Electronic Colloquium on Computational Complexity, Report No. 57 (2023)

Stretching Demi-Bits and Nondeterministic-Secure Pseudorandomness

Iddo Tzameret
Imperial College London

Lu-Ming Zhang
London School of Economics and Political Science

Abstract

We develop the theory of cryptographic nondeterministic-secure pseudorandomness beyond the point reached by Rudich's original work [Rud97], and apply it to draw new consequences in average-case complexity and proof complexity. Specifically, we show the following:

Demi-bit stretch: Super-bits and demi-bits are variants of cryptographic pseudorandom generators which are secure against nondeterministic statistical tests [Rud97]. They were introduced to rule out certain approaches to proving strong complexity lower bounds beyond the limitations set out by the Natural Proofs barrier of Rudich and Razborov [RR97]. Whether demi-bits are stretchable at all had been an open problem since their introduction. We answer this question affirmatively by showing that: every demi-bit $b : \{0,1\}^n \to \{0,1\}^{n+1}$ can be stretched into sublinear many demi-bits $b' : \{0,1\}^n \to \{0,1\}^{n+n^c}$, for every constant $0 < c < 1$.

Average-case hardness: Using work by Santhanam [San20], we apply our results to obtain new average-case Kolmogorov complexity results: we show that $\mathrm{K}^{\mathsf{poly}}[n - O(1)]$ is zero-error average-case hard against $\mathsf{NP}/\mathsf{poly}$ machines iff $\mathrm{K}^{\mathsf{poly}}[n - o(n)]$ is, where for a function $s(n) : \mathbb{N} \to \mathbb{N}$, $\mathrm{K}^{\mathsf{poly}}[s(n)]$ denotes the languages of all strings $x \in \{0,1\}^n$ for which there are (fixed) poly-time Turing machines of description-length at most $s(n)$ that output x.

Characterising super-bits by nondeterministic unpredictability: In the deterministic setting, Yao [Yao82] proved that super-polynomial hardness of pseudorandom generators is equivalent to ("next-bit") *unpredictability*. Unpredictability roughly means that 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 (in the cryptographic regime), and characterise the nondeterministic hardness of generators from an unpredictability perspective. Specifically, we propose four stronger notions of unpredictability: $\mathsf{NP}/\mathsf{poly}$-unpredictability, $\mathsf{coNP}/\mathsf{poly}$-unpredictability, $\cap$-unpredictability and $\cup$-unpredictability, and show that super-polynomial nondeterministic hardness of generators lies between $\cap$-unpredictability and $\cup$-unpredictability.

Characterising super-bits by nondeterministic hard-core predicates: We introduce a nondeterministic variant of hard-core predicates, called *super-core predicates*. We show that the existence of a super-bit is equivalent to the existence of a super-core of some non-shrinking

Department of Computing. Part of this work was done at Simons Institute for the Theory of Computing, UC Berkeley. Part of this project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 101002742). Email: iddo.tzameret@gmail.com

Part of this work was done while at Imperial College London. Email: l534zhang@uwaterloo.ca

ISSN 1433-8092

function. This serves as an analogue of the equivalence between the existence of a strong pseudorandom generator and the existence of a hard-core of some one-way function [GL89, HILL99], and provides a first alternative characterisation of super-bits. We also prove that a certain class of functions, which may have hard-cores, cannot possess any super-core.

Contents

1 Introduction

Pseudorandomness is a natural concept allowing to measure the extent to which a resource-bounded computational machine can identify true random sources. It is an important notion in algorithms, enabling to derandomize efficient probabilistic algorithms by simulating many true random bits using fewer random bits. Another important aspect of pseudorandomness lies in computational complexity and cryptography, and specifically computational lower bounds, where it serves as the foundation of many results in cryptography, hardness vs. randomness trade-offs, and several barriers to proving strong computational lower bounds. Here, we will mostly be interested in the latter aspect of barrier results.

1.1 The theory of nondeterministic-secure pseudorandomness

Razborov and Rudich established a connection between pseudorandomness and the ability to prove boolean circuit lower bounds in their Natural Proofs paper [RR97]. They showed that most lower bounds arguments in circuit complexity contain (possibly implicitly) an efficient algorithm for deciding hardness, in the following sense: given the truth table of a boolean function, the algorithm determines if the function possesses some (combinatorial) properties that imply it is hard for a certain given circuit class (they called this "constructivity" and "usefulness" of a lower bound argument). They moreover showed that this algorithm identifies correctly the hardness of a non-negligible fraction of functions (which is called "largeness" in [RR97]). On the other hand, such an efficient algorithm for determining the hardness of boolean functions (for general circuits) would contradict reasonable assumptions in pseudorandomness, namely the existence of strong pseudorandom generators. This puts a *barrier*, so to speak, against the ability to prove lower bounds using natural proofs.

The notion of natural proofs has had a great influence on computational complexity theory. However, the fact that the plausible nonexistence of certain classes of natural proofs provides an obstacle against *very constructive* lower bound arguments (namely, those arguments that implicitly contain an efficient algorithm to determine when a function is hard) is somewhat less desirable. One would hope to extend the obstacle to less constructive proofs, for instance, proofs whose arguments contain implicitly only short *witnesses* for the hardness of functions.

Indeed, the notion of natural proofs comes to explain the difficulty in *proving* lower bounds, not in efficiently *deciding* hardness of boolean functions. For these reasons, among others, Rudich [Rud97] set to extend the natural proofs barrier so that they encompass non-constructive arguments, namely, arguments implicitly using efficient *witnesses* of the hardness of boolean functions, in contrast to deterministic algorithms. This was done by extending the notion of pseudorandom generators so that they are secure against *nondeterministic* adversaries.

While empirically most known lower bound proofs were shown, at least implicitly, to fall within the scope of $\mathsf{P/poly}$-constructive natural proofs, it is definitely conceivable and natural to assume that some lower bound approaches will necessitate $\mathsf{NP/poly}$-constructive natural proofs. In fact, it is a very interesting open problem in itself to find such an $\mathsf{NP/poly}$-constructive lower bound proof method. Furthermore, in works in proof complexity the role of nondeterministic-secure pseudorandomness is important (cf. recent work by Pich and Santhanam [PS19], as well as Krajíček [Kra04]). Also, note that when dealing with the notion of barriers we refer to impossibly results and so it cannot always be expected to come up with good examples of proof methods we wish to rule out.

Accordingly, to extend the natural proofs barrier, Rudich introduced two primitives: *super-bits,* and its weaker variant *demi-bits*. Super-bits and demi-bits are (non-uniform) nondeterministic variants of strong pseudorandom generators (PRGs). They are secure against nondeterministic adversaries (i.e., adversaries in $\mathsf{NP/poly}$). Demi-bits require their nondeterministic adversaries to break them in a *stronger* sense than super-bits, and hence their existence constitutes a better (i.e., *weaker*) assumption than the existence of super-bits on which to base barrier results; and this is one reason why the concept of demi-bits is important.

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 an output of a truly random process and 0 if it is the output of a pseudorandom generator). Thus, a nondeterministic distinguisher cannot break super-bits nor demi-bits by simply guessing a seed of the generator. Precisely, this is guaranteed as follows: for demi-bits we insist that strings in the image of the pseudorandom generator are always *rejected*, while for super-bits we allow some such pseudorandom strings to be accepted but we insist that many more strings outside the image of the pseudorandom generator are accepted (than strings in the image of the pseudorandom generator).

Formally, we have the following (all the distributions we consider in this work are, by default, uniform, unless otherwise stated, and U_n denotes the uniform distribution over $\{0,1\}^n$):

Definition 1.1 (Nondeterministic hardness [Rud97])**.** *Let $g_n : \{0,1\}^n \to \{0,1\}^{l(n)}$, with $l(n) > n$, be a function in* $\mathsf{P/poly}$. *We call such a function a* ***generator****. The* ***nondeterministic hardness*** *$H_{\mathrm{nh}}(g_n)$ (also called* ***super-hardness****) of 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 standard definition of (deterministic) hardness (Definition 3.6), the order of the two possibilities on the left-hand side is crucial. This order forces a nondeterministic distinguisher to certify the randomness of a given input. Reversing the order, or adding an absolute value to left-hand side, trivializes (as in standard PRGs) 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 $\mathbb{P}[D(y)=1] \leq 1/2$.

Super-bits are exponentially super-hard generators:

Definition 1.2 (Super-bits [Rud97])**.** *A generator $g : \{0,1\}^n \to \{0,1\}^{n+c}$ (computable in* $\mathsf{P/poly}$*), 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 constant $\varepsilon > 0$ and all sufficiently large n's. In particular, if $c = 1$, we call g a super-bit.*

Many candidates of strong PRGs (against deterministic machines) were constructed by exploiting functions conjectured to be one-way and/or their hard-cores (see for example [ACGS88, Kal05, IN89]). The work [ACGS88] presented PRGs based on the assumption that factoring is hard, while [Kal05] presented PRGs based on the conjectured hardness of the discrete-logarithm problem. [IN89] presented PRGs based on the subset sum problem. Similarly, for nondeterministic-secure PRGs (namely, super-bits), Rudich conjectured the existence of a super-bit generator based on the hardness of the subset sum problem [IN89]. We are unaware of any additional conjectured construction of super-bits.

As mentioned above, another hardness measure of generators is introduced by Rudich:

Definition 1.3 (Demi-hardness [Rud97])**.** *Let $g_n : \{0,1\}^n \to \{0,1\}^{l(n)}$ be a generator (computable in* P/poly*). Then the* ***demi-hardness*** *$H_{\mathrm{dh}}(g_n)$ of g_n is the minimal s for which there exists a nondeterministic circuit D of size at most s such that*

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

We note that (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.

Demi-bits are exponentially demi-hard generators, where "demi" here stands for "half":

Definition 1.4 (Demi-bits [Rud97])**.** *$g : \{0,1\}^n \to \{0,1\}^{n+c}$ 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.*

It is worth mentioning that demi-bits g as in Definition 1.4 can be viewed as a *hitting set generator* against NP/poly (see below Section 2.1.2 and Santhanam [San20]).

The difference between super-bits and demi-bits is that demi-bits require their distinguishers to break them in a stronger sense: a demi-bit distinguisher must always be correct on the pseudorandom strings (i.e., always output 0 for strings in the image of the generator). Thus, if g is a super-bit(s) (the plural here denotes that g may have a stretching-length greater than 1; if the stretching length is exactly 1, we say g is *a* super-bit), no algorithm in NP/poly can break g in the weaker sense (1), and hence no algorithm in NP/poly can break g in the stronger sense (2), which means g is also a demi-bit(s). In other words, the existence of super-bits implies the existence of demi-bits (although it is open if any of these two exists).

Remark (Cryptographic vs. complexity-theoretic regime)*.* In this work, we are interested only in the *cryptographic* regime of pseudorandomness. In this regime, the adversary whom the generator tries to fool is allowed to be stronger than the generator and specifically has sufficient computational resources to run the generator. In the *complexity-theoretic regime*, in which the adversary cannot simulate the generator, the notion of nondeterministic secure pseudorandomness was developed in works by, e.g., Klivans and van Melkebeek [KvM02] as well as Shaltiel and Umans [SU05] (see also the recent work by Sdroievski and van Melkebeek [SvM23] and references therein). These complexity-theoretic ideas have also found several applications in cryptography (originating from the work of Barak, Ong and Vadhan [BOV07]). It is also worth mentioning that in the complexity-theoretic regime, one can use the original definition of the hardness of PRGs (Definition 3.6) even against nondeterministic adversaries; while this is not the case in the cryptographic regime, in which a PRG as in Definition 3.6 can never be safe against a nondeterministic adversary who guesses the seed.

1.2 Relations to barrier results

Recall that natural proofs [RR97] for proving circuit lower bounds are proofs that 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 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 if a subset $C' \subseteq C$ satisfies *constructivity*, that is, it is Γ-decidable whether f is in C', and *largeness*, that is, C' constitutes a non-negligible portion of F_n. We say C is *useful* against Λ

if every function family f that 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) is not in $\mathsf{P/poly}$ (or in general, some other complexity class Λ), we identify some natural combinatorial property C of f and show all function families that have property C are not in $\mathsf{P/poly}$ (i.e., the property is useful against $\mathsf{P/poly}$). If f is NP-complete (e.g., SAT), then such a proof concludes $\mathsf{P} \neq \mathsf{NP}$.

Razborov and Rudich argued that, based on the existence of strong PRGs, no $\mathsf{P/poly}$-natural proofs can be useful against $\mathsf{P/poly}$. They showed that many known proofs of lower bounds against (non-monotone) boolean circuits are natural or can be presented as natural in some way.

In this context, the theory of nondeterministic-secure generators allows one to rule out a larger (arguably more natural) class of lower bound arguments:

Theorem 1.5 ([Rud97])**.** *If super-bits exist, then there are no $N\tilde{P}/qpoly$-natural properties useful against* $\mathsf{P/poly}$*, where $N\tilde{P}/qpoly$ is the class of languages recognised by non-uniform, quasi-polynomial-size circuit families.*

This theorem is proved based on the ability to *stretch* super-bits, namely, taking a generator that maps n bits to $n+1$ bits, which we refer to as stretching length 1, to a generator that maps n bits to $n+N$ bits, with $N > 1$, which we call stretching length N. 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) [BM84, Yao82] and can be exploited to construct hard-to-break pseudorandom *function* generators (loosely speaking, generators that generate pseudorandom functions indistinguishable from truly random ones) [GGM86]. 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 secure against nondeterministic adversaries [Rud97]. The proofs of the correctness of such stretching algorithms are based on a technique called the *hybrid argument* [GM84] reviewed below. In contrast, whether a demi-bit can be stretched even to two demi-bits was unknown before the current work, since this cannot be concluded with a direct application of a standard hybrid argument.

2 Contributions, significance and context

We develop the foundations of nondeterministic-secure pseudorandomness. This is the first systematic investigation into nondeterministic pseudorandomness (in the cryptographic regime) we are aware of, building on the primitives proposed by Rudich [Rud97]. We provide new understanding of the primitives of the theory, namely, super and demi-bits, as well as introducing new notions and showing how they relate to established ones. We draw several conclusions from these results in average-case and proof complexity. We also achieve some modest progress on establishing sounder foundations for barrier results: by showing, for instance, that demi-bits can be (moderately) stretched, we provide some hope to strengthen the connection between demi-bits and unprovability results (as of now, it is only known that the existence of a super-bit yield barrier results, while we hope to show that the weaker assumption of the existence of demi-bits suffices for that matter).

2.1 Stretching demi-bits

In Demi-Bit Stretching Algorithm 4.1, we provide an algorithm that achieves a sublinear stretch for any given demi-bit. This solves the open problem of whether a demi-bit can be stretched to 2-bits [Rud97, Open Problem 2] (see also Santhanam [San20, Question 4]).

Theorem (Informal; Theorem 4.2)**.** *Every demi-bit $b : \{0,1\}^n \to \{0,1\}^{n+1}$ can be efficiently converted (stretched) into demi-bits $g : \{0,1\}^n \to \{0,1\}^{n+n^c}$, for every constant $0 < c < 1$.*

2.1.1 Discussion and significance of stretching demi-bits to barrier results

Stretching demi-bits can be viewed as a first step towards showing that the existence of a demi-bit rules out $N\tilde{P}/qpoly$-natural properties useful against $\mathsf{P/poly}$, as we explain below. Providing such a barrier for $N\tilde{P}/qpoly$-natural properties based on the existence of a demi-bit is important, since assuming the existence of a demi-bit is a weaker assumption than assuming the existence of a super-bit.

Why is stretching demi-bits a step towards showing that the existence of a demi-bit would rule out $N\tilde{P}/qpoly$-natural properties useful against $\mathsf{P/poly}$? The reason is that stretching is the first step in the argument to base barrier results on the existence of a super-bit, in the following sense: the existence of a super-bit implies barrier results because one can stretch super-bits to obtain pseudorandom function generators, from which one gets the barrier result as noted in Theorem 1.5 above (and the text that follows it). More precisely, stretching demi-bits is a first (and necessary) step towards Rudich's Open Problem 3, and this problem also implies Rudich's Open Problem 4:

Open problem (Rudich's Open Problem 3 [Rud97])**.** *Given a demi-bit, is it possible to build a pseudorandom function generator with exponential ($2^{n^{\varepsilon}}$) demi-hardness?*

Open problem (Rudich's Open Problem 4 [Rud97])**.** *Does the existence of demi-bits rule out $N\tilde{P}/qpoly$-natural properties useful against $\mathsf{P/poly}$?*

Moreover, the study of the stretchability of demi-bit(s) provides a perspective towards resolving Rudich's Open Problem 1 (a positive answer of which would also resolve positively Open Problem 4):

Open problem (Rudich's Open Problem 1 [Rud97])**.** *Does the existence of a demi-bit imply the existence of a super-bit?*

This is because the stretchability of super-bits is well understood, while previously we did not know anything about the stretchability of demi-bits. We expect that understanding better basic properties of demi-bits, such as stretchability, would shed light on the relation between the existence of demi-bits and the existence of super-bits (Open problem 1).

2.1.2 Applications in average-case complexity

Here we describe an application of Theorem 4.2 to the average-case hardness of time-bounded Kolmogorov complexity.

As observed by Santhanam [San20], a hitting set generator $g : \{0,1\}^n \to \{0,1\}^{n+1}$ exists iff there exists a demi-bit $b : \{0,1\}^n \to \{0,1\}^{n+1}$. To recall, a ***hitting set generator against a class of decision problems*** $\mathcal{C} \subseteq 2^{\{0,1\}^N}$ is a function $g : \{0,1\}^n \to \{0,1\}^N$, for $n < N$, such that the image of g *hits* (namely, intersects) every dense enough set A in $\mathcal{C}$ (that is, $|A| \geq \frac{2^N}{N^{O(1)}}$). And we have:

Proposition (Proposition 4.3; [San20])**.** *Let $n < N$. A hitting set generator $g : \{0,1\}^n \to \{0,1\}^N$ computable in the class $\mathcal{D}$ against $\mathsf{NP}/\mathsf{poly}$ exists iff there exists a demi-bit $b : \{0,1\}^n \to \{0,1\}^N$ computable in $\mathcal{D}$ (against $\mathsf{NP}/\mathsf{poly}$).*

Santhanam [San20, Proposition 3] established an equivalence between (succinct) hitting set generators and average-case hardness of MCSP, where MCSP stands for the *minimal circuit size problem*. However, as mentioned to us by Santhanam [San22], similar arguments can show an equivalence between hitting set generators (not-necessarily succinct ones) and poly-time bounded Kolmogorov complexity zero-error average-case hardness against $\mathsf{NP}/\mathsf{poly}$ machines, as we show in this work.

We define the ***t*-bounded Kolmogorov complexity of string** x, denoted $\mathrm{K}^t(x)$, to be the minimal length of a string D such that the universal Turing machine $U(D)$ (we fix some such universal machine) runs in time at most t and outputs x. See [ABK+06] for more details about time-bounded Kolmogorov complexity and Definition 9 there for the definition of time-bounded Kolmogorov complexity of strings (that definition actually produces the ith bit of the string x given an index i and D as inputs to U, but this does not change our result).

Definition 2.1 (the language $\mathrm{K}^t[s]$ and $\mathrm{K}^{\mathsf{poly}}[s(n)]$)**.** *For a time function $t(n) : \mathbb{N} \to \mathbb{N}$ and a size function $s(n) : \mathbb{N} \to \mathbb{N}$, such that $s(n) \leq n$, let $\mathrm{K}^{t(n)}[s(n)]$ be the language $\{x \in \{0,1\}^* \ : \ |X| = n \ \wedge \ \mathrm{K}^{t(n)}(x) \leq s(n)\}$. We define $\mathrm{K}^{\mathsf{poly}}[s(n)]$ to be the language $\bigcup_{c \in \mathbb{N}} \mathrm{K}^{n^c}[s(n)]$.*

We also need to define the concept of zero-error average-case hardness against the class $\mathsf{NP}/\mathsf{poly}$ (see Definition 4.4). Informally, for a language L to be zero-error average-case *easy* for $\mathsf{NP}/\mathsf{poly}$, there should be a nondeterministic polytime machine with advice such that given an input x the machine guesses a witness for $x \in L$ or a witness for $x \notin L$, and when the witness is found it answers accordingly; and moreover we assume that for a polynomial-small fraction of inputs there are such witnesses (for membership or non-membership in L). If a witness is not found the machine outputs "Don't-Know". (We also assume that there are no pairs of contradicting witnesses for both $x \in L$ and $x \notin L$.)

In Section 4.1 we show the following:

Theorem (equivalence for average-case time-bounded Kolmogorov Complexity; Theorem 4.5)**.** *$\mathrm{K}^{\mathsf{poly}}[n - O(1)]$ is zero-error average-case hard against $\mathsf{NP}/\mathsf{poly}$ machines iff $\mathrm{K}^{\mathsf{poly}}[n - o(n)]$ is zero-error average-case hard against $\mathsf{NP}/\mathsf{poly}$ machines.*

2.1.3 Applications in proof complexity

In proof complexity, Krajíček [Kra04, Kra10] and Alekhnovich, Ben-Sasson, Razborov and Wigderson [ABRW04] developed the theory of *proof complexity generators*. Given a P/poly mapping $g : \{0,1\}^n \to \{0,1\}^\ell$, with $n < \ell$, and a fixed vector $r \in \{0,1\}^\ell$, we denote by $\tau(g)_r$ the $\mathsf{poly}(\ell)$-size propositional formula that encodes naturally the statement $r \notin \mathrm{Im}(g)$, so that if r is not in the image of g then $\tau(g)_r$ is a propositional tautology. For r not in the image of g, the tautology $\tau(g)_r$ is called a *proof complexity generator*, and the hope is that for strong propositional proof systems one can establish (at least conditionally) that there are no $\mathsf{poly}(\ell)$-size proofs of $\tau(g)_r$, under the assumption that the mapping g is sufficiently pseudorandom (see also [Raz15]). Krajíček observed the connection between proof complexity generators and demi-bits (see [Kra04, Corollary 1.3] and the discussion that follows there.)

An immediate corollary of Theorem 4.2 is the following.

Corollary (Stretching proof complexity generators; Corollary 4.6)**.** *Let $b : \{0,1\}^n \to \{0,1\}^{n+1}$ be a demi-bit computable in $\mathsf{P/poly}$. Let $0 < c < 1$ be a constant and $\ell = n + n^c$. Then, there is a proof complexity generator $g : \{0,1\}^n \to \{0,1\}^\ell$ in $\mathsf{P/poly}$, such that for every propositional proof system, with probability at least $1 - \frac{1}{\ell^{\omega(1)}}$ over the choice of $r \in \{0,1\}^\ell$, there are no $\mathsf{poly}(n)$-size proofs of the tautology $\tau(g)_r$.*[1]

2.1.4 Technique overview

We prove the stretchability of demi-bits by a novel and more flexible use of the hybrid argument combined with other ideas.

The *hybrid argument* (a.k.a. the hybrid method, the hybrid technique, etc.) is a common proof technique originating from the work of Goldwasser and Micali [GM84]. It was named by Leonid Levin. (See Section 3.1 for notations used below.) When we have a generator $g : \{0,1\}^n \to \{0,1\}^{m(n)}$, 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, called hybrids, traversing from one extreme, U_m, to another, $g(U_n)$. A concrete example is $H_i := g(U_n)[1...i] \cdot U_{m-i}, 0 \leq i \leq m$ (where $\cdot$ here means concatenation). 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. See Lemma A.2 for a simple demonstration of the hybrid argument.

[1] The points r are taken uniformly from $\{0,1\}^\ell$, and with probability $1 - 1/2^{\ell-n}$ the formula $\tau(g)_r$ is a tautology, because for all $r \in \{0,1\}^\ell \setminus \mathrm{Im}(g)$ the formula $\tau(g)_r$ is a tautology. While in some works, proof complexity generators are supposed to be hard for *every* r outside the image of the generator g, in our formulation the hardness is only with high probability over the r's. It is unclear whether this makes any difference in the theory of proof complexity generators, since we are not aware of a case where the property that $\tau(g)_r$ is hard for every $r \notin \mathrm{Im}(g)$ is used (although all cases of provably hard proof complexity generators against weak proof systems we know of, are hard for every $r \notin \mathrm{Im}(g)$).

As mentioned above, a standard hybrid argument cannot be applied to prove a stretched generator g by some stretching algorithm, from a single demi-bit b, is still demi-bits. We now intuitively explain the reason for this. A usual proof goes like this: we assume, for a contradiction, g are not demi-bits. Then there are some distinguisher D of g and a function p such that $\mathbb{P}\left[D(U_m) = 1\right] - \mathbb{P}\left[D(g(U_n)) = 1\right] \geq 1/p(n)$, and in particular $\mathbb{P}\left[D(g(U_n)) = 1\right] = 0$ as D breaks demi-bits, and we hope to construct a new appropriate distinguisher C of b based on D. However, as we saw above, a standard hybrid argument only yields that $\mathbb{P}\left[D(H_i) = 1\right] - \mathbb{P}\left[D(H_{i+1}) = 1\right] \geq 1/p'(n)$ for some function p' and cannot deduce that $\mathbb{P}\left[D(H_{i+1}) = 1\right] = 0$. Hence, it is unclear how to continue this construction.

The argument for proving Theorem 4.2 proceeds by the contrapositive. We assume there is a distinguisher D which breaks demi-bits g (stretched from a single demi-bit b by Demi-Bit Stretching Algorithm 4.1) in the desired sense, and we want to construct a distinguisher C which breaks b. Rather than applying the hybrid argument directly to D, we apply the hybrid argument to a new distinguisher D' defined based on D: the new distinguisher D' can use nondeterminism to change the pseudorandom part of the hybrids and thus "amplifies" the probability of certificating randomness (intuitively, this can be viewed as changing the average-case analysis in the standard hybrid argument to a worst-case or existence analysis). By applying the hybrid argument to D', we are able to identify a non-empty class, denoted by S_2 in the proof, of random strings $y_{i+1} \dots y_m$, that are not random witnesses (in the sense that, for each $y_{i+1} \dots y_m$ in this class, there are no seeds $x_1, \dots, x_i$ such that $D(b(x_1) \dots b(x_i)\; y_{i+1} \dots y_m) = 1$). Thus, for $y_{i+1} \dots y_m$ in S_2, $y_i y_{i+1} \dots y_m$ can become a random witness (i.e., there are seeds $x_1, \dots, x_{i-1}$ such that $D(b(x_1) \dots b(x_{i-1})\; y_i \dots y_m) = 1$) only if y_i is truly random (i.e., not equal to $b(x)$ for some seed x). The hybrid argument also implies a "good" such $y_{i+1} \dots y_m$ in S_2, which can identify a sufficient portion of truly random y_i. We can thereby build a new distinguisher C to distinguish truly random strings from pseudorandom ones.

A key step that makes this proof work is that nondeterministically guessing seeds $x_1, \dots, x_{i-1}$ in $b(x_1) \dots b(x_{i-1}) z_i$ preserves the "randomness-structure" of $b(x_1) \dots b(x_{i-1}) z_i$, in the sense that: when $z_i = b(\cdot)$ is pseudorandom, the nondeterministic guess preserves the form $b(\cdot) \dots b(\cdot) b(\cdot)$ (i.e., i equal-length pseudorandom chunks); and when $z_i = y$ is truly random, it preserves the form $b(\cdot) \dots b(\cdot) y$ (i.e., $i - 1$ equal-length pseudorandom chunks followed by a truly random chunk y of the same length). For common stretching algorithms that produce exponentially many new bits (e.g., recursively applying a one-bit generator), it is unclear how to use nondeterminism in a way that respects the "randomness-structure" of a given string. Nevertheless, the new proof technique should hopefully inspire researchers to further explore the stretchability of demi-bits. On the other hand, the fact could also be that there is a specific demi-bit which cannot be stretched to exponentially many demi-bits by the standard stretching algorithms which are applied to super-bits and strong PRGs.

2.2 Fine-grained characterisation of nondeterministic security based on unpredictability

Yao [Yao82] defined PRGs as producing sequences that are computationally indistinguishable, by deterministic adversaries, from uniform sequences and proved that this definition of indistinguishability is equivalent to deterministic unpredictability, which was used in an earlier definition of PRGs suggested by Blum and Micali [BM84]. Loosely speaking, unpredictability means, given any strict prefix of a random string, it is infeasible to predict the next bit.

In section 5, we provide a more fine-grained picture of nondeterministic hardness (Definition 1.1), by introducing the concept of nondeterministic unpredictability. This allows us to establish new lower and upper bounds to nondeterministic hardness, in the sense that we sandwich nondeterministic hardness between two unpredictability properties.

Specifically, we propose four notions of unpredictability for probability ensembles:

1. $\mathsf{NP/poly}$-unpredictability: the capacity of being unpredictable by $\mathsf{NP/poly}$ predictors.
2. $\mathsf{coNP/poly}$-unpredictability: the capacity of being unpredictable by $\mathsf{coNP/poly}$ predictors.
3. $\cup$-unpredictability: the capacity of being unpredictable by predictors in the union of $\mathsf{NP/poly}$ and $\mathsf{coNP/poly}$.
4. $\cap$-unpredictability: the capacity of being unpredictable by nondeterministic function-computing predictors.

The names $\mathsf{NP/poly}$-unpredictability, $\mathsf{coNP/poly}$-unpredictability, and $\cup$-unpredictability (a shorthand for $\mathsf{NP/poly} \cup \mathsf{coNP/poly}$-unpredictability) are self-explanatory, while the use of "$\cap$-unpredictability" is somewhat less intuitive. We will see, in Section 5.2 (where we use nondeterministic function-computing machines[2]), that a decision problem is in $\mathsf{NP/poly} \cap \mathsf{coNP/poly}$ if and only if it is decidable by a nondeterministic polynomial-size function-computing algorithm.

We establish the following characterisation of the nondeterministic hardness of generators from an unpredictability perspective:

Theorem (Summary 5.9)**.** *Here, $A \leq B$ means that if a generator has property B, then it also has property A:*

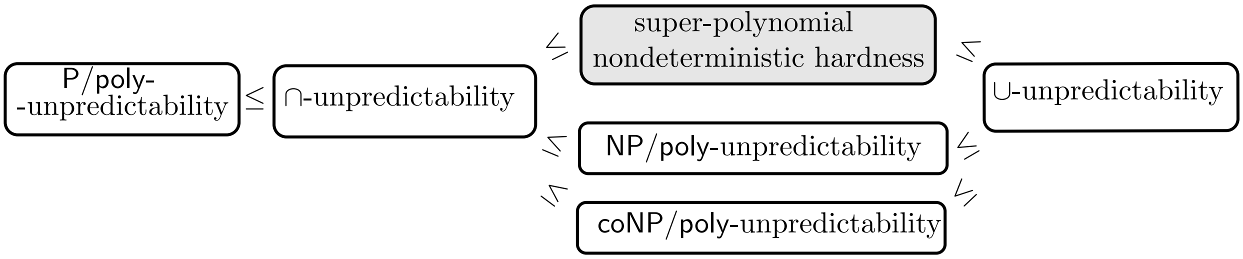

Figure 1: Super-polynomial nondeterministic hardness here refers to Definition 1.1. Note that $\cup$-unpredictability is at least as strong as $\mathsf{NP/poly}$-unpredictability, because it rules out predictors in *both* $\mathsf{NP/poly}$ and $\mathsf{coNP/poly}$. And similarly, $\cup$-unpredictability is at least as strong as $\mathsf{coNP/poly}$-unpredictability.

2.3 Super-cores: hard-core predicates in the nondeterministic setting

In the deterministic context, the existence of strong PRGs is known to be equivalent to the existence of central cryptographic primitives such as one-way functions, secure private-key encryption

[2] We say a nondeterministic algorithm $\mathcal{A}$ is a **function-computing** algorithm, if for every input $x \in \{0,1\}^n$, every computation branch yields one of $\{0, 1, \perp\}$, in which $\perp$ indicates a failure, and there is always a computation branch yielding 0 or 1.

schemes, digital signatures, etc. Liu and Pass [LP20] recently showed that a meta-complexity assumption about mild average-case hardness of the time-bounded Kolmogorov Complexity is also equivalent to the existence of strong PRGs. On the other hand, in the nondeterministic case we have no known such equivalent characterisations (of nondeterministic-secure PRGs, namely, super-bits). In section 6, we introduce a definition of *super-cores* serving as a nondeterministic variant of hard-cores. We then use this concept to draw the first equivalent characterisation of super-bits.

We start by reviewing the concepts of one-way functions and hard-core predicates. 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 domain of f). More precisely, "easy to compute" means f is in P or $\mathsf{P/poly}$, and "hard to invert" means any efficient deterministic 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 $\mathsf{P/poly}$, 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 $b : \{0,1\}^n \to \{0,1\}$ in P or $\mathsf{P/poly}$ is *a hard-core of a function f* if it is impossible to efficiently predict $b(x)$ with probability at least $1/2 + 1/\mathsf{poly}(n)$ 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 to begin with are unknown. Goldreich and Levin [GL89] proved that inner product *mod* 2 is a hard-core for any function of the form $g(x, r) = (f(x), r)$, where f is any one-way function and $|x| = |r|$. Subsequently, Håstad, Impagliazzo, Levin, and Luby [HILL99] showed that strong PRGs exist if and only if one-way functions exist. This theorem can be stated equivalently as: a strong PRG exists if and only if a hard-core of some one-way function exists.

Since a strong PRG exists if and only if a hard-core of some one-way function exists, and a super-bit is the nondeterministic analogue of strong PRGs, a meaningful question to ask is:

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

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 propose a definition of super-cores of a function f, which are secure against both $\mathsf{NP/poly}$ and $\mathsf{coNP/poly}$ predictors in the sense of Definition 6.7, when $f(x)$ is presented as the input. With this definition, we can establish the following equivalence (we say a function $f : \{0,1\}^n \to \{0,1\}^{m(n)}$ is *non-shrinking* if $m(n) \geq n$ for every n):

Theorem (Informal; Theorem 6.13)**.** *There is a super-core of some non-shrinking function if and only if there is a super-bit.*

This result is analogous to 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, cannot possess any super-core. This also suggests that a one-way function could possibly possess no super-cores. (See the text before Theorem 6.16 for the definition of "predominantly one-to-one".)

Theorem (Informal; Theorem 6.16)**.** *If a non-shrinking function f is "predominantly" one-to-one, then f does not have a super-core.*

What we achieve in section 6 provides a step forward to better understand the nondeterministic hardness of PRGs and develop a sensible definition of one-way functions in the nondeterministic setting.

2.4 Justification for the existence of demi-bits and PAC-learning

Here we mention the work of Pich [Pic20] who demonstrated the plausibility of the existence of demi-bits, by showing that if the class of polynomial-size boolean circuits is not PAC-learnable by sub-exponential circuits then demi-bits exist (where the learner is allowed to generate a nondeterministic or co-nondeterministic algorithm approximating the target function). Thus, if the class of polynomial-size boolean circuits is not PAC-learnable by sub-exponential circuits and the existence of demi-bits implies the existence of super-bits, there are no $\mathsf{N\tilde{P}}/\mathsf{qpoly}$-natural properties useful against P/poly. We provide an exposition of these results for self-containedness.

Recall that **PAC-learning** algorithms can be developed from breaking PRGs (e.g., cf. [BFKL94, OS17]). The model of PAC-learning (an abbreviation of probably approximately correct learning) was introduced and developed by Valiant in [Val84, VS84, Val85]. In the PAC-learning model, the goal of a learner 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, P/poly, etc.). 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 can as well be efficiently evaluated on any given input.

Theorem (Informal; Theorem A.3 Pich [Pic20])**.** *Demi-bits exist assuming the class of polynomial-size boolean circuits is not PAC-learnable by sub-exponential circuits, where the learner is allowed to generate a nondeterministic or co-nondeterministic circuit approximating the target function.*

This result demonstrates the plausibility of the existence of demi-bits because it is widely believed (e.g., cf. [RS21]) that learning P/poly is hard. And accordingly, it is reasonable to assume that if the learner is allowed to generate a nondeterministic or co-nondeterministic circuit family, the task remains hard.

In contrast to demi-bits, it is worth mentioning that it is unknown how to relate the existence of super-bits to the hardness of PAC-learnable (namely, the proof of Theorem A.3 does not carry through when we consider super-bits).

3 Preliminaries and basic concepts

3.1 Notations and conventions

We follow 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 $s : \mathbb{N} \to \mathbb{N}$ 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 work are, by default, uniform. U_n denotes the uniform distribution over $\{0,1\}^n$ unless stated otherwise.

For functions $f, g : \{0,1\}^n \to \{0,1\}$, 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.

3.2 Computational models

Definition 3.1 (Randomized circuits; equiv. probabilistic circuit). *A circuit C is a **randomized circuit** (equivalently, a* probabilistic circuit*) if, in addition to the standard input bits (similar to the input bits of a non-randomized circuit), it contains zero or more* random input bits *(i.e., bits taken from a random distribution). We call $\{C_n\}_{n=1}^{\infty}$ a **randomized circuit family** if for every n, C_n is a randomized circuit with n standard input bits.*

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

Definition 3.2 (Nondeterministic and co-nondeterministic circuits). *A circuit $C(\overline{x}, \overline{r})$ is a **(co-)nondeterministic circuit** if, in addition to the standard input bits $\overline{x}$, it contains zero or more* nondeterministic input bits $\overline{r}$ *(namely, bits that are meant to control the nondeterministic decisions made by the circuit). A nondeterministic circuit with a single output bit is said to **accept** an input $\overline{\alpha} \in \{0,1\}^n$ to $\overline{x}$ iff there* exists *an assignment $\overline{\beta} \in \{0,1\}^{|\overline{r}|}$ to $\overline{r}$ such that $C(\overline{\alpha}, \overline{\beta}) = 1$ (and otherwise it is said to reject $\overline{x}$). A co-nondeterministic circuit with a single output bit is said to **reject** an input $\overline{\alpha} \in \{0,1\}^n$ to $\overline{x}$ iff there exists an assignment $\overline{\beta} \in \{0,1\}^{|\overline{r}|}$ to $\overline{r}$ such that $C(\overline{\alpha}, \overline{\beta}) = 0$ (and otherwise it is said to accept $\overline{x}$). We call $\{C_n\}$ a (co-)nondeterministic circuit family if for every n, C_n is a (co-)nondeterministic circuit with n standard input bits.*

Definition 3.3 (Oracle circuits). *C is an **oracle circuit** if it is allowed to use oracle gates. We write C as $C^{f_1,\ldots,f_k}$ if C has oracle gates computing Boolean functions $f_1, \ldots, f_k$.*

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 in the circuit.

3.3 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 3.4 (Natural properties [RR97])**.** *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 function family is not in Λ) is called a **Γ-natural proof** against Λ if it uses, explicitly or implicitly, some Γ-natural combinatorial property useful against Λ. Especially, a $\mathsf{P/poly}$-natural proof against $\mathsf{P/poly}$ is a proof that uses a $\mathsf{P/poly}$-natural combinatorial properties useful against $\mathsf{P/poly}$.

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 against Λ do or do not exist, what we mean, in a mathematically rigorous sense, is Γ-natural combinatorial properties against Λ do or do not exist.

3.4 Pseudorandom generators

We recall here the basic definition of pseudorandom generators. As mentioned in the introduction, all the distributions we consider in this work are, by default, uniform, and U_n denotes the uniform distribution over $\{0,1\}^n$ unless stated otherwise.

Definition 3.5 (Generators)**.** *A function family $g_n : \{0,1\}^n \to \{0,1\}^{l(n)}$ is a* ***generator*** *if $g_n \in$ $\mathsf{P/poly}$ 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 $l(n)$ is called* the stretching length*).*

We note that all the generators in this work will be computable in $\mathsf{P/poly}$, although in more general settings, it is not required that generators are $\mathsf{P/poly}$-computable (cf. [NW94]).

Definition 3.6 (Standard hardness)**.** *Let $g_n : \{0,1\}^n \to \{0,1\}^{l(n)}$ be a generator. Then the* ***hardness*** *$H(g_n)$ of g_n is the minimal s for which there exists a (deterministic) 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 3.7 ((Strong) pseudorandom generators (PRG))**.** *A generator $g : \{0,1\}^n \to \{0,1\}^{l(n)}$ is called a (strong)* ***PRG*** *if for every D in* P/poly*, every polynomial p, and all sufficiently large n's,*

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

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

We shall say that a function $f(n) : \mathbb{N} \to \mathbb{R}^+$ is *(at least) exponential* if there exists some $\varepsilon > 0$ such that $f(n) \geq 2^{n^\varepsilon}$ for all sufficiently large n's. A function is *not* (at least) exponential if for every $\varepsilon > 0$, there exist infinitely many n's such that $f(n) < 2^{n^\varepsilon}$; this latter condition 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 considered quite plausible because many intractable problems (e.g., factoring) seem to provide a basis for constructing such generators (cf. [Gol01]).

Conjecture. *Strong PRGs exist.*

Razborov and Rudich showed that the existence of a strong PRG rules out the existence of P/poly-natural proofs useful against P/poly [RR97].

Concrete examples of strong PRGs are unknown, as 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, generators that can fool classes of weaker distinguishers were constructed (e.g., Nisan and Wigderson [NW94]).[3]

3.5 Super-bits and demi-bits

Here we provide a brief review of the main results and open problems in [Rud97] that are relevant to our work. (Some of the text is repeated from the introduction.)

Definition 3.8 (Nondeterministic hardness)**.** *Let $g_n : \{0,1\}^n \to \{0,1\}^{l(n)}$ be a generator. Then the* ***nondeterministic hardness*** *$H_{\mathrm{nh}}(g_n)$ (also called* ***super-hardness****) of 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 left-hand side 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 generators super-bits:

[3] For weak models, both complexity-theoretic generators and cryptographic generators are known. Complexity-theoretic generators fooling AC^0 was shown by Nisan (which is the Nisan-Wigderson generator with PARITY as the hard function, and is earlier than [NW94]). Cryptographic generators are constructed for example in [DVV16]. For P/poly, both complexity-theoretic generators and cryptographic generators are unknown. However, the assumptions needed for complexity-theoretic generators (e.g, E requires exponential-size) are much weaker than those needed for cryptographic generators (e.g., that one way functions exists).

Definition 3.9 (Super-bits)**.** *A generator (in* $\mathsf{P/poly}$*)* $g_n : \{0,1\}^n \to \{0,1\}^{n+c}$ *for some* $c : \mathbb{N} \to \mathbb{N}$ *is called* c ***super-bit(s)*** *(or a* c*-super-bit(s)) if* $H_{\mathrm{nh}}(g_n) \geq 2^{n^\varepsilon}$ *for some* $\varepsilon > 0$ *and all sufficiently large* n*'s. In particular, if* $c = 1$*, we call* g_n *a super-bit.*

The term *super-bits* thus stands for pseudorandom bits that can fool "super" powerful adversaries. Rudich constructed a candidate super-bit based on the subset sum problem and conjectured that:

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

The main theorem in [Rud97] is the following one, which is proved based on the stretchability of super-bits as discussed above.

Theorem 3.10 ([Rud97])**.** *If super-bits exist, then there are no* $N\tilde{P}/qpoly$*-natural properties useful against* $\mathsf{P/poly}$*, where* $N\tilde{P}/qpoly$ *is the class of languages recognised by non-uniform, quasi-polynomial-size circuit families (where quasi-polynomial means* $n^{log^{O(1)}(n)}$*).*

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 3.4).

Rudich also proposed another notion, called demi-hardness, which he considered to be more intuitive than super-hardness:

Definition 3.11 (Demi-hardness)**.** *Let* $g_n : \{0,1\}^n \to \{0,1\}^{l(n)}$ *be a generator (in* $\mathsf{P/poly}$*). Then the **demi-hardness*** $H_{\mathrm{dh}}(g_n)$ *of* g_n *is the minimal* s *for which there exists a nondeterministic circuit* D *of size at most* s *such that*

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

We note that (2), which requires a distinguisher to make no mistakes on any 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 generators demi-bits, where "demi" is meant to stand for "half" here:

Definition 3.12 (Demi-bits)**.** *A generator (in* $\mathsf{P/poly}$*)* $g_n : \{0,1\}^n \to \{0,1\}^{n+c}$ *for some* $c : \mathbb{N} \to \mathbb{N}$ *is called* c ***demi-bit(s)*** *(or a* c*-demi-bit(s)) if* $H_{\mathrm{dh}}(g_n) \geq 2^{n^\varepsilon}$ *for some* $\varepsilon > 0$ *and all sufficiently large* n*'s. In particular, if* $c = 1$*, we call* g_n *a demi-bit.*

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

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

Accordingly, it is natural to ask:

Open problem ([Rud97])**.** *Does the existence of a demi-bit imply the existence of a super-bit?*

As discussed above, a super-bit can be stretched to polynomially many super-bits and to pseudorandom function generators secure against nondeterministic adversaries. In contrast, whether a demi-bit can be stretched to even two demi-bits was unknown prior to our work:

Open problem ([Rud97]; Resolved in section 4)**.** *Given a demi-bit, is it possible to stretch it to 2-demi-bits?*

The question that remains open is:

Open problem ([Rud97]). *Given a demi-bit, is it possible to build a pseudorandom function generator with exponential demi-hardness?*

A positive answer to the last problem would answer the following:

Open problem ([Rud97]). *Does the existence of a demi-bit rule out the existence of $N\tilde{P}/qpoly$-natural properties against* P/poly*?*

3.6 Infinitely often super-bits and demi-bits

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

Definition 3.13 (Infinitely often super-bits/demi-bits). *A generator (in* P/poly*) $g_n : \{0,1\}^n \to \{0,1\}^{n+c}$, for some $c : \mathbb{N} \to \mathbb{N}$ is called c **infinitely often (i.o.) super-bit(s)/demi-bit(s)**, if $H_{\text{nh}}(g_n) \geq 2^{n^\varepsilon}/H_{\text{dh}}(g_n) \geq 2^{n^\varepsilon}$ for some $\varepsilon > 0$ and infinitely many n's. In particular, if $c = 1$, we call g_n an i.o. super-bit/demi-bit.*

In fact, it is an easy observation that we can construct from a reasonably frequent 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. However, we are unaware if we can construct a super-bit/demi-bit from any i.o. super-bit/demi-bit.

Lemma 3.14. *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 $n \in (n_i)_{i\in\mathbb{N}}$, $H_{\text{nh}}(g_n) \geq 2^{n^\varepsilon}/H_{\text{dh}}(g_n) \geq 2^{n^\varepsilon}$, 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 $n \in S$,

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

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

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

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:

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

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

Remark. Lemma 3.14 is often used implicitly in the constructions of super-bits and demi-bits.

4 Stretching demi-bits

In this section, we answer affirmatively whether a demi-bit is stretchable, which had been an open problem from the original work of Rudich [Rud97].

We propose Demi-Bit Stretching Algorithm 4.1, 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, Demi-Bit Stretching Algorithm 4.1 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. [Yao82, Lub96]). 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 to preserve exponential demi-hardness).

Demi-Bit Stretching Algorithm 4.1. *Suppose $b_n : \{0,1\}^n \to \{0,1\}^{n+1}$ is a demi-bit and $0 < c < 1$ is a constant. We define a new generator $g : \{0,1\}^N \to \{0,1\}^{N+m}$ with input length N and stretching length $m = \lceil N^c \rceil$ as follows: given input x (of length N), let $n = \lfloor \frac{N}{m} \rfloor$ and $x = x_1 x_2 \ldots x_m r$, where each x_i has length n, and define $g(x) = b(x_1) \ldots b(x_m) r$.*

The correctness proof proceeds by contrapositive. That is, we assume there is a distinguisher D which breaks demi-bits g (stretched from a single demi-bit b by Demi-Bit Stretching Algorithm 4.1) in the desired sense, and we want to construct a distinguisher C which breaks b. Rather than applying the hybrid argument directly to D, we apply the hybrid argument to a new distinguisher D' defined based on D: the new distinguisher D' can use nondeterminism to change the pseudorandom part of the hybrids and thus "amplify" the probability of certificating randomness (intuitively, this can be viewed as changing the average-case analysis in the standard hybrid argument to a worst-case or "existence" analysis). By applying the hybrid argument to D', we are able to identify a non-empty class S_2 of random strings $y_{i+1} \ldots y_m$, which are not random witnesses (in the sense that, for each $y_{i+1} \ldots y_m$ in this class, there are no seeds $x_1, \ldots, x_i$ such that $D(b(x_1) \ldots b(x_i)\, y_{i+1} \ldots y_m) = 1$). Thus, for $y_{i+1} \ldots y_m$ in S_2, $y_i y_{i+1} \ldots y_m$ can become a random witness (i.e., there are seeds $x_1, \ldots, x_{i-1}$ such that $D(b(x_1) \ldots b(x_{i-1})\, y_i \ldots y_m) = 1$) only if y_i is truly random (i.e., not equal to $b(x)$ for some seed x). The hybrid argument also implies a "good" such $y_{i+1} \ldots y_m$ in S_2, that can identify a sufficient portion of truly random y_i. We can thereby build a new distinguisher C to distinguish truly random strings from pseudorandom ones.

In the proof, we reserve the **bold face** for random variables.

Theorem 4.2. *(Main theorem for stretching demi-bits) The generator g (with a sub-linear stretching-length), as defined in Demi-Bit Stretching Algorithm 4.1, has at least exponential demi-hardness.*

Proof. Demi-bit b and constant c are as given in Demi-Bit Stretching Algorithm 4.1. Suppose, towards contradiction that g does not have exponential demi-hardness. That is, there is a sub-exponential (in the input length, denoted by N) size nondeterministic circuit D such that, for infinitely many N's (recall $m = \lceil N^c \rceil$ and $n = \lfloor \frac{N}{m} \rfloor$, and without loss of generality, we may assume $m|N$), $\mathbb{P}\left[D(\mathbf{y_1} \dots \mathbf{y_m}) = 1\right] \geq 1/|D|$ and $\mathbb{P}\left[D(b(\mathbf{x_1}) \dots b(\mathbf{x_m})) = 1\right] = 0$, where $\mathbf{x_1}, \dots, \mathbf{x_m}$ are totally independent length-n random strings and $\mathbf{y_1}, \dots, \mathbf{y_m}$ 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' that takes a pair of inputs: the first is the same input as D's, that is, an $(N+m)$-bit string $y_1 \dots y_m$, and the second is $i \in \{0, 1, \dots, m\}$ (with i properly encoded):

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

We observe that:

$\mathbb{P}\left[D'(\mathbf{y_1} \dots \mathbf{y_m}, 0) = 1\right] = \mathbb{P}\left[D(\mathbf{y_1} \dots \mathbf{y_m}) = 1\right] \geq 1/|D|$ (by the definition of D');

$\mathbb{P}\left[D'(b(\mathbf{x_1}) \dots b(\mathbf{x_m}), m) = 1\right] = \mathbb{P}\left[D(b(\mathbf{x_1}) \dots b(\mathbf{x_m})) = 1\right] = 0$
(by assumption on D; otherwise $\mathbb{P}\left[D'(b(\mathbf{x_1}) \dots b(\mathbf{x_m}), m) = 1\right] > 0$ would mean there exist $x_1, \dots, x_m$ such that $D(b(x_1) \dots b(x_m)) = 1$);

D' is also of sub-exponential size.

We can now apply the hybrid argument to D':

$$\begin{aligned} 1/|D| &\leq \mathbb{P}\left[D'(\mathbf{y_1} \dots \mathbf{y_m}, 0) = 1\right] - \mathbb{P}\left[D'(b(\mathbf{x_1}) \dots b(\mathbf{x_m}), m) = 1\right] \\ &= \sum_i (\mathbb{P}\left[D'(b(\mathbf{x_1}) \dots b(\mathbf{x_{i-1}})\, \mathbf{y_i} \dots \mathbf{y_m}, i-1)\right] - \mathbb{P}\left[D'(b(\mathbf{x_1}) \dots b(\mathbf{x_i})\, \mathbf{y_{i+1}} \dots \mathbf{y_m}, i)\right]) \end{aligned}$$

yields there is an $i = i(N)$ (for infinitely many N's) such that

$$P_{i-1} - P_i \geq 1/(m \cdot |D|),$$

where $P_{i-1} := \mathbb{P}\left[D'(b(\mathbf{x_1}) \dots b(\mathbf{x_{i-1}}) \mathbf{y_i} \mathbf{y_{i+1}} \dots \mathbf{y_m}, i-1) = 1\right]$ and
$P_i := \mathbb{P}\left[D'(b(\mathbf{x_1}) \dots b(\mathbf{x_{i-1}}) b(\mathbf{x_i}) \mathbf{y_{i+1}} \dots \mathbf{y_m}, i) = 1\right]$. As m is sublinear in n, $m \cdot |D|$ is still sub-exponential in N.

We denote by $S = \{0,1\}^{(n+1) \times (m-i)}$ the set of $(m-i)$-tuples of $(n+1)$-bit strings, and let

$$S_1 = \{(y_{i+1}, \dots, y_m) \in S : \exists (x_1, \dots, x_i) \in \{0,1\}^{n \times i}\ D(b(x_1) \dots b(x_i) y_{i+1} \dots y_m) = 1\},$$

and

$$S_2 = S \setminus S_1.$$

We note that:

$D'(b(x_1)\dots b(x_{i-1})\, y_i y_{i+1}\dots y_m, i-1) = 1$ if and only if $D'(O\, y_i y_{i+1}\dots y_m, i-1) = 1$, where $O = 0^{(n+1)\cdot(i-1)}$ (because, by construction, $D'(\dots, i-1)$ ignores its first $i-1$ input strings);

$D'(b(x_1)\dots b(x_{i-1}) b(x_i)\; y_{i+1}\dots y_m, i) = 1$ if and only if $(y_{i+1},\dots,y_m) \in S_1$, and thus $\mathbb{P}\left[\mathbf{y_{i+1}}\dots\mathbf{y_m} \in S_1\right] = P_i$.

Therefore,

$$\begin{aligned}
P_{i-1} &= \mathbb{P}\left[D'(b(\mathbf{x_1})\dots b(\mathbf{x_{i-1}})\mathbf{y_i}\mathbf{y_{i+1}}\dots\mathbf{y_m}, i-1) = 1\right] \\
&= \mathbb{P}\left[D'(b(\mathbf{x_1})\dots b(\mathbf{x_{i-1}})\mathbf{y_i}\mathbf{y_{i+1}}\dots\mathbf{y_m}, i-1) = 1 | \mathbf{y_{i+1}}\dots\mathbf{y_m} \in S_1\right] \cdot \mathbb{P}\left[\mathbf{y_{i+1}}\dots\mathbf{y_m} \in S_1\right] \; + \\
&\quad\; \mathbb{P}\left[D'(b(\mathbf{x_1})\dots b(\mathbf{x_{i-1}})\mathbf{y_i}\mathbf{y_{i+1}}\dots\mathbf{y_m}, i-1) = 1 | \mathbf{y_{i+1}}\dots\mathbf{y_m} \in S_2\right] \cdot \mathbb{P}\left[\mathbf{y_{i+1}}\dots\mathbf{y_m} \in S_2\right] \\
&\le 1 \cdot \mathbb{P}\left[\mathbf{y_{i+1}}\dots\mathbf{y_m} \in S_1\right] \; + \\
&\quad\; \mathbb{P}\left[D'(O\,\mathbf{y_i}\mathbf{y_{i+1}}\dots\mathbf{y_m}, i-1) = 1 | \mathbf{y_{i+1}}\dots\mathbf{y_m} \in S_2\right] \cdot \mathbb{P}\left[\mathbf{y_{i+1}}\dots\mathbf{y_m} \in S_2\right] \\
&= P_i + \mathbb{P}\left[D'(O\,\mathbf{y_i}\mathbf{y_{i+1}}\dots\mathbf{y_m}, i-1) = 1 | \mathbf{y_{i+1}}\dots\mathbf{y_m} \in S_2\right] \cdot \mathbb{P}\left[\mathbf{y_{i+1}}\dots\mathbf{y_m} \in S_2\right],
\end{aligned}$$

and thus

$$\mathbb{P}\left[D'(O\,\mathbf{y_i}\mathbf{y_{i+1}}\dots\mathbf{y_m}, i-1) = 1 | \mathbf{y_{i+1}}\dots\mathbf{y_m} \in S_2\right] \cdot \mathbb{P}\left[\mathbf{y_{i+1}}\dots\mathbf{y_m} \in S_2\right] \ge P_{i-1} - P_i \ge 1/(m\cdot|D|).$$

In particular, $\mathbb{P}\left[\mathbf{y_{i+1}}\dots\mathbf{y_m} \in S_2\right] > 0$ and $\mathbb{P}\left[D'(O\,\mathbf{y_i}\,\mathbf{y_{i+1}}\dots\mathbf{y_m}, i-1) = 1 | \mathbf{y_{i+1}}\dots\mathbf{y_m} \in S_2\right] \ge 1/(m\cdot|D|)$, which imply there is a *fixed* $(y_{i+1}\dots y_m) \in S_2$ such that

$$\mathbb{P}\left[D'(O\,\mathbf{y_i}\, y_{i+1}\dots y_m, i-1) = 1\right] \ge 1/(m\cdot|D|).$$

We now define another nondeterministic circuit C by $C(\mathbf{y_i}) := D'(O\,\mathbf{y_i} y_{i+1}\dots y_m, i-1)$ with $\mathbf{y_i} \in \{0,1\}^{n+1}$ as the input variable. 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) $\mathbb{P}\left[C(\mathbf{y_i}) = 1\right] = \mathbb{P}\left[D'(O\,\mathbf{y_i}\, y_{i+1}\dots y_m, i-1) = 1\right] \ge 1/(m\cdot|D|)$, where $m\cdot|D|$ is sub-exponential in n;

(2) $\mathbb{P}\left[C(b(\mathbf{x_i})) = 1\right] = \mathbb{P}\left[D'(O\, b(\mathbf{x_i})\, y_{i+1}\dots y_m, i-1) = 1\right] = 0$, because $y_{i+1}\dots y_m \in S_2$ implies there is no $x_1,\dots,x_i$ such that $D(b(x_1)\dots b(x_i) y_{i+1}\dots y_m) = 1$.

Since C breaks b in the above sense, we reach a contradiction with the assumption that b is a demi-bit. □

The condition $c < 1$ in Demi-Bit Stretching Algorithm 4.1 guarantees $n = N^{1-c}$ is polynomially related to N and an infinite monotone sequence of N yields an infinite monotone sequence of n.

A key step that makes this proof work is that nondeterministically guessing seeds $x_1,\dots,x_{i-1}$ in $b(x_1)\dots b(x_{i-1})z_i$ preserves the "randomness-structure" of $b(x_1)\dots b(x_{i-1})z_i$, in the sense that: when $z_i = b(\cdot)$ is pseudorandom, the nondeterministic guess preserves the form $b(\cdot)\dots b(\cdot)b(\cdot)$ (i.e., i equal-length pseudorandom chunks); and when $z_i = y$ is truly random, it preserves the form $b(\cdot)\dots b(\cdot)y$ (i.e., $i-1$ equal-length pseudorandom chunks followed by a truly random chunk y of the same length). For common stretching algorithms that produce exponentially many new bits (e.g., recursively applying a one-bit generator), it is unclear how to use nondeterminism in a way that respects the "randomness-structure" of a given string. Nevertheless, the new proof technique may inspire researchers to further explore the stretchability of demi-bits. On the other hand, the fact could also be that there is a specific demi-bit which cannot be stretched to exponentially many demi-bits by the standard stretching algorithms which are applied to super-bits and strong PRGs.

4.1 Applications in average-case complexity

In this section we show that Theorem 4.2 implies an equivalence between two different parametric regimes of zero-error average-case hardness of time-bounded Kolmogorov complexity against $\mathsf{NP}/\mathsf{poly}$ machines.

We need the following: a ***hitting set generator against a class of decision problems*** $\mathcal{C} \subseteq 2^{\{0,1\}^N}$ is a function $g : \{0,1\}^n \to \{0,1\}^N$, for $n < N$, such that the image of g *hits* (namely, intersects) every dense enough set A in $\mathcal{C}$ (that is, $|A| \geq \frac{2^N}{N^{O(1)}}$).

As observed by Santhanam [San20]:

Proposition 4.3 ([San20])**.** *Let $n < N$. A hitting set generator $g : \{0,1\}^n \to \{0,1\}^N$ computable in the class $\mathcal{D}$ against* $\mathsf{NP}/\mathsf{poly}$ *exists iff there exists a demi-bit $b : \{0,1\}^n \to \{0,1\}^N$ computable in $\mathcal{D}$ (against* $\mathsf{NP}/\mathsf{poly}$*).*

Proof. If $g : \{0,1\}^n \to \{0,1\}^N$ is a hitting set generator against the class $\mathcal{C}$ of decision problems decidable by nondeterministic polynomial-size circuits, it is also a demi-bit in the sense that no machine $C \in \mathcal{C}$ of polynomial-size $|C| = N^{O(1)}$ can break g, since otherwise $\mathbb{P}[C(U_N) = 1] \geq 1/N^{O(1)}$ and $\mathbb{P}[C(g(U_n)) = 1] = 0$, contradicting the assumption that g is a hitting set generator against $\mathcal{C}$. Conversely, if b is a demi-bit, than it is also a hitting set generator against $\mathcal{C}$, because if a circuit C in $\mathcal{C}$ outputs 1 to a dense enough set of inputs it must also output 1 on a string in the image of b, or else C would break the demi-bit. □

Note that Demi-Bit Stretching Algorithm 4.1 applies also to demi-bits computable in *uniform* polynomial-time (the stretching algorithm is uniform, assuming the original demi-bit is, since it simply applies the demi-bit on different parts of the input). This is important for us, since to talk about Kolmogorov complexity we need machines to be of fixed size, even when the input length changes. Notice, on the other hand, that the proof that the stretching algorithm preserves its hardness necessitates that the adversary D is *non*-uniform (this is the reason in Theorem 4.5 we work against $\mathsf{NP}/\mathsf{poly}$ adversaries).

We define the ***t-bounded Kolmogorov complexity of string*** x, denoted $\mathrm{K}^t(x)$, to be the minimal length of a string D such that the universal Turing machine $U(D)$ (we fix some such universal machine) runs in time at most t and outputs x. See [ABK+06] for more details about time-bounded Kolmogorov complexity and Definition 9 there for the definition of time-bounded Kolmogorov complexity of strings (that definition actually produces the ith bit of the string x given an index i and D as inputs to U, but this does not change our result).

Recall Definition 2.1 of the languages $\mathrm{K}^t[s]$ and $\mathrm{K}^{\mathsf{poly}}[s(n)]$.

We also need to define precisely the concept of zero-error average-case hardness against the class $\mathsf{NP}/\mathsf{poly}$ (equivalently, nondeterministic circuits as in Definition 3.2).

Definition 4.4 (Zero-error average-case hardness against $\mathsf{NP}/\mathsf{poly}$)**.** *We say that a language $L \in \{0,1\}^*$ is* **zero-error average-case easy** *for* $\mathsf{NP}/\mathsf{poly}$ *if there is an* $\mathsf{NP}/\mathsf{poly}$ *machine for which all the following hold: (i) every computation-path terminates with either a Yes, No or Don't-Know state; (ii) for a given input x no two distinct computation-paths terminates with both Yes and No; (iii) we say that the machine answers Yes (No) on input x if there* exists *a computation-path terminating in Yes (resp. No) given x; (iv) otherwise (namely, all computation-paths given input x terminate in Don't-Know) we say that the machine* does not know *the answer for x; (v) the machine never makes a mistake when answering Yes or No (on the other hand, it can answer Don't-Know* *on either members of L or non-members); and (vi) the machine (correctly) answers Yes or No on at least a polynomial fraction of inputs in L (i.e., at least $2^n/n^c$ of strings in $L \cap \{0,1\}^n$, for every sufficiently large n and for some fixed constant c independent of n). If L is not zero-error average-case easy for* $\mathsf{NP/poly}$ *we say that L is* **zero-error average-case hard** *against* $\mathsf{NP/poly}$.

The main equivalence is the following:

Theorem 4.5 (Equivalence for average-case time-bounded Kolmogorov complexity)**.** $\mathrm{K}^{\mathsf{poly}}[n-O(1)]$ *is zero-error average-case hard against* $\mathsf{NP/poly}$ *machines iff* $\mathrm{K}^{\mathsf{poly}}[n-o(n)]$ *is zero-error average-case hard against* $\mathsf{NP/poly}$ *machines.*

Proof. By Theorem 4.2 and the equivalence of (uniform polytime computable) demi-bits and hitting set generators (HSGs) against $\mathsf{NP/poly}$ (Proposition 4.3), it suffices to show that for a size function $s : \mathbb{N} \to \mathbb{N}$ with $s(n) < n - O(1)$, $\mathrm{K}^{\mathsf{poly}}[s(n)]$ is zero-error average-case hard against $\mathsf{NP/poly}$ machines iff there is a HSG computable in uniform polytime $H : \{0,1\}^{s(n)} \to \{0,1\}^n$ against $\mathsf{NP/poly}$.

($\Longleftarrow$) Assume that $H : \{0,1\}^{s(n)} \to \{0,1\}^n$ is a HSG, computable in uniform poly-time, against $\mathsf{NP/poly}$. Then, for every constant k (independent of n) and sufficiently large n, $\mathrm{Im}(H) \cap \{0,1\}^n$ intersects all $\mathsf{NP/poly}$-computable sets $A_n \subseteq \{0,1\}^n$ for which $|A_n| \geq 2^n/n^k$ for all n. We need to show that for every constant c, $\mathrm{K}^{n^c}[s(n)] \subseteq \{0,1\}^n$ is zero-error average-case hard for $\mathsf{NP/poly}$. We show that there is no $\mathsf{NP/poly}$ machine that answers (correctly) one of Yes or No answers on at least $2^n/n^k$ input strings from $\{0,1\}^n$, and on the rest input strings in $\{0,1\}^n$ answers Don't-Know, and moreover makes no mistakes. Assume otherwise, then there is an $\mathsf{NP/poly}$ machine that answers (correctly) No for at least $2^n/n^{O(k)}$ input strings from $\{0,1\}^n$; this is because most input strings do not have short time-bounded Kolmogorov complexity, that is, a polynomial fraction of the inputs $x \in \{0,1\}^n$, for every n, are not in $\mathrm{K}^{\mathsf{poly}}[s(n)]$, for $s(n)$ between $|x|^\varepsilon$ and $|x|$ (for a constant $0 < \varepsilon < 1$; see [ABK$^+$06, Section 2.6] and references therein). Hence, there is an $\mathsf{NP/poly}/$ machine that accepts (correctly) at least $2^n/n^{O(k)}$ "hard strings" from $\{0,1\}^n$ (namely, strings not in $\mathrm{K}^{\mathsf{poly}}[n]$), and rejects all other strings in $\{0,1\}^n$: in particular, the $\mathsf{NP/poly}/$ machine guesses the witness the input string being hard, and if the witness is correct it accepts, and otherwise it rejects.

We thus get a contradiction to H being a HSG against NP: there is a dense NP-language containing at least $2^n/n^{O(k)}$ "hard strings" from $\{0,1\}^n$. But if D is such an NP machine for this language, then D breaks the HSG H: for every string in $\mathrm{Im}(H)$ the machine D Rejects, since it has a small $\mathrm{K}^{\mathsf{poly}}$ complexity by the assumption that H is computable in uniform poly-time (in other words, every string x of length n in $\mathrm{Im}(H)$ is such that $x \in \mathrm{K}^{n^r}[s(n)+O(1)]$, for some constant r independent of n, by assumption that H is computable in uniform poly-time). Hence, H does not hit the dense $\mathsf{NP/poly}$-set defined by D, a contradiction.

($\Longrightarrow$) We assume that $\mathrm{K}^{\mathsf{poly}}[s(n)]$ is zero-error average-case hard against $\mathsf{NP/poly}$. Let $H : \{0,1\}^{s(n)} \to \{0,1\}^n$ be a mapping defined so that the input $x \in \{0,1\}^{s(n)}$ is fed into a universal Turing machine to be ran in time n^k, for some constant k independent of n, and the output of the universal machine is a string of length n (or the string $0 \cdots 0$ of n zeros if the algorithm does not terminate after n^k steps). We show that H is a HSG against $\mathsf{NP/poly}$ (computable in uniform $n^{O(1)}$-time, by assumption).

Assume by way of contradiction that H is not a HSG against $\mathsf{NP/poly}$. Then, there is an $\mathsf{NP/poly}$ machine D that accepts at least $2^n/n^{O(1)}$ strings in $\{0,1\}^n$, but rejects every string in $\mathrm{Im}(H)$. Therefore, there is an $\mathsf{NP/poly}$ machine D that correctly accepts at least $2^n/n^{O(1)}$ strings $x \in \{0,1\}^n$ with $x \notin \mathrm{K}^{\mathsf{poly}}[s(|x|)]$, and rejects all other strings in $\{0,1\}^n$. This contradicts our assumption, because we can construct an $\mathsf{NP/poly}$ machine D' that zero-error decides on average $\mathrm{K}^{\mathsf{poly}}[s(|x|)]$: in D simply replace an Accept state with a Yes state, and a Reject state with a Don't-Know state. □

4.2 Application to proof complexity generators

Corollary 4.6 (Stretching proof complexity generators)**.** *Let $b : \{0,1\}^n \to \{0,1\}^{n+1}$ be a demi-bit computable in* $\mathsf{P/poly}$. *Let $0 < c < 1$ be a constant and $\ell = n + n^c$. Then, there is a proof complexity generator $g : \{0,1\}^n \to \{0,1\}^\ell$ in* $\mathsf{P/poly}$*, such that for every propositional proof system, with probability at least $1 - \frac{1}{\ell^{\omega(1)}}$ over the choice of $r \in \{0,1\}^\ell$, there are no* $\mathsf{poly}(n)$*-size proofs of the tautology $\tau(g)_r$.*

Proof. By Theorem 4.2, we can stretch the demi-bit b to yield new demi-bits $g : \{0,1\}^n \to \{0,1\}^\ell$. By assumption, b is computable in $\mathsf{P/poly}$, and by Demi-Bit Stretching Algorithm 4.1 (namely, the algorithm that stretches the demi-bit, which by inspection involves only applications of the original demi-bit function on different sub-parts of the seed) g is also in $\mathsf{P/poly}$.

Assume by way of contradiction that there is a constant k and a propositional proof system R that admits $\mathsf{poly}(n)$-size proofs of the tautology $\tau(g)_r$, for all r in the set $S \subseteq \{0,1\}^\ell \setminus \mathrm{Im}(g)$ where $|S| \geq 2^\ell/\ell^k$ (namely, it is *not* true that the tautology $\tau(g)_r$ does *not* have polynomial-size R-proofs for all r in $\{0,1\}^\ell \setminus (\mathrm{Im}(g) \uplus T)$ for some T with $|T| \geq 2^\ell/\ell^{\omega(1)}$; note that $|\{0,1\}^\ell \setminus (\mathrm{Im}(g) \uplus T)| = 2^\ell(1 - 1/\ell^{\omega(1)} - 1/2^{\ell-n}) = 2^\ell(1 - 1/\ell^{\omega(1)})$).

Since g is computable in $\mathsf{P/poly}$, the tautology $\tau(g)_r$ is also of size $\mathsf{poly}(n)$. Let D be a nondeterministic circuit that gets an input $r \in S$, "guesses" an R-proof of $\tau(g)_r$ and verifies it is a correct proof. Then D is of size $n^{O(1)}$, and we have

$$\mathop{\mathbb{P}}_{y \in \{0,1\}^\ell}[D(y) = 1] \geq \frac{1}{\ell^k} \quad \text{and} \quad \mathop{\mathbb{P}}_{x \in \{0,1\}^n}[D(g(x)) = 1] = 0, \tag{3}$$

which contradicts our assumption that g are demi-bits. □

5 Nondeterministic predictability

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 this and the next section, we will use a slightly modified definition of super-bits, which is analogous to Definition 3.7 in the deterministic setting: a generator $g : \{0,1\}^n \to \{0,1\}^{m(n)}$ is called super-bits if for every D in $\mathsf{NP/poly}$, every polynomial p, and all sufficiently large n's,

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

Namely, g is super-bits if g is safe against all $\mathsf{NP/poly}$-distinguishers (the difference from the original definition of super-bits is that security is defined against all *sub-exponential*-size nondeterministic circuits, while here we define it against all polynomial-size nondeterministic circuits).

The contribution of this section provides progress in our understanding of the nondeterministic hardness of generators.

5.1 Basic deterministic predictability

We first review the notion of predictability in the deterministic setting and the equivalence between deterministic unpredictability and super-polynomial hardness of generators. The results reviewed in Section 5.1 are adapted from [Gol08].

Definition 5.1 (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 $l(n) \leq p(n)$ and $p(l(n)) \geq n$.*

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

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

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 5.3 ((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 $\mathsf{P/poly}$, a polynomial p, infinitely many n's, and an $i(n) < |Z_n|$ for each of those n's such that*

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

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

Theorem 5.4. *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 variants

In this section, we propose four new notions of unpredictability beyond the deterministic setting and characterise super-hardness of generators 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 **function-computing** algorithm, if for every input $x \in \{0,1\}^n$, every computation branch yields one of $\{0,1,\perp\}$, in which $\perp$ indicates a failure, and there is always a computation branch yielding 0 or 1. $\mathcal{A}(x) = c$ for some $c \in \{0,1\}$ if, on input x, every computation branch either yields c or $\perp$; otherwise $\mathcal{A}(x) = \perp$. Hence, if $\mathcal{A}(x) = \perp$, then there are a computation branch yielding 0 and another computation branch yielding 1. We say $\mathcal{A}$ is total if $\mathcal{A}(x) \in \{0,1\}$ for every $x \in \{0,1\}^n$. Obviously, total function-computing algorithms constitute a subclass of function-computing algorithms. If we restrict the algorithms to be total for defining $\cap$-unpredictability in Definition 5.5, the diagram shown in Summary 5.9 at the end of this section remains unchanged. The advantage of not putting this restriction is that the property $\cap$-unpredictability becomes slightly stronger and thus, with this restriction we achieve a tighter "sandwiched" characterisation of nondeterministic hardness in Summary 5.9.

We also remark that a decision problem is in $\mathsf{NP/poly} \cap \mathsf{coNP/poly}$ if and only if it is decidable by a (total) nondeterministic polynomial-size function-computing algorithm.

Definition 5.5 (Nondeterministic predictability)**.**

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

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

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

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

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

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

$\cup$**-predictability.** *An ensemble $(Z_n)_{n\in\mathbb{N}}$ is called $\cup$-predictable (a short for $(\mathsf{NP/poly} \cup \mathsf{coNP/poly})$-predictable) if it is $\mathsf{NP/poly}$-predictable or $\mathsf{coNP/poly}$-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 $(\mathsf{NP/poly} \cap \mathsf{coNP/poly})$-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*

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

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 $\mathsf{NP/poly}$-predictable $\wedge$ $\mathsf{coNP/poly}$-predictable. In fact, being $\cap$-predictable is a stronger assumption than being $\mathsf{NP/poly}$-predictable $\wedge$ $\mathsf{coNP/poly}$-predictable (see Lemma Lemma 5.6). In other words, for an ensemble (Z_n), being $\cap$-unpredictable is a weaker property than being $\mathsf{NP/poly}$-unpredictable $\vee$ $\mathsf{coNP/poly}$-unpredictable. Nevertheless, being $\cap$-unpredictable is still a non-trivial property because it is stronger than being $\mathsf{P/poly}$-unpredictable (a deterministic algorithm is a total function-computing algorithm).

Lemma 5.6. *If an ensemble (Z_n) is $\cap$-predictable, it is both* $\mathsf{NP/poly}$*-predictable and* $\mathsf{coNP/poly}$*-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

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

We define $\mathcal{A}_1$ in $\mathsf{NP/poly}$ and $\mathcal{A}_0$ in $\mathsf{coNP/poly}$ 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 $\mathcal{A}_j$, if $\mathcal{A}(Y) \in \{0,1\}$, $\mathcal{A}_j(Y) = \mathcal{A}(Y)$. Thus, for $j \in \{0,1\}$, we have

$$\begin{aligned}\mathbb{P}\left[\mathcal{A}_j(Z_n[1\dots i]) = Z_n[i+1]\right] &\geq \mathbb{P}\left[\mathcal{A}_j(Z_n[1\dots i]) = Z_n[i+1] \wedge \mathcal{A}(Z_n[1\dots i]) \in \{0,1\}\right] \\ &= \mathbb{P}\left[\mathcal{A}(Z_n[1\dots i]) = Z_n[i+1] \wedge \mathcal{A}(Z_n[1\dots i]) \in \{0,1\}\right] \\ &= \mathbb{P}\left[\mathcal{A}(Z_n[1\dots i]) = Z_n[i+1]\right] \\ &\geq 1/2 + 1/p(n).\end{aligned}$$

□

Proposition 5.7. *If $g : \{0,1\}^n \to \{0,1\}^{m(n)}$ 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 $i(n) < |g(U_n)|$ for each of those n's such that

$$\mathbb{P}\left[\mathcal{A}(Z_i) = z\right] \geq 1/2 + 1/p(n),$$

where $Z_i = g(U_n)[1\dots i]$ and $z = g(U_n)[i+1]$.

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

> Given input $Y \in \{0,1\}^m$, D runs $\mathcal{A}$ on $Y[1\dots i]$ 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; else, D outputs 0.

For a fixed n, we let $f = \mathbb{P}\left[\mathcal{A}(U_i) = \perp\right]$ and use b to denote a random bit. Recall that $\mathcal{A}(U_i) = \perp$ implies the computation tree of $\mathcal{A}$ on U_i has both 0 and 1 as leaves. Then by the definition of D, we have:

$$\begin{aligned}\mathbb{P}\left[D(U_m) = 1\right] &= \mathbb{P}\left[\mathcal{A}(U_i) = \perp\right] + \mathbb{P}\left[\mathcal{A}(U_i) \in \{0,1\} \wedge \mathcal{A}(U_i) \neq b\right] \\ &= \mathbb{P}\left[\mathcal{A}(U_i) = \perp\right] + \mathbb{P}\left[\mathcal{A}(U_i) \in \{0,1\}\right] \cdot \mathbb{P}\left[\mathcal{A}(U_i) \neq b | \mathcal{A}(U_i) \in \{0,1\}\right] \\ &= f + (1-f) \cdot 1/2 \\ &\geq 1/2.\end{aligned}$$

On the other hand:

$$\begin{aligned}\mathbb{P}\left[D(g(U_n)) = 1\right] &= \mathbb{P}\left[\mathcal{A}(Z_i) = \perp\right] + \mathbb{P}\left[\mathcal{A}(Z_i) \in \{0,1\} \wedge \mathcal{A}(Z_i) \neq z\right] \\ &= \mathbb{P}\left[\mathcal{A}(Z_i) = \perp\right] + \left(\mathbb{P}\left[\mathcal{A}(Z_i) \in \{0,1\}\right] - \mathbb{P}\left[\mathcal{A}(Z_i) \in \{0,1\} \wedge \mathcal{A}(Z_i) = z\right]\right) \\ &= \mathbb{P}\left[\mathcal{A}(Z_i) = \perp\right] + \mathbb{P}\left[\mathcal{A}(Z_i) \in \{0,1\}\right] - \mathbb{P}\left[\mathcal{A}(Z_i) = z\right] \\ &= 1 - \mathbb{P}\left[\mathcal{A}(Z_i) = z\right] \\ &\leq 1/2 - 1/p(n).\end{aligned}$$

Therefore,

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

□

Proposition 5.8. *If $g(U_n)$ is $\cup$-unpredictable (i.e., $\mathsf{NP/poly}$-unpredictable $\wedge$ $\mathsf{coNP/poly}$-unpredictable), where $g : \{0,1\}^n \to \{0,1\}^{m(n)} \in P/poly$ 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 $H_i = g(U_n)[0 \ldots i] \cdot U_m[i+1 \ldots m]$ and denote $Z_i = g(U_n)[1 \ldots i]$ and $z_i = g(U_n)[i]$. Then, by the hybrid argument, there exist an algorithm D in $\mathsf{NP/poly}$, 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

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

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 $\mathsf{NP/poly}$ and $\mathcal{A}_2$ in $\mathsf{coNP/poly}$ 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

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

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

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

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

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 $\mathsf{NP}/\mathsf{poly}$-predictable or $\mathsf{coNP}/\mathsf{poly}$-predictable. □

Summary 5.9. *We summarise Lemma 5.6, Proposition 5.7, and Proposition 5.8 together as the following chain of inequalities. Here, $A \leq B$ means, if an ensemble $g(U_n)$ has property B, then it also has property A (see introduction Section 2.2 for a more pictorial version of these inequalities):*

$$\mathsf{P}/\mathsf{poly}\textit{-unpredictability} \leq \cap\textit{-unpredictability} \tag{4}$$

$$\cap\textit{-unpredictability} \leq \textit{super-polynomial nondeterministic hardness} \leq \cup\textit{-unpredictability} \tag{5}$$

$$\cap\textit{-unpredictability} \leq \mathsf{NP}/\mathsf{poly}\textit{-unpredictability} \leq \cup\textit{-unpredictability.} \tag{6}$$

$$\cap\textit{-unpredictability} \leq \mathsf{coNP}/\mathsf{poly}\textit{-unpredictability} \leq \cup\textit{-unpredictability.} \tag{7}$$

In contrast to Theorem 5.4, which is a precise 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 6.2, 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. *Could the inequalities in **Summary** be further refined or classified? For example, is there any relation between super-hardness and $\mathsf{NP}/\mathsf{poly}$-unpredictable $\vee$ $\mathsf{coNP}/\mathsf{poly}$-unpredictable?*

6 Super-core predicates

In the next subsection we review the concepts of one-way functions and hard-core predicates as well as their known connections to strong PRGs. In Section 6.2 we introduce the concept of a super-core predicate, and investigate its connections to super-bits. This provides a step forward for suggesting a sensible definition of one-way functions in the nondeterministic setting.

6.1 One-way functions and hard-core predicates

We start by reviewing the standard 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 (see [Gol08] for more details).

Definition 6.1 (One-way functions). *A function $f : \{0,1\}^* \to \{0,1\}^*$ 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,*

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

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 $\mathsf{P/poly}$ 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 are equivalent:

Theorem 6.2. *One-way functions exist if and only if strong PRGs exist.*

Since we can construct a length-preserving one-way function from an arbitrary one-way function, Theorem 6.2 can alternatively be stated as:

Theorem 6.3. *Length-preserving one-way functions exist if and only if strong PRGs exist.*

The converse implication of Theorem 6.2 is rather straightforward, while the forward implication, in fact, relies on a concept closely related to one-way functions, called *hard-core predicates*. Even when assuming that a hard-core predicate can produce one more bit from a seed x, the known proof of the forward implication in Theorem 6.3 is still rather involved.

Definition 6.4 (Hard-core predicates)**.** *A predicate $b : \{0,1\}^* \to \{0,1\}$ in $\mathsf{P/poly}$ is called a* ***hard-core*** *of a function f if there do not exist an algorithm $\mathcal{A}$ in $\mathsf{P/poly}$, a polynomial $p(\cdot)$, and infinitely many n's such that*

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

In other words, b is a hard-core of f if it is safe against (in terms of prediction) all $\mathsf{P/poly}$ algorithms, which may be due to an information loss of f (e.g, $b(x) = x[n]$ is a hard-core of $f(x) = x[1\ldots(n-1)]$) 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 6.5. *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 3.14) 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 "essentially" has the same hard-core predicate (where here it's not f itself that has the hard-core predicate, rather f' as above). It is an easy exercise to show that there does not exist a predicate b that is hard core for every one-way function.

Therefore, Theorem 6.3 can be reformulated as:

Theorem 6.6. *There is a hard-core b of some length-preserving one-way function if and only if there is a strong PRG g.*

We will develop a nondeterministic variant (Theorem 6.13) of the last theorem in the following section.

6.2 Nondeterministic variants

In this section, we introduce the concept of super-core predicates and investigate its connection to super-bits as well as which functions may or may not have super-cores.

Definition 6.7 (Super-core predicates)**.** *A predicate $b : \{0,1\}^n \to \{0,1\}$ in* $\mathsf{P/poly}$ *is called a **super-core** of a function $f : \{0,1\}^n \to \{0,1\}^{m(n)}$ if there do not exist an algorithm $\mathcal{A}_1$ in* $\mathsf{NP/poly}$*, an algorithm $\mathcal{A}_2$ in* $\mathsf{coNP/poly}$*, polynomial $p(\cdot)$, and infinitely many n's such that*

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

nor

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

For convenience, we define some abbreviations for the formulas above (the input 1^n is omitted):

$$\begin{aligned}
①(\mathcal{A}_1) &:= \mathbb{P}_{x\in\{0,1\}^n}[\mathcal{A}_1(f(x)) = b(x) = 0],\\
②(\mathcal{A}_2) &:= \mathbb{P}_{x\in\{0,1\}^n}[\mathcal{A}_2(f(x)) = b(x) = 1],\\
③(\mathcal{A}_1) &:= \tfrac{1}{2}\mathbb{P}_{y\in\{0,1\}^{m(n)}}[\mathcal{A}_1(y) = 1],\\
④(\mathcal{A}_2) &:= \tfrac{1}{2}\mathbb{P}_{y\in\{0,1\}^{m(n)}}[\mathcal{A}_2(y) = 0].
\end{aligned}$$

Our intention here is to come up with a nondeterministic variant of hard-core predicates and to build a relation between the existence of super-bits and the existence of this nondeterministic variant. A super-core is safe against both nondeterministic and co-nondeterministic predictors in the above-prescribed sense. Inequality $(\star)$ can be re-written as

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

and splitting the left-hand side in terms of conditional probability on $b(x) = 0$ yields

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

As we will see in Lemma 6.8, a super-core is also a hard-core, which implies $\mathbb{P}_{x\in\{0,1\}^n}[b(x) = 0] \approx 1/2$. Thus, for an $\mathsf{NP/poly}$-predictor $\mathcal{A}_1$ to satisfy $(\star)$, it has to satisfy the following equivalent condition for some polynomial $p(n)$:

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

This condition intuitively means that if the input y given to the $\mathsf{NP/poly}$-predictor $\mathcal{A}_1$ indeed represents some $f(x)$ such that $b(x) = 0$, the $\mathsf{NP/poly}$-predictor $\mathcal{A}_1$ has to be more sensitive to detect it by outputting 0 (than when receiving a random input). $(\diamond)$ is dual to $(\star)$. Note that, under this definition, nondeterministic and co-nondeterministic adversaries are unable to satisfy their corresponding inequality in any trivial way (e.g., by outputting the constant 0 or 1).

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

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

Lemma 6.8. *If b is a super-core of function $f : \{0,1\}^n \to \{0,1\}^{m(n)}$, b is a hard-core of f.*

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

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

either $\textcircled{1}(C) + \textcircled{3}(C) \geq 1/2 + 1/(2p(n))$ or $\textcircled{2}(C) + \textcircled{4}(C) \geq 1/2 + 1/(2p(n))$. □

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

Proposition 6.9. *If b is a super-core of function $f : \{0,1\}^n \to \{0,1\}^n$ in $\mathsf{P/poly}$, 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 $\mathsf{NP/poly}$, a polynomial p, and infinitely many n's such that:

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

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

Assume $Y \in \{0,1\}^n$ is given as input. $\mathcal{A}_1$ runs D on $Y0$ and accepts when D accepts.
$\mathcal{A}_2$ runs D on $Y1$ and accepts when D rejects.

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

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

and

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

Therefore, for infinitely many n's,

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

which implies that either $\textcircled{1}(\mathcal{A}_1) + \textcircled{3}(\mathcal{A}_1) \geq 1/2 + 1/(2p(n))$ for infinitely many n's or $\textcircled{2}(\mathcal{A}_2) + \textcircled{4}(\mathcal{A}_2) \geq 1/2 + 1/(2p(n))$ for infinitely many n's, contradicting the assumption that b is a super-core of f. □

Before we establish Proposition 6.11, which is the converse of Proposition 6.9, we need the following auxiliary lemma:

Lemma 6.10. *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 $\mathsf{P/poly}$, a polynomial p, and infinitely many n's such that

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

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,

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

□

Now, we are able to show:

Proposition 6.11. *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 $f \in \mathsf{P/poly}$.*

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 $\mathcal{A} \in \mathsf{NP/poly}$, a polynomial p, and infinitely many n's such that $\textcircled{1}(\mathcal{A}) + \textcircled{3}(\mathcal{A}) \geq 1/2 + 1/p(n)$. 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 $Y[1 \dots n]$ to get one output bit c of $\mathcal{A}$.

D then outputs 1 if and only if $Y[n+1] = 0$ and $c = 1$.

Then, for infinitely many n's, we have

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

Let $1/s(n) = |\mathbb{P}\left[b(U_n) = 0\right] - 1/2|$. As g is a strong PRG, b is a hard-core of g by Lemma 6.10. 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,

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

□

By combining Proposition 6.9 and Proposition 6.11, we establish the following nondeterministic variant of Theorem 6.6.

Theorem 6.12. *There is a super-core b of some length-preserving $f \in \mathsf{P/poly}$ 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-bit if and only if there exist super-bits (i.e., we can stretch super-bits by [Rud97], we can alternatively state Theorem 6.12 as:

Theorem 6.13. *There is a super-core of some non-shrinking function in $\mathsf{P/poly}$ if and only if there are super-bits.*

Although we can relax 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, $\textcircled{1}(c) + \textcircled{3}(c) = \textcircled{2}(c) + \textcircled{4}(c) = 1/2$ 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 whether the f constructed in Proposition 6.11 from a super-bit g (we have shown f possesses a super-core $b(x) = g(x)[-1]$) is one-way or not.

Open problem 6.14. *If g is a super-bit, is $f(x) := g(x)[1 \ldots n]$ $(n = |x|)$ 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 $f : \{0,1\}^n \to \{0,1\}^{m(n)}$ is a one-way function. Then, a natural question to ask is:

Open problem 6.15. *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 certain 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 6.16. *Given constant integer $c \geq 0$ and $f : \{0,1\}^n \to \{0,1\}^{n+c}$ in $\mathsf{P/poly}$, if there exists infinitely many n's and a polynomial p such that,*

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

then f does not have a super-core.

In particular, when f is length-preserving (i.e., $c = 0$), the inequality becomes:

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

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, $\textcircled{1}(\mathcal{A}_1) = \mathbb{P}[\mathcal{A}_1(f(x)) = b(x) = 0] \geq \mathbb{P}[\mathcal{A}_1(f(x)) = b(x) = 0 \wedge x \in T_1] = \mathbb{P}[b(x) = 0 \wedge x \in T_1]$, and $\textcircled{3}(\mathcal{A}_1) = \frac{1}{2}\mathbb{P}[\mathcal{A}_1(y) = 1] \geq \frac{1}{2}\mathbb{P}[y = f(x) \wedge b(x) = 1 \text{ for some } x \in T_1]$. Similarly, $\textcircled{2}(\mathcal{A}_0) \geq \mathbb{P}[b(x) = 1 \wedge x \in T_1]$, and $\textcircled{4}(\mathcal{A}_0) \geq \frac{1}{2}\mathbb{P}[y = f(x) \wedge b(x) = 0 \text{ for some } x \in T_1]$.

Therefore, $\textcircled{1}(\mathcal{A}_1) + \textcircled{2}(\mathcal{A}_0) + \textcircled{3}(\mathcal{A}_1) + \textcircled{4}(\mathcal{A}_0) = \mathbb{P}\left[x \in T_1\right] + \frac{1}{2}\mathbb{P}\left[y \in f(T_1)\right] = \frac{|T_1|}{2^n} + \frac{1}{2} \cdot \frac{|T_1|}{2^{n+c}} = \frac{2^{1+c}+1}{2^{1+c}} \cdot \frac{|T_1|}{2^n} \geq 1 + \frac{1}{p(n)}$ for infinitely many n's, but this implies that either $\textcircled{1}(\mathcal{A}_1) + \textcircled{3}(\mathcal{A}_1) \geq \frac{1}{2} + \frac{1}{2p(n)}$ or $\textcircled{2}(\mathcal{A}_0) + \textcircled{4}(\mathcal{A}_0) \geq \frac{1}{2} + \frac{1}{2p(n)}$. □

The intuitive 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).In the theorem, we cannot trivially relax c to an arbitrary polynomial in n, as if we do so, the right-hand side of the inequality in the statement may exceed 1. This result can alternatively be proved, when f is length-preserving, by making use of other results mentioned before, 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 6.9. 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 $f(x) = y[1 \ldots n]$ 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 6.16:

Corollary 6.17. *If $f \in \mathsf{P/poly}$ 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 [Gol08]). Thus, the corollary suggests there might be strong PRGs which are not super-bits.

Besides 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:

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

Satisfactory answers to Open problem 6.14 and Open problem 6.15 will shed light on this question. A possible candidate is "a one-way function that has a super-core", but this question needs further exploration.

7 Conclusion and future directions

The current work is the first systematic investigation into nondeterministic pseudorandomness (in the cryptographic regime), building on the primitives proposed by Rudich [Rud97]. Our investigation reveals a fruitful area of potential directions. Here, we propose or summarise some intriguing future directions and open problems that we deem worth pursuing.

> 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. 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 $b : \{0,1\}^n \to \{0,1\}^{2n}$, it is still unclear whether we can construct a PRFG with exponential demi-hardness by any standard algorithm for constructing PRFGs or a novel one.
>
> Can we further refine or classify the inequalities in Summary 5.9? Can we also characterise demi-hardness in terms of unpredictability (i.e., where is the correct place of demi-hardness in that lattice)?
>
> One-way functions are one of the most central cryptographic primitives. Then, a natural question to ask is, what is a sensible definition of nondeterministic-secure one-way functions? (A first attempt may be "a one-way function which has a super-core", but this proposal demands further verification.)
>
> If g is a super-bit, is $f(x) := g(x)[1...n]$ $(n = |x|)$ 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?

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

A Demi-bits exist unless PAC-learning of small circuits is feasible

In this section, we provide a full exposition of Pich's results from [Pic20], who developed PAC-learning algorithms from breaking PRGs. These results are important for the foundations of pseudorandomness against nondeterministic adversaries since they provide justification for the existence of demi-bits.

A.1 PAC-learning

We formulate the following nonuniform version of PAC-learning:

Definition A.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. $\mathop{\mathbb{P}}_{w}[L^f(1^n, w)\ (1-\varepsilon)-$*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, and 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 such a 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 work, 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 $\mathsf{P/poly}$, learning over any polynomially generated distribution is equivalent to learning over the uniform distribution.

3. A possible source of confusion is to leave out Condition (2). However, Condition (2) is indispensable here. The reason is that without this restriction, $\mathsf{P/poly}$ can be efficiently learned, which is widely believed not to be the case (e.g., cf. [RS21]). It is an easy exercise to prove the learnability of $\mathsf{P/poly}$ without the second condition by applying the Occam's Razor theorem established in [BEHW87]. Intuitively, a learner can learn $\mathsf{P/poly}$ 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. [KV94]) 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 a digression from the theme of this work.

A.2 Learning based on nonexistence assumptions

We recall a construction from [BFKL94]: 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 [BFKL94] into our PAC-learning framework as the following lemma:

Lemma A.2. *Given a circuit class $\mathcal{C}$, if there is an m and an $s(n)$-size circuit D such that for every $C \in \mathcal{C}$*

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

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 $1/2m^2s$ 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 on $x_1, ..., x_{i-1}$ to get $C(x_1), ..., C(x_{i-1})$, and outputs C', which predicts C as follows: given any n-bit input x_i, C' emulates D on $(x_1, C(x_1), ..., x_{i-1}, C(x_{i-1}), x_i, r_i, ..., x_m, r_m)$ 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),

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

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

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

Let $p = \mathop{\mathbb{P}}_{\substack{r_1,...,r_m \\ x_1,...,x_m \, except \, x_i}} [\mathop{\mathbb{P}}_{x_i}[C'(x_i) = C(x_i)] >= 1/2 + 1/2ms]$. 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 $\underset{x_i}{\mathbb{P}}[C'(x_i) = C(x_i)] >= 1/2 + 1/2ms$, we have

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

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 $1/m^2s$. □

Comment. 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.

A proof of the following theorem was sketch in [Pic20]:

Theorem A.3. *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 $\mathsf{P/poly}$(whenever the encoding scheme for $Circuit[n^c]$ is reasonable). Because G 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 A.2, $\mathcal{C}$ 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 A.4. *Assume the nonexistence of demi-bits. Then for 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 $\mathsf{NP}/\mathsf{poly}$-natural proofs) rules out the existence of i.o. super-bits [Rud97], by Theorem A.3, we have:

Corollary A.5. *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.*

Acknowledgments

We are indebted to Jan Pich who suggested looking at demi-bits and provided many clarifications regarding his work [Pic20]. We are grateful to Rahul Santhanam for very useful discussions and specifically mentioning the potential application appearing in Theorem 4.5. Finally, we wish to thank Hanlin Ren for very useful comments on the manuscript as well as Oliver Korten and Yufeng Li for further discussions.

References

[ABK$^+$06] Eric Allender, Harry Buhrman, Michal Koucký, Dieter van Melkebeek, and Detlef Ronneburger. Power from random strings. *SIAM J. Comput.*, 35(6):1467–1493, 2006. 2.1.2, 4.1, 4.1

[ABRW04] Michael Alekhnovich, Eli Ben-Sasson, Alexander A. Razborov, and Avi Wigderson. Pseudorandom generators in propositional proof complexity. *SIAM J. Comput.*, 34(1):67–88, 2004. (A preliminary version appeared in Proceedings of the 41st Annual Symposium on Foundations of Computer Science (Redondo Beach, CA, 2000)). 2.1.3

[ACGS88] 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. 1.1

[BEHW87] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth. Occam's razor. *Inf. Process. Lett.*, 24(6):377–380, apr 1987. 3

[BFKL94] 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. 2.4, A.2

[BM84] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudorandom bits. *SIAM J. Comput.*, 13(4):850–864, nov 1984. 1.2, 2.2

[BOV07] Boaz Barak, Shien Jin Ong, and Salil P. Vadhan. Derandomization in cryptography. *SIAM J. Comput.*, 37(2):380–400, 2007. 1.1

[DVV16] Akshay Degwekar, Vinod Vaikuntanathan, and Prashant Nalini Vasudevan. Fine-grained cryptography. Cryptology ePrint Archive, Paper 2016/580, 2016. https://eprint.iacr.org/2016/580. 3

[GGM86] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *J. ACM*, 33(4):792–807, aug 1986. 1.2

[GL89] 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. (document), 2.3

[GM84] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984. 1.2, 2.1.4

[Gol01] Oded Goldreich. *Foundations of cryptography I: Basic Tools*. Cambridge: Cambridge University Press, 2001. 3.4

[Gol08] O. Goldreich. *Computational Complexity: A Conceptual Perspective*. Cambridge University Press, 2008. 5.1, 6.1, 6.2

[HILL99] 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. (document), 2.3

[IN89] 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. 1.1

[Kal05] B. S. Kaliski. Elliptic curves and cryptography : a pseudorandom bit generator and other tools. *Phd Thesis Mit*, 2005. 1.1

[Kra04] Jan Krajíček. Dual weak pigeonhole principle, pseudo-surjective functions, and provability of circuit lower bounds. *The Journal of Symbolic Logic*, 69(1):265–286, 2004. 1.1, 2.1.3

[Kra10] Jan Krajíček. *Forcing with random variables and proof complexity*, volume 382 of *London Mathematical Society Lecture Notes Series*. Cambridge Press, 2010. 2.1.3

[KV94] M. J. Kearns and U. V. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, Cambridge, MA, USA, 1994. 4

[KvM02] Adam R. Klivans and Dieter van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses. *SIAM J. Comput.*, 31(5):1501–1526, 2002. 1.1

[LP20] Yanyi Liu and Rafael Pass. On one-way functions and kolmogorov complexity. In Sandy Irani, editor, *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 1243–1254. IEEE, 2020. 2.3

[Lub96] M. Luby. *Pseudorandomness and Cryptographic Applications*, volume 1. Princeton University Press, 1996. 4

[NW94] N. Nisan and A. Wigderson. Hardness vs randomness. *Journal of Computer and System Sciences*, 49(2):149–167, 1994. 3.4, 3.4, 3

[OS17] 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. 2.4

[Pic20] J. Pich. Learning algorithms from circuit lower bounds. *CoRR*, abs/2012.14095, 2020. 2.4, A, A.2, A.2

[PS19] Jan Pich and Rahul Santhanam. Why are proof complexity lower bounds hard? In *60th Annual IEEE Symposium on Foundations of Computer Science FOCS 2019, November 9-12, 2019, Baltimore, Maryland USA*, 2019. 1.1

[Raz15] Alexander A. Razborov. Pseudorandom generators hard for k-DNF resolution and polynomial calculus resolution. *Annals of Mathematics*, 181:415–472, 2015. 2.1.3

[RR97] A. A. Razborov and S. Rudich. Natural proofs. *Journal of Computer and System Sciences*, 55(1):24–35, 1997. (document), 1.1, 1.2, 3.4, 3.4

[RS21] 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. 2.4, 3

[Rud97] S. Rudich. Super-bits, demi-bits, and NP/qpoly-natural proofs. *Journal of Computer and System Sciences*, 55:204–213, 1997. (document), 1.1, 1.1, 1.2, 1.3, 1.4, 1.5, 1.2, 2, 2.1, 2.1.1, 3.5, 3.5, 3.10, 3.5, 4, 6.2, 7, A.2

[San20] Rahul Santhanam. Pseudorandomness and the minimum circuit size problem. In Thomas Vidick, editor, *11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA*, volume 151 of *LIPIcs*, pages 68:1–68:26. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. (document), 1.1, 2.1, 2.1.2, 4.1, 4.3

[San22] Rahul Santhanam. Personal communication, 2022. 2.1.2

[SU05] Ronen Shaltiel and Christopher Umans. Simple extractors for all min-entropies and a new pseudorandom generator. *J. ACM*, 52(2):172–216, 2005. 1.1

[SvM23] Nicollas Sdroievski and Dieter van Melkebeek. Instance-wise hardness versus randomness tradeoffs for arthur-merlin protocols. *The Electronic Colloquium on Computational Complexity (ECCC)*, 2023. 1.1

[Val84] L. G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, nov 1984. 2.4

[Val85] 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. 2.4

[VS84] 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. 2.4

[Yao82] A. C. Yao. Theory and application of trapdoor functions. In *23rd Annual Symposium on Foundations of Computer Science*, FOCS '82, pages 80–91, 1982. (document), 1.2, 2.2, 4

— Page left blank for ECCC stamp —

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il