}_{n\in\mathbb{N}}$ such that for every $n \in \mathbb{N}$ and for every $x \in \{0,1\}^n$, there exists r such that $f_n(x) = C_n(x,r)$.*

Definition 2.9. ***(The class* NP/poly*)***. NP/poly *is the class of languages that are decidable by non-deterministic polynomial-size circuit families. That is,* NP/poly $= \cup_{c\in\mathbb{N}}$***NSIZE***(n^c).

Definition 2.10. ***(Co-non-deterministic circuit families and language recognition)***. *We call $\{C_n\}$ a* ***co-non-deterministic circuit family*** *if for every n, $C_n(x,r)$ is a co-non-deterministic circuit with n standard input bits (*i.e., $|x| = n$*). Let $T : \mathbb{N} \to \mathbb{N}$ be a function. A $T(n)$-size co-non-deterministic circuit family is a non-deterministic circuit family such that for every $n \in \mathbb{N}$, $|C_n| \leq T(n)$.*
We say that a language L is in $(T(n))$ if there exists a $T(n)$-size co-non-deterministic circuit family $\{C_n\}_{n\in\mathbb{N}}$ such that for every $n \in \mathbb{N}$ and for every $x \in \{0,1\}^n$, $x \notin L$ if and only if there exists r such that $C_n(x,r) = 0$.
Analogously, we say that a function family $f = \{f_n\}_{n\in\mathbb{N}}$ is in $(T(n))$ if there exists $T(n)$-size co-non-deterministic circuit family $\{C_n\}_{n\in\mathbb{N}}$ such that for every $n \in \mathbb{N}$ and for every $x \in \{0,1\}^n$, there exists r such that $f_n(x) = C_n(x,r)$.

Definition 2.11. ***(The class* coNP/poly*)***. coNP/poly *is the class of languages that are decidable by co-non-deterministic polynomial-size circuit families. That is,* coNP/poly $= \cup_{c\in\mathbb{N}}(n^c)$.

Now, we want to generalize the setting in Definition 2.4 to the non-deterministic setting, which means a PRG that can generate pseudorandom strings that are hard to distinguish from truly random strings by polynomial-size non-deterministic circuits. However, at first glance, that seems impossible. Suppose we have a PRG $g_n : \{0,1\}^n \to \{0,1\}^N$ where $N > n$. Given an input z of length N, a non-deterministic circuit C' can always guess a seed r of length n and compute the PRG to check whether $z \overset{?}{=} g_n(r)$. C' outputs 1 if and only if $z = g_n(r)$. In other words, given input $z \in \{0,1\}^N$, C' accepts z if and only if there exists a $r \in \{0,1\}^n$ such that $z = g_n(r)$.

Therefore, $\mathbb{P}_{z\in_R\{0,1\}^N}[C'(z) = 1] \leq \frac{2^n}{2^N} \leq \frac{1}{2}$ and $\mathbb{P}_{x\in_R\{0,1\}^n}[C'(g_n(x)) = 1] = 1$. Thus,

$$|\mathbb{P}_{z\in_R\{0,1\}^N}[C'(z) = 1] - \mathbb{P}_{x\in_R\{0,1\}^n}[C'(g_n(x)) = 1]| \geq \frac{1}{2}$$

Since g_n is computable by polynomial-size circuits, C' is computable by polynomial-size non-deterministic circuits. Therefore, a polynomial-size non-deterministic circuit always exists that can distinguish pseudorandom strings from truly random strings with a success probability of at least $\frac{1}{2}$. By slightly modifying the definition of standard hardness, Rudich [Rud97] provided two non-trivial versions of non-deterministic hardness.

Definition 2.12. *[Rud97] **(Super-bit hardness)**. Let $g = \{g_n : \{0,1\}^n \to \{0,1\}^{p(n)}\}$ be a sequence of functions where p is a polynomial such that $g \in \mathsf{P/poly}$ and for sufficiently large n, $p(n) > n$. The **super-bit hardness** $H_s(g_n)$ of g_n is the minimal S for which there exists a non-deterministic circuit C of size at most S such that*

$$\mathbb{P}_{z \in_R \{0,1\}^{p(n)}}[C(z) = 1] - \mathbb{P}_{x \in_R \{0,1\}^n}[C(g_n(x)) = 1] \geq \frac{1}{S}$$

Unlike the case in standard hardness in Definition 2.4, the order of the probabilities is essential in super-bit hardness. In super-bit hardness, we require that

$$\mathbb{P}_{x \in_R \{0,1\}^n}[C(g_n(x)) = 1] < \mathbb{P}_{z \in_R \{0,1\}^{p(n)}}[C(z) = 1]$$

which eliminates the non-deterministic circuits C' we mentioned above.

It is natural to think PRG is still strong in the non-deterministic setting.

Definition 2.13. ***(Super-bit)**. Let $g = \{g_n : \{0,1\}^n \to \{0,1\}^{p(n)}\}$ be a sequence of functions where p is a polynomial such that $g \in \mathsf{P/poly}$ and for all sufficiently large n, $p(n) > n$. g is a super-bit if for some $\epsilon > 0$, for all sufficiently large n, $p(n) > n$ and $H_s(g_n) \geq 2^{n^\epsilon}$.*

Like Conjecture 1, Rudich [Rud97] gave the following conjecture.

Conjecture 2. *[Rud97] **(Super-bit Conjecture)**. There exists a super-bit.*

However, unlike Conjecture 1, which is believed to be true assuming the average-case hardness of some problems in NP, to believe the validity of Conjecture 2, we also need $\mathsf{NP} \neq \mathsf{coNP}$, which is a stronger statement than $\mathsf{P} \neq \mathsf{NP}$. Otherwise, if $\mathsf{NP} = \mathsf{coNP}$, we can obtain a distinguisher C'' by flipping the output of the non-deterministic circuit C'. Accordingly, C'' would be a polynomial-size co-non-deterministic circuit. As a result, $\mathbb{P}_{z \in_R \{0,1\}^{p(n)}}[C''(z) = 1] \geq \frac{1}{2}$ and $\mathbb{P}_{x \in_R \{0,1\}^n}[C''(g_n(x)) = 1] = 0$.

Since we assume $\mathsf{NP} = \mathsf{coNP}$, there exists a polynomial-size non-deterministic circuit D that produces the same output as C''. Consequently, $\mathbb{P}_{z \in_R \{0,1\}^{p(n)}}[D(z) = 1] \geq \frac{1}{2}$ and $\mathbb{P}_{x \in_R \{0,1\}^n}[D(g_n(x)) = 1] = 0$. Thus,

$$\mathbb{P}_{z \in_R \{0,1\}^{p(n)}}[D(z) = 1] - \mathbb{P}_{x \in_R \{0,1\}^n}[D(g_n(x)) = 1] \geq \frac{1}{2},$$

which implies the existence of a polynomial-size non-deterministic circuit that can break the super-bit.

Rudich also gave another non-deterministic hardness, which he believed to be more intuitive.

Definition 2.14. *[Rud97] **(Demi-bit hardness)**. Let $g = \{g_n : \{0,1\}^n \to \{0,1\}^{p(n)}\}$ be a sequence of functions where p is a polynomial such that $g \in \mathsf{P/poly}$ and for sufficiently large n, $p(n) > n$. The **demi-bit hardness** $H_d(g_n)$ of g_n is the minimal S for which there exists a non-deterministic circuit C of size at most S such that*

$$\mathbb{P}_{z \in_R \{0,1\}^{p(n)}}[C(z) = 1] \geq \frac{1}{S}$$

and

$$\mathbb{P}_{x\in_R\{0,1\}^n}[C(g_n(x))=1]=0$$

Definition 2.15. ***(Demi-bit)***. *Let $g=\{g_n:\{0,1\}^n\rightarrow\{0,1\}^{p(n)}\}$ be a sequence of functions where p is a polynomial such that $g\in\mathsf{P/poly}$ and for all sufficiently large n, $p(n)>n$. g is a demi-bit if for some $\epsilon>0$, for all sufficiently large n, $p(n)>n$ and $H_d(g_n)\geq 2^{n^\epsilon}$.*

Similar to Conjecture 3, we have the following conjecture.

Conjecture 3. *[Rud97]* ***(Demi-bit conjecture)***. *There exists a demi-bit.*

Now, let's consider the relationship between Conjecture 2 and Conjecture 3.

Proposition 2.1. *The Super-bit Conjecture implies the Demi-bit Conjecture.*

Proof. Suppose $g=\{g_n:\{0,1\}^n\rightarrow\{0,1\}^{p(n)}\}$ is a super-bit with super-bit hardness $H_s(g_n)=S\geq 2^{n^\epsilon}$ for some $\epsilon>0$. For all non-deterministic circuits C such that $|C|\leq S$,

$$\mathbb{P}_{z\in_R\{0,1\}^{p(n)}}[C(z)=1]-\mathbb{P}_{x\in_R\{0,1\}^n}[C(g_n(x))=1]<\frac{1}{S}.$$

Suppose g is not a demi-bit with demi-bit hardness $H_d(g_n)=S$. There exists a non-deterministic circuit C' of size $|C'|=S'<S$ such that

$$\mathbb{P}_{z\in_R\{0,1\}^{p(n)}}[C'(z)=1]\geq\frac{1}{S'}$$

and

$$\mathbb{P}_{x\in_R\{0,1\}^N}[C'(g_n(x))=1]=0$$

, which implies

$$\mathbb{P}_{z\in_R\{0,1\}^{p(n)}}[C'(z)=1]-\mathbb{P}_{x\in_R\{0,1\}^N}[C'(g_n(x))=1]\geq\frac{1}{S'}.$$

That is a contradiction. Therefore, g is a demi-bit with demi-bit hardness $H_d(g_n)\geq S=2^{n^\epsilon}$. □

2.3. Stretching super-bit

Recall the concept of *stretching*. Given a PRG $g:\{0,1\}^n\rightarrow\{0,1\}^{n+1}$, for $N>1$, if we can build a PRG $h:\{0,1\}^n\rightarrow\{0,1\}^{N+n}$ based on g, we say we have successfully "stretched" the pseudorandom generator g. A PRG g takes n truly random bits and outputs $n+1$ pseudorandom bits, which means that this PRG can generate one extra pseudorandom bit. Therefore, the concept of *stretching* means that we can generate more than one pseudorandom bit.

In this section, we will focus on the techniques of stretching super-bit. In Rudich's paper [Rud97], he mentioned that using the standard techniques of stretching standard(deterministic) PRG, we can also stretch the super-bit. Here, we provide rigorous theorems and explicit proofs.

Let s be a binary string of length n. Let I be an index set. Then, s_I denotes the sub-string, which consists of all the bits indexed in I. For example, suppose $s = 0101100101$, which is a binary string of length 10, then $s_{\{1,2,5,7,9\}} = 01100$. The following standard stretch technique is taken from [Lub96].

Standard stretch technique : Let x be a binary string of length n and $g_n : \{0,1\}^n \to \{0,1\}^{n+1}$ be a function.

1. $g_n^0(x) = x$
2. $g_n^1(x) = g_n(x)$
3. $\vdots$
4. $g_n^{i+1}(x) = \langle g_n(x)_{\{1\}}, g_n^i(g_n(x)_{\{2,3,\cdots,n+1\}})\rangle$

Theorem 2.2. *[Rud97] If for some $\epsilon > 0$, for all sufficiently large n, $g = \{g_n : \{0,1\}^n \to \{0,1\}^{n+1}\}$ is a super-bit with super-bit hardness $H_s(g_n) = S_n \geq 2^{n^\epsilon}$, using the standard stretch technique, $g^{p(n)} = \{g_n^{p(n)} : \{0,1\}^n \to \{0,1\}^{n+p(n)}\}$ is a super-bit with super-bit hardness $H_s(g_n^{p(n)}) \geq \frac{S_n}{p(n)}$ for all sufficiently large n where p is a polynomial.*

Sketch of Proof. We will use proof by contradiction to prove the above theorem. By assuming $g^{p(n)}$ does not have super-bit hardness $\frac{S}{p(n)}$, we will show that g does not have super-bit hardness S. To be more specific, by assuming $g^{p(n)}$ does not have super-bit hardness $\frac{S}{p(n)}$, there exists a non-deterministic circuit C' distinguish $g^{p(n)}$. Then, we construct a sequence of distributions ranging from truly random bits to totally pseudorandom bits. Based on this sequence of distributions, we show that there exist two adjacent distributions that we can distinguish with C'. Finally, we build a non-deterministic circuit D with C', which can be used to break g.

Proof. Clearly, given the above construction, $g_n^{p(n)}$ maps binary strings of length n to binary strings of length $n + p(n)$. Also, since p is a polynomial, by the above construction, for sufficiently large n, $g_n^{p(n)}$ is computable in polynomial-size circuits. Therefore, we can conclude that $g^{p(n)} \in \mathsf{P/poly}$. What remains is to prove the super-bit hardness $H_s(g_n^{p(n)})$ of $g_n^{p(n)}$ is greater than or equal to $\frac{S_n}{p(n)}$ for large enough n.

For sufficiently large n, suppose $H_s(g_n) = S_n \geq 2^{n^\epsilon}$ for some $\epsilon > 0$ and $H_s(g_n^{p(n)}) < \frac{S_n}{p(n)}$, then there exists a non-deterministic circuit C' such that $|C'| = S'$ where $S' < \frac{S_n}{p(n)}$ and

$$\mathbb{P}_{z\in_R\{0,1\}^{n+p(n)}}[C'(z) = 1] - \mathbb{P}_{x\in_R\{0,1\}^n}[C'(g_n^{p(n)}(x)) = 1] \geq \frac{1}{S'}.$$

In other words, this non-deterministic circuit C' can break $g_n^{p(n)}$.

Now, we define a sequence of distributions ranging from truly random bits to totally pseudorandom bits. Let $Y \in_R \{0,1\}^{p(n)}$ and $X \in_R \{0,1\}^n$. We define the following distributions.

$$\begin{aligned}
&\mathcal{D}_0 = \langle Y, X\rangle \\
&\mathcal{D}_1 = \langle Y_{\{1,2,\cdots,p(n)-1\}}, g_n^1(X)\rangle \\
&\vdots \\
&\mathcal{D}_i = \langle Y_{\{1,2,\cdots,p(n)-i\}}, g_n^i(X)\rangle \\
&\vdots \\
&\mathcal{D}_{p(n)} = g_n^{p(n)}(X)
\end{aligned}$$

Based on the above sequence of distributions, we can easily see that

$$\mathbb{P}_{z\in_R\{0,1\}^{n+p(n)}}[C'(z)=1] = \mathbb{P}_{z\in_R\mathcal{D}_0}[C'(z)=1]$$

and

$$\mathbb{P}_{x\in_R\{0,1\}^n}[C'(g_n^{p(n)}(x))=1] = \mathbb{P}_{z\in_R\mathcal{D}_{p(n)}}[C'(z)=1].$$

Then, we have the following.

$$\mathbb{P}_{z\in_R\mathcal{D}_0}[C'(z)=1] - \mathbb{P}_{z\in_R\mathcal{D}_{p(n)}}[C'(z)=1] \geq \frac{1}{S'}.$$

Therefore,

$$\begin{aligned}
&\mathbb{P}_{z\in_R\mathcal{D}_0}[C'(z)=1] - \mathbb{P}_{z\in_R\mathcal{D}_{p(n)}}[C'(z)=1] \\
=&\sum_{i=0}^{p(n)-1}(\mathbb{P}_{z\in_R\mathcal{D}_i}[C'(z)=1] - \mathbb{P}_{z\in_R\mathcal{D}_{i+1}}[C'(z)=1]) \\
\geq&\frac{1}{S'}
\end{aligned}$$

Consequently, there must exist $0 \leq i < p(n)$ such that

$$\mathbb{P}_{z\in_R\mathcal{D}_i}[C'(z)=1] - \mathbb{P}_{z\in_R\mathcal{D}_{i+1}}[C'(z)=1] \geq \frac{1}{S'\cdot p(n)}.$$

After that, we build a non-deterministic circuit D based on C' to distinguish $g_n(x)$ from truly random $n+1$ bits: Given an binary string s of length $n+1$ and $y \in \{0,1\}^{p(n)-i-1}$ as two inputs, compute $z = \langle y, s_{\{1\}}, g_n^i(s_{\{2,\cdots,n+1\}})\rangle$. At the end, output $C'(z)$.

Now, we analyze the size of D. Since g_n is computable by circuits of size polynomial to n and $i \leq p(n)$ which implies i is polynomially bounded, by the construction of g_n^i given in the standard stretch technique, g_n^i is also computable by polynomial-size circuits. Also, since $|C'| = S' < \frac{S_n}{p(n)}$ and $S_n \geq 2^{n^\epsilon}$ for some $\epsilon > 0$. Therefore, for large enough n, $|D| < S$.

Now, we explain why D can be used to distinguish $g_n(x)$ with truly random bits.

- If there exists $x \in \{0,1\}^n$ such that $g_n(x) = s$ and y is uniformly chosen from $\{0,1\}^{p(n)-i-1}$,

$$\mathbb{P}_{x \in_R \{0,1\}^n, y \in_R \{0,1\}^{p(n)-i-1}}[D(y, g_n(x)) = 1] = \mathbb{P}_{z \in \mathcal{D}_{i+1}}[C'(z) = 1]$$

since

$$\mathcal{D}_{i+1} = \langle Y_{\{1,2,\cdots,p(n)-i-1\}}, g_n^{i+1}(X)\rangle = \langle y, g_n(x)_{\{1\}}, g_n^i(g_n(x)_{\{2,3,\cdots,n+1\}})\rangle.$$

- If $s \in_R \{0,1\}^{n+1}$ and y is uniformly chosen from $\{0,1\}^{p(n)-i-1}$,

$$\mathbb{P}_{s \in_R \{0,1\}^{n+1}, y \in_R \{0,1\}^{p(n)-i-1}}[D(y, s) = 1] = \mathbb{P}_{z \in \mathcal{D}_i}[C'(z) = 1]$$

since

$$\mathcal{D}_i = \langle Y_{\{1,2,\cdots,p(n)-i\}}, g_n^i(X)\rangle = \langle y, s_{\{1\}}, g_n^i(s_{\{2,\cdots,n+1\}}\rangle.$$

Therefore, we have that

$$\begin{aligned}
&\mathbb{P}_{s \in_R \{0,1\}^{n+1}, y \in_R \{0,1\}^{p(n)-i-1}}[D(y, s) = 1] - \mathbb{P}_{x \in_R \{0,1\}^n, y \in_R \{0,1\}^{p(n)-i-1}}[D(y, g_n(x)) = 1] \\
&= \mathbb{P}_{z \in_R \mathcal{D}_i}[C'(z) = 1] - \mathbb{P}_{z \in_R \mathcal{D}_{i+1}}[C'(z) = 1] \\
&\geq \frac{1}{S' \cdot p(n)}.
\end{aligned}$$

Now, we are going to fix a $y \in \{0,1\}^{p(n)-i-1}$ to preserve the bias.

We use $y_0, \cdots y_{2^{p(n)-i-1}-1}$ to denote all the binary string of length $p(n) - i - 1$. Since y is chosen uniformly from $\{0,1\}^{p(n)-i-1}$, for any $0 \leq j \leq 2^{p(n)-i-1} - 1$, we have $\mathbb{P}_{y \in_R \{0,1\}^{p(n)-i-1}}[y = y_j] = \frac{1}{2^{p(n)-i-1}}$. Thus, we have the following.

$$\begin{aligned}
&\mathbb{P}_{s \in_R \{0,1\}^{n+1}, y \in_R \{0,1\}^{p(n)-i-1}}[D(y, s) = 1] - \mathbb{P}_{x \in_R \{0,1\}^n, y \in_R \{0,1\}^{p(n)-i-1}}[D(y, g_n(x)) = 1] \\
&= \sum_{j=0}^{2^{p(n)-i-1}-1} (\mathbb{P}_{s \in_R \{0,1\}^{n+1}, y \in_R \{0,1\}^{p(n)-i-1}}[D(y, s) = 1 | y = y_j]) \cdot \mathbb{P}_{y \in_R \{0,1\}^{p(n)-i-1}}[y = y_j] \\
&- \mathbb{P}_{x \in_R \{0,1\}^n, y \in_R \{0,1\}^{p(n)-i-1}}[D(y, g_n(x)) = 1 | y = y_j] \cdot \mathbb{P}_{y \in_R \{0,1\}^{p(n)-i-1}}[y = y_j]) \\
&\geq \frac{1}{S' \cdot p(n)}
\end{aligned}$$

Thus, there exists a j such that

$$\begin{aligned}
&(\mathbb{P}_{s \in_R \{0,1\}^{n+1}, y \in_R \{0,1\}^{p(n)-i-1}}[D(y, s) = 1 | y = y_j] \\
&- \mathbb{P}_{x \in_R \{0,1\}^n, y \in_R \{0,1\}^{p(n)-i-1}}[D(y, g_n(x)) = 1 | y = y_j]) \cdot \mathbb{P}_{y \in_R \{0,1\}^{p(n)-i-1}}[y = y_j] \\
&\geq \frac{1}{S' \cdot p(n)} \cdot \frac{1}{2^{p(n)-i-1}}
\end{aligned}$$

Therefore, we know that

$$\begin{aligned}
&\mathbb{P}_{s \in_R \{0,1\}^{n+1}, y \in_R \{0,1\}^{p(n)-i-1}}[D(y, s) = 1 | y = y_j] \\
&- \mathbb{P}_{x \in_R \{0,1\}^n, y \in_R \{0,1\}^{p(n)-i-1}}[D(y, g_n(x)) = 1 | y = y_j] \\
&\geq \frac{1}{S' \cdot p(n)}
\end{aligned}$$

Then, we fix y to be y_j in D to build a new non-deterministic circuit D' such that $D'(s) = D(y_j, s)$. From the construction, it is clear that $|D'| = |D| < S$.

$$\begin{aligned}
&\mathbb{P}_{s \in_R \{0,1\}^{n+1}}[D'(s) = 1] - \mathbb{P}_{x \in_R \{0,1\}^n}[D'(g_n(x)) = 1] \\
=&\mathbb{P}_{s \in_R \{0,1\}^{n+1}, y \in_R \{0,1\}^{p(n)-i-1}}[D(y, s) = 1 | y = y_j] \\
&- \mathbb{P}_{x \in_R \{0,1\}^n, y \in_R \{0,1\}^{p(n)-i-1}}[D(y, g_n(x)) = 1 | y = y_j] \\
\geq&\frac{1}{S' \cdot p(n)}
\end{aligned}$$

Consequently, $|D'| < S$, $S' \cdot p(n) < S$ and $H_s(g_n) \leq \max(|D'|, S' \cdot p(n))$ implies $H_s(g_n) < S$, which is a contradiction. □

As we showed above, we can stretch a super-bit to a polynomial length. Here, we demonstrate that we can also stretch a super-bit to an exponential length. The exponential stretch technique given below seems to be more powerful than the above standard stretch technique. However, one advantage of using the standard stretch technique is that the PRGs constructed using the standard stretch technique have a better hardness than the PRGs constructed using the exponential stretch technique. Also, in some circumstances, it is sufficient to stretch a PRG to have a polynomial length output.

One important thing that needs to be mentioned is that given the definition of super-bit, a super-bit is restricted to having a polynomial output length. Therefore, it is impossible to have a super-bit with exponential length output. However, we can still prove the "pseudorandomness" of a function with exponential length output, which means such a function has sufficiently large super-bit hardness.

We use $g_0(x)$ to denote the first n bits of the output of $g_n(x)$ and $g_1(x)$ to denote the last n bits of the output of $g_n(x)$. For a binary string $y = y_1y_2y_3 \cdots y_k$ of length k, we use $G_y(x)$ to denote $g_{y_k}(\cdots g_{y_2}(g_{y_1}(x)) \cdots)$.

Exponential stretch technique : Let x be a binary string of length n and $g_n : \{0,1\}^n \to \{0,1\}^{2n}$ be a function.

1. $g_n^0(x) = x$
2. $g_n^1(x) = g_n(x)$
3. $\vdots$
4. $g_n^{i+1}(x) = \langle g_n^i(g_0(x)), g_n^i(g_1(x)) \rangle$

Theorem 2.3. [Rud97] *If for some $\epsilon > 1$, for all sufficiently large n, $g = \{g_n : \{0,1\}^n \to \{0,1\}^{2n}\}$ is a super-bit with super-bit hardness $H_s(g_n) = S_n \geq 2^{p(n)^\epsilon}$, using the exponential stretch technique, $g^{p(n)} = \{g_n^{p(n)} : \{0,1\}^n \to \{0,1\}^{2^{p(n)}n}\}$ is a function with super-bit hardness $H_s(g_n^{p(n)}) \geq \frac{S_n}{2^{p(n)}}$ for all sufficiently large n where p is a polynomial.*

Since we will use binary trees in the proof, we provide some necessary definitions taken from [BW10] here.

Definition 2.16. ***(Tree)****. A* ***tree*** *is a directed and connected graph with no cycles.*

Definition 2.17. ***(Rooted binary tree)****. A* ***rooted tree*** *is a tree with one vertex designated the root. For a vertex v in the tree, the* ***parent*** *of v is the vertex connected to v on the path to the root. Except for the root, every vertex in the tree has a unique parent. A* ***child*** *of v is a vertex of which v is the parent of it. An* ***ascendant*** *of v is any vertex that is either the parent of v or is recursively an ascendant of a parent of v. A* ***descendant*** *of v is any vertex that is either a child of v or is recursively a descendant of a child of v. A* ***sibling*** *to v is any other vertex in the tree with the same parent as v. A* ***leaf*** *is a vertex with no children. An* ***internal vertex*** *is a vertex that is not a leaf. The* ***depth*** *of a vertex is the number of edges between the root and the vertex. A* ***binary tree*** *is a tree in which each node has at most two children, which are referred to as the* ***left child*** *and the* ***right child****. A* ***rooted binary tree*** *is a binary tree with one root node.*

Definition 2.18. ***(Perfect binary tree)****. A* ***perfect binary*** *tree is a binary tree where all the interior nodes in the binary tree have two children, and all leaves have the same depth.*

Here, we briefly explain the construction of the exponential stretch technique. Consider g^2.

$$\begin{aligned} g^2(x) &= \langle g^1(g_0(x)), g^1(g_1(x)) \rangle \\ &= \langle g_0(g_0(x)), g_1(g_0(x)), g_0(g_1(x)), g_1(g_1(x)) \rangle \\ &= \langle G_{00}(x), G_{01}(x), G_{10}(x), G_{11}(x) \rangle \end{aligned}$$

Let's consider a perfect rooted binary tree T in figure 2.2 given below.

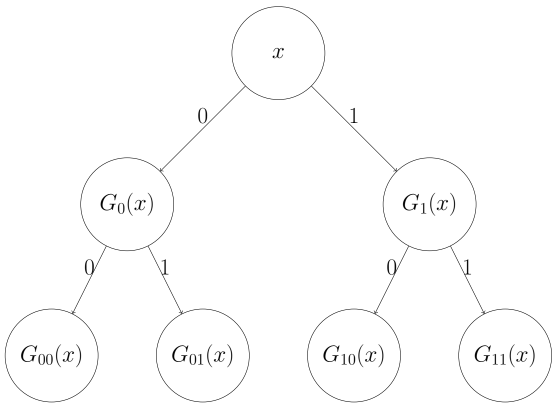

Figure 2.2: Example of a perfect rooted binary tree T

Accordingly, we can view the function $g^2(x)$ as getting input x in the root of the perfect rooted binary tree and computing along the paths of the tree to all the leaves.

The output of $g^2(x)$ is a string of length $4n$ consisting of all the n bits binary strings contained in the leaves of the tree.

Now, we provide the proof of the Theorem 2.3.

Sketch of Proof. We will use proof by contradiction to prove the above theorem. By assuming $g^{p(n)}$ does not have super-bit hardness $\frac{S}{2^{p(n)}}$, we will show that g does not have super-bit hardness S. To be more specific, by assuming $g^{p(n)}$ does not have super-bit hardness $\frac{S}{2^{p(n)}}$, there exists a non-deterministic circuit C' distinguish $g^{p(n)}$. Then, we also construct a sequence of distributions ranging from truly random bits to totally pseudorandom bits like what we did in the proof of Theorem 2.2. Specifically, We construct a sequence of sub-trees and use those sub-trees to construct a sequence of functions. We can view the outputs of those functions as distributions. After that, we can distinguish two adjacent distributions with C'. Eventually, we build a non-deterministic circuit D with C', which can be used to break g.

Proof. For the simplicity of notation, we use g to denote g_n and S to denote S_n. Suppose $g^{p(n)}$ does not have super-bit hardness $H_s(g^{p(n)}) = \frac{S}{2^{p(n)}}$, which means there exists a non-deterministic circuit C' of size $|C'| = S' < \frac{S}{2^{p(n)}}$ such that

$$\mathbb{P}_{z\in_R\{0,1\}^{2^{p(n)}n}}[C'(z) = 1] - \mathbb{P}_{x\in_R\{0,1\}^n}[C'(g^{p(n)}(x)) = 1] \geq \frac{1}{S'}.$$

Similar to the case of $g^2(x)$, for the function $g^{p(n)}(x)$, we construct the following perfect rooted binary tree T' in figure 2.3.

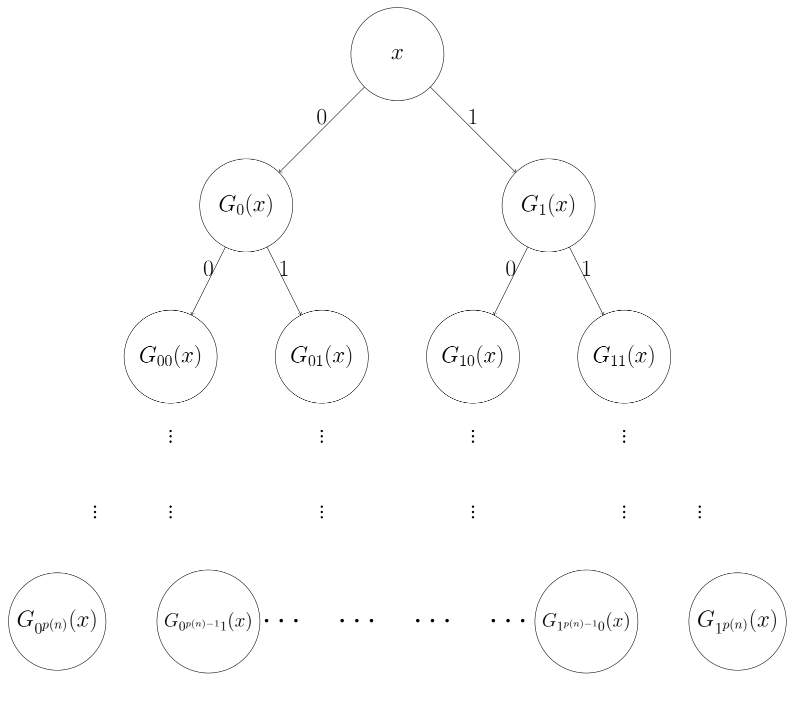

Figure 2.3: The perfect rooted binary tree T' for $g^{p(n)}(x)$

We use $g_0(x)$ to denote the first n bits of the output of $g_n(x)$ and $g_1(x)$ to denote the last n bits of the output of $g_n(x)$. For a binary string $y = y_1y_2y_3\cdots y_k$ of length k, we use $G_y(x)$ to denote $g_{y_k}(\cdots g_{y_2}(g_{y_1}(x))\cdots)$.

Each node in T' contains a binary string of length n. Given a node v, we use s_v to represent the binary string in v. Suppose c_0 is the left child of a node $v \in T'$, and c_1 is the right child of the same node v. We restrict that $s_{c_0} = G_0(s_v)$ and $s_{c_1} = G_1(s_v)$. In other words, the binary string in a node's left child is obtained by applying G_0 on the binary string within that node. Similarly, the binary string in the right child of a node is derived by applying G_1 to the binary string in that node.

Now, we consider a sequence of all the internal nodes in T':

$$v_1, v_2, \cdots, v_{2^{p(n)}-1}$$

where v_1 is the root of T' and v_j is a child of v_i for some $i < j$. We denote this sequence to be OUT. With this sequence OUT, we consider the following sequence of sub-trees of T':

1. T_1 takes the first element of OUT, which is the root of T'.
2. T_2 is a sub-tree of T', which consists of the first 2 nodes in OUT. (The root of T' and one child of it).
3. $\vdots$
4. T_k is a sub-tree of T', which consists of the first k nodes in OUT.
5. $\vdots$
6. $T_{2^{p(n)}-1}$ is the sub-tree of T', consisting of all the internal nodes of T'.

Now, based on the sequence of sub-trees T_k, we construct another sequence of sub-trees T'_K.

1. T'_0 is the root of T'.
2. T'_1 is a sub-tree of T', which consists of T_1 and all children of all leaves in T_1.
3. $\vdots$
4. T'_k is a sub-tree of T', which consists of T_k and all children of all leaves in T_k.
5. $\vdots$
6. $T'_{2^{p(n)}-1}$ is a sub-tree of T', which consists of $T_{2^{p(n)}-1}$ and all children of all leaves in $T_{2^{p(n)-1}}$, which is same as T'.

Now, we consider the following sequence of functions defined based on the sequence of sub-trees we give above: for each integer $0 \leq k \leq 2^{p(n)} - 1$, f_k uniformly chooses a binary string of length n in $\{0,1\}^n$ for each leaf in T'_k, then computes all the descendant in T' of those leaves and outputs all the binary strings in the leaves of T'.

For example, the figure 2.4 gives an example of sub-trees T_3 and T'_3 of T'. The red

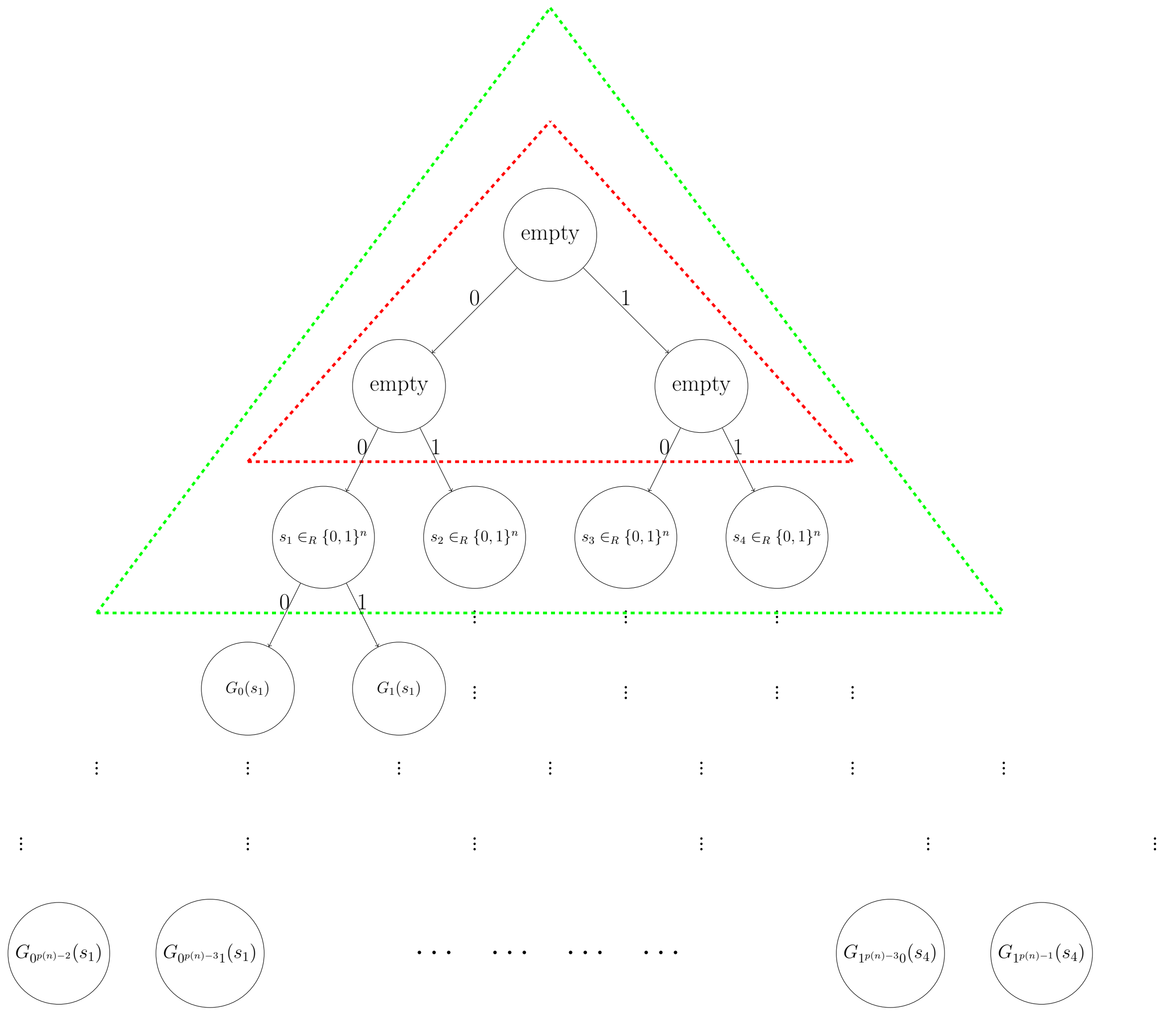

Figure 2.4: Example of sub-trees T_3 and T'_3 of T'

triangle in the figure 2.4 points out the sub-tree T_3. And T'_3 is surrounded by the green triangle. Each leaf in T'_3 uniformly chooses a binary string of length n and computes all the descendants. Now, we can observe that the distribution of uniformly choosing a binary string $z \in_R \{0,1\}^{2^{p(n)}n}$ is equivalent to the distribution of the output of $f_{2^{p(n)}-1}$, and the distribution of uniformly choosing a binary string $x \in_R \{0,1\}^n$ and compute $g^{p(n)}(x)$ is equivalent to the distribution of the output of f_0.

We use $\mathbb{P}[C'(f_k) = 1]$ to denote that the probability of C' outputs 1 on elements that are uniformly chosen from the distribution of the output of f_k.

Thus, we have the following equivalences.

$$\mathbb{P}_{z \in_R \{0,1\}^{2^{p(n)}n}}[C'(z) = 1] = \mathbb{P}[C'(f_{2^{p(n)}-1}) = 1]$$
$$\mathbb{P}_{x \in_R \{0,1\}^n}[C'(g^{p(n)}(x)) = 1] = \mathbb{P}[C'(f_0) = 1]$$

Therefore, we know that there must exist an integer $0 \leq k \leq 2^{p(n)} - 2$ such that

$$\mathbb{P}[C'(f_{k+1}) = 1] - \mathbb{P}[C'(f_k) = 1] \geq \frac{1}{S' \cdot 2^{p(n)}}.$$

By the construction of the sequence of sub-trees $\{T_k\}$, T_{k+1} must contain one more node than T_k, which we denote as v. We denote the left child of v as a and the right child of v as b. There are two easy and necessary claims:

1. **Both a and b are not included in T_{k+1}.** Otherwise, a or b is v_i for some $i < k+1$, which is a contradiction.

2. **v is a child of a leaf in T_k by definition.**

According to the second item above, v is a leaf in the sub-tree T'_k. Because a and b are two children of v, v is contained in T_{k+1} and the first item above, a and b are two leaves in the sub-tree T'_{k+1}. Suppose we denote the set of leaves in T'_k as L_k. By deleting v from L_k and adding a and b into L_k, we get the set of leaves L_{k+1} in T'_{k+1}. Specifically, $L_k \cup \{a, b\} = L_{k+1} \cup \{v\}$.

Now, we can construct a non-deterministic circuit D to distinguish truly random strings and pseudorandom strings of length $2n$. Given a binary string s of length $2n$, D computes f_k with the first n bits of s put into a and the last n bits of s put into b (denote the output as $f_k(s)$). All the leaves except a and b in T'_k uniformly choose a binary string of length n. Finally, D outputs $C'(f_k(s))$.

- If $s \in_R \{0,1\}^{2n}$, the distribution of $f_k(s)$ is the same as the distribution of f_{k+1}, which implies $\mathbb{P}_{s \in_R \{0,1\}^{2n}}[D(s) = 1] = \mathbb{P}[C'(f_{k+1}) = 1]$.

- If there exists a $x \in \{0,1\}^n$ such that $g(x) = s$, the distribution of $f_k(s)$ is the same as the distribution of f_k where $x \in_R \{0,1\}^n$, which implies $\mathbb{P}_{x \in_R \{0,1\}^n}[D(g(x)) = 1] = \mathbb{P}[C'(f_k) = 1]$.

Because of that, we get the following.

$$\begin{aligned}
&\mathbb{P}_{s \in_R \{0,1\}^{2n}}[D(s) = 1] - \mathbb{P}_{x \in_R \{0,1\}^n}[D(g(x)) = 1] \\
=&\mathbb{P}[C'(f_{k+1})] - \mathbb{P}[C'(f_k)] \\
>&\frac{1}{S' \cdot 2^{p(n)}}
\end{aligned}$$

Same as the argument in the proof of Theorem 2.2, we can fix the binary strings in all the leaves except a and b in $f_k(s)$ and preserve the bias. We denote the new non-deterministic circuit with partially fixed inputs as D'. Therefore,

$$\begin{aligned}
&\mathbb{P}_{s \in_R \{0,1\}^{2n}}[D'(s) = 1] - \mathbb{P}_{x \in_R \{0,1\}^n}[D'(g(x)) = 1] \\
=&\mathbb{P}_{s \in_R \{0,1\}^{2n}}[D(s) = 1] - \mathbb{P}_{x \in_R \{0,1\}^n}[D(g(x)) = 1] \\
>&\frac{1}{S' \cdot 2^{p(n)}}
\end{aligned}$$

Now, let's analyze the size of the non-deterministic circuit D'. Since g is computable by polynomial-size circuits, f_k needs circuits of size at most $2^{p(n)} \cdot \mathsf{poly}(n)$ to compute. Also, since $|C'| = S' < \frac{S}{2^{p(n)}}$ and $S \geq 2^{p(n)^\epsilon}$ for some $\epsilon > 1$, we know that for large enough n, $|D'| < S$.

Therefore, $|D'| < S$, $S' \cdot 2^{p(n)} < S$ and $H_s(g) \leq \max(|D'|, S' \cdot 2^{p(n)})$ implies $H_s(g) < S$, which is a contradiction. □

Definition 2.19. ***(Pseudorandom function).*** *A* ***pseudorandom function*** *is a sequence of functions $f(x,y) = \{f_n(x,y) : \{0,1\}^n \times \{0,1\}^{p(n)} \to \{0,1\}\}$ where p is a polynomial such that for any constant $c > 0$, for sufficiently large n, for any Boolean circuit C of size at most $p(n)^c$,*

$$|\mathbb{P}_{s \in_R \{0,1\}^{2^{p(n)}}}[C(s) = 1] - \mathbb{P}_{x \in_R \{0,1\}^n}[C(f_{n,x}) = 1]| < \frac{1}{p(n)^c},$$

where $f_{n,x}(y) = f_n(x,y)$ and $f_{n,x}$ is the truth table[iii] *of $f_{n,x}(\cdot)$. Also, $f_n,x(\cdot)$ is required to be computable in circuits of size polynomial to n.*

Now, we show that we can construct pseudorandom functions based on a super-bit $g = \{g_n : \{0,1\}^n \to \{0,1\}^{2n}\}$. As we defined above, we use $g_0(x)$ to denote the first n bits of the output of $g_n(x)$ and $g_1(x)$ to denote the last n bits of the output of $g_n(x)$. For a binary string $y = y_1 y_2 y_3 \cdots y_k$ of length k, we use $G_y(x)$ to denote $g_{y_k}(\cdots g_{y_2}(g_{y_1}(x)) \cdots)$. Given a binary string y of length $p(n)$, we define $f_x(y) : \{0,1\}^{p(n)} \to \{0,1\}$ as the first bit of $G_y(x)$. The truth table of f_x is a binary string of length $2^{p(n)}$. We define $F(x)$ as the bit sequence that gives the truth table of f_x. When p is a polynomial, $f_x(\cdot)$ is computable by circuits of size polynomial to n.

Corollary 2.4. *[Rud97] If for some $\epsilon > 1$ and for all sufficiently large n, $g = \{g_n : \{0,1\}^n \to \{0,1\}^{2n}\}$ is a super-bit with super-bit hardness $H_s(g_n) = S_n \geq 2^{p(n)^\epsilon}$, then F is a function with super-bit hardness $H_s(F) \geq \frac{S_n}{2^{p(n)}}$ for all sufficiently large n.*

Proof. For the simplicity of notation, we use g to denote g_n and S to denote S_n. Suppose F does not have super-bit hardness $\frac{S}{2^{p(n)}}$. There exists a non-deterministic circuit C' of size $|C'| = S' < \frac{S}{2^{p(n)}}$ and

$$\mathbb{P}_{z \in_R \{0,1\}^{2^{p(n)}}}[C'(z) = 1] - \mathbb{P}_{x \in_R \{0,1\}^n}[C'(F(x)) = 1] > \frac{1}{S'}.$$

Then, we use $g^{p(n)}$ to denote the function given by the construction in the exponential stretch technique. Now, we use C' to construct a distinguisher D for $g^{p(n)}$. For any

[iii]For a Boolean function $f : \{0,1\}^n \to \{0,1\}$, there are 2^n many different inputs. We can order them in a lexicographical order, where the first one is the all-zero string 0^n and the last one is the all-one string 1^n. For the truth table of a Boolean function $f : \{0,1\}^n \to \{0,1\}$, we mean the binary string of length 2^n, where the ith bit of it is the output of f on the lexicographical ith input.

binary string s of length $2^{p(n)}n$, D separates s into $2^{p(n)}$ segments of length n and combines all the first bits of those segments into a new binary string s' of length $2^{p(n)}$. Finally, D outputs $C'(s')$.

- If $s \in_R \{0,1\}^{2^{p(n)}n}$, the distribution of the string s' is identical to the uniform distribution of binary strings of length $2^{p(n)}$. This fact leads to $\mathbb{P}_{z\in_R\{0,1\}^{2^{p(n)}}}[C'(z) = 1] = \mathbb{P}_{s\in_R\{0,1\}^{2^{p(n)}n}}[D(s) = 1]$.
- If there exists a $x \in \{0,1\}^n$ such that $s = g^{p(n)}(x)$, the distribution of the string s' is identical to the distribution of $F(x)$ where $x \in_R \{0,1\}^n$. This implies $\mathbb{P}_{x\in_R\{0,1\}^n}[D(g^{p(n)}(x)) = 1] = \mathbb{P}_{x\in_R\{0,1\}^n}[C'(F(x)) = 1]$.

Hence, we have the following.

$$\begin{aligned}
&\mathbb{P}_{s\in_R\{0,1\}^{2^{p(n)}n}}[D(s) = 1] - \mathbb{P}_{x\in_R\{0,1\}^n}[D(g^{p(n)}(x)) = 1]\\
=&\mathbb{P}_{z\in_R\{0,1\}^{2^{p(n)}}}[C'(z) = 1] - \mathbb{P}_{x\in_R\{0,1\}^n}[C'(F(x)) = 1]\\
>&\frac{1}{S'}
\end{aligned}$$

We can observe that the size of D is the same as C', which means $|D| < S'$. Therefore, $H_s(g^{p(n)}) \leq S' < \frac{S}{2^{p(n)}}$, which is a contradiction from Theorem 2.3. □

In conclusion, if $x \in_R \{0,1\}^n$, $F(x)$ can be viewed as a function whose outputs are truth tables of pseudorandom Boolean functions with input length $p(n)$. Those pseudorandom Boolean functions are computationally hard to distinguish from Boolean functions, which are chosen randomly from $\mathcal{F}_{p(m)}$ where $\mathcal{F}_j$ denotes the set of all Boolean functions with input length j.

2.4. Stretching demi-bit

Unlike super-bit, as mentioned in [Rud97], demi-bit cannot be stretched by the standard techniques given in [Lub96] and [GGM86]. Rudich gave the following open problem in [Rud97]. This open problem has not been answered for over 20 years. However, recently, Tzameret and Zhang [TZ23] showed that super-bit is stretchable. Although their technique can only stretch demi-bit for sub-linear length, this result still offers a glimpse of the solutions to the problem.

Open Problem If you have one demi-bit, can you stretch it to polynomially many demi-bits?

Now, we give the stretching algorithm used to stretch demi-bit.

Demi-bit Stretching Algorithm.[TZ23] Suppose $g = \{g_n : \{0,1\}^n \to \{0,1\}^{n+1}\}$ is a demi-bit and $0 < c < 1$ is a constant. Then, $G = \{G_N : \{0,1\}^N \to \{0,1\}^{N+m}\}$ where $m = \lceil N^c \rceil$ is defined as follows: given an binary string $x = x_1x_2\cdots x_m r$ of length N, $G_N(x) = g_n(x_1)\cdots g_n(x_m)r$ where $n = \lfloor \frac{N}{m} \rfloor$ and each x_i has length n.

Theorem 2.5. *[TZ23] If for all sufficiently large n, $g = \{g_n : \{0,1\}^n \to \{0,1\}^{n+1}\}$ is a demi-bit with demi-bit hardness $H_d(g_n) = S$, $G = \{G_N : \{0,1\}^N \to \{0,1\}^{N+m}\}$, as defined in the Demi-bit Stretching Algorithm, is a demi-bit with demi-bit hardness $H_d(G_N) \geq \frac{S}{m}$ for all sufficiently large N.*

Proof. It is obvious that the output length of G_N is greater than the input length of G_N. Also, since g_n is computable by circuits of size $p(n+1)$ for a polynomial p, G_N is computable by circuits of size $\mathsf{poly}(n) \cdot \lceil \frac{N}{n} \rceil$ which is polynomially bounded by N.

Suppose G_N does not have demi-bit hardness $\frac{S}{m}$, which implies there exists a non-deterministic circuit C' of size $|C'| = S' < \frac{S}{m}$ such that

$$\mathbb{P}_{z \in_R \{0,1\}^{N+m}}[C'(z) = 1] \geq \frac{1}{S'}$$

and

$$\mathbb{P}_{x \in_R \{0,1\}^N}[C'(G_N(x)) = 1] = 0$$

Without loss of generality, we can assume that m divides M, which gives the following.

$$\begin{aligned}
&\mathbb{P}_{z \in_R \{0,1\}^{N+m}}[C'(z) = 1] \\
=&\mathbb{P}_{z_1, \cdots, z_m \in_R \{0,1\}^{n+1}}[C'(z_1 \cdots z_m) = 1] \\
\geq&\frac{1}{S'}
\end{aligned}$$

where each z_i is uniformly chosen from $\{0,1\}^{n+1}$. Also,

$$\begin{aligned}
&\mathbb{P}_{x \in_R \{0,1\}^N}[C'(G_N(x)) = 1] \\
=&\mathbb{P}_{x_1, \cdots, x_m \in_R \{0,1\}^n}[C'(g_n(x_1) \cdots g_n(x_m)) = 1] \\
=&0
\end{aligned}$$

where each x_i is uniformly chosen from $\{0,1\}^n$.

Now, we define a non-deterministic circuit D as follows: given inputs $z = z_1 \cdots z_m$ where each z_j is a binary string of length $n+1$ and a integer $0 \leq i \leq m$, $D(z_1 \cdots z_m, i) = 1$ if and only if there exists binary strings $x_1, \cdots, x_i \in \{0,1\}^n$ such that $C'(g_n(x_1) \cdots g_n(x_i) z_{i+1} \cdots z_m) = 1$.

Then, we can observe that

$$\mathbb{P}_{z_1, \cdots, z_m \in_R \{0,1\}^{n+1}}[D(z_1 \cdots z_m, 0) = 1] = \mathbb{P}_{z_1, \cdots, z_m \in_R \{0,1\}^{n+1}}[C'(z_1 \cdots z_m) = 1] \geq \frac{1}{S'}$$

and

$$\begin{aligned}
&\mathbb{P}_{z_1, \cdots, z_m \in_R \{0,1\}^{n+1}}[D(z_1 \cdots z_m, m) = 1] \\
=&\mathbb{P}_{x_1, \cdots, x_m \in_R \{0,1\}^n}[D(g_n(x_1) \cdots g_n(x_m), m) = 1] \\
=&0 \\
=&\mathbb{P}_{x_1, \cdots, x_m \in_R \{0,1\}^n}[C'(g_n(x_1) \cdots g_n(x_m))]
\end{aligned}$$

Otherwise, if there exists $a_1, \cdots, a_m \in \{0,1\}^n$ such that $D(g_n(a_1)\cdots g_n(a_m), m) = C'(g_n(a_1)\cdots g_n(a_m)) = 1$, then $\mathbb{P}_{x_1,\cdots,x_m \in_R \{0,1\}^n}[C'(g_n(x_1)\cdots g_n(x_m)) = 1] > 0$, which is a contradiction.

Also, since D can guess $x_1, \cdots, x_i$ with a trivial non-deterministic circuit and do whatever C' does, $|D| = |C'| < \frac{S}{m} < S$.

From the above equations, we have the following.

$$\begin{aligned}
&\mathbb{P}_{z_1,\cdots,z_m \in_R \{0,1\}^{n=1}}[C'(z_1\cdots z_m) = 1] - \mathbb{P}_{x_1,\cdots,x_m \in_R \{0,1\}^n}[C'(g_n(x_1)\cdots g_n(x_m))] \\
=&\mathbb{P}_{z_1,\cdots,z_m \in_R \{0,1\}^{n+1}}[D(z_1\cdots z_m, 0) = 1] - \mathbb{P}_{x_1,\cdots,x_m \in_R \{0,1\}^n}[D(g_n(x_1)\cdots g_n(x_m), m) = 1] \\
=&\sum_{i=0}^{m-1}(\mathbb{P}_{x_1,\cdots,x_i \in_R \{0,1\}^n, z_{i+1},\cdots,z_m \in_R \{0,1\}^{n+1}}[D(g_n(x_1)\cdots g_n(x_i) z_{i+1}\cdots z_m, i) = 1] \\
&- \mathbb{P}_{x_1,\cdots,x_{i+1} \in_R \{0,1\}^n, z_{i+2},\cdots,z_m \in_R \{0,1\}^{n+1}}[D(g_n(x_1)\cdots g_n(x_{i+1}) z_{i+2}\cdots z_m, i+1) = 1]) \\
\geq&\frac{1}{S'}
\end{aligned}$$

Therefore, there exists an integer $0 \leq i \leq m-1$ such that

$$\begin{aligned}
&\mathbb{P}_{x_1,\cdots,x_i \in_R \{0,1\}^n, z_{i+1},\cdots,z_m \in_R \{0,1\}^{n+1}}[D(g_n(x_1)\cdots g_n(x_i) z_{i+1}\cdots z_m, i) = 1]- \\
&\mathbb{P}_{x_1,\cdots,x_{i+1} \in_R \{0,1\}^n, z_{i+2},\cdots,z_m \in_R \{0,1\}^{n+1}}[D(g_n(x_1)\cdots g_n(x_{i+1}) z_{i+2}\cdots z_m, i+1) = 1] \\
&\geq \frac{1}{S' \cdot m}
\end{aligned}$$

Now, we use $S = \{0,1\}^{(n+1)\cdot(m-i-1)}$ to denote the set of $(m-i-1)$ tuples of binary strings of length $n+1$. Then, we define

$$S_1 = \{(z_{i+2}, \cdots, z_m) \in S | \exists\, x_1, \cdots, x_{i+1} \in \{0,1\}^n\ C'(g_n(x_1)\cdots g_n(x_{i+1}) z_{i+2}\cdots z_m) = 1\}$$

and

$$S_2 = \{(z_{i+2}, \cdots, z_m) \in S | (z_{i+2}, \cdots, z_m) \notin S_1\}$$

Thus, we can observe that

$$\begin{aligned}
&D(g_n(x_1)\cdots g_n(x_{i+1}) z_{i+2}\cdots z_m, i+1) = 1 \\
\Leftrightarrow&\exists\ x_1, \cdots, x_{i+1} \in \{0,1\}^n\ C'(g_n(x_1)\cdots g_n(x_{i+1}) z_{i+2}\cdots z_m) = 1 \\
\Leftrightarrow&(z_{i+2}, \ldots, z_m) \in S_1.
\end{aligned}$$

The above result implies that

$$\begin{aligned}
&\mathbb{P}_{z_{i+2},\cdots,z_m \in_R \{0,1\}^{n+1}}[(z_{i+2}, \cdots, z_m) \in S_1] \\
=&\mathbb{P}_{x_1,\cdots,x_{i+1} \in_R \{0,1\}^n, z_{i+2},\cdots,z_m \in_R \{0,1\}^{n+1}}[D(g_n(x_1)\cdots g_n(x_{i+1}) z_{i+2}\cdots z_m, i+1) = 1]
\end{aligned}$$

Also, $D(g_n(x_1)\cdots g_n(x_i) z_{i+1}\cdots z_m, i) = 1$ if and only if $D(A z_{i+1}\cdots z_m, i) = 1$ where A is an arbitrary binary string of length $n \cdot i$ since D with the second input i only cares about the last $(n+1)\cdot(m-i)$ bits of the first input.

Consequently, we have the following.

$$\begin{aligned}
&\mathbb{P}_{x_1,\cdots,x_i\in_R\{0,1\}^n,z_{i+1},\cdots,z_m\in_R\{0,1\}^{n+1}}[D(g_n(x_1)\cdots g_n(x_i)z_{i+1}\cdots z_m,i)=1]\\
=&\mathbb{P}_{x_1,\cdots,x_i\in_R\{0,1\}^n,z_{i+1},\cdots,z_m\in_R\{0,1\}^{n+1}}[D(g_n(x_1)\cdots g_n(x_i)z_{i+1}\cdots z_m,i)=1|(z_{i+2},\cdots,z_m)\in S_1]\cdot\\
&\mathbb{P}_{z_{i+2},\cdots,z_m\in_R\{0,1\}^{n+1}}[(z_{i+2},\cdots,z_m)\in S_1]+\\
&\mathbb{P}_{x_1,\cdots,x_i\in_R\{0,1\}^n,z_{i+1},\cdots,z_m\in_R\{0,1\}^{n+1}}[D(g_n(x_1)\cdots g_n(x_i)z_{i+1}\cdots z_m,i)=1|(z_{i+2},\cdots,z_m)\in S_2]\cdot\\
&\mathbb{P}_{z_{i+2},\cdots,z_m\in_R\{0,1\}^{n+1}}[(z_{i+2},\cdots,z_m)\in S_2]\\
\le&1\cdot\mathbb{P}_{z_{i+2},\cdots,z_m\in_R\{0,1\}^{n+1}}[(z_{i+2},\cdots,z_m)\in S_1]+\\
&\mathbb{P}_{z_{i+1},\cdots,z_m\in_R\{0,1\}^{n+1}}[D(Az_{i+1}\cdots z_m,i)=1|(z_{i+2},\cdots,z_m)\in S_2]\cdot\\
&\mathbb{P}_{z_{i+2},\cdots,z_m\in_R\{0,1\}^{n+1}}[(z_{i+2},\cdots,z_m)\in S_2]\\
=&\mathbb{P}_{x_1,\cdots,x_{i+1}\in_R\{0,1\}^n,z_{i+2},\cdots,z_m\in_R\{0,1\}^{n+1}}[D(g_n(x_1)\cdots g_n(x_{i+1})z_{i+2}\cdots z_m,i+1)=1]+\\
&\mathbb{P}_{z_{i+1},\cdots,z_m\in_R\{0,1\}^{n+1}}[D(Az_{i+1}\cdots z_m,i)=1|(z_{i+2},\cdots,z_m)\in S_2]\cdot\\
&\mathbb{P}_{z_{i+2},\cdots,z_m\in_R\{0,1\}^{n+1}}[(z_{i+2},\cdots,z_m)\in S_2]
\end{aligned}$$

Therefore, we know that

$$\begin{aligned}
&\mathbb{P}_{z_{i+1},\cdots,z_m\in_R\{0,1\}^{n+1}}[D(Az_{i+1}\cdots z_m,i)=1|(z_{i+2},\cdots,z_m)\in S_2]\cdot\\
&\mathbb{P}_{z_{i+2},\cdots,z_m\in_R\{0,1\}^{n+1}}[(z_{i+2},\cdots,z_m)\in S_2]\\
\ge&\mathbb{P}_{x_1,\cdots,x_i\in_R\{0,1\}^n,z_{i+1},\cdots,z_m\in_R\{0,1\}^{n+1}}[D(g_n(x_1)\cdots g_n(x_i)z_{i+1}\cdots z_m,i)=1]-\\
&\mathbb{P}_{x_1,\cdots,x_{i+1}\in_R\{0,1\}^n,z_{i+2},\cdots,z_m\in_R\{0,1\}^{n+1}}[D(g_n(x_1)\cdots g_n(x_{i+1})z_{i+2}\cdots z_m,i+1)=1]\\
&\ge\frac{1}{S'\cdot m}
\end{aligned}$$

As a result, we have that

$$\mathbb{P}_{z_{i+2},\cdots,z_m\in_R\{0,1\}^{n+1}}[(z_{i+2},\cdots,z_m)\in S_2]>0$$

and

$$\mathbb{P}_{z_{i+1},\cdots,z_m\in_R\{0,1\}^{n+1}}[D(Az_{i+1}\cdots z_m,i)=1|(z_{i+2},\cdots,z_m)\in S_2]\ge\frac{1}{S'\cdot m}$$

Similar to the argument we give in the proof of Theorem 2.2, we can find a tuple $(z'_{i+2},\cdots,z'_m)\in S_2$ preserving the bias such that

$$\begin{aligned}
&\mathbb{P}_{z_{i+1}\in_R\{0,1\}^{n+1}}[D(Az_{i+1}z'_{i+2}\cdots z'_m,i)=1]\\
=&\mathbb{P}_{z_{i+1},\cdots,z_m\in_R\{0,1\}^{n+1}}[D(Az_{i+1}\cdots z_m,i)=1|(z_{i+2},\cdots,z_m)\in S_2]\\
\ge&\frac{1}{S'\cdot m}
\end{aligned}$$

Now, we build a non-deterministic circuit D' based on D to break g_n. Given a binary string $z\in\{0,1\}^{n+1}$ as input, D' constructs the string $(Azz'_{i+2},\cdots,z'_m)$ first and outputs $D(Azz'_{i+2},\cdots,z'_m,i)$. We can easily observe that $|D'|=|D|<S$

Therefore,

- if $z \in_R \{0,1\}^{n+1}$,

$$\mathbb{P}_{z \in_R \{0,1\}^{n+1}}[D'(z) = 1] = \mathbb{P}_{z_{i+1} \in_R \{0,1\}^{n+1}}[D(Az_{i+1}z'_{i+2} \cdots z'_m, i) = 1] \geq \frac{1}{S' \cdot m}$$

- if there exists a $x \in \{0,1\}^n$ such that $z = g_n(x)$, then

$$\begin{aligned}
&\mathbb{P}_{x \in_R \{0,1\}^n}[D'(g_n(x)) = 1] \\
=&\mathbb{P}_{x \in_R \{0,1\}^n}[D(Ag_n(x)z'_{i+2} \cdots z'_m, i) = 1] \\
=&\mathbb{P}_{x \in_R \{0,1\}^n}[\exists\ x_1, \cdots, x_i \in \{0,1\}^n\ C'(g_n(x_1) \cdots g_n(x_i)g_n(x)z'_{i+2} \cdots z'_m) = 1] \\
=&0
\end{aligned}$$

  because $(z'_{i+2} \cdots z'_m) \in S_2$ which means there does not exist $x_1, \cdots, x_{i+1} \in \{0,1\}^n$ such that $C'[g_n(x_1) \cdots g_n(x_i)g_n(x_{i+1})z'_{i+2} \cdots z'_m) = 1]$. No matter how we choose x, $D'(g_n(x)) = 0$.

Hence, D' can be used to distinguish $g_n(x)$ from truly random bits of length $n+1$. To conclude, we have that $|D'| < S$, $S' \cdot m < S$ and $H_d(g_n) \leq \max(|D'|, S' \cdot m)$, which implies $H_d(g_n) < S$. This is a contradiction. □

From the construction in the Demi-bit Stretching Algorithm, we know that this construction can only be used to stretch demi-bit in sub-linear length. Whether demi-bit can be polynomially or exponentially stretched is still an intriguing open problem.

Because proving Rudich's Conjecture will directly imply the separation of P and NP, it is a very strong conjecture. Hence, instead of directly proving Rudich's Conjecture, it is more practical to give a plausible assumption for it. As mentioned in [Rud97], the existence of super-bit or demi-bit with exponential output length implies Rudich's Conjecture. However, the existence of super-bit is not a standard assumption and compared to super-bit, demi-bit is a more intuitive notion. For this reason, if we are able to stretch demi-bit to exponential length, there will be another more intuitive assumption for Rudich's Conjecture. We will show in Chapter 6 that Conjecture 2 implies Rudich's Conjecture.

With a similar proof of the Cook-Levin theorem [Coo71], it is straightforward to observe that

$$\mathsf{P} \subseteq \mathsf{P/poly}.$$

Therefore, to separate P and NP, it is sufficient to show that there exists a language L in NP such that L cannot be decided by polynomial-size circuits. However, Razborov and Rudich [RR94] gave a barrier, eliminating many proof techniques in this approach.

We use f_n to denote a Boolean function with n variables. It is natural to view f_n[i] as the truth table of that Boolean function. Therefore, f_n is a binary string of length 2^n. We use $\mathcal{F}_n$ to denote the set of all Boolean functions with n variables. $\mathcal{C} = \{\mathcal{C}_n \subseteq \mathcal{F}_n, n \in \mathbb{N}\}$ is a combinatorial property of Boolean functions. With a little abuse of notation, we use $\mathcal{C}_n(f_n)$ to denote the predicate $f_n \overset{?}{\in} \mathcal{C}_n$, where $\mathcal{C}_n(f_n) = 1$ if and only if $f_n \in \mathcal{C}_n$.

Definition 3.1. *[RR94] **(Natural property)**. Let Γ and λ be typical complexity classes (*e.g.*, $\mathsf{P}, \mathsf{NP}, \mathsf{P/poly}, \mathsf{NP/poly}$). Let $\mathcal{C}$ be a combinatorial property of Boolean Functions. Then, we say $\mathcal{C}$ is Γ-**natural** if there exists $n_0 \geq 0$ such that for any $n \geq n_0$, there is a subset $\mathcal{C}_n^* \subseteq \mathcal{C}_n$ with the following two conditions:*

- ***Constructivity**: for any $f_n \in \mathcal{F}_n$, the predicate $f_n \overset{?}{\in} \mathcal{C}_n^*$ is computable in Γ in the size of the truth table of f_n.*
- ***Largeness**: $|\mathcal{C}_n^*| \geq 2^{-O(n)} \cdot |\mathcal{F}_n|$.*

And we say $\mathcal{C}_n$ useful against Λ if it satisfies:

- ***Usefulness**: For any sequence of Boolean functions $f_n \in \mathcal{C}_n$, $\{f_n\} \notin \Lambda$.*

Thus, if we can prove the existence of a combinatorial property $\mathcal{C}$ which contains a function in NP and is useful against $\mathsf{P/poly}$, then we can separate P and NP. However, the following theorem gives some extent of the impossibility of this approach.

Theorem 3.1. *[RR94] If there exists a SPRG $g = \{g_n : \{0,1\}^n \to \{0,1\}^{2n}\}$, there is no $\mathsf{P/poly}$-natural combinatorial property $\mathcal{C}$ useful against $\mathsf{P/poly}$.*

Proof. We prove the above theorem by contradiction. Suppose there exists a $\mathsf{P/poly}$-natural combinatorial property $\mathcal{C}$ useful against $\mathsf{P/poly}$. Without loss of generality, we assume that $\mathcal{C}_n^* = \mathcal{C}_n$. Then, we use the same construction used in Theorem 2.4.

[i]Note that this is an abuse of notation

For simplicity, we use g to denote g_n. Moreover, $g_0 : \{0,1\}^n \to \{0,1\}^n$ be the first n bits of the output of g and $g_1 : \{0,1\}^n \to \{0,1\}^n$ be the last n bits of the output of g. Given a binary string $y = y_1 y_2 \cdots y_k$ of length k where k is polynomially bounded by n, we denote $G_y(x) = g_{y_k}(\cdots g_{y_2}(g_{y_1}(x))\cdots)$. Therefore, G_y maps a binary string of length n to a binary string of length n. Furthermore, we use $f_x(y)$ to denote the first bit of $G_y(x)$, which means for a given $x \in \{0,1\}^n$, the truth table of f_x is a binary string of length 2^k.

Now, if k is polynomially bounded by n, then $f_x(y)$ is computable in circuits of size $\mathsf{poly}(n)$. Since $\mathcal{C}$ is useful against $\mathsf{P/poly}$, $f_x \notin \mathcal{C}_k$ for any $x \in \{0,1\}^n$ for sufficiently large n. As a result, we know that

$$\mathbb{P}_{x \in_R \{0,1\}^n}[\mathcal{C}_k(f_x) = 1] = 0$$

Also, by the largeness of $\mathcal{C}$, we know that

$$\mathbb{P}_{f_k \in_R \mathcal{F}_k}[\mathcal{C}_k(f_k) = 1] \geq 2^{-O(k)}$$

Consequently, by the constructivity of $\mathcal{C}$, there exists a circuit family $C' = \{C'_k\}$ such that $C'_k(f_n) = \mathcal{C}_k(f_n)$.

$$\mathbb{P}_{f_k \in_R \mathcal{F}_k}[C'_k(f_k) = 1] - \mathbb{P}_{x \in_R \{0,1\}^n}[C'_k(f_x) = 1] \geq 2^{-O(k)}$$

Now, we build the same perfect rooted binary tree in figure 3.1 as we did in the proof of Theorem 2.3. Moreover, we consider a sequence of all the internal nodes in T':

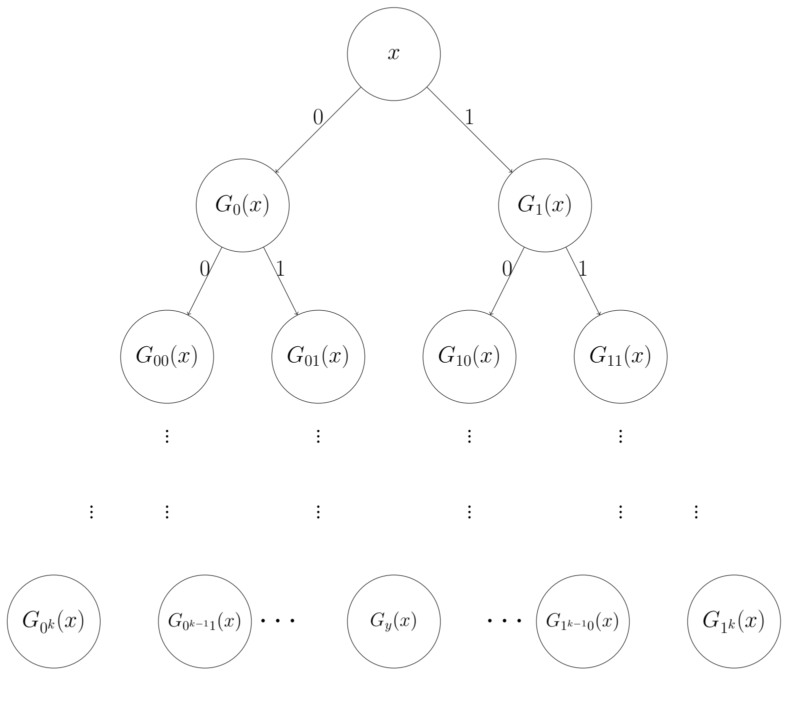

Figure 3.1: The perfect rooted binary tree T for $G_y(x)$

$$v_1, v_2, \cdots, v_{2^k-1}$$

where v_1 is the root of T' and v_j is a child of v_i for some $i < j$. We denote this sequence to be OUT. With this sequence OUT, we consider the following sequence of sub-trees of T':

1. T_1 takes the first element of OUT, which is the root of T'.
2. T_2 is the sub-tree, which consists of the first 2 nodes in OUT. (The root of T' and the left children of it).
3. $\vdots$
4. T_i is the sub-tree, which consists of the first i nodes in OUT.
5. $\vdots$
6. T_{2^k-1} is the sub-tree of T', consisting of all the nodes in OUT. In other words, T_{2^k-1} contains all the internal nodes of T'.

Based on the sequence of sub-trees $\{T_i\}$ above, we define another sequence of sub-trees $\{T'_i\}$. We define T'_0 to be the root of T'. For every integer $1 \leq i \leq 2^k - 1$, we define T'_i to be the sub-tree that contains T_i and all children of leaves in T_i.

We denote the edge between the left child and its parent as 0 and the edge between the right child and its parent as 1. For every path P from the root of T' to a leaf of T', we can write down a unique binary string $y = y_1y_2\cdots y_k$ of length k. Specifically, if the left child is on the path P, the first bit y_1 of y is 0. Otherwise, the first bit y_1 is 1. Suppose the vertex on the path P with depth $i-1$ is a, and the vertex on the path P with depth i is v. If a is the left child of v, the ith bit y_i of y is 0. Otherwise, the ith bit y_i is 1.

Given the above argument, for every $y \in \{0,1\}^k$, there exists a unique path from the root of T' to a leaf v^y of T'. For every integer $0 \leq i \leq 2^k - 1$ and $y \in \{0,1\}^k$, we use $v(i,y)$ to denote a leaf in T'_i which is on the unique path from the root of T' to v^y. We denote the distance between $v(i,y)$ and v^y as $d(i,y)$. Then, we define the function $G_{i,y} = g_{y_k} \circ \cdots \circ g_{y_{k-d(i,y)+1}}$. Moreover, we define the function $f_{i,x}(y) : \{0,1\}^k \to \{0,1\}$ as the first bit of $G_{i,y}(x)$ for $x \in \{0,1\}^n$.

For example, in figure 3.2, the red triangle surrounds the sub-tree T_3, and T'_3 is surrounded by the green triangle since it contains T_3 and each child of each leaf in T_3. Suppose $y = 0^{k-1}1$[ii], then v^y is the second leaf of T' from left to right and $v(3,y)$ is the leaf in T'_3 which is on the unique path from the root of T' to v^y. The distance between $v(3,y)$ and v^y is $k-2$, and we colour the path from the root of T' to v^y with blue.

[ii]0^k means k many bits equal to 0.

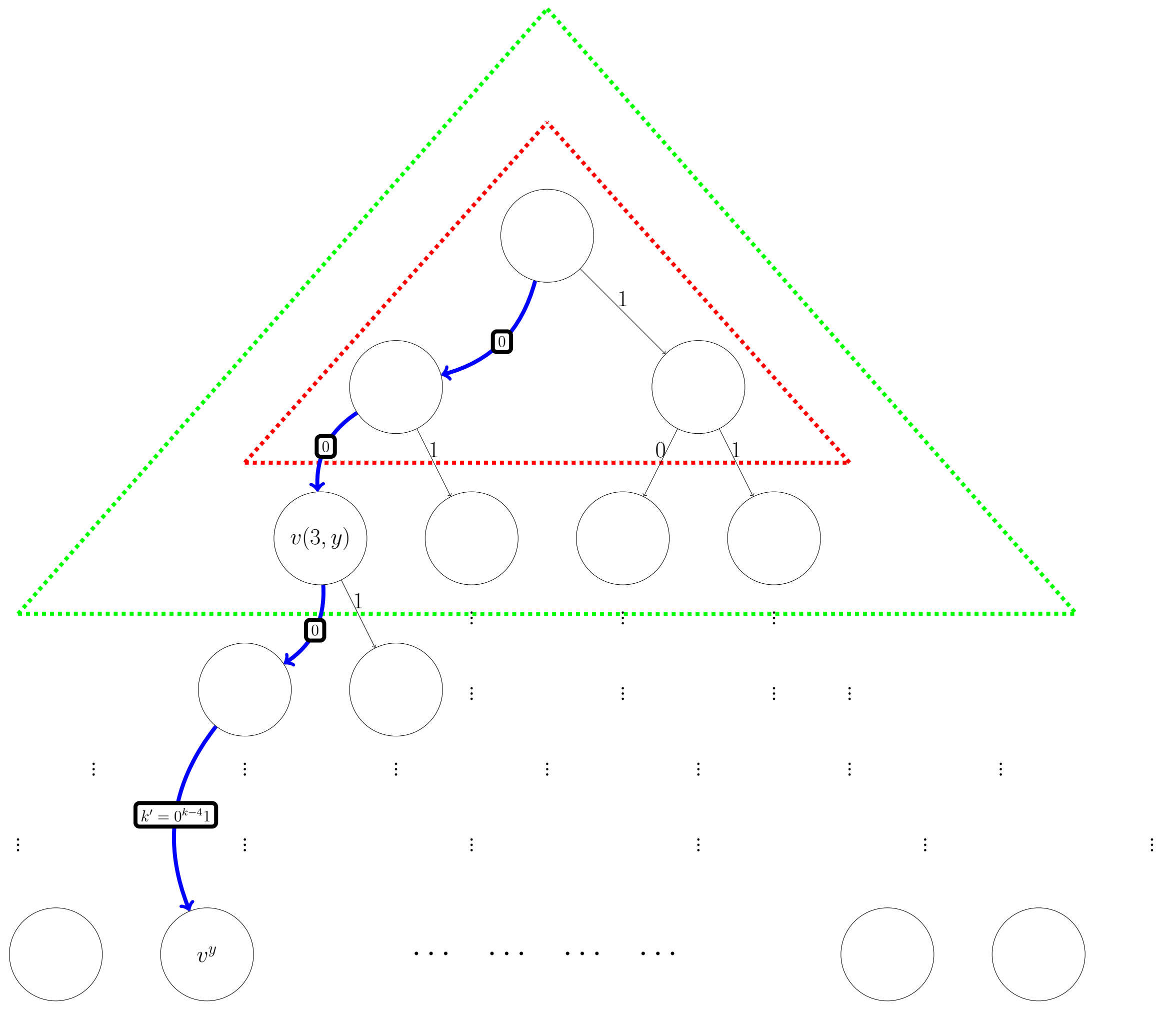

Figure 3.2: Example of sub-trees T_3 and T'_3 of T'

We can observe that $v(0, y)$ is the root of T', which implies $d(0, y) = k$. Consequently, $G_{0,y}(x) = G_{y_k} \circ \cdots \circ G_{y_1}(x) = G_y(x)$, which means $f_{0,x}(y) = f_x(y)$. Therefore,

$$\mathbb{P}_{x \in_R \{0,1\}^n}[C'_k(f_{0,x}) = 1] = \mathbb{P}_{x \in_R \{0,1\}^n}[\mathcal{C}_k(f_x) = 1] = 0$$

Also, $v(2^k - 1, y)$ is the vertex v^y, which implies $d(2^k - 1, y) = 0$. Thus, $G_{2^k-1,y}(x) = x$. Then, when x is chosen uniformly from $\{0, 1\}^n$, $f_{2^k-1,x} = f_k \in \mathcal{F}_k$.

$$\mathbb{P}_{x \in_R \{0,1\}^n}[C'_k(f_{2^k-1,x})] = \mathbb{P}_{f_k \in_R \mathcal{F}_k}[C'_k(f_k) = 1] \geq 2^{-O(k)}$$

Accordingly,

$$\mathbb{P}_{x \in_R \{0,1\}^n}[C'_k(f_{2^k-1,x})] - \mathbb{P}_{x \in_R \{0,1\}^n}[C'_k(f_{0,x}) = 1] \geq 2^{-O(k)}$$

Moreover, there must exist an integer $0 \leq i \leq 2^k - 2$ such that

$$\mathbb{P}_{x \in_R \{0,1\}^n}[C'_k(f_{i+1,x})] - \mathbb{P}_{x \in_R \{0,1\}^n}[C'_k(f_{i,x}) = 1] \geq 2^{-O(k)}$$

Before giving a distinguisher of g, we provide observations about the above result. The sub-tree T_{i+1} has only one extra node v_{i+1} than the sub-tree T_i. We denote

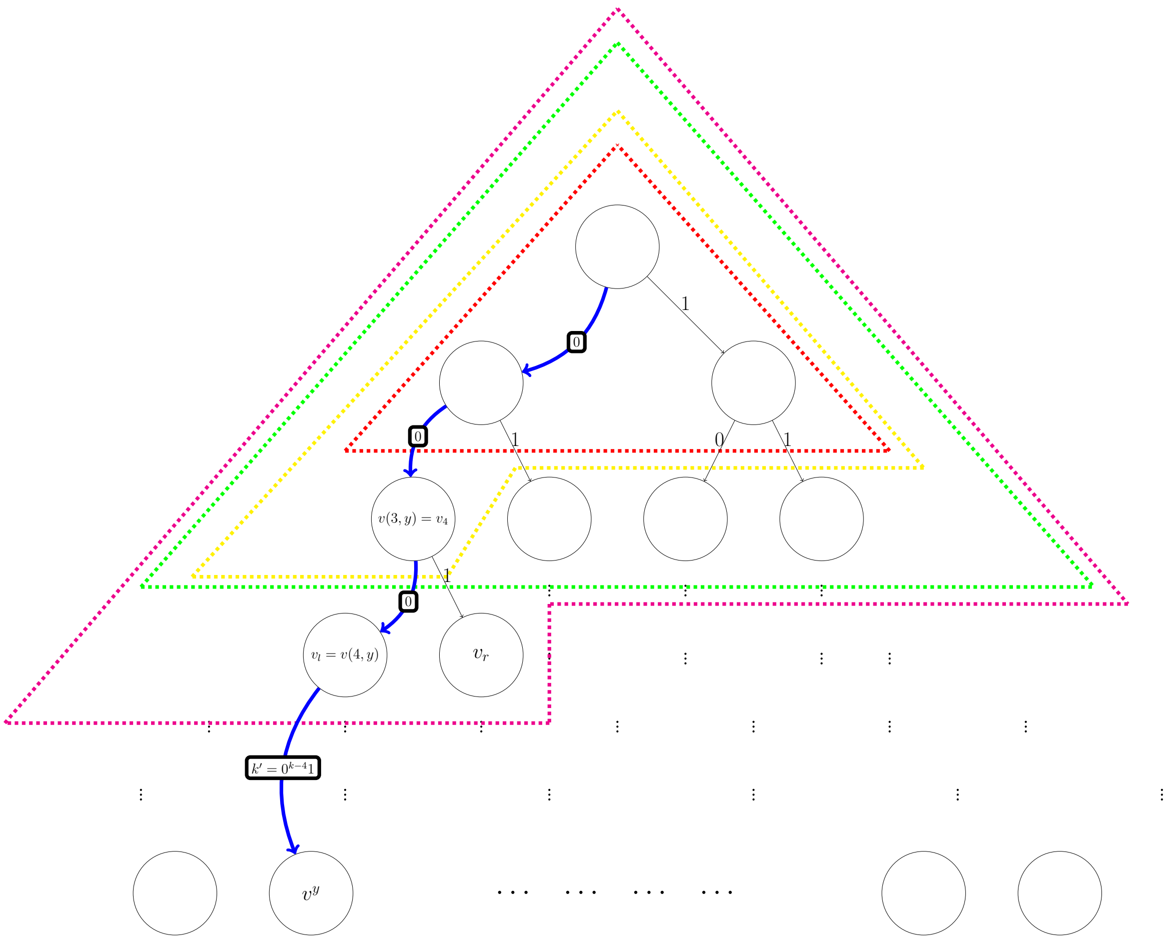

Figure 3.3: Example of sub-trees T_3, T'_3, T_4 and T'_4 of T'

the left child of v_{i+1} as v_l and the right child of v_{i+1} as v_r. If v_{i+1} is not on the unique path from the root of T' to v^y, then neither v_l nor v_r is the node $v(i+1, y)$. Therefore, $v(i, y) = v(i+1, y)$ since v_l and v_r are the only two extra nodes in T'_{i+1} from T'_i. Moreover, $v(i, y) = v(i+1, y)$ implies $d(i, y) = d(i+1, y)$, which implies $G_{i,y}(x) = G_{i+1,y}(x)$.

Also, if $v_{i+1} = v(i, y)$, then either $v_l = v(i+1, y)$ or $v_r = v(i+1, y)$.

For example, in figure 3.3, T_3 is surrounded by the red dashed triangle, and the green dashed triangle surrounds T'_3. A unique path from the root of T' to v^y, where $y = 0^{k-1}1$, is coloured by blue. Suppose $v_4 = v(3, y)$. Since v_4 is the only extra node in T_4 but not in T_3, T_4 is surrounded by the yellow dashed lines, and magenta dashed lines surround T'_4. In this case, $v(4, y)$ is the node v_l.

Now, we construct a Boolean circuit D to distinguish g. Given a binary string s of length $2n$, the circuit first computes the truth table of $f_{i,x}$ in the following way:

- for any $y \in \{0,1\}^k$ such that $v(i, y) \neq v_{i+1}$, compute $G_{i,y}(x)$ in the usual way and output the first bit.

- for any $y \in \{0,1\}^k$ such that $v(i, y) = v_{i+1}$,

- if $v(i+1,y) = v_l$, compute $G_{i,y}(s_1)$ where s_1 is the first n bits of s and output the first bit.
- if $v(i+1,y) = v_r$, compute $G_{i,y}(s_2)$ where s_2 is the last n bits of s and output the first bit.

We denote the truth table computed above as $f_{i,x}(s)$. D output $C'_k(f_{i,x}(s))$.

Now, we observe that

- if there exists a $x \in \{0,1\}^n$ such that $s = g(x)$, then the distribution of the outputs of $f_{i,x}(s)$ is equivalent to the distribution of the outputs of $f_{i,x}$.
$$\mathbb{P}_{x \in_R \{0,1\}^n}[D(g(x)) = 1] = \mathbb{P}_{x \in \{0,1\}^n}[C'_k(f_{i,x}) = 1].$$
- Otherwise, if $s \in_R \{0,1\}^{2n}$, the distribution of the outputs of $f_{i,x}(s)$ is equivalent to the distribution of the outputs of $f_{i+1,x}$.
$$\mathbb{P}_{s \in_R \{0,1\}^{2n}}[D(s) = 1] = \mathbb{P}_{x \in \{0,1\}^n}[C'_k(f_{i+1,x}) = 1].$$

Therefore,

$$\begin{aligned}
&\mathbb{P}_{s \in_R \{0,1\}^{2n}}[D(s) = 1] - \mathbb{P}_{x \in_R \{0,1\}^n}[D(g(x)) = 1] \\
=&\mathbb{P}_{x \in \{0,1\}^n}[C'_k(f_{i+1,x}) = 1] - \mathbb{P}_{x \in \{0,1\}^n}[C'_k(f_{i,x}) = 1] \\
\geq&2^{-O(k)}
\end{aligned}$$

Now, we analyze the size of the circuit D. Since the combinatorial property $\mathcal{C}$ is $\mathsf{P/poly}$-natural, which means the size of C'_k is $2^{O(k)}$. Since k is polynomially bounded by n and g is computable by polynomial-size circuits, the process of computing $f_{i,x}(s)$ is also $2^{O(k)}$, which implies the size of the circuit D is $2^{O(k)}$. Therefore, we know that $H(g) \leq 2^{O(k)}$. Since k is polynomially bounded by n, there exists a $\epsilon > 0$ such that $k = n^{\epsilon}$, which implies $H(g) \leq 2^{O(n^{\epsilon})}$. Since k is arbitrary, which means ϵ can also be arbitrary, g is not an SPRG by Definition 2.5, which is a contradiction. □

In Razborov and Rudich's paper [RR94], they have shown that several existing weak circuit lower bounds like [FSS84, And87, ABFR91, HMP+93, Smo87] can be considered as *natural*. Those properties can be useful against weak models like AC^0. However, as we showed above, any $\mathsf{P/poly}$-natural properties are unlikely to be useful against $\mathsf{P/poly}$. This barrier told complexity theorists to focus on those proof techniques that may not be considered as $\mathsf{P/poly}$-*natural*.

In this dissertation, we focus more on the relationship between natural proofs and Rudich's Conjecture, which will be presented in the following chapters.

4. PROOF COMPLEXITY

In this chapter, we will formalize crucial concepts in studying proof complexity and some general and vital results.

4.1. Propositional proof system

First, we begin with the definition of the language TAUT.

Definition 4.1. ***(TAUT).*** *$\mathsf{TAUT} \subseteq \{0,1\}^*$ is the set such that for any $x \in \mathsf{TAUT}$, x encodes a propositional formula that is true under all possible assignments of the propositional variables in x.*

One important result about the language TAUT is that

$$\mathsf{TAUT} \text{ is } \mathsf{coNP}\text{−}\mathsf{complete}.$$

Now, we define the propositional proof system, a crucial concept in the following chapter.

Definition 4.2. *[CR79]* ***(Propositional proof system).*** *A propositional proof system is a polynomial-time computable relation $R(\cdot,\cdot)$ such that for each $x \in \{0,1\}^*$, $x \in \mathsf{TAUT}$, if and only if there exists $y \in \{0,1\}^*$ such that $R(x,y)$ holds. Given $x \in \mathsf{TAUT}$, any y for which $R(x,y)$ holds is called an R-proof of x. A propositional proof system R is polynomially bounded (**p-bounded**) if there exists a polynomial p such that for each $x \in \mathsf{TAUT}$, there is an R-proof y of x of size at most $p(|x|)$ (*i.e. *$|y| \leq p(|x|)$).*

Note that each propositional proof system P has three requirements:

1. **Completeness**: If $x \in \mathsf{TAUT}$, there exists a P-proof of x.

2. **Soundness**: if there exists a P-proof of x, then $x \in \mathsf{TAUT}$.

3. **Polynomial-time computability**: Given x and y, $P(x,y)$ is polynomial-time computable.

As mentioned in the introduction, Cook and Reckhow [CR79] gave the following theorem, which implies that showing proof complexity lower bounds is an approach to separate P and NP.

Theorem 4.1. *[CR79] $\mathsf{NP} = \mathsf{coNP}$ if and only if there exists a p-bounded propositional proof system.*

Proof. "$\rightarrow$": Suppose $\mathsf{NP} = \mathsf{coNP}$, then TAUT is $\mathsf{coNP}-\mathsf{complete}$ and $\mathsf{coNP}-\mathsf{complete} \subseteq \mathsf{NP}$ implies $\mathsf{TAUT} \in \mathsf{NP}$. Consequently, by the definition of NP, there exists a polynomial-time computable relation $R(x, y)$, and a polynomial p such that

$$x \in \mathsf{TAUT} \Leftrightarrow \exists y \leq p(|x|)\ R(x, y).$$

The completeness, soundness and polynomial-time computability of such relation R are straightforward by its definition. Therefore, we know that R is a p-bounded propositional proof system.

"$\leftarrow$": Suppose there exists a p-bounded propositional proof system R. Then, we know that by definition, for each $x \in \{0, 1\}^*$,

$$x \in \mathsf{TAUT} \Leftrightarrow \exists y \leq p(|x|)\ R(x, y).$$

This implies $\mathsf{TAUT} \in \mathsf{NP}$, and since TAUT is $\mathsf{coNP}-\mathsf{complete}$, we can conclude that $\mathsf{NP} = \mathsf{coNP}$. □

Before considering a variant of the propositional proof system, we provide a variant of the Turing machine.

Definition 4.3. *[AB09] **(Turing machine with advice)**. Given $T, f : \mathbb{N} \rightarrow \mathbb{N}$, the class of languages decidable by (non-deterministic) time $T(n)$ Turing machines with $f(n)$ bits of advice, denoted $\boldsymbol{DTIME}(T(n))/f(n)$ (for non-deterministic, $\boldsymbol{NTIME}(T(n))/f(n)$), contains every language L such that there exists a sequence $\{w_n\}_{n\in\mathbb{N}}$ of strings with $w_n \in \{0, 1\}^{f(n)}$ and a (non-deterministic) Turing machine M satisfying*

$$M(x, w_n) = 1 \Leftrightarrow x \in L$$

for every $x \in \{0, 1\}^n$, where on input (x, w_n) the machine M runs for at most $O(T(n))$ steps.

Analogously, we provide the definition of propositional proof systems with advice.

Definition 4.4. *[CK07] **(Propositional proof system with advice)**. Given $f : \mathbb{N} \rightarrow \mathbb{N}$, a propositional proof system with f bits of advice is a polynomial-time computable relation $R(x, y, z)$ such that for each $n \in \mathbb{N}$, there is $w_n \in \{0, 1\}^*$ of length $f(n)$ satisfying the following condition:*

for each $x \in \{0, 1\}^n$, $x \in \mathsf{TAUT}$ if and only if there exists $y \in \{0, 1\}^*$ such that $R(x, y, w_n)$ holds.

*We call an advice string w_n **good for** R if it satisfies the preceding condition, we call an advice sequence $\{w_n\}$ **good for** R if for each n, w_n is **good for** R. Given $x \in \mathsf{TAUT}$ and advice string $w_{|x|}$, any y for which $R(x, y, w_{|x|})$ holds is called an R-proof of x with advice $w_{|x|}$. A propositional proof system R with advice is **p-bounded*** *if there exists an advice sequence $\{w_i\}$ **good for** R and a polynomial p such that for each $x \in \mathsf{TAUT}$, there is an R-proof y of x with advice $w_{|x|}$ of size at most $p(|x|)$ (*i.e.$|y| \leq p(|x|)$*).*

First, we show that non-deterministic Turing machines with advice give another characterization of $\mathsf{NP/poly}$.

Lemma 4.2. $\mathsf{NP/poly} = \cup_{c,d\in\mathbb{N}} \boldsymbol{NTIME}(n^c)/n^d$

Proof. "$\rightarrow$": For any language $L \in \mathsf{NP/poly}$, we know that there exists a polynomial-size non-deterministic circuit family $\{C_n\}_{n\in\mathbb{N}}$ that computes L:

$$\text{for any } x \in \{0,1\}^n,\ x \in L \text{ if and only if there exists } r \text{ such that } C_n(x,r) = 1 \text{ and } |r| \text{ is polynomially bounded by } n.$$

Since C_n has polynomial size, there exists a constant $c > 0$ such that $|C_n| < n^c$. Then, for every gate in C_n, we assign a number to it. Since every gate has at most two inputs and one output, we can use $3\lceil\log(n^c)\rceil$ bits to represent a gate. For this reason, it is sufficient to use $3n^c\lceil\log(n^c)\rceil$ bits to describe a circuit of size at most n^c. Then, we define a non-deterministic Turing machine M with advice family $\{w_n\}_{n\in\mathbb{N}}$ where w_n is the description of C_n. M guesses the string r and outputs $C_n(x,r)$ for $x \in \{0,1\}^n$.

"$\leftarrow$": We use the same idea as the proof of $\mathsf{P} \subseteq \mathsf{P/poly}$: Given any language $L \in \mathsf{P}$, and a Turing machine M that decides L, we can construct a polynomial-size circuit family $C = \{C_n\}_{n\in\mathbb{N}}$ such that

$$x \in L \Leftrightarrow M(x) = 1 \Leftrightarrow C_{|x|}(x) = 1$$

(See [AB09] for details.) Based on this idea, for any language $L' \in \cup_{c,d\in\mathbb{N}} \mathbf{NTIME}(n^c)/n^d$, which is decided by a Turing machine M with advice family $\{w_n\}_{n\in\mathbb{N}}$ such that $|w_n| \leq p(n)$ for some polynomial p, we can construct a non-deterministic circuit family $C' = \{C'_n\}_{n\in\mathbb{N}}$ such that

$$x \in L' \Leftrightarrow \exists\, y \in \{0,1\}^{q(|x|)} M(x, y, w_{|x|}) = 1 \Leftrightarrow \exists\, y \in \{0,1\}^{q(|x|)} C'_{|x|}(x, y, w_{|x|}) = 1,$$

where q is a polynomial. Furthermore, we fix the advice string $w_{|x|}$ into $C'_{|x|}$ to construct $D_{|x|}$. It is straightforward to see that $|D_{|x|}| = |C'_{|x|}|$ and $C'_{|x|}(x, y, w_{|x|}) = D_{|x|}(x, y)$.

$$x \in L' \Leftrightarrow \exists\, y \in \{0,1\}^{q(|x|)} D_{|x|}(x, y) = 1.$$

Therefore, we can conclude that $L' \in \mathsf{NP/poly}$. □

Now, we provide a connection between the propositional proof system with advice and Turing machines with advice.

Theorem 4.3. $\mathsf{NP} \subseteq \mathsf{coNP/poly}$ *if and only if there exists a p-bounded propositional proof system with polynomial advice.*

Proof. "$\rightarrow$": Since $\mathsf{NP} \subseteq \mathsf{coNP/poly}$, then for the $\mathsf{NP-complete}$ language SAT, there exists a polynomial-size co-non-deterministic circuit family $C = \{C_n\}_{n\in\mathbb{N}}$ such that

$$x \in \mathsf{SAT} \Leftrightarrow \forall\ y \in \{0,1\}^{q(|x|)} C_{|x|}(x,y) = 1,$$

where q is a polynomial. Moreover, we flip the output of C to construct a polynomial-size non-deterministic circuit family $C' = \{C'_n\}_{n\in\mathbb{N}}$ such that

$$x \in \mathsf{SAT} \Leftrightarrow \forall\ y \in \{0,1\}^{q(|x|)} C'_{|x|}(x,y) = 0,$$

which implies

$$x \in \mathsf{UNSAT} \Leftrightarrow \exists\ y \in \{0,1\}^{q(|x|)} C'_{|x|}(x,y) = 1$$

where UNSAT is the complement of SAT. Since we know that UNSAT is $\mathsf{coNP-complete}$ and $\mathsf{TAUT} \in \mathsf{coNP}$, we know that there exists a polynomial-time computable function f such that

$$x \in \mathsf{TAUT} \Leftrightarrow f(x) \in \mathsf{UNSAT}.$$

Due to the above results, it is clear that

$$x \in \mathsf{TAUT} \Leftrightarrow \exists\ y \in \{0,1\}^{q(|f(x)|)} C'_{q(|f(x)|)}(f(x),y) = 1 \Leftrightarrow f(x) \in \mathsf{UNSAT}.$$

Moreover, since f is polynomial-time computable and $\mathsf{P} \subseteq \mathsf{P/poly}$, there exists a polynomial-size circuit family that computes f. Then, we hard-wire those circuits into the corresponding circuits in C' to construct a new polynomial-size non-deterministic circuit family $D = \{D_n\}_{n\in\mathbb{N}}$. It is straightforward to see that

$$x \in \mathsf{TAUT} \Leftrightarrow \exists\ y \in \{0,1\}^{u(|x|)} D_{|x|}(x,y) = 1,$$

where u is a polynomial.

By Lemma 4.2, we know that there exists a polynomial-time computable non-deterministic Turing machine M with advice family $\{a_n\}_{n\in\mathbb{N}}$ such that

$$M(x,y,a_{|x|}) = D_{|x|}(x,y)$$

After that, we construct a proportional proof system R with advice family $\{a_n\}_{n\in\mathbb{N}}$ as follows: Given $x \in \{0,1\}^*$, $x \in \mathsf{TAUT}$ if and only if there exists $y \in \{0,1\}^{l(n)}$ where y is the witness of x in M such that $M(x,y,a_{|x|}) = 1$. The completeness, soundness and polynomial-time computability are easy to check. Therefore, R is a p-bounded propositional proof system.

"$\leftarrow$": Suppose there exists a p-bounded propositional proof system R with polynomial advice family $a = \{a_n\}_{n\in\mathbb{N}}$. Then, for $x \in \{0,1\}^*$,

$$x \in \mathsf{TAUT} \Leftrightarrow \exists\ y \in \{0,1\}^{p(|x|)} R(x,y,a_{|x|}) = 1,$$

where p is a polynomial. Due to the polynomial-time computability of R, there is a polynomial-time computable non-deterministic Turing machine with polynomial advice that computes R. Therefore, we know that $\mathsf{TAUT} \in \mathsf{NP/poly}$. By Lemma 4.2, there exists a polynomial-size non-deterministic circuit family $C = \{C_n\}_{n\in\mathbb{N}}$ with advice family $a = \{a_n\}_{n\in\mathbb{N}}$ such that

$$R(x, y, a_{|x|}) = C_{|x|}(x, y, a_{|x|}).$$

Then, we flip the output of C to get a polynomial-size co-non-deterministic circuit family $C' = \{C'_n\}_{n\in\mathbb{N}}$ with advice family $a = \{a_n\}_{n\in\mathbb{N}}$ such that

$$x \in \mathsf{TAUT} \Leftrightarrow \exists\ y \in \{0,1\}^{p(|x|)} C'_{|x|}(x, y, a_{|x|}) = 0.$$

We define the complement of TAUT as coTAUT, which is a $\mathsf{NP{-}complete}$ language. Then,

$$x \in \mathsf{coTAUT} \Leftrightarrow \forall\ y \in \{0,1\}^{p(|x|)} C'_{|x|}(x, y, a_{|x|}) = 1.$$

Here, we fix the each advice $a_{|x|}$ into the each circuit $C_{|x|}$ to construct a polynomial-size co-non-deterministic circuit family $D = \{D_n\}_{n\in\mathbb{N}}$ such that

$$x \in \mathsf{coTAUT} \Leftrightarrow \forall\ y \in \{0,1\}^{p(|x|)} D_{|x|}(x, y) = 1,$$

which implies $\mathsf{NP} \subseteq \mathsf{coNP/poly}$. $\square$

Now, we give a concept of *simulation* between propositional proof systems.

Definition 4.5. *[CR79] **(Simulation)**. Given two propositional proof systems P and Q, we say P **simulates** Q if and only if there exists polynomial-time computable function p and a polynomial q such that for any $x \in \{0,1\}^*$, there is an s-size Q-proof y of x if and only if there is an $q(s)$-size P-proof $p(y)$ of x.*

We say a propositional proof system R is optimal if it simulates all propositional proof systems.

4.2. *Specific proof systems*

Now, we provide several specific and crucial propositional proof systems in the literature. First, we provide some necessary definitions. Most of the concepts are taken from [Kra19].

Definition 4.6. ***(De Morgan language)**. The **De morgan language** of propositional logic consists of the following:*

- *An infinite set of atoms (*i.e. *propositional variables) denoted by $x, y, p, q, \cdots$*
- *logical connectives, consisting of*
  - *the constants $\top$ (**true**) and $\bot$ (**false**)*
  - *a nary connective $\neg$ (**negation**)*

    - *binary connectives* $\vee$ ***(disjunction)*** *and* $\wedge$ ***(conjunction)***

- *parentheses (,)*

The ***alphabet*** *over De Morgan language consists of all the atoms, logical connectives and parentheses given above. The* ***formulas*** *are finite words over the alphabet, constructed by applying finitely many times in an arbitrary order the following rules:*

- *the constant* $\top$ *and* $\bot$ *are formulas and any atom is a formula;*

- *If* α *is a formula,* $\neg\alpha$ *is also a formula.*

- *If* α, β *are formulas, then* $(\alpha \vee \beta)$ *and* $(\alpha \wedge \beta)$ *are also formulas.*

Definition 4.7. ***(Literal, term and clause)****. A* ***literal*** ℓ *is an atom or its negation. A conjunction of literals is called a* ***term****, and a disjunction of literals is called a* ***clause****.*

Definition 4.8. ***(Finite and complete language)****. We say L is a* ***finite and complete language*** *for propositional logic if L consists of a finite number of connectives interpreted by specific Boolean functions of the appropriate arity*[i] *and having the property that any Boolean function of any arity can be defined by an L-formula, where L-formulas are finite words constructed by applying finitely many times the rules corresponding to the connectives in L, similarly as in definition 4.6.*

Definition 4.9. ***(Frege rule)****. Let L be a finite complete language for propositional logic and let* $\ell \geq 0$*. An* ***ℓ-ary Frege rule*** *(in the language L) is any* $(\ell+1)$*-tuple of L-formulas* $A_0, \cdots, A_\ell$*, written as*

$$\frac{A_1, \cdots, A_\ell}{A_0},$$

such that

$$A_1, \cdots, A_\ell \models A_0.$$

The formulas $A_1, \cdots, A_\ell$ *are called the* ***hypotheses*** *of the rule and* A_0 *is its* ***consequence****. A rule with no hypotheses (*i.e. $\ell = 0$*) is called a* ***Frege axiom scheme****. An axiom scheme is written simply as* A_0*.*

Definition 4.10. ***(Frege proof)****. Let F be a finite set of Frege rules in a finite complete language L. An F-proof of an L-formula C from the L-formulas* $B_1, \cdots, B_t$ *is any sequence of L-formulas* $D_1, \cdots, D_k$ *such that:*

- $D_k = C$*;*

- *for all* $i = 1, \cdots, k$*,*

[i] arity is the number of inputs of a Boolean function

– *either there is an ℓ-ary rule*

$$\frac{A_1, \cdots, A_\ell}{A_0}$$

in F, numbers $j_1, \cdots, j_\ell < i$ and a substitution σ such that

$$\sigma(A_1) = D_{j1}, \cdots, \sigma(A_\ell) = D_{j\ell} \text{ and } \sigma(A_0) = D_i,$$

– *or $D_i \in \{B_1, \cdots, B_t\}$.*

We shall denote by

$$\pi : B_1, \cdots, B_t \vdash_F C$$

the fact that $\pi = (D_1, \cdots, D_k)$ is an F-proof of C from $B_1, \cdots, B_t$, and by the notation

$$B_1, \cdots, B_t \vdash_F C$$

the fact that there exists an F-proof of C from $B_1, \cdots, B_t$.

Definition 4.11. ***(Frege proof system)****. Let L be a finite complete language for propositional logic. A* ***Frege proof system*** *F in the language L is a finite set of Frege rules that is sound and* ***implicationally complete****, meaning that, for any L-formulas $B_1, \cdots, B_t, C$,*

$$B_1, \cdots, B_t \models C \text{ if and only if } B_1, \cdots, B_t \vdash_F C.$$

Before showing other results and proof systems, we define the meaning of the "size" of a proof.

Definition 4.12. ***(The size of a proof)****. For any proof system, the size of a proof is the number of symbols used in the proof.*

Then, Reckhow proved the following important theorem.

Theorem 4.4. *[Rec75] Any two Frege systems F_1, F_2 in any complete languages containing the De Morgan Language simulate each other.*

Now, we give the definition of the Extended Frege proof system.

Definition 4.13. ***(EF-derivation)****. A sequence of formulas*

$$C_1, \cdots, C_k$$

is an ***EF-derivation*** *from the set $B_1, \cdots, B_t$ if every C_i is one of the B_j, or is obtained from some earlier formulas C_u by one of the rules of F, which is a Frege system, or is the* ***extension rule*** *which has the form*

$$q \equiv D$$

where $\equiv$ abbreviates its definition in the language of the particular system F, and

- *the atom q does not occur in any $B_1, \cdots, B_t$*
- *q does not occur in $C_1, \cdots, C_{i-1}$*
- *q does not occur in D*

The formula $q \equiv D$[ii] *is called an **extension axiom**, and the atom q is called the **extension atom** introduced in the axiom.*

Definition 4.14. ***(EF-proof)****. An EF-derivation from the set $B_1, \cdots, B_t$ ending with the formula A is called an **EF-proof** of A from $B_1, \cdots, B_t$ if, in addition,*

no extension atom introduced in the derivation occurs in A.

Definition 4.15. ***(Extended Frege proof system)****. Let L be a finite complete language for propositional logic. A **Extended Frege proof system** EF in the language L is a Frege system with the **extension rules**.*

Lemma 4.5. *[CR79] Extended Frege systems are sound.*

Now, we will divert our attention to another kind of proof system, which is the Resolution proof system.

Definition 4.16. ***(Resolution proof system)****. All lines in a **resolution proof** are just clauses $C : \ell_1 \vee \cdots \vee \ell_w$ written simply as sets of literals $\{\ell_1, \cdots, \ell_w\}$ that form the disjunction. The clause $C \cup \{\ell\} = \ell_1 \vee \cdots \vee \ell_w \vee \ell$ is abbreviated by C, ℓ.*

*Let L be a finite complete language for propositional logic. A **resolution proof system** R in the language L only contains the **resolution rule**:*

$$\frac{C, p \qquad D, \neg p}{C \cup D},$$

where p is an atom.

Instead of viewing resolution proof as proving something, we can view it as refuting something, which means R can be interpreted as **refutation proof system**: instead of proving a formula is a tautology, a refutation proof system proves that a set of clauses is not satisfiable. Then, to prove a propositional formula A is a tautology, we can use R to prove that $\neg A$ is unsatisfiable.

Now, similar to the Extended Frege proof system, we provide the definition of the Extended Resolution proof system.

Definition 4.17. ***(Extended Resolution proof system)****. Let L be a finite complete language for propositional logic. A **Extended resolution proof system** ER in the language L is a resolution system with the **extension rules**:*

[ii]the symbol $\equiv$ can be expressed in the language of the Extended Frege proof system

1. *Pick two literals ℓ_1 and ℓ_2 and a new atom q and add the following three clauses to the clauses:*

$$\{\neg\ell_1, q\}, \qquad \{\neg\ell_2, q\}, \qquad \{\ell_1, \ell_2, \neg q\}$$

2. *Repeat step 1 an arbitrary number of times, at any step, for any literals ℓ_1, ℓ_2 (including the literals introduced in previous steps) but always choosing a new atom in the place of q.*

The variables q introduced in step 1 are called ***extension variables****, and the three clauses are the* ***extension axioms****.*

Lemma 4.6. *[Kra19] Extended Resolution proof systems are sound.*

One vital result about the Extended Frege proof system and the Extended Resolution proof system was given by [CR79].

Theorem 4.7. *[CR79] The Extended Frege proof system and the Extended Resolution proof system simulate each other.*

Within this chapter, we present three distinct ways of formalizing Rudich's Conjecture. These include Rudich's initial formulation as documented in [Rud97], a formulation utilizing the propositional proof system, and the formulation provided in [PS19]. We demonstrate the equality between the first two formalizations, with the third formulation implying the initial two. Furthermore, we illustrate how the Super-bit Conjecture implies Rudich's Conjecture.

First, we give the original formalization given in [Rud97].

Conjecture 4. ***Rudich's Conjecture****.* *There are no* $\mathsf{NP}/\mathsf{poly}$*-natural properties that are useful against* P/poly*.*

Specifically, this means that for any combinatorial property $\mathcal{C}$, such that for any $f_n \in \mathcal{C}_n$, f_n does not have polynomial-size circuits and the predicate $f_n \overset{?}{\in} \mathcal{C}_n$ is in $\mathsf{NP}/\mathsf{poly}$, $|C_n| \leq 2^{-\omega(n)} \cdot |\mathcal{F}_n|$.

We provide another version of Rudich's Conjecture using the propositional proof system. We first define the **circuit lower bound formula**.

Definition 5.1. *[PS19]* ***(Circuit lower bound formula)****.* *An* $s(n)$*-size circuit lower bound for a function* $f_n \in \mathcal{F}_n$ *is a* $2^{O(n)}$*-size propositional formula* $\mathsf{clb}(f,s)$*,*

$$\bigvee_{y\in\{0,1\}^n} f_n(y) \neq C(y),$$

where the formula $f_n(y) \neq C(y)$ *says that a circuit* C *represented by* $\mathsf{poly}(s)$ *variables does not output* $f_n(y)$ *on input* y*.* $\mathsf{clb}(f,s)$ *is a tautology if and only if* f_n *does not have* $s(n)$*-size circuits.*

$\mathsf{clb}(f,s)$ *is a propositional DNF formula of size* $\tilde{O}(2^n s^3)$ [i] *over* $2^{O(n)}$ *variables expressing that* f *does not have Boolean circuits of size* s*.*

In $\mathsf{clb}(f,s)$, there are $\mathsf{poly}(s)$ many variables encoding a circuit of size at most s. Each proper assignment to those variables represents an encoding of a circuit of size at most s. Therefore, if f cannot be computed by the circuit C under all possible $2^{\mathsf{poly}(s)}$ many assignments, $\mathsf{clb}(f,s)$ is a tautology[ii].

Conjecture 5. ***Rudich's Conjecture with proof system****.* *For every propositional proof system* P *with polynomial advice family* $\{w_n\}$*, there exists a constant* $c > 0$ *such that, for large enough* n*, for* $1 - 1/N^{\omega(1)}$ *fraction of Boolean function* $f_n \in \mathcal{F}_n$*,* P *does not have a polynomial-size* P*-proof of* $\mathsf{clb}(f_n, n^c)$*.*

[i] $\tilde{O}(f)$ represents $O(f \cdot \mathsf{poly}(\log(f)))$

[ii] The formula $\mathsf{clb}(f,s)$ can be in principle encoded in many ways but for the purpose of this thesis any standard encoding will work.

Theorem 5.1. *Conjecture 4 is equivalent to Conjecture 5.*

Proof. First, we prove that the Conjecture 4 implies the Conjecture 5. Suppose the Conjecture 4 is true and the Conjecture 5 is false. Then, the failure of Conjecture 5 implies that

there exists a propositional proof system P with polynomial advice family and for all constant $c > 0$ such that for $1 - 1/N^{\omega(1)}$ fraction of Boolean function $f_n \in \mathcal{F}_n$, there are polynomial-size P-proof of $\mathsf{clb}(f_n, n^c)$.

Moreover, by the soundness of P, we know that for a $1 - 1/N^{\omega(1)}$ fraction of Boolean functions, f_n does not have circuits of size n^c for all constant $c > 0$. Consequently, there exists a $1 - 1/N^{\omega(1)}$ fraction of Boolean functions f_n such that f_n does not have polynomial-size circuits.

Now, we define a language L_P such that a binary string x of length 2^n is in L_P if and only if x is a truth table of a Boolean function f_n and for any constant $c > 0$, there is a polynomial-size P-proof of $\mathsf{clb}(f_n, n^c)$. We consider our L_P to be a combinatorial property. We show that L_P is a candidate that contradicts the Conjecture 4.

- **Constructivity**: We can observe that $L_P \in \mathsf{NP/poly}$ since a non-deterministic Turing machine M can decide the language by simply guessing the P-proof of it and verify it with the same polynomial advice family in polynomial-time by the polynomial-time computability of P.
- **Largeness**: Also, $|L_{Pn}| \geq 1 - 1/N^{\omega(1)} \cdot |\mathcal{F}_n| \geq 2^{-O(n)} \cdot |\mathcal{F}_n|$ by property of the propositional proof system P.[iii]
- **Usefulness**: By the definition of the language L_P, all the binary strings in L_P are truth tables of Boolean functions f_n that have P-proof of $\mathsf{clb}(f_n, n^c)$ for any constant $c > 0$. By the soundness of P, f_n does not have polynomial-size circuits, which implies L_P is useful against $\mathsf{P/poly}$.

As a result, the existence of L_P contradicts the Conjecture 4.

Now, we prove that the Conjecture 5 implies the Conjecture 4. Suppose the Conjecture 5 is true and the Conjecture 4 is false. Therefore, the failure of the Conjecture 4 implies that

there exists a $\mathsf{NP/poly}$-natural property $\mathcal{C}$ useful against $\mathsf{P/poly}$ such that $|\mathcal{C}_n| \geq 2^{-O(n)} \cdot |\mathcal{F}_n|$ for sufficiently large n.

Now, we build a propositional proof system Q to contradict the Conjecture 5. Then, for some $c > 0$, we build the propositional proof system Q as follows: $Q(u, v, w) = 1$ if and only if

[iii] $L_{Pn} = L_P \cap \{0,1\}^{2^n}$.

- $u \in \mathsf{TAUT}$ and $v = 01^{2^{|u|}}$ or
- $u = \mathsf{clb}(f_n, n^c)$ where $f_n \in \mathcal{C}_n$, v is the witness of u in the polynomial-size non-deterministic circuit D_n that decides $f_n \overset{?}{\in} \mathcal{C}_n$, and the polynomial-size advice w is D_n (properly encoded),

where a sequence of $\{w_n\}_{n\in\mathbb{N}}$ good for Q is the encoding of the polynomial-size non-deterministic circuit family $\{D_n\}_{n\in\mathbb{N}}$.

Now, we check the completeness, soundness and polynomial-time computability of Q.

- **Completeness**: it is clear that all tautologies can be proved by Q.
- **Soundness**: Since $\mathcal{C}_n$ is useful against $\mathsf{P/poly}$, then all $f_n \in C_n$ does not have circuit of size n^c. Hence, $\mathsf{clb}(f_n, n^c)$ are tautologies, which implies all formulas proved by Q are tautologies.
- **Polynomial-time computability**:
  - if $u \in \mathsf{TAUT}$ and $v = 01^{2^{|u|}}$, then with brute force, Q can check all the possible assignments on u in time $2^{O(|u|)}$. Q is polynomial-time computable in this case.
  - In the other case, Q just constructs the non-deterministic circuit D_n from the advice w_n and runs u and v on it. This is also polynomial-time computable.

By the largeness of $\mathcal{C}$ and c is arbitrary, we know that for any constant $c > 0$, for a $2^{-O(n)}$ fraction of Boolean function, there are polynomial-size Q-proof of $\mathsf{clb}(f_n, n^c)$, which is a contradiction. □

Before giving another formalization, we provide some necessary notations.

Definition 5.2. *[PS19] **(Sparse and dense language)**. Given a language $L \subseteq \{0,1\}^*$, the n-slice L_n of L is $L \cap \{0,1\}^n$. Given a function $\epsilon : \mathbb{N} \to [0,1]$, we say that a language $L \subseteq \{0,1\}^*$ is ϵ-sparse if for each large enough n, $|L_n|/2^n \leq \epsilon(n)$. L is ϵ-dense if for each large enough n, $|L_n|/2^n > \epsilon(n)$.*

Given a string y of length N, $\mathsf{fn}(y)$ is the Boolean function on $\lfloor \log(N) \rfloor$ bits whose truth table is the $2^{\lfloor \log(N) \rfloor}$-bit initial prefix of y. Now, we use these notations to define the MCSP (Minimum Circuit Size Problem).

Definition 5.3. *[KC00] **(Minimum Circuit Size Problem)**. Given a size function $s : \mathbb{N} \to \mathbb{N}$, $\mathsf{MCSP}[s]$ is the set of strings y such that $\mathsf{fn}(y)$ has Boolean circuits of size at most $s(n)$.*

Now, we give the formalization given in [PS19].

Conjecture 6. ***Stronger Rudich's Conjecture.*** *There is a constant $c > 0$ such that for every language L, if $L \subseteq \overline{\mathsf{MCSP}[n^c]}$ and $L \in \mathsf{NP/poly}$, then L is $\frac{1}{N^{\omega(1)}}$-sparse where $N = 2^n$.*

Proposition 5.2. *The stronger Rudich's Conjecture (Conjecture 6) implies Rudich's Conjecture (Conjecture 4).*

Proof. Suppose the Conjecture 6 is true. We assume the Conjecture 4 is wrong, which means

there exists a combinatorial property $\mathcal{C}$, such that for any $f_n \in \mathcal{C}_n$, f_n does not have polynomial-size circuits and the predicate $f_n \overset{?}{\in} \mathcal{C}_n$ is in $\mathsf{NP/poly}$, $|\mathcal{C}_n| \geq 2^{-O(n)} \cdot |\mathcal{F}_n|$.

Then, we consider $\mathcal{C}$ as a language $L_{\mathcal{C}}$. We can see that $L_{\mathcal{C}} \subseteq \overline{\mathsf{MCSP}[n^c]}$ and $L_{\mathcal{C}} \in \mathsf{NP/poly}$, but $L_{\mathcal{C}}$ is not $\frac{1}{N^{\omega(1)}}$-sparse, which is a contradiction. □

Now, we show how the Super-bit Conjecture 2 implies Conjecture 4.

Theorem 5.3. *Super-bit Conjecture implies Rudich's Conjecture.*

Proof. Suppose the Conjecture 2 is true. There exists a $\epsilon > 0$ such that there is a PRG $g = \{g_n : \{0,1\}^n \to \{0,1\}^{n+1}\}$ for all sufficiently large n, $H_s(g_n) \geq 2^{n^\epsilon}$. Then, by Theorem 2.2, we can get a PRG $G = \{G_n : \{0,1\}^n \to \{0,1\}^{2n}\}$ with super-bit hardness $H_s(G_n) \geq 2^{n^{\epsilon'}}$ for some constant $\epsilon' > 0$. Moreover, using the same argument in the proof of Theorem 3.1 with all the deterministic circuits replaced by non-deterministic circuits, we can get Rudich's Conjecture. □

If we can stretch a demi-bit to polynomial length, then we can also show that the Demi-bit Conjecture also implies Rudich's Conjecture with the same argument as above.

At this point, we are clear about the connections between pseudorandom generators in the non-deterministic setting and Rudich's Conjecture. We will move our attention to Razborov's Conjectures in the next chapter.

6. RAZBOROV'S CONJECTURE

In this chapter, we will focus on the formalization and explanation of Razborov's Conjecture. First, we give the original statements in [Raz15] and formalize them later.

Razborov's Conjecture 1: Any Nisan-Wigderson generator based on any polynomial-time computable function that is hard on average for $\mathsf{NC}^1/\mathsf{poly}$ is hard for the Frege proof system.
Razborov's Conjecture 2: Any Nisan-Widgerson generator based on any function in $\mathsf{NP} \cap \mathsf{coNP}$ that is hard on average for P/poly is hard for Extended Frege proof system.

Now, we provide the mathematical definitions of the concepts mentioned above. We first provide the definition of the complexity class NC.

Definition 6.1. *[AB09] (**The class** NC). For every d, a language L is in NC^d if L can be decided by a family of circuits $\{C_n\}$ with bounded fan-in (*i.e.*, the OR and AND gates can only be applied to two inputs) where C_n has $\mathsf{poly}(n)$ size and depth $O(\log^d n)$. The class NC is $\bigcup_{i\geq 1} \mathsf{NC}^i$.*

6.1. Average-case hardness

Now, we will introduce the notion of average-case hardness.

Definition 6.2. *[AB09] (ρ-**average case hardness**). For $f : \{0,1\}^n \to \{0,1\}$ and $0 \leq \rho \leq 1$, we define the ρ-**average case hardness** of f, denoted $H^{\rho}_{avg}(f)$, to be the largest S such that for every circuit C of size at most S,*

$$\mathbb{P}_{x\in_R\{0,1\}^m}[C(x) = f(x)] < \rho.$$

For an infinite $f : \{0,1\}^* \to \{0,1\}$, we let $H^{\rho}_{avg}(f)(n)$ denote $H^{\rho}_{avg}(f_n)$ where f_n is the restriction of f to $\{0,1\}^n$.

Definition 6.3. *[AB09] (**Average-case hardness**). We define the **average-case hardness** of f, denoted $H_{avg}(f)$, to equal $\max\{S : H^{\frac{1}{2}+\frac{1}{S}}_{avg} \geq S\}$. That is, $H_{avg}(f)$ is the largest number S such that*

$$\mathbb{P}_{x\in_R\{0,1\}^n}[C(x) = f(x)] < \frac{1}{2} + \frac{1}{S}$$

for every Boolean circuit C on n inputs with size at most S.

The next several notions are necessary to define the Nisan-Widgerson generator introduced in [NW94].

Definition 6.4. ***((k,m)-design)****. A collection of sets $\{S_1, \cdots, S_n\}$, where $S_i \subseteq \{1, \cdots, l\}$ is called a* ***(k,m)-design*** *if:*

1. *For all i,*
$$|S_i| = m;$$
2. *For all $i \neq j$,*
$$|S_i \cap S_j| \leq k.$$

A $n \times l$ 0-1 matrix is called a ***(k,m)-design*** *if the collection of its n rows interpreted as subsets of $\{1, \cdots, l\}$ is a (k,m)-design.*

Definition 6.5. ***(Boolean function f_A defined by matrix A)****. Let A be a $n \times l$ 0-1 matrix, let f be a single-output Boolean function, and let $x = (x_1 \cdots x_l)$ be a Boolean string. $f_A(x)$ is the n bit vector of bits computed by applying the function f on n-sub-strings of x, where the ith sub-string is obtained by projecting x on coordinates corresponding to 1s in the ith row of A.*

Here, we give an example of matrix A and the Boolean function f_A defined by A.

For example, suppose we have a 3×3 0-1 matrix, which is a (1,2)-design: $\begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}$ and $x = 010$. Then, given a Boolean function f, $f_A(x) = (f_A(x,1), f_A(x,2), f_A(x,3))$, where $f_A(x,i)$ represents the ith bit of the output. As a result, $f_A(x,1) = f(01), f_A(x,2) = f(10), f_A(x,3) = f(00)$.

Now, we define the Nisan-Widgerson generator (NW-generator).

Definition 6.6. ***(Nisan-Widgerson generator)****. Let n, l, k, m be integers where $n > l$; let $f : \{0,1\}^m \to \{0,1\}$ be a Boolean function and let A be a Boolean $n \times l$ matrix, which is a (k,m)-design. Then $f_A(\cdot) : \{0,1\}^l \to \{0,1\}^n$ is a* ***Nisan-Widgerson generator*** *(NW-generator).*

Lemma 6.1. *[NW94] Let m, n, l be integers where $n > l$; let $f : \{0,1\}^m \to \{0,1\}$ be a Boolean function, such that f is average-case n^k-hard for some constant $k \geq 2$; and let A be a Boolean $n \times l$ matrix which is a* ***($\log n, m$)-design****. Then, the NW-generator $G : \{0,1\}^l \to \{0,1\}^n$ given by f_A has standard hardness $H(G) > n$. In other words, for any circuit C of size n,*

$$|\mathbb{P}_{z \in_R \{0,1\}^n}[C(y) = 1] - \mathbb{P}_{x \in_R \{0,1\}^l}[C(G(x)) = 1]| < \frac{1}{n}.$$

Now, as we are clear with the notion of *NW-generator* and *hard on average*, we will explain the meaning of "a hard generator for a propositional proof system".

As shown above, the existence of pseudorandom generators is a useful assumption for conjectures in propositional proof complexity. However, a pseudorandom generator itself can be considered as hard for a propositional proof system. We can ask questions like

If a pseudorandom generator is hard for circuits, is it also hard for some propositional proof systems?

Now, we give the mathematical definition of it.

Definition 6.7. *[ABSRW04, Kra01] **(Generator hard for a proof system).** Given a function $G : \{G_n : \{0,1\}^n \to \{0,1\}^m\}$ where $m > n$, we say G **is a hard generator for a propositional proof system** P if and only if for any sequence of binary strings $b = \{b_m \in \{0,1\}^m\}$, for large enough n, there is no polynomial-size P-proof of the (properly encoded) statement $G_n(x_1, \cdots, x_n) \neq b_m$ where $x_1, \cdots, x_n$ are treated as propositional variables.*

The intuition of this definition is also explained in [ABSRW04]. Since for any function $G_n : \{0,1\}^n \to \{0,1\}^m$, where $m > n$, G_n is not an onto function. Therefore, the definition given above means that the propositional proof system P cannot even efficiently prove that G_n is not an onto function, which is a basic property of G_n.

Krajicek gave another similar suggestion in [Kra01]. Recall the *pigeonhole principle*: if there are m pigeons and we put them into n pigeonholes, where $m > n$, there must exist a pigeonhole that contains more than one pigeon. Now, we formalize this statement into propositional logic. We call this formula the *pigeonhole principle formula*, denoted by $\mathbf{PHP}^m_n$. First, there are $n \times m$ atoms p_{ij}:

$$p_{ij}, \qquad 1 \leq i \leq m, \quad 1 \leq j \leq n.$$

p_{ij} represents that the pigeon i is contained in the pigeonhole j. Then, the $\mathbf{PHP}^m_n$ formula is as following:

$$\neg[\bigwedge_i \bigvee_j p_{ij} \wedge \bigwedge_i \bigwedge_{j \neq j'} \neg(p_{ij} \wedge p_{ij'}) \wedge \bigwedge_j \bigwedge_{i \neq i'} \neg(p_{ij} \wedge p_{i'j})].$$

In this $\mathbf{PHP}^m_n$ formula, $\bigwedge_i \bigvee_j p_{i,j}$ means for every pigeon, at least one pigeonhole contains it. $\bigwedge_i \bigwedge_{j \neq j'} \neg(p_{ij} \wedge p_{ij'})$ means that for every pigeon, it cannot be contained in two pigeonholes, and $\bigwedge_{i=i'} \bigwedge_j \neg(p_{ij} \wedge p_{i'j})$ means that there does not exist a pigeon share a pigeonhole with another pigeon. $\mathbf{PHP}^m_n$ formula states that those statements above cannot happen simultaneously. Proving $\mathbf{PHP}^m_n$ can be viewed as proving the function that maps m pigeons to n pigeonholes is not one-to-one.

For the pigeonhole principle formula $\mathbf{PHP}^m_n$, when $m \geq 2n$, it is called *weak pigeonhole principle formula*, denoted by $\mathbf{WPHP}^m_n$. In [BT88], it has been shown that any Resolution proof system requires at least $\exp(\Omega(\frac{n^2}{m}))$ steps to prove $\mathbf{WPHP}^m_n$. Then, when $m > n^2$, there will be no exponential lower bound results on resolution refutation of $\mathbf{WPHP}^m_n$.

Krajicek [Kra01] considered the dual case of the pigeonhole principle when there are more pigeonholes than pigeons. He considered the statement that if there are m pigeons and we put them into n pigeonholes, where $m^2 = n$, there must exist a pigeonhole that does not contain any pigeon. For a function f, a *dual weak pigeonhole principle* for f is the following first-order formula.

$$\forall\ a > 1\ \mathbf{dPHP}(f)^a_{a^2},$$

where $\mathbf{dPHP}(f)^a_b$ denotes the following first-order formula.

$$\exists\ v < b\ \forall\ u < a\ f(u) \neq v.$$

Then, $\mathbf{dPHP}(f)^a_b$ says that f is not a onto function. For all $a > 1$, fixing $v < a^2$ and using the technique called propositional translation (see [CN10] for detail), if f is in $\mathsf{P/poly}$, we can translate the first-order formula $\forall\ u < a\ f(u) \neq v$ into a propositional formula, which is same as the propositional formula in Definition 6.7 where m is a parameter determined by a^2.

6.4. *Explanation of Razborov's Conjecture*

Now, we formalize Razborov's Conjectures. [i]

Conjecture 7. *Razborov's Conjecture 1.* *Let $m = \{m_i\}_{i\in\mathbb{N}}, n = \{n_i\}_{i\in\mathbb{N}}, l = \{l_i\}_{i\in\mathbb{N}}$ be increasing sequences of integers where $n_i > l_i$ for all $i \in \mathbb{N}$; let $f = \{f_{m_i} : \{0,1\}^{m_i} \to \{0,1\}\}$ be a family of polynomial-time computable Boolean functions, such that f is average-case ${n_i}^k$-hard for any circuit family in $\mathsf{NC}^1/\mathsf{poly}$ for some $k \geq 2$, which means for large enough $i \in \mathbb{N}$, for any circuit C_{m_i}, in a circuit family in $\mathsf{NC}^1/\mathsf{poly}$, of size ${n_i}^k$,*

$$I\!\!P_{x\in_R\{0,1\}^{m_i}}[C(x) = f_{m_i}(x)] < \frac{1}{2} + \frac{1}{{n_i}^k}.$$

And let A_i be a Boolean $n_i \times l_i$ matrix, which is a $(\log n_i, m_i)$-design. Then, $G_i(x) = f_{A_i}(x)$ is hard for the Frege proof system, which means for any Frege proof system P, for any $b \in \{0,1\}^{n_i}$, the (properly encoded) statement $G_i(x_1, \cdots, x_l) \neq b$ does not have polynomial-size P-proof.

[i]Note that in Razborov's original paper [Raz15], he was informal about concepts like "hard on average" and did not give an explicit construction of the NW-generator. We choose a particular setting of parameters for which we formulate Razborov's Conjecture.

Conjecture 8. ***Razborov's Conjecture 2.*** *Let $m = \{m_i\}_{i \in \mathbb{N}}, n = \{n_i\}_{i \in \mathbb{N}}, l = \{l_i\}_{i \in \mathbb{N}}$ be increasing sequences of integers where $n_i > l_i$ for all $i \in \mathbb{N}$; let $f = \{f_{m_i} : \{0,1\}^{m_i} \to \{0,1\}\}$ be a family of Boolean functions in $\mathsf{NP} \cap \mathsf{coNP}$, such that f is average-case ${n_i}^k$-hard for any circuit family in $\mathsf{P/poly}$ for some $k \geq 2$, which means for large enough $i \in \mathbb{N}$, for any circuit C_{m_i}, in a circuit family in $\mathsf{P/poly}$, of size ${n_i}^k$,*

$$\mathbb{P}_{x \in_R \{0,1\}^{m_i}}[C_{m_i}(x) = f_{m_i}(x)] < \frac{1}{2} + \frac{1}{{n_i}^k}.$$

And let A_i be a Boolean $n_i \times l_i$ matrix, which is a $(\log n_i, m_i)$-design. Then, $G_i(x) = f_{A_i}(x)$ is hard for the Extended Frege proof system, which means for any Extended Frege proof system P, for any $b \in \{0,1\}^{n_i}$, the (properly encoded) statement $G_i(x_1, \cdots, x_l) \neq b$ does not have polynomial-size P-proof.

Now, we provide a possible explanation of why Razborov chose complexity classes like $\mathsf{NC}^1/\mathsf{poly}$, $\mathsf{P/poly}$, P and $\mathsf{NP} \cap \mathsf{coNP}$ in these two conjectures. The study of proof complexity has strong connections with the study of circuit complexity, shown in several works, including [DRNV16, DRMN+20, CN10, BP01]. Now, we consider the complexity class NC^1. Since NC^1 contains circuits with polynomial size and $O(\log n)$ depth, then every circuit in NC^1 can be viewed as a polynomial-size propositional formula with connectives $\wedge$, $\vee$ and $\neg$. Therefore, we can view the formulas in a Frege proof as NC^1 circuits. However, for the Extended Frege proof system, since we have the extension rule, we can use this rule to build circuits. The extension rule is crucial. Otherwise, we cannot reuse formulas to build circuits.

The reason why we use the Boolean functions in P and $\mathsf{NP} \cap \mathsf{coNP}$ are as follows: if the function f is not in P or $\mathsf{NP} \cap \mathsf{coNP}$, we do not know how to encode the desired statement $G(x_1, \cdots, x_l) \neq f$ by a propositional formula.

Here, we give a theorem informally stated in [Raz15].

Theorem 6.2. *[Raz15]* ***A consequence of Razborov's Conjectures.*** *Suppose Razborov's Conjecture 1 is true, and Let $m = \{m_i\}_{i \in \mathbb{N}}, n = \{n_i\}_{i \in \mathbb{N}}, l = \{l_i\}_{i \in \mathbb{N}}$ be increasing sequences of integers; Assume the existence of a family of polynomial-time computable Boolean functions $t = \{t_{m_i} : \{0,1\}^{m_i} \to \{0,1\}\}$ such that t is average case $(2^{n_i})^k$-hard for any circuit family in $\mathsf{NC}^1/\mathsf{poly}$ for some $k \geq 2$. Let A_i be a Boolean $2^{n_i} \times l_i$ matrix, which is a (n_i, m_i)-design. Let $G^i = t_{A_i} : \{0,1\}^{l_i} \to \{0,1\}^{2^{n_i}}$. The Boolean function $G^i_x : \{0,1\}^{n_i} \to \{0,1\}$, representing the function on fixed $x \in \{0,1\}^{l_i}$ outputs the $G^i(x)$ at index $y \in \{0,1\}^{n_i}$, is computable by circuits of size s_i. Note that the input of G^i_x is $y \in \{0,1\}^{n_i}$. Then, for large enough $i \in \mathbb{N}$, any Frege system P does not admit polynomial-size proof of $\mathsf{clb}(f_{n_i}, s_i)$ for any $f_n \in \mathcal{F}_{n_i}$.*

Proof. From Lemma 6.1 and Razborov's Conjecture, we know that $G = \{G^i\}_{i \in \mathbb{N}}$ is hard for P, which means for large enough i, for any $b \in \{0,1\}^{n_i}$, the (properly encoded) statement $G^i(x_1, \cdots, x_l) \neq b$ does not have polynomial-size P-proof.

For Boolean function $f_{n_i} \in \mathcal{F}_{n_i}$, the truth table of f_{n_i} is equivalent to $G^i(x)$ for some $x \in \{0,1\}^{l_i}$. Since G^i_x is computable by circuits of size s_i, f_{n_i} is computable by circuits of size s_i. Suppose there exists a Boolean function $f'_{n_i} \in \mathcal{F}_{n_i}$ such that a Frege system P has a polynomial-size proof of $\mathsf{clb}(f'_{n_i}, s_i)$. By substitutions, we can get a polynomial-size proof of $G^i(x_1, \cdots, x_l) \neq f'_{n_i}$ from the polynomial-size P-proof of $\mathsf{clb}(f'_{n_i}, s_i)$, which is a contradiction. $\square$

The above theorem can be viewed as a consequence of Razborov's Conjecture 1. By replacing the Frege system with the Extended Frege proof system and all the other parameters, the consequence of Razborov's Conjecture 2 is similar.

7. *CONNECTION BETWEEN RAZBOROV'S AND RUDICH'S CONJECTURE*

In this chapter, we present the results in [PS19], which shows some connection between Razborov's Conjecture and Rudich's Conjecture. First, we give some necessary concepts from [PS19]. All the N below equals 2^n if there is no additional definition. And the Rudich's Conjecture we mentioned in this chapter is the Conjecture 6.

7.1. Preliminaries

As we have defined the circuit lower bound formula in Definition 5.1 above, we define the proof complexity lower bound formula here.

Definition 7.1. *[PS19] **(Proof complexity lower bound formula)**. Given a propositional proof system R, propositional formula ϕ and size function $s : \mathbb{N} \to \mathbb{N}$, a **proof complexity lower bound formula** $\mathsf{plb}_R(\phi, s)$ is a propositional DNF formula of size $\mathsf{poly}(|\phi| + s)$ over $\mathsf{poly}(|\phi| + s)$ variables expressing that there is no R-proof of ϕ having size s.*

Given a propositional proof system R with advice function $a : \mathbb{N} \to \mathbb{N}$, propositional formula ϕ, size function $s : \mathbb{N} \to \mathbb{N}$ and advice family w of length $a(|\phi|)$, $\mathsf{plb}_R(\phi, s, w)$ is a propositional DNF formula of size $\mathsf{poly}(|\phi| + s)$ over $\mathsf{poly}(|\phi| + s)$ variables, and $\mathsf{plb}_R(\phi, s, w)$ is a tautology if and only if there is no R-proof of ϕ having size s.

As we use pseudorandom generators as the assumption of natural proofs and Rudich's Conjecture, pseudorandomness also plays a vital role in the following proof.

Definition 7.2. *[PS19] **(Hitting and discrepancy set)**. For fixed integers N and t and a parameter $\epsilon \geq 0$, we say that $H_N \subseteq \{0,1\}^N$ is an ϵ-hitting set against size t* (resp. *non-deterministic size t) if $H_N \cap S \neq \emptyset$ for every ϵ-dense* [i] *$S \subseteq \{0,1\}^N$ computable by circuits* (resp. *non-deterministic circuits) of size t. We say that $H_N \subseteq \{0,1\}^N$ is ϵ-discrepancy set against t if for every circuit C of size t on N bits,*

$$|\mathbb{P}_{x \in_R \{0,1\}^N}[C(x) = 1] - \mathbb{P}_{y \in_R H_N}[C(y) = 1]| \leq \epsilon.$$

Given an integer s, we say that an ϵ-hitting set (resp. *an ϵ-discrepancy set) H_N is s-succinct if for each $y \in H_N$, $\mathsf{fn}(y)$ has circuits of size at most s.*

Lemma 7.1. *Let $s : \mathbb{N} \to \mathbb{N}$ and $t : \mathbb{N} \to \mathbb{N}$ be size functions, and let $\epsilon : \mathbb{N} \to \mathbb{R}$ be non-negative. For any $N \in \mathbb{N}$, there are $s(n)$-succinct $\epsilon(N)$-hitting sets over N-bits strings against non-deterministic size $t(N)$ if and only if $\overline{MCSP[s]}$ has no $\epsilon(N)$-dense subsets in non-deterministic size $t(N)$,*

[i]Recall the Definition 5.2.

Proof. "$\rightarrow$": For any $N \in \mathbb{N}$, suppose there are $s(n)$-succinct $\epsilon(N)$-hitting sets H_N over N-bits strings against non-deterministic size $t(N)$, by definition, $H_N \cap S \neq \emptyset$ for every $\epsilon(N)$-dense $S \subseteq \{0,1\}^N$ computable by non-deterministic circuits of size $t(N)$ and for each $y \in H_N$, $\mathsf{fn}(y)$ has circuits of size at most $s(n)$. Suppose $\overline{\mathsf{MCSP}[s(n)]}$ has a $\epsilon(N)$-dense subset S' in non-deterministic size $t(N)$. We know $H_N \cap S' \neq \emptyset$, which means there exists $y \in H_N \cap S' \subseteq H_N$. Since $y \in S' \subseteq \overline{\mathsf{MCSP}[s(n)]}$, $\mathsf{fn}(y)$ does not have circuits of size $s(n)$. However, at the same time, since $y \in H_N$ and H_N is $s(n)$-succinct, $\mathsf{fn}(y)$ has circuits of size at most $s(n)$. This is a contradiction.

"$\leftarrow$": Suppose $\overline{\mathsf{MCSP}[s(n)]}$ has no $\epsilon(N)$-dense subsets in non-deterministic size $t(N)$, then consider $\mathsf{MCSP}[s(n)]$ as our candidate of $s(n)$-succinct $\epsilon(N)$-hitting sets over N-bits strings against non-deterministic size $t(N)$. If there is a $\epsilon(N)$-dense $S \subseteq \{0,1\}^N$ in non-deterministic size $t(N)$ such that $\mathsf{MCSP}[s(n)] \cap S = \emptyset$, we know that S is a $\epsilon(N)$-dense subset of $\overline{\mathsf{MCSP}[s(n)]}$ in non-deterministic size $t(N)$, which contradict with our assumption. Thus, $\mathsf{MCSP}[s(n)]$ is a $s(n)$-succinct $\epsilon(N)$-hitting set over N-bits strings against non-deterministic size $t(N)$. $\square$

Definition 7.3. *[PS19] **(R-easy hitting tautology set)**. Let R be a propositional proof system, $\epsilon : \mathbb{N} \rightarrow \mathbb{R}$ be non-negative, and $s : \mathbb{N} \rightarrow \mathbb{N}$ be a size function. Let $\mathfrak{C}$ be a circuit class. We say W is a set of ϵ-pseudorandom (*resp. *ϵ-hitting) tautologies against $\mathfrak{C}$ that is s-easy for R if there is a polynomial-time computable function $f : \{0,1\}^* \rightarrow \{0,1\}^*$ and a sequence $\{H_N\}$, where for large enough n, $H_N \subseteq \{0,1\}^*$ us an $\epsilon(N)$-discrepancy (resp. $\epsilon(N)$-hitting) set against $\mathfrak{C}$, and moreover:*

1. *$W = \cup_N f(H_N)$*
2. *For all but finitely many $\phi \in W$, we have that $\phi \in \mathsf{TAUT}$*
3. *Each tautology $\phi \in W$ has R-proofs of size $s(|\phi|)$*
4. *$\mathbb{P}_{y \in_R \{0,1\}^N}[f(y) \in \mathsf{TAUT}] = 1 - 1 \backslash N^{\omega(1)}$*
5. *$\mathbb{P}_{y \in_R \{0,1\}^N}[f(y)$ has R-proofs of size $s(|f(y)|)] = o(1)$*

We say a set W is R-easy if W is s-easy for R for some polynomially bounded size function $s : \mathbb{N} \rightarrow \mathbb{N}$.

7.2. Main Theorem

Lemma 7.2. *If Rudich's Conjecture holds, there is a propositional proof system R such that for any k, there is a collection of R-easy $1/N^k$-hitting tautologies against non-deterministic size N^k.*

Proof. As we assume that Rudich's Conjecture holds, there is a constant $c > 0$ such that for every L, if $L \subseteq \overline{\mathsf{MCSP}[n^c]}$ and $L \in \mathsf{NP/poly}$, then L is $1/N^{\omega(1)}$-sparse. Without loss of generality, we assume that $c > 1$.

Claim 1: There exists a constant n_0 such that for all $n > n_0$, $\mathsf{MCSP}[n^c]$ does not have polynomial-size circuits.

Suppose for all constant n_0 there exists a $n > n_0$ such that $\mathsf{MCSP}[n^c]$ can be computed by a polynomial-size circuit, then by flipping the output gate of this circuit, $\overline{\mathsf{MCSP}[n^c]}$ can be computed by a polynomial-size circuit. Hence, $\overline{\mathsf{MCSP}[n^c]} \in \mathsf{P/poly} \subseteq \mathsf{NP/poly}$.

Now, we prove that $\overline{\mathsf{MCSP}[n^c]}$ is not $1/N^{\omega(1)}$-sparse with a simple counting argument. For a Boolean circuit of size n^c, we gave each gate a number. We can use $\lceil \log(n^c) \rceil$ bits to give the number representing the gate. Each gate has at most two inputs and one output, and we can use $3\lceil \log(n^c) \rceil$ bits to represent a node. Therefore, we can use $3n^c\lceil \log(n^c) \rceil$ to represent a Boolean circuit of size n^c, which implies there are at most $2^{3n^c\lceil \log(n^c) \rceil}$ many Boolean circuits of size n^c. Since there are 2^{2^n} many Boolean functions with n variables, $\overline{\mathsf{MCSP}[n^c]}_n = \overline{\mathsf{MCSP}[n^c]} \cap \{0,1\}^{2^n}$ has at least $2^{2^n} - 2^{3n^c\lceil \log(n^c) \rceil}$ elements, which implies $\overline{\mathsf{MCSP}[n^c]}$ is definitely not $1/N^{\omega(1)}$-sparse.

As a result, we know that $\overline{\mathsf{MCSP}[n^c]}$ does not satisfy Rudich's Conjecture, which is a contradiction.

There are a few things we need to construct to get the R-easy $1/N^k$-hitting tautologies against non-deterministic size N^k: a propositional proof system R, a $1/N^k$-hitting set against non-deterministic size N^k, a polynomial-time computable function f maps the hitting sets to tautologies.

Now, we define a sequence $\{z_N\}$ of binary strings, which will be used to define the hitting set. For each $N > 0$, z_N is a binary string of length N. If there exists an integer $n \in \mathbb{N}$ such that $N = 2^n$, then z_N is the binary string that represents the truth table of $\mathsf{MCSP}[m^c]$ on inputs of size n where $m = \lfloor \log n \rfloor$. Otherwise, $z_N = z_{2^{\lfloor \log(N) \rfloor}} 0^{N - 2^{\lfloor \log(N) \rfloor}}$.

Claim 2: for any polynomially bounded function $t : \mathbb{N} \to \mathbb{N}$, for any sufficiently large N, $\mathsf{fn}(z_N)$ does not have circuits of size $t(n)$.

Suppose there is a polynomially bounded function t such that for infinitely many N, $\mathsf{fn}(z_N)$ has circuits of size $t(n)$. Let $n' = \lfloor \log(N) \rfloor$, $2^{n'} = N'$, $m' = \lfloor \log(n') \rfloor$. Since $\mathsf{fn}(z_N)$ has circuits of size $t(n)$, by definition of fn and z_N, $\mathsf{fn}(z_{N'}) = \mathsf{MCSP}[m'^c]$ has circuits of size $t(n)$. With a similar counting argument as above, we can show that $\overline{\mathsf{MCSP}[m'^c]} \cap \{0,1\}^{2^n}$ is not $1/N^{\omega(1)}$-sparse, which contradicts the assumption that Rudich's Conjecture holds.

Since Rudich's Conjecture holds, there is no $1/N^k$-dense subset of $\overline{\mathsf{MCSP}[n^c]}$, which is computable by non-deterministic circuits of size N^k. By Lemma 7.2, there exists a n^c-succinct $1/N^k$-hitting set $H_N \subseteq \{0,1\}^N$ against non-deterministic size N^k.

Now, we construct the hitting set H'_N. We define a sequence of sets $\{H'_N\}$ such that for any N, $H'_N = \{y \oplus z_N | y \in H_N\}$.

Claim 3: $\{H'_N\}$ is a sequence of $1/N^k$-hitting sets against $\mathbf{NSIZE}(N^k)$.

Suppose it is not. To be specific, there exist infinitely many N such that there is a $1/N^k$-dense $S'_N \subseteq \{0,1\}^N$ that is computable by $\mathbf{NSIZE}(N^k)$ and $S'_N \cap H'_N = \emptyset$. We construct $S_N = \{y \oplus z_N | y \in S'_N\}$. For any $y, y' \in S'_N$ such that $y \neq y'$, $y \oplus z_N \neq y' \oplus z_N$. Because of this, $|S_N| = |S'_N|$. Also, if $z_i = 1$[ii], we flip the input of the non-deterministic circuit that computes S'_N to get a non-deterministic circuit with the same size that computes S_N. As a result, we know that S_N is also a $1/N^k$-dense subset of $\{0,1\}^N$ computable by $\mathbf{NSIZE}(N^k)$.

If there exists a $y \in S_N \cap H_N$, then $y \oplus z_N \in H'_N \cap S'_N$, which contradicts our assumption that $H'_N \cap S'_N = \emptyset$. Hence, we know that there exists a $1/N^k$-dense $S_N \subseteq \{0,1\}^N$ computable by $\mathbf{NSIZE}(N^k)$ such that $H_N \cap S_N = \emptyset$, which implies H_N is not a $1/N^k$-hitting set against $\mathbf{NSIZE}(N^k)$. That is a contradiction.

Now, we construct the function $f : \{0,1\}^* \to \{0,1\}^*$ that maps hitting sets to hitting tautologies and the propositional proof system R. We define $f(x) = \mathsf{clb}(\mathsf{fn}(x), \lfloor \log(|x|) \rfloor^c)$. We define the propositional proof system R as follows:

for any $u, v \in \{0,1\}^*$, $R(u,v) = 1$ if and only if

- $u \in \mathsf{TAUT}$ and $v = 01^{2^{|u|}}$ or
- $u = \mathsf{clb}(\mathsf{fn}(z_N \oplus y), n^c)$ and $v = 1\mathsf{enc}(C)$[iii] where C is a Boolean circuit with n inputs of size at most n^c and $\mathsf{fn}(y)$ is computable by C. Also, $N = 2^n$ is an integer greater or equal to N_0 where N_0 is a constant such that for any $N \geq N_0$, $\mathsf{fn}(z_N)$ does not have circuits of size $2n^c + 5$. Due to claim 2, such N_0 exists.

Now, we need to prove that the propositional proof system R we defined is indeed a valid propositional proof system (*i.e.* R satisfies completeness, soundness and polynomial-time computability).

Claim 4: R satisfies completeness, soundness and polynomial-time computability.

- **Completeness:** Since for any $u \in \mathsf{TAUT}$, there is a R-proof $v = 01^{2^{|u|}}$ by definition. The completeness follows.
- **Soundness:**

[ii] z_i is the bit of z_N at index i.

[iii] $\mathsf{enc}(C)$ means the encoding of a circuit C.

  - Since $u \in \mathsf{TAUT}$, we know that in this case, R is sound.
  - First, we can observe that $\mathsf{fn}(z_N \oplus y) = \mathsf{fn}(z_N) \oplus \mathsf{fn}(y)$. Similarly, $\mathsf{fn}(z_N) = \mathsf{fn}(z_N \oplus y) \oplus \mathsf{fn}(y)$. Suppose $\mathsf{clb}(\mathsf{fn}(z_N \oplus y), n^c)$ is not a tautology. In other words, $\mathsf{fn}(z_N \oplus y)$ has a circuit C' of size n^c. Since $v = 1\mathsf{enc}(C)$ where $\mathsf{enc}(C)$ gives a witness showing that $\mathsf{fn}(y)$ has a circuit C of size n^c. By combining C and C' with an XOR-gate, which can be implemented by two negation gates, two AND gates and one OR gate, $\mathsf{fn}(z_N)$ can be computed by this circuit of size $2n^c + 5$. This is a contradiction since $N \geq N_0$ and $\mathsf{fn}(z_N)$ does not have circuits of size $2n^c + 5$ by definition. Therefore, $\mathsf{clb}(\mathsf{fn}(z_N \oplus y), n^c)$ are tautologies. As a result, the soundness follows.

- **Polynomial-time computability:** Given $u, v \in \{0,1\}^*$,
  - If v starts with 0, check whether $u \in \mathsf{TAUT}$ with brute force. This takes time $2^{O(|u|)}$. Since v has length $2^{|u|} + 1$, this case is polynomial-time computable.
  - If v starts with 1, we first check that $\mathsf{enc}(C)$ is a proper encoding of a circuit C, which takes $\mathsf{poly}(|v|)$. We need to check that z_N is the truth table of $\mathsf{MCSP}[m^c]$ where $m = \lfloor \log(n) \rfloor$. Using the counting argument, there are at most $2^{3m^c \log(m^c)}$ many possible circuits of size at most m^c. With brute force, it takes $2^{3m^c \log(m^c)} \cdot 2^m$ times to check whether a binary string of length n is in $\mathsf{MCSP}[m^c]$. Hence, it takes $2^{3m^c \log(m^c)} \cdot 2^m \cdot 2^n$ time to construct the truth table z_N of $\mathsf{MCSP}[m^c]$ on input of length n. Recall the Definition 5.1, $|u| = \tilde{O}(2^n n^{3c})$. Therefore, the process of construction of z_N is in polynomial time. After constructing z_N, we can use z_N to reveal y in polynomial time. Using brute force, we can use $2^{O(n)}$ time to check whether C computes $\mathsf{fn}(y)$. Now, we can conclude that this case is also polynomial-time computable.

Now, with the function f, the hitting sets $\{H'_N\}$ and the propositional proof system R, we define the hitting tautologies $W = \cup_N f(H'_N)$[iv].

Claim 5: W is a set of $1/N^k$-hitting tautologies against $\mathbf{NSIZE}(N^k)$ that is s-easy for R for some polynomially bounded size function $s : \mathbb{N} \to \mathbb{N}$.

1. Condition 1 follows from our definition of W.
2. By claim 2 above, if we take $t(n)$ to be $2n^c + 5$, for sufficiently large N, $\mathsf{fn}(z_N)$ does not have circuit of size $2n^c+5$. We claim that for large enough N, $\mathsf{fn}(y \oplus z_N)$ does not have circuits of size n^c. Otherwise, suppose $\mathsf{fn}(y \oplus z_N)$ has a circuit C

[iv] $f(H'_N) = \{f(x) | x \in H'_N\}$

of size n^c. Recall that H_N is a n^c-succinct $1/N^k$-hitting subset of $\{0,1\}^N$ against non-deterministic size N^k, which implies $\mathsf{fn}(y)$ is computable by a circuit C' of size n^c where $y \in H_N$. By combining C and C' with an XOR gate, which can be implemented by two negation gates, two AND gates and one OR gate, $\mathsf{fn}(z_N)$ can be computed by this circuit of size $2n^c + 5$. Therefore, except for finitely many $x \in W$, $x \in \mathsf{TAUT}$.

3. Now, we show that each tautology $\phi \in W$ has a R-proof of polynomially bounded size. Because H_N is n^c-succinct, for any $y \in H_N$, there must exist a circuit C of size at most n^c that computes $\mathsf{fn}(y)$ by definition. For any $N \geq N_0$, $v = 1\mathsf{enc}(C)$ is a valid proof of $u = \mathsf{clb}(\mathsf{fn}(z_N \oplus y), n^c)$. Since C has size n^c, $|v| = \mathsf{poly}(n) < |u|$. Moreover, for any $N < N_0$, since we only consider tautologies in W, $v = 01^{2^{|u|}}$ is a valid R-proof of u. Since N_0 is a constant, the length of those proofs is bounded by a constant. Hence, we can conclude that for any tautology $\phi \in W$, the length of the R-proof of ϕ is bounded by some polynomial.

4. Now, we need to show that for any binary string $y \in_R \{0,1\}^N$, $f(y)$ is a tautology with a large probability. Using a similar counting argument as we used above, the probability that $\mathsf{fn}(y)$ can be computed by a circuit of size n^c is about $\frac{2^{\mathsf{poly}(n)}}{2^{2^n}}$. Therefore, condition 4 follows.

5. Suppose W and R do not satisfy condition 5. In other words, there exists a constant $\gamma \in (0,1)$ and a integer d such that

$$\mathbb{P}_{y \in_R \{0,1\}^N}[f(y) \text{ has } R\text{-proofs of size } |f(y)|^d] \geq \gamma$$

   Now, we define a language L such that $y \in L$ if and only if $f(y)$ has R-proofs of size $|f(y)|^d$. By the soundness of R, we know that $\mathsf{fn}(y)$ does not have circuits of size n^c, which implies $L \subseteq \overline{\mathsf{MCSP}[n^c]}$. Also, $L \in \mathsf{NP} \subseteq \mathsf{NP/poly}$ since we can guess the polynomial-size witness of any instance x, which is the R-proof of x. Also, by the probability γ, we know that L is not $1/N^{\omega(1)}$-sparse. Therefore, L can be used to refute Rudich's Conjecture, which is a contradiction.

Now, we can conclude that if Rudich's Conjecture holds, there is a propositional proof system R such that for any k, there is a collection of R-easy $1/N^k$-hitting tautologies against non-deterministic size N^k. □

Now, we consider the case when we take Rudich's Conjecture itself as a hard formula.

Definition 7.4. ***(Rudich's Conjecture admits feasible propositional proofs)****.* *We say that* ***Rudich's Conjecture admits feasible propositional proofs*** *if for every large enough integer $d > 0$ and every propositional proof system R with polynomial advice, there is a propositional proof system S, such that if $\{w_m\}$ is a sequence of polynomial-size advice strings good for R, then for all large enough n, for a $1 - o(1)$ fraction of Boolean functions $f_n \in \mathcal{F}_n$, there are polynomial-size S-proofs of $\mathsf{plb}_R(\mathsf{clb}(f_n, n^d), m^d, w_m)$, where $m = |\mathsf{clb}(f_n, n^d)|$.*

Note that in Rudich's Conjecture 5, we require a $1 - 1/N^{\omega(1)}$ fraction of Boolean functions. But in here, we only require $1 - o(1)$ fraction of Boolean functions, which means it is possible that Rudich's Conjecture is false and Rudich's Conjecture admits feasible propositional proofs at the same time.

Lemma 7.3. *If Rudich's Conjecture holds, then Rudich's Conjecture does not admit feasible propositional proofs.*

Proof. Since Rudich's Conjecture holds, by Lemma 7.2, there is a propositional proof system R such that for any k, there is a collection of R-easy $1/N^k$-hitting tautologies against non-deterministic size N^k. In this proof, we use the same R, W, $\{H'_N\}$ and f as we defined in the proof of Lemma 7.2.

Suppose Rudich's Conjecture admits feasible propositional proofs. In other words, for any large enough $d > 0$, there is a propositional proof system S, such that for all large enough n, for a $1 - o(1)$ fraction of Boolean functions $f_n \in \mathcal{F}_n$, there are polynomial-size S-proofs of $\mathsf{plb}_R(\mathsf{clb}(f_n, n^d), m^d)$, where $m = |\mathsf{clb}(f_n, n^d)|$. Therefore, for every large enough N, there exists a constant b_N, for a $1-o(1)$ fraction of binary strings $y \in \{0,1\}^N$, the length of the S-proof of $\mathsf{plb}_R(\mathsf{clb}(\mathsf{fn}(y), n^d), m^d)$ is bounded by $r_N^{b_N}$ where $r_N = |\mathsf{plb}_R(\mathsf{clb}(\mathsf{fn}(y), n^d), m^d)|$ and $m_N = |\mathsf{clb}(\mathsf{fn}(y), n^d)|$. Since m_N is bounded by some polynomial of N and r_N is bounded by some polynomial of m_N, there exists a constant a_N such that $N^{a_N} > {r_N}^{b_N}$.

Now, we define a new language L_S based on S. Given any binary string $y \in \{0,1\}^N$, $y \in L_S$ if and only if there is a S-proof of $\mathsf{plb}_R(\mathsf{clb}(\mathsf{fn}(y), n^d), m^d)$ of size N^{a_N}. By the above argument, we know that for all large enough N, L_S is $1 - o(1)$-dense.

Now, we argue that $L_S \in \mathsf{NP}$. Given any $y \in \{0,1\}^N$, we first construct the formula $\phi = \mathsf{plb}_R(\mathsf{clb}(\mathsf{fn}(y), n^d), m^d)$ in time $\mathsf{poly}(N)$. After that, guess the S-proof p of $\mathsf{plb}_R(\mathsf{clb}(\mathsf{fn}(y), n^d), m^d)$ whose length is also polynomially bounded (*i.e.*, $|p| = \mathsf{poly}(|\phi|)$). By the polynomial-time computability of S, we can verify it in time $\mathsf{poly}(|\phi| + |p|)$, which is still polynomial to N. Therefore, $L_S \in NP$, which implies there exists a constant k such that L_S can be computed by non-deterministic circuits of size N^k.

From the above argument, we can conclude that L_S is $1 - o(1)$-dense and computable by non-deterministic circuits of size N^k. Note that $\{H'_N\}$ are $1/N^k$-hitting sets against $\mathbf{NSIZE}(N^k)$. For sufficiently large N, there exists a sequence $\{y_N\}$ such that $y_N \in L_S \cap H'_N \cap \{0,1\}^N$. Therefore, by W's definition, $f(y_N) \in W$. Recall the proof of condition 3 of W with R in Lemma 7.2, for large enough N, tautologies $\phi \in W$ has R-proof of length less than $|\phi|$. Obviously, ϕ has R-proof of length less than $|\phi|^d$. Hence, $\mathsf{plb}_R(f(y_N), |f(y_N)|^d) = \mathsf{plb}_R(\mathsf{clb}(\mathsf{fn}(y_N), n^d), m^d)$ is not a tautology. By the soundness of S, S does not have polynomial-size proof of $\mathsf{plb}_R(f(y_N), |f(y_N)|^d)$. In fact, S does not even have proof of $\mathsf{plb}_R(f(y_N), |f(y_N)|^d)$. However, since $y_N \in L_S$, by definition of L_S, $\mathsf{plb}_R(f(y_N), |f(y_N)|^d)$ has S-proofs of size N^{a_N}, which is a contradiction.

Now, we can conclude that if Rudich's Conjecture holds, then Rudich's Conjecture does not admit feasible propositional proofs. □

Theorem 7.4. *Rudich's Conjecture does not admit feasible propositional proofs.*

Proof. We will use a win-win argument to prove this theorem. Suppose Rudich's Conjecture holds. By Lemma 7.3, Rudich's Conjecture does not admit feasible propositional proofs.

Now, suppose Rudich's Conjecture does not hold. For the sake of contradiction, we also assume that Rudich's Conjecture admits feasible propositional proofs. Since Rudich's Conjecture does not hold, we know that for all constant $c > 0$, there exists a language L_c and a constant a such that $L_c \subseteq \overline{\mathsf{MCSP}[n^c]}$, $L_c \in \mathsf{NP/poly}$ and L_c is $1/N^a$-dense. We construct a new language $L'_{c'}$ as follows: $L'_{c'}$ only defines on inputs of length $N' = N^{a+1}$ where $N = 2^n$ for some positive integer n. For any binary string y of length N', y can be viewed as N^a blocks where each block y_i for an integer $1 \le i \le N^a$ is a binary string of length N. Then, $y \in L'_{c'}$ if and only if there exists an integer $1 \le i \le N^a$ such that $y_i \in L_{c'+1}$.

Claim 1: for any constant $c' > 0$, there exists a constant $\gamma > 0$ such that $L'_{c'} \subseteq \overline{\mathsf{MCSP}[n'^{c'}]}$, $L'_{c'} \in \mathsf{NP/poly}$ and $L'_{c'}$ is γ-dense for infinitely many N'.

First, we show that $L'_{c'} \subseteq \overline{\mathsf{MCSP}[n'^{c'}]}$. In other words, for $y \in L'_{c'}$, $\mathsf{fn}(y)$ does not have circuits of size $n'^{c'}$. Since $y \in L'_{c'}$, there exists a $1 \le i \le N^a$ such that $y_i \in L_{c'+1}$. By definition of $L_{c'+1}$, $\mathsf{fn}(y_i)$ does not have circuits of size $n^{c'+1}$. Now, we claim that $\mathsf{fn}(y)$ does not have circuits of size $n^{c'+1}$. Otherwise, suppose there exists a circuit C of size $n^{c'+1}$ that computes $\mathsf{fn}(y)$. Since $N = 2^n$ and $N' = N^{a+1}$, the input size of C is $(a+1)n$. By fixing na many bits of C to equal the binary representation of i, we can use C to compute $\mathsf{fn}(y_i)$, which is a contradiction. Therefore, $\mathsf{fn}(y)$ does not have circuits of size $n^{c'+1}$. Since $2^{n'} = N' = 2^{n(a+1)}$ and a is a constant, then for large enough n, $n^{c'+1} > ((a+1)n)^{c'} = n'^{c'}$. Now, we can conclude that $L'_{c'} \subseteq \overline{\mathsf{MCSP}[n'^{c'}]}$.

After that, we show that $L'_{c'} \in \mathsf{NP/poly}$. We can decide $L'_{c'}$ is the following way: given a binary string y, breaks y into N^a blocks and use the non-deterministic circuits C that computes $L_{c'+1}$ to decide whether $y_i \overset{?}{\in} L_{c'+1}$. Then, use $a\log(N)$ many OR gates to connect those N^a many outputs of C. Suppose the size of C is N^k for some constant k, the size of the non-deterministic circuit that computes $L'_{c'}$ as we defined above is $N^a \cdot N^k + a\log(N) = N'^{\frac{a+k}{a+1}} + a\log(N)$ which is polynomial to N'. Therefore, we can conclude that $L'_{c'} \in \mathsf{NP/poly}$.

Finally, we show that $L'_{c'+1}$ is γ-dense for infinitely many N'[v]. By definition, $L_{c'+1}$ is $1/N^a$-dense for all large enough N. Thus, the probability that for all $1 \le i \le N^a$, $y_i \notin L_{c'+1}$, is at most $(1-\frac{1}{N^a})^{N^a} \le \frac{1}{e}$. Given a $y \in_R \{0,1\}^{N'}$, the probability that $y \in L'_{c'+1}$ is at least $1-\frac{1}{e}$, which means $L'_{c'+1}$ is $1-\frac{1}{e}$-dense for infinitely many N'.

Since $L'_{c'} \in \mathsf{NP/poly}$, there exists a non-deterministic Turing machine M that decides $L'_{c'}$ in time at most N'^q with advice of length at most N'^q for some constant q on input of size N'. By choosing $a \ge k$, $L'_{c'}$ can be computed by circuits of size at most $N'^{\frac{a+k}{a+1}} + a\log(N) \le N'^3$. Then, we choose the good advice of M to be the representation of the circuit that computes $L'_{c'}$ and M just constructs the circuit from the advice and runs the input on the circuit. Since the size of the circuit is bounded by N'^3, we can choose c' to be greater than q.

Now, we define a propositional proof system Q with advice such that Given inputs x, y, z, $Q(x,y,z) = 1$ if and only if

- $x \in \mathsf{TAUT}$ and $y = 1^{2^{|x|}}$ or
- $x = \mathsf{clb}(\mathsf{fn}(u), n'^{c'})$ for some $u \in \{0,1\}^{N'}$ and $M(u,y,z) = 1$.

Claim 2: Q is a valid propositional proof system.

- **Completeness:** This is trivial since each tautology has a Q-proof by definition.
- **Soundness:**
  - This first case is trivial.
  - Since $M(u,y,z) = 1$ and M decides $L'_{c'}$, $u \in L'_{c'}$. Recall that $L'_{c'} \subseteq \overline{\mathsf{MCSP}[n'^{c'}]}$, which implies $\mathsf{fn}(u)$ does not have circuits of size $n'^{c'}$. Then, $\mathsf{clb}(\mathsf{fn}(u), n'^{c'})$ is also a tautology.
- **Polynomial-time computability:**
  - Using brute force, try every possible assignment of x. This takes $2^{O(|u|)}$ which is polynomial to $|y|$.
  - Since M is a non-deterministic Turing machine that decides $L'_{c'}$ in polynomial-time and $|x| = \mathsf{poly}(|u|)$, this case is also in polynomial-time.

[v]Although the definition of ϵ-dense requires for all large enough N, the N-slice of language is ϵ-dense. We relax the definition here since it is sufficient for our following proof to have infinitely many N' where the N'-slice of language is ϵ-dense.

Claim 3: for infinitely many N', for at least $1 - \frac{1}{e}$ fraction of $u \in \{0,1\}^{N'}$, $\mathsf{clb}(\mathsf{fn}(u), n'^{c'})$ has $m^{c'}$ size Q-proof, where $m = |\mathsf{clb}(\mathsf{fn}(u), n'^{c'})|$.

Recall that $L'_{c'}$ is $1 - \frac{1}{e}$-dense for infinitely many N'. For any $u \in L'_{c'} \cap \{0,1\}^{N'}$, M accepts u in time at most N'^q. Since we choose $c' > q$, this implies the witness y of x, which is the Q-proof of x, has length at most $N'^{c'}$. Since $m > N'$, we know that for any $u \in L'_{c'} \cap \{0,1\}^{N'}$, u has Q-proof of size at most $m^{c'}$. As a result, the claim follows.

Since we assume that Rudich's Conjecture admits feasible propositional proofs, there exists a propositional proof system S such that for $\{w_m\}$ be a sequence of polynomial-size good advice for Q, for sufficiently large n', for $1 - o(1)$ fraction of Boolean function $f_{n'} \in \mathcal{F}_{n'}$, there are polynomial-size S-proof of $\mathsf{plb}_Q(\mathsf{clb}(f_{n'}, n'^{c'}), m'^{c'}, w_m)$ where $m = |\mathsf{clb}(f_{n'}, n'^{c'})|$. By the soundness of S, we know that for $1 - o(1)$ fraction of Boolean function $f_{n'} \in \mathcal{F}_{n'}$, $\mathsf{clb}(f_{n'}, n'^{c'})$ does not have Q-proof of size $m'^{c'}$. However, in the last paragraph, we have just shown that for infinitely many N', for at least $1 - \frac{1}{e}$ fraction of $u \in \{0,1\}^{N'}$, $\mathsf{clb}(\mathsf{fn}(u), n'^{c'})$ has $m^{c'}$ size Q-proof, where $m = |\mathsf{clb}(\mathsf{fn}(u), n'^{c'})|$. This is a contradiction.

Now, we can conclude that Rudich's Conjecture does not admit feasible propositional proofs. □

Now, we focus on the Extended Frege proof system and show a connection between Rudich's Conjecture and Razborov's Conjecture 2 (Conjecture 8). We use EF to denote an Extended Frege proof system.

Lemma 7.5. *Assume there is an s-size EF-proof of $\mathsf{clb}(f_n, 3n^k)$ for some k and sufficiently large n. Further, let g_n be a function computable by a circuit of size n^k. There is a $\mathsf{poly}(s + K2^n)$-size EF-proof of $\mathsf{clb}(f_n \oplus g_n, n^k)$ where K is an absolute constant.*

Proof. Assume there is an s-size EF-proof of $\mathsf{clb}(f_n, 3n^k)$ for some k and sufficiently large n. By Theorem 4.7, we know that there is an $\mathsf{poly}(s)$-size ER-proof of $\mathsf{clb}(f_n, 3n^k)$ for the same k and sufficiently large n. Instead of giving a derivation of $\mathsf{clb}(f_n \oplus g_n, n^k)$, we give a refutation of the negation of $\mathsf{clb}(f_n \oplus g_n, n^k)$.

To be specific, we let a set of clauses S_0 to express $C_0 = f_n \oplus g_n$ where C_0 is a circuit of size n^k and a set of clauses S_1 to express $C_1 = g_n$ where C_1 is a n^k-size circuit that computes g_n. In other words, S_0 is the negation of $\mathsf{clb}(f_n \oplus g_n, n^k)$. For a binary string $a \in \{0,1\}^n$, we use u_0^a to denote the output of C_0 on a and u_1^a to denote the output of C_1 on a. We define y^a as a new variable for each $a \in \{0,1\}^n$. We let a set of clauses Y to express $y^a = u_0^a \oplus u_1^a$.

Now, from S_0, S_1, and Y, an Extended Resolution proof system can derive a new set of clauses $S_\oplus$ expressing that $C_0 \oplus C_1 = f_n$ in size $K2^n$ where K is a constant. Specifically, for each $a \in \{0,1\}^n$, ER derives $f_n(a) = u_0^a \oplus u_1^a = y^a$ from $u_0^a = f_n(a) \oplus g_n(a)$ in S_0 and $u_1^a = g_n(a)$ in S_1. Since C_0 and C_1 both have size at most n^k, $C_0 \oplus C_1$ has size at most $3n^k$. Consequently, by our assumption that there is an $\mathsf{poly}(s)$-size ER-proof of $\mathsf{clb}(f_n, 3n^k)$, the size of the refutation of $S_\oplus$ is $\mathsf{poly}(s)$. At this point, we can refute $S_0 \cup S_1 \cup Y$ in size $\mathsf{poly}(s) + K2^n$.

Now, we have to fix the variables that represent the circuit C_1. After the substitution of the circuit C_1, there will be no S_1 at all. We only have $S_0 \cup Y'$ where the clauses in Y' express that $y^a = u_0^a$ or $y^a = \neg u_0^a$. Since we can refute $S_0 \cup S_1 \cup Y$ in size $\mathsf{poly}(s) + K2^n$, we have a ER-refutation of $S_0 \cup Y'$ in size $\mathsf{poly}(s) + K2^n$.

Finally, we need to get rid of the set of clauses Y' to get a ER-refutation of S_0. Therefore, we substitute y^a with u_0^a or $\neg u_0^a$ correspondingly. If $y^a = u_0^a$, we substitute y^a by u_0^a. Otherwise, we substitute y^a by $\neg u_0^a$. In this way, we will have a $\mathsf{poly}(s) + K2^n$ size ER-refutation of $S_0 \cup Y''$ where the clauses in Y'' just express $u_0^a = u_0^a$ or $\neg u_0^a = \neg u_0^a$. It can be observed that if you have an $\mathsf{poly}(s) + K2^n$-size ER-refutation of a set of clauses, including clauses of the form $u_0^a = u_0^a$ or $\neg u_0^a = \neg u_0^a$, we can obtain an $\mathsf{poly}(s)+K2^n$-size ER-refutation without using these clauses. Hence, there is a $\mathsf{poly}(s) + K2^n$-size ER-refutation of S_0.

Now, by Theorem 4.7 again, we turn our $\mathsf{poly}(s)+K2^n$-size ER-proof to a $\mathsf{poly}(s+K2^n)$-size EF-proof. The lemma follows. □

Finally, we arrive at our main theorem.

Theorem 7.6. *Assuming Rudich's Conjecture, at least one of the following is true:*

1. *There is no sequence of Boolean functions $\{f_n | f_n \in \mathcal{F}_n\}$, such that $\mathsf{clb}(f_n, n^k)$ has polynomial-size EF-proofs for every $k > 0$.*

2. *There is no propositional proof system Q such that there are polynomial-size Q-proofs of $\mathsf{plb}_{\mathsf{EF}}(\mathsf{clb}(f_n, n^k), m^k)$, where $m = |\mathsf{clb}(f_n, n^k)|$ for a $2^{-O(n)}$ fraction of Boolean functions f_n for all constant k and all large enough n.*

Proof. Suppose there is a consequence of Boolean functions $\{f_n\}$ such that $\mathsf{clb}(f_n, n^k)$ has polynomial-size EF-proofs for every $k > 0$, which means the first item is false. We can replace the propositional proof system R we defined in the proof of Lemma 7.2 with EF. The only two things we need to check are that W and EF satisfy condition 3 and condition 5 of the EF-easy hitting tautologies.

- **Condition 3:** We show that each tautology $\phi \in W$ has a EF-proof of polynomially bounded size. Because H_N is n^c-succinct, for any $y \in H_N$, there must exist a circuit C of size at most n^c that computes $\mathsf{fn}(y)$ by definition. Since we assume that $\{f_n\}$ is a sequence of Boolean functions such that $\mathsf{clb}(f_n, n^k)$ has polynomial-size EF-proofs for every $k > 0$. Moreover, since $\mathsf{clb}(f_n, n^k)$ implies $\mathsf{clb}(f_n, 3n^k)$, we can use the proof of $\mathsf{clb}(f_n, n^k)$ as the proof of $\mathsf{clb}(f_n, 3n^k)$. By Lemma 7.5, by taking g_n to be $\mathsf{fn}(y)$, there is a $\mathsf{poly}(s + K2^n)$-size EF-proof of $\mathsf{clb}(f_n \oplus g_n, n^k)$ where K is an absolute constant, which implies the condition 3 is satisfied.

- **Condition 5:** Suppose W and EF do not satisfy condition 5. In other words, there exists a constant $\gamma \in (0,1)$ and a integer d such that

$$\mathbb{P}_{y \in_R \{0,1\}^N}[f(y) \text{ has } \mathsf{EF}\text{-proofs of size } |f(y)|^d] \geq \gamma$$

  Now, we define a language L such that $y \in L$ if and only if $f(y)$ has EF-proofs of size $|f(y)|^d$. By the soundness of EF, we know that $\mathsf{fn}(y)$ does not have circuits of size n^c, which implies $L \subseteq \overline{\mathsf{MCSP}[n^c]}$. Also, $L \in \mathsf{NP} \subseteq \mathsf{NP/poly}$ since we can guess the polynomial-size witness of any instance x, which is the EF-proof of x. Also, by the probability γ, we know that L is not $1/N^{\omega(1)}$-sparse. Therefore, L can be used to refute Rudich's Conjecture, which is a contradiction.

Therefore, recall the proof of Lemma 7.3, we have shown that assume Rudich's Conjecture, for $R = \mathsf{EF}$, there is no propositional proof system Q such that for $1 - 1/N^{\omega(1)}$ fraction of Boolean functions $f_n \in \mathcal{F}_n$, there are polynomial-size Q-proof of $\mathsf{plb}_R(\mathsf{clb}(f_n, n^k), m^k)$, where $m = |\mathsf{clb}(f_n, n^k)|$ for all large enough n and all constant k. This is equivalent to the second item. □

Recall the Theorem 6.2. We can see that the first item of the above theorem is a consequence of Razborov's Conjecture for EF (Conjecture 8). Therefore, the above theorem can be interpreted as:

Assuming Rudich's Conjecture, we have at least one of the following consequences:

- for an Extended Frege proof system EF, and any Boolean function f, $\mathsf{EF} \nvdash_E \mathsf{clb}(f)$, which is one of the consequences of Razborov's Conjecture for EF under the hardness assumption.
- for every propositional proof system Q, for a fraction of Boolean functions, $Q \nvdash_E \mathsf{plb}(\mathsf{clb}(f))$.

Also, assuming Super-bit Conjecture, by Theorem 5.3, Rudich's Conjecture holds. Suppose the second item of Theorem 7.6 is false, which is that

there is no propositional proof system Q such that for a $2^{-O(n)}$ fraction of Boolean functions f_n, for all constant k and all large enough n, there are polynomial-size Q-proofs of $\mathsf{plb}_{\mathsf{EF}}(\mathsf{clb}(f_n, n^k), m^k)$, where $m = |\mathsf{clb}(f_n, n^k)|$.

. In other words,

there is a propositional proof system Q such that for a $2^{-O(n)}$ fraction of Boolean functions f_n, for all constant k and all large enough n, $\mathsf{plb}_{\mathsf{EF}}(\mathsf{clb}(f_n, n^k), m^k)$ has polynomial-size Q-proofs, where $m = |\mathsf{clb}(f_n, n^k)|$.

This can be interpreted as Rudich's Conjecture for EF admits feasible propositional proofs. Since the second item of Theorem 7.6 does not hold, by Theorem 7.6, the first item of Theorem 7.6, which is a consequence of Razborov's Conjecture for EF (Conjecture 8), must hold.

Therefore, we can interpret our Theorem 7.6 as

> assuming Super-bit Conjecture if Rudich's Conjecture for EF admits feasible propositional proofs, then an influential consequence of Razborov's Conjecture for EF must hold.

8. DISCUSSION

In the last chapter, we will present some discussion and open problems that are crucial.

Self-defeating nature of circuit lower bounds and proof complexity lower bounds. Natural proofs [Rud97] presents a barrier result for proving circuit lower bounds. We can view the assumption of natural proofs, which is the existence of a strong pseudorandom generator, as a circuit lower bound. This is because the existence of a strong PRG implies that there does not exist a polynomial-size circuit that can distinguish pseudorandom strings from truly random strings. If the strong PRG G is computable in P, then the question of deciding whether a given string z is generated by G or not is in NP. A non-deterministic Turing machine can guess the seed x and verify $G(x) \overset{?}{=} z$. Therefore, the existence of strong PRG implies that this NP problem cannot be solved by polynomial-size circuits, which implies $\mathsf{NP} \not\subseteq \mathsf{P/poly}$.

Therefore, natural proofs capture a self-defeating nature of circuit lower bounds [PS21b]: a circuit lower bound assumption implies a barrier to proving circuit lower bounds. Analogously, the result in [PS19] can also be viewed as a self-defeating nature of proof complexity lower bounds. By Theorem 5.1 and Proposition 5.2, Rudich's Conjecture itself implies a proof complexity lower bound: "For any propositional proof system P, for almost all Boolean functions f_n, $P \nvdash_E \mathsf{clb}(f_n)$." The statement that Rudich's Conjecture does not admit feasible propositional proofs, *i.e.* "There exists a propositional proof system R such that for any propositional proof system S, for almost all Boolean functions f_n, $S \nvdash_E \mathsf{plb}_R(\mathsf{clb}(f_n))$," can be interpreted as the hardness of proving proof complexity lower bounds. Then, Lemma 7.3 shows that a proof complexity lower bound assumption implies a barrier for proving proof complexity lower bounds. These similar self-defeating aspects of circuit lower bounds and proof complexity lower bounds are worth pursuing in the future.

Standard assumption for Rudich's Conjecture. Let's return to the assumption of Rudich's Conjecture. Although the Super-bit Conjecture and Demi-bit Conjecture imply Rudich's Conjecture, questions still remain. First, as stated in [Rud97], demi-bits are considered more natural than super-bits. However, currently, we still do not know how to stretch demi-bits to polynomial length, or even to linear length. A simpler question is this: assuming that we can stretch the demi-bits to linear length, how can we further stretch them to polynomial length? Additionally, as proven in Proposition 2.1, the Super-bit Conjecture implies the Demi-bit Conjecture.

The question of whether the Demi-bit Conjecture implies the Super-bit Conjecture remains unanswered. Also, both the Super-bit Conjecture and Demi-bit Conjecture are not standard assumptions in cryptography. Is it possible to establish a standard assumption for Rudich's Conjecture?

Possible connections between Razborov's Conjecture and other standard conjectures in complexity theory. Regarding Razborov's Conjecture, connections between Razborov's Conjecture and standard conjectures in complexity theory have been found in [PS19, PS21a]. However, more connections are needed since discovering these connections is crucial in understanding the position and strength of Razborov's Conjecture.

[AB09] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach.* Cambridge University Press, 2009.

[ABFR91] James Aspnes, Richard Beigel, Merrick Furst, and Steven Rudich. The expressive power of voting polynomials. In *Proceedings of the twenty-third annual ACM symposium on Theory of Computing*, pages 402–409, 1991.

[ABSRW04] Michael Alekhnovich, Eli Ben-Sasson, Alexander A Razborov, and Avi Wigderson. Pseudorandom generators in propositional proof complexity. *SIAM Journal on Computing*, 34(1):67–88, 2004.

[And87] Alexander E Andreev. On a method for obtaining more than quadratic effective lower bounds for the complexity of π-schemes. *Moscow Univ. Math. Bull.*, 42(1):63–66, 1987.

[BGS75] Theodore Baker, John Gill, and Robert Solovay. Relativizations of the P$\overset{?}{=}$NP question. *SIAM Journal on computing*, 4(4):431–442, 1975.

[BM84] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudorandom bits. *SIAM Journal on Computing*, 13(4):850, 1984.

[BP01] Paul Beame and Toniann Pitassi. Propositional proof complexity: Past, present. *Future*, pages 42–70, 2001.

[BT88] Samuel R Buss and György Turán. Resolution proofs of generalized pigeonhole principles. *Theoretical Computer Science*, 62(3):311–317, 1988.

[BW10] Edward A Bender and S Gill Williamson. *Lists, decisions and graphs.* S. Gill Williamson, 2010.

[CK07] Stephen Cook and Jan Krajíček. Consequences of the provability of NP subset P/poly. *The Journal of Symbolic Logic*, 72(4):1353–1371, 2007.

[CN10] Stephen Cook and Phuong Nguyen. *Logical foundations of proof complexity*, volume 11. Cambridge University Press Cambridge, 2010.

[Coo71] Stephen A Cook. The complexity of theorem proving procedure. In *Proc. 3rd Symp. on Theory of Computing*, pages 151–158, 1971.

[CR79] Stephen A Cook and Robert A Reckhow. The relative efficiency of propositional proof systems. *The Journal of Symbolic Logic*, 44(1):36–50, 1979.

[DRMN$^+$20] Susanna De Rezende, Or Meir, Jakob Nordström, Toniann Pitassi, Robert Robere, and Marc Vinyals. Lifting with simple gadgets and applications to circuit and proof complexity. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 24–30. IEEE, 2020.

[DRNV16] Susanna F De Rezende, Jakob Nordström, and Marc Vinyals. How limited interaction hinders real communication (and what it means for proof and circuit complexity). In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 295–304. IEEE, 2016.

[FSS84] Merrick Furst, James B Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical systems theory*, 17(1):13–27, 1984.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM (JACM)*, 33(4):792–807, 1986.

[Gol08] Oded Goldreich. Computational complexity: a conceptual perspective. *ACM Sigact News*, 39(3):35–39, 2008.

[Gol10] Oded Goldreich. *A primer on pseudorandom generators*, volume 55. American Mathematical Soc., 2010.

[HMP$^+$93] András Hajnal, Wolfgang Maass, Pavel Pudlák, Mario Szegedy, and György Turán. Threshold circuits of bounded depth. *Journal of Computer and System Sciences*, 46(2):129–154, 1993.

[KC00] Valentine Kabanets and Jin-Yi Cai. Circuit minimization problem. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 73–79, 2000.

[KL07] Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography: principles and protocols.* Chapman and hall/CRC, 2007.

[Koz78] Dexter Kozen. Indexing of subrecursive classes. In *Proceedings of the tenth annual ACM symposium on Theory of computing*, pages 287–295, 1978.

[Kra01] Jan Krajíček. On the weak pigeonhole principle. *Fundamenta Mathematicae*, 1(170):123–140, 2001.

[Kra19] Jan Krajíček. *Proof complexity*, volume 170. Cambridge University Press, 2019.

[Lub96] Michael Luby. *Pseudorandomness and cryptographic applications*, volume 1. Princeton University Press, 1996.

[NW94] Noam Nisan and Avi Wigderson. Hardness vs randomness. *Journal of computer and System Sciences*, 49(2):149–167, 1994.

[OS18] Igor Carboni Oliveira and Rahul Santhanam. Hardness magnification for natural problems. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 65–76. IEEE, 2018.

[PS19] Jan Pich and Rahul Santhanam. Why are proof complexity lower bounds hard? In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1305–1324. IEEE, 2019.

[PS21a] Ján Pich and Rahul Santhanam. Learning algorithms versus automatability of frege systems. *arXiv preprint arXiv:2111.10626*, 2021.

[PS21b] Ján Pich and Rahul Santhanam. Strong co-nondeterministic lower bounds for NP cannot be proved feasibly. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 223–233, 2021.

[Raz15] Alexander A Razborov. Pseudorandom generators hard for k-dnf resolution and polynomial calculus resolution. *Annals of Mathematics*, pages 415–472, 2015.

[Rec75] Robert A Reckhow. *On the lengths of proofs in the propositional calculus.* PhD thesis, University of Toronto, 1975.

[RR94] Alexander A Razborov and Steven Rudich. Natural proofs. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 204–213, 1994.

[Rud97] Steven Rudich. Super-bits, demi-bits, and NP/qpoly-natural proofs. In *International Workshop on Randomization and Approximation Techniques in Computer Science*, pages 85–93. Springer, 1997.

[Smo87] Roman Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 77–82, 1987.

[TZ23] Iddo Tzameret and Lu-Ming Zhang. Stretching demi-bits and nondeterministic-secure pseudorandomness. *arXiv preprint arXiv:2304.14700*, 2023.

[Yao82] Andrew C Yao. Theory and application of trapdoor functions. In *23rd Annual Symposium on Foundations of Computer Science (SFCS 1982)*, pages 80–91. IEEE, 1982.