Compression of Samplable Sources

Luca Trevisan[*]
Computer Science Division
U.C. Berkeley
615 Soda Hall
Berkeley, CA 94720
luca@cs.berkeley.edu

Salil Vadhan[†]
Division of Engineering & Applied Sciences
Harvard University
33 Oxford Street
Cambridge, MA 02138
salil@eecs.harvard.edu

David Zuckerman [‡]
Department of Computer Science
University of Texas
1 University Station C0500
Austin, TX 78712
diz@cs.utexas.edu

Abstract

We study the compression of polynomially samplable sources. In particular, we give efficient prefix-free compression and decompression algorithms for three classes of such sources (whose support is a subset of $\{0,1\}^n$).

1. *We show how to compress sources X samplable by logspace machines to expected length $H(X)+O(1)$.*

Our next results concern flat sources whose support is in $\mathbf{P}$.

2. *If $H(X) \leq k = n - O(\log n)$, we show how to compress to length $k + \delta \cdot (n-k)$ for any constant $\delta > 0$; in quasi-polynomial time we show how to compress to length $k + O(\mathrm{polylog}\log(n-k))$ even if $k = n - \mathrm{polylog}(n)$.*
3. *If the support of X is the witness set for a self-reducible $\mathbf{NP}$ relation, then we show how to compress to expected length $H(X)+4$.*

∗ Supported by NSF Grant CCR-9984783, a Sloan Research Fellowship and an Okawa Foundation Grant.

† Supported by NSF Grant CCR-0133096 and a Sloan Research Fellowship.

‡ Supported in part by NSF Grants CCR-9912428 and CCR-0310960 and a David and Lucile Packard Fellowship for Science and Engineering.

1. Introduction

Data compression has been studied extensively in the information theory literature (see e.g. [7] for an introduction). In this literature, the goal is to compress a random variable X, which is called a random source. Non-explicitly, the entropy $H(X)$ is both an upper and lower bound on the expected size of the compression (to within an additive $\log n$ term). For explicit (i.e. polynomial-time) compression and decompression algorithms, this bound cannot be achieved for general sources. Thus, existing efficient data-compression algorithms have been shown to approach optimal compression for sources X satisfying various stochastic "niceness" conditions, such as being stationary and ergodic, or Markovian. In this paper, we focus on the feasibility of data compression for sources satisfying *computational* niceness conditions, most notably efficient samplability. Goldberg and Sipser [9] were the first to study compression of sources satisfying computational, rather than stochastic, conditions. Actually, they did not explicitly discuss random sources, but focused on compressing *languages* in $\mathbf{P}$, and thus implicitly considering sources uniformly distributed on all n-bit strings in such a language.

We extend and generalize their study. We focus on sources which are polynomially *samplable*, i.e. can be generated by a probabilistic polynomial-time algorithm. Samplability captures a very general class of sources, and it is arguably a reasonable model for probability distributions generated by various natural and man-made processes. When a distribution is not samplable, the problem of generating the distribution is computationally intractable, and this seems unlikely for "natural" sources.

The languages corresponding to the supports of samplable sources need not be in $\mathbf{P}$. Indeed, while Goldberg & Sipser show that every sparse language in $\mathbf{P}$ can be compressed at least slightly, this is unlikely to be true for all polynomially samplable distributions.[1] Therefore, while seeking classes of samplable distributions that can be optimally compressed, we need to impose computational constraints that rule out the possibility of sampling pseudorandom distributions.

Logspace Samplers

We first study sources that can be sampled by a sampling algorithm that uses *logarithmic space*. (As is usual when studying randomized logspace, we only allow the algorithm one-way access to the random tape.) Such sources generalize Markovian sources (that can be thought of as being sampled by a constant-space sampling algorithm). On the other hand, it is known that no such source can be pseudorandom [17].

We show the existence of a universal compression algorithm for such sources that compresses optimally, up to an additive constant factor, in polynomial time. The compression algorithm is universal in the sense that it optimally compresses a source X without being given a sampler for X, and just knowing the existence of a sampling algorithm and an upper bound to the space used by the sampler. If the sampler is known, we use *arithmetic encoding*, a well known optimal compression method that can be used on any source for which it is possible to compute the *cumulative probability* distribution of the source. Our result is then obtained by giving an algorithm for computing cumulative probabilities for sources sampled by logarithmic space algorithms.

We also prove a general result showing that if optimal compression is possible for a class of samplable distributions *given the sampler* then, with only an additive constant loss, optimal compression is also possible without being given the sampler, i.e. it is possible to do *universal compression* for this class. Applying this to the result above, we obtain a universal compression algorithm for sources samplable in space $c \log n$ for any constant c.

Sources with Membership Algorithms

We next consider samplable sources for which membership in the support can be tested in polynomial time. Without further restrictions, a membership algorithm may not be useful; for example, the support of the source could be $\{0,1\}^n$ but some strings occur with tiny probability. We therefore require that the source be flat, i.e., uniform on its support. Observe that a membership algorithm rules out the possibility that such a distribution is pseudorandom. Indeed, the membership algorithm gives a way to distinguish the source from any other source of higher entropy.

The case of flat distributions with membership algorithms was studied by Goldberg and Sipser [9] who showed that every such source X on $\{0,1\}^n$ could be compressed to $k+3\log n$ bits provided that the entropy of X is smaller than $k = n - O(\log n)$. We show how to improve the compression length to $k+\delta\cdot(n-k)$ for an arbitrarily small constant $\delta > 0$, which saves more than a constant factor in overhead if $k = n - o(\log n)$. While Goldberg and Sipser use arithmetic encoding, we use a completely different method relying on recent constructions of expander graphs with expansion close to the degree [5]. In addition, our compression algorithm is deterministic, whereas the Goldberg–Sipser algorithm is probabilistic.

In our last main result, we show that if the support of the samplable distribution forms the witness set for a *self-reducible* $\mathbf{NP}$ relation, then we can compress almost optimally. As a consequence, we obtain polynomial-time compression algorithms for a wide variety of combinatorial structures for which sampling algorithms are known, e.g. the set of perfect matchings in a bipartite graph [15]. Our compression algorithm

1 In particular, as first observed by Levin, pseudorandom distributions are incompressible and, if pseudorandom generators exist, then there are polynomially samplable pseudorandom distributions. See Section 3 for more details.

computes an “approximate” arithmetic coding, using ideas underlying the proof that sampling implies approximate counting for self-reducible relations [16]. In fact, we show that, for self-reducible relations, near-optimal compression is *equivalent* to almost-uniform sampling (which in turn is known to be equivalent to approximate counting [16, 14]).

Perspective and Open Problems

There are a number of examples where the imposition of complexity-theoretic constraints on traditionally information-theoretic problems has been very fruitful. For example, modern cryptography developed and flourished out of the realization that Shannon’s classic impossibility results [23] could be bypassed via the reasonable assumption that the adversary is computationally bounded [8].

Our restriction to *samplable* sources in particular was motivated by our work in [26], where we consider the somewhat related problem of (deterministic) random extraction, in which one is given a source of a certain entropy and wants to devise an algorithm that given a sample from the source outputs an almost uniform distribution. This deterministic randomness extraction problem was known to be impossible for general sources [21, 6], and it was known to be possible for very structured sources like Markovian sources (just like the data compression problem). In [26], it is shown that, under certain complexity assumptions, randomness extraction is possible for samplable sources. Another, earlier, work showing the promise of restricting to samplable sources is that of Lipton [18], who showed that if the distribution of errors in a channel is samplable, then it is possible to transmit information reliably even above the capacity bound.

As noted above, for data compression, the class of samplable sources is still too general, and thus we have tried to impose sensible additional restrictions that are still computational in nature, yet allow for interesting positive results. However, we have by no means exhausted the possibilities, and there may be other computational constraints that are even more relevant for data compression.

Another motivation for this line of work comes from the general project of understanding information-theoretic aspects of samplable sources. The theory of pseudorandom generators is naturally one major piece of this study. But samplable sources and their information-theoretic properties have also come up in unexpected places, such as in the complete problems for statistical zero knowledge [20, 10]. Understanding the compressibility of samplable sources can contribute to this general study, as it provides another measure of the (computational) randomness in a source. This compressibility measure was introduced by Yao [28], and properties of this measure have been recently studied by Barak and others [4].

A few years ago, Impagliazzo [12] posed an intriguing question about the relationship between compressibility and another standard measure of computational randomness, pseudoentropy. A source has *pseudoentropy* at least k if it is computationally indistinguishable from some distribution having entropy at least k. A source of pseudoentropy k cannot be compressed to $k - \omega(\log n)$ by an efficient algorithm, and the question is whether the converse is true for samplable distributions. That is, does low pseudoentropy imply compressibility for samplable sources? This intriguing question is still an open problem. However, Wee [27] has exhibited an oracle relative to which the answer is no. Specifically, under this oracle there are samplable distributions over $\{0,1\}^n$ of very low entropy and pseudoentropy that cannot be compressed to less than $n - O(\log n)$ bits. It would be very interesting to obtain a similar result without oracles under complexity-theoretic assumptions.

2. Preliminaries

2.1. Basic definitions

A *source* X is a probability distribution on strings of some length. We write $x \overset{\mathrm{R}}{\leftarrow} X$ to indicate that x is chosen randomly according to X. We think of X as being a member of a family of distributions (i.e., a probability *ensemble*), in order for asymptotic notions to make sense. The ensemble will usually be of the form $(X_n)_{n\in\mathbb{Z}^+}$, in which case X_n will be distributed on $\{0,1\}^n$.[2] Sometimes we will consider ensem-

2 Note that this differs from the notation used in classical information theory, where one writes X_i for an individual symbol of an infinite stochastic process $X_1, X_2, \ldots$ and is concerned with compressing a prefix $(X_1, X_2, \ldots, X_n)$ of this process.

bles $(X_x)_{x\in L}$ indexed by strings in some language $L \subseteq \{0,1\}^+$, in which case X_x will be distributed over $\{0,1\}^{p(|x|)}$ for some polynomial p. Here $\Sigma^+ = \Sigma\Sigma^*$ is the set of strings over Σ, excluding the empty string.

We denote $X(a) = \Pr[X = a]$. The *support* of X is $\mathrm{Sup}(X) = \{x|X(x) > 0\}$. A *flat source* is uniform on its support. U_n is the uniform distribution on $\{0,1\}^n$.

Definition 2.1. *The entropy of a distribution X is $H(X) = \mathrm{E}_{x \overset{R}{\leftarrow} X}\left[\log\left(\frac{1}{X(x)}\right)\right]$.*

Here, and throughout the paper, all logs are to base 2.

2.2. Basics of compression

Definition 2.2. *For functions $\mathrm{Enc} : \Sigma^+ \to \Sigma^+$ and $\mathrm{Dec} : \Sigma^+ \to \Sigma^+$, we say $(\mathrm{Enc}, \mathrm{Dec})$* compresses *source X to* length *m if*

1. *For all $x \in \mathrm{Sup}(X)$, $\mathrm{Dec}(\mathrm{Enc}(x)) = x$, and*
2. $\mathrm{E}[|\mathrm{Enc}(X)|] \le m$.

We say that the encoding is prefix-free *if for all $x \neq y$ in $\mathrm{Sup}(X)$, $\mathrm{Enc}(x)$ is not a prefix of $\mathrm{Enc}(y)$.*

All of our codes will be prefix-free. It is well known that a prefix-free encoding is "uniquely decodable"; that is, commas are not needed to send multiple samples of X.

Definition 2.3. *We say source X is* compressible *to length m if there exists functions Enc and Dec such that $(\mathrm{Enc}, \mathrm{Dec})$ compresses X to length m.*

It is well known that a source X is compressible to length $H(X) + 1$ by a prefix-free encoding (see e.g. [7]). If the encoding is required to be uniquely decodable, then X is not compressible to length less than $H(X)$. Although the codes we construct are uniquely decodable, the above definitions are less restrictive (often called "nonsingular" compression) and allow some random variables X to be compressed to length less than $H(X)$. The biggest gap is obtained by the distribution X_n which chooses i uniformly from 0 to n-1 and y uniformly from $\{0,1\}^{n-i-1}$ and outputs $0^i 1 y$. The compressed string is y, which has expected length $H(X_n) - \log n$. We assume the following is known but we do not know a reference.

Lemma 2.1. *A source X_n is not compressible to length less than $H(X_n) - \log H(X_n) - 1$*

Proof. Convert any encoding Enc to a prefix-free encoding Enc' defined by $\mathrm{Enc}'(x) = \ell(x)\mathrm{Enc}(x)$, where $\ell(x) = |\mathrm{Enc}(x)|$ written in binary with leading 0's. Then Enc' is prefix free, and hence uniquely decodable. The new compression length is

$$\begin{aligned} & \mathrm{E}[|\mathrm{Enc}'(X_n)|] \\ \le\ & \mathrm{E}[|\mathrm{Enc}(X_n)| + \log|\mathrm{Enc}(X_n)| + 1] \\ \le\ & \mathrm{E}[|\mathrm{Enc}(X_n)|] + \log \mathrm{E}[|\mathrm{Enc}(X_n)|] + 1\ . \end{aligned}$$

where the last inequality is by Jensen's inequality. By the lower bound for uniquely decodable codes, $\mathrm{E}[|\mathrm{Enc}'(X_n)|] \ge H(X_n)$. Setting $L = \mathrm{E}[|\mathrm{Enc}(X_n)|]$, we have $L + \log L \ge H(X_n) - 1$, which implies that $L \ge H(X_n) - \log(X_n) - 1$, as desired. □

Since tight non-explicit bounds are known, the interesting issue is *efficient* compressibility, e.g. when Enc and Dec are computed by polynomial-time algorithms. Indeed, much of the field of Data Compression is centered around understanding when this is possible. In order for efficient compressibility to make sense, we must specify how the source is presented. Ideally, the compression algorithm is only given a random sample from the source, and does not have any global information about the source other than the fact that it comes from some class of sources:

Definition 2.4 (universal compression). *Let $\mathcal{C}$ be a class of sources (i.e. class of probability ensembles X_n), and let $m = m(h,n)$ be a function. We say that $(\mathrm{Enc}, \mathrm{Dec})$ is a* universal compression *algorithm for $\mathcal{C}$ with compression length m if for every source X_n in $\mathcal{C}$, there is a constant c such that $(\mathrm{Enc}, \mathrm{Dec})$ compresses X_n to length $m(H(X_n), n) + c$.*

For example, the classic Lempel–Ziv method is a universal compression algorithm with compression length $H(X) + o(n)$ for the class of stationary ergodic processes [29]. (That is, the LZ method is guaranteed to effectively compress X_n if there is a stationary ergodic process $Y_1, Y_2, \ldots$ such that $X_n = (Y_1, \ldots, Y_n)$.)

Since universal compression is only known for a fairly restricted class of sources (and for those, only approaches the optimal compression length asymptotically), it is also of interest to study the case when the compression algorithm may depend on the entire source (rather than a single sample). That is, the compression algorithm is given a description d_n of the source X_n, and we require that $\mathrm{Dec}(\mathrm{Enc}(x, d_n), d_n) = x$ for all $x \in \mathrm{Sup}(X_n)$, and $\mathrm{E}[|\mathrm{Enc}(X_n, d_n)|] \leq m$. In other words, $\mathrm{Enc}'(\cdot) = \mathrm{Enc}(\cdot, d_n)$ and $\mathrm{Dec}'(\cdot) = \mathrm{Dec}(\cdot, d_n)$ should form a compression algorithm for X_n in the sense of Definition 2.2.

When the source is described *explicitly* (e.g. by the list of probability masses assigned to each string x), then standard methods, such as Huffman coding (cf. [7]) compress to length $H(X)+1$. But here the input size and the running time of the algorithm are both roughly 2^n. Thus, it is more interesting to consider the case when the source is described in some compact, *implicit* form. Then the question is which implicit representations allow for efficient compression.

One general technique for obtaining efficient compression is *arithmetic coding*, which is feasible if computing the cumulative distribution function is feasible.

Lemma 2.2 (arithmetic coding). *Let X be a source on Σ^n and $\prec$ a total order on $\mathrm{Sup}(X)$. Let $F : \Sigma^n \to [0,1]$ be the following modification of the cumulative distribution function of X: $F(x) = \sum_{a \prec x} X(a) + X(x)/2$. Define $\mathrm{Enc}(x)$ to be the first $\lceil \log(1/X(x)) \rceil + 1$ bits of $F(x)$. Then Enc is one-to-one and monotone, and $(\mathrm{Enc}, \mathrm{Enc}^{-1})$ compresses X to length $H(X) + 2$. The encoding is prefix-free.*

For example, if X is a Markovian source (i.e. the sequence of symbols of X form a Markov chain run for n steps), then it is known that the cumulative distribution function (with respect to the standard lexicographic order) can be computed in polynomial time, and hence so can the arithmetic coding. (See [7].) Note that since Enc is monotone, if Enc can be computed efficiently, then Enc^{-1} can also be computed efficiently by binary search. Several of our positive results will make use of arithmetic coding and variants.

Another useful fact is that it suffices to obtain a decoder which decodes correctly with high probability.

Lemma 2.3. *Suppose X_n is a source on $\{0,1\}^n$ and the time T computable functions Enc and Dec satisfy*

1. $\Pr[\mathrm{Dec}(\mathrm{Enc}(X_n)) = X_n] \geq 1 - \epsilon$, *and*
2. $\mathrm{E}[|\mathrm{Enc}(X_n)|] \leq m$.

Then there exist functions Enc' and Dec' that compress X_n to length $(1-\epsilon)m+\epsilon n+1$ in time $O(T)$. If Enc gives a prefix-free encoding, then so does Enc'.

For example, if X is close (in variation distance) to a source which is highly compressible, then X itself is highly compressible.

Proof of Lemma 2.3. We construct Enc' and Dec' such that for all $x \in \mathrm{Sup}(X_n)$, $\mathrm{Dec}'(\mathrm{Enc}'(x)) = x$. On input x, Enc' first checks if $\mathrm{Dec}(\mathrm{Enc}(x)) = x$. If so, Enc' outputs $0\mathrm{Enc}(x)$ (0 concatenated with $\mathrm{Enc}(x)$). If not, Enc' outputs $1x$. It is easy to see that Enc' and the natural Dec' are as required. $\square$

3. Samplable Sources

Classical results, such as those mentioned in the previous section, show that data compression is feasible for various classes of sources defined by statistical or information-theoretic constraints (e.g., stationary ergodic sources or Markovian sources). We propose to investigate classes of sources defined by *computational constraints*, specifically samplability:

Definition 3.1. *A source X_n is* samplable *if there is an efficient probabilistic algorithm S such that $S(1^n)$ is distributed according to X_n. "Efficient" can be taken to mean a polynomial-time algorithm, a logarithmic space algorithm, a uniform or nonuniform algorithm, or any other complexity constraint, and will be specified in context.*

For sources X_x indexed by strings, we instead require that $S(x)$ is distributed according to X_x.

It is natural to consider samplable sources, since any source X which is polynomially compressible to length $H(X)$, and moreover for all $x \in \mathrm{Sup}(X)$, $|\mathrm{Enc}(x)| = H(X)$, is polynomially samplable. This is because $X = \mathrm{Dec}(U_{H(X)})$. Goldberg and Sipser [9] also studied compression of computationally constrained sources, but they focused on the complexity of deciding membership in the support of the source (for flat sources).

We recall that pseudorandom generators yield samplable sources that are incompressible. (In [9], this observation is attributed to L. Levin.)

Proposition 3.1 (Levin). *If one-way functions exist, then there exist polynomial-time samplable sources X_n of entropy at most n^ϵ that cannot be* *compressed to length $n-3$ by any probabilistic polynomial-time algorithms* $(\mathrm{Enc}, \mathrm{Dec})$.

Proof. (sketch) If one-way functions exist, then there exists a pseudorandom generator $G : \{0,1\}^{n^\epsilon} \to \{0,1\}^n$ [11]. Let $X_n = G(U_{n^\epsilon})$. From the pseudorandom property of G, it follows that $\Pr[\mathrm{Dec}(\mathrm{Enc}(U_n)) = 1] \geq \Pr[\mathrm{Dec}(\mathrm{Enc}(X_n)) = 1] - \mathrm{neg}(n)$, where neg denotes a negligible function. The pseudorandom property of G also implies that $\mathrm{E}[|\mathrm{Enc}(U_n)|] \leq \mathrm{E}[|\mathrm{Enc}(X_n)|] + \mathrm{neg}(n) < n - 2.5$. That is, $(\mathrm{Enc}, \mathrm{Dec})$ compress a $1 - \mathrm{neg}(n)$ fraction of $\{0,1\}^n$ to average length $n - 2.5$. This is impossible by a counting argument. □

Thus, we cannot hope to efficiently compress samplable sources in full generality. Instead, we aim to identify natural subclasses of samplable sources for which compression is feasible. We will focus on the case when the compression algorithms are given the sampling algorithm. This is a natural implicit description of the source (like those discussed in Section 2.2). Moreover, efficient compression in this case implies universal compression (for uniform algorithms):

Lemma 3.1. *Let $\mathcal{S} \subseteq \Sigma^*$ be a class of sampling algorithms (encoded as strings) and $\mathcal{C}$ be the corresponding class of sources. Suppose that there exist algorithms* $(\mathrm{Enc}, \mathrm{Dec})$ *such that for every $S \in \mathcal{S}$,* $(\mathrm{Enc}(\cdot, S), \mathrm{Dec}(\cdot, S))$ *compresses $X_n = S(1^n)$ to length $m = m(H(X_n), n)$ in time* $\mathrm{poly}(n) \cdot f(|S|)$ *for some function f. Then there exists a polynomial-time universal compression algorithm* $(\mathrm{Enc}', \mathrm{Dec}')$ *for $\mathcal{C}$ that compresses to length $m + O(1)$. If each encoding* $\mathrm{Enc}(\cdot, S)$ *is prefix-free, then so is the encoding* Enc'.

Proof. Let $\circ$ denote concatenation, and $\mathrm{bin}_n(i)$ denote the binary representation of i, padded out with leading 0's to $1 + \lfloor \log n \rfloor$ bits. Let $\Sigma^* = \{S_1, S_2, S_3, \ldots\}$ be an enumeration of all strings in lexicorgraphic order. Let $p(n) \cdot f(|S|)$ be the running time of $(\mathrm{Enc}, \mathrm{Dec})$.

$\mathrm{Enc}'(x)$*, on input $x \in \{0,1\}^n$:*

```
1. For each $i = 1, \ldots, n$
   (a) Run $\mathrm{Enc}(x, S_i)$ for $p(n) \cdot n$ steps, and if
       it halts, let $y_i$ be the output.
   (b) Run $\mathrm{Dec}(y_i, S_i)$ for $p(n) \cdot n$ steps. If it
       outputs $x$, set $z_i = \mathrm{bin}_n(i) \circ y_i$.
   (c) If either $\mathrm{Enc}$ or $\mathrm{Dec}$ failed to halt within
       $p(n) \cdot n$ steps, set $z_i = \mathrm{bin}_n(0) \circ x$.
2. Output the shortest string among
   $z_1, z_2, \ldots, z_n$.
```

$\mathrm{Dec}'(i, z)$*:* If $i = 0$, output z. Otherwise output $\mathrm{Dec}(z, S_i)$.

By inspection, the above algorithms run in polynomial time. For the compression length, suppose X_n is sampled by algorithm $S_k \in \mathcal{S}$. For all $n \geq \max\{k, f(|S_k|)\}$, $\mathrm{Enc}(x, S_k)$ and $\mathrm{Dec}(y_k, S_k)$ will halt within $p(n) \cdot f(n) \leq p(n) \cdot f(|S_k|)$ steps and thus z_k will equal $(k, \mathrm{Enc}(x, S_k))$. Thus, the compression length will be at most

$$\begin{aligned} \mathrm{E}[|\mathrm{Enc}'(X_n)|] &\leq \mathrm{E}[|(k, \mathrm{Enc}(X, S_k))|] \\ &\leq m(H(X_n), n) + O(1), \end{aligned}$$

since k is a constant. For $n \leq \max\{k, f(|S_k|)\}$, the compression length is bounded by a constant. □

We will also allow our compression algorithms to be randomized, and in fact allow Enc and Dec to have *shared randomness*. Formally, we require $\mathrm{Dec}(\mathrm{Enc}(x, r), r) = 1$ for all sequences of coin tosses r, and consider the compression length to be $\mathrm{E}[|\mathrm{Enc}(X, R)|]$, where the expectation is taken over X and the coin tosses R. This still maintains the "spirit" of data compression (because the randomness is independent of the source), and moreover the following lemma shows that the randomness can be eliminated at small cost, under plausible complexity assumptions.

Lemma 3.2. *Suppose there is a function in* $\mathbf{E} = \mathbf{DTIME}(2^{O(n)})$ *of circuit complexity* $2^{\Omega(n)}$. *Then for every polynomial-time compression algorithm* $(\mathrm{Enc}, \mathrm{Dec})$ *with shared randomness there exists a deterministic polynomial-time compression algorithm* $(\mathrm{Enc}', \mathrm{Dec}')$ *such that for every source X_n, if* $(\mathrm{Enc}, \mathrm{Dec})$ *compresses X to length $m = m(H(X_n), n)$, then* $(\mathrm{Enc}', \mathrm{Dec}')$ *compresses X_n to length $m + O(\log n)$. If* Enc *gives a prefix-free encoding, then so does* Enc'.

Proof. Let $t(n)$ be a bound on the running time of $(\mathrm{Enc}, \mathrm{Dec})$ on inputs of length n. Under the hypothesis, there is a pseudorandom generator $G : \{0,1\}^{\ell(n)} \to \{0,1\}^{t(n)}$ with $\ell(n) = O(\log n)$ such that no circuit of size $t(n)$ can distinguish the output of G from uniform with advantage greater than $\epsilon = 1/t(n)$ [19, 13]. We define $\mathrm{Enc}'(x)$ to be the shortest string in the set $\{s \circ \mathrm{Enc}(x, G(s)) : s \in$

$\{0,1\}^{\ell(n)}\}$, where $\circ$ denotes concatenation. Now set $\mathrm{Dec}'(s \circ y) = \mathrm{Dec}(y, G(s))$. By inspection, $\mathrm{Dec}'(\mathrm{Enc}'(x)) = x$ for all x.

For the compression length, the pseudorandom property of G implies that for every string $x \in \{0,1\}^n$,

$$\begin{aligned} & \mathrm{E}_S[|\mathrm{Enc}(x, G(S))|] \\ \leq \; & \mathrm{E}_R[|\mathrm{Enc}(x, R)|] + t(n) \cdot \epsilon \\ = \; & \mathrm{E}_R[|\mathrm{Enc}(x, R)|] + 1. \end{aligned}$$

Thus,

$$\begin{aligned} & \mathrm{E}[|\mathrm{Enc}'(X_n)|] \\ = \; & \mathrm{E}_{X_n}[\min_s |(s, \mathrm{Enc}(X_n, G(s)))|] \\ = \; & \mathrm{E}_{X_n}[\min_s |\mathrm{Enc}(X_n, G(s))|] + O(\log n) \\ \leq \; & \mathrm{E}_{X_n}[\mathrm{E}_S[|\mathrm{Enc}(X_n, G(S))|]] + O(\log n) \\ \leq \; & \mathrm{E}_{X_n}[\mathrm{E}_R[|\mathrm{Enc}(X_n, R)|] + 1] + O(\log n) \\ \leq \; & m(H(X_n), n) + O(\log n) \end{aligned}$$

□

4. Sources with Logspace Samplers

In this section we consider sources sampled by logarithmic space randomized algorithms. As usual in the theory of randomized space-bounded algorithms, we consider a model where the space-bounded machine has *one-way* access to a tape containing random bits.

It is known that no pseudorandom generator can be implemented as a log-space machine with one-way access to the seed [17]. (This follows from the fact that deciding if a given string is a possible output of the generator is a problem in nondeterministic log-space, and so it is solvable in polynomial time.)

In the rest of this section we show that optimal compression is possible for sources sampled by one-way log-space algorithms. This complements the result of Goldberg and Sipser [9], who showed optimal compression for flat sources whose support is *decidable* by one-way log-space machines. Moreover, logspace samplers generalize the Markov chain model used often in compression work [29]. This is because a Markov chain with S states can be converted to a machine using space $\log S$. (S is usually viewed as a constant so uniformity issues do not arise.)

Definition 4.1 (Space-bounded Samplable Sources). *We say that a source X_n is* samplable in space $s(n)$ *if there is a probabilistic Turing machine M such that:*

- *$M(1^n)$ has the same distribution as X_n;*
- *For every content of the random tape, the computation $M(1^n)$ uses space at most $s(n)$*
- *M has* one-way *access to the random tape.*

We say that M is a space-$s(n)$ *sampler.*

Notice that the bound on the space implies that M runs in time $n2^{O(s(n))}$ and uses at most as many random bits.

The main lemma of this section says that the cumulative probability distributions of logspace-samplable sources can be computed in polynomial time. (A potentially larger class of sources can be handled using the techniques of [2].)

Lemma 4.1. *There is an algorithm A that on input a space-$s(n)$ sampler M and string $x \in \{0,1\}^n$ runs in time* $\mathrm{poly}(n, 2^{s(n)})$ *and returns the cumulative probability* $\Pr[M(1^n) \preceq x]$, *where $\preceq$ denotes lexicographic ordering.*

Proof. Given M, we define a new probabilistic space-bounded machine M' that uses space $O(s(n))$ and with the property that, for every $x \in \{0,1\}^n$,

$$\Pr[M'(1^n, x) \text{ accepts}] = \Pr[M(1^n) \preceq x]$$

Given $(1^n, x)$, M' simulates $M(1^n)$, and it accepts if and only if the simulated computation outputs a string a such that $a \preceq x$. Since M' does not have enough space to store a, we need to be careful about the way the simulation is performed. Note that if $a \preceq x$ and a and x have the same length, then either $a = x$ or, for some i, a is a string of the form $(x_1, \ldots, x_{i-1}, 0, a_{i+1}, \ldots, a_n)$, where $x_i = 1$. That is, a starts with a (possibly empty) prefix of x, then it has a zero in a position in which x has a one, and then it continues arbitrarily.

At the beginning of the simulation, the head of M' on the input tape is on the first bit of x. Every time the simulated computation of $M(1^n)$ writes on the output tape, M' compares the bit that $M(1^n)$ is going to write with the current bit of x that it sees on the output tape. If the bits are the same, then M' continues the simulation and moves the input-tape head on to the next symbol of x. If $M(1^n)$ is about to write a one, and the corresponding bit of x is zero, then the simulation halts and M' rejects. If $M(1^n)$ is about to write a zero, and the corresponding bit of x is one, then M' accepts. Also, if the simulation of $M(1^n)$ is completed with the input-tape head moving all the way until the end of x, then also M' accepts. It should be clear that the contents of the random tape for which $M'(1^n, x)$ accepts are precisely those for which $M(1^n)$ outputs a string $\preceq x$.

After constructing M', it then remains to compute $\Pr[M(1^n, x) \text{ accepts}]$, which is a standard problem.

We enumerate all $S = n \cdot 2^{O(s)}$ possible states of $M(1^n, x)$, and construct an $S \times S$ matrix P such that $P_{i,j}$ is the probability that $M(1^n, x)$ goes from state i to state j in one step. We let e be the S-dimensional vector such that $e_i = 1$ if i is the start state of the machine, and $e_i = 0$ otherwise, and we compute the vector eP^S. Then, if A is the set of accepting states of the machine, then $\sum_{a \in A}(eP^S)[a]$ gives the probability that the machine accepts. □

Theorem 1 (Compressing log-space Sources). *Let X_n be a source over $\{0,1\}^n$ samplable in space $O(\log n)$. Then there are polynomial time algorithms $(\mathrm{Enc}, \mathrm{Dec})$ that compress X_n to length $H(X_n) + 2$. The encoding is prefix-free.*

Proof. Combine Lemma 2.2 with Lemma 4.1 □

Corollary 4.1 (Universal Compression of log-space Sources). *For every bound $s(n) = O(\log n)$ there are polynomial-time algorithms $(\mathrm{Enc}, \mathrm{Enc}^{-1})$ such that for every source X_n over $\{0,1\}^n$ samplable in space $s(n)$, and for every sufficiently large n, $(\mathrm{Enc}, \mathrm{Dec})$ compress X_n to length $H(X_n) + O(1)$. The encoding is prefix-free.*

Proof. Combine Theorem 1 with Lemma 3.1. □

5. Sources with Membership Oracles

In this section, we consider an alternative approach to bypassing the impossibility of compressing pseudorandom sources. Here we allow the sampler to be an arbitrary probabilistic polynomial-time algorithm, but explicitly impose the constraint that the source is not pseudorandom.

Definition 5.1. *Let X_n be a flat source. We say that X_n is a* source with membership oracle *if there is a polynomial-time algorithm D such that $D(z) = 1 \Leftrightarrow z \in \mathrm{Sup}(X_{|z|})$. For a source X_x indexed by a string x, we require instead that there is a polynomial-time algorithm D such that $D(x, z) = 1 \Leftrightarrow z \in \mathrm{Sup}(X_x)$.*

Note that a source with membership oracle cannot be pseudorandom; indeed the algorithm D distinguishes it from all sources of higher entropy.

Are all samplable sources with membership oracles efficiently compressible? Goldberg and Sipser [9] showed that any source with membership oracle can be compressed to length $n - \Theta(\log n)$ (provided $H(X_n) < n - (3 + \delta) \log n$). But can they be compressed to length roughly $H(X_n)$? (Think of, say, $H(X_n) = n/2$.) This is an intriguing open question, which we first heard from Impagliazzo [12]. Goldberg and Sipser [9] and Wee [27] provide oracles relative to which the $n - \Theta(\log n)$ bound cannot be improved, and relative to which deterministic compression is impossible.[3] We know of no other evidence regarding this question without oracles.

In this section, we present two positive results about sources with membership oracles. In the first, we show how to compress better than Goldberg–Sipser while using *deterministic* compression and decompression algorithms. In particular, if X_n is a source with membership oracle and $H(X_n) \le k = n - O(\log n)$, then Goldberg & Sipser showed how to compress X_n to length $k + 3 \log n$ with high probability. We show how to compress to length $k + \delta \cdot (n - k)$ in polynomial-time, for any $\delta > 0$. Thus, for $k = n - o(\log n)$ this is more than a constant factor savings in overhead. In deterministic quasi-polynomial time, we only require $k \ge n - (\log n)^{O(1)}$, and we show how to compress to length $k + O(\mathrm{polylog}(n - k)) \le k + O(\mathrm{polylog} \log n)$.

Our technique is completely different than that of [9]. Instead of arithmetic coding, we use the recent explicit construction of constant-degree "lossless" expanders [5], together with an idea from distributed algorithms for routing in expander-based networks [3].

In the second result, we show how to compress to length $H(X) + O(1)$ for a large class of sources

3 We note that Goldberg and Sipser measure compression by the worst-case length (except for a finite number of exceptions, which makes no difference in the Turing machine model), whereas our definitions involve the average-case length, as in [27]. Nevertheless, our construction for high entropy flat sources below actually gives a worst-case bound on compression length.

with membership oracles, namely those whose supports are self-reducible in the sense of [22].

5.1. Compressing high entropy sources

We prove the following theorem.

Theorem 2. *Let X_n be a flat source with membership oracle and $H(X_n) \leq k$. Then X_n is compressible:*

1. *to length $k + \operatorname{polylog}(n-k)$ in time $n^{O((n-k)/\log(n-k))}$, and*
2. *to length $k+\delta \cdot (n-k)$ in polynomial time, for any constant $\delta > 0$, if $k \geq n - O(\log n)$,*

The encodings are prefix-free.

In particular, we get better compression as k approaches n. Yet even for $k = n - \operatorname{polylog}(n)$, we achieve compression to $k + O(\operatorname{polylog}\log n)$ in quasi-polynomial time.

The starting idea of the proof is that we wish to condense the input distribution, without many collisions of points in the support. Lossless condensers, first defined and constructed in [25], do exactly this. We prove that a good condensing function can be used to compress, and then use the expanders constructed by Capalbo et al. [5] as condensing functions. A simple application of this approach would only compress the source to $k + O(\log n)$ bits. Using an idea from [3], we improve the bound to $k + \delta \cdot (n-k)$.

We begin with the following lemma, which shows how a good condensing function can be used to compress.

Lemma 5.1. *Suppose X_n is a flat source with membership oracle and $S = \operatorname{Sup}(X)$. Fix a function $f : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$. Call $z \in \{0,1\}^m$ S-unique if there is exactly one element $(x,r) \in S \times \{0,1\}^r$ such that $f(x,r) = z$. Suppose that $\Pr_{x\in X, r\in U_d}[\text{Enc}(x,r) \text{ is } S\text{-unique}] \geq 1-\epsilon$. Then X_n is compressible to length $m + \epsilon n + 1$ in time $(T_f + T_{f^{-1}})\operatorname{poly}(n)$. Here $T_{f^{-1}}$ denotes the time to compute the set $f^{-1}(y)$ on input y. The encoding is prefix-free.*

Proof. Let $\operatorname{Enc}(x)$ be 0 concatenated with the lexicographically first y of the form $f(x,r)$ which is S-unique, or $1x$ if there is no such y. Let $\operatorname{Dec}(1x) = x$ and $\operatorname{Dec}(0y)$ be the $x \in S$ such that $f(x,r) = y$. Then Enc and Dec satisfy the conclusions of the lemma. □

The function f is essentially a disperser. A disperser is a type of expanding graph where the expansion is required only for sets of a particular size. We will need the expansion close to the degree. For our improvements we will need f to represent a true expander, as defined as follows.

Definition 5.2. *A bipartite graph $G = (V, W, E)$ is a (K, A)-expander if for all $T \subseteq V$, $|T| \leq K$, $|\Gamma(T)| \geq A|T|$.*

The following lemma is self-evident.

Lemma 5.2. *Let $G = (\{0,1\}^n, \{0,1\}^m, E)$ be a $(|S|, (1-\epsilon/2)D_L)$-expander with left degree D_L and right degree D_R. Assume the edges are labeled from $\{0,1\}^d$ with unique labels out of a given node in $\{0,1\}^n$. Define $f(x,r)$ to be the neighbor of x labeled by r. Then f satisfies the conditions of Lemma 5.1.*

We take G to be the expander explicitly constructed by Capalbo et al. [5]:

Theorem 3. *[5][4] Let $N = 2^n \geq K = 2^k$, There are explicitly constructible $(K, (1-\epsilon/2)D_L$ regular expanders $G = (\{0,1\}^n, \{0,1\}^m, E)$ with left degree $D_L = 2^d$ (d to be specified below), and $M = 2^m = O(KD_L/\epsilon)$ such that the set of neighbors of a vertex in $\{0,1\}^n$ is computable in time $\operatorname{poly}(n, D_L)$ and the neighbors of a vertex in $\{0,1\}^m$ are computable in time $\operatorname{poly}(n, D_L, N/M)$ with either of the following values of d:*

1. *$d = \operatorname{poly}(\log(n-k), \log(1/\epsilon))$, or*
2. *$d = \delta \cdot (n-k) + O(\log(1/\epsilon))$, where δ is an arbitrarily small constant.*

Setting $\epsilon = 1/n$, the expanders in Part 2 yield compression length $m = d + k + \log(1/\epsilon) + O(1) = k + \alpha(n-k) + O(\log n) = k + O(\log n)$ for $k = n - O(\log n)$. This compression differs from optimal by an additive $O(\log n)$ term, like in [9]. This was because we had to set $\epsilon = 1/n$. We now describe a method where we can use a larger ϵ.

First note that Hall's theorem implies that in a $(K, 1)$-expander, there is a matching which matches

4 Part 2 is not stated in [5], but can be obtained by using the extractor of [30] to construct the "small conductors" used in the zig-zag product there. In [5], the computation time of neighborhoods of right-vertices is also not stated, but it can be deduced from the the computation time of neighborhoods of left-vertices and the high-level structure of the construction.

all vertices in any K-subset of $\{0,1\}^n$. Such a matching may be used as the compression function. We show that better expansion allows us to construct the matching efficiently.

The idea is as follows. For $S \subseteq V$, let $\mathrm{Uniq}(S) = \{w \in W \,|\Gamma(w) \cap S| = 1\}$ be the set of unique neighbors of S. Let $S_0 = S = \mathrm{Sup}(X_n)$. If there is an r such that $y = f(x,r)$ is S-unique, then we encode x by the lexicographically first such y. This gives a maximal matching from S_0 to $\mathrm{Uniq}(S_0)$. If there is no such r, let S_1 denote all the unmatched elements in S, i.e., $S_1 = S_0 \setminus \Gamma(\mathrm{Uniq}(S_0))$. We now encode x by the lexicographically first $y = f(x,r)$ which is S_1-unique, if it exists. This gives a maximal matching from S_1 to $\mathrm{Uniq}(S_1)$.

In general, in the ith stage, if we haven't encoded x already, we encode x by the lexicographically first $y = f(x,r)$ which is S_i-unique, if it exists. This gives a maximal matching from S_i to $\mathrm{Uniq}(S_i)$. We then set $S_{i+1} = S_i \setminus \Gamma(\mathrm{Uniq}(S_i))$. We do this for at most $t = \lceil (\log n)/\log(1/\epsilon) \rceil$ stages.

The number of stages is chosen so that all but $1/n$ fraction of S have been matched. This is because in each stage, a $1-\epsilon$ fraction of nodes gets matched, so all told $\epsilon^t \leq 1/n$ nodes remain unmatched. It suffices to compress all but $1/n$ fraction because of Lemma 2.3.

Note that the number of oracle calls to check if a node is in S_i is at most $(D_L \cdot D_R)^i$. Here D_R denotes the right degree, $D_R = D_L N/M$.

Using the expanders from Theorem 3 gives compression length

$$m = k + d + \log(1/\epsilon) + O(1)$$

The number of oracle calls is

$$\begin{aligned}(D_L \cdot D_R)^t &= (ND_L^2/M)^t \\ &= \mathrm{poly}(1/\epsilon^t, (N/K)^t, D_L^t).\end{aligned}$$

We also need to multiplying by the time to compute neighborhoods, $\mathrm{poly}(n, D_L, N/M)$, which gives a total running time of

$$\mathrm{poly}(n, 1/\epsilon^t, (N/K)^t, D_L^t).$$

To optimize compression length, we use the expanders from Part 1 and take $\epsilon = 1/(n-k)$. This yields compression length $k + O(\mathrm{polylog}(n-k))$ and running time $\mathrm{poly}(n, 1/(n-k)^t, 2^{(n-k)\cdot t}, 2^{\mathrm{polylog}(n-k)\cdot t}) = n^{O((n-k)/\log(n-k))}$.

To optimize the running time for $k \geq n - O(\log n)$, we use the expanders from Part 2 and set $\epsilon = 2^{-\delta\cdot(n-k)}$ for an arbitrarily small constant $\delta > 0$. This gives running time $\mathrm{poly}(n)$ and compression length $m = k + O(\delta\cdot(n-k)) = k + \delta'\cdot(n-k)$.

We now mention a couple of types of non-flat distributions X_n to which we can extend the above result. Details will be given in the final version.

- The result extends to any X_n whose support is contained in polynomial-time decidable set S of size 2^k (for k satisfying the same constraints as above). This follows because a compression algorithm for the uniform distribution on S is also a compression algorithm for X_n. We also can obtain results in case $\Pr[X_n \in S] \geq 1-\epsilon$ via Lemma 2.3.
- If, instead of having a membership oracle for the support of X_n, we have an oracle for computing the probability mass under X_n, the result extends to any X_n of min-entropy at least $n - O(\log n)$. (This requires a slightly augmented version of our construction.) With a probability-mass oracle, we also obtain nontrivial compression for non-flat sources of low entropy. For example, if $H(X_n) \leq n/4$, then X_n lands in the polynomial-time set $S = \{x : X_n(x) \leq n/2\}$ with probability at least $1/2$, so we can apply the previous bullet.

5.2. Self-Reducible Sets

For a source X_x with membership oracle, the relation $R = \{(x,z) : z \in \mathrm{Sup}(X_x)\}$ is decidable in polynomial time. Thus sources with membership oracles correspond to the uniform distribution on $\mathbf{NP}$ witness sets. Many natural $\mathbf{NP}$ witness sets have the following property of self-reducibility:

Definition 5.3 ([22]). *A polynomially balanced relation $R \subseteq \Sigma^* \times \Sigma^*$ is* self-reducible *if there exist polynomial-time computable functions $\sigma : \Sigma^* \to \mathbb{N}$ and $\rho : \Sigma^* \times \Sigma^* \to \Sigma^*$ such that for all $x, w = w_1 \cdots w_m \in \Sigma^*$*

1. $\sigma(x) = O(\log |x|)$,
2. $(x, w_1 w_2 \cdots w_m) \in R$ *if and only if* $(\rho(x, w_1 \cdots w_{\sigma(x)}), w_{\sigma(x)+1} \cdots w_m) \in R$,
3. $|\rho(x, w_1 w_2 \cdots w_{\sigma(x)})| \leq |x|$.

Intuitively, this definition says that the witness set for a given input can be expressed in terms of witness sets for smaller inputs. Specifically, the witnesses for x which begin with initial segment $w_1 \cdots w_{\sigma(x)}$ are in one-to-one correspondence with the witnesses for the instance $\rho(x, w_1 \cdots w_{\sigma(x)})$. Many natural witness relations are self-reducible in this sense, e.g. satisfying assignments of boolean formulae and perfect matchings in bipartite graphs. Jerrum, Valiant, and Vazirani [16] proved that, for self-reducible relations, witnesses can be generated almost uniformly at random if and only if approximate counting of witnesses can be done in probabilistic polynomial time. And, indeed, there are now many approximate counting algorithms known that have been obtained by first constructing almost-uniform samplers (typically via the Markov chain Monte Carlo method).

The main result of this section (see the Appendix for a proofs) adds compression of the witness set to the list of tasks equivalent to sampling and counting.

Theorem 4. *Let R be a self-reducible relation, and for every x, let X_x be the uniform distribution on $\{w : (x,w) \in R\}$. If the sources X_x are samplable, then they can be efficiently compressed (with shared randomness) to length $H(X_x) + 4$. The encodings are prefix-free.*

Proof. We will show how to compute an "approximate arithmetic encoding" for the sources X_x. A similar approach was used by Goldberg and Sipser [9] in their main result, but as mentioned above they were only able to compress to length $n - O(\log n)$. We use the ideas in the reduction from approximate counting to sampling [16] to obtain an almost-optimal compression length.

The first step is to argue that we can efficiently approximate probabilities of witness prefixes. For an input x and a witness prefix $z = z_1 \cdots z_{\sigma(x)}$, let $p(x,z) = \Pr\left[X_x|_{\sigma(x)} = z\right]$, where $a|_\ell$ denotes the first ℓ bits of a.

Claim 5.1. *There is a probabilistic algorithm $A(x,z,\epsilon,\delta)$ running in time $\mathrm{poly}(|x|, 1/\epsilon, \log(1/\delta))$ such that*

$$\Pr\left[|A(x,z,\epsilon,\delta) - p(x,z)| > \epsilon\right] \le \delta$$

The algorithm A simply takes $\mathrm{poly}(1/\epsilon, \log(1/\delta))$ samples from X_x and outputs the fraction that begin with prefix z. The claim follows from a Chernoff Bound.

Fix an input length n, and set $\delta = 2^{-3n}$, $\epsilon = 1/n^{2c}$, for a large constant c to be specified later. For x of length at most n, z of length at most $\sigma(x)$, and a sequence r of $(\mathrm{poly}(n))$ coin tosses for A, define $q_r(x,z) = A(x,z,\epsilon,\delta;r)$.

Taking a union bound over all x, z, the following holds with probability at least $1 - 2^{-n}$ over r:

$$|q_r(x,z) - p(x,z)| \le \epsilon \qquad \forall |x| \le n, |z| = \sigma(x). \tag{1}$$

Our compression and decompression algorithms will choose r at random, so we may assume they have an r that satisfies this condition (the exponentially rare r's which violate this condition will only increase the expected compression length by at most $\mathrm{poly}(n)/2^{-n}$).

Once r is fixed, the q_r's induce approximating distributions $\hat{X}_{x,r}$ via self-reducibility:

$\hat{X}_{x,r}$**:** Select a prefix $z \in \{0,1\}^{\sigma(x)}$ according to the distribution $q_r(x,\cdot)$. Recursively sample $z' \leftarrow \hat{X}_{\rho(x,z),r}$. Output zz'.

Moreover, we can recursively compute the cumulative distribution function $\hat{F}_{x,r}(w)$ for $\hat{X}_{x,r}$ with respect to the lexicographic order as follows, writing $w = zz'$ with $|z| = \sigma(x)$:

$$\hat{F}_{x,r}(zz') = \left(\sum_{u<z} q_r(x,u)\right) + q_r(x,z) \cdot \hat{F}_{\rho(x,z),r}(z').$$

Thus we can compute the arithmetic coding $(\widehat{\mathrm{Enc}}_{x,r}, \widehat{\mathrm{Dec}}_{x,r})$ (Lemma 2.2) for $\hat{X}_{x,r}$ in polynomial time. Our compression algorithms $(\mathrm{Enc}, \mathrm{Dec})$ for X_x itself are as follows:

$\mathrm{Enc}(x,w,r)$**:** Let $c = \widehat{\mathrm{Enc}}_{x,r}(w)$. If $|c| \le n$, output $0c$. Otherwise output $1x$.

$\mathrm{Dec}(x,bc,r)$**:** If $b = 0$, output $\widehat{\mathrm{Dec}}_{x,r}(c)$. Otherwise output c.

By inspection, $\mathrm{Dec}(x, \mathrm{Enc}(x,w,r), r) = x$ for all w. Thus, we only need to verify the compression length. To do this, we argue about how well $\hat{X}_{x,r}$ approximates X.

Claim 5.2. *With probability at least $1 - 1/n^2$ over $w \leftarrow X_x$, $X_x(w) \le \sqrt{2}\hat{X}_{x,r}(w)$.*

To prove this claim, we call a prefix $z \in \Sigma^{\sigma(x)}$ *light* if $\Pr\left[X_x|_{\sigma(x)} = z\right] \le 1/(n^c|\Sigma|^{\sigma(x)})$. Then, by a union bound over all $z \in \Sigma^{\sigma(x)}$, the probability that $z \leftarrow X_x|_{\sigma(x)}$ is light is at most $1/n^c$. Thus, if we sample from X_x by first sampling a prefix z

and then recursively sampling from $X_{\rho(x,z)}$, we encounter a light prefix somewhere along the way with probability at most $m \cdot (1/n^c)$, where $m = \mathrm{poly}(n)$ is a bound on the length of witnesses. For a sufficiently large choice of c, this probability is at most $1/n^2$.

So we only need to argue that if the sampling of w involves no light prefixes, then $X_x(w) \leq \sqrt{2}\hat{X}_{x,r}(w)$. Let z be the first prefix. By Property (1) of the q_r's, we have

$$\begin{aligned} q_r(x,z) &\geq p(x,z) - \epsilon \\ &= p(x,z) - \frac{1}{n^{2c}} \\ &\geq p(x,z) \cdot \left(1 - \frac{|\Sigma|^{\sigma(x)}}{n^c}\right) \\ &\geq p(x,z) \cdot \left(1 - \frac{1}{3m}\right), \end{aligned}$$

for a sufficiently large choice of the constant c. Expanding $\Pr\left[\hat{X}_{x,r} = zz'\right] = q_r(x,z) \cdot q_r(\rho(x,z),z') \cdots$ and similarly for X_x, we have $\hat{X}_{x,r}(w) \geq (1 - 1/3m)^m \cdot X_x(w) \geq X_x(w)/\sqrt{2}$, as desired.

We can now estimate the compression length of X_x under $(\mathrm{Enc}(x,\cdot,r), \mathrm{Dec}(x,\cdot,r))$. Recall that the arithmetic coding $\widehat{\mathrm{Enc}}_{x,r}(w)$ compresses an individual string w to length $\lceil \log(1/\hat{X}_{x,r}(w)) \rceil$. If r and w satisfy the Inequalities (1) and the conclusion of Claim 5.2, then we can bound this length as

$$\lceil \log(1/\hat{X}_{x,r}(w)) \rceil \leq \log(1/X_x(w)) + 3/2.$$

The probability that r and w do not satisfy either the Inequalities (1) or the conclusion of Claim 5.2 is at most $2^{-n} + 1/n^2$. Thus, the average compression length is at most

$$\begin{aligned} & \mathrm{E}_{w \leftarrow X_{x,r}}[|\mathrm{Enc}(x,w,r)|] \\ = \; & \mathrm{E}_{w \leftarrow X_{x,r}}[\max\{|\widehat{\mathrm{Enc}}_{x,r}(w)|, n\}] + 1 \\ \leq \; & \mathrm{E}_{w \leftarrow X_x}[\log(1/X_x(w)) + 2] \\ & + (1/n^2 + 2^{-n}) \cdot n + 1 \\ = \; & H(X_x) + 3, \end{aligned}$$

for large enough n, as desired. □

The randomization in the compression algorithms above can be eliminated via Lemma 3.2, under a complexity assumption. However, if we do not care for a full derandomization, and only to eliminate the *shared* randomness, we can use a "random perturbation" trick from [9] to do it without a complexity assumption.

Proposition 5.1. *Let R be a self-reducible relation, and for every x, let X_x be the uniform distribution on $\{w : (x,w) \in R\}$. If the sources X_x are samplable, then they can be compressed by probabilistic polynomial-time algorithms with no shared randomness to length $H(X_x) + O(\log n)$. The encodings are prefix-free.*

Proof. (sketch) The reason that the shared randomness is needed above is so that Enc and Dec compute the same approximations $q_{x,r}$. Thus, it suffices to ensure that they compute the same approximations with high probability even if they use independent randomness. This can be done by perturbing the approximations by a random noise $\eta \leftarrow [-1/n^a, 1/n^a]$ and then rounding the approximations to the nearest multiple of $1/n^b$ for appropriate constants a, b. η should be included with the compressed string (to $O(\log n)$ bits of accuracy). □

The above theorem actually only requires that X_x can be *approximately sampled*. That is, there is a probabilistic algorithm S such that for all x, the output distribution of $S(x,\epsilon)$ is within variation distance at most ϵ of X_x, and $S(x,\epsilon)$ runs in time $\mathrm{poly}(|x|, 1/\epsilon)$. Thus, we obtain compression algorithms for the wide variety of self-reducible structures for which almost-uniform samplers are known, most notably the set of perfect matchings in a bipartite graph [15]. The ability to compactly store combinatorial substructures of a graph could be useful, for example, in storing substructures of huge graphs such as the World Wide Web; indeed, there have been recent efforts at compressing Web graphs [1].

In addition, we can show that compression and almost-uniform sampling are *equivalent*.

Theorem 5. *Let R be a self-reducible relation, and for every x, let X_x be the uniform distribution on $W_x = \{w : (x,w) \in R\}$. Then the following conditions are equivalent:*

1. *X_x can be approximately sampled in polynomial time.*
2. *X_x can be compressed to length $H(X_x) + O(1)$ by probabilistic polynomial-time compression algorithms with shared randomness.*
3. *X_x can be compressed to length $H(X_x) + O(\log n)$ by probabilistic polynomial-time compression algorithms with no shared randomness.*

Proof. (sketch) We have already argued that sampling (Item 1) implies compression (Items 2 and 3). For the converse, suppose $(\mathrm{Enc}, \mathrm{Dec})$ compresses X_x to length $m \leq H(X_x) + c \log n$ with shared randomness. By the results of Sinclair and Jerrum [24] (building on [16]), approximate sampling follows if we can approximate $|W_x|$ to within a $\mathrm{poly}(n)$ accuracy factor. This would be easy if we could estimate the average compressed length m; unfortunately, random sampling from X_x is unavailable to us.

Instead, we use random sampling from the compressed space and decompressing. In particular, we will use random sampling to estimate $p_\ell = \Pr_{y \in U_{\leq \ell}}[\mathrm{Dec}(y) \in W_x]$, where $U_{\leq \ell}$ denotes the uniform distribution on $\{0,1\}^{\leq \ell}$, the set of strings of length $\leq \ell$. With high probability, we find the largest integer $\hat{m}$ such that $p_{\hat{m}} \geq 1/n^{c+2}$.

We claim that $2^{\hat{m}}$ approximates $|W_x|$ to within a polynomial factor. For one direction, note that $|W_x| \geq p_{\hat{m}} |\{0,1\}^{\leq \hat{m}}| \geq 2^{\hat{m}}/n^{c+2}$. For the other direction, note that Markov's inequality implies that $\Pr_{w \in X_x}[|\mathrm{Enc}(w)| \leq m+1] \geq 1/(m+1)$. Therefore, the number of encodings of W_x with length at most $m+1$ is at least $|W_x|/(m+1) \geq 2^m/(n^c(m+1)) \geq 2^{m+2}/n^{c+2} > |\{0,1\}^{\leq m+1}|/n^{c+2}$. Hence $\hat{m} \geq m+1 \geq H(X_x) - O(1)$ and $2^{\hat{m}} \geq \Omega(|W_x|)$. □

A final extension we mention is that our results also apply to some non-uniform distributions on the witness set $\{w : (x,w) \in R\}$. Specifically, it applies to sources X_x that are compatible with the self-reduction in the sense that the distribution of X_x conditioned on having prefix z is precisely $z \circ X_{\rho(x,z)}$. An example is perfect matchings on weighted bipartite graphs, where each edge has a nonnegative weight and the probability of matching is proportional to the product of the weights on its edges. The algorithm of [15] can also sample from such distributions, and hence we can also compress such distributions close to the entropy.

References

[1] M. Adler and M. Mitzenmacher. Toward compressing web graphs. In *Proceedings of the 2001 Data Compression Conference*, 2001.

[2] E. Allender, D. Bruschi, and G. Pighizzini. The complexity of computing maximal word functions. *Computational Complexity*, 3(4):368–391, 1993.

[3] S. Arora, F. T. Leighton, and B. M. Maggs. On-line algorithms for path selection in a nonblocking network. *SIAM Journal on Computing*, 25(3):600–625, 1996.

[4] B. Barak, R. Shaltiel, and A. Wigderson. Computational analogues of entropy. In *11th International Conference on Random Structures and Algorithms*, 2003.

[5] M. Capalbo, O. Reingold, S. Vadhan, and A. Wigderson. Randomness conductors and constant-degree lossless expanders. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing*, pages 659–668, 2002.

[6] B. Chor and O. Goldreich. Unbiased bits from sources of weak randomness and probabilistic communication complexity. *SIAM Journal on Computing*, 17(2):230–261, 1988.

[7] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley & Sons, Inc., 1991.

[8] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions in Information Theory*, IT-22(6):644–654, 1976.

[9] A. Goldberg and M. Sipser. Compression and ranking. *SIAM Journal on Computing*, 20:524–536, 1991.

[10] O. Goldreich and S. Vadhan. Comparing entropies in statistical zero-knowledge with applications to the structure of SZK. In *Proc. of Conference on Computational Complexity*, pages 54–73, 1999.

[11] J. Håstad, R. Impagliazzo, L. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28:1364–1396, 1999.

[12] R. Impagliazzo, October 1999. Remarks in Open Problem session at the DIMACS Workshop on Pseudorandomness and Explicit Combinatorial Constructions.

[13] R. Impagliazzo and A. Wigderson. P = BPP if E requires exponential circuits: Derandomizing the XOR lemma. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 220–229, 1997.

[14] M. Jerrum and A. Sinclair. Approximating the permanent. *SIAM Journal on Computing*, 18(6):1149–1178, 1989.

[15] M. Jerrum, A. Sinclair, and E. Vigoda. A polynomial-time approximation algorithm for the permanent of a matrix with non-negative entries. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing*, pages 712–721, 2001.

[16] M. Jerrum, L. Valiant, and V. Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theoretical Computer Science*, 43:169–188, 1986.

[17] M. Kharitonov, A. V. Goldberg, and M. Yung. Lower bounds for pseudorandom number generators. In *Proceedings of the 30th Annual IEEE Symposium on Foundations of Computer Science*, pages 242–247, 1989.

[18] R. Lipton. A new approach to information theory. In *Proc. o f11th Symposium on Theoretical Aspects of Computer Science*, pages 699–708, 1994.

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

[20] A. Sahai and S. Vadhan. A complete problem for statistical zero knowledge. *Journal of the ACM*, 50(2):196–249, March 2003. Extended abstract in *FOCS '97*.

[21] M. Santha and U. V. Vazirani. Generating quasi-random sequences from semi-random sources. *Journal of Computer and System Sciences*, 33:75–87, 1986.

[22] C. Schnorr. Optimal algorithms for self-reducible problems. In *Proceedings of the 3rd International Colloquium on Automata, Languages, and Programming*, pages 322–337, 1976.

[23] C. E. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28:656–715, 1949.

[24] A. Sinclair and M. Jerrum. Approximate counting, uniform generation and rapidly mixing Markov chains. *Information and Computation*, 82:93–133, 1989.

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

[26] L. Trevisan and S. P. Vadhan. Extracting randomness from samplable distributions. In *Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science*, pages 32–42, 2000.

[27] H. Wee. On pseudoentropy versus compressibility. These Proceedings, 2004.

[28] A. C. Yao. Theory and applications of trapdoor functions. In *Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science*, pages 80–91, 1982.

[29] J. Ziv and A. Lempel. Compression of individual sequences by variable rate coding. *IEEE Transactions on Information Theory*, 24:530–536, 1978.

[30] D. Zuckerman. Randomness-optimal oblivious sampling. *Random Structures and Algorithms*, 11:345–367, 1997.