Making Hard Problems Harder

Joshua Buresh-Oppenheim[*]
Simon Fraser University
jburesho@cs.sfu.ca

Rahul Santhanam
Simon Fraser University
rsanthan@cs.sfu.ca

January 8, 2006

Abstract

We consider a general approach to the hoary problem of (im)proving circuit lower bounds. We define notions of hardness condensing and hardness extraction, in analogy to the corresponding notions from the computational theory of randomness. A hardness condenser is a procedure that takes in a Boolean function as input, as well as an advice string, and outputs a Boolean function on a smaller number of bits which has greater hardness when measured in terms of input length. A hardness extractor takes in a Boolean function as input, as well as an advice string, and outputs a Boolean function defined on a smaller number of bits which has close to maximum hardness. We prove several positive and negative results about these objects.

First, we observe that hardness-based pseudo-random generators can be used to extract deterministic hardness from non-deterministic hardness. We derive several consequences of this observation. Among other results, we show that if $\mathsf{E}/O(n)$ has exponential non-deterministic hardness, then $\mathsf{E}/O(n)$ has deterministic hardness $2^n/n$, which is close to the maximum possible. We demonstrate a rare downward closure result: E with sub-exponential advice is contained in non-uniform space $2^{\delta n}$ for all $\delta > 0$ if and only if there is $k > 0$ such that P with quadratic advice can be approximated in non-uniform space n^k.

Next, we consider limitations on natural models of hardness condensing and extraction. We show lower bounds on the length of the advice required for hardness condensing in a very general model of "relativizing" condensers. We show that non-trivial black-box extraction of deterministic hardness from deterministic hardness is essentially impossible.

Finally, we prove positive results on hardness condensing in certain special cases. We show how to condense hardness from a biased function without any advice, using a hashing technique. We also give a hardness condenser without advice from average-case hardness to worst-case hardness. Our technique involves a connection between hardness condensing and certain kinds of explicit constructions of covering codes.

1 Introduction

Proving circuit lower bounds for natural problems is one of the most fundamental problems in complexity theory. Showing superpolynomial circuit lower bounds for a language in non-deterministic polynomial time would separate non-deterministic polynomial time from deterministic polynomial time, and showing superpolynomial circuit lower bounds for a language in linear exponential time would yield subexponential time deterministic simulations of probabilistic algorithms via pseudo-random generators. However, as is typical in complexity theory, "fundamental" is synonymous with "fiendishly difficult": no superlinear lower bounds are known for any function in linear exponential

---

time. Indeed, the possibility that all of non-deterministic exponential time can be solved by depth-3 polynomial size circuits with AND, OR and MOD_6 gates is consistent with our current state of knowledge.

There have been attempts to explain the lack of success in coming up with strong lower bound techniques. Perhaps the most compelling effort in this direction is due to Razborov and Rudich [RR97]. They showed that several popular combinatorial and algebraic lower bound proofs, such as those using the polynomial technique of Razborov [Raz87] and Smolensky [Smo93], or the approximation technique of Razborov [Raz89], fall under the category of "natural proofs". A natural proof of a lower bound satisfies two conditions, the "largeness" condition and the "constructivity" condition. The largeness condition says that the proof of the lower bound holds not just for any specific function but for most functions. The constructivity condition says that the set of truth tables of functions for which the lower bound technique works is in some reasonably efficient non-uniform complexity class (say, polynomial-size circuits). The evidence against existence of natural proofs comes from the fact that they can be used to break pseudo-random function generators. A pseudo-random function generator is a procedure producing a small family of "pseudo-random" Boolean functions on n bits such that no feasible test can distinguish a random function in this family from a function chosen uniformly in the space of all Boolean functions on n bits. Since there are constructions of such generators based on widely believed hardness assumptions such as the hardness of the discrete logarithm function [GGM86], natural proofs of strong circuit lower bounds are unlikely to exist.

Thus, while the circuit lower bound problem is fundamental, there are significant obstructions to making progress on it using current techniques. Given this impasse, rather than make a direct attack on the problem, we consider the following situation. Imagine that we are given a Boolean function f that does not have circuits of size $s(n)$ on inputs of length n. Is there a procedure that computes from f, perhaps by "compressing" the truth table of f in some non-trivial way, a function f' on fewer input bits which is harder as a function of its input length? The existence of such a procedure could be valuable in proving (or at least improving) circuit lower bounds. The uniform complexity of f' as obtained by applying the procedure would be greater than the uniform complexity of f since the input length of f' is smaller, but we do not mind this too much as long as we gain in hardness.

We call such a procedure a "hardness condenser". In this paper, we formally define hardness condensers and a strong version thereof called "hardness extractors", which output functions that are close to maximally hard in terms of their input length. We explore the conditions under which hardness condensers and hardness extractors exist, as well as the limitations on constructing them in a natural way. We give a more detailed description of our results in Subsection 1.1.

The notions we define are closely analogous to, and indeed inspired by, the notions of "randomness condensing" and "randomness extraction" from the computational theory of randomness. Randomness condensers are explicit procedures which take as input a sample from a distribution containing a certain amount of randomness (as measured by the min-entropy of the distribution), together with a short random seed, and output a sample from a distribution which has greater min-entropy as a function of its input length. Randomness extractors are strong versions of randomness condensers in which the output distribution is close to uniform. There is a long line of research (see [Sha02] for a survey) on randomness extractors, and explicit constructions of these objects with close to optimal parameters are known. The study of randomness condensers does not have as rich a history but there has recently been a lot of interesting work [TSUZ01, BIW04, BKS$^+$05, Raz05] on constructions of these objects.

The fact that we draw an analogy between the theory of randomness and the theory of hard-

ness is far from accidental. The theory of pseudo-randomness is based on the connection between hardness and randomness: if a function is hard to approximate by polynomial-size circuits, then a random input to the function together with the value of the function on that input "appears" random to any polynomial-size circuit. This idea was originally used [BM84, Yao82] to construct pseudo-random generators for cryptographic applications. Here a pseudo-random generator is a procedure mapping short random strings to longer "pseudo-random" strings such that no polynomial-size circuit can distinguish between random strings and strings drawn uniformly from the output of the generator. Cryptographic pseudo-random generators are required to be computable in polynomial time. Nisan and Wigderson [NW94] constructed "complexity-theoretic" generators for which the efficiency criterion for computing the generator is relaxed to exponential time in the length of the seed. The Nisan-Wigderson generator yields pseudo-randomness based on *any* hard function in E. Following on the Nisan-Wigderson result, a series of papers [NW94, Imp95, IW97, ISW99, SU01, Uma02] have established a tight relationship between circuit lower bounds in E and complexity-theoretic pseudo-random generators: E requires circuits of size $s(n)^{\Omega(1)}$ if and only if there is a complexity-theoretic pseudo-random generator mapping a seed of $O(n)$ bits to $s(n)^{\Omega(1)}$ pseudorandom bits.

Thus the problem of proving circuit lower bounds in E is reduced to the problem of constructing good pseudo-random generators. However, it might seem that this does not help us in terms of techniques to prove hardness of E since essentially the only known technique for constructing good pseudo-random generators is to prove hardness of some function and use it in a hardness-based generator. We break this cycle by observing that pseudo-random generators can be interpreted as hardness extractors of a certain kind, This observation can be used to derive non-trivial relationships between uniform and non-uniform classes by exploiting properties of known constructions of hardness-based pseudo-random generators (more details in Subsection 1.1).

Given that the natural application of hardness extractors is to proving circuit lower bounds, the question immediately arises if the Razborov-Rudich obstacle applies to these objects. Suppose there is a natural proof of hardness for the input function to a hardness condenser. Does this imply a natural proof of hardness for the output function? It is an encouraging feature of our approach that neither condition required of a natural proof seems to be preserved by the hardness condensing transformation. Since many input functions can be mapped to the same output function by the hardness condenser, the fact that a proof of hardness works for most input functions does not imply a proof of increased hardness for most output functions. Thus the largeness condition is not necessarily preserved. Also, the existence of a small circuit recognizing truth tables of input functions for which the proof of hardness works does not imply the existence of a small circuit recognizing truth tables of output functions. This is because the hardness condenser may be a function which is hard to invert; indeed, the use of a pseudo-random generator as a hardness condenser drives home this point in a strong way.

Having provided the context for our approach, we move on to a description of our results.

1.1 Our Results

We formalize the notions of "hardness condenser" and "hardness extractor", and prove three kinds of results: explicit constructions of hardness extractors with small advice, limitations on constructing relativizing and black-box hardness condensers, and hardness condensers without advice for special types of functions, namely biased functions and average-case hard functions.

The ideal construction of a hardness condenser would be completely uniform and have the property that if the input function f is hard for deterministic circuits of a certain size, then the

output function f' is maximally hard for deterministic circuits. We do not know how to achieve this. However, we can achieve something close. The caveats are firstly that the hardness condenser is not completely uniform but instead requires a small amount of advice, and secondly that we only prove that the output function is almost maximally hard for deterministic circuits if the input function is hard for *non-deterministic* circuits.

Our construction follows from the observation that relativizing hardness-based pseudo-random generators [NW94, Uma02] can be interpreted as hardness extractors from non-deterministic hardness to deterministic hardness. The basic idea is to interpret the output of the pseudo-random generator on random seed y as the truth table of a function f'_y. If f'_y has deterministic circuits of size $2^n/n$ for each y, then we can define a non-deterministic circuit that distinguishes the output of the generator from uniform by guessing a circuit of size $2^n/n$ and checking that the output is the truth table corresponding to the circuit. This circuit is indeed a distinguisher because the truth table of a random function has circuit complexity more than $2^n/n$. The existence of such a non-deterministic distinguisher implies that the function on which the generator is based has small non-deterministic circuits, contradicting the assumption on the input function. Thus it must be the case that the output of the generator is almost maximally hard for some value of the advice.

We remark that the basic idea of interpreting the output of a pseudo-random generator as the truth table of a function and arguing about the circuit complexity of the function is not new. It has appeared in several contexts [KC00, Kab01, IKW02, All01]. However, our perspective on it in the context of hardness extraction and the consequences we derive from it do seem to be new.

The hardness extractors we obtain in this way have interesting consequences for complexity theory. For instance, we can show that if there is a Boolean function in E with linear advice which has high non-deterministic circuit complexity, then there is a function in E with linear advice which has almost maximum deterministic circuit complexity.

Theorem 1. *If there is a constant $\epsilon > 0$ such that $\mathsf{E}/O(n) \not\subseteq \mathsf{NSIZE}(2^{\epsilon n})$, then $\mathsf{E}/O(n) \not\subseteq \mathsf{SIZE}(2^n/n)$.*

By using the Nisan-Wigderson generator, which is very efficiently computable, as a hardness extractor, and considering hardness with respect to non-uniform space rather than non-uniform time, we derive a rare equivalence between simulations of linear exponential time with advice and simulations of polynomial time with advice.

Theorem 2. *(Informal) There is a constant k such that P/n^2 can be approximated on a $1/2 + 1/n^k$ fraction of inputs of length n in non-uniform space n^k if and only if for all constants $\delta > 0$ there is a constant $\epsilon > 0$ such that $\mathsf{E}/2^{\epsilon n}$ is contained in non-uniform space $2^{\delta n}$*

One can ask whether there is a fundamental reason why our hardness extractors take advice or whether it is just an artifact of our specific technique. We provide evidence that the former is the case by considering a very general model of "relativizing" hardness condensers and showing lower bounds on the length of advice required for such hardness condensers. A significant aspect of our lower bounds is that the model of hardness for the input function and the model of hardness for the output function of the hardness condenser are not required to be the same. Thus, for instance, our lower bounds also hold for the hardness extractors constructed from pseudo-random generators above.

Theorem 3. *(Informal) Relativizing condensers which take as input any function on n bits having hardness s and output a function on n' bits with hardness s' must take advice of length $\Omega(n - n' - \log s)$.*

4

We also consider a natural "black-box" model of hardness condensing where the output function must be hard for deterministic circuits if the input function is hard for deterministic circuits, and show using Theorem 3 that non-trivial hardness condensing is essentially impossible in this model.

Finally, we consider some special cases in which hardness condensing is possible without advice. The first case we consider is that of biased functions, i.e., functions where there is an imbalance between the number of inputs that evaluate to 0 and the number of inputs that evaluate to 1. We use a technique involving repeated hashing to prove the following condensing result for such functions.

Theorem 4. *(Informal) Let f be an n-bit function such that f is 1 with probability α and requires circuit size s. Then f can be condensed to a function on $n - \Omega(\frac{1}{H(\alpha)})$ bits that requires circuits of size $\Omega((s-n)/n)$.*

The second case we consider is that of average-case hard functions, i.e., functions for which no small circuit can approximate the function on more than a $1/2 + \delta$ fraction of inputs, for some small function δ. We show how to obtain from such a function another function whose input length is smaller which has essentially the same hardness in the worst-case. Our technique involves a connection between hardness condensing in this case and constructions of certain explicit kinds of covering codes. The constructions of covering codes we obtain may be of independent interest.

Theorem 5. *(Informal) Let f be an n-bit function such that no circuit of size s can compute f on a $1/2 + \delta$ fraction of the inputs. Then f can be condensed to a function on $n - (2 - o(1)) \log \frac{1}{\delta}$ bits that cannot be computed by circuits of size s/n^c for some constant $c > 0$.*

2 Preliminaries

2.1 Functions and complexity models

We will abuse notation by writing $f : m \to n$ to denote $f : \{0,1\}^m \to \{0,1\}^n$. We often identify a Boolean function f on n bits with its truth table, which is a string of size 2^n. It will be clear from the context whether we are referring to the function or its truth table.

In general we consider circuits with only one output. We refer the reader to [BDG88] for definitions of deterministic, nondeterministic and oracle circuits.

An *alternating circuit* on n inputs is a dag with $n+m$ sources labelled $x_1, \ldots, x_n, (y_1, i_1), \ldots (y_m, i_m)$. For each j, i_j is either 0 or 1 corresponding respectively to "existential" and "universal" gates. Let the quantifier Q_j be $\exists$ if $j = 0$ and $\forall$ if $j = 1$. The circuit A computes a function in the following way: for a 0/1 setting to x_j's, we say $f(x) = 1$ if and only if $Q_{i_1} y_1 \ldots Q_{i_m} y_m A(x_1 \ldots x_n, y_1 \ldots y_m)$.

Given a function $f : n \to 1$, we say that f requires circuits (respectively, non-deterministic circuits, oracle circuits, etc.) of size s if there is no circuit of size $< s$ that computes f. We will also say f is worst-case hard (or simply "hard") for circuits of size s to mean the same thing. For $0 \leqslant \delta < 1/2$, we say that f is δ-hard for circuits of size s if there is no circuit of size $\leq s$ that computes a function that agrees with f on more than a $1 - \delta$ fraction of the inputs.

It is well-known that that almost all functions require circuits of size more than $2^n/n$ ([Lup59]) and that every function can be computed by circuits of size $\frac{2^n}{n}(1 + O(\log n/n))$ ([FM05]). Hence, functions that require circuits of size $2^n/n$ are essentially as hard as possible.

A language L is a subset of $\{0,1\}^*$. For any n, L_n is the characteristic function of L restricted to $\{0,1\}^n$. A language L' δ-approximates L if, for each n, L'_n agrees with L_n on at least a δ fraction of inputs.

For any function $s : \mathbb{N} \to \mathbb{N}$, $\mathsf{SIZE}(s)$ is the class of languages L such that for each n, L_n has deterministic circuits of size $s(n)$. $\mathsf{NSIZE}(s)$ is the class of languages L such that for each n, L_n has non-deterministic circuits of size $s(n)$. $\mathsf{ASIZE}(s)$ is the class of languages L such that for each n, L_n has alternating circuits of size $s(n)$.

For a uniform complexity class $\mathcal{C}$ and a function $a : \mathbb{N} \to \mathbb{N}$, C/a is the class of languages L decidable in $\mathcal{C}$ with advice length a.

For a complexity class $\mathcal{C}$ and a function $s : \mathbb{N} \to [0,1]$, $heur_s - \mathcal{C}$ is the class of languages L for which there is a language $L' \in \mathcal{C}$ such that L' s-approximates L.

2.2 Hardness Condensers and Extractors

We begin by formally defining hardness condensers.

Definition 6 ((Hardness Condenser)). *Let $\mathcal{A}$ and $\mathcal{B}$ be complexity models (eg. deterministic circuits, nondeterministic circuits, circuits with oracle D, formulas, branching programs etc.). An (n, s, n', s') hardness condenser with advice length ℓ from worst-case (resp. δ-average-case) hardness for $\mathcal{A}$ to worst-case (resp. δ'-average-case) hardness for $\mathcal{B}$ is a function*

$$C_n \ : \ 2^n \times \ell \longrightarrow 2^{n'}$$

with $n' < n$ such that if $f \ : \ n \to 1$ requires (resp. is δ-hard for) size s in $\mathcal{A}$, then there is a string $y \in \{0,1\}^l$ for which $C_n(f, y)$ requires (resp. is δ'-hard for) size s' in $\mathcal{B}$.

An important issue when studying hardness condensers is the efficiency of computing them. Clearly, (n, s, n', s') hardness condensers exist nonuniformly whenever s' is less than the maximum possible complexity of a Boolean function on n' bits, if we fix the complexity models $\mathcal{A}$ and $\mathcal{B}$: a Boolean function on n' bits with hardness s' can be encoded into the condenser, and the condenser could just output that function, ignoring the input function and advice. Thus only uniform constructions of condensers are interesting (unless we consider stronger versions of condensers which work for families of complexity models, as in Section 4). Hence, in future we consider families C of (n, s, n', s') hardness condensers $C_n, n \in \mathbb{N}$ with advice length ℓ, where s, n', s', l are functions of n, unless otherwise stated. We define two notions of efficiency: the first one corresponds to the complexity of computing the entire output truth table given the input truth table and advice, while the second corresponds to the complexity of computing any individual bit of the output truth table from the advice given oracle access to the input truth table.

Definition 7. *An (n, s, n', s') hardness condenser C with advice length ℓ is* explicit *if C is computable in time polynomial in $2^n + \ell$.*

Note that we do not need to take the size of the output into consideration when giving our criterion of efficiency since $|C(F, y)| < |F|$ for any hardness condenser C.

Definition 8. *An (n, s, n', s') hardness condenser C with advice length ℓ is* super-explicit *if there is a procedure computing the ith bit of $C(F, y)$ in time polynomial in $\ell + 2^{n'}$ when given oracle access to F.*

This is a much stronger criterion of efficiency since it is possible that n' ¡¡ n.

To give some intuition for the parameters in Definition 6, we explore some special cases.

The first case we explore is where the advice length is very large, larger than the hardness required of the output function. For this setting of parameters, hardness condensers exist trivially.

Proposition 9. *There is a super-explicit (n, s, n', s') hardness condenser C with advice length ℓ from worst-case hardness for deterministic circuits to worst-case hardness for deterministic circuits if $\ell/(2\log(\ell/2)) > s'$ and $2^{n'}$ is time-constructible.*

Proof. C just copies its advice string into the first ℓ bits of its output and sets all the other bits of the output to 0. It is not hard to see that C is super-explicit. Let ℓ' be the largest power of 2 less than ℓ. Then there is some Boolean function f on $\log(\ell')$ bits which requires circuits of size $\ell'/\log(\ell') > \ell/(2\log(\ell/2)) > s'$. Now let $y \in \{0,1\}^\ell$ be some string with prefix f. Clearly, $C(F, y)$ requires circuits of size s' for any $F \in \{0,1\}^{2^n}$. $\qquad\square$

Proposition 9 shows that "interesting" advice lengths for hardness condensers should be less than the hardness of the output function. Indeed, for the most interesting application of hardness condensers, where we have a function in some uniform class with high circuit complexity and wish to derive a function with higher circuit complexity by feeding the original function to a hardness condenser, if the advice length is too large, the output function is too non-uniform and hence we do not obtain anything nontrivial.

The second special case is when the hardness required of the output function is very small, i.e., logarithmic in the size of the original truth table. In this case, hardness condensing can be accomplished by exhaustive search.

Proposition 10. *There is an explicit (n, s, n', s') hardness condenser with no advice from worst-case deterministic hardness to worst-case deterministic hardness if $s' = O(n)$ and n' is time-constructible.*

We would like (n, s, n', s') hardness condensers with advice length l where l is as small as possible, s' is as close to s as possible, and n' is as close to $\log(s')$ as possible. These requirements have different motivations: l as small as possible means the construction is as uniform as possible, s' as close to s as possible means that most of the hardness of the input function is retained in the output function, and n' as close to $\log(s')$ as possible means the output function has close to maximum possible hardness as a function of its input length.

In analogy with the computational theory of randomness, we call a hardness condensers with close to ideal parameters a "hardness extractor".

Definition 11 ((Hardness Extractor)). *An (n, s) hardness extractor is an (n, s, n', s') hardness condenser with advice length $O(\log(n))$ for which $n' = \Omega(\log(s)$ and $s' \geqslant 2^{n'}/n'$.*

2.3   Pseudo-random generators

Pseudo-random generators are functions mapping short random seeds to "pseudo-random" strings such that no efficient adversary can distinguish a random string from a string in the range of the generator. There is a long line of research [[NW94, IW97, KvM02, SU01, Uma02]] on deriving pseudo-random generators from "hard" functions, where the hardness required of the function is related to the efficiency of the adversary. Our main construction of hardness condensers are actually pseudo-random generators. To make the connection more transparent, we define pseudo-random generators in close correspondence to hardness extractors :

Definition 12. *(Pseudo-random generators) Let $\mathcal{A}$ be a complexity model. An (n, s, m) PRG with error parameter ϵ and seed length l based on hardness (resp. δ average-case hardness) for $\mathcal{A}$ is a*

function $G : 2^n \times \ell \to m$ such that if f is a Boolean function on n bits which requires (resp. is δ-hard for) size s in $\mathcal{A}$, then for any Boolean function C on m bits which has size m in $\mathcal{A}$,

$$| \Pr_{x \sim \{0,1\}^\ell}[C(G(F,x)) = 1] - \Pr_{x \sim \{0,1\}^m}[C(x) = 1]| \leq \epsilon$$

This definition differs from the standard definition of PRGs in that we only consider hardness-based PRGs, i.e., PRGs that can be based on *any* hard function. All known PRGs based on hardness for deterministic circuits or more powerful classes are hardness-based.

In future, we assume that the output length of the generator is a power of 2. Any PRG can be modified to satisfy this condition without changing the other parameters by only outputting the first m' bits, where m' is the highest power of 2 less than m.

Again, we look at infinite families of PRGs, one for each n, and call the family *explicit* if G_n can be computed in time polynomial in $2^n + \ell$. Again, it's super-explicit if each bit of its output can be computed in time polynomial in $m + \ell$ with oracle access to the first input F.

There are several known constructions of hardness-based PRGs. In this paper, we will mainly be concerned with a construction due to Nisan and Wigderson [NW94], and a construction due to Umans [Uma02].

The Nisan-Wigderson PRG was originally based on average-case hardness for deterministic circuits. Arvind and Kobler [AK97] observed that the construction could also be based on average-case hardness for non-deterministic circuits. An important property of this construction for our applications is that it is super-explicit - each output bit is a bit in the input truth table whose index can be determined from the seed in polynomial time.

Theorem 13 ([NW94, AK97]). *Let $s(n)$ be any constructible function. There is a super-explicit $(n, s, \sqrt{s})$ PRG G_{NW} with error parameter ϵ and seed length $O(n^2/\log(s))$ based on $1/2 - \epsilon/s$ average-case hardness for non-deterministic circuits.*

The PRG due to Umans does not satisfy such a strong efficiency criterion; however, it is based on worst-case hardness rather than average-case hardness.

Theorem 14 ([Uma02]). *Let $s(n)$ be any constructible function. There is an explicit $(n, s, s^{\Omega(1)})$ PRG G_U with error parameter $1/s$ and seed length $O(n)$ based on worst-case hardness for deterministic circuits.*

3 Extracting deterministic hardness from non-deterministic hardness

3.1 A little non-deterministic hardness goes a long way

In this section, we prove that pseudo-random generators against non-deterministic adversaries are hardness extractors from hardness for non-deterministic circuits to hardness for deterministic circuits. In particular, the PRG in Theorem 13 is a super-explicit hardness extractor from average-case hardness, and the PRG in Theorem 14 is an explicit hardness extractor from worst-case hardness.

Lemma 15. *If G is an $(n, s, s^{\Omega(1)})$ PRG with error parameter $1 - 1/s$ and seed length l based on worst-case (resp. δ average-case) hardness for non-deterministic circuits, then G is an (n, s)-hardness extractor with advice length l from worst-case (resp. δ average-case) hardness for non-deterministic circuits to worst-case hardness for deterministic circuits.*

Proof. The proof is straightforward in contrapositive form. Interpreting the output of G as the truth table of a Boolean function f', if f' had small circuit size, a non-deterministic adversary could guess a small circuit for f'. It could then verify that the circuit is correct by running the circuit for each possible input and checking that the output is the same as the corresponding bit of the truth table of f'. This would distinguish the output from random since a random truth-table will have very high circuit complexity.

More formally, let G be an (n, s, s^c) PRG with error parameter $1 - 1/s$ and seed length l based on worst-case hardness for non-deterministic circuits, for some constant $c > 0$. Now consider the following non-deterministic test A. Given an input x of length m (where m is a power of 2), A first guesses a deterministic circuit B of size $m/\log(m)$. Then, interpeting x as the truth table of a function f_x on $\log(m)$ bits, A checks that the circuit B accepts f_x. It does this by cycling over possible values of i between 1 and m, and for each such i checking that x_i is the same as the output of B on input i. A single such check can be done in time $O(m)$ and hence all the checks can be done in time $O(m^2)$. If all the checks accept, A accepts, otherwise it rejects. The time taken by A is the time required to guess B plus the time required to perform all checks, which is $O(m^2)$. Thus A can be implemented as a non-deterministic circuit of size $am^2 \log(m)$, for some constant a.

Now, set m to be the largest power of 2 such that $am^2 \log(m) < s^c$. A accepts an input x of length m if and only if f_x has circuits of size $m/\log(m)$. Since a random function on $\log(m)$ bits has circuit complexity less than $m/\log(m)$ with probability at most $1/m^{\omega(1)}$, the probability that A accepts on a random input is bounded above by $1/s$. If A accepts whenever it is fed the first m bits of an output of G, then A is a non-deterministic test of size s^c distinguishing the output of G from random with probability at least $1 - 1/s$. Since G is a pseudo-random generator based on worst-case non-deterministic hardness, this implies that the function on which G is based has non-deterministic circuits of size s. Contrapositively, if the function on which G is based does not have non-deterministic circuits of size s, there must be some seed of length l for which the first $m = s^{\Omega(1)}$ bits of the output of G constitute the truth table of a function that does not have deterministic circuits of size $m/\log(m)$. This is just a re-statement of what it means to be an (n, s) hardness extractor with advice length l from worst-case hardness for non-deterministic circuits to worst-case hardness for deterministic circuits.

Precisely the same argument works to establish that a PRG based on average-case non-deterministic hardness is a hardness extractor from average-case hardness for non-deterministic circuits. $\square$

Lemma 15 shows that pseudo-random generators yield hardness extractors, but in fact hardness extractors correspond more naturally to *hitting-set generators* which are only required to hit any efficiently recognizable sets of high enough density rather than approximate such sets. It is known that hitting-set generators and pseudo-random generators are essentially equivalent for most interesting complexity models, hence we use the strongest known constructions of PRGs here. As a consequence, our hardness extractors yield very hard functions for most values of the advice.

Lemma 15 works equally well for extracting average-case hardness from average-case hardness. However, the parameters we obtain for such an adaptation depend on the hardness of a random function; we stick to extraction of worst-case hardness in this paper because the parameters are simpler to state.

It follows from Lemma 15 that the Nisan-Wigderson generator and the Umans generator are hardness extractors.

Corollary 16. G_{NW} *is a super-explicit (n, s) hardness extractor with advice length $n^2/\log(s)$ from $1/2 - 1/s$ average-case hardness for non-deterministic circuits to hardness for deterministic circuits.*

Corollary 17. G_U *is an explicit* (n, s) *hardness extractor with advice length* $O(n)$ *from hardness for non-deterministic circuits to hardness for deterministic circuits.*

An interesting application of Corollary 17 is that if E with linear advice has a Boolean function of exponential non-deterministic circuit complexity, then E with linear advice has a Boolean function of close-to-maximum deterministic circuit complexity. The following theorem is a restatement of Theorem 1.

Theorem 18. *If there is a constant* $\epsilon > 0$ *such that* $\mathsf{E}/O(n) \not\subseteq \mathsf{NSIZE}(2^{\epsilon n})$, *then* $\mathsf{E}/O(n) \not\subseteq \mathsf{SIZE}(2^n/n)$.

Proof. Let f be a Boolean function requiring non-deterministic circuits of size $2^{\epsilon n}$ for some $\epsilon > 0$. The basic idea is to apply the hardness extractor G_U to f to obtain a new function f' with close-to-maximum deterministic hardness. Using explicitness of G_U, we would ideally like to show $f' \in \mathsf{E}$ if $f \in \mathsf{E}$. Two issues arise when trying to show this. First, we do not know how to generate f' uniformly from f. We only know that for some advice string y, the output function f' of the hardness extractor is very hard. Secondly, when we are trying to compute f' on some input length n', it is not immediately clear which input length n we should evaluate f at. Since the hardness extractor effectively compresses truth tables, many different input lengths n map to the same input length n'. The function f may be easy on some input lengths and hard on others. We need to evaluate f at an input length where it is hard in order to ensure the hardness extraction property helps us.

The solution to both issues is to use a small amount of advice when computing f'. The advice will tell us which input length n to evaluate f at, as well as which advice string for the hardness extractor results in an output function that is very hard.

Now assume $f \in \mathsf{E}/O(n)$. We apply Corollary 17 with $s = 2^{\epsilon n}$. Since G_U is an $(n, 2^{\epsilon n})$ hardness extractor, there is some constant $\epsilon' > 0$ such that G_U is an $(n, 2^{\epsilon n}, n' = \lfloor \epsilon' n \rfloor, 2^{n'}/n')$ hardness condenser from hardness for non-deterministic circuits to hardness for deterministic circuits. We define f' implicitly by defining an advice-taking machine M running in linear exponential time and a sequence of advice strings $\{a_m\}, m = 1 \ldots \infty$ such that for each m, M with advice a_m decides f'_m.

On input length m, the advice string a_m consists of three parts. The first part of the advice string codes an integer n such that $m = \lfloor \epsilon n \rfloor$. This part of the advice is of a fixed constant size, since there are only $\lceil 1/\epsilon' \rceil$ possible values for n. The second part of the advice is a string y which is of length $O(n)$, corresponding to the advice string for G_U. The third part is a string z which is also of length $O(n)$, corresponding to the advice required for computing f.

The machine M operates as follows on input x: after determining n and z from its advice, it generates the truth table of f_n. This can be done in linear exponential time since $f \in \mathsf{E}/O(n)$ and z is the correct advice string for computing f on inputs of length n. Then M determines y from its advice and computes $G_U(f_n, y)$. This can be done in linear exponential time by explicitness of G_U. Finally, M reads off $f'(x)$ from the truth table output by G_U.

Clearly, $|a_m| = O(m)$, thus $f' \in \mathsf{E}/O(m)$. We argue that if there are infinitely many n such that f_n does not have circuits of size $2^{\epsilon n}$, then there are infinitely many m such that f'_m does not have circuits of size $2^m/m$. We do this by specifying which advice string to pick for each m. If there is n such that $m = \lfloor \epsilon' n \rfloor$ and f_n does not have circuits of size $2^{\epsilon n}$, we pick the first such n and code it into the first part of the advice. If not, the first part of the advice is arbitrary. For the third part of the advice, we pick a string z which is the correct advice for computing f in linear exponential time on inputs of length n.

In the second part of the advice, if there is some y such that $G_U(f_n, y)$ requires circuits of size $2^m/m$, we code the first such y. If f_n is hard enough, such a string y exists since G_U is a hardness extractor. Now if there are infinitely many n for which f_n are hard, there are infinitely many m for which the first part of the advice codes a hard input length for f. In this case, by the hardness extraction property of G_U, there is some string y such that $G_U(f, y)$ requires circuits of size $2^m/m$. Thus it follows that f' requires circuits of size $2^m/m$ for infinitely many m. $\qquad\square$

We could get rid of the non-uniformity in the definition of f' above if we could argue that $\mathsf{E}/O(n) \not\subseteq 2^n/n$ implies $\mathsf{E} \not\subseteq 2^n/n - O(n)$. However, standard arguments only give that $\mathsf{E}/cn \not\subseteq 2^n/n$ implies $\mathsf{E} \not\subseteq 2^{n/c}/n - O(n)$, which does not yield any non-trivial improvement in hardness in this context. In Section 4, we show that the introduction of non-uniformity is intrinsic to the approach to hardness extraction we take in this section.

We can derive a result where the extracted function is uniform by moving to a higher uniform class.

Theorem 19. *If there is a constant $\epsilon > 0$ such that $\mathsf{E} \not\subseteq \mathsf{NSIZE}(2^{\epsilon n})$, then $\mathsf{E}^{\mathsf{NP}} \not\subseteq \mathsf{SIZE}(2^n/n)$.*

Proof Sketch. Applying Theorem 18, we have $f' \in \mathsf{E}/O(n)$ such that $f' \notin SIZE(2^n/n)$. We can now derive a function $f'' \in \mathsf{E}^{\mathsf{NP}}$ with hardness $2^n/n$ by cycling over all advice strings and using the first one which yields a function with hardness $2^n/n$. Checking whether the output on a specific advice string has hardness $2^n/n$ is an NP question. $\qquad\square$

3.2 Downward closures

We use Corollary 16 to show some rare downward closure results. It is common in complexity theory, and usually straightforward, to show that inclusion of one complexity class in another can be translated "upward", meaning that the inclusion also holds if the classes are given proportionately more resources. For instance, if $\mathsf{NP} = \mathsf{P}$, then $\mathsf{NEXP} = \mathsf{EXP}$, and the proof is a simple padding argument. However, interesting examples of the converse, i.e., showing that inclusion of one class in another implies an inclusion where the classes have proportionately fewer resources (this is equivalent to translating separations of classes "upward"), are much rarer. The only non-trivial examples we know are the result of Impagliazzo and Naor [IN88] showing that if $\mathsf{NP} = \mathsf{P}$, then $\mathsf{NPOLYLOG} \cap \mathsf{coNPOLYLOG} = \mathsf{DPOLYLOG}$ and the result [IKW02, For97] that $\mathsf{BPE} = \mathsf{EE}$ if and only if $\mathsf{BPP} = \mathsf{EXP}$.

Theorem 20. *If $\mathsf{P} \not\subseteq heur_{1/2+1/n^k} - \mathsf{NSIZE}(n^k)$ for any fixed k, then for each $\epsilon > 0$, $\mathsf{E}/2^{\epsilon n} \not\subseteq \mathsf{SIZE}(2^n/n)$.*

Proof. The proof is analogous to the proof of Theorem 18, except that we use G_{NW} as a hardness extractor rather than G_U.

Given the assumption, we fix $\epsilon > 0$ and show that there is a language in $\mathsf{E}/2^{\epsilon n}$ which doesn't have deterministic circuits of size $2^n/n$. From Corollary 16, we have that G_{NW} is an $(n, s, n' = \lfloor c \log(s) \rfloor, 2^{n'}/n')$ condenser with advice length $O(n^2)$ from $1/2 - 1/s$ average-case hardness for non-deterministic circuits to hardness for deterministic circuits, for some constant $c > 0$. Let $k = 2c/\epsilon$. We consider G_{NW} with parameter $s = n^k$, which implies $n' = \lfloor 2 \log(n)/\epsilon \rfloor$. By assumption, there is a Boolean function f in P which is not $1/2 - 1/n^k$ approximated by non-deterministic circuits of size n^k.

We construct a Boolean function $f' \in \mathsf{E}/2^{\epsilon m}$ such that f does not have deterministic circuits of size $2^m/m$. As in the proof of Theorem 18, we define f' implicitly by specifying an advice-taking

11

machine M operating in linear exponential time and a sequence of advice strings $\{a_m\}, m = 1 \ldots \infty$ with $|a_m| \leqslant O(2^{\epsilon m})$, such that for each input length m, M with advice a_m computes f'_m.

The advice string consists of two parts: the first part codes a number n such that $m = \lfloor 2 \log(n)/\epsilon \rfloor$, and the second part is a string y of length $O(n^2)$. M operates as follows: on an input of length m, it first determines n and y from the advice. It then computes the bit of $G_{NW}(f_n, y)$ corresponding to its input. Since G_{NW} is super-explicit, the computation takes time polynomial in $2^m + n^2 < 2^{m+1}$ with oracle access to f_n. Note that there can be at most $poly(2^m)$ oracle calls made to f_n (in fact, G_{NW} satisfies a stronger condition: it makes just one oracle call!), and each oracle call can be answered in time $poly(2^m)$ by running the polynomial-time algorithm for f. Thus, M' takes linear exponential time.

The argument that f' is hard is essentially the same as in the proof of Theorem 18. It follows from the fact that for infinitely many m, there must be a "good" advice string, i.e., an advice string which codes an input length n on which f is hard, and advice y for G_{NW} such that $G_{NW}(f_n, y)$ is a function with hardness $2^m/m$, by the hardness extraction property of G_{NW}. $\qquad \square$

It would be nice to have an equivalence rather than a one-way implication in the statement of Theorem 20. There are two obstacles to this. The first is that the assumption on hardness of P concerns non-deterministic circuits, while the assumption on hardness of E with advice concerns hardness with respect to deterministic circuits. A traditional upward translation technique would work only if the hardness assumptions were with respect to the same kind of circuit. The second is that the assumption on hardness of P is an average-case assumption, while the assumption on hardness of E with advice is a worst-case assumption.

We circumvent these obstacles by considering hardness with respect to non-uniform space rather than non-uniform time. Non-uniform space is closed under non-deterministic reductions, thus we get around the first obstacle. To get around the second obstacle, we give an upward translation technique for average-case simulations, and then show that worst-case hardness of E with advice is essentially equivalent to average-case hardness.

The non-uniform space model we consider is the model of alternating circuits, which generalizes non-deterministic circuits by allowing an arbitrary number of alternations. The fact that the Nisan-Wigderson generator relativizes [KvM02] yields a generator against alternating circuits.

Theorem 21 ([NW94, AK97]). *Let $s(n)$ be any constructible function. The generator G_{NW} in Theorem 13 is a super-explicit $(n, s, \sqrt{s})$ PRG with error parameter ϵ and seed length $O(n^2/\log(s))$ based on $1/2 - \epsilon/s$ average-case hardness for alternating circuits.*

Next, we utilize the connection of pseudo-random generators with hardness extraction.

Theorem 22. *G_{NW} is a super-explicit (n, s) hardness extractor with advice length $n^2/\log(s)$ from $1/2 - 1/s$ average-case hardness for alternating circuits to hardness for alternating circuits.*

Proof Sketch.

We relativize the proof of Lemma 15 to alternating time. If the output of G_{NW} did not have close-to-maximum hardness for any advice string, a non-deterministic circuit with oracle access to alternating time could distinguish the output of the generator from a random string. A non-deterministic circuit with oracle access to alternating time can be simulated by an alternating circuit of size polynomially bounded in the size of the original circuit. Such a distinguisher would yield an approximation to the input function by Theorem 21, contradicting the hardness assumption on the input function. Hence there must be some advice string on which the output function of G_{NW} has close to maximum alternating size, thus proving the hardness extraction property. $\qquad \square$

The novel feature of Theorem 22 as compared to our previous results on hardness extraction is that the complexity model we extract from is the same as the complexity model we extract to. By a proof exactly analogous to the proof of Theorem 20, we obtain the following result.

Theorem 23. *There is a constant γ such that if $\mathsf{P}/n^2 \not\subseteq heur_{1/2+1/n^k} - \mathsf{ASIZE}(n^k)$ for any fixed k, then for each $\epsilon > 0$, $\mathsf{E}/2^{\epsilon n} \not\subseteq \mathsf{ASIZE}(2^n/\gamma n)$.*

Using the fact that $\mathsf{ASIZE}(s) \subseteq \mathsf{DSPACE}(s^2)/s^2 \subseteq \mathsf{ASIZE}(s^4)$, we get the following corollary, which we state in contrapositive form.

Corollary 24. *There is a constant $\delta > 0$ such that the following holds: if there is a constant $\epsilon > 0$ for which $\mathsf{E}/2^{\epsilon n} \subseteq \mathsf{DSPACE}(2^{\delta n})/2^{\delta n}$, then there is a constant k such that $\mathsf{P}/n^2 \subseteq heur_{1/2+1/n^k} - \mathsf{DSPACE}(n^k)/n^k$.*

We now proceed to show that the converse to Corollary 24 holds. We use a translation technique translating average-case simulations upward to show this. The technique only works if the average-case simulation has at least a constant advantage over random, so we first need a hardness amplification lemma.

We use a version of Yao's XOR lemma [GNW95] with appropriate parameters for the hardness amplification.

Lemma 25. *For a Boolean function f on n bits, define $f^{\oplus t}$ to be the Boolean function on tn bits which is the parity of t independent copies of f. For any $k' > 1$, there is $k > 1$ such that if f is a Boolean function on n bits which is $1/n$-hard for $\mathsf{DSPACE}(n^k)/n^k$, then $f^{\oplus n^2}$ is a Boolean function on $m = n^3$ bits which is $1/2 - 1/m^{k'}$ hard for $\mathsf{DSPACE}(m^{k'})/m^{k'}$.*

Lemma 25 can be obtained from Lemma 1 in the survey of Goldreich, Nisan and Wigderson [GNW95] by appropriate parameter substitutions. The lemma in the survey pertains to amplification with respect to deterministic circuits but the proof can be seen to go through for amplification with respect to non-uniform space.

Notice that if f is computable in P with $a(n)$ bits of advice, then so is $f^{\oplus n^2}$ (and indeed the advice length for computing $f^{\oplus n^2}$ is smaller as a function of its input length). Thus we have the following corollary to Lemma 25.

Corollary 26. *If there is a constant k' such that $\mathsf{P}/n^2 \subseteq heur_{1/2+1/n^{k'}} - \mathsf{DSPACE}(n^{k'})/n^{k'}$, then there is a constant k such that $\mathsf{P}/n^2 \subseteq heur_{1-1/n} - \mathsf{DSPACE}(n^k)/n^k$.*

We are now ready to give the argument for the upward translation.

Lemma 27. *If there is a constant k such that $\mathsf{P}/n^2 \subseteq heur_{1/2+1/n^k} - \mathsf{DSPACE}(n^k)$, then for each constant $\delta > 0$, there is a constant $\epsilon > 0$ for which $\mathsf{E}/2^{\epsilon n} \subseteq \mathsf{DSPACE}(2^{\delta n})/2^{\delta n}$.*

Proof. Assume there is a constant k such that $\mathsf{P}/n^2 \subseteq heur_{1/2+1/n^k} - \mathsf{DSPACE}(n^k)/n^k$. Then, by Corollary 26, there is a constant k' such that $\mathsf{P}/n^2 \subseteq heur_{1-1/n} - \mathsf{DSPACE}(n^{k'})/n^{k'}$. We will show that for each constant $\delta' > 0$, there is a constant $\epsilon' > 0$ such that $\mathsf{E}/2^{\epsilon' n} \subseteq heur_{1-1/n^{\omega(1)}} - \mathsf{DSPACE}(2^{\delta' n})/2^{\delta' n}$. The consequent of the theorem follows from this by using a standard worst-case to average-case reduction via random self-reducibility [BF90, Lip90] .

Upward translation results for worst-case simulations are typically proved using a padding argument. We adapt the padding argument to the average-case setting, as in [FS04]. Let $\delta' > 0$ be a constant, and let $\epsilon' = 2\delta'/k'$. Let L be a language in $\mathsf{E}/2^{\epsilon' n}$.

Now, define the language $L' = \{xy | x \in L, |x| + |y| = 2^{\delta' |x|/k}\}$. It is not hard to see that $L' \in \mathsf{P}/n^2$, and hence $L' \in heur_{1-1/n} - \mathsf{DSPACE}(n^{k'})/n^{k'}$. Let M' be an advice-taking space bounded machine running in space $n^{k'}$ and $\{a_n\}, n = 1 \ldots \infty$ be a sequence of advice strings such that for each n, $|a_n| = n^{k'}$ and M' with advice a_n solves L' on at least a $1 - 1/n$ fraction of strings of length n. We define an advice-taking machine M operating in space $2^{\delta' m}$ and a sequence of advice strings $\{b_m\}$ with $|b_m| = 2^{\epsilon' m}$ such that M with advice b_m solves L on at least a $1 - 2/m$ fraction of strings of length m.

M operates as follows: on input x of length m, it runs M' with advice b_m on input xy for each string y of length $2^{\delta' m/k} - m$, and outputs the majority answer. The space required by M is the space needed to store y plus the space needed to run M', which is bounded above by $2^{\delta' m}$. Set $n = 2^{\delta' m/k}$. We show that for the sequence of advice strings $b_m = a_n$, M approximates L well.

Call an input x of length m "good" if $M'(xy)$ with advice $b_m = a_n$ is the same as $L'(xy)$ for the majority of strings y of length $n - m$. Since M' solves L' correctly on a $1 - 1/n$ fraction of strings of length n, by an averaging argument, at least a $1 - 2/n$ fraction of inputs of length m are good. Since $n >> m$, we have that at least a $1 - 1/m^{\omega(1)}$ fraction of inputs of length m are good. M computes L correctly on each good input when given advice b_m, hence the theorem is proved. $\square$

Lemma 27 and Corollary 24 together yield the formal version of Theorem 2, which we state below.

Theorem 28. *There is a constant k such that $\mathsf{P}/n^2 \subseteq heur_{1/2+1/n^k} - \mathsf{DSPACE}(n^k)$ if and only if for each constant $\delta > 0$, there is a constant $\epsilon > 0$ for which $\mathsf{E}/2^{\epsilon n} \subseteq \mathsf{DSPACE}(2^{\delta n})/2^{\delta n}$.*

An interesting feature of Theorem 28 is that while the statement does not involve randomized classes, the proof goes through pseudo-random generators! Indeed, since probabilistic alternating time can be simulated polynomially in alternating time, derandomization is not an issue here, however we still obtain non-trivial relationships between deterministic time classes and alternating size using pseudo-random generators.

It would be interesting to prove a version of Corollary 24 which involves uniform classes. We are able to derive a version where the classes in the consequent are uniform. We do this by showing that a simulation of P on average in non-uniform space implies a simulation of P on average in uniform space. The corresponding implication for worst-case simulation is a standard Karp-Lipton style result [KL82], where the non-uniformity is eliminated by enumerating over all advice strings and using the local checkability of languages in P to find the right advice string. The standard notion of local checkability does not help us in the average-case setting. Nevertheless, we are able to use the fact that there are "error-tolerant" proofs for languages in P derive our result. This fact follows from known results about probabilistically checkable proofs (PCPs); however, we do not need the full power of PCPs and it suffices for us to encode the conventional proofs with a good locally decodable error-correcting code. We omit the details.

Theorem 29. *There is a constant $\delta > 0$ such that the following holds: if there is a constant $\epsilon > 0$ such that $\mathsf{E}/2^{\epsilon n} \subseteq \mathsf{DSPACE}(2^{\delta n})/2^{\delta n}$, then there is a constant k such that $\mathsf{P} \subseteq heur_{1-1/n} - \mathsf{DSPACE}(n^k)$*

Proof Sketch.

First, using the proof technique of Corollary 24, an average-case simulation of P in a fixed non-uniform space bound follows from the assumption. The proof proceeds by showing that an average-case simulation of P in non-uniform space implies an average-case simulation of P in uniform

space. This is an "average-case" Karp-Lipton theorem. The technique involves encoding locally-checkable proofs of membership for a language in P with a locally decodable code, and then arguing that there must be some non-uniform space algorithm which can be verified to work on a large proportion of the inputs. Such a non-uniform algorithm can be found space-efficiently by cycling over all possibilities and determining the one for which the verification has the highest probability of success. □

4 Limitations of hardness condensers

A couple of natural questions arise, given the results in the previous section. First, whether hardness condensers require access to an advice string. If it were possible to eliminate the advice, the condensing transformation would preserve uniformity and we could derive stronger results. Second, whether there are hardness condensers from deterministic hardness to deterministic hardness. This is the most interesting case if one is trying to use hardness condensing to prove (or at the least improve) circuit lower bounds in NP or even in E.

In this section, we consider two natural classes of hardness condensers: relativizing condensers and the more restricted class of black-box condensers. We show advice lower bounds for relativizing condensers, thus giving a negative answer to the first question in this setting. Also, we show that essentially no deterministic hardness condensing is possible in the black-box model. We interpret these negative results to indicate that radically new techniques are required to obtain completely uniform hardness condensers or to extract deterministic hardness.

4.1 Relativizing extractors

We begin by defining relativizing condensers. The idea behind the definition is very simple: we require a condenser from hardness in complexity model $\mathcal{A}$ to hardness in $\mathcal{B}$ to work even when the elements of $\mathcal{A}$ and $\mathcal{B}$ are given access to an oracle. It is implicit in this definition that it is meaningful for complexity models $\mathcal{A}$ and $\mathcal{B}$ to have access to an oracle. We refer to such complexity models as "relativizable". We do not define this notion formally, merely observing that most interesting complexity models such as formulas or circuits of any depth are relativizable.

We assume that all complexity models in this section come with a bit-encoding scheme. In this section, we will always take the *size* or complexity of an element of $\mathcal{A}$ or $\mathcal{B}$ to be the length of its bit-encoding. So in the notation "(n, s, n', s')-hardness condenser," s and s' refer to lengths of bit-encodings. For concreteness, we assume that an element of $\mathcal{A}$ or $\mathcal{B}$ that makes a single call to an oracle with n bits of input and immediately outputs the result is encoded by a bit string of length $O(n)$.

Definition 30. *Let $\mathcal{A}$ and $\mathcal{B}$ be relativizable complexity models. For any oracle D, let $\mathcal{A}^D$ and $\mathcal{B}^D$ be the complexity models induced by giving $\mathcal{A}$ and $\mathcal{B}$ access to D. C is a relativizing hardness condenser from $\mathcal{A}$ to $\mathcal{B}$ if for any oracle D, C is a hardness condenser from hardness in $\mathcal{A}^D$ to hardness in $\mathcal{B}^D$.*

Relativizing hardness condensers are a very general construct. We note that all the hardness condensers in the previous section are relativizing hardness condensers. From a more conceptual point of view, we emphasize the following features of Definition 30:

1. There is no restriction on the relation between complexity models $\mathcal{A}$ and $\mathcal{B}$; the only condition on them is that they are relativizable.

2. There is no restriction on the efficiency of computing the hardness condenser. The hardness condenser is not required to be uniform (note that the argument from Section 2 that non-uniform hardness extractors exist trivially does not hold for relativizing hardness extractors). Nor is there any explicitness condition - a hardness condenser could require circuits of exponential size to compute and still satisfy the definition.

3. There is no restriction on the proof that C is a relativizing hardness condenser. In particular, it is conceivable that there is no easy way to derive a small element of $\mathcal{A}$ computing the input function from a small element of $\mathcal{B}$ computing the output function.

Relativizing hardness condensers do exist, as evidenced by the constructions in the previous section. We show that advice is necessary for relativizing hardness condensers by giving lower bounds for the advice length in terms of the other parameters of the condenser.

Our lower bounds work via counting arguments. In the following, we will think of a (n, s, n', s') hardness condenser C as a bipartite graph $G_C = (L, R, E)$, where $E \subseteq (L \times R)$. The left set L represents the set of functions from which we condense hardness, and the right set R represents the set of functions to which we condense hardness. There is an edge between a node u on the left and a node v on the right if there is some advice string y such that $C(u, y) = v$ (actually, we use a slightly different graph for our formal argument, but this is the most natural graph to use).

A natural lower bound approach would be to show that G_C is a disperser a la Trevisan [Tre01]. This does not seem to be immediate, since different nodes in R could conceivably have different degrees. However, a modified counting argument does work.

To gain intuition, we first sketch the argument that hardly any non-trivial hardness extraction is possible for a relativizing hardness extractor with no advice. If there is no advice, then every node in L has exactly one neighbour in R. Also, in the case of extraction, the size of the truth tables in R is about s. The basic idea is that if s is not too close to 2^n, there must be some node v_{max} on the right with large degree, by the pigeon-hole principle. Now consider an oracle which is precisely the function represented by v_{max}. With respect to the oracle, v_{max} has very low complexity. On the other hand, since v_{max} has a large number of neighbours, one of these neighbors must have high complexity even relative to the oracle v_{max}, again by the pigeon-hole principle. This is a contradiction to the hardness extraction assumption.

We now present a more formal and more general argument:

Theorem 31. *Any (n, s, n', s') relativizing hardness condenser requires ℓ bits of advice where*

$$2^\ell > \frac{2^n + \Omega(s') - s}{2^{n'+1}}.$$

Theorem 31 is the formal version of Theorem 3. Before proving it, we observe a few implications of this bound. First of all, in the case of no advice ($\ell = 0$), it's clear that s' cannot be much bigger than s, because $n > n'$. This makes sense since it should be impossible to make a problem harder without adding any information to it. Secondly, to extract all the hardness of the original function (that is, s' is almost s and is exponential in n'), we get $\ell \in \Omega(n - \log s)$. This bound is tight up to a constant factor, since the hardness extractor in Corollary 17 has advice length $O(n)$ for any s.

Theorem 31 also explains why we could not get rid of the advice in Theorem 1 and obtain an improvement in hardness for a uniform function in E.

Proof of Theorem 31. Let $\mathcal{A}$ and $\mathcal{B}$ be complexity models and let C be an (n, s, n', s') relativizing hardness condenser with ℓ bits of advice from $\mathcal{A}$ to $\mathcal{B}$.

16

Now consider the bipartite graph $G_C = (L, R, E)$ corresponding to C. L is the set of all truth-tables of size 2^n. R is the set of all ℓ-tuples of truth-tables of size $2^{n'}$. Hence, $|L| = 2^{2^n}$ and $|R| = 2^{2^{\ell+n'}}$. Each node in L maps to one node in R corresponding to the outputs for each advice string.

The average degree of a node in R is $2^{2^n - 2^{\ell+n}}$. For r to be fixed later, let X_r denote the set of r nodes of largest degree in R. Let $N(X_r)$ denote the set of neighbors of X_r. Then we have $|N(X_r)| \geqslant r 2^{2^n - 2^{\ell+n'}}$.

We define the oracle D as follows: D is a function on $\log(r) + \ell + n'$ bits. The first $\log(r)$ bits are an index to a node v in X_r, the next ℓ bits are an index to one of the functions at node v and the remaining bits index an entry of that function. The output of D is simply the corresponding entry of the function. For any node v in X_r, the complexity of every function at v in $\mathcal{B}^D$ is $O(\log(r) + \ell + n')$, since just one oracle call to D is required to compute any bit of each of these functions. These functions are "easy" if this complexity is less than s', so set r as big as possible such that this holds. So, we have $r = 2^{\Omega(s') - \ell - n'}$. Then,

$$|N(X_r)| \geq 2^{2^n + \Omega(s') - \ell - n' - 2^{\ell+n'}}.$$

By the defining property of the condenser, every function in X_r must have complexity less than s. So, it better be that

$$
\begin{aligned}
2^s &> 2^{2^n + \Omega(s') - \ell - n' - 2^{\ell+n'}} \\
s &> 2^n + \Omega(s') - \ell - n' - 2^{\ell+n'} \\
&> 2^n + \Omega(s') - (2^\ell + 1)(2^{n'} + 1) \\
&> 2^n + \Omega(s') - 2(2^\ell)(2^{n'}).
\end{aligned}
$$

So,

$$2^\ell > \frac{2^n + \Omega(s') - s}{2^{n'+1}}.$$

$\square$

4.2 Black-box hardness condensers

We now turn to the question of whether there is a hardness condenser from hardness for deterministic circuits to hardness for deterministic circuits. One approach to constructing them would be to follow our construction of condensers from nondeterministic circuits to deterministic circuits; that is, to try to show that a deterministic adversary can estimate the circuit size of given truth-table. This is essentially the Minimum Circuit Satisfiability Problem (MCSP). Unfortunately, Kabanets and Cai [KC00] show that if MCSP were in P, the discrete logarithm problem could be solved efficiently, which is considered unlikely.

Barring that, the most natural approach would be to use a construction that allows some sort of "black-box" reconstruction technique. We define a natural black-box model of hardness condensing and show an impossibility result in this model.

The defining feature of the black-box model is that given a set of small circuits for all the outputs of a given input into the hardness condenser, it should be possible to construct a small circuit for the input truth table.

Definition 32. *An (n, s, n', s') hardness condenser C for deterministic circuits is said to be t-black box if C is relativizing and there is an oracle circuit of size t which computes the input function of C when given oracle access to the outputs of C.*

Theorem 33. *Any t-black box (n, s, n', s') hardness extractor has $t > \frac{2^n + \Omega(s') - s}{2^{n'+1}}$.*

Proof. Theorem 31 shows that there are at least $\frac{2^n + \Omega(s') - s}{2^{n'+1}}$ outputs. An oracle circuit for the input function must make at least that many oracle calls. $\square$

Any black box proof of (n, s, n', s') hardness condensing would require a t-black box condenser such that $ts' < s$. In other words, the proof would construct a circuit of size less than s for an input function given circuits of size less than s' for each of the possible output functions and the oracle circuit for the condenser.

To argue that this can't happen, we first need to revisit what it means to be an effective hardness condenser. The point of a hardness condenser is to output a function that is relatively harder than the input function. In the case of an infinite family of hardness condensers, we can express this by saying that any $(n, s(n), n'(n), s'(n))$-hardness condenser should satisfy $s'(n) \in \omega(s(n'(n)))$. We assume that $s(n) \in o(2^n)$, since otherwise not much condensing is necessary. In this case, $\frac{s(n'(n))}{2^{n'(n)}} \geq \frac{s(n)}{2^n}$ and $\frac{s'(n)}{2^{n'(n)}} \in \omega\left(\frac{s(n)}{2^n}\right)$.

Finally, we can explore the relationships between t, s and s'. We have

$$
\begin{aligned}
ts' &> \frac{2^n + \Omega(s') - s}{2^{n'+1}} s' \\
&> s \frac{2^n + \Omega(s') - s}{s} \frac{s'}{2^{n'+1}} \\
&> s \left(\frac{2^n}{s} - 1\right) \frac{s'}{2^{n'+1}} \\
&> s \, \Omega\left(\frac{2^n}{s} \frac{s'}{2^{n'}}\right) \quad \text{for sufficiently large } n \\
&> s
\end{aligned}
$$

5 Hardness condensing in special cases

In this section we give two cases where we can achieve hardness condensing from deterministic circuits to deterministic circuits. In both cases, the input hard function has some clear redundancy to it that can be eliminated to form a smaller function without sacrificing much hardness.

5.1 Biased functions

Lupanov ([Lup65]) shows that any function $F : \{0,1\}^n \to \{0,1\}$ that outputs 1 (or, respectively, 0) with probability at most $\alpha \leq 1/2$ can be computed by a circuit of size $O(H(\alpha)2^n)$ (actually, Lupanov proves a much more precise bound). There is a proof of this fact due to [Imp05] that can be adapted to show that there is a hardness condenser for biased functions that reduces the truth table to $O(H(\alpha)2^n)$. Of course, this is the answer that one would expect by analogy to information theory. Another way to interpret this is, if a function is biased towards, say, 0 and has some hardness, then one can sort of throw out excess 0's to form a smaller, balanced function that is relatively harder (recall $H(\alpha) = -\alpha \log \alpha - (1 - \alpha) \log(1 - \alpha)$, which, for $\alpha \leq 1/2$ is $O(\alpha \log \frac{1}{\alpha})$). This is not much more than the 2α that would arise from "throwing out" all the excess 0's.

Definition 34. *For $F : \{0,1\}^n \to \{0,1\}$, let $\alpha_F = \min\{\Pr_x(F(x) = 1), \Pr_x(F(x) = 0)\}$.*

Clearly $\alpha_F \leq 1/2$.

Theorem 35. *For some constant c, there is an explicit $(n, s, n - \lfloor c \log \frac{1}{H(\alpha_F)} \rfloor, \Omega((s - n)/n))$-hardness condenser from deterministic worst-case hardness to deterministic worst-case hardness that requires no advice.*

Theorem 35 is the formal version of Theorem 4. Of course, the theorem is meaningful only when $s(n)$ is somewhat bigger than n. To get actual hardness-condensing, however, there is a tradeoff between α_F and $s(n)$. For instance, if $\alpha_F = O(1/n^2)$ and F requires circuits of size $2^{\epsilon n}$, then the output is in fact relatively harder than the input.

Proof of Theorem 35. Throughout the proof, we will use constructions of pairwise-independent hash functions from $\{0,1\}^n$ to $\{0,1\}^\ell$ for various values of ℓ. In particular, let $U \subset \{0,1\}^n$ and $V \subset \{0,1\}^\ell$ such that $|V| = 2^{\ell-1}$. We assume that in time $2^{O(n)}$ we can find a hash function $h : n \to \ell$ such that at most $3|U|/4$ elements of U hash to V. Moreover, h should be computable in time $O(n)$. Such parameters can be achieved by doing an exhaustive search over all hash functions in the affine subspace construction of a pairwise-independent, universal family of hash functions (see [LWly], for example).

Let T_1 be the set $F^{-1}(1)$ and let U_1 be $F^{-1}(0)$. Let $\ell_1 = \lceil \log(2|T_1|) \rceil$. Choose a hash function $h_1 : \{0,1\}^n \to \{0,1\}^{\ell_1}$ such that at most $3|U_1|/4$ elements of U_1 hash to $h_1(T_1)$, where $h_1(T_1)$ denotes the range of h on T_1. Let U_2 be those elements of U_1 and let $T_2 = T_1$. Define $F_1 : \{0,1\}^{\ell_1} \to \{0,1\}$ to be 1 on input x iff there exists a $y \in T_1$ such that $h_1(y) = x$.

In general, given U_i and T_i such that $|U_i| \geq |T_i|$ (if $|U_i| < |T_i|$, reverse the roles of U_i and T_i and 0 and 1), let $\ell_i = \lceil \log(2|T_i|) \rceil$ and define $F_i : \{0,1\}^{\ell_i} \to \{0,1\}$ analogously: choose a hash function $h_i : \{0,1\}^n \to \{0,1\}^{\ell_i}$ that hashes at most $3|U_i|/4$ elements of U_i to $h_i(T_i)$—call these elements U_{i+1}. Let $T_{i+1} = T_i$ and let $F_i(x) = 1$ iff there exists a $y \in T_i$ such that $h_i(y) = x$.

Let k be the first stage at which each of U_k and T_k has at most, say, 512 elements. Let $h_k : \{0,1\}^n \to \{0,1\}^{200}$ be a hash function that hashes $U_k \cup T_k$ perfectly. Define $F_k : \{0,1\}^{1024} \to \{0,1\}$ in the usual way. Finally, let F' be the boolean function whose truth table is the concatenation of the truth-tables of $F_1, \ldots, F_k$ (if this is not a power of 2, pad it out with 0's). The hardness condenser outputs F'.

Now that we have described the construction, we must argue its correctness. First of all, $k \in O(n)$. This is because U_1 and T_1 are at most 2^n and, in each iteration, one of them shrinks by a factor of $3/4$.

We now compute the size of the truth-table of F'. Let $\alpha = \alpha_F$. In each iteration $(1...k-1)$, we add $2\min\{|U_i|, |T_i|\}$ bits to the truth-table of F' (and a constant number in iteration k). Split the sequence of iterations into two parts: let j be the first point at which $|U_j| \leq |T_1|$. In iterations $1...j-1$, we add $2|T_1| = 2\alpha 2^n$ bits to the truth-table of F'. Since $|U_1| = (1-\alpha)2^n$ and it shrinks by $3/4$ in every iteration before j, it must be the case that $j - 1 \in O(\log \frac{1-\alpha}{\alpha})$, which is $O(-\log \alpha)$. After iteration $j - 1$, the value $\min\{|U_i|, |T_i|\}$ shrinks by $3/4$ every two iterations, so from then on, we add $O(\alpha 2^n)$ bits to the truth-table of F'. Hence, the truth-table of F' has size at most $O((\alpha \log \frac{1}{\alpha})2^n)$, which is $O(H(\alpha)2^n)$.

Finally, we show that if F' can be computed by a circuit of size t, then F can be computed by a circuit of size $O(n(t+n))$. The circuit for F will need access to $h_1, \ldots, h_k$ and will need to know whether $|U_i| \geq |T_i|$ for each iteration. To compute $F(x)$, first compute $y_1 = h_1(x)$ and use the circuit for F' to compute $F_1(y_1)$. If it's 0, answer 0. Otherwise continue. In general, if we haven't output a value in the first $i - 1$ iterations, let $y_i = h_i(x)$ and look up $F_i(y_i)$ using the circuit for F'. If the value is 0 and $|U_i| \geq |T_i|$, answer 0; if the value is 1 and $|T_i| > |U_i|$, answer 1; otherwise continue. We need size $O(n)$ to compute each hash function and there are $O(n)$ rounds. $\square$

The circuit upper bound for biased functions gives a proof that functions that are hard for very large circuits have some average-case hardness (this is very similar to a proof of [ACR99]). More precisely, let f be any function that requires circuits of size $s = \omega(H(\delta)2^n)$. Assume f is not δ-hard for circuits of size $s/2$. In other words, there is a function g that agrees with f on a $1 - \delta$ fraction

that can be computed by circuits of size $s/2$. Then, the function $f \oplus g$ must require circuits of size at least $s/2$, but $\alpha_{f \oplus g} \leq \delta$ which contradicts the circuit upper bound.

We can use a similar argument to show that any hard language in E either has average-case hardness of a sort or is not as hard as some other function in E. In particular, if there is a language in the PH that is as hard as any language in E, then that function must have this type of average-case hardness. This is perhaps interesting due to the difficulty of black-box worst-case to average-case reductions within the polynomial hierarchy [FF93, BT03, Vio05].

Theorem 36. *Let $s(n) \in \Omega(2^{\epsilon n})$ for some $\epsilon > 0$ and let $\delta = \delta(n) \in o(1)$. If L is a language in E that requires circuits of size $s(n)$ almost everywhere, then either*
(1) There is a language in E that requires circuits of size $\Omega(s(n)/(nH(\delta)))$ almost everywhere; or
(2) L cannot be δ-approximated by any language in E that has circuits of size $s(n)/2$.

Proof. Assume there is a language $A \in E$ that δ-approximates L and has circuits of size $s(n)/2$. View A_n and L_n as boolean functions from $\{0,1\}^n$ to $\{0,1\}$ and define B to be the language such that $B_n = A_n \oplus L_n$ for each n. B is then a language that requires circuits of size at least $s(n)/2$ and $\Pr(B_n(x) = 1) \leq \delta$ for each n. Let C_n be the hardness-condenser of Theorem 35 and let c be the constant there. Apply C_n to B_n for each n. Let $L'_{n'}$ be the function computed by choosing an n such that $n' = n - \lfloor c \log \frac{1}{H(\delta)} \rfloor$ and applying $C_n(B_n)$. For almost all n', $L'_{n'}$ will require circuits of size $\Omega(\frac{s(n)/2-n}{n})$ for the corresponding n.. Hence, L' is a language in E that requires circuits of size at least $\Omega(s(n)/(nH(\delta)))$ almost everywhere. $\qquad\square$

5.2 Average-case hard functions

In this subsection, we show how to non-trivially condense average-case deterministic hardness to worst-case deterministic hardness. Our technique involves a connection between a certain kind of explicit construction of *covering codes* and hardness condensing. Before we describe this connection, we give a simple illustrative example which already yields some non-trivial hardness condensing.

Let f be a Boolean function on n bits which is γ average-case hard for deterministic circuits of size s. Let $\delta = 1/2 - \gamma$; this is often called the *advantage* for circuits of size s on f. We show how to explicitly condense f to a function f' on $n - \lfloor \log(1/\delta) \rfloor$ bits which is hard for deterministic circuits of size $s - O(n)$. Assume wlog that $1/\delta$ is an odd integer (if not, perturb δ slightly so that this becomes true). Imagine the string $tt(f)$ to be divided into blocks $B_i, i = 1 \ldots \gamma 2^n$, each block consisting of $1/\epsilon$ consecutive bits in $tt(f)$. Now we define f' on input i to be the majority function on the bits in B_i. Clearly, f' can be derived explicitly from f.

We show that f' has worst-case hardness $s - O(n)$. Let C' be a circuit of size s' computing f'. Then the following circuit C computes a $1/2 + \delta$ approximation to f: C first determines which block B_i its input is mapped to. This can be done with a linear number of gates. C then runs C' on i. Since C' computes the majority value of the bits in B_i and since $|B_i|$ is odd, C computes a $1/2 + \delta$ approximation to f over inputs mapped to B_i. Since this is true for each i, C computes a $1/2 + \delta$ approximation to f over all n-bit inputs. The size of C is bounded above by $s' + O(n)$. By the assumption on average-case hardness of f, $s' + O(n) > s$, which implies $s' > s - O(n)$.

Note that the input size of the condensed function depends only on the maximum advantage a circuit can achieve on the input function, and not on the size of such a circuit. If the advantage is exponentially small, the input size is condensed by a constant factor. We now describe a strengthening of the above approach, which involves a connection between hardness condensing and covering codes. From here on, we will assume we are dealing with a function and circuit size such that $\delta \in o(1)$, since otherwise the condensing is minimal.

20

First we define covering codes.

Definition 37. *A (K, N, R) covering code is a function $A : K \to N$ such that for each $y \in \{0, 1\}^N$, there is some string $x \in \{0, 1\}^K$ such that the Hamming distance between y and $A(x)$ is at most R.*

Intuitively, a covering code corresponds to a set of N-bit strings such that the set of Hamming balls of radius R centered on these strings covers the whole space of N-bit strings. The smaller K and R are as functions of N, the better the covering code.

Given fixed N and R, a simple lower bound on K, known as the "sphere-covering bound" can be shown by a counting argument. Let $V(N, R)$ denote the number of N-bit strings within a Hamming ball of radius R. Then:

Proposition 38. $K \geqslant N - \log(V(N, R))$

It can be shown using a probabilistic argument that covering codes with parameters that almost meet this lower bound exist non-explicitly.

Proposition 39 ([CHLL97]). *For any N and R, there exists a (K, N, R) covering code with $K = N - \log(V(N, R)) + \log(N) + O(1)$.*

We will require explicit covering codes with the additional properties of *local computability* and *efficient recoverability*. We now define these concepts.

Definition 40. *A (K, N, R) covering code A is t-local if for each $x \in \{0, 1\}^K$ there is a circuit of size $poly(t, \log(N))$ with oracle access to x which, given as input an index i between 1 and N, outputs the ith bit of $A(x)$. A is efficiently recoverable if there is a polynomial time procedure rec_A which given a string $y \in \{0, 1\}^N$ as input, outputs a string $x \in \{0, 1\}^K$ such that the Hamming distance between $A(x)$ and y is at most R.*

Since the notion of efficient recoverability is concerned with uniformity, in the definition above we are implicitly referring to a family of (k, n, r) covering codes, where k and r are functions of n. In future, we only consider families of covering codes.

We are interested in t-local efficiently recoverable covering codes with t as small as possible. We now attempt to give some intuition about the connection between such codes and hardness condensing.

Let f be a function on n bits which is $R/2^n$ average-case hard for circuits of size s. We wish to "compress" f to a function with smaller input size which is worst-case hard for circuits of size approximately s. Sudan, Trevisan and Vadhan [STV01] showed that a worst-case hard function can be converted to an average-case hard function by encoding it with a locally list-decodable error correcting code. Our technique is a dual to theirs. Let A be a t-local efficiently recoverable $(2^k, 2^n, R)$ covering code. Our condensed function f' will be the function on k bits with truth table $rec_A(tt(f))$. The efficient recoverability property implies that our putative hardness condenser is explicit.

To prove the hardness condensing property, we need to show that f' does not have small circuits. Suppose f' has circuits of size s'. By t-locality of A, $g = fn(A(tt(f')))$ is a function on n bits with circuits of size $poly(t, n)s'$. Also, since A is a $(2^k, 2^n, R)$ covering code, g is a $1 - R/N$ approximation to f. This is a contradiction to average-case hardness of f if $poly(t, n)s' < s$.

We now give a formal statement of the connection.

Theorem 41. *If there is a t-local efficiently recoverable $(2^k, 2^n, R)$ covering code, then there is a constant c such that there is an explicit $(n, s, k, s/(t + n)^c)$ hardness condenser from $R/2^n$ average-case hardness for deterministic circuits to worst-case hardness for deterministic circuits.*

To derive the benefits of this connection, we need to find constructions of local, efficiently recoverable covering codes with good parameters. We first investigate the best-case scenario: what would the best possible code (by Proposition 38) give us? In other words, if we fix $R = \gamma N$, what is the smallest K can be? It is $K = N - \log V(N, \gamma N)$. We can estimate (see, for example, [Ash65], p. 121)

$$\frac{1}{8N} 2^{NH(\gamma)} \le V(N, \gamma N) \le 2^{NH(\gamma)}.$$

Computing $H(\gamma)$, we get

$$
\begin{aligned}
H(\gamma) &= -\gamma \log \gamma - (1-\gamma)\log(1-\gamma) \\
&= (\tfrac{1}{2} - \delta)\log \tfrac{2}{1-2\delta} + (\tfrac{1}{2} + \delta)\log \tfrac{2}{1+2\delta} \\
&= (\tfrac{1}{2} - \delta)(1 - \log(1-2\delta)) + (\tfrac{1}{2} + \delta)(1 - \log(1+2\delta)) \\
&= (\tfrac{1}{2} - \delta)(1 + 2\delta + 2\delta^2 + o(\delta^2)) + (\tfrac{1}{2} + \delta)(1 - 2\delta + 2\delta^2 - o(\delta^2)) \\
&= 1 - (2 + o(1))\delta^2.
\end{aligned}
$$

So, $V(N, \gamma N) \le 2^{(1-(2+o(1))\delta^2)N}$ and $K \ge (2+o(1))\delta^2 N$. If K were actually equal to this expression, then this would lead to condensing from input size n to $n - 2\log\frac{1}{\delta} + O(1)$ if the code were sufficiently local and s were sufficiently large as a function of n.

Our approach to finding a good code will be iterative: we first describe a way of composing covering codes which preserves the locality, and later give a "basic" construction which we compose with itself several times to get the parameters we want.

Definition 42. *Let A be a (K_1, N_1, R_1) covering code and B be a (K_2, N_2, R_2) covering code. The direct sum $A \oplus B$ of A and B is the function $f : K_1 + K_2 \to N_1 + N_2$ such that $f(x_1 \circ x_2) = A(x_1) \circ B(x_2)$ for any $x_1 \in \{0,1\}^{K_1}$ and $x_2 \in \{0,1\}^{K_2}$, where $\circ$ is the concatenation operator on strings. $\oplus^m A$ denotes the m-fold direct sum of A with itself.*

The direct sum operation has some nice properties.

Proposition 43. *If A is a t-local efficiently recoverable (K_1, N_1, R_1) covering code and B is a t-local efficiently recoverable (K_2, N_2, R_2) covering code, then $A \oplus B$ is a t-local efficiently recoverable $(K_1 + K_2, N_1 + N_2, R_1 + R_2)$ covering code. Moreover, for any m, $\oplus^m A$ is a t-local efficiently recoverable (mK_1, mN_1, mR_1) covering code.*

With this notation, we can interpret the example at the beginning of this subsection in terms of our methodology. Let R_D be the $(1, D, \lfloor D/2 \rfloor)$ repetition code, meaning that the output bit is merely repeated D times in the output. Clearly, R_D is 1-local. The covering code $\oplus^{\epsilon N} R_{1/\epsilon}$ was implicit in the construction of the example, and the fact that the construction is a hardness condenser follows from Theorem 41 and Proposition 43. As we can see, however, this code is not optimal since it condenses from n to about $n - \log\frac{1}{\delta}$ rather than the optimal $n - 2\log\frac{1}{\delta}$. The hope is that we can get better parameters using more sophisticated constructions of codes.

Indeed, we do achieve this by using Nisan's pseudo-random generator for L as a basic construction and then taking the direct sum of this construction with itself several times.

Definition 44. *A function $G : K \to N$ is a pseudorandom generator for space S with error ϵ if, for all space-S algorithms A and all inputs to A,*

$$\left| \Pr_{y \sim \{0,1\}^N} (A(y) \text{ accepts}) - \Pr_{x \sim \{0,1\}^K} (A(G(x)) \text{ accepts}) \right| \le \epsilon.$$

Lemma 45. *There is a constant a such that any PRG $G : K \to N$ for space $a \log N$ with error ϵ is a (K, N, R) covering code, provided that $V(N, R) > \epsilon 2^N$.*

Proof. Let A be the following algorithm: on input $x \in \{0,1\}^N$ and random string $y \in \{0,1\}^N$, compare x and y bit by bit. If they differ in at most R positions, accept. Certainly A can be implemented in logspace. Moreover, on truly random y, A accepts with probability $p = \frac{V(N,R)}{2^N}$, so on $G(x)$ for random x, A accepts with probability at least $p - \epsilon > 0$. Hence, for any x, there is an output of G that is within hamming distance R of x. $\square$

Theorem 46 (Nisan [Nis92]). *For any $S, N \leq 2^S$, there is a PRG $G : O(S \log N) \to N$ for space S with error 2^{-S}. Moreover, each output bit of G is computable in linear time.*

Lemma 47. *Let $\gamma = \gamma(n) < 1/2$ and $\delta = 1/2 - \gamma \in o(1)$. For any constant c, there is a $(O(\log^2 Q), Q, \gamma Q)$ covering code where $Q = \delta^{-2}$.*

Proof. By Theorem 46, for any a, there is a b such that PRG $G : O(\log^2 Q) \to Q$ for space $a \log Q$ with error b/Q. By Lemma 45, this is a $(O(\log^2 Q), Q, \gamma Q)$ covering code if $V(Q, \gamma Q) > \frac{b}{Q} 2^Q$. As shown above, for small δ,

$$\begin{aligned} V(Q, \gamma Q) &\geq \frac{1}{\sqrt{8Q}} 2^{H(\gamma)Q} \\ &= \frac{1}{\sqrt{8Q}} 2^{(1-(2+o(1))\delta^2)Q}. \end{aligned}$$

So,

$$\frac{V(Q, \gamma Q)}{2^Q} \geq \frac{1}{\sqrt{8Q}} 2^{(-(2+o(1))\delta^2)Q}.$$

If $Q = \delta^-2$, this is certainly less than b/Q. $\square$

Theorem 48. *Let $\gamma = \gamma(n) < 1/2$ and $\delta = 1/2 - \gamma \in o(1)$. Let $N = 2^n$. There is a $(O(\delta^2 \log^2(\frac{1}{\delta})N), N, \gamma N)$ covering code that is $O(\log^2 \frac{1}{\delta})$-local and recoverable in time $poly(N, 2^{\log^2 \frac{1}{\delta}})$.*

Proof. Simply apply Proposition 43 to the code of Lemma 47 with $m = \delta^2 N$. Certainly the code of Lemma 47 is $O(\log^2 \frac{1}{\delta})$-local since that is the size of the entire input. Also, since we can recover that code in time $poly(\delta^{-2}, 2^{\log^2 \frac{1}{\delta}})$ by trying every possible input, recovering the direct-product code can be done in the same amount of time times m, which is less than N. $\square$

As an immediate corollary using Theorem 41, we get:

Corollary 49. *There is an almost-explicit (running in time $poly(2^n, 2^{\log^2 \delta^{-1}})$) $(n, s, n - \lfloor 2 \log \frac{1}{\delta} + 2 \log \log \frac{1}{\delta} + O(1) \rfloor, s/n^c)$ hardness condenser from $\gamma = 1/2 - \delta = 1/2 - o(1)$ hardness for deterministic circuits to worst-case hardness for deterministic circuits. The condenser uses no advice.*

Corollary 49 is the formal version of Theorem 5. The condenser is "almost-explicit" because if $\delta = 2^{-\epsilon n}$, then the condenser takes time polynomial in 2^{n^2}.

As an immediate consequence, we get:

Theorem 50. *If there is a language $L \in$ EXP such that no circuit of size $2^{\epsilon n}$ achieves advantage $2^{-\delta n}$ almost everywhere, then there is a language $L' \in$ EXP that is worst-case hard for circuits of size $2^{\epsilon' n})$ almost everywhere, where $\epsilon' > \epsilon/(1 - 2\delta) - o(1)$.*

Proof. Define $L'_{n'}$ as follows: choose n such that $n' = n - \lfloor 2 \log 2^{\delta n} + 2 \log \log 2^{\delta)n} + O(1) \rfloor = (1 - 2\delta)n + o(n)$ and apply the condenser of Corollary 49. $\square$

6   Concluding Remarks

Our positive results on hardness extraction in the general case suffer from two deficiencies: we are only able to extract deterministic hardness from *non-deterministic* hardness, and our hardness extraction method requires an auxiliary advice string. In this section, we speculate on the possibility of removing these deficiencies.

In the direction of extracting from deterministic hardness, it would be interesting to give an alternate proof of Theorem 1. Our proof goes via pseudo-random generators but perhaps a more direct proof could be obtained using some kind of restriction method.

Such a proof may also be useful in the context of improving lower bounds for constant-depth circuits. No strongly exponential lower bounds (i.e., lower bounds of the form $2^{\epsilon n}$, where $\epsilon > 0$ is a constant) for Boolean functions in E are known relative to circuits of depth $d \geqslant 3$. The lower bounds obtained by Hastad [Hås86] are still essentially the best known. Using some variant of hardness extraction, it may be possible to obtain strongly exponential lower bounds for Boolean functions in E/n^k for some fixed k. Such a lower bound would imply an unconditional simulation of probabilistic AC_0 in $\mathsf{P}/polylog(n)$. Currently it is not even known whether probabilistic AC_0 can be simulated unconditionally in $\mathsf{P}/o(n)$.

We believe that the non-uniformity is a much more critical issue for hardness extraction, as evidenced by our lower bounds in Section 4. Performing hardness extraction uniformly is likely to require a more sophisticated understanding of the structure of Boolean circuits than we currently possess. It is an interesting question, though, whether the non-uniformity can be avoided when condensing hardness with respect to constant-depth circuits, for which fairly strong structural theorems are known [Hås86, LMN93].

7   Acknowledgments

We wish to thank Russell Impagliazzo for suggesting this problem. Discussions with Russell and Valentine Kabanets were very helpful.

References

[ACR99]   Alexander E. Andreev, Andrea E. F. Clementi, and José D. P. Rolim. Worst-case hardness suffices for derandomization: A new method for hardness-randomness tradeoffs. *TCS: Theoretical Computer Science*, 221, 1999.

[AK97]   Vikraman Arvind and Johannes Köbler. On pseudorandomness and resource-bounded measure. In *Proceedings of 17th Conference on the Foundations of Software Technology and Theoretical Computer Science*, pages 235–249. Springer, 1997.

[All01]   Eric Allender. When worlds collide: Derandomization, lower bounds, and kolmogorov complexity. *Foundations of Software Technology and Theoretical Computer Science*, 21, 2001.

[Ash65]   R.B. Ash. *Information Theory.* Dover, 1965.

[BDG88]   José Luis Balcázar, Josep Díaz, and Joaquim Gabarró. *Structural Complexity 1.* Springer-Verlag, New York, NY, 1988.

[BF90]   Donald Beaver and Joan Feigenbaum. Hiding instances in multi-oracle queries. In *Proceedings of the 7th Annual Symposium on Theoretical Aspects of Computer Science*, pages 37–48, 1990.

[BIW04]   Boaz Barak, Russell Impagliazzo, and Avi Wigderson. Extracting randomness using few independent sources. In *Proceedings of 45th Annual IEEE Symposium on Foundations of Computer Science*, pages 384–393, 2004.

[BKS+05] Barak, Kindler, Shaltiel, Sudakov, and Wigderson. Simulating independence: New constructions of condensers, ramsey graphs, dispersers, and extractors. In *Proceedings of 37th Annual ACM Symposium on Theory of Computing*, pages 1–10, 2005.

[BM84]   Manuel Blum and Silvio Micali. How to generate cryptographically strong sequence of pseudo-random bits. *SIAM Journal on Computing*, 13:850–864, 1984.

[BT03]   Andrej Bogdanov and Luca Trevisan. On worst-case to average-case reductions for NP problems. In *Proceedings of 44th Annual IEEE Symposium on Foundations of Computer Science*, pages 308–317, 2003.

[CHLL97] Gerard Cohen, Iiro Honkala, Simon Litsyn, and Antoine Lobstein. *Covering Codes*. Elsevier, 1997.

[FF93]   Joan Feigenbaum and Lance Fortnow. Random-self-reducibility of complete sets. *SIAM Journal on Computing*, 22(5):994–1005, 1993.

[FM05]   Frandsen and Miltersen. Reviewing bounds on the circuit size of the hardest functions. *IPL: Information Processing Letters*, 95, 2005.

[For97]   Lance Fortnow. Nondeterministic polynomial time versus nondeterministic logarithmic space: Time-space tradeoffs for satisfiability. In *Proceedings, Twelfth Annual IEEE Conference on Computational Complexity*, pages 52–60, Ulm, Germany, 24–27 June 1997. IEEE Computer Society Press.

[FS04]   Lance Fortnow and Rahul Santhanam. Hierarchy theorems for probabilistic polynomial time. In *Proceedings of the 45th IEEE Symposium on Foundations of Computer Science*, pages 316–324, 2004.

[GGM86]   Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, 1986.

[GNW95]   Oded Goldreich, Noam Nisan, and Avi Wigderson. On yao's XOR-lemma. *Electronic Colloquium on Computational Complexity, TR95-050*, 1995.

[Hås86]   Johan Håstad. Almost optimal lower bounds for small depth circuits. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing*, pages 6–20, 1986.

[IKW02]   Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. In search of an easy witness: Exponential time vs. probabilistic polynomial time. *Journal of Computer and System Sciences*, 65(4):672–694, 2002.

[Imp95]   Russell Impagliazzo. A personal view of average-case complexity. In *Proceedings of the 10th Annual Conference on Structure in Complexity Theory*, pages 134–147, 1995.

[Imp05]   R. Impagliazzo. Personal communication, 2005.

[IN88]   Russell Impagliazzo and Moni Naor. Decision trees and downward closures (extended abstract). In *Annual IEEE Conference on Computational Complexity (formerly Annual Conference on Structure in Complexity Theory)*, volume 3, 1988.

[ISW99]   R. Impagliazzo, R. Shaltiel, and A. Wigderson. Near-optimal conversion of hardness into pseudo-randomness. In IEEE, editor, *40th Annual Symposium on Foundations of Computer Science: October 17–19, 1999, New York City, New York,*, pages 181–190, 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA, 1999. IEEE Computer Society Press.

[IW97]   Russell Impagliazzo and Avi Wigderson. P = BPP if E requires exponential circuits: Derandomizing the XOR lemma. In *Proceedings of the 29th Annual ACM Symposium on the Theory of Computing*, pages 220–229, 1997.

[Kab01]   Valentine Kabanets. Easiness assumptions and hardness tests: Trading time for zero error. *Journal of Computer and System Sciences*, 63(2):236–252, 2001.

[KC00]   Valentine Kabanets and Jin-Yi Cai. Circuit minimization problem. In ACM, editor, *Proceedings of the thirty second annual ACM Symposium on Theory of Computing: Portland, Oregon, May 21–23, [2000]*, pages 73–79, New York, NY, USA, 2000. ACM Press.

[KL82]   Richard Karp and Richard Lipton. Turing machines that take advice. *L'Enseignement Mathématique*, 28(2):191–209, 1982.

[KvM02]   Adam Klivans and Dieter van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial hierarchy collapses. *SIAM Journal of Computing*, 31(5):1501–1526, 2002.

[Lip90]   Richard Lipton. Efficient checking of computations. In *Proceedings of 7th Annual Symposium on Theoretical Aspects of Computer Science*, pages 207–215, 1990.

[LMN93]   Nathan Linial, Yishay Mansour, and Noam Nisan. Constant depth circuits, Fourier transform and learnability. *Journal of the ACM*, 40(3):607–620, 1993.

[Lup59]   O. B. Lupanov. A method of circuit synthesis. *Izvestiya VUZ, Radiofizika*, 1(1):120–140, 1959.

[Lup65]   O. B. Lupanov. About a method of circuit design–local coding principle. *Problemy Kibernet*, 10:31–110, 1965.

[LWly]   Michael Luby and Avi Wigderson. Pairwise independence and derandomization. Technical Report TR-95-035, International Computer Science Institute, Berkeley, 1995, July.

[Nis92]   Noam Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12(4):449–461, 1992.

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

[Raz87]   Alexander Razborov. Lower bounds on the size of bounded-depth networks over the complete basis with logical addition. *Mathematical Notes of the Academy of Sciences of the USSR*, 41(4):333–338, 1987.

[Raz89]   A. A. Razborov. On the method of approximations. In *Proceedings of the Twenty First Annual ACM Symposium on Theory of Computing*, pages 167–176, Seattle, Washington, 15–17 May 1989.

[Raz05]   Ran Raz. Extractors with weak random seeds. In *Proceedings of 37th ACM Symposium on Theory of Computing*, pages 11–20, 2005.

[RR97]   Alexander Razborov and Steven Rudich. Natural proofs. *Journal of Computer and System Sciences*, 55(1):24–35, 1997.

[Sha02]   Ronen Shaltiel. Recent developments in explicit constructions of extractors. *Bulletin of the European Association for Theoretical Computer Science*, 77, 2002.

[Smo93]   Roman Smolensky. On representations by low-degree polynomials. In *FOCS*, pages 130–138, 1993.

[STV01]   Madhu Sudan, Luca Trevisan, and Salil Vadhan. Pseudorandom generators without the xor lemma. *Journal of Computer System Sciences*, 62(2):236–266, 2001.

[SU01]   Ronen Shaltiel and Christopher Umans. Simple extractors for all min-entropies and a new pseudo-random generator. In *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science*, 2001.

[Tre01]   Luca Trevisan. Extractors and pseudorandom generators. *Journal of the ACM*, 48(4):860–879, 2001.

[TSUZ01]   Amnon Ta-Shma, Christopher Umans, and David Zuckerman. Loss-less condensers, unbalanced expanders, and extractors. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing*, pages 143–152, 2001.

[Uma02]   Christopher Umans. Pseudo-random generators for all hardnesses. In *STOC: ACM Symposium on Theory of Computing (STOC)*, 2002.

[Vio05]   Emanuele Viola. The complexity of constructing pseudorandom generators from hard functions. *Computational Complexity*, 13(3):147–188, 2005.

[Yao82]   Andrew Yao. Theory and application of trapdoor functions. In *Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science*, pages 80–91, 1982.