On Extracting Randomness From Weak Random Sources * Extended Abstract

Amnon Ta-Shma[†]

Abstract

We deal with the problem of extracting as much randomness as possible from a defective random source. We devise a new tool, a "merger", which is a function that accepts d strings, one of which is uniformly distributed, and outputs a single string that is guaranteed to be uniformly distributed. We show how to build good explicit mergers, and how mergers can be used to build better extractors.

Previous work has succeeded in extracting "some" of the randomness from sources with "large" min-entropy. We improve on this in two respects. First, we build extractors for any source, whatever its min-entropy is, and second, we extract all the randomness in the given source.

Efficient extractors have many applications, and we show that using our extractor we get better results in many of these applications, e.g., we achieve the first explicit N-superconcentrators of linear size and $polyloglog(N)$ depth.

1 Introduction

1.1 The Problem

We deal with the problem of "extracting" as much randomness as possible from a "defective" source of randomness. There are various ways to define what a "defective" source is, the most general one [Zuc90] assumes nothing about the nature of the source, except that no string has too high a probability in the given distribution:

DEFINITION 1.1 *The min-entropy of a distribution D is $H_\infty(D) = min_x(-log(D(x)))$.*

An extractor [NZ93] is a function $E(x, y)$ s.t. the distribution of $E(x, y)$ is very close to uniform, for any source X with large min-entropy, when x is taken from X and y is a short truly random string. I.e., the extractor uses the short random string y to extract the randomness present in X.

DEFINITION 1.2 (A variation on [NZ93][1]) $E : \{0,1\}^n \times \{0,1\}^t \mapsto \{0,1\}^{m'}$ *is an (n, m, t, m', ϵ)-extractor, if for any distribution X on $\{0,1\}^n$ whose min-entropy is at least m, the distribution of $E(x, y)$ when choosing x according to the distribution X and y uniformly in $\{0,1\}^t$, is within statistical distance of ϵ from uniform.*

1.2 Previous Work and Our New Extractor

Extractors have been studied extensively in many papers [Zuc90, Zuc91, NZ93, SZ94, Zuc93, Zuc]. Much research has also been done on a related structure, called disperser, in which the output bits are not necessarily close to uniform, but have a weaker random property that suffices ,e.g., for "RP simulations" [WZ93, SSZ95]. In the following table we list some of the currently known explicitly constructible extractors:

m min-ent	t random bits	m'	reference
$\Omega(n)$	$O(log^2 n \cdot log(\frac{1}{\epsilon}))$	$\Omega(m)$	[NZ93]
$\Omega(n)$	$O(log(n) + log(\frac{1}{\epsilon}))$	$\Omega(m)$	[Zuc]
$\Omega(n^{1/2+\gamma})$	$O(log^2 n \cdot log(\frac{1}{\epsilon}))$	$n^\delta, \delta \leq \gamma$	[SZ94]
$\Omega(n^\gamma)$	$O(logn)$	$n^\delta, \delta < \gamma$	[SSZ95] Disperser

*This work was supported by BSF grant 92-00043 and by a Wolfeson award administered by the Israeli Academy of Sciences.

[†]Institute of Computer Science, Hebrew University of Jerusalem. email: am@cs.huji.ac.il

[1]see remark 3.1.

STOC'96, Philadelphia PA,USA
© 1996 ACM 0-89791-785-5/96/05..$3.50

We improve upon previous results in two ways. First, our extractor works for any min-entropy, *small or large*. Second, we extract *all* the randomness present in the source:

Theorem 1 *For every $\epsilon = \epsilon(n)$ and $m = m(n) \leq n$, there is an explicit family of $(n, m, t = poly(log(n), log(\frac{1}{\epsilon})), m' = m, \epsilon)$-extractors E_n, that can be constructed in polynomial time in n.*

In addition we use our new technique to construct a different extractor:

Theorem 2 *For every constant k and $\gamma > 0$ there is some constant $\delta > 0$ and an $(n, n^{\gamma}, O(log(n)log^{(k)}n), \Omega(n^{\delta}), \frac{1}{n})$ extractor, where $log^{(k)}n = \underbrace{loglog \ldots log}_{k}\ n$.*

Non-constructive extractors require only $O(log(n) + log(\frac{1}{\epsilon}))$ truly random bits, matching the current lower bound of [NZ93], while our construction requires $poly(log(n), log(\frac{1}{\epsilon}))$ truly random bits. Thus, we can add to the above table:

m	t random bits	m'	ref
any m	$poly(log(n) + log(\frac{1}{\epsilon}))$	m	Thm 1
$\Omega(n^{\gamma})$	$O(log(n) \cdot log^{(k)}n)$	$\Omega(n^{\delta})$	Thm 2
any m	$O(log(n) + log(\frac{1}{\epsilon}))$	m	Lower bound

Reducing t to the optimal remains the major open problem.

1.3 Our Technique

The main new tool we use is the notion of a "merger". This is a function that accepts d strings, one of which is uniformlly distributed, and outputs a single string which is guaranteed to be uniformly distributed. The difficulty of designing such a merger is that we do not know beforehand which of the strings is the uniformly distributed one, and that the other strings may be correlated with it in an arbitrary manner. In fact, we use the merger under more general conditions: the index of the "uniformly distributed" string may be a random variable itself, and may be correlated with the input strings.

Mergers, as extractors, extract randomness from a defective source of randomness. However, mergers deal with random sources having "simple" structure, which simplifies the task of constructing good mergers.

We briefly illustrate how mergers can be used to build extractors. Suppose for any random source X there is some hint $Y = Y(x)$ s.t. it is easy to extract randomness from $(X \mid Y)$, and suppose we do not know what this value $Y = Y(X)$ is. Using mergers, we can build an extractor by trying to extract randomness from $(X \mid Y = y)$ for all possible values y of Y, and then *merging* all these possibilities into a single quasi-random string using a merger.

1.4 Applications

Our new construction improves many of the known applications of extractors. Most of the applications we list here, can be obtained by plugging our new extractor into previous constructions.

Corollary 1.1 *(following [WZ93]) For any N and $1 \leq a \leq N$ there is an explicitly constructible a-expanding graph with N vertices, and maximum degree $O(\frac{N}{a}2^{polyloglog(N)})$* [2] .

Corollary 1.1 has applications on sorting and selecting in k rounds. See [WZ93, Pip87, AKSS89] for more details.

Corollary 1.2 *(following [WZ93]) For every N there is an efficiently constructible depth 2 superconcentrator over N vertices with size $O(N \cdot 2^{polyloglog(N)})$.*

[WZ93] prove that a direct corollary of cor 1.3 is:

Corollary 1.3 *([WZ93]) For any N there is an explicitly constructible superconcentrator over N vertices, with linear size and $polyloglog(N)$ depth* [3].

Corollary 1.4 *For any $\delta > 0$ and $k > 0$, BPP can be simulated in time $n^{O(log^{(k)}n)}$ using a weak random source X with minentropy at least n^{δ}* [4] .

Our results can also prove a deterministic version of the hardness of approximating the iterated log of MaxClique. See [Zuc93] for more details.

1.5 Paper Organization

In section 2 we describe our notation, and in section 3 we give some preliminary definitions and results. In section 4 we prove Theorem 1, assuming two theorems on mergers and composition of extractors which we prove in section 5 and section 6 respectively.

Due to lack of space, we omit many of the technical proofs. Also, the proof of Theorem 2 will only appear in the final version of the paper.

[2] The obvious lower bound is $\frac{N}{a}$ The previous upper bound [WZ93, SZ94] was $O(\frac{N}{a} \cdot 2^{log(N)^{1/2+o(1)}})$

[3] This improves the current upper bound of $O(log(N)^{1/2+o(1)})$

[4] This improves the [SZ94] $n^{O(log(n))}$ bound. For RP, [SSZ95] showed this can be done in polynomial time. The corollary immediately follows Theorem 2. A proof of Theorem 2 will appear in the final version of the paper.

2 Notation

We use standard notation for random variables and distributions. We distinguish between random variables and distributions: the value of a random variable is the result of some trial done in a probability space, and thus random variables may be correlated, as opposed to distributions that are merely functions assigning probabilities to events. Given a random variable A we denote by $\overline{A}$ the distribution A induces.

We use capital letters to denote random variables and distributions, and small letters to denote elements. If $\overline{X}$ is a distribution, $x \in \overline{X}$ denotes picking x according to the distribution $\overline{X}$. If A and B are (possibly correlated) random variables then $(A \mid B = b)$ is the conditional distribution of A given that $B = b$. We denote by U_t the uniform distribution over $\{0,1\}^t$.

For a random variable $X = X_1 \circ \ldots \circ X_n$ over $\{0,1\}^n$ we write $X_{[i,j]}$ as an abbreviation to the random variable $X_i \circ X_{[i+1]} \ldots \circ X_j$, where $\circ$ stands for concatenation. The same applies for instances $x_{[i,j]}$.

We define the variation distance between two distributions $\overline{X}$ and $\overline{Y}$ defined over the same space, as $d(\overline{X},\overline{Y}) = \frac{1}{2}|\overline{X} - \overline{Y}|_1 = \frac{1}{2}\Sigma_a|\overline{X}(a) - \overline{Y}(a)|$. We say $\overline{X}$ is ϵ-close to $\overline{Y}$ if $d(\overline{X},\overline{Y}) \le \epsilon$. We say two random variables A and B are ϵ–close if $d(\overline{A},\overline{B}) \le \epsilon$. We say $\overline{X}$ is ϵ quasi-random if it is ϵ-close to uniform.

3 Preliminaries

3.1 Extractors

DEFINITION 3.1 *The min-entropy of a distribution $\overline{D}$ is $H_\infty(\overline{D}) = min_x(-log(\overline{D}(x)))$.*

DEFINITION 3.2 *$E : \{0,1\}^n \times \{0,1\}^t \mapsto \{0,1\}^{m'}$ is an (n,m,t,m',ϵ)-extractor, if for any distribution $\overline{X}$ on $\{0,1\}^n$ with $H_\infty(\overline{X}) \ge m$, the distribution of $E(x,y)$ when choosing $x \in \overline{X}$ and $y \in U_t$, is ϵ close to $U_{m'}$.*

REMARK 3.1 *This definition is slightly different from the one in [NZ93, SZ94], where E is called an extractor if $E(x,y) \circ y$ is close to uniform, while we only demand that $E(x,y)$ is close to uniform.*

3.2 Explicit Extractors

DEFINITION 3.3 *We say $E = \{E_n\}$ is an explicit (n,m,t,m',ϵ) -extractor, if there is a Turing machine that given $x \in \{0,1\}^n$ and $y \in \{0,1\}^t$, outputs $E_n(x,y)$ in polynomial time in $n+t$.*

As mentioned in the introduction, various explicit extractors have been built so far. Of those we use the following two extractors:

Lemma 3.1 *[SZ94] There is some constant $c > 1$ s.t. for any $m = \Omega(log(n))$ there is an explicit $(n, 2m, m, cm, 2^{-m/2})$- extractor A_m. We denote this constant c by c_{sz}.*

Lemma 3.2 *[SZ94] [5] Let $m(n) \ge n^{1/2+\gamma}$ for some constant $\gamma > 0$, then for any ϵ there is an explicit $(n, m(n), O(log^2 n \cdot log(\frac{1}{\epsilon})), \frac{m^2(n)}{n}, \epsilon)$- extractor.*

The first extractor is actually a restatement of the existence of explicit tiny families of hash functions [SZ94, GW94], and can easily be achieved using small ϵ–biased sample spaces [SZ94].

We also need the following simple lemma from [NZ93]:

Lemma 3.3 *[NZ93] Let X and Y be two correlated random variables. Let $\overline{B}$ be a distribution, and call an x "bad" if $(Y \mid X = x)$ is not ϵ close to $\overline{B}$. If $Prob_{x \in \overline{X}}(x \text{ is bad}) \le \eta$ then $\overline{X \circ Y}$ is $\epsilon + \eta$ close to $\overline{X} \times \overline{B}$.*

3.3 More of The Same

Suppose we have an extractor E that extracts randomness from any source having at least m min-entropy, how much randomness can we extract from sources having M min-entropy when $M >> m$?

The following algorithm is implicit in [WZ93]: use the same extractor E many times over the same string x, each time with a fresh truly random string r_i, until you get $M - m$ output bits. The idea is that as long as the length of $E(x,r_1) \circ \ldots \circ E(x,r_k)$ is less then $M - m$, then with high probability $(X \mid E(x,r_1) \circ \ldots \circ E(x,r_k))$ still contains m min-entropy, and therefore we can use the extractor E to further extract randomness from it. Simple calculations give the following two lemmas:

Lemma 3.4 *Suppose that for some $m = m(n)$ there is an explicit (n,m,t,m',ϵ)-extractor E_m. Then, for any $M = M(n) \ge m$, and any safety parameter $s > 0$, there is an explicit $(n, M, kt, min\{km', M - m - s\}, k(\epsilon + 2^{-s}))$-extractor.*

Lemma 3.5 *Suppose that for any $m \ge \bar{m}$ there is an explicit $(n, m, t(n), \frac{m}{f(n)}, \epsilon(n))$-extractor E_m. Then, for any m, there is an explicit $(n, m, O(f(n)log(n)t(n)), m - \bar{m}, log(n)(\epsilon + 2^{-t(n)}))$-extractor.*

4 Main Theorem

In this section we prove Theorem 1, using our new tool of mergers. In subsection 4.1 we define what

[5] The parameters here are simplified. The real parameters appearing in [SZ94] are somewhat better

a "merger" is, and in subsection 4.2 we explicitly build "mergers". In subsection 4.3 we use mergers to efficiently compose extractors, and in subsection 4.4 we show how good somewhere random mergers imply good extractors. Finally, in subsection 4.5 we use this to prove our main theorem.

4.1 Somewhere Random Mergers

DEFINITION 4.1 *$X = X_1 \circ \ldots \circ X_d$ is a d-block (m, ϵ, η) somewhere random source, if each X_i is a random variable over $\{0,1\}^m$, and there is a random variable Y over $[0..d]$ s.t. for any $i \in [1..d]$: $d((X_i | Y = i), U_m) \le \epsilon$ and $Prob(Y = 0) \le \eta$. We also say that Y is an (m, ϵ, η) selector for X.*

A somewhere random merger is a function extracting randomness from a somewhere random source:

DEFINITION 4.2 *$M : \{0,1\}^{dm} \times \{0,1\}^t \mapsto \{0,1\}^{m'}$ is a (d, m, t, m', ϵ)-somewhere random merger, if for any d-block $(m, 0, 0)$ somewhere random source X, the distribution of $E(x, y)$ when choosing $x \in \overline{X}$ and $y \in U_t$, is ϵ close to $U_{m'}$.*

4.2 Explicit Somewhere Random Mergers

DEFINITION 4.3 *We say $M = \{M_n\}$ is an explicit (d, m, t, m', ϵ) -somewhere random merger, if there is a Turing machine that given $x \in \{0,1\}^{dm}$ and $y \in \{0,1\}^t$ outputs $M_n(x, y)$ in polynomial time in $dm + t$.*

It turns out that any $(2m, m, t, m', \epsilon)$–extractor is a $(2, m, t, m', \epsilon)$–somewhere random merger. Having a 2–block somewhere random merger, we build a d–block merger by merging the d blocks in pairs in a tree wise fashion. Thus, we have the following theorem, which we prove in detail in section 5:

Theorem 3 *Assume for every m there is an explicit $(2m, m, t(m), m - k(m), \epsilon(m))$ extractor E_m, for some monotone functions t and k. Then there is an explicit $(2^l, m, l \cdot t(m), m - l \cdot k(m), l \cdot \epsilon(m))$ somewhere random merger.*

The [SZ94] extractor of lemma 3.2 works for any source with $H_\infty(\overline{X}) \ge n^{1/2+\gamma}$. Thus, using lemma 3.4, by repeatedly using the [SZ94] extractor, we can extract from a source having $H_\infty(\overline{X}) \ge \frac{n}{2}$ at least $\frac{n}{2} - n^{1/2+\gamma}$ quasi-random bits. Thus, we have a 2–merger that does not lose much randomness in the merging process. Applying Theorem 3 we get a good n–merger. Thus we have the following lemma, that is proven in detail in Appendix A :

Lemma 4.1 *Let $b > 1$ be any constant and suppose $f = f(m) = f(m(n))$ is a function s.t. $f(m) \le \sqrt[3]{m}$ and for every $m \ge m_0(n)$: $f(m) \ge b \cdot log(n)$. Then for every $m \ge m_0$ there is an explicit $(n, m, log(n) \cdot polylog(m) \cdot f^2(m) \cdot log(\frac{1}{\epsilon}), m - \frac{m}{b}, log(n) \cdot poly(m) \cdot \epsilon)$ somewhere random merger.*

Corollary 4.2 *For every $m \ge 2^{\sqrt{log(n)}}$, there is an $(n, m, polylog(n) \cdot log(\frac{1}{\epsilon}), \Omega(m), poly(n) \cdot \epsilon)$-somewhere random merger $M = M_m$.*

Proof: Take $f(m) = log^d m$ for some constant $d > 2$. For any constant b, $m \ge 2^{\sqrt{log(n)}}$ and n large enough, $log^d m \ge b \cdot log(n)$, and the corollary follows lemma 4.1. □

Notice that Theorem 3 and corollary 4.2 take advantage of the simple structure of somewhere random sources, giving us an explicit somewhere random merger that works even for sources with very small min-entropy that can not use the [SZ94] extractor of lemma 3.2.

4.3 Composing Extractors

[CG88] showed how to extract randomness from sources X that can be "broken" into blocks $X_1 \circ X_2$, s.t. X_1 and $(X_2 \mid X_1 = x_1)$ contain a lot of randomness. Furthermore, for any source X, for most $x \in X$ there is a good splitting point i (this is highly informal, and will be stated precisely soon). This leads to the following algorithm:

Given an input string x split it into two consecutive strings $x_1 \circ x_2$, use E_1 to extract randomness from x_2, and use E_2 with the *extracted randomness* to further extract randomness from x_1. Obviously, given a string x we do not know what is the right splitting point, so we try all $|x| = n$ possible splitting points. This gives us a somewhere random source with n blocks, that can be merged into a single quasi-random string by a good somewhere random merger.

ALGORITHM 4.1 *Suppose E_1 is an $(n, m_1, t_1, t_2, \zeta_1)$-extractor, E_2 is an $(n, m_2, t_2, t_3, \zeta_2)$-extractor, and M is an $(n, t_3, \mu_1, o_1, \zeta_3)$-somewhere random merger. Define the function $E_2 \overset{M}{\odot} E_1$ as follows: Given $a \in \{0,1\}^n$, toss $r_1 \in \{0,1\}^{t_1}$, and $r_2 \in \{0,1\}^{\mu_1}$.*

- *Let $q_i = E_1(a_{[i,n]}, r_1)$ and $z_i = E_2(a_{[1,i-1]}, q_i)$, for $i = 1, \ldots, n$.*
- *Let $E_2 \ominus E_1 = z_1 \circ \ldots \circ z_n$, and $E_2 \overset{M}{\odot} E_1 = M(E_2 \ominus E_1, r_2)$.*

Theorem 4 *Suppose E_1 is an $(n, m_1, t_1, t_2, \zeta_1)$-extractor, E_2 is an $(n, m_2, t_2, t_3, \zeta_2)$-extractor, and M is an $(n, t_3, \mu_1, o_1, \zeta_3)$-somewhere random merger. Then for every safety parameter $s > 0$, $E_1 \overset{M}{\odot} E_2$ is an $(n, m_1 + m_2 + s, t_1 + \mu_1, o_1, \zeta_1 + \zeta_2 + \zeta_3 + 8n2^{-s/3})$-extractor.*

Now we define composure of many extractors by:

DEFINITION 4.4 *Suppose E_i is an $(n, m_i, t_i, t_{i+1} + s_{i+1}, \zeta_i)$–extractor, for $i = 1, \ldots, k$, $s_i \geq 0$ and $s_2 = 0$. Suppose M_i is an $(n, t_{i+2} + s_{i+2}, \mu_i, t_{i+2}, \bar{\zeta}_i)$ –somewhere random merger, for any $i = 1 \ldots k - 1$. We define by induction the function $E_k \overset{M_{k-1}}{\odot} E_{k-1} \overset{M_{k-2}}{\odot} \ldots E_2 \overset{M_1}{\odot} E_1$ to equal $E_k \overset{M_{k-1}}{\odot} (E_{k-1} \overset{M_{k-2}}{\odot} \ldots E_2 \overset{M_1}{\odot} E_1)$.*

Theorem 5 *Suppose E_i, M_i are as above, then for any safety parameter $s > 0$, $E = E_k \overset{M_{k-1}}{\odot} E_{k-1} \overset{M_{k-2}}{\odot} \ldots E_2 \overset{M_1}{\odot} E_1$ is an $(n, \Sigma_{i=1}^{k} m_i + (k-1)s, t_1 + \Sigma_{i=1}^{k-1} \mu_i, t_{k+1}, \Sigma_{i=1}^{k} \zeta_i + \Sigma_{i=1}^{k-1} \bar{\zeta}_i + (k-1)n2^{-s/3+3})$ –extractor. If E_i, M_i are explicit, then so is E.*

REMARK 4.1 *It may appear that left associativity is more economic in terms of number of truly random bits used. However, we know how to implement right associativity composure in polynomial time (using a dynamic programming procedure) and we do not know of such an algorithm for left associativity composure-s.*

4.4 Good Somewhere Random Mergers Imply good Extractors

Assume for every $m \geq \bar{m}$ we have a good somewhere random merger M. Then we can let $E = A_m \overset{M}{\odot} \ldots A_{b^2\bar{m}} \overset{M}{\odot} A_{b\bar{m}} \overset{M}{\odot} A_{\bar{m}}$, where A_i is the extractor of lemma 3.1 and b is some constant, $1 < b < c_{sz}$, to get an extractor that extracts $\Omega(m)$ bits from sources having m min-entropy. Using lemma 3.5, we get an extractor that extracts m bits. Thus, we see that good somewhere random mergers imply good extractors. We prove the following lemma in detail in Appendix A :

Lemma 4.3 *Suppose $\bar{m} = \bar{m}(n)$ is a function s.t. for every $m \geq \bar{m}(n)$ there is an explicit $(n, m, t, m - \frac{m}{\bar{c}}, \epsilon)$ somewhere random merger, where $\bar{c}$ is the constant $\frac{2c_{sz}}{c_{sz}-1}$, and c_{sz} is the constant in lemma 3.1. Then for any m there is an explicit $(n, m, O(\bar{m} \cdot log(n) \cdot log(\frac{1}{\epsilon}) + log^2(n) \cdot t), m, poly(n) \cdot \epsilon)$– extractor.*

Unfortunately, corollary 4.2 asserts the existence of good mergers only for $m \geq 2^{\sqrt{log(n)}}$, and therefore plugging this into lemma 4.3 we get:

Corollary 4.4 *For every m there is an $(n, m, O(2^{\sqrt{log(n)}} \cdot polylog(n) \cdot log(\frac{1}{\epsilon})), m, poly(n) \cdot \epsilon)$–extractor B_m.*

4.5 Putting It Together

The extractor B in corollary 4.4 uses $O(2^{\sqrt{log(n)}} \cdot polylog(n) \cdot log(\frac{1}{\epsilon}))$ truly random bits to extract all of the randomness in the given source. Although $O(2^{\sqrt{log(n)}} \cdot polylog(n) \cdot log(\frac{1}{\epsilon}))$ is quite a large amount of truly random bits, we can use the [SZ94] extractor, to extract $n^{1/3}$ bits from $n^{2/3}$ min-entropy, and then use these $n^{1/3} >> O(2^{\sqrt{log(n)}} \cdot polylog(n) \cdot log(\frac{1}{\epsilon}))$ bits, to further extract all of the remaining min-entropy. More precisely, if B is the extractor in corollary 4.4, E_{sz} the extractor from lemma 3.2 and M the merger from corollary 4.2, then $E = B \overset{M}{\odot} E_{sz}$ extracts $\Omega(m)$ bits from sources having $m \geq n^{2/3}$ min-entropy, using only $polylog(n)$ truly random bits! That is, we get the following lemma (whose proof appears in Appendix A) :

Lemma 4.5 *Let $\epsilon \geq 2^{-n^{\gamma}}$ for some constant $\gamma < 1$. There is some constant $\beta < 1$ s.t. for every $m \geq n^{\beta}$ there is an explicit $(n, m, polylog(n) \cdot log(\frac{1}{\epsilon}), \Omega(m), poly(n) \cdot \epsilon)$ extractor.*

Now that we know how to extract all the randomness from sources having $\Omega(n^{\beta})$ min-entropy with only $polylog(n)$ truly random bits, by lemmas 3.5 and Theorem 3 we have good somewhere random mergers, for every m. Thus by lemma 4.3 we have good extractors for every m. We prove in detail in Appendix A that:

Theorem: *For every constant $\gamma < 1$, $\epsilon \geq 2^{-n^{\gamma}}$, and every $m = m(n)$ there is an explicit $(n, m, polylog(n) \cdot log(\frac{1}{\epsilon}), m, \epsilon)$–extractor.*

5 Explicit Somewhere Random Mergers

In this section we build explicit somewhere random mergers. We observe that a 2–block merger can be obtained from the previously designed extractors of [NZ93, SZ94]. Once such a merger is obtained, any number of blocks can be merged in a binary-tree fashion.

5.1 A 2-block somewhere random merger

A d–block (m, ϵ, η)–somewhere random source X, can be viewed intuitively as a source composed of d strings of length m, with a selector that for all but an η fraction of the inputs, can find a block that is ϵ quasi-random. Thus, it is natural to suspect that $\bar{X}$ is $\epsilon + \eta$ close to a distribution with m min-entropy. The following lemma (proved in appendix B.1) states this precisely:

Lemma 5.1 *(1) Any (m, ϵ, η) somewhere random source X is $\epsilon + \eta$ close to an $(m, 0, 0)$–somewhere random source X'. (2) For any $(m, 0, 0)$ somewhere random source X, $H_{\infty}(\bar{X}) \geq m$.*

Since a $(2m, m, t, m', \epsilon)$-extractor E extracts randomness from any source X with $H_\infty(\overline{X}) \geq m$, in particular it extracts randomness from any $(m, 0, 0)$ somewhere random source, and therefore by definition E is a $(2, m, t, m', \epsilon)$-somewhere random merger. Thus:

Corollary 5.2 *Any $(2m, m, t, m', \epsilon)$-extractor is a $(2, m, t, m', \epsilon)$-somewhere random merger.*

5.2 A d–block somewhere random merger

Given a d–block somewhere random source, we merge the blocks in pairs in a tree like fashion, resulting in a single block. We show that after each level of merges we still have a somewhere random source, and thus the resulting single block is necessarily quasi-random.

ALGORITHM 5.1 *Assume we can build a $(2, m, t(m), m - k(m), \epsilon(m))$-somewhere random merger E, for some monotone functions t and k. We build $M_l : \{0,1\}^{2^l m} \times \{0,1\}^{l\ t(m)} \mapsto \{0,1\}^{m - l \cdot k(m)}$, by induction on l:*

Input *: $x^l = x^l_1 \circ \ldots x^l_{2^l}$, where each $x^l_i \in \{0,1\}^m$.*

Output *: Toss $t = t_1 \circ \ldots t_l$, where $t_j \in \{0,1\}^{t(m)}$. If $l = 0$ output x^l, otherwise:*

- *Let $x^{l-1}_i = E(x^l_{2i-1} \circ x^l_{2i}\ ,\ t_l)$, for $i = 1, \ldots, 2^{l-1}$.*
- *Let the output be $M_{l-1}(x^{l-1}_1 \circ \ldots x^{l-1}_{2^{l-1}}\ ,\ t_1 \circ \ldots t_{l-1})$.*

Theorem: *[Same as Theorem 3] M_l is a $(2^l, m, l \cdot t(m), m - l \cdot k(m), l \cdot \epsilon(m))$ somewhere random merger.*

Proof: For $j = l, \ldots, 0$ denote by Z^j the random variable whose value is $x^j = x^j_1 \circ \ldots x^j_{2^j}$, where the input x is chosen according to $\overline{X}$, and t is uniform. Notice that $\overline{Z^l}$ is the distribution $\overline{X}$, and $\overline{Z^0}$ is the distribution of the output.

The theorem follows immediately from the following claim:

Claim 5.1 *Denote $m_j = m - (l - j)k(m)$. If X is an $(m_l, 0, 0)$ somewhere random source, then for any $1 \leq i \leq 2^j$, $d(\ (Z^j_i \mid Y \in [2^{l-j}(i-1) + 1, 2^{l-j} i])\ ,\ U_{m_j}\ ) \leq (l - j) \cdot \epsilon(m)$*

□

The proof of the claim is by downward induction on j. The basis $j = l$ simply says that for any i , $d((X_i \mid Y = i)\ ,\ U_m\ ) = 0$, which is exactly the hypothesis. Suppose it is true for j. Informally, the induction hypothesis says that if Y "points" to Z^j_{2i-1} then Z^j_{2i-1} is uniform, and if Y "points" to Z^j_{2i} then Z^j_{2i} is uniform. Thus, it is natural to suspect that if Y "points" to Z^j_{2i} or to Z^j_{2i}, then $Z^j_{2i-1} \circ Z^j_{2i}$ is a 2–block somewhere random source. Indeed, we show that $Z^j_{2i-1} \circ Z^j_{2i}$ with the right conditioning on Y is $(l - j) \cdot \epsilon(m)$ close to some $\overline{W}$ with $H_\infty(\overline{W}) \geq m_j$. Since $Z^{j-1}_i = E(Z^j_{2i-1} \circ Z^j_{2i}\ ,\ t_j)$ and E is an extractor the claim follows. A full proof appears in Appendix B.

REMARK 5.1 *Notice that we use the same random string t_j for all merges occuring in the j'th layer, and that this is possible because in a somewhere random source we do not care about dependencies between different blocks. Also notice that the error is additive in the depth of the tree of merges (i.e. in l), rather than in the size of the tree (2^l).*

6 Composing Extractors

In subsection 6.1 we show how to efficiently compose two extractors, and in subsection 6.2 we compose many extractors.

6.1 Composing Two Extractors

The algorithm computing the composed extractor was presented before as algorithm 4.1, along with Theorem 4, that we prove here.

Proof: [Of Theorem 4]

Obviously, it is enough to show that $E_1 \ominus E_2$ is an $(t_3, \zeta_1 + \zeta_2, 8n2^{-s/3})$-somewhere random source. To prove this, assume $H_\infty(\overline{X}) \geq m_1 + m_2 + s$. Denote by Q_i and Z_i the random variables with values q_i and z_i respectively. Also, let $\epsilon_3 = 2^{-s/3}$, $\epsilon_2 = 2\epsilon_3$, and $\epsilon_1 = 2\epsilon_2$. We define a selector for $Z = Z_1 \circ \ldots \circ Z_n = E_1 \ominus E_2$ in two phases: first we define a function f which is almost the selector but has few "bad" values, then we correct f to obtain the selector Y.

DEFINITION 6.1 *Define $f(w)$ to be the last i s.t $Prob(X_{[i,n]} = w_{[i,n]} \mid X_{[1,i-1]} = w_{[1,i-1]}) \leq (\epsilon_2 - \epsilon_3) \cdot 2^{-m_1}$.*

DEFINITION 6.2 *Define w to be "bad" if $f(w) = i$ and:*

1. *$Prob_{x \in X}(f(x) = i) \leq \epsilon_1$, or,*
2. *$Prob_{x \in X}(f(x) = i \mid x_{[1,i-1]} = w_{[1,i-1]}) \leq \epsilon_2$, or,*
3. *$Prob_{x \in X}(X_i = w_i \mid x_{[1,i-1]} = w_{[1,i-1]}) \leq \epsilon_3$*

We denote by B the set of all bad w. We denote by B_i ($i = 1, 2, 3$) the set of all w satisfying condition (i).

DEFINITION 6.3 *Let Y be the random variable, obtained by taking the input a and letting $Y = Y(a)$ where $Y(w) = 0$ if w is bad, and $f(w)$ otherwise.*

It holds that $Prob(w \text{ is bad}) \leq n(\epsilon_1 + \epsilon_2 + \epsilon_3) \leq 8n \cdot 2^{-s/3}$ (the proof is easy, see appendix C.2). We complete the proof by showing that $(Z_i \mid Y = i)$ is $\zeta_1 + \zeta_2$-close to uniform.

Claim 6.1 *If $Prob(Y = i \mid X_{[1,i-1]} = w_{[1,i-1]}) > 0$ then $H_\infty(X_{[i,n]} \mid Y = i$ and $X_{[1,i-1]} = w_{[1,i-1]}) \geq m_1$*

Therefore, for any such $w_{[1,i-1]}$, $(Q_i \mid Y = i$ *and* $X_{[1,i-1]} = w_{[1,i-1]})$ is ζ_1-close to random (since E_1 is an extractor). Hence by lemma 3.3, the distribution $(X_{[1,i-1]} \mid Y = i) \times (Q_i \mid Y = i$ *and* $X_{[1,i-1]} = w_{[1,i-1]})$ is ζ_1-close to the distribution $(X_{[1,i-1]} \mid Y = i) \times U$. But,

Claim 6.2 $H_\infty(X_{[1,i-1]} \mid Y = i) \geq m_2$.

Therefore, using the extractor E_2 we get that $(Z_i \mid Y = i)$ is $\zeta_1 + \zeta_2$-close to uniform.

□

The proofs of claims 6.1 and 6.2 appear in appendix C.

6.2 Composing Many Extractors

In this subsection we prove Theorem 5, which claims that $E_k \overset{M}{\odot} \ldots \overset{M}{\odot} E_1$ can be efficiently calculated:

Proof: [of Theorem 5]

Correctness :

By induction on k. For $k = 2$ this follows from theorem 4. For larger k's this is a straight forward combination of the induction hypothesis and Theorem 4.

Running time :

We compute $E_k \overset{M_{k-1}}{\odot} E_{k-1} \overset{M_{k-2}}{\odot} \ldots E_2 \overset{M_1}{\odot} E_1$ using a dynamic programming procedure:

- Given $x \in \overline{X}$, we toss $y \in \{0,1\}^{t_1}$ and $y_j \in \{0,1\}^{\mu_j}$ for $j = 1, \ldots, k$.
- Next, we compute the matrix M where $M[j,i] = (E_j \overset{M_{j-1}}{\odot} E_{j-1} \overset{M_{j-2}}{\odot} \ldots E_2 \overset{M_1}{\odot} E_1)(x_{[i,n]}, y \circ y_1 \circ \ldots \circ y_j)$, for $1 \leq i \leq n$ and $1 \leq j \leq k$.

  The entries of the first row of M, $M[1,i]$ can be filled by calculating $E_1(x_{[i,n]}, y)$. Suppose we know how to fill the j'th row of M. We show how to fill the $j+1$'th row.

  - Denote $q_l = M[j,l]$ for $l = i, \ldots, n$, and let $z_l = E_{j+1}(x_{[i,l-1]}, q_l)$.
  - Let $M[j+1,i] = M_j(z_i \circ \ldots z_n, y_j)$.

  By the composition definition $M[j,i]$ has the correct value, and clearly, the computation takes polynomial time in n.

□

7 Acknowledgments

I would like to thank my advisor Noam Nisan for his collaboration on this research. Thanks to David Zuckerman for his hospitality when I visited him last year. I would like to thank David Zuckerman, Avi Wigderson, Michael Saks, Arvind Srinivasan and Shiyu Zhou for interesting discussions. Many special thanks go to Roy Armoni for our many interesting talks on the subject.

References

[AKSS89] M. Ajtai, J. Komlos, W. Steiger, and E. Szemeredi. Almost sorting in one round. In *Advances in Computer Research*, volume 5, pages 117–125, 1989.

[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.

[GW94] O. Goldreich and A. Wigderson. Tiny families of functions with random properties: A quality-size trade-off for hashing. In *Proceedings of the 26th Annual ACM Symposium on the Theory of Computing*, ACM, pages 574–583, 1994.

[NZ93] N. Nisan and D. Zuckerman. More deterministic simulation in logspace. In *Proceedings of the 25th Annual ACM Symposium on the Theory of Computing*, ACM, pages 235–244, 1993.

[Pip87] N. Pippenger. Sorting and selecting in rounds. *SIAM Journal on Computing*, 16:1032–1038, 1987.

[SSZ95] M. Saks, A. Srinivasan, and S. Zhou. Explicit dispersers with polylog degree. In *Proceedings of the 26th Annual ACM Symposium on the Theory of Computing*, ACM, 1995.

[SZ94] A. Srinivasan and D. Zuckerman. Computing with very weak random sources. In *Proceedings of the 35th Annual IEEE Symposium on the Foundations of Computer Science*, 1994.

[WZ93] A. Wigderson and D. Zuckerman. Expanders that beat the eigenvalue bound: Explicit construction and applications. In *Proceedings of the 25th Annual ACM Symposium on the Theory of Computing*, ACM, pages 245–251, 1993.

[Zuc] D. Zuckerman. Randomness-optimal sampling, extractors, and constructive leader election. Private Communication.

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

[Zuc91] D. Zuckerman. Simulating BPP using a general weak random source. In *Proceedings of the 32nd Annual IEEE Symposium on the Foundations of Computer Science*, pages 79–89, 1991.

[Zuc93] D. Zuckerman. NP-complete problems have a version that's hard to approximate. In *Proceedings of the 8th Structures in Complexity Theory*, IEEE, pages 305–312, 1993.

A Main Theorem

A.1 Proof of lemma 4.1

- By lemma 3.2 there is an explicit $(m, \frac{m}{f(m)}, O(log^2 m \cdot log(\frac{1}{\epsilon})), \frac{m}{f^2(m)}, \epsilon)$– extractor.
- By lemma 3.4 there is an explicit $(2m, m, O(f^2(m) \cdot log^2 m \cdot log(\frac{1}{\epsilon})), m - \frac{m}{f(m)}, poly(m) \cdot \epsilon)$– extractor.
- By Theorem 3 there is an explicit $(n, m, O(log(n) \cdot polylog(m) \cdot f^2(m) \cdot log(\frac{1}{\epsilon}), m - log(n) \cdot \frac{m}{f(m)}, log(n) \cdot poly(m) \cdot \epsilon)$– somewhere random merger. For any $m \geq m_0$ $\frac{m}{f(m)} \leq \frac{m}{b\, log(n)}$, hence $log(n) \cdot \frac{m}{f(m)} \leq \frac{m}{b}$.

A.2 Proof of lemma 4.3

Proof: The lemma follows easily from lemma A.1 using lemma 3.5. □

Lemma A.1 *Suppose for any $\bar{m} \leq m \leq \bar{\bar{m}}$ there is an explicit $(n, m, t, m - \frac{m}{\bar{c}}, \epsilon)$ somewhere random merger, where $\bar{c}$ is the constant in lemma 4.3. Then, for any $\bar{m} \leq m \leq \bar{\bar{m}}$ there is an explicit $(n, m, O(\bar{m} \cdot log(\frac{1}{\epsilon}) + log(n) \cdot t), \Omega(m), poly(n) \cdot \epsilon)$, extractor E.*

Proof: Let $b = c_{sz} \cdot (1 - \frac{1}{\bar{c}})$. Clearly b is a constant, and $1 < b < c_{sz}$. Define $m_i = b^i \cdot \bar{m} \cdot log(\frac{1}{\epsilon})$, and let l be the first integer s.t. $\Sigma_{i=1}^{l} 2m_i \leq \frac{m}{2}$.

Define $E = E_l \overset{M_{l-1}}{\odot} E_{l-1} \ldots \overset{M_1}{\odot} E_1$, where:

- E_i is the $(n, 2m_i, m_i, c_{sz} \cdot m_i, 2^{-m_i/2})$– extractor A_{m_i} from lemma 3.1
- M_i is the $(n, c_{sz} \cdot m_{i+1}, t, b \cdot m_{i+1}, \epsilon)$–somewhere random merger promised in the hypothesis of the lemma.

Now we use theorem 5 with $t_i = m_i$ and $s_i = (c_{sz} - b)m_{i-1}$, and we also take $s = \frac{m}{2l}$. By Theorem 5, E is an $(n, \Sigma_{i=1}^{l} 2m_i + (l-1)s, t_1 + \Sigma_{i=1}^{l-1} t, t_{l+1}, \Sigma_{i=1}^{l} 2^{-m_0/2} + \Sigma_{i=1}^{l-1} \epsilon + (l-1)n2^{-s/3+3})$ –extractor. Since $l = O(log(n))$ and $\epsilon \geq 2^{-s/3}$ (otherwise the result is trivial), E is an extractor as required. Since A_i, M_i are explicit, so is E. □

A.3 Proof of lemma 4.5

Proof: Choose $\delta = \frac{1-\gamma}{2}$ and $\beta = 1 - \frac{\delta}{2}$. Let the extractor E be $E = B_m \overset{M}{\odot} E_{sz}$ where

- E_{sz} is the $(n, n^{\beta}, O(log^2 n \cdot log(\frac{1}{\epsilon})), n^{2\beta-1}, \epsilon)$ – extractor of lemma 3.2.
- B_m is the extractor from corollary 4.4.
- M is the merger from corollary 4.2.

Since $n^{2\beta-1} = n^{\delta} \cdot n^{\gamma} = \Omega(2^{\sqrt{log(n)}} \cdot log(\frac{1}{\epsilon}))$, $E = B_m \overset{M}{\odot} E_{sz}$ is well-defined. By theorem 4, for every m, E is an explicit $(n, m + n^{\beta} + n^{\gamma}, polylog(n) \cdot log(\frac{1}{\epsilon}), \Omega(m), poly(n) \cdot \epsilon)$–extractor. In particular if $H_{\infty}(X) = \Omega(n^{\beta})$ we extract $\Omega(H_{\infty}(X))$ as required. □

A.4 Proof of theorem 4.5

- By lemma 3.5, lemma 4.5 implies an explicit $(2n, n, polylog(n) \cdot log(\frac{1}{\epsilon}), n - n^{\beta}, poly(n) \cdot \epsilon)$ extractor.
- There is some constant d (that depends only on γ) s.t. for every $log^d n \leq m \leq n$, $log(n) \cdot m^{\beta} \leq \frac{m}{\bar{c}}$. Therefore by theorem 3, for every m there is an explicit $(n, m, polylog(n) \cdot log(\frac{1}{\epsilon}), m - \frac{m}{\bar{c}}, poly(n) \cdot \epsilon)$ somewhere random merger.
- By lemma 4.3, this implies an explicit $(n, m, polylog(n) \cdot log(\frac{1}{\epsilon}), m, poly(n) \cdot \epsilon)$– extractor, for any m. Plugging $\epsilon' = \frac{\epsilon}{poly(n)}$, gives the theorem.

B Somewhere Random Mergers

B.1 A Somewhere Random source has large min-entropy

Lemma B.1 *If $X = X_1 \circ \ldots \circ X_d$ is an (m, ϵ, η) somewhere random source, then X is η–close to an $(m, \epsilon, 0)$ somewhere random source X'.*

Proof: [of lemma B.1]

Let Y be an (m, ϵ, η) selector for X. Denote $p = Prob(Y = 0) \leq \eta$. Define the distribution $\overline{D}$ by:

$$\overline{D}(i, x) = \begin{cases} 0 & \text{If } i = 0 \\ \frac{Prob(\ (Y,X)=(i,x)\ )}{1-p} & \text{otherwise} \end{cases}$$

It is easy to see that $\overline{D}$ is a distribution. Define the random variable $Y' \circ X'$ as the result of the picking $(i, x) \in \overline{D}$, i.e. $\overline{Y' \circ X'} = \overline{D}$. It is clear that $d(\overline{X}, \overline{X'}) \leq d(\overline{Y \circ X}, \overline{Y' \circ X'}) = p \leq \eta$.

Now we want to show that Y' is an $(m, \epsilon, 0)$ selector for X'. It is clear that $Prob(Y' = 0) = 0$. It is not hard to see that for any $i > 0$ we have: $Prob(X' = x \mid Y' = i) = Prob(X = x \mid Y = i)$.

Therefore, since we know that $(X_i | Y = i)$ is ϵ–close to U_m, we also know that $(X'_i | Y' = i)$ is ϵ close to U_m, thus completing the proof.

□

Lemma B.2 *Let $X = X_1 \circ \ldots \circ X_d$ be an $(m, \epsilon, 0)$–somewhere random source, then X is ϵ close to an $(m, 0, 0)$–somewhere random source Z.*

Proof: [of lemma B.2]

Let Y be an $(m, \epsilon, 0)$ selector for X. Fix some $i \in [1..d]$. We know that $d((X_i \mid Y = i), U_m) \leq \epsilon$. Define a distribution $Z^{(i)}$ by:

$Z^{(i)}(x) = \frac{1}{2^m} \cdot Prob(X = x \mid X_i = x_i \text{ and } Y = i)$ - if $Prob(X_i = x_i \text{ and } Y = i) > 0$, $Z^{(i)}(x) = \frac{1}{2^m}$- if $Prob(X_i = x_i \text{ and } Y = i) = 0$, and $Z^{(i)}(x) = 0$ otherwise.

It is easy to check that $Z^{(i)}$ is indeed a distribution, and that $Z_i^{(i)} = U_m$. Define $Y \circ Z$ to be the random variable obtained by choosing i according to Y, then choosing z according to $Z^{(i)}$, i.e., for all $i > 0$, $(Z \mid Y = i) = Z^{(i)}$. Also, denote $X^{(i)} = (X \mid Y = i)$. Then:

$$Prob(Z_i = z_i \mid Y = i) = Z_i^{(i)}(z_i) = 2^{-m}$$

We soon prove that:

Claim B.1 $d(X^{(i)}, Z^{(i)}) \leq \epsilon$.

Thus:

$$d(\overline{X}, \overline{Z}) \leq d(\overline{Y \circ X}, \overline{Y \circ Z}) =$$

$$\Sigma_{i>0}\ Pr(Y = i) \cdot d((X \mid Y = i), (Z \mid Y = i)) =$$

$$\Sigma_{i>0}\ Pr(Y = i) \cdot d(X^{(i)}, Z^{(i)}) \leq \epsilon$$

Hence Z satisfies the requirements of the lemma.

□

Proof: [of claim B.1]

We need to show that for any $A \subseteq \Lambda_X$, $|X^{(i)}(A) - Z^{(i)}(A)| \leq \epsilon$. It is sufficient to show this for the set A containing all $x \in \Lambda_X$ s.t. $X^{(i)}(x) > Z^{(i)}(x)$. This can be easily seen, using the fact that for any $x \in A$: $Pr(Z = x \mid Z_i = a_i \text{ and } Y = i) = \frac{Pr(Z=x \mid Y=i)}{Pr(Z_i=a_i \mid Y=i)} = Pr(X = x \mid X_i = a_i \text{ and } Y = i)$.

□

Lemma B.3 *Let $X = X_1 \circ \ldots \circ X_d$ be an $(m, 0, 0)$ somewhere random source, then $H_\infty(\overline{X}) \geq m$.*

Proof: Suppose Y is an $(m, 0, 0)$ selector for X.

$$Prob(X = x) =$$

$$\Sigma_{i\in[1..d]}\ Prob(Y = i) \cdot Prob(X_i = x_i \mid Y = i) \leq$$

$$\Sigma_{i\in[1..d]}\ Prob(Y = i) \cdot 2^{-m} = 2^{-m}$$

□

Combining lemmas B.1, B.2 and lemma B.3 we get lemma 5.1.

B.2 Proof of claim 5.1

Proof:

The proof is by downward induction on j. The basis $j = l$ simply says that for any i, $d((X_i \mid Y = i), U_m) = 0$, which is exactly the hypothesis. Suppose it is true for j, we prove it for $j - 1$. By the induction hypothesis:

- $d(\ (Z^j_{2i-1} \mid Y \in [2^{l-j}(2i-2)+1, 2^{l-j}(2i-1)])\ ,\ U_{m_j}\ ) \leq (l-j)\epsilon(m)$
- $d(\ (Z^j_{2i} \mid Y \in [2^{l-j}(2i-1)+1, 2^{l-j}2i])\ ,\ U_{m_j}\ ) \leq (l-j)\cdot\epsilon(m)$

Soon we prove the following lemma:

Lemma B.4 *Let A, B and Y be any random variables. Suppose that $d((A \mid Y \in S_1), U_m) \leq \epsilon$ and $d((B \mid Y \in S_2), U_m) \leq \epsilon$ for some disjoint sets S_1 and S_2. Then $(A \circ B \mid Y \in S_1 \cup S_2)$ is ϵ-close to some $\overline{X}$ with $H_\infty(\overline{X}) \geq m$.*

Therefore:

- $(Z^j_{2i-1} \circ Z^j_{2i} \mid Y \in [2^{l-j+1}(i-1)+1, 2^{l-j+1}i])$ is $(l-j)\cdot\epsilon(m)$ close to some $\overline{W}$ with $H_\infty(\overline{W}) \geq m_j$.
- Since $Z_i^{j-1} = E(Z^j_{2i-1} \circ Z^j_{2i}\ ,\ t_j)$, it follows that $(Z_i^{j-1} \mid Y \in [2^{l-j+1}(i-1)+1, 2^{l-j+1}i])$ is $(l-j)\cdot\epsilon(m)$ close to $E_{m_j}(x, t_k)$ where $x \in \overline{X}$ and $H_\infty(\overline{X}) \geq m_j$. Therefore, it is $(l-j)\cdot\epsilon(m) + \epsilon(m)$ close to random, as required.

□

B.2.1 Proof of lemma B.4

Proof: We define random variables $Y', A' \circ B'$ as follows:

- Pick $Y' = i \in S_1 \cup S_2$ with: $Pr(Y' = i) = Pr(Y = i \mid Y \in S_1 \cup S_2)$.
- Choose $a' \circ b' \in (A \circ B \mid Y = i)$.

it is easy to prove that:

Claim: $Pr(A' = a' \mid Y' = i) = Pr(A = a' \mid Y = i)$ and $Pr(B' = b' \mid Y' = i) = Pr(B = b' \mid Y = i)$.

Define

$$Z' = \begin{cases} 1 & \text{If } Y' \in S_1 \\ 2 & \text{Otherwise, i.e. } Y' \in S_2 \end{cases}$$

It is not hard to see that:

Claim B.2 *$(A' \mid Z' = 1) = (A \mid Y \in S_1)$ and $(B' \mid Z' = 2) = (B \mid Y \in S_2)$.*

Hence, Z' is an $(m, \epsilon, 0)$ selector for $A' \circ B'$.

Therefore by lemma 5.1, $\overline{A' \circ B'}$ is ϵ-close to some $\overline{X}$ with $H_\infty(\overline{X}) \geq m$. However, it is not hard to see that:

Claim: $\overline{A' \circ B'} = (A \circ B \mid Y \in S_1 \cup S_2)$.

Thus, $(A \circ B \mid Y \in S_1 \cup S_2) = \overline{A' \circ B'}$ is ϵ-close to some $\overline{X}$ with $H_\infty(\overline{X}) \geq m$, completing the proof. □

C Composing Extractors

C.1 Composing Two Extractors

Proof: [of claim 6.1]

For any w s.t. $Y(w) = i$:

$$\begin{array}{ll} Prob(X_{[i,n]} = w_{[i,n]} | X_{[1,i-1]} = w_{[1,i-1]}, Y(x) = i) & \leq \\ \frac{Prob(\ X_{[i,n]} = w_{[i,n]} \ \mid \ X_{[1,i-1]} = w_{[1,i-1]}\ )}{Prob(Y(x) = i \ \mid \ X_{[1,i-1]} = w_{[1,i-1]}\ )} & \leq \\ \frac{(\epsilon_2 - \epsilon_3) \cdot 2^{-m_1}}{Prob(Y(x) = i \ \mid \ X_{[1,i-1]} = w_{[1,i-1]}\ )} & \leq \\ \frac{(\epsilon_2 - \epsilon_3)\ 2^{-m_1}}{\epsilon_2 - \epsilon_3} = 2^{-m_1} & \end{array}$$

The first line is true since $Prob(A \mid B) \leq \frac{Prob(A)}{Prob(B)}$, the second line since $f(w) = i$, and the third follows Claim C.1. □

Proof: [of claim 6.2]

Take any $w_{[1,i-1]}$ that can be extended to some w with $Y(w) = i$.

$$Prob(X_{[1,i-1]} = w_{[1,i-1]}) = $$
$$\frac{Prob(X_{[1,n]} = w_{[1,n]})}{Prob(X_{[i,n]} = w_{[i,n]} \mid X_{[1,i-1]} = w_{[1,i-1]})} = $$
$$\frac{Prob(X_{[1,n]} = w_{[1,n]})}{Prob(X_i = w_i | X_{[1,i-1]}) Prob(X_{[i+1,n]} = w_{[i+1,n]} | X_{[1,i]})}$$

However,

- $Prob(X_{[i+1,n]} = w_{[i+1,n]} | X_{[1,i]}) \geq (\epsilon_2 - \epsilon_3) 2^{-m_1}$
- $Prob(X_i = w_i \mid X_{[1,i-1]} = w_{[1,i-1]}) \geq \epsilon_3$
- $Prob(X_{[1,n]} = w_{[1,n]}) \leq 2^{-(m_1 + m_2 + s)}$

The first line is true since $f(w) = i$, the second since $w \notin B_3$, the third since $H_\infty(X) \geq m_1 + m_2 + s$. Thus,

$$Prob(X_{[1,i-1]} = w_{[1,i-1]}) \leq \frac{2^{-m_2 - s}}{\epsilon_3 \cdot (\epsilon_2 - \epsilon_3)} \quad (1)$$

Therefore,

$$\begin{array}{ll} Prob(X_{[1,i-1]} = w_{[1,i-1]} \mid Y(x) = i) & \leq \\ \frac{Prob(\ X_{[1,i-1]} = w_{[1,i-1]})}{Prob(Y(x) = i)} & \leq \\ \frac{2^{-m_2 - s}}{\epsilon_3 \cdot (\epsilon_2 - \epsilon_3)\ Prob(Y(x) = i)} & \leq \\ \frac{2^{-m_2 - s}}{\epsilon_3 \ (\epsilon_2 - \epsilon_3)(\epsilon_1 - \epsilon_2 - \epsilon_3)} = 2^{-m_2} & \end{array}$$

The first line is true since $Prob(A \mid B) \leq \frac{Prob(A)}{Prob(B)}$. The second by Eq. (1). The third by Claim (C.2). □

C.2 Technical Lemmas

In this subsection we state some easy technical lemmas used in subsection 6.1.

Claim C.1 *For any i and $w_{[1,i-1]}$, if $Prob_{x \in X}(Y(x) = i \mid x_{[1,i-1]} = w_{[1,i-1]}) > 0$, Then $Prob_{x \in X}(Y(x) = i \mid x_{[1,i-1]} = w_{[1,i-1]}) \geq \epsilon_2 - \epsilon_3$.*

Claim C.2 *For any i, if $Prob_{x \in X}(Y(x) = i) > 0$, then $Prob_{x \in X}(Y(x) = i) \geq \epsilon_1 - \epsilon_2 - \epsilon_3$.*

Claim C.3

1. *$Prob(x \in B_i) \leq n\epsilon_i$, for $i = 1, 2, 3$.*
2. *For any i: $Prob(f(x) = i$ and $x \in B_2) \leq \epsilon_2$*
3. *For any i and $w_{[1,i-1]}$: $Prob(f(x) = i$ and $x \in B_3 \mid x_{[1,i-1]} = w_{[1,i-1]}) \leq \epsilon_3$*
4. *For any i: $Prob(f(x) = i$ and $x \in B_3) \leq \epsilon_3$*