COMPRESSION OF SAMPLABLE SOURCES

Luca Trevisan, Salil Vadhan, and David Zuckerman

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 $k + \text{polylog}(n - k)$.
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 $H(X) + 5$.

Keywords. expander graphs, arithmetic coding, randomized logspace, pseudorandom generators, approximate counting
Subject classification. 68P30 Coding and information theory

1. Introduction

Data compression has been studied extensively in the information theory literature (see e.g. Cover & Thomas (1991) 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 & Sipser (1991) 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 **P**, and thus implicitly considering sources uniformly distributed on all n-bit strings in such a language.

Samplable Sources with Membership Algorithms. 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 **P**. Indeed, while Goldberg & Sipser show that every sparse language in **P** can be compressed at least slightly, this is unlikely to be true for all polynomially samplable distributions. In particular, as first observed by Levin,[1] pseudorandom distributions are incompressible and, if pseudorandom generators exist, then there are polynomially samplable pseudorandom distributions. (See Section 3 for more details.)

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. We do this by considering sources for which membership in the support can be tested in polynomial time. We first study logspace sources, which have this property implicitly, while later we study flat sources with explicit membership algorithms.

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 (which 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; see Kharitonov *et al.* (1989).

We show the existence of a universal compression algorithm for such sources

[1] according to Goldberg & Sipser (1991)

that compresses optimally, up to an additive constant, 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 constant additive 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.

Flat Sources with Membership Algorithms. We next consider more general 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 & Sipser (1991) 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 $k + \mathrm{polylog}(n-k) \leq k + \mathrm{polylog}\log n$. 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, due to Capalbo *et al.* (2002). In addition, our compression algorithm is deterministic, whereas the Goldberg–Sipser algorithm is probabilistic. Our algorithm, however, only achieves good *average* compression length, while the Goldberg–Sipser algorithm compresses every element of the support of the source.

In our last main result, we show that if the support of the samplable distribution forms the witness set for a *self-reducible* **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; see Jerrum *et al.* (2001). Our compression algorithm computes an "approximate" arithmetic coding, using ideas underlying the proof, due to Jerrum *et al.* (1986), that sampling implies approximate counting for self-reducible relations. 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, cf. Jerrum & Sinclair (1989); Jerrum *et al.* (1986)).

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, Shannon (1949), could be bypassed via the reasonable assumption that the adversary is computationally bounded Diffie & Hellman (1976). Our restriction to *samplable* sources in particular was motivated by the work of Trevisan & Vadhan (2000), who 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, see Chor & Goldreich (1988); Santha & Vazirani (1986), and it was known to be possible for very structured sources like Markovian sources (just like the data compression problem). Trevisan & Vadhan (2000) show 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 (1994), 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, see Goldreich & Vadhan (1999); Sahai & Vadhan (2003). Understanding the compressibility of samplable sources can contribute to this general study, as it provides another measure of the (computational) randomness in a source. Indeed Yao (1982) proposed such a compressibility measure of randomness, and this is one of the several measures of computational randomness recently studied by Barak *et al.* (2003). In the same spirit, a few years ago, Impagliazzo (1999) posed an intriguing question about the relationship between the 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 (2004) 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 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, but rather under complexity-theoretic assumptions.

Finally, the notion of compression we study is in some sense the common generalization of two other problems widely studied in the computational complexity literature — specifically "randomness condensers" (or hashing) and "resource-bounded Kolmogorov complexity". Loosely speaking, in the study of condensers one is interested in efficient compression algorithms, with no bounds on the complexity of decompressing, while in resource-bounded Kolmogorov complexity one is interested in efficient decompression algorithms, with no bounds on the complexity of compressing.

A *lossless condenser* for a source (see e.g. Raz & Reingold (1999); Ta-Shma *et al.* (2001)) is a randomized procedure that, with high probability, is injective (or approximately injective) when applied to samples from the source. The output of the condenser is efficiently computable and can be seen as a compression of the sample; however, no efficient decompressing algorithm is required to exist. Condensers have been studied for their applications to randomness *extractors* (Nisan & Zuckerman (1996)), and no assumption is typically made on the source that they're applied to, other than the source having bounded "min-entropy".

Resource-bounded Kolmogorov complexity (cf. Li & Vitanyi (1997)) focuses on the following question: for a fixed universal Turing machine U, given a time bound t and a string x, what is the shortest encoding y of x such that $U(y)$ will output x within t time steps? Thus, here one studies efficient decompression without the requirement that the compressed representation be computable by an efficient algorithm. For example, while the output of a pseudorandom generator is an incompressible source according to our definition, each of the possible outputs of the generator has low resource-bounded Kolmogorov complexity (because the corresponding seed s is an efficiently decompressible representation of the output $G(s)$) . The study of *language compression* (see e.g. Buhrman *et al.* (2004) for recent results and references to earlier work) focuses on the worst-case compressibility (in the above sense) for sources that are flat over an efficiently decidable support (i.e. sources with membership oracles, just as we study).

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 ensembles $(X_x)_{x \in L}$ 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 alphabet 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*

[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.

(i) For all $x \in \mathrm{Sup}(X)$, $\mathrm{Dec}(\mathrm{Enc}(x)) = x$, *and*

(ii) $\mathrm{E}[|\mathrm{Enc}(X)|] \leq m$.

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 exist functions* Enc *and* Dec *such that* (Enc, Dec) *compresses* X *to length* m.

The following simple lemma allows us to assume throughout that all encodings are of length at most $n+1$.

LEMMA 2.4. *If a source* X_n *is compressible to length* m*, then it is compressible to length* $m+1$ *where for all* $x \in \mathrm{sup}(X_n)$, $|\mathrm{Enc}(x)| \leq n+1$.

PROOF. Let (Enc, Dec) compress X_n to length m. Define $\mathrm{Enc}'(x)$ to be $0\mathrm{Enc}(x)$ (0 concatenated with $\mathrm{Enc}(x)$) if $|\mathrm{Enc}(x)| < n$, and $1x$ otherwise. Then $|\mathrm{Enc}'(x)| \leq n+1$, $\mathrm{E}[|\mathrm{Enc}(X_n)|] \leq m+1$, and there is an efficient inverse Dec'. □

It is well known that a source X is compressible to length $H(X)+1$ by a prefix-free encoding (see e.g. Cover & Thomas (1991)). 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, Definition 2.2 above is less restrictive (often called "nonsingular" compression) and allows 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^i1y. 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 2.5. *A source* X_n *is not compressible to length less than* $H(X_n) - \lceil \log(n+1) \rceil$

PROOF. Convert any encoding Enc to a prefix-free encoding Enc′ as follows. If $|\mathrm{Enc}(x)| \leq n$, then we define $\mathrm{Enc}'(x) = \ell(x)\mathrm{Enc}(x)$, where $\ell(x)$ is the number $|\mathrm{Enc}(x)|$ written in binary, padded to length $\lceil \log(n+1) \rceil$. If $|\mathrm{Enc}(x)| > n$, then we define $\mathrm{Enc}'(x) = \ell(x)x$, where $\ell(x)$ is the number $n+1$ written in binary. Then Enc' is prefix free, and hence uniquely decodable. The new compression length is $\mathrm{E}[|\mathrm{Enc}'(X_n)|] \leq \mathrm{E}[|\mathrm{Enc}(X_n)| + \lceil \log(n+1) \rceil]$. By the lower bound for uniquely decodable codes, we have $\mathrm{E}[|\mathrm{Enc}'(X_n)|] \geq H(X_n)$. Thus, $\mathrm{E}[\mathrm{Enc}(X_n)] \geq H(X_n) - \lceil \log(n+1) \rceil]$, as desired. □

We remark that, by a more careful reduction (specifically, encoding $\ell(x)$ in a prefix-free way without padding to length $\lceil \log(n+1) \rceil$), the loss of $\log(n+1)$ can be replaced with a term depending logarithmically on only $H(X_n)$ (rather than n).

We mention that for *flat* sources, $H(X) - O(1)$ is again a lower bound.

LEMMA 2.6. *A flat source X_n is not compressible to length less than $H(X_n) - 3$.*

PROOF. It suffices to show that if X_n is uniform on a set of size 2^k for some integer k, then it is not compressible to length less than $k - 2$. The optimal compression has support uniform on all strings of length less than k, plus some given string of length k. Ignoring the string of length k, this compresses X_n to length greater than

$$\frac{1}{2}(k-1) + \frac{1}{4}(k-2) + \frac{1}{8}(k-3) + \ldots \geq k-2,$$

as needed. □

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 2.7 (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 $(X_n)_{n \in \mathbb{Z}^+}$ 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 (Ziv & Lempel (1978)). (That is, the Lempel–Ziv 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. Cover & Thomas (1991)) 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 2.8 (arithmetic coding, cf. Cover & Thomas 1991). *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 Cover & Thomas (1991).) 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. Specifically, we say that $(\mathrm{Enc}, \mathrm{Dec})$ compresses a source X with *decoding error* ϵ if $\Pr[\mathrm{Dec}(\mathrm{Enc}(X)) \neq X] \leq \epsilon$. The following lemma shows that we can eliminate small decoding error at a small price in compression length and efficiency.

LEMMA 2.9. *Suppose X_n is a source on $\{0,1\}^n$ that is compressible to length m with decoding error ϵ, by algorithms* (Enc, Dec) *computable in time T. Suppose further that for all $x \in \mathrm{Sup}(X_n)$, $|\mathrm{Enc}(x)| \geq m_0$. Then X_n is compressible to length $m + \epsilon(n - m_0) + 1$ (with zero decoding error), by algorithms* $(\mathrm{Enc}', \mathrm{Dec}')$ *computable 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 (Proof of Lemma 2.9). 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. □

2.3. Randomized compression. We will also consider compression algorithms that are randomized. Here we consider two variants, one where Enc and Dec have *independent randomness* and one where they have *shared randomness*. In both cases, we measure the compression length as $\mathrm{E}[|\mathrm{Enc}(X, R)|]$, where the expectation is taken over X and the coin tosses R of Enc. They differ in the definition of decoding error. For independent randomness, the decoding error refers to a bound on $\Pr[\mathrm{Dec}(\mathrm{Enc}(X, R_1), R_2) \neq X]$, whereas with shared randomness it refers to a bound on $\Pr[\mathrm{Dec}(\mathrm{Enc}(X, R), R) \neq X]$. Unless otherwise specified, we require that the decoding error is 0 (even with zero error, randomization could be useful in achieving a small compression length with polynomial-time algorithms). Note that zero-error randomized compression algorithms (with either shared or independent randomness) are subject to the same lower bounds on compression length as in Lemmas 2.5 and 2.6. The reason is that for each fixing of the coin tosses, the lower bounds for deterministic compression algorithms apply.

We also note that in the case of shared randomness, decoding error can be eliminated in a manner analogous to Lemma 2.9 (with the same price on compression length and efficiency):

LEMMA 2.10. *Suppose X_n is a source on $\{0,1\}^n$ that is compressible to length m with decoding error ϵ by algorithms* (Enc, Dec) *with shared randomness, computable in time T. Suppose further that for all $x \in \mathrm{Sup}(X_n)$, $|\mathrm{Enc}(x)| \geq$*

m_0. Then X_n is compressible to length $m + \epsilon \cdot (n - m_0) + 1$ with shared randomness and zero decoding error, by algorithms $(\mathrm{Enc}', \mathrm{Dec}')$ *computable in time $O(T)$. If* Enc *gives a prefix-free encoding, then so does* Enc$'$.

For independent randomness, it is not clear how to eliminate decoding error (while maintaining the independence of the randomness used by Enc and Dec). However, it can be made exponentially small:

LEMMA 2.11. *Suppose X_n is a source on $\{0,1\}^n$ that is compressible to length m with decoding error ϵ by algorithms* $(\mathrm{Enc}, \mathrm{Dec})$ *with independent randomness, computable in time T. Suppose further that for all $x \in \mathrm{Sup}(X_n)$, $|\mathrm{Enc}(x)| \geq m_0$. Then X_n is compressible to length $m + 3\epsilon \cdot (n - m_0) + 2$ with independent randomness and decoding error 2^{-n}, by algorithms* $(\mathrm{Enc}', \mathrm{Dec}')$ *computable in time $O(n \cdot T)$. If* Enc *gives a prefix-free encoding, then so does* Enc$'$.

PROOF (Proof of Lemma 2.11). We construct Enc$'$ as follows. On input x, Enc$'(x)$ computes $y \leftarrow \mathrm{Enc}(x)$. It then runs $O(n)$ independent executions of Dec(y). If at least a .6 fraction of these executions output x, then it outputs $0y$. Otherwise, it outputs $1x$.

Before describing Dec$'$, we analyze the compression length of Enc$'$. Assume without loss of generality that $\epsilon \leq 1/3$. Then by Markov's inequality, with probability at least $1 - 3\epsilon$ over $x \xleftarrow{\mathrm{R}} X_n$ and the coin tosses r_1 of Enc, we have $\Pr_{R_2}[\mathrm{Dec}(\mathrm{Enc}(x, r_1), R_2) \neq x] \leq 1/3$. For each such x and r_1, Enc$'$ will output $1x$ with probability at most 2^{-n} (by a Chernoff bound). Thus, the probability that Enc$'$ outputs $1x$ rather than $0\mathrm{Enc}(x, r_1)$ is at most $3\epsilon + 2^{-n}$. This implies that the average compression length increases by at most $(3\epsilon + 2^{-n}) \cdot (n - m_0) + 1 \leq 3\epsilon \cdot (n - m_0) + 2$.

Now we describe Dec$'$. On an input of the form $1x$, Dec$'$ outputs x. On an input of the form $0y$, Dec$'$ runs $O(n)$ independent executions of Dec(y) and outputs the value that appears most often (breaking ties arbitrarily).

Note that decoding errors can only occur when Enc$'(x)$ outputs a compressed string of the form $0y$, for $y = \mathrm{Enc}(x, r_1)$. For any x, r_1, we consider two cases. If $\Pr[\mathrm{Dec}(y) = x] \geq .55$, then by a Chernoff bound, Dec$'$ will decode correctly with probability at least $1 - 2^{-n}$. If $\Pr[\mathrm{Dec}(y) = x] \leq .55$, then by a Chernoff bound, Enc$'$ will output $1x$ with probability at least $1 - 2^{-n}$. Thus the decoding error is at most 2^{-n}. □

Finally, we observe that randomized compression algorithms can be converted into deterministic ones at a small cost, under plausible complexity assumptions.

LEMMA 2.12. *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 $\epsilon = 1/t(n)$ (Impagliazzo & Wigderson (1997); Nisan & Wigderson (1994)). 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 $x \in \{0,1\}^n$,

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

Thus,

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

□

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 for every $n \in \mathbb{N}$. "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 $(X_x)_{x \in L}$ indexed by strings, we instead require that $S(x)$ is distributed according to X_x for every $x \in L$.

It is natural to consider samplable sources, since any flat 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 & Sipser (1991) 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. (Goldberg & Sipser (1991) attribute this observation to L. Levin.)

Proposition 3.2 (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* (Enc, Dec)*, even if the algorithms are allowed to use shared randomness.*

Proof. (sketch) Håstad *et al.* (1999) showed that if one-way functions exist, then there exists a pseudorandom generator $G : \{0,1\}^{n^\epsilon} \to \{0,1\}^n$. Let $X_n = G(U_{n^\epsilon})$. From the pseudorandom property of G, it follows that

$$\Pr[\mathrm{Dec}(\mathrm{Enc}(U_n)) = U_n] \geq \Pr[\mathrm{Dec}(\mathrm{Enc}(X_n)) = X_n] - \mathrm{neg}(n) = 1 - \mathrm{neg}(n),$$

where neg denotes a negligible function. Otherwise, the following procedure would be a distinguisher: on input x, accept if and only if $\mathrm{Dec}(\mathrm{Enc}(x)) = x$.

The pseudorandom property of G also implies that

$$\mathrm{E}[|\mathrm{Enc}(U_n)|] \leq \mathrm{E}[|\mathrm{Enc}(X_n)|] + \mathrm{neg}(n).$$

Otherwise, there would be an encoding length ℓ such that there is a noticeable difference between the probability that $|\mathrm{Enc}(U_n)| = \ell$ and the probability that $|\mathrm{Enc}(X_n)| = \ell$, and we would have a distinguisher.

Suppose that $\mathrm{E}[|\mathrm{Enc}(X_n)|] \leq n-3$. Then, by the above reasoning we have that, for sufficiently large n,

$$\Pr[\mathrm{Dec}(\mathrm{Enc}(U_n)) = U_n] \geq .99 \tag{3.3}$$

and

$$\mathrm{E}[|\mathrm{Enc}(U_n)|] \leq n - 2.99 \;. \tag{3.4}$$

By (3.3), there are $.99 \cdot 2^n$ or more elements x of $\{0,1\}^n$ such that $\mathrm{Dec}(\mathrm{Enc}(x)) = x$. The contribution of these to the expectation in (3.4) is at least $n-2.01$, by a calculation similar to the one in the proof of Lemma 2.6. This is a contradiction. □

Conversely, Wee (2004) proves that if one-way functions don't exist, then every polynomial-time samplable flat source X_n can be compressed to length $H(X_n) + O(\log n)$ (for infinitely many n).

Thus, we do not expect 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 3.5. *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 let $\Sigma^* = \{S_1, S_2, S_3, \ldots\}$ be an enumeration of all strings in lexicographic 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 $z_i = 0^i 1 \circ y_i$.

(c) If either Enc or Dec failed to halt within $p(n) \cdot n$ steps, set $z_i = 1 \circ x$.

2. Output the shortest string among $z_1, z_2, \ldots, z_n$.

$\mathrm{Dec}'(0^i 1 \circ z)$: 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 $0^k 1 \circ \mathrm{Enc}(x, S_k))$. Thus, the compression length will be at most

$$\begin{aligned} \mathrm{E}[|\mathrm{Enc}'(X_n)|] &\leq \mathrm{E}[|0^k 1 \mathrm{Enc}(X, S_k)|] \\ &\leq m(H(X_n), n) + O(1), \end{aligned}$$

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

Before moving on to our positive results, we observe that good compression of a samplable source implies that the source's entropy can be approximated.

Proposition 3.6. *If a polynomial-time samplable source X_x distributed over $\{0,1\}^n$ (for $n = n(x)$) is compressible to length $m = m(x)$ by a probabilistic polynomial-time encoding algorithm* Enc *(even sharing randomness with the decoding algorithm* Dec*), then there is a probabilistic polynomial-time algorithm A such that* $\Pr[A(x) \in [H(X_x) - \log n - 1, m + 1/2]] \geq 2/3$.

Proof. By Lemma 2.5 and hypothesis, we have

$$H(X_x) - \log n - 1/2 \leq \mathrm{E}[|\mathrm{Enc}(X_x, R)|] \leq m.$$

A simply estimates the average compression length $\mathrm{E}[|\mathrm{Enc}(X_x, R)|]$ by taking polynomially many independent samples $x_1, \ldots, x_k \stackrel{\mathrm{R}}{\leftarrow} X_x$ and sequences of coin tosses $r_1, \ldots, r_k$ for Enc, and computing the average of the $|\mathrm{Enc}(x_i, r_i)|$'s. Taking $k = O(n^2)$, we obtain an approximation of $\mathrm{E}[|\mathrm{Enc}(X_x, R)|]$ to within $\pm 1/2$ with high probability. □

In particular, one way to show that a family of sources X_x does not have good polynomial-time compression algorithms is to show that it is intractable to approximate the entropy of X_x. For example, Goldreich & Vadhan (1999) showed that the problem of approximating the entropy of a general polynomial-time samplable source is complete for **SZK**, the class of problems possessing statistical zero knowledge proofs. (More precisely, the problem is the following: given a boolean circuit $C : \{0,1\}^m \to \{0,1\}^n$, approximate the entropy of distribution $X_C = C(U_m)$.) Using this, we obtain the following.

PROPOSITION 3.7. *If* $\mathbf{SZK} \neq \mathbf{BPP}$, *then there is a family of samplable sources* $\{X_x\}_{x \in L}$ *that cannot be compressed to length* $H(X_x) + n^{1-\alpha}$ *by a probabilistic polynomial-time encoding algorithm* Enc *(even sharing randomness with the decoding algorithm* Dec*), for any constant* $\alpha > 0$.

This result is incomparable to Proposition 3.2. Proposition 3.7 only requires that the encoding algorithm be efficient, but Proposition 3.2 rules out compression even to length $n-3$ and uses a qualitatively weaker assumption (Ostrovsky (1991), Ostrovsky & Wigderson (1993) have shown that $\mathbf{SZK} \neq \mathbf{BPP}$ implies the existence of a variant of one-way functions).

We note that Proposition 3.7 relies on the fact that the compression algorithm is not given a bound on the entropy of X_x. In fact, the literature on *lossless condensers*, cf. Raz & Reingold (1999); Ta-Shma *et al.* (2001), gives efficient, randomized encoding algorithms that near-optimally compress flat sources (with a small decoding error) when given a bound on the entropy. Condensers do not, however, provide an efficient decoding algorithm. (Indeed, if the decoding algorithm were efficient, then one could eliminate the need to know a bound on the entropy by trying $k = 1, \ldots, n$, using the one that gives the smallest encoding length and decodes correctly, and including the value of k used in the compressed string.)

Note that for flat sources, an additive approximation to $H(X_x)$ is equivalent to a multiplicative approximation to $|\mathrm{Sup}(X_x)|$, which is an approximate counting problem in the usual sense. In Section 7, we will exploit this relationship between compression and approximate counting in the opposite direction, using techniques from approximate counting algorithms to develop compression algorithms for a certain class of sources.

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.

Kharitonov *et al.* (1989) have shown that no pseudorandom generator can be implemented as a log-space machine with one-way access to the seed. (This follows from the fact that deciding if a given string is a possible output of the generator is a problem in non-deterministic 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 & Sipser (1991), 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 (see e.g. Ziv & Lempel (1978)). 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 its random tape,*
- *M has write-only access to its output.*

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 Allender *et al.* (1993).)

LEMMA 4.2. *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. $\square$

THEOREM 4.3 (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* (Enc, Dec) *that compress X_n to length $H(X_n) + 2$. The encoding is prefix-free.*

PROOF. Combine Lemma 2.8 with Lemma 4.2 $\square$

COROLLARY 4.4 (Universal Compression of log-space Sources). *For every bound $s(n) = O(\log n)$ there are polynomial-time algorithms* (Enc, Dec) *such that for every source X_n over $\{0,1\}^n$ samplable in space $s(n)$, and for every sufficiently large n,* (Enc, Dec) *compress X_n to length $H(X_n) + O(1)$. The encoding is prefix-free.*

PROOF. Combine Theorem 4.3 with Lemma 3.5. □

5. Sources with Membership Algorithms

In the rest of this paper, 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 algorithm *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 a membership algorithm cannot be pseudorandom; indeed the algorithm D distinguishes it from all sources of higher entropy.

Are all samplable sources with membership algorithms efficiently compressible? Goldberg & Sipser (1991) showed that any source with membership algorithm 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 (1999). Goldberg & Sipser (1991) and Wee (2004) 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 the next two sections, we present two positive results about sources with membership algorithms. 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 algorithm 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 $k + \mathrm{polylog}(n - k) \le k + \mathrm{polylog} \log n$.

Our technique is completely different than that of Goldberg & Sipser (1991). Instead of arithmetic coding, we use the recent explicit construction by Capalbo *et al.* (2002) of constant-degree "lossless" expanders.

[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 does the work of Wee (2004).

In the second result, we show how to compress to length $H(X) + O(1)$ for a large class of sources with membership algorithms, namely those whose supports are self-reducible in the sense of Schnorr (1976).

6. Compressing High Entropy Sources

We prove the following theorem.

THEOREM 6.1. *Let X_n be a flat source with membership algorithm and $H(X_n) \leq k$. Then X_n is compressible to any length $k+\mathrm{polylog}(n-k)$ in time $\mathrm{poly}(n, 2^{n-k})$. In particular, if $k = n - O(\log n)$, then the compression is polynomial time. The encoding is prefix-free.*

The 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 by Raz & Reingold (1999) and Ta-Shma *et al.* (2001), 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.* (2002) as condensing functions.

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

LEMMA 6.2. *Suppose X_n is a flat source with membership algorithm and $S = \mathrm{Sup}(X)$. 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 $\Pr_{x \in X, r \in U_d}$[f(x,r) is S-unique] $\geq 1 - \epsilon$. Then X_n is compressible to length m with decoding error ϵ in (deterministic) time $T_f \cdot T_{f^{-1}} \cdot \mathrm{poly}(n)$. Here T_f denotes the time to compute the set $\{f(x, r) : r \in \{0,1\}^d\}$ on input x and $T_{f^{-1}}$ denotes the time to compute the set $f^{-1}(y)$ on input y. The encoding is prefix-free.*

PROOF. Let $\mathrm{Enc}(x)$ be the lexicographically first y of the form $f(x, r)$ that is S-unique, and let $\mathrm{Dec}(y)$ be the lexicographically first x such that $(x, r) \in f^{-1}(y)$ for some r and $x \in S$ (or 0^m if no such x exists). Note that computing $\mathrm{Enc}(x)$ can be done by enumerating the set $F(x) = \{f(x, r) : r \in \{0,1\}^d\}$, which takes time T_f, and testing each element $y \in F(x)$ for S-uniqueness. Testing a string y for S-uniqueness can be done by enumerating the set $f^{-1}(y)$, which takes time $T_{f^{-1}}$, and testing each element of $f^{-1}(y)$ for membership in S, which takes time $\mathrm{poly}(n)$. Computing $\mathrm{Dec}(y)$ can similarly be done by enumerating $f^{-1}(y)$ and testing each element for membership in S. □

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. It is convenient to use a true expander, as then we don't need to know $|S|$ (which corresponds to not needing to know $H(X_n)$ in Theorem 6.1, but only an upper bound). Known dispersers also do not appear to improve our bounds.

DEFINITION 6.3. *A bipartite graph $G = (V, W, E)$ is a (K, A)-expander if, for all subsets $T \subseteq V$ such that $|T| \leq K$, we have $|\Gamma(T)| \geq A \cdot |T|$, where $\Gamma(T)$ denotes the set of neighbors of the vertices in T.*

The following lemma is self-evident.

LEMMA 6.4. *Let $G = (\{0,1\}^n, \{0,1\}^m, E)$ be a $(K, (1-\epsilon/2)D_L)$-expander with left degree $D_L = 2^d$ and right degree D_R. Assume the edges out of a given node in $\{0,1\}^n$ are labelled with unique labels from $\{0,1\}^d$. Define $f(x, r)$ to be the neighbor of x labelled by r. Then f satisfies the conditions of Lemma 6.2 for any source X_n whose support S is of size at most K.*

We take G to be the expander explicitly constructed by Capalbo *et al.* (2002):

THEOREM 6.5 (Capalbo *et al.* 2002). *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 $D_L = 2^d$, $d = \mathrm{poly}(\log(n-k), \log(1/\epsilon))$, and $M = 2^m = O(KD_L/\epsilon)$. The set of neighbors of a vertex in $\{0,1\}^n$ is computable in time $\mathrm{poly}(n, D_L)$ and the set of neighbors of a vertex in $\{0,1\}^m$ are computable in time $\mathrm{poly}(n, D_L, N/K)$*

In Capalbo *et al.* (2002), the computation time of the neighborhoods of the right-vertices is not explicitly stated. For completeness, we sketch how to obtain these in Appendix A.

Applying these expanders with $\epsilon = 1/(n-k)$ yields compression length

$$m = k + d + \log(1/\epsilon) + O(1) = k + \mathrm{polylog}(n-k)$$

and running time $\mathrm{poly}(n, 2^{\mathrm{polylog}(n-k)}, 2^n/2^k) = \mathrm{poly}(n, 2^{n-k})$. Removing decoding errors via Lemma 2.9 increases the compression length by $\epsilon \cdot (n-m) + 1 < 2$ bits. This completes the proof of Theorem 6.1.

In the original version of this paper (see Trevisan *et al.* (2004)), we had a weaker version of Theorem 6.1 with a more involved proof, because we did not choose parameters optimally. Yet ideas in that proof can be used to achieve slightly better compression in the sense that a larger fraction of elements from the source are compressed to length $k + \mathrm{polylog}(n-k)$ (all but a $1/n$ fraction rather than a $1/(n-k)$ fraction); this improvement is not reflected in the average compression length (which is the measure we use).

The idea in the original proof, based upon Arora *et al.* (1996), was to first compress the S-unique strings as above. Then, however, we consider the set S_1 of remaining strings, and compress them recursively using the same algorithm. By setting the number of levels of recursion in this process, we can trade off the fraction of strings compressed with the running time.

6.1. Generalizing to Non-Flat Sources. We now extend the above techniques to non-flat distributions. We are able to obtain bounds on compression length in terms of a "truncated" variant of entropy, defined as follows.

DEFINITION 6.6. *Let X_n be a source. For $x \in \{0,1\}^n$ and $\Delta \leq n$, define $h(x) = \log(1/\Pr[X_n = x])$ and*

$$h^{\Delta}(x) = \begin{cases} n-\Delta & \text{if } h(x) < n-\Delta, \\ h(x) & \text{if } h(x) \in [n-\Delta, n], \\ n & \text{if } h(x) > n, \end{cases}$$

and

$$H^{\Delta}(X_n) = \mathrm{E}_{X_n}[h^{\Delta}(X_n)].$$

Also, recall that the *min-entropy* $H_\infty(X)$ of a source X is defined as the minimum of $h(x)$ over all the elements x of the support of x, that is,

$$H_\infty(X) = \min_{x:\Pr[X=x]\neq 0} h(x) \; .$$

For a non-flat distribution X_n, a natural generalization of a membership algorithm is a *probability mass algorithm*: a polynomial-time algorithm D such that for every $z \in \{0,1\}^n$, $D(z) = \Pr[X_n = z]$. For sources with a probability mass algorithm we prove the following result.

THEOREM 6.7. *Let X_n be a source with probability mass algorithm and let c be any constant. Then X_n is compressible to length $H^{c\log n}(X_n) + \mathrm{polylog}(n - H^{c\log n}(X_n))$ in polynomial time. The encodings are prefix-free.*

PROOF. First, we observe that the proof of Theorem 6.1 immediately works for sources that are "nearly flat," in the sense that every two elements of

$\mathrm{Sup}(X_n)$ have probability mass within a factor of, say, 2 of each other. This is the case because if the algorithm correctly compresses all but a $\epsilon = 1/(n-k)$ fraction of elements of $\mathrm{Sup}(X_n)$, then it will also correctly compress all but a 2ϵ fraction of the distribution X_n. Removing errors via Lemma 2.9 will then increase the compression length by at most $2\epsilon \cdot (n-m)+1 < 3$ bits.

To handle general sources X_n, we bucket the elements of $\mathrm{Sup}(X_n)$ into sets S_i, consisting of elements of probability mass in the interval $[2^{-i}, 2^{-(i+1)})$. Note that given $x \in \mathrm{Sup}(X_n)$, we can determine its bucket $i(x)$ using the probability mass algorithm for X_n.

To compress elements x of S_i (where $i = i(x)$), we use the compression algorithm from the proof of Theorem 6.1, replacing the parameter k with $k_i = \max\{i+1, n-(c+1)\log n\}$. We denote such an encoding of a string $x \in S_i$ by $\mathrm{Enc}_i(x)$. The final encoding $\mathrm{Enc}(x)$ of a string x is as follows: If $i(x) \leq n$ then we set $\mathrm{Enc}(x) = 0 \circ (n - i(x)) \circ 1 \circ \mathrm{Enc}_{i(x)}(x)$ where $(n-i(x))$ is written as a string of length $2\lceil \log(n-i(x)) \rceil$ by taking its binary expansion and replacing each 0 with 00 and each 1 with 01. If $i(x) \geq n$, then we set $\mathrm{Enc}(x) = 1 \circ x$.

Since the running time of the compression algorithms in Theorem 6.1 are decreasing in k, the running time of the new compression algorithm can be bounded by substituting $k = n-(c+1)\log n$ into the same expressions, yielding polynomial running time. The compression length can be bounded as follows:

$$
\begin{aligned}
&\mathrm{E}_{X_n}[|\mathrm{Enc}(X_n)|] \\
&\leq \mathrm{E}_{X_n}[k_{i(X_n)} + \mathrm{polylog}(n - k_{i(X_n)}) + \log(n - i(X_n)) + O(1)] \\
&\leq \mathrm{E}_{X_n}[\max\{i(X_n), n - c\log n\} + \mathrm{polylog}(n - \max\{i(X_n), n - c\log n\}) + O(1)] \\
&\leq \mathrm{E}_{X_n}[\max\{i(X_n), n - c\log n\}] + \mathrm{polylog}(\mathrm{E}_{X_n}[n - \max\{i(X_n), n - c\log n\}]) + O(1) \\
&\leq H^{c\log n}(X_n) + \mathrm{polylog}(n - H^{c\log n}(X_n)) + O(1).
\end{aligned}
$$

The second inequality above is obtained by separately considering the cases that $i(X_n)+1 \leq n-(c+1)\log n$ (in which case the $\log(n-i(X_n))$ term is absorbed into the first term) and $i(X_n)+1 > n-(c+1)\log n$ (in which case the $\log(n-i(X_n))$ term is absorbed into the second). The third inequality is obtained by applying Jensen's inequality to the function $f(x) = \log^m x$, which is concave when m is constant and x is sufficiently large. $\square$

We now deduce two corollaries of the above, giving compression bounds in terms of the actual entropy of the source.

Corollary 6.8. *Let X_n be a source with probability mass algorithm having min-entropy at least $n - c\log n$ for some constant c. Then X_n is polynomial-*

time compressible to length $H(X_n) + \mathrm{polylog}(n - H(X_n))$*, via a prefix-free encoding.*

PROOF. If X_n has min-entropy at least $n - c\log n$, then $H^{c\log n}(X_n) \le H(X_n)$. □

COROLLARY 6.9. *Let* X_n *be a source with probability mass algorithm. Then for any constant* $c > 0$*,* X_n *is polynomial-time compressible to length*

$$n - \left(1 - \frac{H(X_n)}{n}\right) \cdot c\log n + O(1).$$

PROOF. Let $H(X_n) = n - \Delta$, so our goal is to compress to length $n - (\Delta/n) \cdot c\log n + O(1)$.

First, we may assume that $\Delta \ge 2c\log n$; otherwise compressing to length n (say via the identity map) suffices. Call a string x *light* if $h(x) > n - c\log n$. Let L be the set of light strings. By Markov's inequality, $\Pr[X_n \in L] \le H(X_n)/(n - c\log n)$. So

$$\begin{aligned} \Pr[X_n \notin L] &\ge 1 - H(X_n)/(n - c\log n) \\ &= (\Delta - c\log n)/(n - c\log n) \\ &\ge \Delta/2n \end{aligned}$$

Thus,

$$\begin{aligned} H^{c\log n}(X_n) &\le \Pr[X_n \in L] \cdot n + \Pr[X_n \notin L] \cdot (n - c\log n) \\ &= n - \Pr[X_n \notin L] \cdot (c\log n) \\ &\le n - (\Delta/2) \cdot (c\log n) \end{aligned}$$

Thus, setting $\Delta' = (\Delta/2)\cdot(c\log n)$ and applying Theorem 6.7, we can compress X_n to length

$$n - \Delta' + \mathrm{polylog}\Delta' \le n - \Delta'/2 + O(1) = n - \Delta \cdot (c\log n)/4 + O(1).$$

Increasing c by a factor of 4 yields the desired bound. □

Notice that the results in this section do not require that the source X_n is samplable, but only that X_n has a membership algorithm (in the case of flat sources) or a probability-mass algorithm (in the case of general sources). For flat sources of entropy at least $n - O(\log n)$, a membership algorithm implies samplability: one can sample by randomly picking elements of $\{0,1\}^n$ and testing if they are in the support of X_n. But our results also apply to sources of entropy smaller than $n-O(\log n)$ (though they will only achieve compression length $n-O(\log n)$). This leads to the question of whether better compression can be achieved based on just the membership algorithm condition. Below we give evidence that the membership algorithm condition alone is unlikely to imply near-optimal compression, even for sources of entropy *zero*.

PROPOSITION 6.10. *Suppose that every family of flat sources $(X_x)_{x\in L}$ of zero entropy with a membership algorithm can be compressed to length $m = m(n)$ by a polynomial-time compression algorithm* $(\mathrm{Enc}, \mathrm{Dec})$ *with shared randomness. Then* SAT *is in* $\mathbf{RTIME}(\mathrm{poly}(n) \cdot 2^{m(n)})$.

PROOF. We show that the hypothesis implies a randomized algorithm for finding satisfying assignments to formulas with a *unique* satisfying assignment, and then apply the Valiant & Vazirani (1986) reduction from SAT to UNIQUE-SAT. For a boolean formula φ, consider the source X_φ that is uniform on the satisfying assignments of φ, and let L be the set of formulas φ with exactly one satisfying assignment. Then $(X_\varphi)_{\varphi\in L}$ has a membership algorithm because checking whether an assignment satisfies a formula is easy. Now, if $(\mathrm{Enc}, \mathrm{Dec})$ compress X_φ for $\varphi \in L$ to expected length m, then with probability at least $1/(m+1)$ over the coin tosses R of Enc and Dec, the unique satisfying assignment of φ gets compressed to length at most m. In such a case, the satisfying assignment can be found by enumerating all $O(2^m)$ strings z of length at most m and computing $\mathrm{Dec}(z, R)$. Repeating for $O(m)$ independent choices of R amplifies the probability of finding an assignment to $1/2$.

Valiant & Vazirani (1986) give a randomized polynomial-time reduction mapping any formula ψ to a formula φ on the same number n of variables such that if ψ is satisfiable, then with constant probability φ has has exactly one satisfying assignment, and if ψ is unsatisfiable, then with probability 1 φ is unsatisfiable. Composing this reduction with the above algorithm for finding unique satisfying assignments yields the claimed algorithm for SAT. □

Thus, if SAT requires time $2^{\Omega(n)}$, the above gives a family of zero-entropy sources that cannot be compressed to length $o(n)$. The argument can be modified to give an incompressible family of sources indexed only by input length, under an assumption about "unique nondeterministic exponential time".

PROPOSITION 6.11. *Suppose that every family of flat sources $(X_n)_{n\in\mathbb{N}}$ of entropy at most 1 with a membership algorithm can be compressed to length $m = m(n)$ by a polynomial-time compression algorithm* $(\mathrm{Enc}, \mathrm{Dec})$ *with shared randomness. Then* $\mathbf{UTIME}(2^n) \subseteq \mathbf{RTIME}(2^{O(n)} \cdot 2^{2m(2^{n+O(1)})})$.

PROOF. Let M be a nondeterministic Turing machine running in time $T(\ell) = 2^\ell$ on inputs of length ℓ such that M has zero or one accepting computation on each input. Our aim is to show that, under the hypothesis, $L(M)$ can be decided by a randomized algorithm running in time $2^{O(\ell)} \cdot 2^{2m(2^{\ell+O(1)})}$ on inputs of length ℓ.

Let M' be a nondeterministic TM running in time 2^ℓ that has exactly one more accepting computation than M on each input (by adding a trivial accepting computation). We view each possible input of length ℓ to M' as a binary number n in the interval $[2^{\ell+c}, 2^{\ell+c} + 2^\ell]$, where 2^c upper bounds the branching factor of M'. Since M' has running time 2^ℓ, computations of M' can be described by strings of length n. We define X_n to be the the uniform distribution on the accepting computations of M'. (For n not in an interval $[2^{\ell+1}, 2^{\ell+1} + 2^\ell]$, X_n can be taken to be the distribution that always outputs 0^n.) Notice that X_n has entropy zero or 1, depending on whether M' has 1 or 2 accepting computations on input n. Membership in the support of X_n can be decided in time $2^{O(\ell)} = \mathrm{poly}(n)$.

Now we argue that a good compression algorithm can be used to decide $L(M)$, specifically by yielding an efficient algorithm to find all accepting computations of M'. If $(\mathrm{Enc}, \mathrm{Dec})$ compress X_n to length m, then with probability at least $1/(m+1)$ over the coin tosses R of Enc, all accepting computations of M' are compressed to length at most $2m$. (At worst, one is compressed to length $2m$ and the other to length zero.) Thus, the accepting computations can be found in time $\mathrm{poly}(n) \cdot 2^{2m(n)} = 2^{O(\ell)} \cdot 2^{2m(2^{\ell+c})}$. □

If we impose the additional condition that X_n is samplable, then we do not know of any evidence suggesting the intractability of near-optimal compression other than the oracle result of Wee (2004).

7. 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 **NP** witness sets. Many natural **NP** witness sets have the following property of self-reducibility:

DEFINITION 7.1 (Schnorr 1976). *A polynomially balanced relation $R \subseteq \Sigma^* \times \Sigma^*$ is* self-reducible *if there exist polynomial-time computable functions $\ell : \Sigma^* \to \mathbb{N}$, $\sigma : \Sigma^* \to \mathbb{N}$, and $\rho : \Sigma^* \times \Sigma^* \to \Sigma^*$ such that for all $x, w = w_1 \cdots w_m \in \Sigma^*$,*

(i) $(x, w) \in R \Rightarrow |w| = \ell(x)$.

(ii) *For all* x, $\sigma(x) \leq \ell(x)$, *and* $\ell(x) > 0 \Rightarrow \sigma(x) > 0$.

(iii) $\sigma(x) = O(\log |x|)$,

(iv) $(x, w_1 w_2 \cdots w_{\ell(x)}) \in R$ *if and only if* $(\rho(x, w_1 \cdots w_{\sigma(x)}), w_{\sigma(x)+1} \cdots w_{\ell(x)}) \in R$,

(v) $|\rho(x, w_1 w_2 \cdots w_{\sigma(x)})| \leq |x|$.

(vi) *If* $\ell(x) = 0$, *then* R *can be decided in polynomial time.*

As usual, the language *associated with* R *is* $L_R = \{x : \exists w (x, w) \in R\}$.

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.

Example 7.2. (perfect matchings) Let R be the relation consisting of pairs (G, M), where G is a bipartite graph and M is a perfect matching in G. This is self-reducible because the perfect matchings in $G = (V, E)$ that contain some edge $e = (i, j) \in E$ are in one-to-one correspondence with the perfect matchings in $G' = (V \setminus \{i, j\}, E \setminus \{e\})$, and those that do not contain e are in one-to-one correspondence with the perfect matchings in $G'' = (V, E \setminus \{e\})$.

More formally, we represent G by its $n \times n$ adjacency matrix, and if G has m edges, then M is represented by a bit vector $M_1 M_2 \cdots M_m$ where M_i indicates whether or not edge i is included in the perfect matching. Then we set $\ell(G) = m$, $\sigma(G) = 1$ (unless $m = 0$, in which case $\sigma(G) = 0$), and define $\rho(G, 0)$ to be the graph obtained by removing edge 1 from G (but keeping its endpoints as vertices), and $\rho(G, 1)$ to be the graph obtained by removing edge 1 and its endpoints from G. $\diamondsuit$

Jerrum *et al.* (1986) 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; see the surveys of Jerrum & Sinclair (1996); Kannan (1994); Randall (2003)).

The main result of this section adds compression of the witness set to the list of tasks equivalent to sampling and counting.

THEOREM 7.3. *Let R be a self-reducible relation, and for every $x \in L_R$, let X_x be the uniform distribution on $W_x = \{w : (x, w) \in R\}$. If the sources $(X_x)_{x \in L_R}$ are samplable, then they can be efficiently compressed to length $H(X_x)+5$ with shared randomness and zero decoding error. 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 & Sipser (1991) in their main result, but as mentioned above they were only able to compress to length $n - O(\log n)$. (Their algorithm, however, compresses every string in the support of the source, and it does not require the self-reducibility condition that we have in this theorem.) We use the ideas in the reduction from approximate counting to sampling of Jerrum *et al.* (1986) 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 $a|_t$ denotes the first t bits of a.

CLAIM 7.4. *There is a probabilistic algorithm $A(x, z, \epsilon, \delta; r)$ (where r are the coin tosses) running in time $\mathrm{poly}(|x|, 1/\epsilon, \log(1/\delta))$ such that*

(i) For every x, z, ϵ, δ, $\Pr\left[|A(x, z, \epsilon, \delta) - p(x, z)| > \epsilon\right] \le \delta$, and

(ii) For every x, ϵ, δ, r, $A(x, \cdot, \epsilon, \delta; r)$ is a probability measure on $\Sigma^{\sigma(x)}$. That is, $\sum_{z \in \Sigma^{\sigma(x)}} A(x, z, \epsilon, \delta; r) = 1$ and for every $z \in \Sigma^{\sigma(x)}$, $A(x, z, \epsilon, \delta; r) \in [0, 1]$.

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:

$$|q_r(x, z) - p(x, z)| \le \epsilon \qquad \forall |x| \le n, |z| = \sigma(x). \tag{7.5}$$

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:

$\hat{X}_{x,r}$**:** If $\ell(x) = 0$, output the empty string. Otherwise:

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

$$\hat{F}_{x,r}(zz') = \left(\sum_{u<z} q_r(x,u)\right) + q_r(x,z) \cdot \hat{F}_{\rho(x,z),r}(z'). \tag{7.6}$$

Thus we can compute the arithmetic coding $(\widehat{\mathrm{Enc}}_{x,r}, \widehat{\mathrm{Dec}}_{x,r})$ (Lemma 2.8) 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 $s = \widehat{\mathrm{Enc}}_{x,r}(w)$. If $|s| \le \ell(x)$, output $0s$. Otherwise output $1w$.

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

By inspection, $\mathrm{Dec}(x, \mathrm{Enc}(x,w,r), r) = w$ 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_x.

Claim 7.7. *With probability at least $1 - 1/(n \cdot \ell)$ over $w \leftarrow X_x$ (where $\ell = \ell(x)$), we have $X_x(w) \le \sqrt{2}\hat{X}_{x,r}(w)$.*

Proof of claim: To prove this claim, we call a prefix $z \in \Sigma^{\sigma(x)}$ *light* if

$$\Pr\left[X_x|_{\sigma(x)} = z\right] \le 1/(n^c \cdot |\Sigma|^{\sigma(x)}).$$

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 $\ell \cdot (1/n^c)$, because there are at most ℓ levels of recursion. For a sufficiently large choice of c, this probability is at most $1/(n \cdot \ell)$.

So we only need to argue that if the sampling of w involves no light prefixes, then $X_x(w) \le \sqrt{2}\hat{X}_{x,r}(w)$. Let z be the first prefix. By Property (7.5) of the q_r's, we have

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

for a sufficiently large choice of the constant c. By the definition of self-reducibility and the fact that X_x is uniform on W_x, we can expand $X_x(w)$ for any x and $w = z_1 \cdots z_t \in W_x$ as follows:

$$\Pr\left[X_x = z_1 z_2 \cdots z_t\right] = p(x_0, z_1) \cdot p(x_1, z_2) \cdot p(x_2, z_3) \cdots p(x_{t-1}, z_t), \tag{7.8}$$

where $x_0 = x$, $|z_i| = \sigma(x_{i-1})$, $x_i = \rho(x_{i-1}, z_i)$, and $\sigma(x_t) = 0$. Similarly, by the recursive definition of $\hat{X}_{x,r}$, we have:

$$\Pr\left[\hat{X}_{x,r} = z_1 z_2 \cdots z_t\right] = q_r(x_0, z_1) \cdot q_r(x_1, z_2) \cdot q_r(x_2, z_3) \cdots q_r(x_{t-1}, z_t), \tag{7.9}$$

Putting all of the above together, we have $\hat{X}_{x,r}(w) \ge (1 - 1/3\ell)^\ell \cdot X_x(w) \ge X_x(w)/\sqrt{2}$, 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 $\lceil \log(1/\hat{X}_{x,r}(w)) \rceil + 1$. If r and w satisfy the Inequalities (7.5) and the conclusion of Claim 7.7, then we can bound this length as

$$\left|\widehat{\mathrm{Enc}}_{x,r}(w)\right| = \lceil \log(1/\hat{X}_{x,r}(w) \rceil + 1 \le \log(1/X_x(w)) + 5/2.$$

The probability that r and w do not satisfy either the Inequalities (7.5) or the conclusion of Claim 7.7 is at most $2^{-n} + 1/(n \cdot \ell)$. Thus, the average compression length is at most

$$\begin{aligned} \mathrm{E}_{w \leftarrow X_{x,r}}[|\mathrm{Enc}(x,w,r)|] &= \mathrm{E}_{w \leftarrow X_{x,r}}[\max\{|\widehat{\mathrm{Enc}}_{x,r}(w)|, \ell\}] + 1 \\ &\le \mathrm{E}_{w \leftarrow X_x}[\log(1/X_x(w)) + 5/2] + (1/(n \cdot \ell) + 2^{-n}) \cdot \ell + 1 \\ &\le H(X_x) + 4, \end{aligned}$$

for large enough n, as desired. □

The randomization in the compression algorithms above can be eliminated via Lemma 2.12, 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 of Goldberg & Sipser (1991) to do it without a complexity assumption.

PROPOSITION 7.10. *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 to length $H(X_x) + O(\log n)$ with independent randomness and decoding error 2^{-n}. The encodings are prefix-free.*

PROOF. The only use of randomness in the above proof is to compute the approximations q_r satisfying Property (7.5), and this randomness r needs to be shared so that both the encoder and decoder utilize the same approximations. Thus, it suffices to show how they can compute their approximations independently, yet have the approximations be equal with high probability. Roughly speaking, we do this by perturbing the approximations with random noise η and rounding. It turns out that the noise only needs to specified to $O(\log n)$ bits and thus can be included as part of the compressed string.

We now proceed with the details. The randomness used by Enc and Dec consists of two parts — r, which is not shared, and η which will be shared (by explicit inclusion in the compressed string). To compute an approximation $q_{r,\eta}(x, z)$, we first use the algorithm $A(x, z, \epsilon, \delta; r)$ from Claim 7.7, setting $\delta = 2^{-3n}$ (as before) and $\epsilon = 1/n^{4c}$ (instead of $1/n^{2c}$). Then we take η, which is a random number in $\{0, 1, \ldots, n^c - 1\}$, and set $q'_{r,\eta}(x, z)$ to equal $A(x, z, \epsilon, \delta; r) + \eta/n^{4c}$ *rounded to the nearest multiple of* $1/n^{3c}$. Note that the noise and rounding increase the error (in approximating $p(x, z)$) by at most $2/n^{3c}$. However, $q'_{r,\eta}(x, \cdot)$ no longer defines a probability measure (because the perturbations have all been positive). Thus we observe that we can (deterministically) convert $q'_{r,\eta}(x, \cdot)$ into a probability measure $q_{r,\eta}(x, \cdot)$, while reducing each entry by at most $2/n^{3c}$.

Notice that with probability at least $1 - 2^{-n}$ over r and η, the $q_{r,\eta}$'s satisfy the following analogue of Property (7.5):

$$|q_{r,\eta}(x, z) - p(x, z)| \le \epsilon + O\left(\frac{1}{n^{3c}}\right) < \frac{1}{n^{2c}} \qquad \forall |x| \le n, |z| = \sigma(x).$$

Thus, if the encoding algorithm uses the $q_{r,\eta}$'s in in place of the q_r's, the bound on compression length will hold just as before, except that we add $O(\log n)$ bits to specify the noise $\eta \in \{0, \ldots, n^c - 1\}$.

So all that remains is to argue that decoding is correct with high probability. For this, we argue that the encoder and decoder compute the same approximations with high probability. Specifically, we argue that for every x and z,

$$\Pr_{r_1,r_2,\eta}[q_{r_1,\eta}(x,z) = q_{r_2,\eta}(x,z)] \geq 1 - 2/n^c. \tag{7.11}$$

First, by Claim 7.7, we know that with probability at least $1 - 2 \cdot 2^{-n}$, both $A(x,z,\epsilon,\delta;r_1)$ and $A(x,z,\epsilon,\delta;r_2)$ differ from $p(x,z)$ by at most $\epsilon = 1/n^{4c}$, so they differ from each other by at most $2/n^{4c}$. Thus there are at most two values of $\eta \in \{0,1,\ldots,n^c-1\}$ such that $A(x,z,\epsilon,\delta;r_1) + \eta/n^{4c}$ and $A(x,z,\epsilon,\delta;r_2) + \eta/n^{4c}$ round to different multiples of $1/n^{3c}$.

To complete the proof, we argue that (whp) both Enc and Dec evaluate the $q_{r,\eta}$'s on some $p(n) = \mathrm{poly}(n)$ inputs (x,z) where $p(n)$ is a fixed polynomial independent of the choice of the constant c, and the sequence of inputs is independent of r and η. Thus, by Inequality (7.11), the probability that the two algorithms "see" any difference in their approximations (and decoding possibly fails) is at most $p(n)\cdot(2/n^c)$. By Lemma 2.11, we can reduce this decoding error to 2^{-n} while increasing the compression length by at most $(2p(n)/n^c)\cdot\ell+2 < 3$ bits for a sufficiently large constant c. So we proceed to argue that the number and sequence of evaluations of $q_{r,\eta}$ is indeed fixed (independent of c and the randomness). By inspection, we see that the arithmetic coding $\widehat{\mathrm{Enc}}_{x,r,\eta}(w)$ (Lemma 2.8) only requires evaluating the cumulative distribution function $\hat{F}_{x,r,\eta}$ at w and its predecessor. By Equation (7.6), we see that evaluating $\hat{F}_{x,r,\eta}$ requires only a fixed polynomial number of evaluations of $q_{r,\eta}$ and the evaluation points are independent of r and η. This handles the encoding algorithm Enc. Now recall that the decoding algorithm decodes $\widehat{\mathrm{Enc}}_{x,r,\eta}(w)$ by using $\hat{F}_{x,r,\eta}$ to do binary search for the sample w. By inspection, if the decoding algorithm were given the *same* function $q_{r,\eta}$ as the encoding algorithm, then the evaluations made in the binary search for w would be independent of r and η (because it would successfully traverse the path down to w). $\square$

The above results actually only require that X_x can be *approximately sampled* in the following sense.

DEFINITION 7.12. *A family of sources $(X_x)_{x\in L}$ is* approximately samplable *if there is a probabilistic algorithm S such that for every $x \in L$ and $\epsilon > 0$, the output $S(x,\epsilon)$ has statistical difference (i.e. variation distance) at most ϵ from X_x, and $S(x,\epsilon)$ runs in time* $\mathrm{poly}(|x|, 1/\epsilon)$.

PROPOSITION 7.13. *Let R be a self-reducible relation, and for every $x \in L_R$, let X_x be the uniform distribution on $W_x = \{w : (x, w) \in R\}$. If the sources $(X_x)_{x \in L_R}$ are approximately samplable, then they can be efficiently compressed to length $H(X_x) + 6$ with shared randomness and zero decoding error, and to length $H(X_x) + O(\log n)$ with independent randomness and decoding error 2^{-n}. The encodings are prefix-free.*

PROOF. In the proof of Theorem 7.3, both the encoding and decoding algorithms use the sampling algorithm for the distributions X_x only as an oracle to obtain samples from the distribution. Since they make only poly(n) queries to the oracle, if we replace the oracle with a distribution at statistical difference ϵ, the statistical difference of the outcome (i.e. the compressed string, and an indicator for whether or not decoding is successful) will be at most $\epsilon \cdot \text{poly}(n)$. Choosing ϵ to be a sufficiently small polynomial, we can make this statistical difference smaller than $1/(2\ell')$, where ℓ' is the maximum encoding length. This implies that the average encoding length changes by at most $(1/(2\ell')) \cdot \ell' = 1/2$ and the probability of unsuccessful decoding is at most $1/(2\ell')$. Applying Lemma 2.9 completes the proof. $\square$

Thus, we obtain compression algorithms for the wide variety of self-reducible structures for which almost-uniform samplers are known. For example:

COROLLARY 7.14. *The following families of sources X_x can be efficiently compressed to length $H(X_x) + 6$ with shared randomness and zero decoding error, and to length $H(X_x) + O(\log n)$ with independent randomness and decoding error 2^{-n}:*

(i) X_G = *the uniform distribution on all perfect matchings in bipartite graph G, cf.* Jerrum et al. *(2001).*

(ii) X_G = *the uniform distribution on all matchings in graph G, cf. Jerrum & Sinclair (1989).*

(iii) X_G = *the uniform distribution on all independent sets in graph G of degree at most 4, cf. Luby & Vigoda (1999).*

(iv) $X_{(a_1,\ldots,a_n,b)}$ = *the uniform distribution on all "knapsack solutions", i.e. subsets $S \subseteq [n]$ such that $\sum_{i \in S} a_i \leq b$, where $a_1, \ldots, a_n, b$ are positive real numbers, cf. Morris & Sinclair (1999).*

(v) X_ϕ = *the uniform distribution on satisfying assignments of DNF formula φ, cf.* Jerrum et al. *(1986);* Karp et al. *(1989).*

The citations refer to the papers establishing the approximate samplability of the given distributions. Actually, for DNF formula, the ideas underlying the approximate counting algorithm of Karp *et al.* (1989) directly yields a simple compression algorithm: given a satisfying assignment $w \in \{0,1\}^t$ of a DNF formula $\varphi = C_1 \vee \cdots \vee C_m$ with minimum clause length k, we define $\mathrm{Enc}_\varphi(w)$ to be $(i, \alpha) \in [m] \times \{0,1\}^{t-k}$, where C_i is the first clause satisfied by w and α is the restriction of w to the variables outside C_i. It is easy to check that this encoding is efficiently decodable, and compresses to length $\lceil \log m \rceil + t - k \leq \lceil \log m \rceil + H(X_\varphi)$. Compressing to length $H(X_\varphi) + O(1)$, however, seems less immediate.

The ability to compactly store combinatorial substructures of a graph (as in the above corollary) 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; see Adler & Mitzenmacher (2001). There are many other examples of self-reducible relations to which our technique can be applied; see the surveys Jerrum & Sinclair (1996); Kannan (1994); Randall (2003) and the references therein.

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

THEOREM 7.15. *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:*

(i) X_x can be approximately sampled in polynomial time.

(ii) X_x can be compressed to length $H(X_x)+O(1)$ by probabilistic polynomial-time compression algorithms with shared randomness and zero decoding error.

(iii) X_x can be compressed to length $H(X_x)+O(\log n)$ by probabilistic polynomial-time compression algorithms with independent randomness and decoding error 2^{-n}.

(iv) X_x can be compressed to length $H(X_x)+O(\log n)$ by probabilistic polynomial-time compression algorithms with shared randomness and decoding error $1/n$.

PROOF. By Proposition 7.13, sampling (Item i) implies compression in the sense of Items ii and iii. Each of these latter two items imply Item iv, so we need only argue that Item iv implies Item i. So suppose $(\mathrm{Enc}, \mathrm{Dec})$ compresses X_x to length $m \leq H(X_x) + c \log n$ with shared randomness. We may assume there is zero decoding error, by Lemma 2.10. By the results of Sinclair & Jerrum (1989) (building on work by Jerrum *et al.* (1986)), approximate sampling follows if we can approximate $|W_x|$ to within a poly(n) accuracy factor in polynomial time. 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

$$p_\ell = \mathrm{E}_r \left[\Pr_{y \leftarrow U_{\leq \ell}} [\mathrm{Dec}(x,y,r) \in W_x \& \mathrm{Enc}(x, \mathrm{Dec}(x,y,r), r) = y] \right],$$

where $U_{\leq \ell}$ denotes the uniform distribution on $\{0,1\}^{\leq \ell}$, the set of strings of length $\leq \ell$. By sampling, with high probability we can find an integer $\hat{m}$ such that $p_{\hat{m}} \geq 1/(8 \cdot n^c \cdot (m+1))$ and $p_i < 1/(4 \cdot n^c \cdot (m+1))$ for all $i > \hat{m}$. (Note that we need only estimate p_i for i up to, say, n times the running time of Enc, because beyond that, p_i is exponentially small.)

We claim that $2^{\hat{m}}$ approximates $|W_x|$ to within a polynomial factor. For one direction, note that when we restrict to y's satisfying the condition $\mathrm{Enc}(x, \mathrm{Dec}(x,y,r), r) = y$, the mapping $y \mapsto \mathrm{Dec}(x,y,r)$ is injective. Thus,

$$|W_x| \geq p_{\hat{m}} \cdot |\{0,1\}^{\leq \hat{m}}| \geq \frac{2^{\hat{m}}}{8 \cdot n^c \cdot (m+1)}.$$

For the other direction, note that Markov's inequality implies that

$$\Pr_{w \xleftarrow{\mathrm{R}} X_x, r} [|\mathrm{Enc}(x,w,r)| \leq m+1] \geq 1 - \frac{m}{m+1} = \frac{1}{m+1}.$$

Therefore, the expected number of encodings of W_x with length at most $m+1$ is at least

$$\frac{|W_x|}{m+1} \geq \frac{2^m}{n^c} \cdot \frac{1}{m+1} > \frac{|\{0,1\}^{\leq m+1}|}{4 \cdot n^c \cdot (m+1)},$$

Hence $p_{m+1} \geq 1/(4 \cdot n^c \cdot (m+1))$

and thus with high probability, $\hat{m} \geq m+1 \geq H(X_x) - O(\log n)$ (by Lemma 2.5) and thus $2^{\hat{m}} \geq |W_x|/\mathrm{poly}(n)$. $\square$

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 following sense.

Definition 7.16. *Let R be a self-reducible* **NP** *relation, with corresponding functions $\ell : \Sigma^* \to \mathbb{N}$, $\sigma : \Sigma^* \to \mathbb{N}$, and $\rho : \Sigma^* \times \Sigma^* \to \Sigma^*$ as in Definition 7.1. We say that the sources $(X_x)_{x \in L_R}$* are compatible *with R (and ℓ, ρ, σ) if*

(i) The support of X_x is a subset of $\Sigma^{\ell(x)}$.

(ii) When $\ell(x) > 0$ (equivalently, $\sigma(x) > 0$), then for every $z \in \Sigma^{\sigma(x)}$ such that X_x has nonzero probability of having prefix z, the distribution of X_x conditioned on having prefix z is precisely $z \circ X_{\rho(x,z)}$.

The above conditions imply that that for every $x \in L_R$, the support of X_x is a subset of $W_x = \{w : (x, w) \in R\}$. It can be verified that setting X_x equal to the uniform distribution on W_x is compatible with R. An example of a non-uniform family of sources compatible with a self-reducible relation is the following generalization of Example 7.2:

Example 7.17. weighted perfect matchings Let R be the relation consisting of pairs $((G, w), M)$, where G is a bipartite graph *with positive real weights $w(e)$ on each edge* and M is a perfect matching in G. G and w are encoded by the $n \times n$ weighted adjacency matrix whose (i, j)'th entry is $w(i, j)$ if (i, j) is an edge, and 0 otherwise. This relation is self-reducible for the same reason as Example 7.2. We define the distribution $X_{G,w}$ to be the one where a perfect matching M is sampled with probability proportional to its weight $w(M) = \prod_{e \in M} w(e)$. (Note that the total weight $\sum_M w(M)$ equals the permanent of the weighted adjacency matrix.) It can be verified that these distributions are compatible with the self-reducibility of the relation R (e.g., when we remove an edge e and its endpoints, every perfect matching M in G that contains e becomes a perfect matching in $G \setminus \{e\}$ with weight $w(M \setminus \{e\}) = w(M)/w(e)$.) $\diamondsuit$

We can also compress such distributions:

Theorem 7.18. *Let R be a self-reducible relation, and let $(X_x)_{x \in L_R}$ be a family of sources compatible with R. If the sources $(X_x)_{x \in L_R}$ are approximately samplable, then they can be efficiently compressed to length $H(X_x) + 6$ with shared randomness and zero decoding error and to length $H(X_x) + O(\log n)$ with independent randomness and decoding error 2^{-n}. The encodings are prefix-free.*

Proof. The proof is identical to that of Theorem 7.3 and Proposition 7.13. The only use of the fact that X_x equals the uniform distribution on W_x is in the proof of Claim 7.7, specifically to establish Equation (7.8). By inspection, this equation holds for any family of sources compatible with R. $\square$

Many of the known approximate sampling algorithms for self-reducible relations generalize to natural non-uniform distributions that are compatible with the relation. Often, these distributions have interpretations in statistical physics (namely being the "Gibbs distribution" of some physical system). Some examples follow.

Corollary 7.19. *The following families of sources X_x can be efficiently compressed to length $H(X_x) + 6$ with shared randomness and zero decoding error, and to length $H(X_x) + O(\log n)$ with independent randomness and decoding error 2^{-n}:*

(i) $X_{G,w}$ *= perfect matchings in weighted bipartite graph* (G, w)*, as in Example 7.17 (a.k.a. the Gibbs distribution on a dimer system) (Jerrum* et al. *(2001)).*

(ii) $X_{G,w}$ *= matchings on a weighted graph* (G, w)*, where the weights are presented in unary (a.k.a. the Gibbs distribution on monomer-dimer systems) (Jerrum & Sinclair (1989)).*

(iii) $X_{G,\lambda}$ *= the weighted distribution on independent sets in graph* G*, where independent set* I *has weight* $\lambda^{|I|}$*, and the maximum degree of* G *is at most* $2/\lambda + 2$ *(a.k.a. the Gibbs distribution for the hard-core gas model) (Luby & Vigoda (1999)).*

Monte Carlo experiments in statistical physics estimate the expectation of various quantities in a physical system (such as the "mean energy") by randomly sampling configurations of the system (e.g., according to the Gibbs distribution). Compression algorithms such as in Corollary 7.19 could be possible to compactly store the configurations used in such experiments (e.g., for archival purposes, or to reuse the samples later).

Acknowledgements

Luca Trevisan's work was supported by NSF Grant CCR-9984783, US-Israel BSF grant 2002246, a Sloan Research Fellowship and an Okawa Foundation Grant. Salil Vadhan's work was supported by NSF grant CCR-0133096, US-Israel BSF grant 2002246, ONR grant N00014-04-1-0478, and a Sloan Research Fellowship. This work was done in part while he was fellow at the Radcliffe Institute for Advanced Study at Harvard University. David Zuckerman's work was supported in part by NSF Grants CCR-9912428 and CCR-0310960, a David and Lucile Packard Fellowship for Science and Engineering, a Radcliffe Institute Fellowship, and a Guggenheim Fellowship.

We thank Boaz Barak, Nenad Dedić, Ned Dimitrov, Troy Lee, and the anonymous referees for helpful comments and discussions.

References

Micah Adler & Michael Mitzenmacher (2001). Toward Compressing Web Graphs. In *Proceedings of the 2001 Data Compression Conference*.

Eric Allender, Danilo Bruschi & Giovanni Pighizzini (1993). The complexity of computing maximal word functions. *Computational Complexity* **3**(4), 368–391. ISSN 1016-3328.

Sanjeev Arora, Frank T. Leighton & Bruce M. Maggs (1996). On-Line Algorithms for Path Selection in a Nonblocking Network. *SIAM Journal on Computing* **25**(3), 600–625.

Boaz Barak, Ronen Shaltiel & Avi Wigderson (2003). Computational Analogues of Entropy. In *11th International Conference on Random Structures and Algorithms*.

Harry Buhrman, Troy Lee & Dieter van Melkebeek (2004). Language Compression and Pseudorandom Generators. In *Proceedings of the 19th Annual IEEE Conference on Computational Complexity*, 15–28.

Michael Capalbo, Omer Reingold, Salil Vadhan & Avi Wigderson (2002). Randomness Conductors and Constant-Degree Lossless Expanders. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing*, 659–668.

Benny Chor & Oded Goldreich (1988). Unbiased Bits from Sources of Weak Randomness and Probabilistic Communication Complexity. *SIAM Journal on Computing* **17**(2), 230–261.

Thomas M. Cover & Joy A. Thomas (1991). *Elements of Information Theory*. John Wiley & Sons, Inc.

Whitfield Diffie & Martin E. Hellman (1976). New Directions in Cryptography. *IEEE Transactions in Information Theory* **IT-22**(6), 644–654.

Andrew V. Goldberg & Michael Sipser (1991). Compression and Ranking. *SIAM Journal on Computing* **20**, 524–536.

Oded Goldreich & Salil Vadhan (1999). Comparing Entropies in Statistical Zero-Knowledge with Applications to the Structure of SZK. In *Proc. of Conference on Computational Complexity*, 54–73.

Johan Håstad, Russell Impagliazzo, Leonid A. Levin & Michael Luby (1999). A Pseudorandom Generator From Any One-Way Function. *SIAM Journal on Computing* **28**, 1364–1396.

Russell Impagliazzo (1999). Remarks in Open Problem session at the DIMACS Workshop on Pseudorandomness and Explicit Combinatorial Constructions.

Russell Impagliazzo & Avi Wigderson (1997). P = BPP if E Requires Exponential Circuits: Derandomizing the XOR Lemma. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, 220–229.

Mark Jerrum & Alistair Sinclair (1989). Approximating the Permanent. *SIAM Journal on Computing* **18**(6), 1149–1178.

Mark Jerrum & Alistair Sinclair (1996). The Markov Chain Monte Carlo Method: an Approach to Approximate Counting and Integration. In *Approximation Algorithms for NP-hard Problems*, D.S. Hochbaum, editor, chapter 12, 482–520. PWS Publishing.

Mark Jerrum, Alistair Sinclair & Eric Vigoda (2001). 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*, 712–721.

Mark R. Jerrum, Leslie G. Valiant & Vijay V. Vazirani (1986). Random Generation of Combinatorial Structures from a Uniform Distribution. *Theoretical Computer Science* **43**, 169–188.

Ravi Kannan (1994). Markov chains and polynomial time algorithms. In *Proceedings of the 35th Annual IEEE Symposium on Foundations of Computer Science*, 656–671. IEEE Comput. Soc. Press, Los Alamitos, CA.

Richard M. Karp, Michael Luby & Neal Madras (1989). Monte Carlo approximation algorithms for enumeration problems. *Journal of Algorithms* **10**(3), 429–448. ISSN 0196-6774.

Michael Kharitonov, Andrew V. Goldberg & Moti Yung (1989). Lower bounds for pseudorandom number generators. In *Proceedings of the 30th Annual IEEE Symposium on Foundations of Computer Science*, 242–247.

Ming Li & Paul Vitanyi (1997). *An Introduction to Kolmogorov Complexity.* Springer. 2nd ed.

Richard J. Lipton (1994). A New Approach to Information Theory. In *Proc. of 11th Symposium on Theoretical Aspects of Computer Science*, 699–708.

Michael Luby & Eric Vigoda (1999). Fast convergence of the Glauber dynamics for sampling independent sets. *Random Structures & Algorithms* **15**(3-4), 229–241. ISSN 1042-9832. Statistical physics methods in discrete probability, combinatorics, and theoretical computer science (Princeton, NJ, 1997).

Ben Morris & Alistair Sinclair (1999). Random walks on truncated cubes and sampling 0-1 knapsack solutions (preliminary version). In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science*, 230–240. IEEE.

Noam Nisan & Avi Wigderson (1994). Hardness vs. Randomness. *Journal of Computer and System Sciences* **49**, 149–167.

Noam Nisan & David Zuckerman (1996). Randomness is Linear in Space. *Journal of Computer and System Sciences* **52**(1), 43–52.

Rafail Ostrovsky (1991). One-Way Functions, Hard on Average Problems, and Statistical Zero-Knowledge Proofs. In *Proceedings of the Sixth Annual Structure in Complexity Theory Conference*, 133–138. IEEE Computer Society Press,, Chicago, Illinois.

Rafail Ostrovsky & Avi Wigderson (1993). One-Way Fuctions are Essential for Non-Trivial Zero-Knowledge. In *Proceedings of the Isreali Symposium on Theoretical Computer Science*, 3–17.

Dana Randall (2003). Mixing. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, 4–15. IEEE, Cambridge, MA.

Ran Raz & Omer Reingold (1999). On Recycling the Randomness of States in Space Bounded Computation. In *Proceedings of the 31st Annual ACM Symposium on Theory of Computing*, 159–168.

Omer Reingold, Salil Vadhan & Avi Wigderson (2000). Entropy Waves, the Zig-Zag Graph Product, and New Constant-Degree Expanders and Extractors. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science (FOCS '00)*, 3–13. IEEE, Redondo Beach, CA.

Amit Sahai & Salil Vadhan (2003). A complete problem for statistical zero knowledge. *Journal of the ACM* **50**(2), 196–249. Extended abstract in *FOCS '97*.

MIKLOS SANTHA & UMESH V. VAZIRANI (1986). Generating Quasi-Random Sequences from Semi-Random Sources. *Journal of Computer and System Sciences* **33**, 75–87.

CLAUS-PETER SCHNORR (1976). Optimal Algorithms for Self-Reducible Problems. In *Proceedings of the 3rd International Colloquium on Automata, Languages, and Programming*, 322–337.

CLAUDE E. SHANNON (1949). Communication theory of secrecy systems. *Bell System Technical Journal* **28**, 656–715.

ALISTAIR J. SINCLAIR & MICHAEL R. JERRUM (1989). Approximate Counting, Uniform Generation and Rapidly Mixing Markov Chains. *Information and Computation* **82**, 93–133.

AMNON TA-SHMA, CHRISTOPHER UMANS & DAVID ZUCKERMAN (2001). Loss-Less Condensers, Unbalanced Expanders, and Extractors. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing*, 143–152.

LUCA TREVISAN, SALIL VADHAN & DAVID ZUCKERMAN (2004). Compression of Samplable Sources. In *Proceedings of the 19th Annual IEEE Conference on Computational Complexity*, 1–14.

LUCA TREVISAN & SALIL P. VADHAN (2000). Extracting Randomness from Samplable Distributions. In *Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science*, 32–42.

LESLIE G. VALIANT & VIJAY V. VAZIRANI (1986). NP Is as Easy as Detecting Unique Solutions. *Theoretical Computer Science* **47**(1), 85–93.

HOETECK WEE (2004). On Pseudoentropy versus Compressibility. In *Proceedings of the 19th Annual IEEE Conference on Computational Complexity*, 29–41.

ANDREW C. YAO (1982). Theory and applications of trapdoor functions. In *Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science*, 80–91.

JACOB ZIV & ABRAHAM LEMPEL (1978). Compression of Individual Sequences by Variable Rate Coding. *IEEE Transactions on Information Theory* **24**, 530–536.

A. Expander Graphs

In this section, we sketch how Theorem 6.5 can be obtained from the techniques of Capalbo *et al.* (2002). In doing so, we assume familiarity with the notation and terminology of that paper. The expander graphs claimed in Theorem 6.5 are equivalent to explicit constructions of "$(k, \epsilon/2)$ lossless conductors" $\mathrm{E} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$, where n, d, m, k, ϵ are as in the statement of Theorem 6.5, and $\mathrm{E}(x, r)$ is the r'th neighbor of left-vertex x. Theorem 6.5 thus follows directly from Theorem 7.3 of Capalbo *et al.* (2002) (setting the parameter t to equal $t = n-k-c\log^3((n-k)/\epsilon)$), except that it does not claim the computation time of right-hand vertices. We explain how this follows from the construction below.

Computation Time of Right-Hand Vertices. The lossless conductors of Capalbo *et al.* (2002) are obtained via the zig-zag product of Reingold *et al.* (2000):

DEFINITION A.1 (zig-zag product). *Let* $\langle \mathrm{E}_1, \mathrm{C}_1 \rangle : \{0,1\}^{n_1} \times \{0,1\}^{d_1} \mapsto \{0,1\}^{m_1} \times \{0,1\}^{b_1}$, $\langle \mathrm{E}_2, \mathrm{C}_2 \rangle : \{0,1\}^{n_2} \times \{0,1\}^{d_2} \mapsto \{0,1\}^{d_1} \times \{0,1\}^{b_2}$, *and* $\mathrm{E}_3 : \{0,1\}^{b_1+b_2} \times \{0,1\}^{d_3} \mapsto \{0,1\}^{m_3}$ *be three functions. Set the parameters*

$$\begin{aligned} n &= n_1 + n_2, \\ d &= d_2 + d_3, \\ m &= m_1 + m_3 \end{aligned}$$

and define the zig-zag product

$$\mathrm{E} : \{0,1\}^n \times \{0,1\}^d \mapsto \{0,1\}^m$$

of these functions as follows: For any $x_1 \in \{0,1\}^{n_1}$, $x_2 \in \{0,1\}^{n_2}$,$r_2 \in \{0,1\}^{d_2}$ *and* $r_3 \in \{0,1\}^{d_3}$ *define*

$$\mathrm{E}(x_1 \circ x_2, r_2 \circ r_3) \stackrel{\text{def}}{=} y_1 \circ y_2, \text{ where}$$

$$\begin{aligned} \langle r_1, z_1 \rangle &\stackrel{\text{def}}{=} \langle \mathrm{E}_2, \mathrm{C}_2 \rangle (x_2, r_2) \\ \langle y_1, z_2 \rangle &\stackrel{\text{def}}{=} \langle \mathrm{E}_1, \mathrm{C}_1 \rangle (x_1, r_1), \text{ and} \\ y_2 &\stackrel{\text{def}}{=} \mathrm{E}_3(z_1 \circ z_2, r_3). \end{aligned}$$

In the proof of Theorem 7.3 of Capalbo *et al.* (2002), the above construction is applied with functions satisfying the following conditions:

- $m_1 = n_1$, $b_1 = d_1$, $\langle \mathrm{E}_1, \mathrm{C}_1 \rangle : \{0,1\}^{n_1} \times \{0,1\}^{d_1} \mapsto \{0,1\}^{m_1} \times \{0,1\}^{b_1}$ is a permutation, and both $\langle \mathrm{E}_1, \mathrm{C}_1 \rangle$ and $\langle \mathrm{E}_1, \mathrm{C}_1 \rangle^{-1}$ can be computed in polynomial time (in the the input length $n_1 + d_1$).

- Both $\langle \mathrm{E}_2, \mathrm{C}_2 \rangle$ and E_3 can be computed in polynomial time (in their input lengths).

- The parameters satisfy $b_1 = d_1 \leq n_2 = O(t + d + \log(1/\epsilon)) = O(n - k + d + \log(1/\epsilon))$ and $b_2 \leq n_2 + d_2$.

Given these facts, we can efficiently enumerate the elements of the set $\mathrm{E}^{-1}(y_1 \circ y_2)$ as follows:

- For all $r_2 \in \{0,1\}^{d_2}$, $r_3 \in \{0,1\}^{d_3}$, $x_2 \in \{0,1\}^{n_2}$, and $z_2 \in \{0,1\}^{b_1}$, do the following:

    1. Compute $\langle x_1, r_1 \rangle = \langle \mathrm{E}_1, \mathrm{C}_1 \rangle^{-1}(y_1, z_2)$.
    2. Verify that $\langle \mathrm{E}_2, \mathrm{C}_2 \rangle(x_2, r_2)$ is of the form $\langle r_1, z_1 \rangle$ for some $z_1 \in \{0,1\}^{b_2}$.
    3. Verify that $y_2 = \mathrm{E}_3(z_1 \circ z_2, r_3)$.
    4. If both verifications pass, output $x_1 \circ x_2$.

The computation time of this procedure is at most

$$\begin{aligned} 2^{d_2+d_3+n_2+b_1} \cdot (\mathrm{poly}(n_1, d_1) + \mathrm{poly}(n_2, d_2) + \mathrm{poly}(b_1 + b_2, d_3)) \\ &= 2^{O(d+n-k+\log(1/\epsilon))} \cdot \mathrm{poly}(n, d) \\ &= \mathrm{poly}(n, D_L, N/K), \end{aligned}$$

where in the last inequality we use the fact that $D_L = 2^d > 1/\epsilon$.

Manuscript received December 31, 2004

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