Extractors from Reed-Muller Codes

Amnon Ta-Shma
Department of Computer Science
Tel-Aviv University
Israel 69978.
email: amnon@post.tau.ac.il. *

David Zuckerman
Department of Computer Science
University of Texas
Austin, TX 78712.
email: diz@cs.utexas.edu †

Shmuel Safra
Department of Computer Science
Tel-Aviv University
Israel 69978.
email: safra@post.tau.ac.il

April 26, 2001

Abstract

Finding explicit extractors is an important derandomization goal that has received a lot of attention in the past decade. This research has focused on two approaches, one related to hashing and the other to pseudorandom generators. A third view, regarding extractors as good error correcting codes, was noticed before. Yet, researchers had failed to build extractors directly from a good code, without using other tools from pseudorandomness. We succeed in constructing an extractor directly from a Reed-Muller code. To do this, we develop a novel proof technique.

Furthermore, our construction is the first and only construction with degree close to linear. In contrast, the best previous constructions had brought the log of the degree within a constant of optimal, which gives polynomial degree. This improvement is important for certain applications. For example, it follows that approximating the VC dimension to within a factor of $N^{1-\delta}$ is AM-hard for any positive δ.

*Some of this work was done while the author was at the University of California at Berkeley, and supported in part by a David and Lucile Packard Fellowship for Science and Engineering and NSF NYI Grant CCR-9457799.

†Some of this work was done while the author was on leave at the University of California at Berkeley. Supported in part by a David and Lucile Packard Fellowship for Science and Engineering, NSF Grant CCR-9912428, NSF NYI Grant CCR-9457799, and an Alfred P. Sloan Research Fellowship.

min-entropy k	t truly random bits	m output bits	reference
any k	$t = \log n + \Theta(1)$	$m = t + k - \Theta(1)$	Lower bound and non-explicit. [RTS00]
$k = \Omega(n)$	$t = O(\log^2 n)$	$m = \Omega(k)$	[NZ96]
$k = \Omega(n)$	$t = O(\log n)$	$m = \Omega(k)$	[Zuc97]
any k	$t = O(\frac{\log^2 n}{\log k})$	$m = k^{1-\alpha}$	[Tre99]
any k	$t = O(\log n)$	$m = k/\log n$	[RSW00]
any k	$t = O(\log n + \log^{2+o(1)} k)$	$m = k + t - O(1)$	[TSUZ01]
$k \geq \sqrt{nm}\log^2(n)$	$t = \log n + O(\log(\log^* m))$	m	This paper
$k = \Omega(n)$	$t = \log n + O(\log\log n)$	$m = \Omega(k)$	This paper

Table 1: Milestones in building explicit extractors. The error ε is a constant.

1 Introduction

1.1 History and Background

Sipser [Sip88] and Santha [San87] were the first to realize that extractor-like structures can be used to save on randomness. Sipser and Santha showed the existence of such objects, and left open the problem of explicitly constructing them. True extractors were first defined in [NZ96]:

Definition 1.1. *[NZ96] $E : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$ is an* ε-extractor *for a class of distributions $\mathbb{X}$ over $\{0,1\}^n$, if for every distribution $X \in \mathbb{X}$ the distribution $E(X,U_t)$ is within statistical distance ε from the uniform distribution on $\{0,1\}^m$.* [1] *E is explicit if $E(x,y)$ can be computed in time polynomial in the input length $n+t$. E is a* (k,ε)-extractor *if E is an extractor for all distributions with min-entropy k.*[2]

When we refer simply to extractors, we mean (k,ε)-extractors; if we mean extractors for other classes of distributions we will specify that. Extractors, thus, extract the entropy from a defective random source using few additional truly random bits. The goal is to construct extractors for any min-entropy k with t, the number of truly random bits, as small as possible and m, the number of output bits, as large as possible.

Building on earlier work of Zuckerman [Zuc90, Zuc96], Nisan and Zuckerman [NZ96] built an extractor with $t = O(\log^2 n)$ when the entropy of the source k was high, $k = \Omega(n)$. Srinivasan and Zuckerman extended this solution to the case $k = n^{1/2+\varepsilon}$ and Ta-Shma [TS96] further extended it for any entropy k. Also, Ta-Shma was the first to extract all the entropy from the source. Zuckerman [Zuc97] showed a construction with $t = O(\log n)$ working for high entropies $k = \Omega(n)$. All of this work used hashing and k-wise independence in various forms.

Departing from previous techniques, Trevisan [Tre99] showed a connection between pseudorandom generators for small circuits and extractors. Trevisan then used the Nisan-Wigderson pseudorandom generator [NW94] to construct a simple and elegant extractor that achieves $t = O(\log n)$ when $k = n^{\Omega(1)}$ and $t = O(\log^2 n)$ for the general case. Trevisan's work was extended in [ISW99, ISW00, RSW00, TSUZ01] to work for every k with only $t = O(\log n)$ truly random bits. These extensions added to the complexity of the extractor.

Thus, in the current state of the art, there are two techniques that are used in various forms and combinations and different degrees of complexity. Even after all that work, all the known constructions use $t = O(\log n)$ while the lower bound is only $t = \log n + O(1)$. This progress is summarized in Table 1.1 for the case of constant error ε.

[1] U_t denotes the uniform distribution on t bits, and $E(X,U_t)$ denotes the distribution obtained by evaluating $E(x,y)$ for x chosen according to X and y according to U_t. Also, see Section 2 for the definition of statistical distance, also known as variation distance.

[2] See Section 2 for the definition of min-entropy.

1.2 The significance of the extractor degree

Besides their straightforward applications to simulating randomized algorithms using weak sources, extractors have had applications to many areas in derandomization that are seemingly unrelated to weak sources, and below we list some of them. Extractors have been used to construct expanders that beat the second eigenvalue method [WZ99], superconcentrators and non-blocking networks [WZ99], sorting and selecting in rounds [WZ99], pseudorandom generators for space-bounded computation [NZ96], unapproximability of clique [Zuc96] and certain Σ_2^P minimization problems [Uma99], time versus space complexities [Sip88], leader election [Zuc97, RZ98], another proof that BPP $\subseteq$ PH [GZ97], random sampling using few random bits [Zuc97], and error-correcting codes with strong list decoding properties [TSZ01]. Hastad [Hås96] uses non-explicit dispersers[3] in his result that CLIQUE is unapproximable to within $n^{1-\alpha}$ for any $\alpha > 0$. The use of non-explicit dispersers make the result depend on the assumption that NP $\neq$ ZPP; a derandomized version would assume that NP $\neq$ P.

In many of these applications, extractors are viewed as highly unbalanced strong expanders. In this view an extractor is a bipartite graph $G = (V_1, V_2, E)$ with V_1 being $\{0,1\}^n$, $V_2 = \{0,1\}^m$ and an edge (x,z) exists iff there is some $y \in \{0,1\}^d$ such that $E(x,y) = z$. Thus, the degree of each vertex of V_1 is $D = 2^d$, and the extractor hashes the input $x \in \{0,1\}^n$ to a random one of its D neighbors in $\{0,1\}^m$.

Often this degree D is of more interest than $d = \log D$. For example, in the samplers of [Zuc97] the degree is the number of samples; in the simulation of BPP using weak sources [Zuc96] the degree is the number of calls to the BPP algorithm; in the extractor codes of [TSZ01] D is the length of the code; and in the unapproximability of CLIQUE, the size of the graph is closely related to D.

As stated before, all previous constructions have degree $D = \text{poly}(n) = \text{poly}(\log|V_1|)$ while the lower bound (that matches non-explicit constructions) is only $D = O(n) = O(\log|V_1|)$. In fact, when the error ε allowed is very large and close to 1, dispersers require degree which is smaller than n, namely $O(\frac{n}{\log\frac{1}{1-\varepsilon}})$. Getting such a disperser would derandomize Hastad's result.

Our construction breaks the polynomial degree bound and is the first to be close to linear. An unapproximability result that required just the extractors we construct is due to Mossel and Umans [MU01]. They showed that if dispersers with degree $n^{1+\delta}$ can be constructed for all positive δ, then it is AM-hard to approximate the VC dimension to within a factor of $N^{1-\gamma}$ for all positive γ. Our degree is even smaller, and extractors are stronger than dispersers, so the unconditional unapproximability of VC dimension follows.

1.3 Our construction

Our construction uses error-correcting codes. Codes are known to be related to extractors. For example, Ta-Shma and Zuckerman [TSZ01] showed that extractors are equivalent to codes over large alphabets having good list decoding properties. Explicit extractors correspond to codes with explicit encoding. However, they used extractors to give good list decodable codes, but not the reverse.

Earlier, Trevisan used error-correcting codes in his extractor construction. However, this construction seems to draw its power from pseudorandom generators rather than from coding theory. Indeed, Trevisan himself emphasized the use of pseudorandom generators, since he obtained relatively strong extractors without using codes at all.

Trevisan's extractor can be viewed as first encoding the weak source input $x \in \{0,1\}^n$ with any good error correcting code (good here means minimum distance close to half) and then using the truly random bits y to select bits from the encoded string using designs. Can we use a specific good code to allow y to be used in a more efficient way?

Our extractor construction is the first to do this, and we use y in a trivial way. Our good code is a Reed-Muller code. Specifically, we view the input x from the weak source as defining a low degree multivariate polynomial $\hat{x} : \mathbb{F}^d \to \mathbb{F}$ over some large field $\mathbb{F}$. We use the truly random bits to choose an element $a \in \mathbb{F}^d$ and the m outputs correspond to the values of $\hat{x}$ on the m points $a + (1,0,\dots), a + (2,0,\dots,0), \dots, a + (m,0,\dots,0)$. For each one of these points we compute $\hat{x}(a + (i,0,\dots,0))$, we encode it again using a good binary code and we output a random bit of it. Thus, the construction can be viewed as using a Reed-Muller concatenated with a good binary code for encoding, and selecting m consecutive points for the output.

[3] A disperser is a one-sided version of extractor.

For simplicity we will focus on the bivariate case, though the multivariate case works as well, and gives different parameters. Notice the simple way in which we use the random string y to select the output bits. Indeed, this gives an extractor using only $t = \log n + O(\log \frac{m}{\varepsilon})$ truly random bits. Formally, we prove:

Theorem 1. *For every $m = m(n), k = k(n)$ and $\varepsilon = \varepsilon(n)$ such that $\sqrt{n} \cdot m \cdot \log^2 n \leq k \leq n$. There is an explicit family of (k, ε) extractors $E_n : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$ with $t = \log n + O(\log \frac{m}{\varepsilon}) + O(1)$.*

To further improve our result, we notice that our construction can be viewed as an efficient reduction from the problem of constructing extractors for general sources, to the problem of constructing extractors for almost semi-random sources. We then show an efficient construction for almost semi-random sources. We use this to give a construction using only $t = \log n + O(\log(\log^* m))$ truly random bits. Formally,

Theorem 2. *For every $m = m(n), k = k(n)$ and $\varepsilon = \varepsilon(n)$ such that $\sqrt{n} \cdot m \cdot \log^2 n \leq k \leq n$ and $m \geq 2\log^2 \frac{1}{\varepsilon}$, there is an explicit family of (k, ε) extractors $E_n : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$ with $t = \log n + \mathrm{polylog}(\varepsilon^{-1}) + O(\log(\log^* m))$* [4].

Notice, that while we dramatically improve the random bit complexity, the construction works only for entropies $k = n^{1/2+\gamma}$ and the number of output bits is only k^δ for some $\delta < \gamma$. We can make the construction work for smaller entropies by using multivariate polynomials instead of bivariate polynomials; see the appendix for details. We can also extract more output bits in the case where the entropy is $k = \Omega(n)$. Formally,

Theorem 3. *For any constants $\delta, \varepsilon > 0$, there is a constant γ such that there is an explicit family of $(\delta n, \varepsilon)$ extractors $E : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$, where $t = \log n + O(\log\log n)$ and $m = \gamma n$.*

Beyond the significance of the better bounds on the degree of Theorems 2 and 3, Theorem 1 is the first (and only) purely algebraic extractor construction, and the only one relying solely on error-correcting codes. Moreover, the simplicity and good constants may make it practical, which is probably not the case for previous constructions.

1.4 Our proof technique

We want to prove that whenever X is large enough, the distribution $E(X;U)$ is close to uniform. Recall that we view an input $x \in \{0,1\}^n$ as a bivariate polynomial $\hat{x} : \mathbb{F}^2 \to \mathbb{F}$. We then use a truly random string $(a_1, a_2) \in \mathbb{F}^2$, and the m output bits are based on the m values $\hat{x}(a_1+1, a_2), \ldots, \hat{x}(a_1+m, a_2)$. Now suppose $E(X;U)$ is not close to uniform. Then there must be a distinguisher that on average can learn the value $\hat{x}(a_1+i, a_2)$ from the values $\hat{x}(a_1+1, a_2), \ldots, \hat{x}(a_1+i-1, a_2)$.

We now play a mental game. We pick a random line L and we assume someone is giving us the correct values of $\hat{x}$ on $i-1$ consecutive parallel left-shifts of L, i.e., $L-(1,0)$ through $L-(i-1,0)$. Each point on L is preceded by $i-1$ points for which we already know (by assumption) the right value of $\hat{x}$. Hence we can use the predictor to predict that point, with some moderately good success probability. Overall, the predictor is correct for many points on L. We now use the fact that $\hat{x}$ restricted to L is a low-degree polynomial to actually find the value of $\hat{x}$ on that line (using list-decoding for Reed-Solomon codes). We then learn $L+(1,0)$, $L+(2,0)$, etc., until we learn enough lines to reconstruct $\hat{x}$ itself.

Playing that mental game we can prove that there is a set of about md queries (and recall that d is about $\sqrt{n}$ and m is the number of output bits) such that almost every $x \in X$ can be reconstructed given the answers to these queries. Note that d^2 queries can always reconstruct X; our gain is that we can reconstruct X using only dm instead of d^2 queries. This shows that X has size at most $|\mathbb{F}|^{md}$, or equivalently has entropy at most $md \log q$. We conclude that if X is larger than that the distribution $E(X,U)$ is close to uniform.

The technique was inspired by work done on list decoding of Reed-Muller codes, and its application to hardness amplification in [STV99]. Aside from applying an error-correcting technique to a different information theoretic setting, our proof techniuqe has additional ideas. For example, we obtain our savings by learning a line using previously learned lines, and this whole notion of recycling queries makes sense only in our setting and does not appear in previous constructions. We believe this is a completely new, clean and elegant way of constructing extractors.

[4] The $\mathrm{polylog}(\varepsilon^{-1})$ term can be improved to $O(\log \varepsilon^{-1})$ but we postpone it to the full version of the paper.

2 Preliminaries

Throughout, $\mathbb{F} = \mathbb{F}_q$ denotes a field of size q. As usual, $[n]$ denotes the set $\{1,2,\ldots,n\}$. If S is a set and t is an element, then $S+t$ denotes the set $\{s+t : s \in S\}$. All logarithms are to the base 2. We will assume, when needed, that various quantities are integers. It is not hard to check that this has only a negligible effect on our analysis.

2.1 Statistical Distance and Min-entropy

Definition 2.1. *The* statistical distance *(or* variation distance*) between two distributions D_1 and D_2 on the same space S is*

$$\max_{T \subseteq S} |D_1(T) - D_2(T)| = \frac{1}{2} \sum_{s \in S} |D_1(s) - D_2(s)|.$$

Definition 2.2. *The* min-entropy *of a distribution D on a probability space S is* $\min_{s \in S}\{-\log_2 D(s)\}$.

2.2 Polynomials

We will use the following lemma due to Sudan.

Lemma 2.3. *[Sud97] Given a sequence of m distinct pairs pairs $\{x_i, y_i\} \in \mathbb{F}^2$, there are less than $2m/a$ degree d polynomials p such that $p(x_i) = y_i$ for at least a values of $i \in [m]$, provided that $a \geq \sqrt{2dm}$.*

Corollary 2.4. *For each element $u \in \mathbb{F}$ assign a set S_u of size at most A. Then there are less than $2A/\delta$ degree d polynomials p such that $p(u) \in S_u$ for at least a δ fraction of points, provided that $\delta \geq \sqrt{2dA/|\mathbb{F}|}$.*

2.3 Binary Codes

An $[n,k]$ code is a linear binary code of length n and dimension k. When we refer to relative measures, we mean the ordinary measures divided by the length. We will need binary codes with good combinatorial list decoding properties.

Definition 2.5. *A binary code has combinatorial list decoding property α if every Hamming ball of relative radius $\frac{1}{2} - \alpha$ has $O(1/\alpha^2)$ codewords.*

We will use codes from the following code construction due to [GHSZ00].

Fact 2.6. *There is a polynomial-time (in fact, Logspace) constructible $[n,k]$ code with combinatorial list decoding property α, where $n = O(k/\alpha^4)$.*

Simpler and more efficient constructions can be achieved with somewhat worse parameters, e.g. [NN93, AGHP92].

2.4 Reed-Muller Codes

Our construction uses a (h,D) Reed-Muller code over $\mathbb{F}_q$. In such a code the message specifies a polynomial f in D variables over $\mathbb{F}_q$ of total degree at most h, and the output is all the values of f over $\mathbb{F}_q^D$. Every polynomial in D variables of total degree $\leq h$ can be represented by the coefficients of the different monomials $x_1^{i_1} \cdot \ldots \cdot x_D^{i_D}$ with $i_1 + \ldots i_D \leq h$, and there are exactly $\binom{h+D}{D}$ such monomials. It follows that such a code has length q^D and dimension $\binom{h+D}{D}$.

3 Top-down overview

Define $h = \left\lceil \sqrt{\frac{2n}{\log q}} \right\rceil$. Let $\mathbb{F} = \mathbb{F}_q$ be a field of size $q \gg h$. We will need the characteristic of $\mathbb{F}$ to be bigger than h, so it is easiest to think of q as prime. The reduction begins by viewing the n-bit input string $x \in \{0,1\}^n$ as a function $f : \mathbb{F}^2 \to \mathbb{F}$ of total degree at most $h-1$. This is possible since every degree $h-1$ bivariate polynomial f can be specified using $\binom{h+1}{2}$ coefficients from $\mathbb{F}_q$ and $\binom{h+1}{2} \log q \geq \frac{h^2}{2} \log q \geq n$.

<u>*The function* **Z**</u>

Input : $x \in \{0,1\}^n$.

Setting : $\mathbb{F}$ is a field of size q, $h = \left\lceil \sqrt{\frac{2n}{\log q}} \right\rceil$. We associate with $x \in \{0,1\}^n$ a function $\hat{x} : \mathbb{F}^2 \to \mathbb{F}$ of total degree at most $h-1$.

Binary code : **BC** is a linear binary code of dimension $\ell = \log q$, combinatorial list decoding property α (see Definition 2.5), α will be determined later, and length $\bar{\ell}$.

Random coins : $a = (a_1, a_2) \in \mathbb{F}^2$, $j \in [\bar{\ell}]$.

Output : $\mathbf{Z}(x; a, j)_i = \mathbf{BC}(\hat{x}(a_1 + i, a_2))_j$ for $i = 1, \ldots, m$

For formulating our assertion about $\mathbf{Z}(X;U)$ we need a definition of a variant of semi-random sources. Semi random sources were defined and studied in many early papers, most notably [SV86, CG88, NZ96, SZ99]. We extend this definition to a β–almost semi-random source:

Definition 3.1. *A distribution $X = X_1 \circ X_2 \circ \ldots \circ X_m$ is β–almost α semi-random if for every $i = 1, \ldots, m$*

$$\Pr_{\substack{x \in X \\ x = x_1 \circ \ldots \circ x_m}} \left[\Pr(X_i = x_i \mid X_1 = x_1, \ldots, X_{i-1} = x_{i-1}) > \frac{1}{2} + \alpha\right] \quad \leq \quad \beta$$

If $\beta = 0$ we say X is α semi-random.

We claim:

Theorem 4. *For every n, m and $\alpha, \beta > 0$ set q the smallest prime larger than $\frac{2^9}{\beta^2 \alpha^2} h$. Let us denote $\mathbf{Z} = \mathbf{Z}(X, U_t)$ where $\mathbf{Z} = \mathbf{Z}_1 \circ \mathbf{Z}_2 \circ \ldots \circ \mathbf{Z}_m$. Then:*

- *If $\mathbf{Z}$ is not β-almost α semi-random, then $|X| \leq \frac{4}{\beta} q^{(m + \log n)h}$.*
- *$t \leq \log n + O(\log(\frac{1}{\alpha\beta})) + O(1)$ and $\mathbf{Z}$ can be computed in $O(\log q)$ space and* $\mathrm{poly}(q) = \mathrm{poly}(n, \frac{1}{\alpha\beta})$ *time.*

To verify parameters, note that $\bar{\ell} = O(\log q)$ and $\mathbf{Z}$ uses $t = 2\log q + \log \bar{\ell}$ random bits. We have: $t = 2\log q + \log \bar{\ell} \leq 2\log h + O(\log \frac{1}{\alpha\beta}) + \log\log q + O(1) = \log n - \log\log q + \log\log q + O(\log(\frac{1}{\alpha\beta})) + O(1)$. Also, the running time of $\mathbf{Z}$ is dominated by the complexity of evaluating $\hat{x}(a)$, which can be done with $O(\log q)$ space and $\mathrm{poly}(q)$ time. We will later deal with the challenging part of proving the reduction correctness.

We are now ready to prove Theorem 1:

Proof. (Of Theorem 1) Suppose $m = m(n)$, $k = k(n)$ and $\varepsilon = \varepsilon(n)$ are such that $\sqrt{n} \cdot m \cdot \log^2 n \leq k \leq n$. Let us set $\alpha = \beta = \Theta(\frac{\varepsilon}{m})$ and $q = \Theta(\sqrt{\frac{n}{\log n}} \frac{1}{\alpha^2 \beta^2})$. We claim $\mathbf{Z}(X;U)$ is the desired (k, ε) extractor.

To see that, let $X \subseteq \{0,1\}^n$ be an arbitrary set of cardinality at least $K = 2^k$. As $k \geq \sqrt{n} \cdot m \cdot \log^2 n$, $|X| \geq \frac{4}{\beta} q^{(m + \log n)\sqrt{n}}$ and by Theorem 4, $\mathbf{Z} = \mathbf{Z}(X;U)$ is a β–almost $(-\log_2(\frac{1}{2} + \alpha), \ldots, -\log_2(\frac{1}{2} + \alpha))$ semi-random source. We finish the proof with:

Claim 3.2. *$\mathbf{Z}(X;U)$ is $O(m(\alpha + \beta)) = O(\varepsilon)$ close to uniform.*

Proof. We can start with $i = m$ going down to $i = 1$. For each such i there is at most β fraction of bad prefixes. For each such bad prefix, we can redistribute its weight uniformly on all extension of the prefix. This makes at most an $m\beta$ difference to the distribution, and the new distribution is a true semi-random source.

Next, notice that for every prefix, the probability of the next bit being 0 or 1 is at most $\frac{1}{2} + \alpha$. We can again go down from $i = m$ to $i = 1$ and redistribute the weight so that it is perfectly uniform. The resulting difference is again $O(m\alpha)$. Thus, altogether, $\mathbf{Z}(X;U)$ is $O(m(\alpha+\beta))$ close to uniform. □

□

3.1 Further reducing the number of truly random bits

We now want to save the $O(\log m)$ extra term we have in t. To do that we work with a constant α, say $\alpha = .1$. Then the distribution $\mathbf{Z} = \mathbf{Z}(X;U)$ is not close to uniform. However, $\mathbf{Z}$ is β–almost α semi-random, with constant entropy in each of the bits. It therefore suffices to give an extremely efficient extractor for almost semi-random sources.

An extractor working on a general distribution over n input bits need at least $t \geq \log n - O(1)$ truly random bits, even when the allowed error ε is a constant. In contrast, we show that extractors for semi-random sources require only $t = O(1)$ truly random bits. We also show that efficient extractors for the β-almost semi-random case exist.

Theorem 5. *For every* $\alpha, \beta > 0$ *there is an* ε*-extractor* $\mathbf{F} : \{0,1\}^m \times \{0,1\}^r \to \{0,1\}^{m'}$ *for* β*-almost* α *semi-random sources with* $r = \mathrm{poly}(\log \frac{1}{p\varepsilon})$ *and* $m' = m - O(\log^* m \log \frac{1}{\varepsilon})$.

For lack of space we present the proof in the appendix. We now prove Theorem 2.

Proof. (Of Theorem 2) Suppose $m = m(n)$,$k = k(n)$ and $\varepsilon = \varepsilon(n)$ are such that $\sqrt{n} \cdot m \cdot \log^2 n \leq k \leq n$ and $m \geq 2\log^2 \frac{1}{\varepsilon}$. Let us set $-\log_2(\frac{1}{2}+\alpha) = \frac{1}{2}$, $\beta = \Theta(\frac{\varepsilon}{\log^* m})$ and $q = \Theta(\sqrt{\frac{n}{\log n}\frac{1}{\beta^2}})$. Let $\mathbf{F}(X;U)$ be the extractor for β–almost semi-random sources, of Theorem 5. Our extractor is

$$E(x; y_1, y_2) \quad = \quad \mathbf{F}(\mathbf{Z}(x; y_1); y_2).$$

That is, we first apply $\mathbf{Z}$ on the input x together with the truly random string y_1, and then we apply the extractor $\mathbf{F}$ for β–almost semi-random sources together with a new truly random string y_2.

Correctness : Let $X \subseteq \{0,1\}^n$ be an arbitrary set of cardinality at least $K = 2^k$. As $k \geq \sqrt{n} \cdot m \cdot \log^2 n$, $|X| \geq \frac{4}{\beta} q^{(m+\log n)\sqrt{n}}$ and by Theorem 4, $\mathbf{Z} = \mathbf{Z}(X;U)$ is a β–almost .1 semi-random source. It then follows by Theorem 5 that $E(X;U)$ is $O(\varepsilon)$ close to uniform.

Parameters : By Theorem 4, the length of y_1 is $\log n + 4\log(\frac{1}{\beta}) + O(1) = \log n + \log(\log^* m)) + \log \frac{1}{\varepsilon} + O(1)$. By theorem 5 the length of y_2 is $\mathrm{polylog}(\varepsilon^{-1})$. Thus the number of truly random bits used is as required. The output length of $\mathbf{Z}$ is m, and thus the output length of E is $m - O(\log^* m \log \frac{1}{\varepsilon}) \geq \frac{m}{2}$ provided that $m \geq 2\log^2 \frac{1}{\varepsilon}$. To get error ε and m output bits, just apply the above construction with $m' = 2m$ and $\varepsilon' = \Omega(\varepsilon)$.

□

3.2 Increasing the output length

We can apply techniques from [RSW00, NZ96] to increase the length of the output and prove Theorem 3:

Proof. (of Theorem 3, Sketch.) Reingold, et.al. [RSW00] showed how to add $O(\log\log n)$ truly random bits to the input X and extract a block B of length $\frac{\delta}{2}n$, such that B is close to a $2\gamma n$ source and X is close to a source with min-entropy $\frac{\delta}{4}n$, even conditioned on B. We can then apply techniques implicit in [NZ96] and explicit in [SZ99]. We use the extractor of Theorem 2 to add $\log n + O(\log^* n)$ truly random bits to X and extract $\mathrm{polylog}(n)$ almost-random bits that are close to uniform even conditioned on B. We now use these $\mathrm{polylog}(n)$ bits and the original extractor of [NZ96] (though the more complex [Zuc97] is better) to extract γn almost-random bits from B. □

4 The reduction to almost semi-random sources

We want to prove Theorem 4, i.e., we want to show that if $\mathbf{Z}$ is not as required than there exists a large subset X'' of X such that the answers to a small number of queries distinguishes elements of X''. This shows that X'' is small, and therefore X itself is small. We begin with some definitions.

Definition 4.1. *A predictor P for bivariate polynomials is a probabilistic function that on input $a \in F^2$ picks a set of queries $Q(a) = \{v_1, \dots, v_s\}$ of points in F^2, and gets s answers $b_1, \dots, b_s \in F$ from an oracle. We stress that the set $Q(a)$ may be chosen at random and may or may not depend on a. It then computes a subset $P(a; v_1, \dots, v_s, b_1, \dots, b_s) \subseteq F$.*

P has A possible answers if for every $a \in F^2$, every possible set of queries $v_1, \dots, v_s \in F^2$ and every set of answers $b_1, \dots, b_s \in F$ it holds that the size of $P(a; v_1, \dots, v_s, b_1, \dots, b_s)$ is at most A. P predicts $f : F^2 \to F$ with success p if

$$\Pr_{a \in F^2, P}[f(a) \in P(a; v_1, \dots, v_s, f(v_1), \dots, f(v_s))] \geq p$$

P predicts $S \subseteq \{0,1\}^n$ with success p if, for every $x \in S$, P predicts $\hat{x}$ with success p. P has preprocessed queries if Q does not depend on a, and P is deterministic if it does not use random coins (neither to choose Q nor to compute its answer).

Notation 4.2. *Suppose P is a predictor for bivariate polynomials and $f : \mathbb{F}^2 \to \mathbb{F}$. $P^f_Q(a)$ denotes the value $P(a; v_1, \dots, v_s, f(v_1), \dots, f(v_s))$, i.e., the query points are $Q(a)$ and the answers are the values of f on these points. Whenever the set of queries Q is clear from the context we write $P^f(a)$ to denote this value. If P is deterministic, the set of queries Q is completely determined by a, and then we always denote this value by $P^f(a)$.*

Theorem 4 follows from the following lemma:

Proposition 1. *If $\mathbf{Z} = \mathbf{Z}(X;U)$ is not β–almost α semi-random, then there exists a deterministic predictor for a set X'', of size at least $|X''| \geq \frac{\beta}{4}|X|$, using $(m + \log n)h$ preprocessed queries, 1 possible answer and success 1. Hence $|X| \leq \frac{4}{q} q^{(m+\log n)h}$.*

Notice that while in general we need about $\frac{h^2}{2}$ values to determine an arbitrary degree $h-1$ bivariate polynomial, here only about mh queries suffice. This immediately translates to X being small.

4.1 Evaluating a Point

We now fix a set X such that $\mathbf{Z} = \mathbf{Z}(X;U)$ is not β–almost α semi-random. We begin with a certain "segment predictor."

Definition 4.3. *P is a segment predictor if the set of queries is*

$$Q(a_1, a_2) = \{(a_1 - s, a_2), \dots, (a_1 - 1, a_2)\}$$

Our first step towards proving Proposition 1 will be to prove the following.

Lemma 4.4. *There exists a segment predictor $\mathbf{EP}$ for $X' \subset X$ of cardinality $|X'| \geq \frac{\beta}{2}|X|$ making at most m queries, $A = O(1)$ possible answers and $p = \frac{\beta}{4}$ success.*

Proof. Since $\mathbf{Z}$ is not β–almost α semi-random, then there is an $i_0 \in \{1, \dots, m\}$ such that

$$\Pr_{z \in \mathbf{Z}} [\Pr(\mathbf{Z}_i = z_i | \mathbf{Z}_{i-1} = z_{i-1}, \dots, \mathbf{Z}_1 = z_1) > \frac{1}{2} + \alpha] \quad \geq \quad \beta$$

We can now define $T : \{0,1\}^{i-1} \to \{0,1\}$ by letting $T(z_1, \dots, z_{i-1})$ be the most frequent value of $(\mathbf{Z}_i \mid \mathbf{Z}_1 = z_1, \dots, \mathbf{Z}_{i-1} = z_{i-1})$ breaking ties arbitrarily. We then have

$$\Pr_{z \in \mathbf{Z}} [\Pr(T(z_1, \dots, z_{i_0 - 1}) = z_{i_0}) > \frac{1}{2} + \alpha] \quad \geq \quad \beta$$

An averaging argument shows that there exists a subset $X' \subset X$ of cardinality at least $\frac{\beta}{2}|X|$ such that for every $x \in X'$,

$$\Pr_{\substack{u\in U\\ z\in \mathbf{Z}(x;u)}} [\Pr(T(z_1,\dots,z_{i_0-1}) = z_{i_0}) > \frac{1}{2}+\alpha] \quad \geq \quad \frac{\beta}{2}$$

Similarly, for every $x \in X'$

$$\Pr_{a\in\mathbb{F}^2} [\Pr_{\substack{j\in[\bar{\ell}]\\ z=\mathbf{Z}(x;a,j)}} [\Pr(T(z_1,\dots,z_{i_0-1}) = z_{i_0}) > \frac{1}{2}+\alpha] \geq \frac{\beta}{4}] \quad \geq \quad \frac{\beta}{4} \tag{1}$$

We can now define our segment predictor.

<u>**EP** *: Evaluate Point*</u>

Input : $a = (a_1,a_2) \in \mathbb{F}^2$

Queries : The query points are $Q(a_1,a_2) = \{(a_1-(i_0-1),a_2),\dots,(a_1-1,a_2)\}$. The answer to the query (a_1-i,a_2) is b_i.

Algorithm : For every $j \in [\bar{\ell}]$ compute

$$g_j(a) \quad = T(\mathbf{BC}(b_{i_0-1})_j,\dots,\mathbf{BC}(b_1)_j)$$

and set $g(w) = g_1(w)\dots g_{\bar{\ell}}(w)$.

Output : $\mathbf{EP}(w)$ is the set of all codewords of $\mathbf{BC}$ that have at least $\frac{1}{2}+\alpha$ relative agreement with $g(w)$.

Note that **EP** is deterministic and that the queries depend on the input a. We claim:

Claim 4.5. *For every $x \in X'$ we have* $\Pr_a(\hat{x}(a) \in \mathbf{EP}^{\hat{x}}(a)) \geq \frac{\beta}{4}$.

Proof. When the answers $b_1,\dots,b_{i_0-1}$ reflect the values of $\hat{x}$ we have $b_i = \hat{x}(a_1-i,a_2)$ and:

$$g_j(a) \quad = T(\mathbf{BC}(\hat{x}(a_1-i_0+1,a_2))_j,\dots,\mathbf{BC}(\hat{x}(a_1-1,a_2))_j)$$

By Equation (1) for every $x \in X'$, for at least $\frac{\beta}{4}$ of the points $a \in \mathbb{F}^2$,

$$\Pr_{j\in[\bar{\ell}]}[g_j(a) = \mathbf{BC}(\hat{x}(a))_j] \quad \geq \quad \frac{1}{2}+\alpha$$

However, whenever $\Pr_j(g_j(a) = \mathbf{BC}(\hat{x}(a))_j) \geq \frac{1}{2}+\alpha$ we have that $\hat{x}(a) \in \mathbf{EP}^{\hat{x}}(a)$. □

Finally, since **BC** has combinatorial list decoding property α for any a, $|\mathbf{EP}^{\hat{x}}(a)| \leq O(\frac{1}{\alpha^2})$. □

4.2 Evaluating a Line

We use the procedure **EP** to build a procedure **EL** (for "Evaluate-Line") that given a line L makes some specific queries and outputs a single polynomial over $\mathbb{F}$.

$\mathbf{EL}^p(L)$: *Evaluate Line*

Input : A line $L : \mathbb{F} \to \mathbb{F}^2$.

Random coins : Pick $\ell = \log q$ random points $j_1, \ldots, j_\ell \in_R \mathbb{F}$.

Algorithm :

- For every $i = 1, \ldots, q$:
  - Make the queries $\left\{v_1^i, \ldots, v_{i_0-1}^i\right\}$ necessary for evaluating $\mathbf{EP}(L(i))$. Let $b_1^i, \ldots, b_{i_0-1}^i$ be their answers.
  - Let $\mathbf{EP}(L(i)) = \mathbf{EP}(L(i); v_1^i, \ldots, v_{i_0-1}^i, b_1^i, \ldots, b_{i_0-1}^i)$.
- Form the set $S = \{(i,w) \mid i \in [1,q] \ , \ w \in \mathbf{EP}(L(i))\}$.
- Compute the list G of all univariate polynomials $g : \mathbb{F} \to \mathbb{F}$ of degree at most $h-1$ with agreement at least $\frac{\beta}{8}q$ with S.

Output :

- Query the points $L(j_1), \ldots, L(j_\ell)$, let $b_1, \ldots, b_\ell$ be the answers.
- If there is a single polynomial $g \in G$ such that $g(j_i) = b_i$ for all $i \in [\ell]$ then output g and $\mathbf{EL}(L) = g$. Otherwise output "don't know".

We now show that **EL** does well on random lines.

Lemma 4.6. *For every $x \in X'$:*

$$\Pr_{L, j_1, \ldots, j_\ell}[\mathbf{EL}^{\hat{x}}(L) \neq \hat{x}(L)] \quad \leq \quad \eta = \frac{17}{\beta q}$$

Proof. Our first claim shows that almost always G contains the right polynomial.

Claim 4.7. *For every $x \in X'$,* $\Pr_L \ (\hat{x}(L) \notin G) \ \leq \ O(\frac{1}{\beta q})$

Proof. Call $v \in \mathbb{F}^2$ *nice* for $p : \mathbb{F}^2 \to \mathbb{F}$ if $p(v) \in \mathbf{EP}^p(v)$. We know that:

- For every $x \in X'$, $\Pr_{v \in \mathbb{F}^2}[v \text{ is nice for } \hat{x}] \geq \frac{\beta}{4}$ and
- For every $p : \mathbb{F}^2 \to \mathbb{F}$ and v, $|\mathbf{EP}^p(v)| \leq A = O(\frac{1}{\alpha^2})$.

Fix any $x \in X'$. Let Y_i be the random variable indicating whether $L(i)$ is nice, and $Y = \sum_{i=1}^q Y_i$. We have $E(Y) \geq \frac{\beta}{4}q$, i.e., for every $x \in X'$ we expect to see many nice points on a random line L. We say L is bad for x if $Y \leq \frac{\beta}{8}q$. As L is a random line the points on L are pairwise independent and it follows that

$$\Pr[Y \leq \frac{E(Y)}{2}] \leq \frac{4\sigma^2(Y)}{(E(Y))^2} \leq \frac{4}{E(Y)} = \frac{16}{\beta q}$$

If the line L is bad for x (which as we saw happens with probability $O(\frac{1}{\beta q})$) we lose. Otherwise, $Y > \frac{E(Y)}{2}$ and the line L contains at least $\frac{\beta}{8}q$ nice points $v = L(i)$. Therefore, $\hat{x}(L) \in G$. $\square$

Our second claim shows that G has very few polynomials.

Claim 4.8. $|G| \leq O(\frac{1}{\beta})$

Proof. Since $|S| \leq qA \leq O(\frac{q}{\alpha^2})$ and $q \geq \frac{2^9 h}{\alpha^2\beta^2}$ we have:

$$\frac{\beta q}{8} \quad \geq \quad \sqrt{2h \cdot |S|}$$

By Lemma 2.3, $|G| \leq \frac{2qA}{\beta q/8} = O(\frac{1}{\beta})$. $\square$

Now, $j_1, \dots, j_\ell$ are taken uniformly from $\mathbb{F}$. The probability two different polynomials of degree at most $h-1$ agree on l random points is at most $(\frac{h-1}{q})^\ell$. Therefore, the probability there are two or more solutions that agree with the query on $\hat{x}(L(j_i))$ for all $i \in [\ell]$ is at most

$$\binom{|G|}{2}\left(\frac{h}{q}\right)^\ell \quad < \quad O(\frac{1}{\beta^2})(\frac{h}{q})^\ell \; < \; (\frac{1}{2})^\ell \; = \; \frac{1}{q}$$

Altogether, we fail only if the line does not have enough nice points or if we end up with two or more solutions in the last phase, and otherwise we output the right solution. $\square$

We now take a closer look at the queries done in $\mathbf{EL}(L)$. The following claim is easy to check.

Claim 4.9. *If L is not parallel to the x-axis then $\mathbf{EL}(L)$ queries the $i_0 - 1$ lines $L - (j,0)$, $j \in [i_0 - 1]$, and ℓ random points on the line L.*

As the values of $\hat{x}$ on a line can be determined by querying h points on that line, $\mathbf{EL}^{\hat{x}}$ queries only $(i_0-1)h + \ell$ points. We define the "line-shape" $LS(L)$ for a line L to be the set of $(i_0-1)h+\ell$ points in $\mathbb{F}^2$ that are queried by $\mathbf{EL}(L)$.

4.3 Proof of Proposition 1

Proof. We now give a procedure **EA** (for "Evaluate-All") that queries few points and outputs, with good probability, the unique polynomial $\hat{x} \in \hat{X}'$ that agrees with the queries.

EA *: Evaluate All*

Input : none.

Algorithm : Pick a random line L, and query the points in $LS(L)$.

For $j = 1$ to $h - i_0$

- Evaluate $\mathbf{EL}(L + (j,0))$. Note that most of the queries needed to evaluate this have been made or deduced previously; only the queries corresponding to $j_1, \dots, j_\ell$ need to be made.

Now we have evaluated a $h \times h$ block, and we extend it to the whole plane.

Now **EA** queries $(i_0-1)h + (h - i_0 + 1)\ell \leq (m+\ell)h$ points. We define the "shape" SH for a line L to be the set of points in $\mathbb{F}^2$ that are queried by **EA** when **EA** picks the random line L. We say SH is a shape, if it is the shape for some line L. Let us say that a shape SH is good for $f : \mathbb{F}^2 \to \mathbb{F}$ if $\mathbf{EA}^f_{SH}$, the output of $\mathbf{EA}^f$ when picking the shape SH, is f.

Claim 4.10. *For every $x \in X'$, $\Pr_{SH}[SH$ is not good for $\hat{x}] \leq h\eta$.*

Proof. For each of the h lines we learn, the probability we fail (given the right answers to the shape we use) is at most η. By the union bound (regardless of correlations) the claim follows. $\square$

Now, $h\eta \leq \frac{17}{\beta}\frac{h}{q} \leq \frac{17}{\beta}\frac{\beta^2}{2^9} \leq \frac{1}{2}$, thus, for every $x \in X'$, $\Pr_{SH}[SH$ is good for $\hat{x}] \geq \frac{1}{2}$. Hence, there is at least one fixed choice of a shape SH and a subset $X'' \subseteq X'$ of cardinality at least $|X''| \geq \frac{1}{2}|X'|$ such that SH is good for every $x \in X''$. Since the total number of queries is at most $(m+\ell)h \leq (m + \log n)h$, we have proved Proposition 1 and hence Theorem 4. $\square$

Acknowledgements

We thank Chris Umans and Oded Goldreich for helpful discussions and comments.

References

[AGHP92] N. Alon, O. Goldreich, J. Håstad, and R. Peralta. Simple constructions of almost k–wise independent random variables. *Random Structures and Algorithms*, 3(3):289–303, 1992.

[CG88] B. Chor and O. Goldreich. Unbiased bits from sources of weak randomness and probabilistic communication complexity. *SIAM Journal on Computing*, 17(2):230–261, 1988.

[GHSZ00] V. Guruswami, J. Hastad, M. Sudan, and D. Zuckerman. Combinatorial bounds for list decoding. In *Proceedings of the 38th Annual Allerton Conference on Communication, Control, and Computing*, pages 603–612, 2000.

[GZ97] O. Goldreich and D. Zuckerman. Another proof that BPP $\subseteq$ PH (and more). Technical Report TR97-045, Electronic Colloquium on Computational Complexity, 1997.

[Hås96] J. Håstad. Clique is hard to approximate within $n^{1-\varepsilon}$. In *Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science*, pages 627–636, 1996.

[ISW99] R. Impagliazzo, R. Shaltiel, and A. Wigderson. Near-optimal conversion of hardness into pseudorandomness. In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science*, pages 181–190, 1999.

[ISW00] R. Impagliazzo, R. Shaltiel, and A. Wigderson. Extractors and pseudo-random generators with optimal seed length. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing*, pages 1–10, 2000.

[MU01] E. Mossel and C. Umans. On the complexity of approximating the vc dimension. In *Proceedings of the 16th Annual IEEE Conference on Computational Complexity*, 2001.

[NN93] J. Naor and M. Naor. Small–bias probability spaces: Efficient constructions and applications. *SIAM Journal on Computing*, 22(4):838–856, 1993.

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

[NZ96] N. Nisan and D. Zuckerman. Randomness is linear in space. *Journal of Computer and System Sciences*, 52(1):43–52, 1996.

[RRV99] R. Raz, O. Reingold, and S. Vadhan. Extracting all the randomness and reducing the error in Trevisan's extractors. In *Proceedings of the 31st Annual ACM Symposium on Theory of Computing*, pages 149–158, 1999.

[RSW00] O. Reingold, R. Shaltiel, and A. Wigderson. Extracting randomness via repeated condensing. In *Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science*, pages 22–31, 2000.

[RTS00] J. Radhakrishnan and A. Ta-Shma. Bounds for dispersers, extractors, and depth-two superconcentrators. *SIAM Journal on Discrete Mathematics*, 13(1):2–24, 2000.

[RZ98] A. Russell and D. Zuckerman. Perfect-information leader election in $\log^* n + O(1)$ rounds. In *Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science*, pages 576–583, 1998. Final version to appear in the Journal of Computer and System Sciences.

[San87] M. Santha. On using deterministic functions in probabilistic algorithms. *Information and Computation*, 74(3):241–249, 1987.

[Sip88] M. Sipser. Expanders, randomness, or time vs. space. *Journal of Computer and System Sciences*, 36:379–383, 1988.

[STV99] M. Sudan, L. Trevisan, and S. Vadhan. Pseudorandom generators without the XOR lemma. In *Proceedings of the 31st Annual ACM Symposium on Theory of Computing*, pages 537–546, 1999.

[Sud97] M. Sudan. Decoding of Reed Solomon codes beyond the error-correction bound. *Journal of Complexity*, 13, 1997.

[SV86] M. Santha and U. V. Vazirani. Generating quasi-random sequences from semi-random sources. *Journal of Computer and System Sciences*, 33:75–87, 1986.

[SZ99] A. Srinivasan and D. Zuckerman. Computing with very weak random sources. *SIAM Journal on Computing*, 28:1433–1459, 1999.

[Tre99] L. Trevisan. Construction of extractors using pseudo-random generators. In *Proceedings of the 31st Annual ACM Symposium on Theory of Computing*, pages 141–148, 1999.

[TS96] A. Ta-Shma. On extracting randomness from weak random sources. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*, pages 276–285, 1996.

[TSUZ01] A. Ta-Shma, C. Umans, and D. Zuckerman. Loss-less condensers, unbalanced expanders, and extractors. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing*, 2001.

[TSZ01] A. Ta-Shma and D. Zuckerman. Extractor codes. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing*, 2001.

[Uma99] C. Umans. Hardness of approximating Σ_2^p minimization problems. In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science*, pages 465–474, 1999.

[WZ99] A. Wigderson and D. Zuckerman. Expanders that beat the eigenvalue bound: Explicit construction and applications. *Combinatorica*, 19(1):125–138, 1999.

[Zuc90] D. Zuckerman. General weak random sources. In *Proceedings of the 31st Annual IEEE Symposium on Foundations of Computer Science*, pages 534–543, 1990.

[Zuc96] D. Zuckerman. Simulating BPP using a general weak random source. *Algorithmica*, 16:367–391, 1996.

[Zuc97] D. Zuckerman. Randomness-optimal oblivious sampling. *Random Structures and Algorithms*, 11:345–367, 1997.

A Extractors for almost semi-random sources

A.1 The semi-random case

Extractors semi-random sources are really extractors for the more general block-wise sources.

Definition A.1. *A distribution $X = X_1 \circ X_2 \circ \ldots \circ X_m$ is a β–almost $(k_1, \ldots, k_m)$ block-wise source if for every $i = 1, \ldots, m$*

$$\Pr_{\substack{x \in X \\ x = x_1 \circ \ldots \circ x_m}} [\Pr(X_i = x_i \mid X_1 = x_1, \ldots, X_{i-1} = x_{i-1}) > 2^{-k_i}] \quad \leq \quad \beta$$

If $\beta = 0$ we say X is a $(k_1, \ldots, k_m)$ block-wise source.

The problem of constructing efficient extractors for block-wise sources was studied in [NZ96] and we present their technique. We are given a source $Z = Z_1 \circ \ldots \circ Z_b$ that is a $(k_1, \ldots, k_b)$ block-wise source, and each Z_i is distributed over $\{0,1\}^{\ell_i}$. Thus, Z is distributed over $\{0,1\}^{\ell}$ with $\ell = \sum_{i=1}^{b} \ell_i$. We also assume we have b extractors $E_1, \ldots, E_b$ where $E_i : \{0,1\}^{\ell_i} \times \{0,1\}^{r_i} \to \{0,1\}^{r_{i-1}}$. Define $E : \{0,1\}^{\ell} \times \{0,1\}^{r_b} \to \{0,1\}^{r_1}$ by

$$E(z_1, \ldots, z_b; y) \quad \stackrel{\text{def}}{=} \quad E_1(z_1; E_2(z_2; \ldots E_{b-1}(z_{b-1}; E_b(z_b; y)) \ldots)$$

Nisan and Zuckerman showed

Lemma A.2. *If each E_i is a (k_i, ε_i) extractor than E is an $\varepsilon = \sum_{i=1}^{b} \varepsilon_i$ extractor.*

We now specify the parameters and extractors. The extractor we will be working with is from ([RRV99], Theorem 4). This extractor has almost optimal entropy loss, and so

$$r_i \quad = \quad r_{i+1} + k_{i+1} - 2\log\frac{1}{\varepsilon_i} - O(1)$$

We choose the errors ε_i to be $\varepsilon_i = \frac{\varepsilon}{2 \cdot 2^{b-i}}$ and thus the total error is $\sum_{i-1}^{b} \varepsilon_i \leq \varepsilon$. We also fix the rate $\frac{\ell_i}{k_i}$ and call it δ, and so $k_i = \delta \ell_i$. In order for the extractors E_i to work we need that

$$r_i \quad \geq \quad c_0 \cdot \log^3 \ell_i \cdot \log\frac{1}{\varepsilon_i}$$

for some constant c_0. Alternatively, we can think of it as requiring that $\ell_i \geq 2^{\sqrt[3]{\frac{r_i}{c_0(b-i+1+\log\frac{1}{\varepsilon})}}}$. We pick $r_b = \max\{(c_0 \log\frac{2}{\varepsilon})^2, (2\log\frac{1}{\delta})^6, (4\log r_b)^6\}$. Working out the parameters we see that $\ell_b \geq 2^{r_b^{1/6}}$ and $r_{b-1} \geq k_b \geq 2^{r_b^{1/6}-1}$. Proceeding with this we see that $r_i > 2^{r_{i+1}^{1/6}-1}$. This reveals that after some constant number of steps, for every i $r_{i-2} \geq 2^{r_i}$ and hence $b \leq 2\log^* \ell + O(1)$. Summarizing this we get:

Theorem 6. *For the above parameters, $E : \{0,1\}^{\ell} \times \{0,1\}^{r_b} \to \{0,1\}^{r_1}$ is an ε-extractor for $(k_1, \ldots, k_b)$ block-wise sources, with $r_b = \mathrm{poly}(\log\frac{1}{\varepsilon\delta})$ and $O((\log^* \ell)^2 \log\frac{1}{\varepsilon})$ entropy loss.*

We remark that using the above technique we can further reduce the number of truly random bits to $O(\log\frac{1}{\varepsilon\delta})$ but we will not use this and we skip the details.

A.2 Reducing β–almost block-wise sources to block-wise sources

It is clear that a β–almost block-wise source with m blocks is $m\beta$ close to a block-wise source. However, it is important for us to avoid this $m\beta$ penalty. The key observation is a simple one:

Lemma A.3. *Suppose X is β–almost $(k, \ldots, k)$ block-wise source. For every $1 \leq a \leq b \leq m$*

$$\Pr_{x \in X}[\Pr(X_b = x_b \wedge \ldots \wedge X_a = x_a \mid X_1 = x_1, \ldots, X_{a-1} = x_{a-1}) > 2^{-\frac{(b-a+1)k}{2}}] \quad \leq \quad 2\beta.$$

That is, if we look at ℓ consecutive blocks, then instead of being $\ell\beta$ close to the behavior we expect from a block-wise source, we are 2β close to it.

Proof. (of Lemma A.3) Let us say X can take V possible values $x^{(1)},\ldots,x^{(V)}$, the i'th value probability p_i. Let us build a $V \times M$ table. In the i'th place of the v'th row of the table we put '*' iff

$$\Pr_{x\in X}\left(X_i = x_i^{(v)} \mid X_1 = x_1^{(v)},\ldots,X_{i-1} = x_{i-1}^{(v)}\right) \quad > \quad 2^{-k}$$

If we pick rows of the table according to the distribution X (i.e., we give the v'th row weight p_v) then we know that for every column the probability we see a '*' is at most β. In particular, if we look at the columns $a, a+1,\ldots,b$ then the probability we see a '*' is at most β. It follows that at most 2β of the rows are "bad" and contain more than $(b-a+1)/2$ '*' in the columns $a+1,\ldots,b$, and all other rows are "good". As for every good row we have $\Pr(X_b = x_b \wedge \ldots \wedge X_a = x_a \mid X_1 = x_1,\ldots,X_{a-1} = x_{a-1}) \leq 2^{-(b-a+1)k/2}$ the proof is complete. $\square$

Now, suppose $X = X_1 \circ \ldots \circ X_m$ is β–almost $(p,\ldots,p)$ block-wise source for some $p > 0$, and each X_i is distributed over $\{0,1\}$. Partition the m bits into b blocks $Z_1 \circ \ldots \circ Z_b$ of length $\ell_1,\ldots,\ell_b$ as in the preceding section. Then by Lemma A.3 Z is $O(b) = O(\beta \log^* m)$ close to a $(\ell_1,\ldots,\ell_b)$ block-wise source with $\delta = \frac{p}{2}$. In particular, the same extractor E of Theorem 6 also works here, only with a slightly larger error, and we get Theorem 5.

B The Multivariate Extractor

We now describe a generalization of the bivariate extractor to D dimensions, where $D \geq 3$. Let h be the smallest integer such that $\binom{h+D}{D} \geq n$. Let $F = \mathbb{F}_q$ be a field of size $q \gg h$. We view the n-bit input string $x \in \{0,1\}^n$ as a function $f : F^D \to F$ of total degree at most h. This is possible since to specify f we need to specify $\binom{h+D}{D}$ coefficients from F_q.

<u>*The function* **ME**$_D$</u>

Input : $x \in \{0,1\}^n$.

Setting : F is a field of size q, $h = \lfloor \frac{D}{e}(\frac{n}{\log q})^{1/D} - D \rfloor$. We identify $x \in \{0,1\}^n$ with a function $x : H^D \to F$ of total degree h.

Binary code : **BC** is a binary code with dimension $\log q$, length Z and combinatorial list decoding property $\alpha = \frac{\varepsilon}{8m}$ (see Subsection 2.3).

Random coins : $a \in F^D$, $j \in [D-1]$, $z \in [Z]$.

Output : $\mathbf{ME}(x;a,j,z)_i = \mathbf{BC}(\hat{x}(a+ie_j))_z$ for $i \in [m]$. Here e_j denotes the basis vector in F^D with a 1 in the jth position and 0's elsewhere.

Theorem 7. *For every n and m, set $\alpha = \frac{1}{8m}$ and q the smallest power of 2 larger than $\Omega(m^{\max(4,D-1)}h)$. Then* **ME** $: \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$ *is a (k,ε) strong extractor, with $k = \Omega(m^{D-1}n^{1/D}(\log n))$, $t \leq \log n + O(D^2 \log m)$, and $\varepsilon = 1 - \frac{1}{8D}$. The extractor runs in $O(\log n)$ space and time polynomial in n.*

We check parameters:

$$\begin{aligned}
t &\leq D\log q + \log D + \log(\log q \cdot (\frac{m}{\varepsilon})^4) \\
&\leq D\log(m^{D+1}h) + \log D + \log\log q + O(\log m) \\
&\leq O(D^2\log m) + \log(h^D) + \log\log q \\
&\leq O(D^2\log m) + O(D\log D) + \log n - \log\log q + \log\log q \\
&= \log n + O(D^2\log m)
\end{aligned}$$

B.1 Preliminaries

We record a generalization of Corollary 2.4.

Lemma B.1. *For each element $u \in \mathbb{F}^d$ assign a set S_u of size at most A. Then there are less than $4A/\delta$ multivariate polynomials p of total degree d such that $p(u) \in S_u$ for at least a δ fraction of points, provided that $\delta \geq 2\sqrt{2dA/|\mathbb{F}|^d}$.*

Proof. Suppose there were $4A/\delta$ such polynomials. Pick a random line L, and consider the polynomials restricted to L. By Chebychev, every such polynomial has, with high probability, at least a $\delta/2$ fraction of points $u \in L$ that satisfy $p(u) \in S_u$. By the union bound, with high probability, all such polynomials satisfy $p(u) \in S_u$ for at least a $\delta/2$ fraction of $u \in L$. Also, for every two different polynomials a random restriction will, with high probability, leave the two restrictions different. We can now apply Corollary 2.4. $\square$

B.2 A predictor in each direction

Lemma B.2. *Suppose $U \circ ME(X,U)$ is not ε close to uniform. Then there are tests $T_1, \dots, T_{D-1} : \{0,1\}^{m-1} \to \{0,1\}$ and a subset $X' \subseteq X$ of cardinality at least $\frac{1}{2}|X|$ such that for every $x \in X'$ and every $j \in [D-1]$:*

$$\Pr_{y \in \mathbf{ME}(x;U_a,j,U_z)}[T_j(U_a, j, U_z, y_1, \dots, y_{m-1}) = y_m] \quad \geq \quad \frac{1}{2} + \frac{1}{4m} \tag{2}$$

Proof. There is a test $T : \{0,1\}^m \to \{0,1\}$ that $\varepsilon = 1 - \frac{1}{8D}$ distinguishes between $\mathbf{ME}(X,U)$ and the uniform distribution. Letting U^j denote the uniform distribution on a, z with j held fixed, we get that for each $j \in [D-1]$,

$$\Pr[T(U^{j_0}, \mathbf{ME}(X,U^j)) = 1] - \Pr[T(U_{t+m}) = 1] \quad = \quad 1 - \delta_j$$

with $\sum \delta_j \leq \frac{D-1}{8D} < \frac{1}{8}$. By a Markov argument, for each j, for at least a $1 - 4\delta_j$ fraction of $x \in X$,

$$\Pr[U^j \circ T(\mathbf{ME}(x,U^j)) = 1] - \Pr[T(U_{t+m}) = 1] \geq \frac{3}{4}. \tag{3}$$

Therefore, the fraction of $x \in X$ for which (3) holds for all j is at least $1 - 4\sum \delta_j > \frac{1}{2}$. This set is X'. Now, for every $j \in [D-1]$ we use Yao's next bit predictor argument to convert T into a predictor T_j for each j. By the symmetry of **ME** , we can assume that the predictor predicts the last bit well. We obtain that for all $x \in X'$, Equation (2) holds. $\square$

B.3 Evaluating a point using a given direction

We introduce our point evaluator **EP** .

<u>**EP** *: Evaluate Point*</u>

Input : $w \in F^D$, $j \in [D-1]$

Queries : The query points are $w-(m-1)e_j, w-(m-2)e_j, \dots, w-e_j$; the answers are $b_1, \dots, b_{m-1}$.

Algorithm : For all $z \in [Z]$ compute

$$g_z(w) \quad = T(w, j, z, \mathbf{BC}(b_1)_z, \dots, \mathbf{BC}(b_{m-1})_z)$$

and set $g(w) = g_1(w) \cdots g_Z(w)$.

Output : $\mathbf{EP}(w)$ is the set of all codewords of **BC** that have at least $\frac{1}{2} + \alpha$ agreement with $g(w)$.

The proof of the following claim is the same as that for Claim 4.5.

Claim B.3. *For every $x \in X'$ and $j \in [D-1]$, $\Pr_w[\hat{x}(w) \in \mathbf{EP}^{\hat{x}}(w,j)] \geq \alpha$.*

B.4 Evaluating All

To evaluate everything, we pick a random line and call **Eval**(D,L). **Eval**(d,L) tries to compute $\hat{x}$ on the affine subspace Let $U_{d,L} = L + \text{span}\{e_1,\ldots,e_d\}$. Thus, for $d=0$ we have $U_{0,L} = L$ and we try to learn the values of $\hat{x}$ on the one-dimensional line L. For $d=1$ we try to learn the values of $\hat{x}$ on a two-dimensional affine subspace $U_{1,L} = L + \text{span}\{e_1\}$ and so forth. L is picked at random and with high probability $\text{span}\{e_1,\ldots,e_{d-1},L\}$ will be d–dimensional.

$\underline{\mathbf{Eval}^f(d,L)}$

Input : A line L and a dimension d. The queries are answered by f.

Algorithm :

If $\text{span}\{e_1,\ldots,e_{d-1},L\}$ is not d–dimensional we fail and output "don't know". Otherwise:

If $d=0$:

$U=L$. Query U on h points, interpolate the unique polynomial $p(\lambda)$ of degree less than h, and deduce $f(x)$ for all $x \in L$.

If $d \geq 1$:

- For $i = 0,\ldots,m-2$ perform **Eval**$(d-1, L+ie_d)$.
  After this we deduce a (hopefully correct) value $f(y)$ for each $y \in L + ie_d + \text{span}\{e_1,\ldots,e_d - 1\}$ and $i = 0,\ldots,m-2$.
- For $i = m-1$ to $h-1$
  - For every $u \in U_i \stackrel{\text{def}}{=} L + ie_d + \text{span}\{e_1,\ldots,e_{d-1}\}$ evaluate **EP**(u,d).
    Note that the queries needed for **EP**(u,d) have been made or previously deduced.
  - Define an affine map $\phi_i : F^d \to U_i$ and form the set $S = \{(v,w) \mid v \in F^d\ ,\ w \in \mathbf{EP}(\phi_i(v),d)\}$.
    Compute the list G of all d-variate polynomials $g : F^d \to F$ of degree at most h with agreement at least $\frac{\alpha}{2}q^d$ with S.
  - Pick $\ell = \Theta(d \log q + \log \frac{1}{\alpha})$ random points $j_1,\ldots,j_\ell \in F^d$. Query the points $\phi_i(j_1),\ldots,\phi_i(j_\ell)$, and let $b_1,\ldots,b_\ell$ be the answers.
    If there is a single polynomial $g \in G$ such that $g(j_i) = b_i$ for $i \in [\ell]$, then deduce that for all v, $f(\phi_i(v)) = g(v)$. Otherwise output "don't know".

Lemma B.4. $\Pr_L[\mathbf{Eval}(d,L) \neq \hat{x}(U_{d,L})] \leq O(\frac{m^{d-1}h}{\alpha q})$.

Proof. The probability that $\text{span}\{e_1,\ldots,e_{D-1},L\}$ is not D–dimensional is at most $\frac{1}{q}$. If $\text{span}\{e_1,\ldots,e_{D-1},L\}$ is D–dimensional then for every $d = 1,\ldots,D-1$ it must be that $\text{span}\{e_1,\ldots,e_{d-1},L\}$ is d–dimensional. From now on we assume $\text{span}\{e_1,\ldots,e_{D-1},L\}$ is D–dimensional.

Let us define error_d to be the probability (over choosing a random line L) that $\mathbf{Eval}(d,L) \neq \hat{x}(U_{d,L})$, given that all queries are answered correctly. Clearly, $\text{error}_0 = 0$. Also, we will soon prove the recursion:

Claim B.5. $\text{error}_d \leq (m-1)\text{error}_{d-1} + O(\frac{h}{\alpha q^d})$

and solving the recursion we get our result. □

Proof. (of Claim B.5) The first term in the recursion comes from the stage where i runs from 0 to $m-2$ and we call **Eval**$(d-1,\cdot)$, which has probability of error_{d-1} to fail.

Each i from $m-1$ to $h-1$ causes an additional error, which we analyze analogously to **EL**. Call $v \in F^D$ *nice* for $p : F^D \to F$ if $p(v) \in \mathbf{EP}^p(v)$. We know that:

- For every $x \in X'$, $\Pr_{v \in F^D}[v \text{ is nice for } \hat{x}] \geq \alpha$ and

- For every $p : F^D \to F$ and v, $|\mathbf{EP}^p(v)| \le A \le \frac{1}{\alpha^2}$.

Fix any $x \in X'$. For $v \in U_i$, let Y_v be the indicator random variable that is 1 iff v is nice, and $Y = \sum_{v \in U_i} Y_v$. We have $E(Y) \ge \alpha |U_i| = \alpha q^d$, i.e., for every $x \in X'$ we expect to see many nice points in U_i. We say U_i is bad for x if $Y \le E(Y)/2$. As the points in U_i are pairwise independent,

$$\Pr[Y \le \frac{E(Y)}{2}] \le \frac{4\sigma^2(Y)}{(E(Y))^2} \le \frac{4}{E(Y)} = O(\frac{1}{\alpha q^d})$$

If U_i is bad for x we lose. Otherwise, $Y > \frac{E(Y)}{2}$ and U_i contains at least $\frac{\alpha}{2} q^d$ nice points. Therefore, $\hat{x}(L) \in G$.

We can then apply Lemma B.1 to bound the number of polynomials in G, and we deduce that $|G| \le \frac{8A}{\alpha}$. Now, $j_1, \ldots, j_\ell$ are taken uniformly from F. The probability two different polynomials of degree at most h agree on l random points is at most $(\frac{h}{q})^\ell$. Therefore, the probability there are two or more solutions that agree with the query on $\hat{x}(L(j_i))$ for all $i \in [\ell]$ is at most

$$\binom{|G|}{2}\left(\frac{h}{q}\right)^\ell \quad < \quad O(\frac{A^2}{\alpha^2}(\frac{h}{q})^\ell \;\le\; O(\frac{1}{\alpha^6}(\frac{1}{2})^\ell) \;\le\; \frac{1}{q^d}$$

Altogether, we fail only if the line does not have enough nice points or if we end up with two or more solutions in the last phase, and otherwise we output the right solution. □

Now let queries$_d$ denote the number of queries made in $\mathbf{Eval}(d, \cdot)$.

Lemma B.6. *queries$_d \le m^{d-1}(m+\ell)h \le 2m^d h$.*

Proof. Note that queries$_0 = h$, and for $d \ge 1$

$$\text{queries}_d \le (m-1)\text{queries}_{d-1} + \ell(h - m + 1).$$

Now solve the recursion. □

Notice that there is nothing special about our choice of $e_1, \ldots, e_{D-1}$. Any set of $D-1$ independent vectors will do for the construction.

The extractor $\mathbf{ME}$ works with one of $D-1$ fixed directions $e_1, \ldots, e_{D-1}$ and has a very large error (close to one). We can instead define an extractor $\mathbf{ME}'$ that picks a direction as part of its random coins. We claim that $\mathbf{ME}'$ has a very small error. To see that, notice that if $\mathbf{ME}'$ is not an extractor, then there must be $D-1$ independent directions for which the $\mathbf{ME}'$ does not work. We then continue as with the proof for $\mathbf{ME}$ and we reach a contradiction. We leave the details for the full version of the paper.