Imperial College
London

Nondeterministic Variants of Pseudorandomness, and Learning Theory

Lu-Ming Zhang
Imperial College London

A thesis submitted for the degree of
MSc in Artificial Intelligence
September 2022

ABSTRACT

Super-bits and demi-bits introduced by Rudich [33] are nondeterministic variants of strong pseudorandom generators. Demi-bits require their distinguishers to break them in a stronger sense than super-bits. Whether a demi-bit is stretchable at all had been an open problem for about 25 years. We answer this question affirmatively by providing an algorithm that achieves a sublinear-stretch for any given demi-bit.

Furthermore, we demonstrate the plausibility of the existence of demi-bits, by giving a rigorous proof, based on a sketch by Pich [28], that demi-bits exist assuming the class of polynomial-size boolean circuits is not PAC-learnable by sub-exponential circuits. If we further assume the existence of demi-bits implies the existence of super-bits, we are able to rule out the existence of $N\tilde{P}/qpoly$-natural properties useful against $P/poly$ based on the assumption that the class of polynomial-size boolean circuits is not PAC-learnable by sub-exponential circuits.

In the deterministic setting, Yao [39] proved that the super-polynomial hardness of pseudorandom generators is equivalent to unpredictability. Unpredictability loosely means, given any strict prefix of a random string, it is infeasible to predict the next bit. We initiate the study of unpredictability beyond the deterministic setting, and characterise the nondeterministic hardness of pseudorandom generators from a predictability perspective. Specifically, we propose four stronger notions of unpredictability: $NP/poly$-unpredictability, $coNP/poly$-unpredictability, $\cap$-unpredictability, $\cup$-unpredictability, and show that super-polynomial nondeterministic hardness is stronger than $\cap$-unpredictability but weaker than $\cup$-unpredictability.

Moreover, we propose a nondeterministic variant of hard-core predicates, called super-core predicates. We establish that the existence of a super-bit is equivalent to the existence of a super-core of some non-shrinking function, which serves as a counterpart of the known equivalence between the existence of a strong pseudorandom generator and the existence of a hard-core of some one-way function [17, 20]. We also prove that a certain class of functions, which may have hard-cores, can not possess any super-core.

ACKNOWLEDGEMENTS

I shall give deep thanks to my thesis supervisor Prof Iddo Tzameret for his meticulous and remarkable supervision and guidance, and deep thanks to my friend Yufeng Li for his timely assistance with many LaTeX related issues. I shall also give thanks to Dr Iain Phillips for providing helpful midway feedback and to Prof Rahul Santhanam and Dr Ján Pich for discussion in the mid-course. Without their help, the completion of this project would have been impossible.

Contents

1 INTRODUCTION

In this chapter, we introduce the relevant background and summarise our contributions in the first chapter. We defer formal definitions of concepts to the chapters that follow. Specifically, we review basic concepts appearing across different chapters in Chapter 2, and recall or introduce concepts belonging to a single chapter in the corresponding chapter. We assume readers have basic knowledge of complexity. Please see Section 1.3 for notation declarations.

1.1 Background and motivation

Pseudorandomness and **pseudorandom generators** (PRGs) are essential concepts in fields such as cryptography, artificial intelligence, complexity and algorithms, and beyond. Widespread application of PRGs is revealed in derandomization, secure encryption, zero-knowledge proofs, cryptographic protocols, etc [14]. PRGs efficiently expands short random seeds into longer strings which are computationally indistinguishable from truly random strings. We call such generated strings pseudorandom. Pioneering work on computational indistinguishability were developed in [18, 34] in the context of encryption. Yao [39] firstly defined PRGs as producing sequences that are computationally indistinguishable from uniform sequences and proved this definition of indistinguishability is equivalent to deterministic unpredictability, which was used in an earlier definition of PRGs suggested by Blum and Micali [9]. Loosely speaking, unpredictability means, given any strict prefix of a random string, it is infeasible to predict the next bit.

The stretching length of a PRG $g : \{0,1\}^n \longrightarrow \{0,1\}^{m(n)}$, defined as $m - n$, measures how much g can stretch a random seed x, and the hardness of g describes how secure g is. We say a PRG is C-hard if it can sufficiently fool all distinguishers in class C. Strong PRGs are ones that are P-hard in the uniform setting or $P/poly$-hard in the nonuniform setting. Unfortunately, any concrete example of strong PRGs is unknown because the existence of such a PRG implies $P \neq NP$ in the uniform setting and $P/poly \neq NP/poly$ in the nonuniform setting. Nevertheless, PRGs that can fool classes of weaker distinguishers were constructed. Nisan and Wigderson [26] constructed PRGs safe against constant-depth circuits based on a parity lower bound given in [19]. Babai, Nisan, and Szegedy [5] constructed logspace-hard PRGs by considering multiparty communication protocol lower bounds. Other kinds of special-purpose PRGs were also suggested. T-wise independent generators in [3, 11] fool any distinguishers that only observe t locations in the output sequence. Small-bias generators in [4, 25] fool any linear test. Expander Random Walk Generator in [1] produces a sequence of stings that hit any dense subset of strings with probability close to the hitting probability of a random string.

One-way functions and **hard-core predicates** are concepts intimately related to PRGs and particularly to constructing candidates of strong PRGs. The notion of one-way functions was introduced by Diffie and Hellman [12], and the notion of hard-core predicates originates from the work of Blum and Micali [9]. Loosely speaking, a one-way function (family) f is a one that is easy to compute but hard to invert on average (with the probability taken over the range of f). More precisely, "easy to compute" means f is in P or $P/poly$, and "hard to invert" means any efficient algorithm can only invert a negligible portion of $y = f(x)$ when x is unseen. By "efficient", in the uniform setting, we mean an algorithm in bounded-error probabilistic polynomial time (BPP), and in the nonuniform setting, an algorithm in $P/poly$, and by "invert", we mean finding an x' for a given y in $range(f)$ such that $f(x') = y$. A negligible portion for us means a portion that is less than $1/p(n)$ for any polynomial p and all large n's. We say $b : \{0,1\}^n \longrightarrow \{0,1\}$ in P or $P/poly$ is a hard-core of a function f if it is infeasible (a negligible portion more than $1/2$) to efficiently predict $b(x)$ given $f(x)$.

The existence of hard-core predicates is known (e.g., $b(x) = x[-1]$, the last bit of a string x, is a hard-core of the function $f(x) = x[1]$, the first bit of x), but the existence of a hard-core for a one-way function and the existence of any one-way function at all are unknown. Goldreich and Levin [17] proved the fact that the inner product *mod* 2 is a hard-core for any one-way function of the form $g(x, r) = (f(x), r)$ (where f is any function and $|x| = |r|$) supposing some one-way function exists at all. Sequentially, Håstad, Impagliazzo, Levin, and Luby [20] showed that strong PRGs exist if and only if one-way functions exist. This theorem can also be stated equivalently as: a strong PRG exists if and only if a hard-core of some one-way function exists.

Many candidates of strong PRGs were constructed by exploiting functions conjectured to be one-way or/and their hard-cores (e.g, cf. [2, 21, 22]). [2] presented PRGs based on the intractability assumption of factoring. [22] presented PRGs based on the intractability assumption of discrete-logarithm problem. [21] presented PRGs based on the subset sum problem.

The existence of strong PRGs also imposes limitations on certain proof techniques, called natural proofs, used in establishing circuit lower bounds.

The notion of **natural proofs** was introduced by Razborov and Rudich in [31]. Natural proofs for proving circuit lower bounds are proofs that somehow use a natural combinatorial property of boolean functions. A combinatorial property (or a property for short) C of boolean functions is a set of boolean functions. We say a function f has the property C if f is in C. Let Γ and Λ be complexity classes, and F_n be the set of all $f : \{0, 1\}^n \to \{0, 1\}$. We say C is Γ-natural it itself or at least some subset C' of it satisfies constructivity (whether an $f \in C'$ is Γ-computable) and largeness (C' constitutes a nonnegligible portion of F_n). We say C is useful against Λ if every function family f which has property C infinitely often is not computable in Λ. The idea of natural proofs is that, if we want to prove some function family f (e.g., the boolean satisfiability problem SAT, or some other problem) in not in $P/poly$ (or some complexity class Λ in general), we identify some natural combinatorial property C of f and show all function families which have property C is not in $P/poly$ (i.e., useful against $P/poly$). If f is NP-complete (e.g., SAT), then such a proof concludes $P \neq NP$.

Unfortunately, similarly to the result of Baker, Gill, and Solovay [6] who showed that relativizing proof techniques could not solve problems such as $P \neq NP$, Razborov and Rudich argued that, based on the existence assumption of strong PRGs, no $P/poly$-natural proofs can be useful against $P/poly$. They showed that all the known proofs of lower bounds against non-monotone boolean circuits are natural or can be presented as natural in some way. Examples include: [13, 19, 40] established lower bounds for the parity function by AC^0-natural proofs, and [7, 30, 35] established $AC^0[q]$ lower bounds with q being a prime power by NC^2-natural proofs ($AC^0[m]$ is the class of poly-size constant-size circuits allowing *mod* m gates).

In a follow-up paper of [31], Rudich [33] introduced, in the nonuniform setting, **super-bits** and **demi-bits** as nondeterministic variants of strong PRGs, and showed that the existence of super-bits rules out the existence of a larger class of natural proofs (than $P/poly$-natural proofs) useful against $P/poly$. Super-bits and demi-bits are PRGs secure against nondeterministic adversaries (i.e., adversaries in $NP/poly$). More specifically, super-bits and demi-bits both require a nondeterministic adversary to meaningfully distinguish truly random strings from pseudorandom ones by certifying truly random ones (i.e., an adversary outputs 1 if it thinks a given string is random and 0 otherwise). Thus, a nondeterministic distinguisher cannot break super-bits nor demi-bits by simply guessing a seed. The difference between super-bits and demi-bits is that demi-bits require their distinguishers to break them in a stronger sense: a distinguisher must always be correct on the pseudorandom strings (i.e., always output 0). Thus, if a PRG g is super-bit(s) (the plural here denotes g may have stretching length more than 1; if the stretching length is exactly 1, we say g is a super-bit), no algorithms in $NP/poly$ can break g in the weaker sense, and hence no algorithms in $NP/poly$ can break g in the stronger sense, which means g is also demi-bit(s). In other words, the existence of super-bits implies the existence of demi-bits, although both are unknown (obviously, the existence of super-bits also implies the existence of strong PRGs). In this paper, Rudich conjectured a concrete super-bit generator based on the hardness of the subset sum problem studied in [21].

In the standard theory of pseudorandomness, a hard-bit (i.e., a strong PRG with stretching length 1) is shown to be stretchable to polynomially many hard-bits (i.e. a polynomial stretching-length) [9, 39] and can even be exploited to construct hard-to-break pseudorandom function generators (loosely speaking, generators that generate pseudorandom functions indistinguishable from truly random ones) [16]. As a hard-bit, a super-bit can also be stretched, using similar stretching algorithms, to polynomially many super-bits and to pseudorandom function generators safe against nondeterministic adversaries [33]. The proofs of the correctness of such stretching algorithms are based on a technique called the hybrid argument [18] reviewed below. In contrast, whether a demi-bit can be stretched to two demi-bits was even unknown before our work as a standard hybrid argument does not apply. However, demi-bits stretching may have intriguing applications to proof complexity generators and in average-case complexity.

We mentioned that a strong PRG exists if and only if a hard-core of some one-way function exists. Thus, just as super-bits is the nondeterministic correspondence of strong PRGs, a meaningful question to ask is:

What are the nondeterministic correspondences of one-way functions and hard-cores?

We shall call these nondeterministic variants of one-way functions and hard-cores “super” one-way functions and super-cores, respectively. To come up with a reasonable definition of super one-way functions is not an easy task because, for any function f, a nondeterministic algorithm can always invert a range-element y by guessing some x and checking if $f(x) = y$. Similarly, a reasonable definition of super-core predicates is non-trivial as well: for any function f and predicate b, a nondeterministic algorithm can predict $b(x)$ when given $f(x)$ as input by guessing x and then applying b.

We also mentioned that, in the deterministic setting, the hardness of PRGs is equivalent to unpredictability, but unpredictability beyond the deterministic setting had not attracted proper attention before our work. A better understanding of unpredictability beyond the deterministic setting would help to better understand the nondeterminitic hardness of PRGs.

On the other hand, **PAC-learning** algorithms can be developed from breaking PRGs (e.g., cf. [8, 27]). The model of PAC-learning (an abbreviation of probably approximately correct learning) was introduced and elaborated by Valiant in [36–38]. In the PAC-learning model, the goal of a leaner is to learn an arbitrary target function f drawn from a target set (e.g., the class of decision trees, the class of boolean conjunctions, or even $P/poly$). The target function is invisible to the learner. The learner receives samples (randomly or by querying) from an f-oracle and selects a generalization function f', called the hypothesis, from some hypothesis class. The selected function must have low generalization error (the “approximately correct” in “PAC”) with high probability (the “probably” in “PAC”). Furthermore, the learner is expected to be efficient and to output hypotheses that are as well efficiently evaluable on any given input.

The **hybrid argument** (a.k.a. the hybrid method, the hybrid technique, etc.) is a common proof technique in the study of complexity and is also used repeatedly in this thesis. The hybrid argument originated from the work of Goldwasser and Micali [18] and was named by Leonid Levin. (Please refer to Section 1.3 for notations used below.) When we have a PRG $g : \{0,1\}^n \to \{0,1\}^{m(n)}$, a distinguisher D, and a function p (usually a polynomial) such that

$$\mathbb{P}[D(U_m) = 1] - \mathbb{P}[D(g(U_n)) = 1] \geq 1/p(n), \tag{$*$}$$

where U_m stands for the truly random strings and $g(U_n)$ stands for the pseudorandom ones, the standard hybrid argument defines a spectrum (i.e. an ordered set) of random variables H_i's, call hybrids, traversing from one extreme, U_m, to another, $g(U_n)$. A concrete example is $H_i := g(U_n)[1...i] \cdot U_{m-i}, 0 \leq i \leq m$. In this example, indeed $H_0 = U_m$ and $H_m = g(U_n)$. Then the inequality $(*)$ can be written as:

$$\begin{aligned} 1/p(n) &\leq \mathbb{P}[D(U_m) = 1] - \mathbb{P}[D(g(U_n)) = 1] \\ &= \sum_i (\mathbb{P}[D(H_i) = 1] - \mathbb{P}[D(H_{i+1}) = 1]). \end{aligned}$$

Thus, a usual next step is to claim there exists some i such that

$$\mathbb{P}[D(H_i) = 1] - \mathbb{P}[D(H_{i+1}) = 1] \geq \frac{1}{k \cdot p(n)},$$

where k is the total number of hybrids (in the above example, $k = m$). In a nutshell, we show that if we can distinguish U_m from $g(U_n)$ by a $1/p(n)$ portion, then we can distinguish some neighbouring pair of hybrids H_i from H_{i+1} by a $1/(k \cdot p(n))$ portion. Please see Lemma 4.2.1 for a simple demonstration of the hybrid argument.

As we have mentioned, a standard hybrid argument cannot be applied to prove a stretched PRG g by some stretching algorithm, from a single demi-bit b, is still demi-bits. We now intuitively explain the reason of this. A usual proof goes like this: we assume, for a contradiction, g is not demi-bits. Then there are some distinguisher D of g and function p such that $\mathbb{P}[D(U_m) = 1] - \mathbb{P}[D(g(U_n)) = 1] \geq 1/p(n)$, and in particular $\mathbb{P}[D(g(U_n)) = 1] = 0$ as D is a breaker of demi-bits, and we hope we can construct some qualified distinguisher D' of b based D. However, as we saw above, a standard hybrid argument only yields that $\mathbb{P}[D(H_i) = 1] - \mathbb{P}[D(H_{i+1}) = 1] \geq 1/p'(n)$ for some function p' and cannot deduce that $\mathbb{P}[D(H_{i+1}) = 1] = 0$. Then we are usually at a wit's end of constructing any D' such that $\mathbb{P}[D'(b(U_n)) = 1] = 0$.

1.2 Our contributions and organisation

This thesis defines and studies nondeterministic variants and counterparts of established concepts and results, related to pseudorandom, in the deterministic setting. As Rudich did in [33], our study is also in the nonuniform setting. We summarise our main contributions in this section.

In **Chapter 2**, we formulate the concepts of infinitely often (i.o.) super-bits and demi-bits, which are only secure for infinitely many n's (where n denotes the length of input seeds), and show that we can construct a (full) super-bit/demi-bit defined as usual from an i.o. one which is "decently" often.

In **Chapter 3**, we provide an algorithm that achieves a sublinear-stretch for any given demi-bit and thus solve the open problem whether a demi-bit is stretchable at all. We overcome the barrier for proving the stretchability of demi-bits by a clever and more flexible use of the hybrid technique combined with some other tricks. We encourage the reader to visit the exact proof of Theorem 3.1.2 for the details.

In **Chapter 4**, we formulate rigorously a nonuniform version of PAC-learning. Furthermore, based on a sketch by Pich [28], we give a rigorous proof that:

THEOREM 1.2.1 (INFORMAL). *Demi-bits exist assuming the class of polynomial-size boolean circuits is not PAC-learnable by sub-exponential circuits.*

The informal theorem stated here is, in fact, a contrapositive to theorems appearing in Chapter 4. This result demonstrates the plausibility of the existence of demi-bits because it is widely believed (e.g., cf. [29]) that learning $P/poly$ is hard.

In **Chapter 5**, We initiate the study of unpredictability beyond the deterministic setting. We propose four stronger notions of unpredictability for probability ensembles:

(1) $NP/poly$-unpredictability: the capacity of being unpredictable by $NP/poly$ predictors.
(2) $coNP/poly$-unpredictability: the capacity of being unpredictable by $coNP/poly$ predictors.
(3) $\cup$-unpredictability: the capacity of being unpredictable by predictors in $NP/poly$ and predictors in $coNP/poly$.
(4) $\cup$-unpredictability: the capacity of being unpredictable by nondeterministic function-computing predictors.

We then characterise the nondeterministic hardness of pseudorandom generators from a predictability perspective (here, $A \leq B$ denotes B implies A):

THEOREM 1.2.2 (INFORMAL).

$$\cap\text{-}\textit{unpredictable} \leq \textit{super-polynomially nondeterministic-hard} \leq \cup\text{-}\textit{unpredictable}$$

and

$$\cap\text{-}\textit{unpredictable} \leq NP/poly\text{-}\textit{unpredictable} \vee coNP/poly\text{-}\textit{unpredictable} \leq \cup\text{-}\textit{unpredictable}.$$

Moreover, we pin down a sensible definition of super-core predicates served as a nondeterministic correspondence of hard-core predicates. We establish:

THEOREM 1.2.3 (INFORMAL). *There is a super-core b of some non-shrinking function if and only if there is a super-bit.*

as a counterpart of the known equivalence between the existence of a hard-core of some one-way function and the existence of a strong PRG. We also show that a certain class of functions, which may have hard-cores, can not possess any super-core:

THEOREM 1.2.4 (INFORMAL). *If a length-preserving function f is "predominantly" one-to-one, then f does not have a super-core.*

We say a function f is "predominantly" one-to-one if $|f^{-1}(f(x))| = 1$ for at least a 2/3-portion of x's in $domain(f)$. What we achieve in chapter 5 provides a big step forward to better understand nondeterministic hardness of PRGs and to suggest a sensible definition of one-way functions in the nondeterministic setting.

1.3 Notations and conventions

The thesis follows the following conventions:

- $\mathbb{N}$ denotes the set of positive integers (excluding 0). For $n \in \mathbb{N}$, $[n]$ denotes $\{1, ..., n\}$. $[0]$ is the empty set $\emptyset$.
- The size of a Boolean circuit C, denoted as $size(C)$ or $|C|$, is the total number of gates (including the input gates). $Circuit[s]$ denotes the Boolean circuits of size at most s. If $s : \mathbb{N} \longrightarrow \mathbb{N}$ is a function, $Circuit[s]$ contains all the Boolean circuit families C_n such that $|C_n| \leq s(n)$ for all large n's.
- All the distributions we consider in this dissertation are, by default, uniform. U_n denotes the uniform distribution over $\{0, 1\}^n$ unless stated otherwise.
- For functions $f, g : \{0, 1\}^n \longrightarrow \{0, 1\}$, we say g γ-approximates f if $\mathbb{P}_x[f(x) = g(x)] \geq \gamma$.

- For a string x, where its bits are indexed from left to right by 1 to $|x|$, $x[i]$ denotes its i-th bit, and $x[i...j]$ denotes the sub-string indexed from i to j of x (if $i > j$, $x[i...j] = \varepsilon$, the empty string). $x[-i]$ denotes its i-th last bit.
- For strings x, y, we may use any of the following to denote the concatenation of x and y: xy, (x, y), $x \cdot y$.
- We may not verbally distinguish a function with its string representation (this can be a truth table, or a string encoding a circuit representation of this function, etc.) when there is no ambiguity.

We may also follow other common conventions used in the complexity community or literature.

2 PRELIMINARIES AND BASIC CONCEPTS

This chapter reviews and introduces preliminaries that appear across the subsequent chapters.

2.1 Computation models

Definition 2.1.1 (randomized circuits). *A circuit C is a* ***randomized circuit*** *if, in addition to the standard input bits, it contains zero or more random input bits (i.e., bits taken from a random distribution). We call $\{C_n\}$ a* ***randomized circuit family*** *if for every n, C_n is a randomized circuit with n "user"-input bits (i.e., standard input bits as a non-randomized circuit has).*

If a randomized circuit family $\{C_n\}$ is in $Circuit[s(n)]$, that means for every large n, the randomized circuit C_n has size at most $s(n)$, which automatically constrains the number of the random input bits that C_n is allowed to have.

Definition 2.1.2 (nondeterministic/co-nondeterministic circuits). *A circuit C is a* ***(co-)nondeterministic circuit*** *if, in addition to the standard input bits, it contains zero or more nondeterministic input bits. We call $\{C_n\}$ a* ***(co-)nondeterministic circuit family*** *if for every n, C_n is a randomized circuit with n "user"-input bits. For a nondeterministic/co-nondeterministic circuit with one-bit output, it is said to accept/reject if and only if there is a choice of nondeterministic bits that causes the circuit to accept/reject its input (composed of the deterministic part and the nondeterministic choice).*

Definition 2.1.3 (oracle circuits). *C is an* ***oracle circuit*** *if it is allowed to equip with oracle gates. We write C as $C^{f_1,\dots,f_k}$ if C has oracle gates computing Boolean functions $f_1, \dots, f_k$.*

We note that an oracle gate computing a Boolean function $f : \{0,1\}^n \to \{0,1\}$ has fan-in n, and in our model, an oracle gate is allowed to appear in any place of the circuit.

2.2 Natural proofs

Let F_n be the set of all functions $f : \{0,1\}^n \to \{0,1\}$ and Γ and Λ be complexity classes. We call $C = (C_n)_{n\in\mathbb{N}}$ a **combinatorial property** of boolean functions if each $C_n \subseteq F_n$.

Definition 2.2.1 (natural properties [31]). *We say a combinatorial property $C = (C_n)_{n\in\mathbb{N}}$ is* ***Γ-natural*** *if some $C' = (C'_n)_{n\in\mathbb{N}}$ with $C'_n \subseteq C_n$ for each n satisfies:*

- ***Constructivity.*** *Whether $f \in C'_n$ is computable in Γ when f is encoded by its truth table as input.*
- ***Largeness.*** *$|C'_n| \geq 2^{-O(n)} \cdot |F_n|$ for all large n's.*

We say C is ***useful*** *against Λ if it satisfies:*

- ***Usefulness.*** *For any function family $f = (f_n)_{n\in\mathbb{N}}$, if $f_n \in C_n$ infinitely often, then $f \notin \Lambda$.*

A circuit lower bound proof that some explicit function (family) is not in Γ is call a Γ-**natural proof** against Λ if it uses, more or less explicitly, some Γ-natural combinatorial property useful against Λ. Especially, a $P/poly$-natural proof against $P/poly$ is a proof that uses a $P/poly$-natural combinatorial properties useful against $P/poly$.

We note that the notion of natural proofs, unlike natural combinatorial property, is not defined in a mathematically rigorous sense. Nevertheless, the use of the terminology "natural proof" in a statement more intuitively embodies our intention and also does not affect the rigorousness of the statement. Whenever we say Γ-natural proofs do or do not exist, what we mean, in a mathematically rigorous sense, is Γ-natural combinatorial properties do or do not exist.

2.3 Pseudorandom generators

Definition 2.3.1 (pseudorandom generators). *A function family $g_n : \{0,1\}^n \longrightarrow \{0,1\}^{l(n)}$ is a* ***pseudorandom generator (PRG)*** *if $g_n \in P/poly$ and $l(n) > n$ for every n. We call such an l a stretching function and call $l(n) - n$ the stretching length of g_n (sometimes, we alternatively call $l(n)$ the stretching length).*

In fact, in more general settings, the requirement of being $P/poly$-computable for PRGs is unnecessary when, for example, in the context of derandomization (cf. [26]). Nevertheless, all the PRGs concerned in following thesis will be in $P/poly$.

We review the definition of the standard hardness for PRGs:

Definition 2.3.2 (standard hardness). *Let $g_n : \{0,1\}^n \longrightarrow \{0,1\}^{l(n)}$ be a PRG. Then the* ***hardness*** *$H(g_n)$ of g_n is the minimal s for which there exists a circuit D of size at most s such that*

$$\left| \underset{y\in\{0,1\}^{l(n)}}{\mathbb{P}}[D(y)=1] - \underset{x\in\{0,1\}^n}{\mathbb{P}}[D(g_n(x))=1] \right| \geq 1/s(n).$$

The order of the two terms in the absolute value and the absolute value itself are immaterial since in the deterministic setting, we can always flip the output bit of a distinguisher D.

Definition 2.3.3 (strong pseudorandom generators). *A PRG $g : \{0,1\}^n \longrightarrow \{0,1\}^{l(n)}$ is called a* ***strong PRG*** *if for every D in $P/poly$, every polynomial p, and all sufficiently large n's,*

$$\left| \underset{y\in\{0,1\}^{l(n)}}{\mathbb{P}}[D(y)=1] - \underset{x\in\{0,1\}^n}{\mathbb{P}}[D(g(x))=1] \right| < 1/p(n).$$

A strong PRG is defined to be a PRG safe against all polynomial-size distinguishers. An alternative definition used in some resources is: a PRG with hardness at least 2^{n^ε} for some $\varepsilon > 0$ and all large n's, which defines a "stronger" PRG.

We comment that a function $f(n) : \mathbb{N} \longrightarrow \mathbb{R}^+$ is (at least) exponential if there is some $\varepsilon > 0$ such that $f(n) \geq 2^{n^\varepsilon}$ for all large n's. A function is not (at least) exponential if for every $\varepsilon > 0$, there exist infinitely many n's such that $f(n) < 2^{n^\varepsilon}$, which is equivalent to: there is an infinite monotone sequence $(n_i) \subseteq \mathbb{N}$ such that $f(n_i)$ is sub-exponential in n_i (i.e., $f(n_i) = 2^{n_i^{o(1)}}$).

The existence of a strong PRG is quite plausible because many intractable problems (e.g., factoring) seem to provide a basis for constructing a such generator [31]. PRGs used in practice (built from block ciphers) have also been subjected to significant public scrutiny and resisted all attempts at attacks so far [32].

Conjecture 2.3.4. *There is a strong PRG.*

The existence of a strong PRG rules out the existence of $P/poly$-natural proofs useful against $P/poly$ [31].

2.4 Super-bits and demi-bits

Rudich [33] proposed nondeterministic variants of the concept of standard hardness. In Section 2.4, we conduct a brief literature review of the main results and open problems in [33] that are relevant to our thesis.

Definition 2.4.1 (Nondeterministic Hardness). *Let $g_n : \{0,1\}^n \longrightarrow \{0,1\}^{l(n)}$ be a PRG. Then the* ***nondeterministic hardness*** *$\mathrm{H_{nh}}(g_n)$ (also called* ***super-hardness****) of the PRG g_n is the minimal s for* *which there exists a nondeterministic circuit D of size at most s such that*

$$\underset{y\in\{0,1\}^{l(n)}}{\mathbb{P}}[D(y)=1]-\underset{x\in\{0,1\}^{n}}{\mathbb{P}}[D(g_n(x))=1]\geq\frac{1}{s}. \tag{1}$$

In contrast to the definition of deterministic hardness, the order of the two possibilities on the LHS is crucial. This order forces a nondeterministic distinguisher to certify the randomness of a given input. Reversing the order or keeping the absolute value trivialize the task of breaking g: a distinguisher D can simply guess a seed x and check if $g(x)$ equals the given input. For such a D, we have $\mathbb{P}[D(g(x))=1]=1$ and $\mathbb{P}[D(y)=1]\leq 1/2$.

We call exponentially super-hard PRGs super-bits:

Definition 2.4.2 (Super-bits). *$g:\{0,1\}^n\to\{0,1\}^{n+c}$ for some $c:\mathbb{N}\to\mathbb{N}$ is called c **super-bit(s)** (or a c-super-bit(s)) if $H_{\mathrm{nh}}(g)\geq 2^{n^{\varepsilon}}$ for some $\varepsilon>0$ and all sufficiently large n's. In particular, if $c=1$, we call g a super-bit.*

Super-bits semantically means pseudorandom bits that can fool "super" powerful adversaries. Rudich constructed a candidate super-bit based on subset sum and conjectured that:

Conjecture 2.4.3 (Super-bit conjecture). *There exists a super-bit.*

The core theorem in this paper of him is as follows, which is proved based on the stretchability of super-bits as we have discussed in Chapter 1.

theorem 2.4.4. *If super-bits exist, then there are no $N\tilde{P}/qpoly$-natural properties useful against $P/poly$, where $N\tilde{P}/qpoly$ is the class of languages recognised by non-uniform, quasi-polynomial-size circuit families.*

We remark that, in this theorem, the "largeness" requirement of $N\tilde{P}/qpoly$-natural properties can in fact be relaxed to $|C_n'|\geq 2^{-n^{O(1)}}\cdot|F_n|$ (cf. Definition 2.2.1).

Rudich also proposed another more intuitive notion (in his opinion), call demi-hardness, than super-hardness:

Definition 2.4.5 (Demi-Hardness). *Let $g_n:\{0,1\}^n\to\{0,1\}^{l(n)}$ be a PRG in $P/poly$. Then the **demi-hardness** $H_{\mathrm{dh}}(g_n)$ of the PRG g_n is the minimal s for which there exists a nondeterministic circuit D of size at most s such that*

$$\underset{y\in\{0,1\}^{l(n)}}{\mathbb{P}}[D(y)=1]\geq\frac{1}{s}\quad\text{and}\quad\underset{x\in\{0,1\}^{n}}{\mathbb{P}}[D(g_n(x))=1]=0. \tag{2}$$

We note (2), which requires a distinguisher to make no mistake on generated strings, is a stronger requirement than (1). Thus, $H_{\mathrm{nh}}(g)\leq H_{\mathrm{dh}}(g)$ for every generator g.

We call exponentially demi-hard PRGs demi-bits, where "demi" semantically means (in Rudich's opinion) "half".

Definition 2.4.6 (Demi-bits). *$g:\{0,1\}^n\to\{0,1\}^{n+c}$ for some $c:\mathbb{N}\to\mathbb{N}$ is called c **demi-bit(s)** (or a c-demi-bit(s)) if $H_{\mathrm{dh}}(g)\geq 2^{n^{\varepsilon}}$ for some $\varepsilon>0$ and all sufficiently large n's. In particular, if $c=1$, we call g a demi-bit.*

As $H_{\mathrm{nh}}(g)\leq H_{\mathrm{dh}}(g)$, it is reasonable to conjecture:

Conjecture 2.4.7 (Demi-bit conjecture). *There exists a demi-bit.*

and to ask:

Open Problem 2.4.8. *Does the existence of a demi-bit imply the existence of a super-bit?*

As we have discussed, A super-bit can be stretched to polynomially many super-bits and to pseudorandom function generators safe against nondeterministic adversaries. In contrast, whether a demi-bit can be stretched to two demi-bits was unknown before our work. We will still list this question here as it was proposed in Rudich's paper but resolve this open problem in our next chapter.

Open Problem 2.4.9. *If you have a demi-bit, can you stretch it to a 2-demi-bits?*

The question which is still open is:

Open Problem 2.4.10. *If you have a demi-bit, can you build a pseudorandom function generator with exponential demi-hardness?*

A positive answer to the last problem would answer:

Open Problem 2.4.11. *Does the existence of a demi-bit rules out the existence of $N\tilde{P}/qpoly$-natural properties against $P/poly$?*

2.5 Infinitely often super-bits and demi-bits

We formalize a weaker variant of super-bits/demi-bits, which only requires infinitely many n's to be "hard" (recall super-bits/demi-bits require hardness for all sufficiently large n's). This variant is occasionally concerned implicitly in the literature but may not have been formally defined.

Definition 2.5.1 (infinitely often super-bits/demi-bits). *$g : \{0,1\}^n \to \{0,1\}^{n+c}$ for some $c : \mathbb{N} \to \mathbb{N}$ is called c **infinitely often (i.o.) super-bit(s)/demi-bit(s)** if $H_{\mathrm{nh}}(g) \geq 2^{n^{\varepsilon}}/H_{\mathrm{dh}}(g) \geq 2^{n^{\varepsilon}}$ for some $\varepsilon > 0$ and infinitely many n's. In particular, if $c = 1$, we call g an i.o. super-bit/demi-bit.*

In fact, it is an easy observation that we can construct, from a reasonably often i.o. super-bit/demi-bit, a super-bit/demi-bit, by properly choosing a prefix of a given input n and applying the i.o. algorithm to the prefix. More details follow next. However, we (the author) are unaware if we can construct a super-bit/demi-bit from any i.o. super-bit/demi-bit.

Lemma 2.5.2. *Assume $g_n : \{0,1\}^n \to \{0,1\}^{n+c}$ for some $c \in \mathbb{N}$ are c i.o. super-bits/demi-bits. If there exist a polynomial p and an infinite monotone sequence $(n_i)_{i\in\mathbb{N}} \subseteq \mathbb{N}$ such that: (1) $n_{i+1} \leq p(n_i)$, and (2) for some $\varepsilon > 0$ and every $n \in (n_i)_{i\in\mathbb{N}}$, $H_{\mathrm{nh}}(g_n) \geq 2^{n^{\varepsilon}}/H_{\mathrm{dh}}(g_n) \geq 2^{n^{\varepsilon}}$, then there exists a c-super-bits/c-demi-bits constructed from g_n.*

Proof. We present the proof for constructing super-bits from i.o. super-bits, and the proof for constructing demi-bits from i.o. demi-bits is almost identical.

Assume g, (n_i), p are as given in the lemma statement. Denote $m = n+c$ and $m_i = n_i+c$. We construct a new generator G as follows:

> Given $x_n \in \{0,1\}^n$, there is an i such that $n_i \leq n < n_{i+1}$. Define $G(x_n) = g(a) \cdot b$, where $a = x_n[1...n_i], b = x_n[n_i+1...n]$. We note $|G(x_n)| = |g(a)| + |b| = n + c$.

We want to show that G is indeed super-bits. Suppose, for a contradiction, G is not. Then there exist an infinite monotone sequence $S \subseteq \mathbb{N}$, a sub-exponential function s, and a distinguisher D of size s such that for every $n \in S$,

$$1/s(n) \leq \mathbb{P}[D(U_m) = 1] - \mathbb{P}[D(G(U_n)) = 1] = \mathbb{P}[D(U_{m_i}U_{m-m_i}) = 1] - \mathbb{P}[D(g(U_{n_i})U_{m-m_i}) = 1]$$

Thus, for every $n \in S$, there exists a fixed string $w = w(n)$ such that

$$1/s(n) \leq \mathbb{P}[D(U_{m_i}w) = 1] - \mathbb{P}[D(g(U_{n_i})w) = 1].$$

We now construct a new distinguisher D' for g as follows:

Given $Y \in \{0,1\}^{m_i}$ as input, if there exists an $n \in S$ such that $n_i \leq n < n_{i+1}(\leq p(n_i))$, D' outputs $D(Yw(n))$, and otherwise D' always outputs 0 (which means D' fails to do anything for such an n_i).

Note that the (1) $n_{i+1} \leq p(n_i)$ assumption guarantees the efficiency of D'. As S is infinite and every $n \in S$ is between some n_i and n_{i+1}, there are infinitely many n_i's such that:

$$\mathbb{P}[D'(U_{m_i}) = 1] - \mathbb{P}[D'(g(U_{n_i})) = 1] = \mathbb{P}[D(U_{m_i}w) = 1] - \mathbb{P}[D(g(U_{n_i})w) = 1] \geq 1/s(n)$$

This shows, for infinitely many n_i's, $H_{\text{nh}}(g(n_i)) \leq 1/s'(n)$ for some sub-exponential s', which contradicts the assumption (2) in the lemma statement. ■

Remark. Lemma 2.5.2 is often used implicitly in the constructions of super-bits and demi-bits.

2.6 Complexity class separation

Although in this thesis, we will not consider class separation problems exclusively, it is worth mentioning that the existence assumptions of certain PRGs are stronger than certain complexity class separations.

If there exists a strong PRG g, then $P/poly \neq NP/poly$ because an $NP/poly$-distinguisher can easily break g by guessing a seed x and outputs 1 if and only if $g(x)$ equals the given input. Similarly, if there exists a demi-bit b, then $NP/poly \neq coNP/poly$ because an $coNP/poly$-distinguisher can easily break g by guessing a seed x and outputs 0 if and only if $g(x)$ equals the given input.

3 CONSEQUENCES OF THE PURPORTED EXISTENCE OF DEMI-BITS

In this chapter, we answer affirmatively whether a demi-bit is stretchable, which had been an open problem for about 25 years.

3.1 Stretching a demi-bit

We propose Algorithm 3.1.1, which stretches a single demi-bit to n^c demi-bits for any $c < 1$, and verify the correctness of this stretching algorithm (i.e., verify the exponential demi-hardness of the new elongated generator). Intuitively, Algorithm 3.1.1 partitions a given seed into disjoint pieces and apply the 1-demi-bit generator to each piece. The algorithm proposed here is very similar to other stretching or amplification algorithms seen in the literature (e.g., cf. [24, 39]). The real difficulty of demi-bit stretching comes from to prove the correctness of the attempted stretch (i.e., to proof the stretching algorithm applied preserve exponential demi-hardness).

Algorithm 3.1.1. *Suppose b is a demi-bit and $m = \lceil N^c \rceil$ for some $0 < c < 1$. We define a new PRG g as follows: given input x of length N, let $n = \lfloor \frac{N}{m} \rfloor$ and say $x = x_1 x_2 ... x_m r$, where each x_i has length n. $g(x) = b(x_1)...b(x_m)r$. We note that g has stretching length m.*

We make use of a modified hybrid argument combined with clever use of nondeterminism and nonuniformity to establish the following theorem. In the proof, we reserve the **bold face** for random variables.

Theorem 3.1.2. *g defined in Algorithm 3.1.1 has at least exponential demi-hardness.*

Proof. Given any $c < 1$, let $m = \lceil N^c \rceil$. Suppose, for a contradiction, g does not have exponential demi-hardness. That is, there is a sub-exponential size nondeterministic circuit D such that, for infinitely many N's (without loss of generality, we assume $m|N$ and let $n = N/m$), $\mathbb{P}[D(\mathbf{y_1}, \ldots, \mathbf{y_m}) = 1] \geq 1/|D|$ and $\mathbb{P}[D(b(\mathbf{x_1}), \ldots, b(\mathbf{x_m})) = 1] = 0$, where $\mathbf{x_1}, \ldots, \mathbf{x_m}$ are totally independent length-n random strings and $\mathbf{y_1}, \ldots, \mathbf{y_m}$ are totally independent length-$(n+1)$ random strings. Our aim is to efficiently break b in the desired sense.

We define a new nondeterministic circuit D' which takes a pair of parameters: the first is the same as D's, an $(N+m)$-bit string $y_1 ... y_m$, and the second is $i \in \{0, 1, \ldots, m\}$ (with i properly encoded):

> Given input $(y_1 ... y_m, i)$, D' guesses n-bit strings $x_1, \ldots, x_i$ and does whatever D does on $b(x_1)...b(x_i)y_{i+1}...y_m$.

We observe that $\mathbb{P}[D'(\mathbf{y_1}...\mathbf{y_m}, 0) = 1] = \mathbb{P}[D(\mathbf{y_1}...\mathbf{y_m}) = 1] \geq 1/|D|$ and $\mathbb{P}[D'(b(\mathbf{x_1})...b(\mathbf{x_m}), m) = 1] = \mathbb{P}[D(b(\mathbf{x_1})...b(\mathbf{x_m})) = 1] = 0$. D' is also of sub-exponential size.

A hybrid argument applied to terms $\mathbb{P}[D'(\mathbf{y_1}...\mathbf{y_m}, 0) = 1], \ldots, \mathbb{P}[D'(b(\mathbf{x_1})...b(\mathbf{x_i})\mathbf{y_{i+1}}...\mathbf{y_m}, i) = 1], \ldots,$ $\mathbb{P}[D'(b(\mathbf{x_1})...b(\mathbf{x_m}), m) = 1]$ gives that, for each of those N, there is an $i = i(N)$ such that

$$\mathbb{P}[D'(b(\mathbf{x_1})...b(\mathbf{x_{i-1}})\mathbf{y_i}...\mathbf{y_m}, i-1) = 1] \geq 1/f(N)$$

and

$$\mathbb{P}[D'(b(\mathbf{x_1})...b(\mathbf{x_i})\mathbf{y_{i+1}}...\mathbf{y_m}, i) = 1] \leq 1/s(N)$$

for some sub-exponential $f(N)$ and some at least exponential $s(N)$.

We let $S = \{0,1\}^{(n+1)\times(m-i)}$, the set of $(m-i)$-tuples of $(n+1)$-bit strings, and let $S_1 = \{(y_{i+1}, \ldots, y_m) \in S : \exists (x_1, \ldots, x_i) \in \{0,1\}^{n \times i}, D(b(x_1)...b(x_i)y_{i+1}...y_m) = 1\}$ and $S_2 = S \setminus S_1$. We note that (1) $D'(b(x_1)...b(x_{i-1})y_i y_{i+1}...y_m, i-1) = 1$ if and only if $D'(b(0^n)...b(0^n)y_i y_{i+1}...y_m, i-1) = 1$, and (2) $D'(b(x_1)...b(x_{i-1})b(x_i)y_{i+1}...y_m, i) = 1$ if and only if $(y_{i+1}, \ldots, y_m) \in S_1$.

Therefore,

$$\begin{aligned}
1/f(N) &\leq \mathbb{P}[D'(b(\mathbf{x_1})...b(\mathbf{x_{i-1}})\mathbf{y_i}\mathbf{y_{i+1}}...\mathbf{y_m}, i-1) = 1] \\
&= \mathbb{P}[D'(b(\mathbf{x_1})...b(\mathbf{x_{i-1}})\mathbf{y_i}\mathbf{y_{i+1}}...\mathbf{y_m}, i-1) = 1 | \mathbf{y_{i+1}}...\mathbf{y_m} \in S_1] * \mathbb{P}[\mathbf{y_{i+1}}...\mathbf{y_m} \in S_1] + \\
&\quad \mathbb{P}[D'(b(\mathbf{x_1})...b(\mathbf{x_{i-1}})\mathbf{y_i}\mathbf{y_{i+1}}...\mathbf{y_m}, i-1) = 1 | \mathbf{y_{i+1}}...\mathbf{y_m} \in S_2] * \mathbb{P}[\mathbf{y_{i+1}}...\mathbf{y_m} \in S_2] \\
&\leq 1 * \mathbb{P}[\mathbf{y_{i+1}}...\mathbf{y_m} \in S_1] + \\
&\quad \mathbb{P}[D'(b(0^n)...b(0^n)\mathbf{y_i}\mathbf{y_{i+1}}...\mathbf{y_m}, i-1) = 1 | \mathbf{y_{i+1}}...\mathbf{y_m} \in S_2] * \mathbb{P}[\mathbf{y_{i+1}}...\mathbf{y_m} \in S_2] \\
&= \mathbb{P}[D'(b(\mathbf{x_1})...b(\mathbf{x_{i-1}})b(\mathbf{x_i})\mathbf{y_{i+1}}...\mathbf{y_m}, i) = 1] + \\
&\quad \mathbb{P}[D'(b(0^n)...b(0^n)\mathbf{y_i}\mathbf{y_{i+1}}...\mathbf{y_m}, i-1) = 1 | \mathbf{y_{i+1}}...\mathbf{y_m} \in S_2] * \mathbb{P}[\mathbf{y_{i+1}}...\mathbf{y_m} \in S_2] \\
&\leq 1/s(N) + \mathbb{P}[D'(b(0^n)...b(0^n)\mathbf{y_i}\mathbf{y_{i+1}}...\mathbf{y_m}, i-1) = 1 | \mathbf{y_{i+1}}...\mathbf{y_m} \in S_2] * \mathbb{P}[\mathbf{y_{i+1}}...\mathbf{y_m} \in S_2] \\
&\quad \text{(in particular, this implies } \mathbb{P}[\mathbf{y_{i+1}}...\mathbf{y_m} \in S_2] > 0) \\
&\leq 1/s(N) + \mathbb{P}[D'(b(0^n)...b(0^n)\mathbf{y_i}\mathbf{y_{i+1}}...\mathbf{y_m}, i-1) = 1 | \mathbf{y_{i+1}}...\mathbf{y_m} \in S_2]
\end{aligned}$$

Hence, there is a sub-exponential f' such that

$$\mathbb{P}[D'(b(0^n)...b(0^n)\mathbf{y_i}\mathbf{y_{i+1}}...\mathbf{y_m}, i-1) = 1 | \mathbf{y_{i+1}}...\mathbf{y_m} \in S_2] \geq 1/f'(N),$$

which means there is a particular $y_{i+1}...y_m \in S_2$ such that

$$\mathbb{P}[D'(b(0^n)...b(0^n)\mathbf{y_i}y_{i+1}...y_m, i-1) = 1] \geq 1/f'(N).$$

We define another nondeterministic circuit C by $C(y_i) = D'(b(0^n)...b(0^n)y_iy_{i+1}...y_m, i-1)$ with $y_i \in \{0,1\}^{n+1}$ as input. As D' is of sub-exponential size in N and n is polynomially related to N, C is of sub-exponential size in n. We now argue that C breaks b in the desired sense. For infinitely many n's, (1) $\mathbb{P}[C(\mathbf{y_i}) = 1] = \mathbb{P}[D'(b(0^n)...b(0^n)\mathbf{y_i}y_{i+1}...y_m, i-1) = 1] \geq 1/f'(N)$, and (2) $\mathbb{P}[C(b(\mathbf{x_i})) = 1] = \mathbb{P}[D'(b(0^n)...b(0^n)b(\mathbf{x_i})y_{i+1}...y_m, i-1) = 1] = 0$ because $y_{i+1}...y_m \in S_2$ implies there is no $x_1, \ldots, x_i$ such that $D(b(x_1)...b(x_i)y_{i+1}...y_m) = 1$. (1) and (2) contradict that b is a demi-bit. ■

The condition $c < 1$ in Algorithm 3.1.1 guarantees $n = N^{1-c}$ is polynomially related to N and an infinite monotone sequence of N yields an infinite monotone sequence of n.

4 CONSEQUENCES OF THE PURPORTED NONEXISTENCE OF DEMI-BITS

In this chapter, we develop PAC-learning algorithms from breaking PRGs.

4.1 PAC-learning

We formulate the following nonuniform version of PAC-learning:

Definition 4.1.1 (PAC-learning, nonuniform). *A circuit class $\mathcal{C}$ is **learnable** (over the uniform distribution) by a circuit class $\mathcal{D}$ up to error ε with confidence δ if there is a randomized oracle family $L = \{D_n\} \in \mathcal{D}$ such that for every family $f : \{0,1\}^n \to \{0,1\}$ computable by $\mathcal{C}$ and every large enough n, we have:*

(1) $\mathbb{P}_w[L^f(1^n, w)\ (1-\varepsilon)-\text{approximates } f] \geq \delta$, where w is the random input bits to L^f and the output $L^f(1^n, w)$ of L^f is a string representation of an approximator f' of f. $L^f(1^n, w)$ can be a description of a uniform or nonuniform algorithm of the following types: deterministic, nondeterministic, co-nondeterministic, randomized.

(2) For every f and w, $L^f(1^n, w)$ is $\mathcal{D}$-evaluable: there is another circuit family $E \in \mathcal{D}$ such that for every possible output $L^f(1^n, w)$ of L and every $x \in \{0,1\}^n$ given as the input to E, E computes $L^f(1^n, w)$ on x. When $L^f(1^n, w)$ is (co-)nondeterministic or randomized, E is allowed to be (co-)nondeterministic or randomized respectively.

For a such learner L, we also say $\mathcal{C}$ is learnable by L or every $C \in \mathcal{C}$ is learnable by L.

Remarks.

1. We say L^f uses membership query if it somehow selects the set of queries made to the oracle gates. We say L^f uses uniformly distributed random examples if the set of queries made is sampled uniformly at random. The above learning model is general enough to admit both kinds of learners and also any kind of mixture. In this thesis, we will only consider learners using uniformly distributed random examples. Thus by "learning", we always mean learning using uniformly distributed random examples.
2. The defined learning model is also general enough to admit learning over other distributions (i.e. we change the distribution of w in condition (1)). When $\mathcal{D}$ contains $P/poly$, learning over any polynomially generated distribution is equivalent to learning over the uniform distribution.
3. A common mistake in the literature is to leave out Condition (2). However, Condition (2) is indispensable here. The reason is that without this restriction, $P/poly$ can be efficiently learned, which is widely believed not to be the case (e.g., cf. [29]). It is an easy exercise to prove the learnability of $P/poly$ without the second condition by applying the Occam's Razor theorem established in [10]. Intuitively, a learner can learn $P/poly$ efficiently by remembering the samples and postponing all the "learning" to the hypothesis evaluation stage.
4. The confidence δ and accuracy $1-\varepsilon$ of a learner in $\mathcal{D}$ can be efficiently boosted to constants less than 1 in standard ways (cf. [23]) when they are not negligible with respect to $\mathcal{D}$. For example, when $\mathcal{D} = P/poly$, it is sufficient for δ and $1-\varepsilon$ to achieve $p(n)$ and $1/2 + q(n)$ respectively for any polynomials p and q. Beyond the remark here, boosting will be digress from the theme of this thesis.

4.2 Learning based on nonexistence assumptions

We recall a construction from [8]: for a positive integer m and a circuit $C : \{0,1\}^n \to \{0,1\}$, define generator $G_{m,C} : \{0,1\}^{mn} \to \{0,1\}^{mn+m}$, which maps m n-bit strings $x_1, ..., x_m$ to $x_1, C(x_1), ..., x_m, C(x_m)$.

We reformulate Theorem 7 on average-case learning in [8] into our PAC-learning framework as the following lemma:

Lemma 4.2.1. *Given a circuit class $\mathcal{C}$, if there is an m and an $s(n)$-size circuit D such that for every $C \in \mathcal{C}$*

$$\mathbb{P}[D(y) = 1] - \mathbb{P}[D(G_C(x)) = 1] \geq 1/s, \text{ where } G_C = G_{m,C},$$

then there is a randomized polynomial time (in n and $|\langle D\rangle|$, where $\langle D\rangle$ is a given string representation of D) algorithm L that learns $\mathcal{C}$ with confidence $1/2m^2 s$ up to error $1/2 - 1/2ms$.

In particular, if D is a nondeterministic or co-nondeterministic circuit, the output of L is allowed to be a nondeterministic or co-nondeterministic algorithm.

Proof. Given any $C \in \mathcal{C}$, L randomly chooses an $i \in [m]$, bits $r_1, ..., r_m$, and n-bit strings $x_1, ..., x_m$ except x_i, queries $C(x_1), ..., C(x_{i-1})$, and outputs C', which predicts C as follows: given any n-bit input x_i, C' emulates D on $(x_1, C(x_1), ..., x_{i-1}, C(x_{i-1})$ to get an output bit p_i. If $p_i = 1$, C' outputs $\bar{r_i}$; if $p_i = 0$, C' outputs r_i.

We next want to prove that L indeed learns C in the desired sense. Given random bits $r_1, ..., r_m$ and random n-bit strings $x_1, ..., x_m$, we define $p_i := D(x_1, C(x_1), ..., x_{i-1}, C(x_{i-1}), x_i, r_i, ..., x_m, r_m)$. Then (note: here we are applying a hybrid argument),

$$\begin{aligned}1/s &\leq \mathbb{P}[D(x) = 1] - \mathbb{P}[D(G_C(x)) = 1]\\ &= \mathbb{P}[p_1 = 1] - \mathbb{P}[p_m = 1]\\ &= \sum_{i=1}^{m-1}(\mathbb{P}[p_i = 1] - \mathbb{P}[p_{i+1} = 1])\end{aligned}$$

Hence, there exist i such that $\mathbb{P}[p_i = 1] - \mathbb{P}[p_{i+1} = 1] \geq 1/ms$, and therefore the i L chooses satisfies $\mathbb{P}[p_i = 1] - \mathbb{P}[p_{i+1} = 1] \geq 1/ms$ with probability $\geq 1/m$. When L has successfully chosen such an i,

$$\begin{aligned}&\mathop{\mathbb{P}}_{\substack{r_1,...,r_m\\ x_1,...,x_m}}[C'(x_i) = C(x_i)]\\ &= \mathbb{P}[p_i = 1, r_i \neq C(x_i)] + \mathbb{P}[p_i = 0, r_i = C(x_i)]\\ &= \frac{1}{2}\mathbb{P}[p_i = 1 | r_i \neq C(x_i)] + \frac{1}{2}\mathbb{P}[p_i = 0 | r_i = C(x_i)]\\ &= \frac{1}{2}\mathbb{P}[p_i = 1 | r_i \neq C(x_i)] + \frac{1}{2}(1 - \mathbb{P}[p_i = 1 | r_i = C(x_i)])\\ &= \frac{1}{2} + \frac{1}{2}\mathbb{P}[p_i = 1 | r_i \neq C(x_i)] - \frac{1}{2}\mathbb{P}[p_i = 1 | r_i = C(x_i)]\\ &= \frac{1}{2} + \frac{1}{2}\mathbb{P}[p_i = 1 | r_i \neq C(x_i)] + (\frac{1}{2}\mathbb{P}[p_i = 1 | r_i = C(x_i)] - \mathbb{P}[p_i = 1 | r_i = C(x_i)])\\ &= \frac{1}{2} + (\mathbb{P}[p_i = 1, r_i \neq C(x_i)] + \mathbb{P}[p_i = 1, r_i = C(x_i)]) - \mathbb{P}[p_i = 1 | r_i = C(x_i)]\\ &= \frac{1}{2} + \mathbb{P}[p_i = 1] - \mathbb{P}[p_{i+1} = 1]\\ &\geq \frac{1}{2} + \frac{1}{ms}\end{aligned}$$

Let $p = \mathop{\mathbb{P}}_{\substack{r_1,...,r_m\\ x_1,...,x_m \text{ except } x_i}}[\mathop{\mathbb{P}}_{x_i}[C'(x_i) = C(x_i)] >= 1/2 + 1/2ms]$. As there exist $0 \leq a < 1/2$ such that $1/2 + 1/2ms + a$ represents the average accuracy of the predictor C' when $r_1, ..., r_m, x_1, ..., x_{i-1}, x_{i+1}, ..., x_m$

satisfy $\mathbb{P}_{x_i}[C'(x_i) = C(x_i)] >= 1/2 + 1/2ms$, we have

$$p(\frac{1}{2} + \frac{1}{2ms} + a) + (1-p)(\frac{1}{2} + \frac{1}{2ms}) \geq \mathbb{P}_{\substack{r_1,\ldots,r_m\\x_1,\ldots,x_m}}[C'(x_i) = C(x_i)] \geq \frac{1}{2} + \frac{1}{ms},$$

and thus

$$p \geq \frac{1}{2ms} \cdot \frac{1}{a} \geq \frac{1}{ms}.$$

Therefore, L outputs a $(1/2 + 1/2ms)$-accurate C' with confidence $1/m^2 s$. ■

Remarks. When D is nondeterministic: (1) if $r_i = 0$, then the C' learned by L^C is also nondeterministic; (2) if $r_i = 0$, then the C' learned by L^C is co-nondeterministic.

In the remaining of this section, we are going to derive learning algorithms based on the assumptions of the nonexistence of i.o. demi-bits and demi-bits respectively. It is obvious that the non-existence of i.o. demi-bits is a stronger assumption than the non-existence of demi-bits. We will also see that, given our proof strategy, the stronger assumption yields a better learning result, in a sense that will be clear later.

Based on a sketch in [28], we establish the following theorem:

THEOREM 4.2.2. *Assume the nonexistence of i.o. demi-bits. Then for every $c \in \mathbb{N}$, $Circuit[n^c]$ is learnable by $Circuit[2^{n^{o(1)}}]$ (by taking random examples) with confidence $1/2^{n^{o(1)}}$ up to error $1/2 - 1/2^{n^{o(1)}}$, where the learner is allowed to generate a nondeterministic or co-nondeterministic algorithm approximating the target function.*

PROOF. Assume the nonexistence of i.o. demi-bits and $c \in \mathbb{N}$. There is an encoding scheme for $Circuit[n^c]$ such that every $C \in Circuit[n^c]$ with n-bit input can be encoded as a string $\langle C \rangle$ of length $\leq d = d(n)$, where $d(n)$ is a polynomial. Set $m = n^d + 1$ and consider a generator $G : \{0,1\}^{mn+n^d} \to \{0,1\}^{mn+m}$, which interprets the last n^d input bits as a description of some $C \in Circuit[n^c]$ and then computes on the remaining mn bits of input as $G_{m,C}$. We note that G is in $P/poly$(whenever the encoding scheme for $Circuit[n^c]$ is reasonable). Because $G_{m,C}$ is not an i.o. demi-bit, there is nondeterministic circuit D of sub-exponential size such that for all sufficiently large n's

$$\mathbb{P}[D(y) = 1] \geq 1/|D| \text{ and } \mathbb{P}[D(G(x) = 1] = 0.$$

In particular, for every $C \in Circuit[n^c]$, $\mathbb{P}[D(G(\langle C \rangle, x) = 1] = 0$. That is $\mathbb{P}[D(G_{m,C}(x)) = 1] = 0$ for every $C \in Circuit[n^c]$. Therefore,

$$\mathbb{P}[D(y) = 1] - \mathbb{P}[D(G_{m,C}(x)) = 1] \geq 1/|D|.$$

Because $m = n^d + 1$ is polynomial in n and $|D|$ is sub-exponential, by Lemma 4.2.1, C can be learned by a randomized circuit family of size $2^{n^{o(1)}}$ with confidence $1/2^{n^{o(1)}}$ up to error $1/2 - 1/2^{n^{o(1)}}$. ■

If we weaken the assumption to the non-existence of demi-bits, with the same proof, we are able to learn $Circuit[n^c]$ in a weaker sense formulated as below:

THEOREM 4.2.3. *Assume the nonexistence of demi-bits. Then for every every $c \in \mathbb{N}$, there is an infinite monotone sequence $\{n_i\} \subseteq \mathbb{N}$ such that $Circuit[n^c]$ is learnable by $Circuit[2^{n^{o(1)}}]$ (by taking random examples) with confidence $1/2^{n^{o(1)}}$ up to error $1/2 - 1/2^{n^{o(1)}}$ for every $n \in \{n_i\}$, where the learner is allowed to generate a nondeterministic or co-nondeterministic algorithm approximating the target function.*

Because the existence of $N\tilde{P}/qpoly$-natural proofs (a weaker assumption than the existence assumption of $NP/poly$-natural proofs) rules out the existence of i.o. super-bits [33], by Theorem 4.2.2, we have:

Corollary 4.2.4. *Assume the existence of i.o. demi-bits implies the existence of i.o. super-bits. If there exists an* $N\tilde{P}/qpoly$*-natural property useful against* $P/poly$*, then for every* $c \in \mathbb{N}$, $Circuit[n^c]$ *is learnable by* $Circuit[2^{n^{o(1)}}]$ *(by taking random examples) with confidence* $1/2^{n^{o(1)}}$ *up to error* $1/2 - 1/2^{n^{o(1)}}$*, where the learner is allowed to generate a nondeterministic or co-nondeterministic algorithm approximating the target function.*

5 SUPER-BITS: CHARACTERIZATIONS AND CONNECTIONS

In Section 5.1, we review the notion of predictability in the deterministic setting. In Section 5.2, we introduce new notions of predictability beyond the deterministic setting and study nondeterministic hardness from a predictability aspect. In Section 5.3, we review the concepts of one-way functions and hard-core predicates and as well their known connections with strong PRGs. In Section 5.4, the last section in this chapter, we introduce a dramatically new concept, super-core predicates, and investigate on its connections with super-bits.

In this chapter, we will use a slightly modified definition of super-bits, which is parallel to Definition 2.3.3 in the deterministic setting: a PRG $g : \{0,1\}^n \to \{0,1\}^{m(n)}$ is called super-bits if for every D in $NP/poly$, every polynomial p, and all sufficiently large n's,

$$\mathbb{P}[D(U_{m(n)}) = 1] - \mathbb{P}[D(g(U_n)) = 1] < 1/p(n).$$

Namely, g is super-bits if g is safe against all $NP/poly$-distinguishers.

What we have achieved in this chapter provides a big step forward to better understand the nondeterministic hardness of PRGs and to suggest a sensible definition of one-way functions in the nondeterministic setting.

5.1 Deterministic predictability

In this section, we review the notion of predictability in the deterministic setting and the equivalence between deterministic unpredictability and super-polynomial hardness of PRGs. The results reviewed in Section 5.1 are mostly adapted from [15].

Definition 5.1.1 (probability ensembles). *A **probability ensemble** (or **ensemble** for short) is an infinite sequence of random variables $(Z_n)_{n\in\mathbb{N}}$. Each Z_n ranges over $\{0,1\}^{l(n)}$, where $l(n)$ is polynomially related to n (i.e., there is a polynomial p such that for every n it holds that $l(n) \le p(n)$ and $p(l(n)) \ge n$.*

We say an ensemble is polynomially generated if there is $g \in P/poly$ such that $g(U_n) = Z_n$. In this paper, we are only interested in polynomially generated ensembles because they are easy to generate to imitate other probability ensembles (e.g., the uniform ensemble as we will see in the next definition) from a cryptography perspective. Thus, every ensemble we consider from now on will be implicitly assumed to be polynomially generated if not specified otherwise.

Definition 5.1.2 (strong-pseudorandom ensembles). *The probability ensemble $(Z_n)_{n\in\mathbb{N}}$ is **strongly pseudorandom** if for every algorithm D in $P/poly$, every polynomial p, and all sufficiently large n's,*

$$|\mathbb{P}_{U_{m(n)}}[D(U_{m(n)}, 1^n) = 1] - \mathbb{P}_{Z_n}[D(Z_n, 1^n) = 1]|,$$

where $m = |Z_n|$ and U_m is the uniform distribution

The input 1^n allows D is run poly-time in n and informs D of the value n. But for the sake of notation simplicity, 1^n is often omitted and will be omitted from now on.

We note that if g is a strong PRG, then $g(U_n)$ is a strong-pseudorandom ensemble. We will see, for $g(U_n)$, being strongly pseudorandom is equivalent to being unpredictable.

Definition 5.1.3 ((deterministic) predictability). *An ensemble $(Z_n)_{n\in\mathbb{N}}$ is called **predictable** or **P/poly-predictable** is there exist an algorithm $\mathcal{A}$ in $P/poly$, a polynomial p, infinitely many n's, and an $i(n) < |Z_n|$ for each of those n's such that*

$$\mathbb{P}[\mathcal{A}(Z_n[1...i] = Z_n[i+1])] \ge \frac{1}{2} + 1/p(n).$$

*An ensemble $(Z_n)_{n\in\mathbb{N}}$ is **unpredictable** if it is not predictable.*

THEOREM 5.1.4. *An ensemble is strong-pseudorandom if and only if it is unpredictable.*

We will see in the next section a nondeterministic variant of the last theorem.

5.2 Nondeterministic predictability

In this section, we propose four new notions of unpredictability beyond the deterministic setting and characterise super-hardness of PRGs based on the new notions.

Before we define nondeterministic predictability, we emphasise an important distinction between a decision problem and a single-bit-output computing problem in the nondeterministic setting. We say a nondeterministic algorithm $\mathcal{A}$ is a (total) **function-computing** algorithm, which **computes** a function $f : \{0,1\}^n \to \{0,1\}$ if for every input $x \in \{0,1\}^n$, (1) a computation branch either yields $\perp$ (which indicates a failure) or $f(x)$, and (2) there is a computation branch yielding $f(x)$. In the following discussion, by default, any algorithm $\mathcal{A} : \{0,1\}^n \to \{0,1\}$ will still be considered as an algorithm for solving decision problems, unless we explicitly mention that $\mathcal{A}$ is function-computing.

DEFINITION 5.2.1 (NONDETERMINISTIC PREDICTABILITY).

NP/poly-predictability. *An ensemble $(Z_n)_{n\in\mathbb{N}}$ is NP/poly-predictable if there exist an algorithm $\mathcal{A}$ in NP/poly, a polynomial p, infinitely many n's, and an $i(n) < |Z_n|$ for each of those n's such that*

$$\mathbb{P}[\mathcal{A}(Z_n[1...i]) = Z_n[i+1]] \geq 1/2 + 1/p(n).$$

An ensemble $(Z_n)_{n\in\mathbb{N}}$ is NP/poly-unpredictable if it is not NP/poly-predictable.

coNP/poly-predictability. *An ensemble $(Z_n)_{n\in\mathbb{N}}$ is coNP/poly-predictable if there exist an algorithm $\mathcal{A}$ in coNP/poly, a polynomial p, infinitely many n's, and an $i(n) < |Z_n|$ for each of those n's such that*

$$\mathbb{P}[\mathcal{A}(Z_n[1...i]) = Z_n[i+1]] \geq 1/2 + 1/p(n).$$

An ensemble $(Z_n)_{n\in\mathbb{N}}$ is coNP/poly-unpredictable if it is not coNP/poly-predictable.

$\cup$-predictability. *An ensemble $(Z_n)_{n\in\mathbb{N}}$ is called $\cup$-predictable (a short for $(NP/poly \cup coNP/poly)$-predictable) if it is NP/poly-predictable or coNP/poly-predictable. An ensemble $(Z_n)_{n\in\mathbb{N}}$ is $\cup$-unpredictable if it is not $\cup$-predictable.*

$\cap$-predictability. *An ensemble $(Z_n)_{n\in\mathbb{N}}$ is called $\cap$-predictable (a short for $(NP/poly \cap coNP/poly)$-predictable) if there exist a poly-time nondeterministic **function-computing** algorithm $\mathcal{A}$, a polynomial p, infinitely many n's, and an $i(n) < |Z_n|$ for each of those n's such that*

$$\mathbb{P}[\mathcal{A}(Z_n[1...i]) = Z_n[i+1]] \geq 1/2 + 1/p(n).$$

An ensemble $(Z_n)_{n\in\mathbb{N}}$ is $\cap$-unpredictable if it is not $\cap$-predictable.

Remark. A reader should not mistake the definition of being $\cap$-predictable as being *NP/poly*-predictable $\wedge$ *coNP/poly*-predictable. In fact, being $\cap$-predictable is a stronger assumption than being *NP/poly*-predictable and *coNP/poly*-predictable because a function f is nondeterministically computable if and only if it is in $NP/poly \cap coNP/poly$. A more detailed proof is provided below. Nevertheless, being $\cap$-unpredictable is still a non-trivial property because it is stronger than *P/poly*-unpredictable (as $P/poly \subseteq NP/poly \cap coNP/poly$).

LEMMA 5.2.2. *If an ensemble (Z_n) is $\cap$-predictable, it is both NP/poly-predictable and coNP/poly-predictable.*

PROOF. Suppose there exist a poly-time nondeterministic **function-computing** algorithm $\mathcal{A}$, a polynomial p, infinitely many n's, and an $i(n) < |Z_n|$ for each of those n's such that

$$\mathbb{P}[\mathcal{A}(Z_n[1...i]) = Z_n[i+1]] \geq 1/2 + 1/p(n).$$

We define $\mathcal{A}_1$ in $NP/poly$ and $\mathcal{A}_0$ in $coNP/poly$ as follows:

> Given input $Y \in \{0,1\}^i$, $\mathcal{A}_j (j = 0, 1)$ mimics $\mathcal{A}$ on input Y to get an output bit c. If $c = \perp$, $\mathcal{A}_j$ outputs $1 - j$, and otherwise $\mathcal{A}_j$ outputs c.

By the definition of $\mathcal{A}_j$, $\mathcal{A}_j(Y) = \mathcal{A}(Y)$. Thus, $\mathbb{P}[\mathcal{A}_j(Z_n[1...i]) = Z_n[i+1]] \geq 1/2 + 1/p(n)$ for $j = 0, 1$. ■

Proposition 5.2.3. *If $g : \{0,1\}^n \longrightarrow \{0,1\}^{m(n)}$ is super-bit(s), then $g(U_n)$ is $\cap$-unpredictable.*

Proof. Suppose, for a contradiction, $g(U_n)$ is $\cap$-predictable. there exist a poly-time nondeterministic **function-computing** algorithm $\mathcal{A}$, a polynomial p, infinitely many n's, and an $i(n) < |Z_n|$ for each of those n's such that

$$\mathbb{P}[\mathcal{A}(Z_n[1...i]) = Z_n[i+1]] \geq 1/2 + 1/p(n).$$

For a fixed n, denote $Z_i = g(U_n)[1...i]$ and $z_{i+1} = g(U_n)[i+1]$.

We construct algorithm D to distinguish U_m from $g(U_n)$:

> Given input $Y \in \{0,1\}^m$, D runs $\mathcal{A}$ on $Y[1...i]$ to obtain an output bit c. If $c = \perp$, D outputs 0. When $c \in \{0,1\}$, if $c \neq Y[i+1]$, D outputs 1, and else D outputs 0.

For a fixed n, we denote $Z_i = g(U_n)[1...i]$ and $z_{i+1} = g(U_n)[i+1]$ and use b to denote a random bit. Now, we have:

$$\begin{aligned}
\mathbb{P}[D(U_m) = 1] - \mathbb{P}[D(g(U_n)) = 1] &= \mathbb{P}[\mathcal{A}(U_i) \neq b] - \mathbb{P}[\mathcal{A}(Z_i) \neq z + i + 1] \\
&= 1/2 - (1 - \mathbb{P}[\mathcal{A}(Z_i) = z + i + 1]) \\
&= \mathbb{P}[\mathcal{A}(Z_i) = z + i + 1] - 1/2 \\
&\geq 1/p(n).
\end{aligned}$$

■

Proposition 5.2.4. *If $g(U_n)$ is $\cup$-unpredictable (i.e., $NP/poly$-unpredictable $\wedge$ $coNP/poly$-unpredictable), where $g : \{0,1\}^n \longrightarrow \{0,1\}^{m(n)} \in P/poly$ and $m(n) > n$, then g is super-bit(s).*

Proof. Suppose, for a contradiction, g is not super-bits. For any fixed n, define hybrids $H_i = g(U_n)[0...i]U_m[i+1...m]$ and denote denote $Z_i = g(U_n)[1...i]$ and $z_i = g(U_n)[i]$. Then there exist an algorithm D in $P/poly$, a polynomial p, infinitely many n's, and an i for each of these n's such that

$$\mathbb{P}[D(H_i) = 1] - \mathbb{P}[D(H_{i+1}) = 1] \geq 1/p(n).$$

Therefore, for each such n, there is a fixed string w such that

$$\mathbb{P}[D(Z_i b, w) = 1] - \mathbb{P}[D(Z_{i+1}, w) = 1] \geq 1/p(n), \tag{$*$}$$

where we use b to denote a random bit. We may omit writing w from now on.

We construct an algorithm $\mathcal{A}_1$ in $NP/poly$ and $\mathcal{A}_2$ in $coNP/poly$to predict z_{i+1} based Z_i as follows:

> Given $Y_i \in \{0,1\}^i$ as input, $\mathcal{A}_1$ does whatever D does on $Y_i 0 w$; $\mathcal{A}_2$ does whatever D does on $Y_i 1 w$ but then flip the bit got.

The definitions of $\mathcal{A}_1, \mathcal{A}_2$ imply that $\mathcal{A}_1(Y_i) = D(Y_i 0)$ and $\mathcal{A}_2(Y_i) = \overline{D(Y_i 1)}$ (w omitted). Hence,

$$\begin{aligned}
(1) &:= \mathbb{P}[\mathcal{A}_1(Z_i) = z_{i+1}] \\
&= \mathbb{P}[D(Z_i 0) = z_{i+1}] \\
&= \mathbb{P}[D(Z_i 0) = 0 \wedge z_{i+1} = 0] + \mathbb{P}[D(Z_i 0) = 1 \wedge z_{i+1} = 1] \\
&= \mathbb{P}[D(Z_i z_{i+1}) = 0 \wedge z_{i+1} = 0] + \mathbb{P}[D(Z_i \overline{z_{i+1}}) = 1 \wedge z_{i+1} = 1],
\end{aligned}$$

and

$$\begin{aligned}(2) &:= \mathbb{P}[\mathcal{A}_2(Z_i) = z_{i+1}] \\ &= \mathbb{P}[D(Z_i 1) \neq z_{i+1}] \\ &= \mathbb{P}[D(Z_i 1) = 0 \land z_{i+1} = 1] + \mathbb{P}[D(Z_i 1) = 1 \land z_{i+1} = 0] \\ &= \mathbb{P}[D(Z_i z_{i+1}) = 0 \land z_{i+1} = 1] + \mathbb{P}[D(Z_i \overline{z_{i+1}}) = 1 \land z_{i+1} = 0].\end{aligned}$$

Therefore,

$$(1) + (2) = \mathbb{P}[D(Z_i z_{i+1}) = 0] + \mathbb{P}[D(Z_i \overline{z_{i+1}}) = 1].$$

As

$$\mathbb{P}[D(Z_i z_{i+1}) = 0] = 1 - \mathbb{P}[D(Z_{i+1}) = 1]$$

and

$$\begin{aligned}2\mathbb{P}[D(Z_i, b) = 1] &= 2\mathbb{P}[D(Z_i, b) = 1 \land b = z_{i+1}] + 2\mathbb{P}[D(Z_i, b) = 1 \land b \neq z_{i+1}] \\ &= 2\mathbb{P}[b = z_{i+1}]\mathbb{P}[D(Z_i, b) = 1 | b = z_{i+1}] + 2\mathbb{P}[b \neq z_{i+1}]\mathbb{P}[D(Z_i, b) = 1 | b \neq z_{i+1}] \\ &= \mathbb{P}[D(Z_i, z_{i+1}) = 1] + \mathbb{P}[D(Z_i, \overline{z_{i+1}}) = 1],\end{aligned}$$

$$\begin{aligned}(1) + (2) &= (1 - \mathbb{P}[D(Z_{i+1}) = 1]) + (2\mathbb{P}[D(Z_i b) = 1] - \mathbb{P}[D(Z_{i+1}) = 1]) \\ &= 1 + 2(\mathbb{P}[D(Z_i b) = 1] - \mathbb{P}[D(Z_{i+1}) = 1]) \\ &\geq 1 + 2/p(n) \text{ by } (*).\end{aligned}$$

Hence, either $(1) = \mathbb{P}[\mathcal{A}_1(Z_i) = z_{i+1}] \geq 1/2 + 1/p(n)$ for infinitely many n's or $(2) = \mathbb{P}[\mathcal{A}_2(Z_i) = z_{i+1}] \geq 1/2 + 1/p(n)$ for infinitely many n's. That is, the ensemble (Z_n) is either $NP/poly$-predictable or $coNP/poly$-predictable. ■

Summary. We summarise Lemma 5.2.2, Proposition 5.2.3, and Proposition 5.2.4 together as (here, $A \leq B$ denotes B implies A):

$$\cap\text{-unpredictable} \leq \text{super-polynomially nondeterministic-hard} \leq \cup\text{-unpredictable}$$

and

$$\cap\text{-unpredictable} \leq NP/poly\text{-unpredictable} \lor coNP/poly\text{-unpredictable} \leq \cup\text{-unpredictable}.$$

In contrast to Theorem 5.1.4, which is an precises characterization of standard hardness from a predictability perspective, we have so far only obtained inaccurate characterizations of super-hardness in the nondeterministic setting. I am inclined to the viewpoint that we might be unable to obtain an exact characterization of super-hardness in terms of predictability, because we will see in Section 5.4, super-hardness is not just about algorithm (e.g., algorithms used to witness randomness or used to predict) behaviour on the range elements (i.e., y such that $\mathbb{P}[Z_n = y] > 0$) but may also concern behaviour on the non-range elements (i.e., y such that $\mathbb{P}[Z_n = y] = 0$). In other words, it is very possible that at least some of the inequalities above are strict.

Open Problem 5.2.5. *Could the inequalities in* ***Summary*** *be further refined or classified? For example, is there any relation between super-hardness and $NP/poly$-unpredictable $\lor$ $coNP/poly$-unpredictable?*

5.3 One-way functions and hard-core predicates

In this section, we review the concepts of one-way functions and hard-core predicates and the equivalence between the existence of a strong PRG and the existence of a hard-core of some one-way function. The results reviewed in Section 5.3 are mostly adapted from [15].

Definition 5.3.1 (one-way functions). *A function $f : \{0,1\}^* \longrightarrow \{0,1\}^*$ in $P/poly$ is called* ***one-way*** *if for every $\mathcal{A}$ in $P/poly$, every polynomial $p(\cdot)$, and all sufficiently large n's,*

$$\mathop{\mathbb{P}}_{x\in\{0,1\}^n}[\mathcal{A}(f(x),1^n) \in f^{-1}(f(x))] < \frac{1}{p(n)}.$$

The input 1^n is for technical reason. It allows the algorithm $\mathcal{A}$ to run in polynomial time in $n = |x|$, which is important when f dramatically shrinks its input (e.g., when $|f(x)| = O(log|x|)$). When the auxiliary input 1^n is not necessary (e.g., when f is length-preserving), we may omit it. Intuitively, a function f in $P/poly$ is one-way if it is "typically" hard to invert, when the probability is taken over the input distribution, for all efficient algorithms.

It is known that the existence of one-way functions and the existence of strong PRGs:

theorem 5.3.2. *One-way functions exist if and only if strong PRGs exist.*

As we can construct a length-preserving one-way function from an arbitrary one-way function, Theorem 5.3.2 can be alternatively stated as:

theorem 5.3.3. *Length-preserving one-way functions exist if and only if strong PRGs exist.*

The backward construction of Theorem 5.3.2 is rather straightforward, while the forward, in fact, relies on a concept closed related to one-way functions, called hard-core predicates. Even though with the help of hard-core predicates to produce one more bit from a seed x, the known proof of the forward direction in Theorem 5.3.3 is still rather involved.

Definition 5.3.4 (hard-core predicates). *A predicate $b : \{0,1\}^* \longrightarrow \{0,1\}$ in $P/poly$ is called a* ***hard-core*** *of a function f if there do not exist an algorithm $\mathcal{A}$ in $P/poly$, a polynomial $p(\cdot)$, and infinitely many n's such that*

$$\mathop{\mathbb{P}}_{x\in\{0,1\}^n}[\mathcal{A}(f(x),1^n) = b(x)] \geq \frac{1}{2} + \frac{1}{p(n)}.$$

In other words, b is a hard-core of f if it is safe against (in terms of prediction) all $P/poly$ algorithms, which may be due to an information loss of f (e.g, $b(x) = x[n]$ is a hard-core of $f(x) = x[1...(n-1)]$) or to the difficulty of inverting f.

If b is a hard-core of any f, then $\mathbb{P}[b(x) = 0] \approx \mathbb{P}[b(x) = 1] \approx 1/2$ (otherwise, we can predict b by outputting the constant $argmax_{b\in\{0,1\}}(\mathbb{P}[b(x) = b])$).

One-way functions and hard-core predicates are "paired" by the construction of a "generic" hard-core:

theorem 5.3.5. *For any one-way function f, the inner-product mod 2 of x and y, denoted as $\langle x, y\rangle$, is a hard-core of $f'(x,y) := (f(x), y)$.*

In particular, if f is length-preserving, so is f'. Indeed, in the theorem, we should also define f' and b on the odd length inputs (x, y, b), where $|x| = |y|$ and b is a single bit, but this case is often omitted as we can trivially "ignore" the last bit (use the same idea as in Lemma 2.5.2) and define $f'(x, y, b) = (f(x), y, b)$ and $b(x, y, b) = \langle x, y\rangle$.

This theorem states that every one-way function f, in a weaker sense, "has" a hard-core; or in other words, $\langle x, y\rangle$ is "almost universal". It is an easy exercise that a true universal hard-core for every one-way function does not exist.

Therefore, Theorem 5.3.3 can be reformulated as:

THEOREM 5.3.6. *There is a hard-core b of some length-preserving one-way function if and only if there is strong PRG g.*

We will develop a non-deterministic variant of the last theorem in the next section.

5.4 Super-core predicates

In this section, we introduce a dramatically new concept, super-core predicates, and investigate on its connections with super-bits and on which functions could or could not have super-cores.

DEFINITION 5.4.1 (SUPER-CORE PREDICATES). *A predicate $b : \{0,1\}^n \longrightarrow \{0,1\}$ in $P/poly$ is called a **super-core** of a function $f : \{0,1\}^n \longrightarrow \{0,1\}^{m(n)}$ if there do not exist an algorithm $\mathcal{A}_1$ in $NP/poly$, an algorithm $\mathcal{A}_2$ in $coNP/poly$, polynomial $p(\cdot)$, and infinitely many n's such that*

$$\mathbb{P}_{x\in\{0,1\}^n}[\mathcal{A}_1(f(x),1^n) = b(x) = 0] + \frac{1}{2}\mathbb{P}_{y\in\{0,1\}^m}[\mathcal{A}_1(y,1^n) = 1] \geq \frac{1}{2} + \frac{1}{p(n)}$$

or

$$\mathbb{P}_{x\in\{0,1\}^n}[\mathcal{A}_2(f(x),1^n) = b(x) = 1] + \frac{1}{2}\mathbb{P}_{y\in\{0,1\}^m}[\mathcal{A}_2(y,1^n) = 0] \geq \frac{1}{2} + \frac{1}{p(n)}$$

For convenience, we define some abbreviations for formulas above: $①(\mathcal{A}_1) := \mathbb{P}_{x\in\{0,1\}^n}[\mathcal{A}_1(f(x)) = b(x) = 0]$, $②(\mathcal{A}_2) := \mathbb{P}_{x\in\{0,1\}^n}[\mathcal{A}_2(f(x)) = b(x) = 1]$, $③(\mathcal{A}_1) := \frac{1}{2}\mathbb{P}_{y\in\{0,1\}^{m(n)}}[\mathcal{A}_1(y) = 1]$, $④(\mathcal{A}_2) := \frac{1}{2}\mathbb{P}_{y\in\{0,1\}^{m(n)}}[\mathcal{A}_2(y) = 0]$ (the input 1^n is omitted).

Our intention here is to come up with a nondeterministic variant of hard-core predicates and to build some kind of relation between the existence of super-bits and the existence of this nondeterministic counterpart. The inequalities in Definition 5.4.1 concerns not only elements in the range but also elements that are not. A nondeterministic or co-nondeterministic adversary is unable to satisfy its corresponding inequality in any trivial way (e.g., by guessing a seed or by outputting the constant 0 or 1), though a more intuitive explanation of Definition 5.4.1 seems desirable.

We observe that, if b is a super-core of f, then b is also a super-core of any "shortened" f. Precisely, if $i : \mathbb{N} \longrightarrow \mathbb{N}$ is such that $i(n) \leq |f(1^n)|$ for every n, then b is a super-core of $f'(x) := f(x)[1...i(|x|)]$ because $f'(x)$ provides less information than $f(x)$.

Just as a super-bit is also a strong PRG, we have:

LEMMA 5.4.2. *If b is a super-core of function $f : \{0,1\}^n \longrightarrow \{0,1\}^{m(n)}$, b is a hard-core of f.*

PROOF. Suppose, for a contradiction, b is not a hard-core of f. Then there exist C in $P/poly$, a polynomial p, and infinitely many n's such that $\mathbb{P}[C(f(U_n) = b(U_n)] \geq 1/2 + 1/p(n)$. We note C is in both $NP/poly$and $coNP/poly$. As

$$①(C) + ②(C) + ③(C) + ④(C) = \mathbb{P}[C(f(U_n) = b(U_n)] + 1/2 \geq 1 + 1/p(n),$$

either $①(C) + ③(C) \geq 1/2 + 1/(2p(n))$ or $②(C) + ④(C) \geq 1/2 + 1/(2p(n))$. ∎

We now show that we can construct a super-bit from a super-core for some length-preserving function:

PROPOSITION 5.4.3. *If b is a super-core of function $f : \{0,1\}^n \longrightarrow \{0,1\}^n$ in $P/poly$, then $g(x) := f(x)b(x)$ is a super-bit.*

Proof. Suppose, for a contradiction, g is not a super-bit. Then there exist a distinguisher D in $P/poly$, a polynomial p, and infinitely many n's such that:

$$\mathbb{P}[D(U_{n+1}) = 1] - \mathbb{P}[D(g(U_n)) = 1] \geq 1/p(n).$$

We define algorithm $\mathcal{A}_1$ in $NP/poly$ and algorithm $\mathcal{A}_2$ in $coNP/poly$ to predict b as follows:

> Assume $Y \in \{0,1\}^n$ is given as input. $\mathcal{A}_1$ emulates D on $Y0$ and outputs what D outputs. $\mathcal{A}_2$ emulates D on $Y1$ and outputs the opposite of what D outputs.

The definitions of $\mathcal{A}_j$ imply $\mathcal{A}_1(Y) = D(Y0)$ and $\mathcal{A}_2(Y) = \overline{D(Y1)}$. Now, we have:

$$\begin{aligned}
①(\mathcal{A}_1) + ②(\mathcal{A}_2) &= \mathbb{P}[\mathcal{A}_1(f(U_n)) = b(U_n) = 0] + \mathbb{P}[\mathcal{A}_2(f(U_n)) = b(U_n) = 1] \\
&= \mathbb{P}[D(f(U_n)0) = 0 \wedge b(U_n) = 0] + \mathbb{P}[D(f(U_n)1) = 0 \wedge b(U_n) = 1] \\
&= \mathbb{P}[D(f(U_n)b(U_n)) = 0 \wedge b(U_n) = 0] + \mathbb{P}[D(f(U_n)b(U_n)) = 0 \wedge b(U_n) = 1] \\
&= \mathbb{P}[D(f(U_n)b(U_n)) = 0] \\
&= 1 - \mathbb{P}[D(g(U_n)) = 1],
\end{aligned}$$

and

$$\begin{aligned}
③(\mathcal{A}_1) + ④(\mathcal{A}_2) &= \frac{1}{2}\mathbb{P}[\mathcal{A}_1(U_n) = 1] + \frac{1}{2}\mathbb{P}[\mathcal{A}_2(U_n) = 0] \\
&= \frac{1}{2}\mathbb{P}[D(U_n0) = 1] + \frac{1}{2}\mathbb{P}[D(U_n1) = 1] \\
&= \mathbb{P}[D(U_{n+1}) = 1]
\end{aligned}$$

Therefore, for infinitely many n's,

$$①(\mathcal{A}_1) + ②(\mathcal{A}_2) + ③(\mathcal{A}_1) + ④(\mathcal{A}_2) = 1 + \mathbb{P}[D(U_{n+1}) = 1] - \mathbb{P}[D(g(U_n)) = 1] \geq 1 + 1/p(n),$$

which implies either $①(\mathcal{A}_1) + ③(\mathcal{A}_1) \geq 1 + 1/(2p(n))$ for infinitely many n's or $②(\mathcal{A}_2) + ④(\mathcal{A}_2) \geq 1 + 1/(2p(n))$ for infinitely many n's. ■

Before we establish Proposition 5.4.5, which is the converse of Proposition 5.4.3, we need the following auxiliary lemma:

Lemma 5.4.4. *If $g : \{0,1\}^n \to \{0,1\}^{n+1}$ is a strong PRG and define $f(x)b(x) := g(x)$, where $b(x)$ is the last bit of $g(x)$, then b is a hard-core of $f \in P/poly$.*

Proof. Suppose for contradiction, b is not a hard-core of f. Then there exist $\mathcal{A}$ in $P/poly$, a polynomial p, and infinitely many n's such that

$$\mathbb{P}[\mathcal{A}(f(U_n)) = b(U_n)] \geq 1/2 + 1/p(n).$$

We construct D to break g as follows:

> Given $Y \in \{0,1\}^{n+1}$ as input, D outputs 1 if and only if $\mathcal{A}(Y[1..n]) = Y[n+1]$.

Let b denote a random bit. Then, for infinitely many n's,

$$\begin{aligned}
\mathbb{P}[D(g(U_n)) = 1] - \mathbb{P}[D(U_{n+1}) = 1] &= \mathbb{P}[\mathcal{A}(f(U_n)) = b(U_n)] - \mathbb{P}[b = \mathcal{A}(U_n)] \\
&\geq 1/2 + 1/p(n) - 1/2 \\
&= 1/p(n).
\end{aligned}$$

■

Now, we are able to show:

Proposition 5.4.5. *If $g : \{0,1\}^n \to \{0,1\}^{n+1}$ is a super-bit and define $f(x)b(x) := g(x)$, where $b(x)$ is the last bit of $g(x)$, then b is a super-core of $f \in P/poly$.*

Proof. Suppose, for a contradiction, b is not a super-core of f. We present a proof for the case, in which there exist an $\mathcal{A} \in NP/poly$, a polynomial p, and infinitely many n's such that $①(\mathcal{A}) + ③(\mathcal{A}) \geq 1/2 + 1/p(n)$, and the proof for the other case is similar.

We define a distinguisher D to break g as follow:

> Given $Y \in \{0,1\}^{n+1}$ as input, D runs $\mathcal{A}$ on $Y[1...n]$ to get one output bit c of $\mathcal{A}$. D then outputs 1 if and only if $Y[n+1] = 0 \wedge c = 1$.

Then, infinitely many n's, we have

$$\begin{aligned}
&\mathbb{P}[D(U_{n+1}) = 1] - \mathbb{P}[D(f(U_n)b(U_n)) = 1] \\
=\ &\mathbb{P}[\mathcal{A}(U_n) = 1 \wedge U_1 = 0] - \mathbb{P}[\mathcal{A}(f(U_n)) = 1 \wedge b(U_n) = 0] \\
=\ &1/2 \cdot \mathbb{P}[\mathcal{A}(U_n) = 1] - (\mathbb{P}[b(U_n) = 0] - \mathbb{P}[\mathcal{A}(f(U_n)) = 0 \wedge b(U_n) = 0]) \\
=\ &①(\mathcal{A}) + ③(\mathcal{A}) - \mathbb{P}[b(U_n) = 0] \\
\geq\ &1/2 + 1/p(n) - \mathbb{P}[b(U_n) = 0]
\end{aligned}$$

Let $1/s(n) = |\mathbb{P}[b(U_n) = 0] - 1/2|$ As g is a strong PRG, b is a hard-core of g by Lemma 5.4.4. Thus, for every polynomial q (in particular, for $q(n) = 2p(n)$) and every sufficiently large n, $1/s(n) \leq 1/q(n)$. Therefore, for infinitely many n's,

$$\mathbb{P}[D(U_{n+1}) = 1] - \mathbb{P}[D(f(U_n)b(U_n)) = 1] \geq 1/2 + 1/p(n) - \mathbb{P}[b(U_n) = 0] \geq 1/2p(n).$$

■

By combining Proposition 5.4.3 and Proposition 5.4.5, we establish, as promised, the following nondeterministic variant of Theorem 5.3.6.

theorem 5.4.6. *There is a super-core b of some length-preserving $f \in P/poly$ if and only if there is a super-bit g.*

We say a function $f : \{0,1\}^n \to \{0,1\}^{m(n)}$ is **non-shrinking** if $m(n) \geq n$ for every n. Because (1) b is a super-core of some length-preserving function if and only if b is a super-core of some non-shrinking function and (2) there exists a super-bits if and only if there exist super-bits, we can alternatively state Theorem 5.4.6 as:

theorem 5.4.7. *There is a super-core b of some non-shrinking $f \in P/poly$ if and only if there are super-bits g.*

Although we can loose the condition "length-preserving" to "non-shrinking", the requirement "non-shrinking" is not redundant because there is an easy construction of a super-core b of some function f which shrinks its input, but whether a super-bit exists is unknown. Define $f(x) = x[1]$ and $b(x) = x[-1]$, then b is a super-core of f as there are only four functions from $\{0,1\}$ to $\{0,1\}$, and none of them can predict b given $f(x) \in \{0,1\}$ in the required sense (indeed, $①(c) + ③(c) = ②(c) + ④(c) = 1/2$ for any $c : \{0,1\} \to \{0,1\}$).

At this point, we may want to understand more about the relation between being one-way and having a super-core for a function f. Let's recall what we know in the deterministic setting: (1) if f has a hard-core b, f is not necessarily one-way because the possession of a hard-core can be due to an information loss of f, and (2) if f is one-way, then we can construct a hard-core b of $f'(x,y) = (f(x), y)$.

The first point is the same in the nondeterministic setting. The aforementioned $f(x) = x[1]$ is clearly not one-way but has a super-core $b(x) = x[-1]$, which is due to a dramatic loss of information. A more interesting question is that whether the f constructed in Proposition 5.4.5 from a super-bit g, which we know possesses a super-core $b(x) = g(x)[-1]$.

Open Problem 5.4.8. *If g is a super-bit, if $f(x) := g(x)[1...n]$ $(n = |x|)$ is a one-way function?*

As for the second point, however, since being a super-core is stronger requirement than being a hard-core, it is not necessarily true that there is a "universal" super-core in the sense that some predicate b (e.g., $b(x, y) = \langle x, y\rangle$) is a super-core for every $f'(x, y) = (f(x), y)$, where $f : \{0, 1\}^n \longrightarrow \{0, 1\}^{m(n)}$ is a one-way function. Then, a natural question to ask is:

Open Problem 5.4.9. *Suppose f is a one-way function. If we want $\langle x, y\rangle$ to be a super-core of $f'(x, y) = (f(x), y)$, what other properties does f need to have if any?*

Rather, as for the second point, we can show that it is impossible for certain f's to have a super-core even if they are possibly one-way. Such f's include length-preserving functions which are "predominantly 1-1" infinitely often. To state this more precisely, we say x is of ***type 1*** if $|f^{-1}(f(x))| = 1$ and x is of ***type 2*** otherwise. We define $T_1(n)$ to be the set of $x \in \{0, 1\}^n$ of *type 1* and $T_2(n)$ to be the set of $x \in \{0, 1\}^n$ of *type 2* . Now, we establish:

theorem 5.4.10. *Given $f : \{0, 1\}^n \longrightarrow \{0, 1\}^n$ in $P/poly$, if there exists infinitely many n's and a polynomial p such that,*

$$\frac{|T_1|}{2^n} \geq \frac{2}{3} + \frac{1}{p(n)},$$

then f does not have a super-core.

Proof. Assume f has the stated property. Suppose, for a contradiction, b is a super-bit of f. We construct as follows two algorithms $\mathcal{A}_j (j = 0, 1)$, in which $\mathcal{A}_0$ is co-nondeterministic and $\mathcal{A}_1$ is nondeterministic, to predict b:

> Given $y \in \{0, 1\}^n$ as input, $\mathcal{A}_j$ guesses x' such that $f(x') = y$. If $\mathcal{A}_j$ fails to guess such an x', it outputs $1 - j$. Otherwise, it outputs $b(x')$.

We note that if $y = f(x)$ for some $x \in T_1$, then there is always exactly one correct guess x' (i.e., x itself) for $\mathcal{A}_j$ such that $f(x') = y = f(x)$, and whenever such an x' is guessed, $b(x') = b(x)$. Hence, ①$(\mathcal{A}_1) = \mathbb{P}[\mathcal{A}_1(f(x)) = b(x) = 0] \geq \mathbb{P}[\mathcal{A}_1(f(x)) = b(x) = 0 \cap x \in T_1] = \mathbb{P}[b(x) = 0 \cap x \in T_1]$, and ③$(\mathcal{A}_1) = \frac{1}{2}\mathbb{P}[\mathcal{A}_1(y) = 1] \geq \frac{1}{2}\mathbb{P}[y = f(x) \cap b(x) = 1 \text{ for some } x \in T_1]$. Similarly, ②$(\mathcal{A}_0) \geq \mathbb{P}[b(x) = 1 \cap x \in T_1]$, and ④$(\mathcal{A}_0) \geq \frac{1}{2}\mathbb{P}[y = f(x) \cap b(x) = 0 \text{ for some } x \in T_1]$.

Therefore, ①$(\mathcal{A}_1)$ + ②$(\mathcal{A}_0)$ + ③$(\mathcal{A}_1)$ + ④$(\mathcal{A}_0)$ $= \mathbb{P}[x \in T_1] + \frac{1}{2}\mathbb{P}[y = f(x) \text{ for some } x \in T_1] = \mathbb{P}[x \in T_1] + \frac{1}{2}\mathbb{P}[x \in T_1] = \frac{3}{2}\mathbb{P}[x \in T_1] \geq 1 + \frac{1}{p(n)}$ for some polynomial p and infinitely many n's, but this implies that either ①$(\mathcal{A}_1)$ + ③$(\mathcal{A}_1) \geq 1/2 + 1/2p(n)$ or ②$(\mathcal{A}_0)$ + ④$(\mathcal{A}_0) \geq 1/2 + 1/2p(n)$. ■

The intuition reason that such f's do not have a super-core is: such an f preserves most of the information (thus, a unique pre-image can be guessed), and not many non-range elements are there to be identified. This result can also be proved alternatively by making using of other aforementioned results as follows. Suppose, for a contradiction, b is a super-core of f, then $g(x) := f(x)b(x)$ is a super-bit by Proposition 5.4.3. However, we can then easily break g as follows: we witness the randomness of a given $y \in \{0, 1\}^{n+1}$ by guessing an $x \in \{0, 1\}^n$ such that $f(x) = y[1...n]$ and test if $b(x) = y[n + 1]$; if $b(x) \neq y[n + 1]$, y is random.

We state a weaker but more concise corollary of Theorem 5.4.10:

Corollary 5.4.11. *If $f \in P/poly$ is length-preserving and 1-1 (or at least 1-1 for infinitely many n's), then f does not have a super-core.*

This corollary contrasts the result that a length-preserving and 1-1 $f \in P/poly$ could have a hard-core (in fact, if b is a hard-core of a length-preserving and 1-1 $f \in P/poly$, then $g(x) := f(x)b(x)$ is a strong PRG [15]). Thus, the corollary suggests there might be strong PRGs which are not super-bits.

Besides all the above investigation on which functions can or cannot have a super-core and the relation between being a one-way function and having a super-core, another central question is to ask is:

Open Problem 5.4.12. *What is a sensible definition of the "nondeterministic one-way functions" if any?*

Satisfactory answers to Open Problem 5.4.8 and Open Problem 5.4.9 will shed light on this question. A possible candidate is "a one-way function which has a super-core", but this unsettled question clearly needs further exploration.

6 FUTURE DIRECTIONS

In the final chapter, we propose some future directions and open problems that are worth pursuing.

Demi-bits stretching.

We have provided an algorithm that achieves a sublinear-stretch for any given demi-bit. If we are greedier, we may wonder if we can stretch more. More specifically, if we can stretch one demi-bit to linearly many demi-bits, or to polynomially many demi-bits, or even to a pseudorandom function generator (PRFG) with exponential demi-hardness. Even if assuming we do have some n-demi-bits $b : \{0,1\}^n \rightarrow \{0,1\}^{2n}$, it is still unclear whether we can construct an PRFG with exponential demi-hardness by any standard algorithm for constructing PRFGs or a novel one. If we can construct an exponentially demi-hard PRFG from a single demi-bit, then we can prove: the existence of a demi-bit rules out the existence of $N\tilde{P}/qpoly$-natural properties against $P/poly$, which is an improvement of Theorem 2.4.4 [33].

On the other hand, stretching algorithms of demi-bits have intriguing applications, which we are working on, to proof complexity generators and average-case complexity.

"Super" one-way functions.

One of the open problem we proposed in Chapter 5 is: what is a sensible definition of super one-way functions? A candidate definition is "a one-way function which has a super-core", but the proposal demands further verification. Together with this question, We rehearse all the open problems proposed in Chapter 5 here:

- What is a sensible definition of super one-way functions?
- If g is a super-bit, if $f(x) := g(x)[1...n]$ $(n = |x|)$ is a one-way function?
- Suppose f is a one-way function. If we want $\langle x, y\rangle$ to be a super-core of $f'(x,y) = (f(x), y)$, what other properties does f need to have if any?
- Can we further refine or classify the inequalities in **Summary** at the end of section 5.2? Can we also characterise demi-hardness from an unpredictability point of view?

Better answers to the questions list above would help us to better understand the hardness of PRGs in the nondeterministic setting and shed light on the open problem "whether the existence of demi-bits implies the existence of super-bits".

REFERENCES

[1] M. Ajtai, J. Komlos, and E. Szemeredi. Deterministic simulation in logspace. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, STOC '87, page 132–140, New York, NY, USA, 1987. Association for Computing Machinery. doi:10.1145/28395.28410.

[2] W. Alexi, B. Chor, O. Goldreich, and C. P. Schnorr. Rsa and rabin functions: Certain parts are as hard as the whole. *SIAM Journal on Computing*, 17(2):194–209, 1988. arXiv:https://doi.org/10.1137/0217013, doi:10.1137/0217013.

[3] N. Alon, L. Babai, and A. Itai. A fast and simple randomized parallel algorithm for the maximal independent set problem. *Journal of Algorithms*, 7(4):567–583, 1986. URL: https://www.sciencedirect.com/science/article/pii/0196677486900192, doi:https://doi.org/10.1016/0196-6774(86)90019-2.

[4] N. Alon, O. Goldreich, J. Håstad, and R. Peralta. Simple construction of almost k-wise independent random variables. In *Proceedings [1990] 31st Annual Symposium on Foundations of Computer Science*, pages 544–553 vol.2, 1990. doi:10.1109/FSCS.1990.89575.

[5] L. Babai, N. Nisan, and M. Szegedy. Multiparty protocols and logspace-hard pseudorandom sequences. In *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing*, STOC '89, page 1–11, New York, NY, USA, 1989. Association for Computing Machinery. doi:10.1145/73007.73008.

[6] T. Baker, J. Gill, and R. Solovay. Relativizations of the p =? np question. *SIAM Journal on Computing*, 4(4):431–442, 1975. arXiv:https://doi.org/10.1137/0204037, doi:10.1137/0204037.

[7] D. A. Barrington. *A Note on a Theorem of Razborov*. University of Massachusetts at Amherst. Computer and Information Science [COINS], 1987. URL: https://books.google.co.uk/books?id=2JAjrgEACAAJ.

[8] A. Blum, M. Furst, M. Kearns, and R. J. Lipton. Cryptographic primitives based on hard learning problems. In Douglas R. Stinson, editor, *Advances in Cryptology — CRYPTO' 93*, pages 278–291, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.

[9] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM J. Comput.*, 13(4):850–864, nov 1984. doi:10.1137/0213053.

[10] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth. Occam's razor. *Inf. Process. Lett.*, 24(6):377–380, apr 1987. doi:10.1016/0020-0190(87)90114-1.

[11] B. Chor and O. Goldreich. On the power of two-point based sampling. *Journal of Complexity*, 5(1):96–106, 1989. URL: https://www.sciencedirect.com/science/article/pii/0885064X89900150, doi:https://doi.org/10.1016/0885-064X(89)90015-0.

[12] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976. doi:10.1109/TIT.1976.1055638.

[13] M. L. Furst, J. B. Saxe, and M. Sipser. Parity, circuits and the polynomial time hierarchy. *Mathematical Systems Theory*, 17:13–27, 1984. doi:10.1007/BF01744431.

[14] O. Goldreich. *Foundations of Cryptography*, volume 1. Cambridge University Press, 2001. doi:10.1017/CBO9780511546891.

[15] O. Goldreich. *Computational Complexity: A Conceptual Perspective*. Cambridge University Press, 2008. doi:10.1017/CBO9780511804106.

[16] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *J. ACM*, 33(4):792–807, aug 1986. doi:10.1145/6490.6503.

[17] O. Goldreich and L. A. Levin. A hard-core predicate for all one-way functions. In *21st Annual ACM Symposium on Theory of Computing*, STOC '89, page 25–32, New York, NY, USA, 1989. Association for Computing Machinery. doi:10.1145/73007.73010.

[18] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984. URL: https://www.sciencedirect.com/science/article/pii/0022000084900709, doi:https://doi.org/10.1016/0022-0000(84)90070-9.

[19] J. Håstad. *Computational Limitations for Small-Depth Circuits*. Mit Press, 1987.

[20] J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999. arXiv:https://doi.org/10.1137/S0097539793244708, doi:10.1137/S0097539793244708.

[21] R. Impagliazzo and M. Naor. Efficient cryptographic schemes provably as secure as subset sum. In *30th Annual Symposium on Foundations of Computer Science*, pages 236–241, 1989. doi:10.1109/SFCS.1989.63484.

[22] B. S. Kaliski. Elliptic curves and cryptography : a pseudorandom bit generator and other tools. *Phd Thesis Mit*, 2005.

[23] M. J. Kearns and U. V. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, Cambridge, MA, USA, 1994.

[24] M. Luby. *Pseudorandomness and Cryptographic Applications*, volume 1. Princeton University Press, 1996. URL: http://www.jstor.org/stable/j.ctvs32rpn.

[25] J. Naor and M. Naor. Small-bias probability spaces: Efficient constructions and applications. In *Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing*, STOC '90, page 213–223, New York, NY, USA, 1990. Association for Computing Machinery. doi:10.1145/100216.100244.

[26] N. Nisan and A. Wigderson. Hardness vs randomness. *Journal of Computer and System Sciences*, 49(2):149–167, 1994. URL: https://www.sciencedirect.com/science/article/pii/S0022000005800431, doi:https://doi.org/10.1016/S0022-0000(05)80043-1.

[27] I. C. Oliveira and R. Santhanam. Conspiracies between learning algorithms, circuit lower bounds, and pseudorandomness. In *Proceedings of the 32nd Computational Complexity Conference*, CCC '17, Dagstuhl, DEU, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[28] J. Pich. Learning algorithms from circuit lower bounds. *CoRR*, abs/2012.14095, 2020. URL: https://arxiv.org/abs/2012.14095, arXiv:2012.14095.

[29] N. Rajgopal and R. Santhanam. On the Structure of Learnability Beyond P/Poly. In Mary Wootters and Laura Sanità, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021)*, volume 207 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 46:1–46:23, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: https://drops.dagstuhl.de/opus/volltexte/2021/14739, doi:10.4230/LIPIcs.APPROX/RANDOM.2021.46.

[30] A. A. Razborov. Lower bounds on the size of bounded depth circuits over a complete basis with logical addition. *Mathematical notes of the Academy of Sciences of the USSR*, 41:333–338, 1987.

[31] A. A. Razborov and S. Rudich. Natural proofs. *Journal of Computer and System Sciences*, 55(1):24–35, 1997. URL: https://www.sciencedirect.com/science/article/pii/S002200009791494X, doi:https://doi.org/10.1006/jcss.1997.1494.

[32] M. Rosulek. *The Joy of Cryptography*. Oregon State University, 2021. https://joyofcryptography.com. URL: https://joyofcryptography.com.

[33] S. Rudich. Super-bits, demi-bits, and np/qpoly-natural proofs. *Journal of Computer and System Sciences*, 55:204–213, 1997.

[34] A. Shamir. On the generation of cryptographically strong pseudo-random sequences. In Shimon Even and Oded Kariv, editors, *Automata, Languages and Programming*, pages 544–550, Berlin, Heidelberg, 1981. Springer Berlin Heidelberg.

[35] R. 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*, STOC '87, page 77–82, New York, NY, USA, 1987. Association for Computing Machinery. doi:10.1145/28395.28404.

[36] L. G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, nov 1984. doi:10.1145/1968.1972.

[37] L. G. Valiant. Learning disjunction of conjunctions. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'85, page 560–566, San Francisco, CA, USA, 1985. Morgan Kaufmann Publishers Inc.

[38] L. G. Valiant and J. C. Shepherdson. Deductive learning [and discussion]. *Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 312(1522):441–446, 1984. URL: http://www.jstor.org/stable/37444.

[39] A. C. Yao. Theory and application of trapdoor functions. In *23rd Annual Symposium on Foundations of Computer Science*, FOCS '82, pages 80–91, 1982. doi:10.1109/SFCS.1982.45.

[40] A. C. Yao. Separating the polynomial-time hierarchy by oracles. In *26th Annual Symposium on Foundations of Computer Science*, FOCS '85, pages 1–10, 1985. doi:10.1109/SFCS.1985.49.