Language Compression and Pseudorandom Generators

Harry Buhrman Troy Lee
CWI and University of Amsterdam
Harry.Buhrman@cwi.nl Troy.Lee@cwi.nl

Dieter van Melkebeek*
University of Wisconsin-Madison
dieter@cs.wisc.edu

Abstract

The language compression problem asks for succinct descriptions of the strings in a language A such that the strings can be efficiently recovered from their description when given a membership oracle for A. We study randomized and nondeterministic decompression schemes and investigate how close we can get to the information theoretic lower bound of $\log \|A^{=n}\|$ for the description length of strings of length n.

Using nondeterminism alone, we can achieve the information theoretic lower bound up to an additive term of $O((\sqrt{\log \|A^{=n}\|} + \log n) \log n)$; using both nondeterminism and randomness, we can make do with an excess term of $O(\log^3 n)$. With randomness alone, we show a lower bound of $n - \log \|A^{=n}\| - O(\log n)$ on the description length of strings in A of length n, and a lower bound of $2 \cdot \log \|A^{=n}\| - O(1)$ on the length of any program that distinguishes a given string of length n in A from any other string. The latter lower bound is tight up to an additive term of $O(\log n)$.

The key ingredient for our upper bounds is the relativizable hardness versus randomness tradeoffs based on the Nisan-Wigderson pseudorandom generator construction.

1 Introduction

Data compression pervades computer science – both theory and practice. For a given language A, one would like to devise an efficient scheme that allows one to represent strings in A using few bits. Depending on the context, efficiency can refer to the compression and/or the decompression procedures. In this paper, we only worry about the efficiency of the decompression. We study generic schemes in which every string in A can be efficiently printed from its compressed form given access to a membership oracle for A, and we shoot for compression lengths that are as close as possible to the information theoretic lower bound. A standard diagonalization argument shows that we cannot realize that goal using deterministic schemes. We investigate schemes that use randomness and/or nondeterminism for the decompression.

On the positive side, we exhibit nondeterministic schemes that achieve a compression ratio which asymptotically reaches the information theoretic lower bound. The key idea is the use of relativizable hardness versus randomness tradeoffs to obtain short descriptions of strings with respect to an oracle. In order to get our nearly optimal results, we exploit recent progress on these tradeoffs in the information theoretic context of extractors, and translate it back to the computational setting of pseudorandom generators. If we allow the decompression algorithm to use randomness as well as nondeterminism, we can realize a compression that is only a negligible

*Partially supported by NSF Career award CCR-0133693.

additive term away from the information theoretic lower bound. On the negative side, we extend the standard diagonalization result to generating schemes that use randomness only. We also show that randomness alone cannot achieve a compression ratio better than twice the information theoretic optimum even if the describing program is not required to generate the string but only to distinguish it from all other strings.

1.1 Language Compression Problem

Kolmogorov Complexity was originally developed as a way to measure the amount of randomness in a string by considering the length of a shortest program which prints the string. Far beyond this initial purpose it has become an important tool in complexity theory, witnessing applications in many areas [11]. Almost all of these applications at some point make use of the following basic theorem: For any recursively enumerable set A and all $x \in A$ of length n,

$$\mathrm{C}(x) \leq \log \|A^{=n}\| + O(\log n). \tag{1}$$

This is because x can be described by its index in the enumeration of $A^{=n}$.

For certain applications, particularly in the area of derandomization, it would be useful to have an analogue of this theorem when resource bounds are placed on the program which reconstructs a string from its description. A prime example of such an application is Sipser's original proof that BPP is in the polynomial hierarchy [18]. Sipser defined a relaxation of printing complexity called distinguishing complexity. The distinguishing complexity of a string x given advice s, denoted $\mathrm{CD}(x \mid s)$, is the length of a shortest polynomial time program which on input y, s accepts if and only if $y = x$. Sipser shows there is an advice string s of length polynomial in n, and a polynomial time bound $p(n)$ such that for all $x \in A^{=n}$,

$$\mathrm{CD}^{p,A}(x|s) \leq \log \|A^{=n}\| + O(\log n).$$

In fact, Sipser argues that most advice strings s of the appropriate length work for all $x \in A^{=n}$.

While this theorem is essentially optimal in terms of program length, it has the drawback of requiring a polynomial sized advice string. Buhrman, Fortnow, and Laplante [2] eliminate this advice string at the expense of adding a factor of 2 to the program size.

Theorem 1 *There is a polynomial $p(n)$ such that for any set A and for all $x \in A^{=n}$,*

$$\mathrm{CD}^{p,A}(x) \leq 2 \log \|A^{=n}\| + O(\log n).$$

Furthermore, there is a program that achieves this bound and only queries the oracle A on its input, rejecting immediately if the answer is negative.

Buhrman, Laplante, and Miltersen [3] demonstrate a set A with $\|A\| = 2^{\Omega(n)}$ such that the factor of 2 in the description length is necessary. Thus, Theorem 1 is essentially optimal for the deterministic distinguishing version of the language compression problem. The authors of [3] further ask if the factor of 2 is also necessary for the nondeterministic variant of distinguishing complexity, that is the length of a shortest nondeterministic polynomial time program which accepts $x \in A^{=n}$ and only x when given oracle access to A.

1.2 Our Results

We answer this question and show that the factor of 2 is not necessary. In fact, we show that we can asymptotically achieve the optimal factor of 1:

Theorem 2 *There is a polynomial $p(n)$ such that for any set A and for all $x \in A^{=n}$,*

$$\mathrm{CN}^{p,A}(x) \leq \log \|A^{=n}\| + O((\sqrt{\log \|A^{=n}\|} + \log n) \log n).$$

Furthermore, there is a program that achieves this bound and only queries the oracle at length n, rejecting immediately if an answer is negative.

The notation $\mathrm{CN}^{p,A}(x)$ in Theorem 2 refers to the length of a shortest nondeterministic program that runs in time $p(|x|)$ and, when given oracle access to A, outputs x on every accepting computation path, of which there is at least one. Note that the distinction between distinguishing complexity and printing complexity disappears in a nondeterministic context since the printing program can exploit its nondeterminism to guess the unique input accepted by the distinguishing program. In particular, CN essentially coincides with nondeterministic distinguishing complexity.

Although the bound in Theorem 2 is asymptotically optimal, the excess term of $O((\sqrt{\|A^{=n}\|} + \log n) \log n)$ is larger than one might hope. By allowing the printing program to use randomness as well as nondeterminism, we can reduce the excess term to $O(\log^3 n)$. The printing procedure can be cast as an Arthur-Merlin game – Merlin can help Arthur to produce the correct string x with high probability by answering a question Arthur asks and, no matter what Merlin does, he cannot trick Arthur into outputting a string different from x except with small probability. We use the notation $\mathrm{CAM}^{p,A}(x)$ for the description length of a shortest such Arthur-Merlin protocol for x that runs in time $p(|x|)$ and in which Arthur has oracle access to A.

Theorem 3 *There is a polynomial $p(n)$ such that for any set A and for all $x \in A^{=n}$,*

$$\mathrm{CAM}^{p,A}(x) \leq \log \|A^{=n}\| + O(\log^3 n).$$

Furthermore, there is a program that achieves this bound and only queries the oracle at length n, rejecting immediately if an answer is negative.

Finally, we address the question whether randomness alone, without nondeterminism, is able to achieve the same compression ratio. We show that this is not the case in a strong sense. We show that there are sets A such that the length of efficient randomized generating programs for *any* string $x \in A^{=n}$ cannot even reach the same ballpark as the information theoretic lower bound of $\log \|A^{=n}\|$.

Theorem 4 *For any integers n, k, and t such that $0 \leq k \leq n$, there exists a set A such that $\log \|A^{=n}\| = k$ and for every $x \in A^{=n}$,*

$$\mathrm{CBP}^{t,A}(x) \geq n - \log \|A^{=n}\| - \log t - 5.$$

Here, $\mathrm{CBP}^{t,A}(x)$ denotes the minimum length of a randomized program p that runs in time t and outputs x with probability at least 2/3 when given oracle access to A.

Even for the randomized version of distinguishing of complexity, CBPD, the length of an optimal program can be up to a factor of 2 away from the information theoretic lower bound:

Theorem 5 *There exist positive constants c_1, c_2, and c_3 such that for any integers n, k, and t satisfying $k \leq c_1 n - c_2 \log t$ there exists a set A with $\log \|A^{=n}\| = k$ and a string $x \in A^{=n}$ such that*

$$\mathrm{CBPD}^{t,A}(x) \geq 2 \log \|A^{=n}\| - c_3.$$

Note that Theorem 1 implies that $\mathrm{CBPD}^{p,A}(x) \leq 2k + O(\log n)$ for some polynomial p and every $x \in A^{=n}$. Theorem 5 shows that the upper bound on CBPD implied by Theorem 1 is tight up to an additive term of $O(\log n)$.

Theorem 5 contrasts Sipser's result on CD complexity, where he showed that a random piece of information does allow us to achieve the optimal compression ratio. The distinguishing program in Sipser's result depends on the random choice, though, whereas CBPD complexity is based on a fixed program that can flip coins.

We mention that Theorem 2 has recently been applied in [10] to show a relativized world where symmetry of information fails for nondeterministic distinguishing complexity in a strong way. They also show, using Theorem 3, that a weak form of symmetry of information holds for nondeterministic distinguishing complexity with randomness.

1.3 Our Technique

We use the hardness versus randomness tradeoffs based on the Nisan-Wigderson pseudorandom generator construction [12]. Given the truth-table $x \in \{0,1\}^n$ of a Boolean function, these tradeoffs define a pseudorandom generator $G_x : \{0,1\}^d \to \{0,1\}^m$ with seed length d much less than the output length m that has the following property: If the pseudorandom distribution $G_x(U_d)$ lands in a set $B \subseteq \{0,1\}^m$ with significantly different probability than the uniform distribution U_m over $\{0,1\}^m$, then x has a succinct description with respect to B and can be efficiently recovered from that description given oracle access to B [8].

We apply the hardness versus randomness tradeoffs in the following way. Consider a set A and let $k = \log \|A^{=n}\|$. If we set B equal to the union of the range of G_x over all $x \in A^{=n}$ and set m to be slightly larger than $k + d$, then for every string x in $A^{=n}$ the pseudorandom distribution $G_x(U_d)$ lands in B with 100% certainty whereas the uniform distribution U_m lands in B with significantly smaller probability. We conclude that every $x \in A^{=n}$ can be efficiently constructed from a succinct description given oracle access to B. Moreover, the set B can be decided efficiently by a nondeterministic machine that has oracle access to A. This allows us to replace the oracle queries to B by nondeterminism and oracle queries to A, which is what we need for Theorem 2.

A similar (but simpler) reconstructive argument underlies the analysis of recent extractor constructions à la Trevisan (see [16] for an excellent survey). Trevisan [23] viewed the above hardness versus randomness tradeoffs as a transformation

$$\mathrm{TR} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$$

that takes two inputs, namely a truth-table $x \in \{0,1\}^n$ and a seed $y \in \{0,1\}^d$, and outputs the pseudorandom string $G_x(y)$. He observed that TR defines an extractor: For every distribution X on $\{0,1\}^n$ with sufficient minentropy k, the distribution $\mathrm{TR}(X, U_d)$ behaves very similar to the uniform distribution U_m with respect to *every* possible set B. The argument goes as follows: For a given set B, let us call a string $x \in \{0,1\}^n$ "bad" if $\mathrm{TR}(x, U_d)$ and U_m land in B with probabilities that are more than ϵ apart (where ϵ is some parameter). Since bad strings x with respect to B can be reconstructed from a short description, say of length $\ell(m, \epsilon)$, and each individual string

x has probability at most 2^{-k} in a source of minentropy k, the extracted distribution lands in B with the same probability as the uniform distribution up to an error term of no more than $\epsilon + 2^{\ell(m,\epsilon)-k}$. So, in order to extract as much of the minentropy of the source as possible, one needs to minimize the description length $\ell(m, \epsilon)$. This is exactly what we need for our compression result of Theorem 2. Thus, our goals run parallel to those for constructing "reconstructive" extractors that extract almost all of the minentropy of the source. We employ similar tools (such as weak designs [14]) but need to deal with a few additional complications:

- In the extractor setting, it is sufficient to argue that a nonnegligible fraction of the bad strings x has a short description. In particular, the averaging argument in the standard analysis only shows that a fraction $\Theta(\epsilon/m)$ of the bad strings x has a short description. This slack in the analysis increases the error bound for the extractor only from $\epsilon + 2^{\ell(m,\epsilon)-k}$ to $\epsilon + \Theta(m/\epsilon)2^{\ell(m,\epsilon)-k}$. In our setting, however, we cannot afford to miss any string because we need a short description for *every* string in $A^{=n}$ with respect to a single oracle B.

- As a result, our descriptions need to include more information than in the extractor setting. There are two main components in the description, namely one depending on the weak designs underlying the Nisan-Wigderson pseudorandom generator, and one specifying $O(m)$ random bits used in the averaging argument. The latter component is the one which is needed in our setting but not in the extractor context. Balancing the two contributions optimally leads to the descriptions of length $m + O(\sqrt{m} \log n)$ from Theorem 2. By allowing the describing program not only the power of nondeterminism but also the power of randomness, we can, in some sense, mimic the averaging argument from the extractor setting and eliminate the need for the second component. This results in the shorter descriptions of length about m used in Theorem 3.

- Our descriptions need to be efficient – an element $x \in A^{=n}$ can be computed in polynomial time from its description and access to an oracle for B. This implies a return from the information theoretic setting to the computational setting which formed the starting point for Trevisan's and later extractors based on the reconstructive argument. Our efficiency requirements are not as strict as in the pseudorandom generator context, though, where each bit of x can be reconstructed in randomized time $(\log n)^{O(1)}$. We can afford reconstruction times of the order $n^{O(1)}$ but the process typically needs to be deterministic.

In the above argument, the Nisan-Wigderson construction may be replaced by the recent pseudorandom generators or reconstructive extractors based on multivariate polynomials [22, 17]. However, although the latter lead to optimal hardness versus randomness tradeoffs in some sense [24], they yield worse parameters than the Nisan-Wigderson construction in our context.

1.4 Organization

In Section 2, we provide some background on Kolmogorov complexity, and formally define the Kolmogorov measures we use. We also describe two key ingredients in the recent extractor constructions, namely combinatorial designs and error-correcting codes. Section 3 contains our key Lemma, the Compression Lemma, which translates some of the recent progress on extractors back to the pseudorandom generator setting. We use the Compression Lemma to derive our upper bounds for nondeterministic schemes in Section 4, and for schemes that use both nondeterminism and randomness in Section 5. Finally, in Section 6, we present our lower bound for schemes that only use randomness.

2 Preliminaries

We use standard complexity theoretic notation as in [1] and [13]. For background and notation in Kolmogorov complexity we refer the reader to [11]. We use λ to denote the empty string, $|x|$ to denote the length of a string x, and $\|A\|$ for the cardinality of a set A. By $A^{=n}$ we mean the intersection of A with the set of strings of length n. All logarithms are base 2.

2.1 Kolmogorov Complexity

The theory of Kolmogorov Complexity begins with a universal Turing machine U, which is able to simulate the running of any other Turing machine with only a constant additive factor overhead in program length. For the theory of resource-bounded Kolmogorov complexity, we need further that the universal machine U be able to do this *efficiently*. This can be done by the well known simulation of Hennie and Stearns [7]. Thus we now fix such a universal machine U.

For a fully time constructible function t satisfying $t(n) \geq n$, the conditional time t bounded printing complexity is defined as

$$C^t(x \mid y) = \min_p \{|p| : U(p, y) = x \text{ in at most } t(|x| + |y|) \text{ steps}\}.$$

We set $C^t(x) = C^t(x \mid \lambda)$ where λ denotes the empty string. Note that the running time depends not on the length of the input p, but rather the length of the output x and the given string y. When no superscript is indicated, as in $C(x \mid y)$, we mean the above definition with no time bound restriction.

We consider a randomized version of printing complexity, CBP, defined as follows:

Definition 6 *Let U be a universal machine. Then $\mathrm{CBP}^t(x \mid y)$ is the length of a shortest program p such that*

1. $\Pr_{r \in \{0,1\}^t}[U(p, y, r)$ *outputs* $x] > 2/3$
2. $U(p, y, r)$ *runs in* $\leq t(|x| + |y|)$ *steps for all* $r \in \{0,1\}^t$

We set $\mathrm{CBP}^t(x) = \mathrm{CBP}^t(x \mid \lambda)$.

We define a nondeterministic version of printing complexity, CN, in terms of single valued nondeterministic functions (see the survey by Selman [15] for a formal definition of the latter).

Definition 7 *Let U_n be a universal nondeterministic machine. Then $\mathrm{CN}^t(x \mid y)$ is defined as the length of a shortest program p such that*

1. $U_n(p, y)$ *has at least one accepting path*
2. $U_n(p, y)$ *outputs x on every accepting path*
3. $U_n(p, y)$ *runs in* $\leq t(|x| + |y|)$ *steps.*

We set $\mathrm{CN}^t(x) = \mathrm{CN}^t(x \mid \lambda)$.

Finally, we investigate decompression algorithms that make use of both nondeterminism and randomness. For this we define a version of printing complexity based on the complexity class AM:

Definition 8 *Let U_n be a universal nondeterministic machine. The* $\mathrm{CAM}^t(x \mid y)$ *is the length of a shortest program p such that*

1. $\mathrm{Pr}_{r\in\{0,1\}^t}[U_n(p, y, r)$ *accepts, and all accepting paths output* $x] > 2/3$
2. $U_n(p, y, r)$ *runs in* $\leq t(|x| + |y|)$ *steps.*

We set $\mathrm{CAM}^t(x) = \mathrm{CAM}^t(x \mid \lambda)$.

Sipser defined a relaxation of printing complexity called *distinguishing complexity*. The time t distinguishing complexity of x given y, denoted $\mathrm{CD}^t(x \mid y)$, is the length of a shortest program which runs in time $t(|x| + |y|)$ and accepts only the string x.

Nondeterministic distinguishing complexity, CND, was originally defined in [2]. It can be seen that the measures CND and CN essentially coincide, up to additive logarithmic terms. One direction is obvious. To see $\mathrm{CN}^{t+O(|x|)}(x) \leq \mathrm{CND}^t(x) + O(\log |x|)$: if p is a nondeterministic distinguishing program for x, a nondeterministic machine given p and $|x|$ can guess a string of length $|x|$ which is accepted by p and output this string. By the nature of p, the new nondeterministic machine has at least one accepting computation path and outputs x on every accepting computation path. As $U_n(p, x)$ runs in time t, the whole procedure will take time at most $t + O(|x|)$. Thus in the sequel we will refer only to CN. A similar argument holds for CAM and its distinguishing complexity analogue.

If we only allow randomness, however, it is no longer clear if distinguishing complexity and printing complexity coincide. As our results concerning randomized decompression algorithms are lower bounds, we give the definition here for randomized distinguishing complexity which allows for stronger statements.

Definition 9 *Let U be a universal machine. Then* $\mathrm{CBPD}^t(x \mid y)$ *is the length of a shortest program p such that*

1. $\mathrm{Pr}_{r\in\{0,1\}^t}[U(p, x, y, r) = 1] > 2/3$
2. $\mathrm{Pr}_{r\in\{0,1\}^t}[U(p, z, y, r) = 0] > 2/3$ *for all* $z \neq x$
3. $U(p, z, r)$ *runs in* $\leq t(|z| + |y|)$ *steps for all* $z \in \{0, 1\}^*$

We set $\mathrm{CBPD}^t(x) = \mathrm{CBPD}^t(x \mid \lambda)$.

All of the above Kolmogorov measures can be relativized by giving the universal machine access to an oracle A. We mention the oracle as a superscript after the measure abbreviation.

2.2 Combinatorial Designs

A key ingredient of the Nisan-Wigderson generator is a collection of sets with small pairwise intersection. Following [12], a set system $\mathcal{S} = S_1, \ldots, S_m \subseteq [d]$ is called a (ℓ, ρ) *design* if for all i, $\|S_i\| = \ell$ and for all $i \neq j$ the intersection $\|S_i \cap S_j\| \leq \log \rho$.

Raz, Reingold, and Vadhan [14] observe that a weaker property on the set system $\mathcal{S}$ suffices for the construction of the Nisan-Wigderson generator. Namely, the quantity essentially used in the analysis of the generator is a bound on $\sum_{j<i} 2^{\|S_i \cap S_j\|}$. Set systems with this sum bounded by $\rho \cdot (m-1)$ for all i are called (ℓ, ρ) *weak designs*. Unlike the case with designs, there exist weak designs where the universe size d does not depend on the number of sets m.

For our purposes, we need to draw another distinction in design terminology. We need a bound on $\sum_{j<i} 2^{\|S_i \cap S_j\|}$ in terms of i. Such designs were already constructed in [14] but went unnamed. As the distinction will be important later, we give them their own name, referring to them as *uniform weak designs*.

Definition 10 *Let $\mathcal{S} = (S_1, S_2, \ldots, S_m)$ be a family of sets where for all i, $S_i \subseteq [d]$ and $\|S_i\| = \ell$.*

1. *$\mathcal{S}$ is a* weak (ℓ, ρ) design *if $\sum_{j<i} 2^{\|S_i \cap S_j\|} \le \rho \cdot (m-1)$ for all i.*
2. *$\mathcal{S}$ is a* uniform weak (ℓ, ρ) design *if $\sum_{j<i} 2^{\|S_i \cap S_j\|} \le \rho \cdot (i-1)$ for all i.*

Raz, Reingold, and Vadhan show the following lemma [14]:

Lemma 11 *For every ℓ, m and $\rho = \rho(\ell, m) > 1$ there exists a set system $\mathcal{S} = (S_1, S_2, \ldots, S_m) \subseteq [d]$ constructible in $\mathrm{poly}(m, d)$ time, with either of the following properties:*

1. *$\mathcal{S}$ is a weak $(\ell, 1)$ design with $d = O(\ell^2 \log m)$.*
2. *$\mathcal{S}$ is a uniform weak (ℓ, ρ) design with $d = O(\ell^2 / \log \rho)$.*

It is worth noting that [14] also show a matching lower bound, up to constant multiplicative factors, to the above construction of uniform weak designs. In particular, this means that $(\ell, 1)$ weak designs *cannot* be made uniform with the parameters given in item 1 [14, Remark 19].

2.3 Error-Correcting Codes

The benefits of composing the Nisan-Wigderson generator with a good list-decodable code are well demonstrated [23, 20]. We will use a concatenation of a Reed-Solomon code with an Hadamard code. The combinatorial list-decoding properties of this code suffice for our main theorems; however, using additionally the fact that this code has efficient list-decoding [19, 9] allows us to prove a stronger form of our main lemma, the Compression Lemma (Lemma 14). The properties we need are summarized in the next two lemmata.

Lemma 12 *For every integer $n \ge 0$ and positive $\delta = \delta(n)$, there is a code $\mathrm{LDC}_{n,\delta} : \{0,1\}^n \to \{0,1\}^{\bar{n}}$ where $\bar{n} = \mathrm{poly}(n/\delta)$ with the following properties:*

1. *$\mathrm{LDC}_{n,\delta}$ can be evaluated in time $\mathrm{poly}(n/\delta)$.*
2. *Given any string $\hat{y} \in \{0,1\}^{\bar{n}}$, the list of all strings $x \in \{0,1\}^n$ such that $\hat{x} = \mathrm{LDC}_{n,\delta}(x)$ and $\hat{y}$ agree in at least a $1/2 + \delta$ fraction of the positions can be generated in time $\mathrm{poly}(n/\delta)$.*

Lemma 13 *Let $\mathrm{LDC}_{n,\delta}$ be as above and $\hat{x} = \mathrm{LDC}_{n,\delta}(x)$. For every rational $\delta = \delta(n)$ there is a time bound $t = \mathrm{poly}(n/\delta)$ such that for any $\hat{y} \in \{0,1\}^{\bar{n}}$ which agrees with $\hat{x}$ on a $1/2 + \delta$ fraction of positions,*

$$C^t(x \mid \hat{y}) \le C^{t/2}(\delta) + O(\log(n/\delta)).$$

Proof: With $C^{t/2}(\delta) + O(\log n)$ bits we can describe δ, n, and the code $\mathrm{LDC}_{n,\delta}$ being used. Given $\hat{y}, n, \delta$, we can print the $\mathrm{poly}(n/\delta)$ codewords which agree with $\hat{y}$ on more than a $1/2 + \delta$ fraction of positions. By further specifying the index i of $\hat{x}$ in this list we can identify $\hat{x}$ and decode it to print x. This index i can be given with $O(\log(n/\delta))$ bits. As $\mathrm{LDC}_{n,\delta}$ is efficiently list decodable there is a function $t = \mathrm{poly}(n/\delta)$ bounding the running time of the above procedure. $\square$

3 Compression Lemma

In this section, we translate some of the recent progress on extractors back into the pseudorandom generator setting, resulting in the main tool for our upper bound results, the Compression Lemma. We first describe the function underlying Trevisan's and later extractors, hereafter referred to as Trevisan's function.

Let $P : \{0,1\}^\ell \to \{0,1\}$ be any Boolean function, and let $\mathcal{S} = (S_1, \ldots, S_m)$ be a collection of subsets of $[d]$ where $\|S_i\| = \ell$. For a string $y \in \{0,1\}^d$ let $y|_{S_i}$ be the string in $\{0,1\}^\ell$ obtained by projecting y onto the coordinates specified by S_i. Then the Nisan-Wigderson generator $\mathrm{NW}_{\mathcal{S},P}$ is defined as

$$\mathrm{NW}_{\mathcal{S},P}(y) = P(y|_{S_1}) \cdots P(y|_{S_m}).$$

Given an input length n, an output length m, a quality parameter $\delta = \delta(m)$, and a design parameter $\rho = \rho(m) > 1$, we define the following function after Trevisan. Let $\mathrm{LDC}_{n,\delta}$ be as in Lemma 12 and let $\ell = \log \bar{n}$. For $u \in \{0,1\}^n$, we view $\mathrm{LDC}(u)$ as a Boolean function $\hat{u} : \{0,1\}^\ell \to \{0,1\}$. Let $\mathcal{S}$ be a (ℓ, ρ) uniform weak design. The function $\mathrm{TR}_{\delta,\rho} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is defined as

$$\mathrm{TR}_{\delta,\rho}(u, y) = \mathrm{NW}_{\mathcal{S},\hat{u}}(y) = \hat{u}(y|_{S_1}) \cdots \hat{u}(y|_{S_m}).$$

Note that while n, m are arbitrary, we need to take the auxiliary input length d so as to satisfy the conditions of the uniform weak design.

The property of Trevisan's function that is crucial for the recent extractor constructions and for our results is the following lemma. It is a refinement of similar statements shown in [12, 23, 14], where the result was stated for circuit size in [12] and (nonuniform) description size in [23, 14]. The new feature of our version of the lemma is the combination of completeness, succinctness, and efficiency of the descriptions: *every* "bad" string with respect to B has a *very* succinct description from which it can be *efficiently* recovered given oracle access to B.

Lemma 14 (Compression Lemma) *Let $B : \{0,1\}^m \to \{0,1\}$. Given $\epsilon = \epsilon(m) > 0$, let $\delta = \epsilon/m$. If*

$$|\Pr[B(\mathrm{TR}_{\delta,\rho}(u, U_d) = 1)] - \Pr[B(U_m) = 1]| \geq \epsilon$$

then for a time bound $t = \mathrm{poly}(n/\epsilon)$, we have

$$C^{t,B}(u) \leq \rho \cdot m + d + C^{t/2}(\epsilon) + O(\log(m/\epsilon)).$$

Furthermore, there is a program that achieves this bound and only makes nonadaptive queries to B.

Proof: We follow the by now standard proof as in [12, 23, 14]. The idea is to use the distinction from the uniform distribution that can be seen with B to find a bit of the output of TR which can be predicted with advantage – with this advantage we can then approximate $\hat{u}$ and give u a short printing program using Lemma 13.

Finding a bit of the output which can be predicted with advantage can be done using the hybrid argument of Goldwasser and Micali [6]. We define $m+1$ distributions, $D_0, \ldots, D_m$, where the first i bits of D_i are distributed according to the first i bits of $\mathrm{TR}(u, U_d)$, and the last $m-i$ bits of D_i are distributed according to the last $m-i$ bits of U_m. Thus note that D_0 is distributed as U_m and D_m is distributed as $\mathrm{TR}(u, U_d)$. As $|\Pr[B(D_m)] - \Pr[B(D_0)]| \geq \epsilon$, for some i it must be the case that $|\Pr[B(D_i)] - \Pr[B(D_{i-1})]| \geq \epsilon/m$. For convenience we remove the absolute value sign by choosing $b_0 \in \{0,1\}$ such that $\Pr[B'(D_i)] - \Pr[B'(D_{i-1})] \geq \epsilon/m$, where $B'(x) = b_0 \oplus B(x)$.

Writing the distributions D_{i-1}, D_i out explicitly, we now have:

$$\Pr_{\substack{y \\ r_i,\ldots,r_m}} [B'(\hat{u}(y|_{S_1}) \cdots \hat{u}(y|_{S_{i-1}}) \hat{u}(y|_{S_i}) r_{i+1} \ldots r_m)] - \Pr_{\substack{y \\ r_i,\ldots,r_m}} [B'(\hat{u}(y|_{S_1}) \cdots \hat{u}(y|_{S_{i-1}}) r_i r_{i+1} \ldots r_m)] > \epsilon/m$$

By an averaging argument, we can fix the bits of y outside of S_i, and fix $r_{i+1}, \ldots, r_m$ to some values $c_{i+1}, \ldots, c_m$, while preserving the above difference. Renaming $y|_{S_i}$ as x, we note that x varies uniformly over $\{0,1\}^\ell$, while $\hat{u}(y|_{S_j})$ for $j \neq i$, is now a function $\hat{u}_j$ which depends only on $\|S_i \cap S_j\|$ bits of x. That is,

$$\Pr_{x,b}[B'(\hat{u}_1(x) \cdots \hat{u}_{i-1}(x) \hat{u}(x) c_{i+1} \cdots c_m)] - \Pr_{x,b}[B'(\hat{u}_1(x) \cdots \hat{u}_{i-1}(x) b c_{i+1} \cdots c_m)] > \epsilon/m \qquad (2)$$

Let $F(x,b) = \hat{u}_1(x) \cdots \hat{u}_{i-1}(x) b c_{i+1} \cdots c_m$. Our program to approximate $\hat{u}$ does the following. On input x, b it evaluates $B'(F(x,b))$ and outputs b if this evaluates to 1 and $1-b$ otherwise. Let $g_b(x)$ denote the outcome of this process. We can estimate the probability that $g_b(x)$ agrees with $\hat{u}(x)$ over the choice of x, b as follows:

$$\begin{aligned}
\Pr_{x,b}[g_b(x) = \hat{u}(x)] &= \Pr_{x,b}[g_b(x) = \hat{u}(x) | b = \hat{u}(x)] \Pr_{x,b}[b = \hat{u}(x)] + \Pr_{x,b}[g_b(x) = \hat{u}(x) | b \neq \hat{u}(x)] \Pr_{x,b}[b \neq \hat{u}(x)] \\
&= \frac{1}{2} \Pr_{x,b}[B'(F(x,b)) = 1 | b = \hat{u}(x)] + \frac{1}{2} \Pr_{x,b}[B'(F(x,b)) = 0 | b \neq \hat{u}(x)] \\
&= \frac{1}{2} + \frac{1}{2}\Big(\Pr_{x,b}[B'(F(x,b)) = 1 | b = \hat{u}(x)] - \Pr_{x,b}[B'(F(x,b)) = 1 | b \neq \hat{u}(x)] \Big) \\
&= \frac{1}{2} + \frac{1}{2}\Big(\Pr_{x}[B'(F(x,\hat{u}(x))) = 1] - \Pr_{x}[B'(F(x, 1-\hat{u}(x))) = 1] \Big) \\
&= \frac{1}{2} + \Pr_{x,b}[B'(F(x,\hat{u}(x)) = 1] - \Pr_{x,b}[B'(F(x,b)) = 1] \\
&\geq \frac{1}{2} + \frac{\epsilon}{m}
\end{aligned}$$

By an averaging argument there is a bit $b_1 \in \{0,1\}$ such that $g_{b_1}(x)$ agrees with $\hat{u}(x)$ on at least a $1/2 + \epsilon/m$ fraction of x. Note that the queries to B' are nonadaptive and that the running time of the approximation is $2^{O(\ell)} = \bar{n}^{O(1)} = \mathrm{poly}(n/\epsilon)$.

To optimize the description size of the above program, it will be useful to separate its contributions into three parts:

1. the index i, the bits b_0, b_1 and $O(\log m)$ bits to make the entire description prefix free.

2. the contribution from the seed length, that is the $d - \ell$ bits fixed outside of x.

3. the setting of the bits $c_{i+1}, \ldots, c_m$ and description of the functions $\hat{u}_1, \ldots, \hat{u}_{i-1}$.

Clearly the first item costs $O(\log m)$ bits and the second at most d. We now focus on item three.

Each function $\hat{u}_j$ is a function on $\|S_j \cap S_i\|$ bits, thus we can completely specify it by its truth table with $2^{\|S_i \cap S_j\|}$ bits. Hence we can describe all the functions $\hat{u}_1, \ldots, \hat{u}_{i-1}$ with $\sum_{j=1}^{i-1} 2^{\|S_j \cap S_i\|}$ bits, by concatenating their truth functions. We can compute the set system $\mathcal{S}$ in polynomial time and given the value of i, we can compute the sizes of $\|S_j \cap S_i\|$ and uniquely decode each function $\hat{u}_j$. Thus as $\mathcal{S}$ is a (ℓ, ρ) uniform weak design, we can describe all the functions $\hat{u}_1, \ldots, \hat{u}_{i-1}$ in $\rho \cdot (i-1)$ bits. Now adding $m - i$ bits to describe $c_{i+1}, \ldots, c_m$ we see that item (3) will cost less than $\rho \cdot (m-1)$ bits.

Putting these three items together, we conclude there is a string $\hat{y}$ which agrees with $\hat{u}$ on a $1/2 + \epsilon/m$ fraction of positions and with $C^{p,B}(\hat{y}) \leq \rho \cdot m + d + O(\log m)$. Now applying Lemma 13, we obtain the statement of the lemma. □

Substituting the uniform weak design parameters from Lemma 11 into the Compression Lemma, and optimizing with respect to ρ, we find the minimum is achieved when $\rho = 1 + \ell/\sqrt{m}$. For future reference, we record this in the following corollary.

Corollary 15 *Let B, ϵ, δ be as in Lemma 14, and let $\rho = 1 + \ell/\sqrt{m}$. If*

$$|\Pr[B(\mathrm{TR}_{\delta,\rho}(u, U_d) = 1)] - \Pr[B(U_m) = 1]| \geq \epsilon$$

then for a time bound $t = \mathrm{poly}(n/\epsilon)$, we have

$$C^{t,B}(u) \leq m + C^{t/2}(\epsilon) + O(\sqrt{m}\log(n/\epsilon)).$$

Furthermore, there is a program that achieves this bound and only makes nonadaptive queries to B.

4 Language Compression By Nondeterminism

In this section, we exhibit the power of nondeterminism in the context of the language compression problem. We show that Trevisan's function leads to compression close to the information-theoretic lower bound such that the compressed string can be recovered from its description by an efficient nondeterministic program that has oracle access to the containing language A.

The proof is an application of the Compression Lemma. In order to give short CN programs relative to A, it suffices to find a set B such that:

- Queries to B can be efficiently answered with an oracle for A and nondeterminism.
- For any $x \in A$, the distribution $\mathrm{TR}(x, U_d)$ lands in B with significantly different probability than the uniform distribution U_m.

Letting B be the set containing all strings of the form $\mathrm{TR}(x, e)$ where x ranges over A and e over all seeds of the appropriate length d, the first item will be satisfied. By taking the output length to be slightly larger than $\log \|A\| + d$, that is taking it to be "too long", we can also ensure that the second item is satisfied. We say "too long" as for this setting of m, Trevisan's function will not be an extractor for sources of min-entropy $\log \|A\|$, see also [21]. We now go through the details.

Proof: (of Theorem 2) Fix n and let $k = \log \|A^{=n}\|$. Let $\mathrm{TR}_{\delta,\rho} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ be Trevisan's function with $m = k + d + 1$. The parameters δ, ρ will be fixed later.

Define $B \subseteq \{0,1\}^m$ to be the image of $A \times \{0,1\}^d$ under the function TR. That is, $B = \{y : \exists x \in A, \exists e : \mathrm{TR}(x,e) = y\}$.

By the choice of m it is clear that $\Pr[B(U_m)] \leq 1/2$. For any element $x \in A$, however, $\Pr[B(\mathrm{TR}(x, U_d))] = 1$. Thus applying Lemma 14 with $\epsilon = 1/2$ and $\rho = 1 + \ell/\sqrt{k}$ we obtain $C^{p,B}(x) \leq (1 + \ell/\sqrt{k})(k + d + 1) + d + O(\log n)$. As $\ell = O(\log n)$ and $d = O(\sqrt{k}\log n)$ with this choice of ρ, simplifying gives $C^{p,B}(x) \leq k + O((\sqrt{k} + \log n)\log n)$.

We now show how an oracle for B can be replaced by a nondeterministic program with an oracle for A. By Lemma 14 we may assume that the queries to B are nonadaptive. It is clear the "yes" answers of the oracle B can be answered nondeterministically with an oracle for A. As the queries to B are nonadaptive, by additionally telling the program the number q of yes answers, the program can guess the q element subset of the queries which are "yes" answers and verify them. On any path where the incorrect q element subset has been guessed, at least one "yes" answer will not be verified and thus this path will reject. The description of q will only increase the program size by $O(\log n)$ bits. □

The positive use of the oracle in Theorem 2 also allows us to state the following corollary about the CN complexity of strings from an NP language.

Corollary 16 *For any set $A \in \mathrm{NP}$ there is a polynomial $p(n)$ such that for all $x \in A^{=n}$,*

$$\mathrm{CN}^p(x) \leq \log \|A^{=n}\| + O((\sqrt{\log \|A^{=n}\|} + \log n)\log n).$$

Proof: Consider the nondeterministic program with oracle access to A given by Theorem 2. Replace the oracle queries by guessing a membership witness and verifying it, rejecting whenever the verification fails. This gives us the nondeterministic generating program we need. □

5 Language Compression By Nondeterminism and Randomness

In this section, we show that if we allow the decompression algorithm both the power of nondeterminism and randomness, then we can reduce the excess in the description length over the information theoretic lower bound from $O((\sqrt{k} + \log n)\log n)$ to $O(\log^3 n)$.

In the proof of the Compression Lemma, we included as part of the description of $u \in A$ a setting of the random bits $c_{i+1}, \ldots, c_m$ fixed after position i. Including a setting of these bits in our description seems wasteful – the averaging argument of Lemma 14 shows that a $\theta(\epsilon/m)$ fraction of all $m - i$ bit strings would work equally well to describe u. In spite of this, we do not see how to avoid specifying them with nondeterminism only. However, if we allow randomization in our nondeterministic programs, or more precisely, if we consider Arthur-Merlin generating programs, then we can replace giving a fixed setting of random bits after position i, by sampling over a polynomial number of possible settings of these bits. The main benefit of not including these bits is that now, as in the extractor setting, we can use weak designs instead of uniform weak designs, and by the first part of Lemma 11, use designs with the optimal parameter $\rho = 1$.

One difficulty we need to address is that the number of positive oracle calls to the oracle B from Section 4 depends on the sequence of $m-i$ random bits $c_{i+1}, c_{i+2}, \ldots, c_m$ chosen. In the proof of Theorem 2, we included that number in the description of elements from A because this allowed us to replace oracle calls to B by oracle calls to A. When Arthur randomly picks $s(n) = \mathrm{poly}(n)$ such sequences $r_1, r_2, \ldots, r_s$, we cannot include the number of positive oracle calls to B for every possible choice of r in the description. Instead, we include the average number of acceptances $\bar{a}$ over all possible values of r. With high probability, the total number of acceptances for the strings $r_1, \ldots, r_s$ will be within a bounded range of $s \cdot \bar{a}$. If the total number of acceptances for the strings $r_1, \ldots, r_s$ is indeed within this range, then Merlin will have limited leeway in his choice of demonstrating particular acceptances. Hence we can show that a nonnegligible fraction of $r_1, \ldots, r_s$ will give approximations to $\hat{u}$, or else we will catch Merlin cheating. The leeway Merlin has can lead to approximations of encodings $\hat{v}$ different from $\hat{u}$. However, only a small number of strings $\hat{v}$ can occur with probability comparable to that of $\hat{u}$ or better. We can thus specify $\hat{u}$ by distinguishing it from the other high likelihood encodings $\hat{v}$ with a small additional number of bits by the method of Theorem 1.

The technique of providing approximations to the average number of positive NP queries to limit Merlin's ability to cheat has been exploited before, e.g., in the context of random selfreducibility [5] and more recently in hardness-versus-randomness tradeoffs for nondeterministic circuits [17].

Proof: (of Theorem 3) We follow the proof of Theorem 2. Fix n and let $k = \log \|A^{=n}\|$. Because of the averaging argument, we will need to correct from more errors in the list decodable code and now take $\delta = 1/8m$. We will use Trevisan's function where the underlying set system $\mathcal{S}$ is a $(\ell, 1)$ weak design. Thus let $\mathrm{TR}_{\delta,\rho} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ be Trevisan's function with $m = k+d+1$. As the universe size d for weak designs depends on m, the equation $m = k + d + 1$ needs to be solved in terms of m. By Lemma 11, there are $(\ell, 1)$ weak designs with $d = O(\ell^2 \log m) = O(\log^3 n)$ and thus there is a solution to $m = k + d + 1$ with $m \le k + O(\log^3 n)$.

As in the previous proof, we let the set $B \subseteq \{0,1\}^m$ be the image of $A \times \{0,1\}^d$ under the function TR. By the choice of m, for any $u \in A^{=n}$,

$$\Pr[B(\mathrm{TR}(u, U_d))] - \Pr[B(U_m)] \ge 1/2.$$

By the hybrid argument, there is an $i \in [m]$, and a setting of the bits of y outside of S_i such that

$$\Pr_{\substack{x \in \{0,1\}^\ell, b \\ r \in \{0,1\}^{m-i}}} [B(\hat{u}_1(x) \cdots \hat{u}_{i-1}(x)\hat{u}(x)r)] - \Pr_{\substack{x \in \{0,1\}^\ell, b \\ r \in \{0,1\}^{m-i}}} [B(\hat{u}_1(x) \cdots \hat{u}_{i-1}(x)br)] \ge 1/2m. \tag{3}$$

For convenience in what follows, let $F(x, b, r) = \hat{u}_1(x) \cdots \hat{u}_{i-1}(x)br$.

Consider the following approach of approximating $\hat{u}$: On input x, pick a random $b \in \{0,1\}$ and $r \in \{0,1\}^{m-i}$ and compute $B(F(x, b, r))$; if this evaluates to 1, then output b, otherwise output $1 - b$. Let $g_b(x, r)$ be the function computing this operation. As in the argument after Equation (2), from Equation (3) it follows that $\Pr_{x,b,r}[\hat{u}(x) = g_b(x, r)] \ge 1/2 + 1/2m$. We set b to a value $b_1 \in \{0,1\}$ which preserves this prediction advantage. This value b_1 will be included as part of our description. Arthur cannot compute the function $g_{b_1}(x, r)$ himself as he needs Merlin to demonstrate witnesses for acceptance in B. We now show how to approximate the computation of $g_{b_1}(x, r)$ with an Arthur-Merlin protocol.

We say that r gives a α-approximation to $\hat{u}$ if $\Pr_x[g_{b_1}(x, r) = \hat{u}(x)] \ge \alpha$. For fixed r, we identify $g_{b_1}(x, r)$ with the string $z_{b_1,r}$ where $z_{b_1,r}$ has bit b_1 in position x if and only if $g_{b_1}(x, r) = 1$.

For convenience we assume without loss of generality that $b_1 = 1$ and drop the subscript. Note that with this choice the number of ones in z_r is the number of strings x for which B accepts $\hat{u}_1(x) \cdots \hat{u}_{i-1}(x) b_1 r$. With $w(z)$ we denote the number of ones in a string z.

Arthur randomly selects strings $r_1, \ldots, r_s$, each of length $m - i$, for a polynomial $s = s(n)$. Whereas in the proof of Theorem 2 we included in the description the number of acceptances by B for a particular setting of bits $c_{i+1}, \ldots, c_m$, we now include the average $\bar{a} = 2^{i-m} \sum_{x,r} g_{b_1}(x, r)$ number of acceptances over all $r \in \{0,1\}^{m-i}$. To limit Merlin's freedom in providing these acceptances, we want the number of acceptances by B over the strings $r_1, \ldots, r_s$ to be close to the expected $s \cdot \bar{a}$.

The next claim shows that with high probability the strings $r_1, \ldots, r_s$ will satisfy our requirements.

Claim 17 *For any $\gamma = \gamma(m, \bar{n}) > 0$, there exists $s = O(\bar{n}^2/\gamma^2)$ such that with probability at least $3/4$ over Arthur's choice of $r_1, \ldots, r_s$ the following two things will simultaneously happen:*

1. *A $1/8m$ fraction of $r_1, \ldots, r_s$ will give $\frac{1}{2} + \frac{1}{4m}$ approximations to $\hat{u}$.*
2. *The total number of acceptances by B over the strings $r_1, \ldots, r_s$ will be within γs of the expected. That is,*

$$|\sum_{j=1}^{s} w(z_j) - s\bar{a}| \leq \gamma s.$$

Proof: To lower bound the probability that both of these events happen, we upper bound the probability that each event individually does not happen and use a union bound.
Item (1): Notice that for a given r, if

$$\Pr_{x,b}[B(\hat{u}_1(x) \cdots \hat{u}_{i-1}(x)\hat{u}(x)r)] - \Pr_{x,b}[B(\hat{u}_1(x) \cdots \hat{u}_{i-1}(x)br)] \geq 1/4m$$

then r gives a $\frac{1}{2} + \frac{1}{4m}$-approximation of $\hat{u}$. We will say that r is bad if it does not yield a $\frac{1}{2} + \frac{1}{4m}$ approximation to $\hat{u}$. By Equation (3) and Markov's inequality,

$$\Pr_r[r \in \text{bad}] \leq \frac{1 - 1/2m}{1 - 1/4m} < 1 - 1/4m.$$

By a Chernoff bound, for some constant $c_1 > 0$,

$$\Pr_{r_1,\ldots,r_s}[\|\text{bad}\| \geq (1 - 1/8m)s] \leq \exp(-c_1 s/m^2).$$

Item (2): By a Chernoff bound, for some constant $c_2 > 0$,

$$\Pr[|1/s \sum_{j=1}^{s} w(z_j) - \bar{a}| \geq \gamma] \leq 2\exp(-c_2\gamma^2 s/\bar{n}^2).$$

By taking $s = c_3\bar{n}^2/\gamma^2$ for a sufficiently large constant c_3, the probability of each item will be less than $1/8$, and the claim follows. □

After choosing the strings $r_1, \ldots, r_s$, Arthur requests Merlin to provide $s\bar{a} - s\gamma$ many witnesses for acceptances in B. Arthur verifies these witnesses and rejects if any of them fail. From the acceptances provided by Merlin, Arthur constructs the strings $z'_{r_1}, \ldots, z'_{r_s}$, where position x of the string z_j has a one if and only if Merlin provided a witness for $B(F(x, b_1, r_j)) = 1$. We now show that, with high probability, no matter which acceptances Merlin chooses to demonstrate, at least a $1/16m$ fraction of $z'_{r_1}, \ldots, z'_{r_s}$ will give $\frac{1}{2} + \frac{1}{8m}$ approximations of $\hat{u}$.

Claim 18 *If $r_1, \ldots, r_s$ satisfy the two conditions of the previous claim with $\gamma = \bar{n}/256m^2$, then for any demonstration of acceptances by Merlin at least a $1/16m$ fraction of $z'_{r_1}, \ldots, z'_{r_s}$ will be $\frac{1}{2} + \frac{1}{8m}$ approximations to $\hat{u}$.*

Proof: By assumption, the number of acceptances for the strings $r_1, \ldots, r_s$ is between $s\bar{a} - s\gamma$ and $s\bar{a} + s\gamma$. Since Merlin has to provide witnesses for $s\bar{a} - s\gamma$ acceptances and can never fool Arthur in providing an invalid witness, Merlin has at most $2s\gamma$ acceptances to play with. Consider them as Merlin's potential to fool Arthur.

How can z_{r_j} and z'_{r_j} differ? As Arthur verifies the witnesses provided by Merlin, wherever z'_{r_j} has a one, z_{r_j} must also have a one. Thus, if z_{r_j} and z'_{r_j} differ in t positions, then Merlin has to spend at least t units of his potential on r_j. Since Merlin's total potential is bounded by $2s\gamma$, we have that the number of r_j's such that z_{r_j} and z'_{r_j} differ in t or more positions is bounded by $2s\gamma/t$.

Under the conditions of the claim, a $1/8m$ fraction of the z_{r_j} are $\frac{1}{2} + \frac{1}{4m}$ approximations of $\hat{u}$. Setting $t = \bar{n}/8m$ and $\gamma = \bar{n}/256m^2$, we have that a fraction a least $\frac{1}{8m} - \frac{2\gamma}{t} = \frac{1}{16m}$ of the z'_{r_j}'s are approximations that agree with $\hat{u}$ in a fraction at least $\frac{1}{2} + \frac{1}{4m} - \frac{1}{8m} = \frac{1}{2} + \frac{1}{8m}$ of the positions. □

Now putting these claims together, and taking s to be a sufficiently large polynomial, say $s = \omega(m^4)$, we have that with probability $3/4$ over Arthur's choice of $r_1, \ldots, r_s$, at least a $1/16m$ fraction of these settings will give $\frac{1}{2} + \frac{1}{8m}$ approximations to $\hat{u}$. A particular r_j can give a $\frac{1}{2} + \frac{1}{8m}$ approximation to at most the number of codewords that agree with it on a fraction at least $\frac{1}{2} + \frac{1}{8m}$ of the positions. By Lemma 12, this number is bounded by a polynomial $q(m)$.

Let us say that $\hat{v} \in$ LIKELY if at least a $1/32m$ fraction of r give a $\frac{1}{2} + \frac{1}{8m}$ approximation of $\hat{v}$. Note that the size of the set LIKELY is at most $32mq$. By Theorem 1, there is a distinguishing program p_1 of length $2\log(32mq)$ such that $p_1(\hat{u})$ accepts and $p_1(\hat{v})$ rejects for any $\hat{u} \neq \hat{v} \in$ LIKELY.

We make a list of all codewords $\hat{v}$ which agree with any of $z'_{r_1}, \ldots, z'_{r_s}$ on at least a $\frac{1}{2} + \frac{1}{8m}$ fraction of positions. We then remove all elements of this list which occur fewer than $s/16m$ times. With probability more than $2/3$, $\hat{u}$ is on this list and all elements $\hat{v}$ on the list are in LIKELY. In that case, from the elements on the list, the distinguishing program p_1 will accept $\hat{u}$ and $\hat{u}$ only. As the list is explicit, the distinguishing program p_1 does not need to make any oracle calls.

To carry out the above procedure, we need the following information:

1. the index i, the bit b_1, the average number of acceptances $\bar{a}$ to high enough precision, and the distinguishing program p_1, and

2. a description of the functions $\hat{u}_1, \ldots, \hat{u}_{i-1}$.

Note that $O(\log n)$ bits of precision is enough to encode $\bar{a}$. Thus, the first item costs $O(\log n)$ bits. As we took $\mathcal{S}$ to be a $(\ell, 1)$ weak design, the second item costs less than $m = \log \|A^{=n}\| + O(\log^3 n)$ bits.

With probability more than $2/3$, Merlin can make Arthur accept, and whenever Arthur accepts he produces u as output. Moreover, Arthur only queries the oracle A on strings of length $n = |u|$, and rejects whenever an oracle query is answered negatively. □

The positive use of the oracle in Theorem 3 implies the following corollary on the CAM complexity of strings from a language in AM.

Corollary 19 *For any set $A \in \mathrm{AM}$ there is a polynomial $p(n)$ such that for all $x \in A^{=n}$,*

$$\mathrm{CAM}^p(x) \leq \log \|A^{=n}\| + O(\log^3 n).$$

Proof: Consider the Arthur-Merlin process from Theorem 3 in which Arthur makes queries to the oracle A. Since Arthur rejects whenever the oracle responds negatively, we can make all oracle queries in parallel and simulate them without oracle access by running the Arthur-Merlin game that defines A; we boost the confidence of the latter game by standard parallel repetition and majority voting, and reject whenever a majority vote rejects. The resulting process can be viewed as an AMAM game, which can be transformed into an equivalent AM game using standard techniques. This gives us the Arthur-Merlin process we need; its description length is only $O(\log n)$ longer than the one given in Theorem 3. □

6 Language Compression By Randomness Only

We have shown the power of nondeterminism in decompression algorithms, and also the benefit of randomness in conjunction with nondeterminism in further reducing description size. We now address the case of decompression algorithms that use randomness alone. For this case, we establish some negative results.

First, we argue that randomness barely helps to efficiently generate a string from a short description. In fact, the following result proves that there are sets A of size 2^k such that *no* string in A can be generated with probability at least $2/3$ by an efficient randomized program of size a bit less than $n - k$. Recall that achieving the information-theoretic bound would require programs of size k.

Proof: (of Theorem 4) We will argue that there are many strings x of length n that (i) are not generated with high probability by a randomized program p of small size with access to the empty oracle and (ii) have a small probability of being queried by any program p of small size that runs in time t and has access to the empty oracle. Putting 2^k such strings in the oracle A does not affect the output distribution of any of these programs p by much, so they still cannot generate any of the strings x we put in A.

Let's call a randomized program p small if its length is less than some integer ℓ which we'll determine later. Let B_i denote the set of inputs x of length n for which there exists a small program p that outputs x with probability at least $1/2$ on the empty oracle. Since every program can induce at most two elements in B_i and there are less than 2^ℓ small programs, we have that $\|B_i\| \leq 2^{\ell+1}$.

Consider the set of strings y such that p queries y with probability at least 2^{-s} on the empty oracle, where s is another integer we'll set later. If p runs in time t, the size of this set is bounded by $2^s t$. Let B_q denote the set of all queries y of length n that are asked with probability at least 2^{-s} by at least one small program p on the empty oracle. We have that $\|B_q\| \leq 2^{\ell+s} t$.

Let A be a set of 2^k strings of length n that are neither in B_i nor in B_q. Such a set exists provided

$$2^k \leq 2^n - 2^{\ell+s+1}t. \tag{4}$$

Now, consider any small program p with access to oracle A. Since A does not contain any string in B_q, the probability that p outputs something different on the empty oracle and on oracle A is no more than 2^{k-s}. Thus, for any string x outside of B_i, the probability that p outputs x on oracle A is less than $\frac{1}{2} + 2^{k-s}$. Setting $s = k + \log 6$ and using the fact that every string in A is outside of B_i, we have that no string in A can be generated by p with probability at least $\frac{1}{2} + \frac{1}{6} = \frac{2}{3}$ on oracle A. Setting $\ell = n - k - \log t - 5$ satisfies (4), and thereby finishes the proof. □

In the absence of nondeterminism, the distinction between generating programs and distinguishing programs becomes relevant. Indeed, Theorem 1 implies that randomized distinguishing programs can do much better than the randomized generating programs from Theorem 4: We can realize an upper bound of roughly $2 \log \|A^{=n}\|$ in the case of distinguishing programs, even for deterministic ones. [3] proved that the factor of 2 is tight in the deterministic setting. We now extend that result to the randomized setting, i.e., we exhibit a set A that contains an exponential number 2^k of strings of length n such that at least one of these strings cannot be distinguished from the other strings in A by a randomized program of length a little bit less than $2k$ with oracle access to A.

As in [3], the core of the argument is a combinatorial result on cover free set systems. A family $\mathcal{F}$ of sets is called K-cover free if for any different sets $F_0, \ldots, F_k \in \mathcal{F}$, $F_0 \not\subseteq \cup_{j=1}^{K} F_j$. The combinatorial result we use states that K-cover free families of more than K^3 sets need a universe of at least K^2 elements.

Lemma 20 ([4]) *If $\mathcal{F}$ is a K-cover free family containing M sets over a universe of L element universe, and $M > K^3$ then $L \geq \frac{K^2 \log M}{2 \log K + c}$ for some constant c.*

The connection between distinguishing programs and cover free families is the following. Recall that for a given string x and oracle A, a randomized distinguishing program accepts x with probability at least $2/3$ on oracle A, and rejects every other string with probability at least $2/3$ on oracle A. Let F_x^A denote the set of randomized programs of length less than ℓ that accept x with probability more than $1/2$ on oracle A. If every string in A has a randomized distinguishing program of size less than ℓ on oracle A, then the family $\{F_x^A \mid x \in A^{=n}\}$ is K-cover free for $K = \|A^{=n}\| - 1$.

The size of this family is only $M = K + 1$. In order to obtain a larger family, we argue that if all strings in A are of length n and Kolmogorov random with respect to the other strings in A, then no short efficient program p on input $x \in A$ has a noticeable probability of querying a string in A other than x. Thus, $p^A(x)$ and $p^{\{x\}}(x)$ behave essentially the same. Notice that $p^{\{x\}}(x)$ does not depend on A. This allows us to consider a larger set B containing $M > K^3$ strings x of length n that are Kolmogorov random with respect to the other strings in B. Assuming every subset A of B of size $2^k = K - 1$ has an efficient randomized distinguishing program of size less than ℓ when given oracle access to A, we have that the family

$$\mathcal{F} = \{F_x^{\{x\}} \mid x \in B\} \tag{5}$$

is a K-cover free family of size $M > K^3$. Lemma 20 then implies that $\ell \geq 2k - O(1)$.

We now fill in the details of the proof.

Proof: (of Theorem 5) Let z be a string of length Mn such that $C(z) \geq |z|$, where $M = 2^m$ will be determined later. Let B consist of the strings of length n obtained by chopping up z into M pieces of equal size. All M strings are guaranteed to be different as long as $m \leq n/2 - O(\log n)$; otherwise, we could obtain a short description of z by describing one of its length n segments as a copy of another one.

A key observation is the following:

Claim 21 *For every subset A of B, every $x \in A$, and every randomized program p of length less than ℓ running in time t,*

$$|\Pr[p^A(x) \text{ accepts}] - \Pr[p^{\{x\}}(x) \text{ accepts}]| < 1/6,$$

provided $n > \ell + c(m + \log(t+n))$, where c is some universal constant.

Proof: We will argue that every random bit sequence that leads to a different outcome for $p^A(x)$ and $p^{\{x\}}(x)$, has a short description with respect to z. Since there can only be few random bit sequences with a short description, this implies the claim.

Let us denote the outcome of p on input x, oracle O, and random bit sequence $r \in \{0,1\}^t$ by $p^O(x,r)$. If $p^A(x,r) \neq p^{\{x\}}(x,r)$, then $p^{\{x\}}(x,r)$ must query some string $y \in A$. We can describe this y with p, x, r, and an index of size $\log t$ indicating the time when the query takes place. By adding the remaining parts of z, the indices of x and y in z, and making everything prefix free, we obtain a description of $\langle z, r\rangle$. This shows

$$C(\langle z,r\rangle) \leq |z| + |r| - n + \ell + 2m + O(\log(t+n)).$$

Symmetry of information tells us that

$$C(\langle z,r\rangle) \geq C(z) + C(r|z) - O(m + \log(t+n)).$$

Since $C(z) = |z|$, we conclude that

$$C(r|z) \leq |r| - n + \ell + O(m + \log(t+n)).$$

We can make the fraction of random bit strings r that have such a short description less than $1/6$ by choosing $n > \ell + c(m + \log(t+n))$ for some sufficiently large constant c. The claim follows. □

Now, suppose that for every subset A of B of size 2^k, every string $x \in A$ satisfies $\mathrm{CBPD}^{t,A}(x) < \ell$, where k, t, and ℓ are some integers. Then the family $\mathcal{F}$ defined by (5) is K-cover free for $K = 2^k - 1$. Indeed, consider any subset A of B containing the 2^k different strings $x_0, x_1, \ldots, x_K$ from B. Let p be a randomized program of length less than ℓ that runs in time t, such that $p^A(x_0)$ accepts with probability at least $2/3$, and $p^A(x_i)$ rejects with probability at least $2/3$ for $1 \leq i \leq K$. Claim 21 implies that $p \in F_{x_0}^{\{x_0\}}$ and $p \notin F_{x_i}^{\{x_i\}}$ for any $1 \leq i \leq K$. Thus, $F_{x_0}^{\{x_0\}}$ is not covered by the union of the K sets $F_{x_i}^{\{x_i\}}$, $1 \leq i \leq K$.

Since the family $\mathcal{F}$ is of size $M = 2^m$, Lemma 20 implies that $\ell \geq 2k - c_3$ for some constant c_3, provided $M > K^3$. All size conditions can be met for values of k up to $c_1 n - c_2 \log t$ for some positive constants c_1 and c_2. □

Recall that [3] established the same lower bound as in Theorem 5 for CD complexity instead of CBPD complexity. They also extended their result to CD complexity with access to an oracle in NP $\cap$ coNP. Similar to the formulation of Theorem 5, their extension can be phrased as follows: For every robust (NP $\cap$ coNP) machine M, there exist constants c_1, c_2, and c_3 such that for any integers n, k, and t satisfying $k \leq c_1 n - c_2 \log t$, there exists a set A with $\log \|A^{=n}\| = k$ and a string $x \in A$ such that

$$\mathrm{CD}^{t,M^A}(x) \geq 2 \log \|A^{=n}\| - c_3.$$

The robustness condition is implicit in the proof in [3]. By a robust (NP $\cap$ coNP) machine M, we mean an oracle machine M such that for every oracle B, M^B behaves like an (NP$\cap$coNP) machine. Note, though, that Theorem 2 implies the existence of a promise-(NP $\cap$ coNP) machine M and a polynomial p such that for any set A and every $x \in A$,

$$\mathrm{CD}^{p,M^A}(x) \leq \log \|A^{=n}\| + O(\delta(n)),$$

where $\delta(n) = (\sqrt{\log \|A^{=n}\|} + \log n) \log n$.

In a similar way, we can extend Theorem 5 as follows: For every robust (AM$\cap$coAM) machine M, there exist constants c_1, c_2, and c_3 such that for any integers n, k, and t satisfying $k \leq c_1 n - c_2 \log t$, there exists a set A with $\log \|A^{=n}\| = k$ and a string $x \in A$ such that

$$\mathrm{CD}^{t,M^A}(x) \geq 2 \log \|A^{=n}\| - c_3.$$

However, without the robustness requirement, Theorem 3 implies the existence of a promise-(AM$\cap$ coAM) machine M and a polynomial p such that for any set A and every $x \in A$,

$$\mathrm{CD}^{p,M^A}(x) \leq \log \|A^{=n}\| + O(\log^3 n).$$

Acknowledgments

We would like to thank Lance Fortnow for helpful discussions and Andrei Romashchenko for beneficial comments on an earlier version of the paper.

References

[1] J. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I*, volume 11 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, 1995.

[2] H. Buhrman, L. Fortnow, and S. Laplante. Resource bounded Kolmogorov complexity revisited. *SIAM Journal on Computing*, 31(3):887–905, 2002.

[3] H. Buhrman, S. Laplante, and P. Bro Miltersen. New bounds for the language compression problem. In *Proceedings of the 15th IEEE Conference on Computational Complexity*, pages 126–130. IEEE, 2000.

[4] A. G. Dyachkov and V. V. Rykov. Bounds on the length of disjunctive codes. *Problemy Peredachi Informatsii*, 18:7–13, 1982. In Russian.

[5] J. Feigenbaum and L. Fortnow. On the random-self-reducibility of complete sets. *SIAM Journal on Computing*, 22:994–1005, 1993.

[6] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28:270–299, 1984.

[7] F. Hennie and R. Stearns. Two-tape simulation of multitape Turing machines. *Journal of the ACM*, 13:533–546, 1966.

[8] A. Klivans and D. van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial hierarchy collapses. *SIAM Journal on Computing*, 31(5):1501–1526, 2002.

[9] R. Kumar and D. Sivakumar. Proofs, codes, and polynomial-time reducibilities. In *Proceedings of the 14th IEEE Conference on Computational Complexity*, pages 46–53. IEEE, 1999.

[10] T. Lee and A. Romashchenko. On polynomially time bounded symmetry of information. In J. Fiala, V. Koubek, and J. Kratochvil, editors, *29th International Symposium on the Mathematical Foundations of Computer Science*, volume 3153 of *Lecture Notes in Computer Science*, pages 463–475. Springer-Verlag, 2004.

[11] M. Li and P. Vitányi. *An Introduction to Kolmogorov Complexity and its Applications*. Springer-Verlag, New York, second edition, 1997.

[12] N. Nisan and A. Wigderson. Hardness vs. randomness. *Journal of Computer and System Sciences*, 49:149–167, 1994.

[13] C. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[14] R. Raz, O. Reingold, and S. Vadhan. Extracting all the randomness and reducing the error in Trevisan's extractors. *Journal of Computer and System Sciences*, 65(1):97–128, 2002.

[15] A. Selman. Much ado about functions. In *Proceedings of the 11th IEEE Conference on Computational Complexity*, pages 198–212. IEEE, 1996.

[16] R. Shaltiel. Recent developments in explicit construction of extractors. *Bulletin of the European Association for Theoretical Computer Science*, 77:67–95, 2002.

[17] R. Shaltiel and C. Umans. Simple extractors for all min-entropies and a new pseudo-random generator. In *Proceedings of the 42st IEEE Symposium on Foundations of Computer Science*, pages 648–657. IEEE, 2001.

[18] M. Sipser. A complexity theoretic approach to randomness. In *Proceedings of the 15th ACM Symposium on the Theory of Computing*, pages 330–335. ACM, 1983.

[19] M. Sudan. Decoding of Reed Solomon codes beyond the error-correction bound. *Journal of Complexity*, 13(1):180–193, 1997.

[20] M. Sudan, L. Trevisan, and S. Vadhan. Pseudorandom generators without the XOR lemma. *Journal of Computer and System Sciences*, 62(2):236–266, 2001.

[21] A. Ta-Shma, C. Umans, and D. Zuckerman. Loss-less condensers, unbalanced expanders, and extractors. In *Proceedings of the 33rd ACM Symposium on the Theory of Computing*, pages 143–152. ACM, 2001.

[22] A. Ta-Shma, D. Zuckerman, and S. Safra. Extractors from Reed-Muller codes. In *Proceedings of the 42st IEEE Symposium on Foundations of Computer Science*, pages 638–647. IEEE, 2001.

[23] L. Trevisan. Construction of extractors using pseudo-random generators. *Journal of the ACM*, 48(4):860–879, 2001.

[24] C. Umans. Pseudo-random generators for all hardnesses. *Journal of Computer and System Sciences*, 67:419–440, 2003.