Computational Complexity:

A Conceptual Perspective

Oded Goldreich

Department of Computer Science and Applied Mathematics
Weizmann Institute of Science, Rehovot, Israel.

February 16, 2008

Appendix D

Probabilistic Preliminaries and Advanced Topics in Randomization

What is this? Chicken Curry and Seafood Salad?
Fine, but in the same plate? This is disgusting!

Johan Håstad at Grendel's, Cambridge (1985)

Summary: This appendix lumps together some preliminaries regarding probability theory and some advanced topics related to the role and use of randomness in computation. Needless to say, each of these topics appears in a separate section.

The probabilistic preliminaries include our conventions regarding random variables, which are used throughout the book. Also included are overviews of three useful probabilistic inequalities: Markov's Inequality, Chebyshev's Inequality, and Chernoff Bound.

The advanced topics include hashing, sampling, and randomness extraction. For hashing, we describe constructions of pairwise (and t-wise independent) hashing functions and (a few variants of) the Leftover Hashing Lemma (used a few times in the main text). We then review the "complexity of sampling": that is, the number of samples and the randomness complexity involved in estimating the average value of an arbitrary function defined over a huge domain. Finally, we provide an overview on the question of extracting almost perfect randomness from sources of weak (or defected) randomness.

D.1 Probabilistic preliminaries

Probability plays a central role in complexity theory (see, for example, Chapters 6–9). We assume that the reader is familiar with the basic notions of probability theory. In this section, we merely present the probabilistic notations that are used throughout the book and three useful probabilistic inequalities.

D.1.1 Notational Conventions

Throughout the entire book we refer only to *discrete* probability distributions. Specifically, the underlying probability space consists of the set of all strings of a certain length ℓ, taken with uniform probability distribution. That is, the sample space is the set of all ℓ-bit long strings, and each such string is assigned probability measure $2^{-\ell}$. Traditionally, *random variables* are defined as functions from the sample space to the reals. Abusing the traditional terminology, we use the term random variable also when referring to functions mapping the sample space into the set of binary strings. We often do not specify the probability space, but rather talk directly about random variables. For example, we may say that X is a random variable assigned values in the set of all strings such that $\Pr[X = 00] = \frac{1}{4}$ and $\Pr[X = 111] = \frac{3}{4}$. (Such a random variable may be defined over the sample space $\{0,1\}^2$, so that $X(11) = 00$ and $X(00) = X(01) = X(10) = 111$.) One important case of a random variable is the output of a randomized process (e.g., a probabilistic polynomial-time algorithm, as in Section 6.1).

All our probabilistic statements refer to random variables that are defined beforehand. Typically, we may write $\Pr[f(X) = 1]$, where X is a random variable defined beforehand (and f is a function). An important convention is that *all occurrences of the same symbol in a probabilistic statement refer to the same* (unique) *random variable*. Hence, if $B(\cdot,\cdot)$ is a Boolean expression depending on two variables, and X is a random variable then $\Pr[B(X, X)]$ denotes the probability that $B(x, x)$ holds when x is chosen with probability $\Pr[X = x]$. For example, for every random variable X, we have $\Pr[X = X] = 1$. We stress that if we wish to discuss the probability that $B(x, y)$ holds when x and y are chosen independently with identical probability distribution, then we will define *two* independent random variables each with the same probability distribution. Hence, if X and Y are two independent random variables then $\Pr[B(X, Y)]$ denotes the probability that $B(x, y)$ holds when the pair (x, y) is chosen with probability $\Pr[X = x] \cdot \Pr[Y = y]$. For example, for every two independent random variables, X and Y, we have $\Pr[X = Y] = 1$ only if both X and Y are trivial (i.e., assign the entire probability mass to a single string).

Throughout the entire book, U_n denotes a random variable uniformly distributed over the set of all strings of length n. Namely, $\Pr[U_n = \alpha]$ equals 2^{-n} if $\alpha \in \{0,1\}^n$ and equals 0 otherwise. We often refer to the distribution of U_n as the uniform distribution (neglecting to qualify that it is uniform over $\{0,1\}^n$). In addition, we occasionally use random variables (arbitrarily) distributed over $\{0,1\}^n$ or $\{0,1\}^{\ell(n)}$, for some function $\ell : \mathbb{N} \to \mathbb{N}$. Such random variables are typically denoted by X_n, Y_n, Z_n, etc. We stress that in some cases X_n is distributed over

$\{0,1\}^n$, whereas in other cases it is distributed over $\{0,1\}^{\ell(n)}$, for some function ℓ (which is typically a polynomial). We often talk about probability ensembles, which are infinite sequence of random variables $\{X_n\}_{n\in\mathbb{N}}$ such that each X_n ranges over strings of length bounded by a polynomial in n.

Statistical difference. The statistical distance (a.k.a variation distance) between the random variables X and Y is defined as

$$\frac{1}{2}\cdot\sum_v |\mathsf{Pr}[X=v]-\mathsf{Pr}[Y=v]| \;=\; \max_S\{\mathsf{Pr}[X\in S]-\mathsf{Pr}[Y\in S]\}. \tag{D.1}$$

We say that X is δ-close (resp., δ-far) to Y if the statistical distance between them is at most (resp., at least) δ.

D.1.2 Three Inequalities

The following probabilistic inequalities are very useful. These inequalities refer to random variables that are assigned real values and provide upper-bounds on the probability that the random variable deviates from its expectation.

D.1.2.1 Markov's Inequality

The most basic inequality is Markov's Inequality that applies to any random variable with bounded maximum or minimum value. For simplicity, this inequality is stated for random variables that are lower-bounded by zero, and reads as follows: *Let X be a non-negative random variable and v be a non-negative real number. Then*

$$\mathsf{Pr}\left[X\geq v\right]\leq\frac{\mathsf{E}(X)}{v} \tag{D.2}$$

Equivalently, $\mathsf{Pr}[X\geq r\cdot\mathsf{E}(X)]\leq\frac{1}{r}$. The proof amounts to the following sequence:

$$\begin{aligned}
\mathsf{E}(X) &= \sum_x \mathsf{Pr}[X=x]\cdot x\\
&\geq \sum_{x<v}\mathsf{Pr}[X=x]\cdot 0+\sum_{x\geq v}\mathsf{Pr}[X=x]\cdot v\\
&= \mathsf{Pr}[X\geq v]\cdot v
\end{aligned}$$

D.1.2.2 Chebyshev's Inequality

Using Markov's inequality, one gets a potentially stronger bound on the deviation of a random variable from its expectation. This bound, called Chebyshev's inequality, is useful when having additional information concerning the random variable (specifically, a good upper bound on its variance). For a random variable X of finite expectation, we denote by $\mathsf{Var}(X)\stackrel{\text{def}}{=}\mathsf{E}[(X-\mathsf{E}(X))^2]$ the variance of X, and observe that $\mathsf{Var}(X) = \mathsf{E}(X^2) - \mathsf{E}(X)^2$. Chebyshev's Inequality then reads as follows: *Let X be a random variable, and $\delta > 0$. Then*

$$\mathsf{Pr}\left[|X - \mathsf{E}(X)| \geq \delta\right] \leq \frac{\mathsf{Var}(X)}{\delta^2} \tag{D.3}$$

Proof: We define a random variable $Y \stackrel{\text{def}}{=} (X - \mathsf{E}(X))^2$, and apply Markov's inequality. We get

$$\begin{aligned} \mathsf{Pr}\left[|X - \mathsf{E}(X)| \geq \delta\right] &= \mathsf{Pr}\left[(X - \mathsf{E}(X))^2 \geq \delta^2\right] \\ &\leq \frac{\mathsf{E}[(X - \mathsf{E}(X))^2]}{\delta^2} \end{aligned}$$

and the claim follows. ■

Corollary (Pairwise Independent Sampling)**:** Chebyshev's inequality is particularly useful in the analysis of the error probability of approximation via repeated sampling. It suffices to assume that the samples are picked in a pairwise independent manner, where $X_1, X_2, ..., X_n$ are pairwise independent if for every $i \neq j$ and every α, β it holds that $\mathsf{Pr}[X_i = \alpha \wedge X_j = \beta] = \mathsf{Pr}[X_i = \alpha] \cdot \mathsf{Pr}[X_j = \beta]$. The corollary reads as follows: *Let $X_1, X_2, ..., X_n$ be pairwise independent random variables with identical expectation, denoted μ, and identical variance, denoted σ^2. Then, for every $\varepsilon > 0$, it holds that*

$$\mathsf{Pr}\left[\left|\frac{\sum_{i=1}^n X_i}{n} - \mu\right| \geq \varepsilon\right] \leq \frac{\sigma^2}{\varepsilon^2 n} \tag{D.4}$$

Proof: Define the random variables $\overline{X}_i \stackrel{\text{def}}{=} X_i - \mathsf{E}(X_i)$. Note that the $\overline{X}_i$'s are pairwise independent, and each has zero expectation. Applying Chebyshev's inequality to the random variable $\sum_{i=1}^n \frac{X_i}{n}$, and using the linearity of the expectation operator, we get

$$\begin{aligned} \mathsf{Pr}\left[\left|\sum_{i=1}^n \frac{X_i}{n} - \mu\right| \geq \varepsilon\right] &\leq \frac{\mathsf{Var}\left[\sum_{i=1}^n \frac{X_i}{n}\right]}{\varepsilon^2} \\ &= \frac{\mathsf{E}\left[\left(\sum_{i=1}^n \overline{X}_i\right)^2\right]}{\varepsilon^2 \cdot n^2} \end{aligned}$$

Now (again using the linearity of expectation)

$$\mathsf{E}\left[\left(\sum_{i=1}^n \overline{X}_i\right)^2\right] = \sum_{i=1}^n \mathsf{E}\left[\overline{X}_i^2\right] + \sum_{1 \leq i \neq j \leq n} \mathsf{E}\left[\overline{X}_i \overline{X}_j\right]$$

By the pairwise independence of the $\overline{X}_i$'s, we get $\mathsf{E}[\overline{X}_i \overline{X}_j] = \mathsf{E}[\overline{X}_i] \cdot \mathsf{E}[\overline{X}_j]$, and using $\mathsf{E}[\overline{X}_i] = 0$, we get

$$\mathsf{E}\left[\left(\sum_{i=1}^n \overline{X}_i\right)^2\right] = n \cdot \sigma^2$$

The corollary follows. ■

D.1.2.3 Chernoff Bound

When using pairwise independent sample points, the error probability in the approximation decreases linearly with the number of sample points (see Eq. (D.4)). When using totally independent sample points, the error probability in the approximation can be shown to decrease exponentially with the number of sample points. (Recall that the random variables $X_1, X_2, ..., X_n$ are said to be totally independent if for every sequence $a_1, a_2, ..., a_n$ it holds that $\mathsf{Pr}[\wedge_{i=1}^n X_i = a_i] = \prod_{i=1}^n \mathsf{Pr}[X_i = a_i]$.) Probability bounds supporting the foregoing statement are given next. The first bound, commonly referred to as Chernoff Bound, concerns 0-1 random variables (i.e., random variables that are assigned as values either 0 or 1), and asserts the following. *Let $p \le \frac{1}{2}$, and $X_1, X_2, ..., X_n$ be independent 0-1 random variables such that $\mathsf{Pr}[X_i = 1] = p$, for each i. Then, for every $\varepsilon \in (0, p]$, it holds that*

$$\mathsf{Pr}\left[\left|\frac{\sum_{i=1}^n X_i}{n} - p\right| > \varepsilon\right] \; < \; 2 \cdot e^{-c \cdot \varepsilon^2 \cdot n} \text{, where } c = \max(2, \tfrac{1}{3p}). \tag{D.5}$$

The more common formulation sets $c = 2$, but the case $c = 1/3p$ is very useful when p is small and one cares about a *multiplicative* deviation (e.g., $\varepsilon = p/2$).

Proof Sketch: We upper-bound $\mathsf{Pr}[\sum_{i=1}^n X_i - pn > \varepsilon n]$, and $\mathsf{Pr}[pn - \sum_{i=1}^n X_i > \varepsilon n]$ is bounded similarly. Letting $\overline{X}_i \stackrel{\text{def}}{=} X_i - \mathsf{E}(X_i)$, we apply Markov's inequality to the random variable $e^{\lambda \sum_{i=1}^n \overline{X}_i}$, where $\lambda \in (0, 1]$ will be determined to optimize the expressions that we derive. Thus, $\mathsf{Pr}[\sum_{i=1}^n \overline{X}_i > \varepsilon n]$ is upper-bounded by

$$\frac{\mathsf{E}[e^{\lambda \sum_{i=1}^n \overline{X}_i}]}{e^{\lambda \varepsilon n}} = e^{-\lambda \varepsilon n} \cdot \prod_{i=1}^n \mathsf{E}[e^{\lambda \overline{X}_i}]$$

where the equality is due to the independence of the random variables. To simplify the rest of the proof, we establish a sub-optimal bound as follows. Using a Taylor expansion of e^x (e.g., $e^x < 1 + x + x^2$ for $|x| \le 1$) and observing that $\mathsf{E}[\overline{X}_i] = 0$, we get $\mathsf{E}[e^{\lambda \overline{X}_i}] < 1 + \lambda^2 \mathsf{E}[\overline{X}_i^2]$, which equals $1 + \lambda^2 p(1-p)$. Thus, $\mathsf{Pr}[\sum_{i=1}^n X_i - pn > \varepsilon n]$ is upper-bounded by $e^{-\lambda \varepsilon n} \cdot (1 + \lambda^2 p(1-p))^n < \exp(-\lambda \varepsilon n + \lambda^2 p(1-p)n)$, which is optimized at $\lambda = \varepsilon/(2p(1-p))$ yielding $\exp(-\frac{\varepsilon^2}{4p(1-p)} \cdot n) \le \exp(-\varepsilon^2 \cdot n)$. □

The foregoing proof strategy can be applied in more general settings.[1] A more general bound, which refers to independent random variables that are each bounded but are not necessarily identical, is given next (and is commonly referred to as Hoefding Inequality). *Let $X_1, X_2, ..., X_n$ be n independent random variables, each ranging in the (real) interval $[a, b]$, and let $\mu \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^n \mathsf{E}(X_i)$ denote the average*

[1] For example, verify that the current proof actually applies to the case that $X_i \in [0, 1]$ rather than $X_i \in \{0, 1\}$, by noting that $\mathsf{Var}[X_i] \le p(1-p)$ still holds.

expected value of these variables. Then, for every $\varepsilon > 0$,

$$\Pr\left[\left|\frac{\sum_{i=1}^{n} X_i}{n} - \mu\right| > \varepsilon\right] < 2 \cdot e^{-\frac{2\varepsilon^2}{(b-a)^2} \cdot n} \tag{D.6}$$

The special case (of Eq. (D.6)) that refers to identically distributed random variables is easy to derive from the foregoing Chernoff Bound (by recalling Footnote 1 and using a linear mapping of the interval $[a, b]$ to the interval $[0, 1]$). This special case is useful in estimating the average value of a (bounded) function defined over a large domain, especially when the desired error probability needs to be negligible (i.e., decrease faster than any polynomial in the number of samples). Such an estimate can be obtained provided that we can sample the function's domain (and evaluate the function).

D.1.2.4 Pairwise independent versus totally independent sampling

To demonstrate the difference between the sampling bounds provided in §D.1.2.2 and §D.1.2.3, we consider the problem of estimating the average value of a function $f : \Omega \to [0, 1]$. In general, we say that a random variable Z provides an (ε, δ)-approximation of a value v if $\Pr[|Z - v| > \varepsilon] \leq \delta$. By Eq. (D.6), the average value of f evaluated at $n = O((\varepsilon^{-2} \cdot \log(1/\delta))$ *independent* samples (selected uniformly in Ω) yield an (ε, δ)-approximation of $\mu = \sum_{x \in \Omega} f(x)/|\Omega|$. Thus, the number of sample points is polynomially related to ε^{-1} and logarithmically related to δ^{-1}. In contrast, by Eq. (D.4), an (ε, δ)-approximation by n *pairwise independent* samples calls for setting $n = O(\varepsilon^{-2} \cdot \delta^{-1})$. We stress that, *in both cases the number of samples is polynomially related to the desired accuracy of the estimation* (i.e., ε). *The only advantage of totally independent samples over pairwise independent ones is in the dependency of the number of samples on the error probability* (i.e., δ).

D.2 Hashing

Hashing is extensively used in complexity theory (see, e.g., §6.2.2.2, Section 6.2.3, §6.2.4.2, §8.2.5.3, and §8.4.2.1). The typical application is for mapping arbitrary (unstructured) sets "almost uniformly" to a structured set of adequate size. Specifically, hashing is used for mapping an arbitrary 2^m-subset of $\{0,1\}^n$ to $\{0,1\}^m$ in an "almost uniform" manner.

For any fixed set S of cardinality 2^m, there exists a 1-1 mapping $f_S : S \to \{0,1\}^m$, but this mapping is not necessarily efficiently computable (e.g., it may require "knowing" the entire set S). On the other hand, no single function $f : \{0,1\}^n \to \{0,1\}^m$ can map every 2^m-subset of $\{0,1\}^n$ to $\{0,1\}^m$ in a 1-1 manner (or even approximately so). Nevertheless, for every 2^m-subset $S \subset \{0,1\}^n$, a random function $f : \{0,1\}^n \to \{0,1\}^m$ has the property that, with overwhelmingly high probability, f maps S to $\{0,1\}^m$ such that no point in the range has too many f-preimages in S. The problem is that a truly random function is unlikely to have a succinct representation (let alone an efficient evaluation algorithm). We thus seek families of functions that have a "random mapping" property (as in Item 1 of the following definition), but do have a succinct representation as well as an efficient evaluation algorithm (as in Items 2 and 3 of the following definition).

D.2.1 Definitions

Motivated by the foregoing discussion, we consider families of functions $\{H_n^m\}_{m<n}$ such that the following properties hold:

1. For every $S \subset \{0,1\}^n$, with high probability, a function h selected uniformly in H_n^m maps S to $\{0,1\}^m$ in an "almost uniform" manner. For example, we may require that, for any $|S| = 2^m$ and each point y, with high probability over the choice of h, it holds that $|\{x \in S : h(x) = y\}| \leq \mathrm{poly}(n)$.
2. The functions in H_n^m have succinct representation. For example, we may require that $H_n^m \equiv \{0,1\}^{\ell(n,m)}$, for some polynomial ℓ.
3. The functions in H_n^m can be efficiently evaluated. That is, there exists a polynomial-time algorithm that, on input a representation of a function, h (in H_n^m), and a string $x \in \{0,1\}^n$, returns $h(x)$. In some cases we make even more stringent requirements regarding the algorithm (e.g., that it runs in linear space).

Condition 1 was left vague on purpose. At the very least, we require that the expected size of $\{x \in S : h(x) = y\}$ equals $|S|/2^m$. We shall see (in Section D.2.3) that different interpretations of Condition 1 are satisfied by different families of hashing functions. We focus on t-wise independent hashing functions, defined next.

Definition D.1 (t-wise independent hashing functions): *A family H_n^m of functions from n-bit strings to m-bit strings is called t-wise independent if for every t distinct domain elements $x_1, ..., x_t \in \{0,1\}^n$ and every $y_1, ..., y_t \in \{0,1\}^m$ it holds that*

$$\mathsf{Pr}_{h \in H_n^m}[\wedge_{i=1}^t h(x_i) = y_i] \ = \ 2^{-t \cdot m}$$

That is, a uniformly chosen $h \in H_n^m$ maps every t domain elements to the range in a totally uniform manner. Note that for $t \geq 2$, it follows that the probability that a random $h \in H_n^m$ maps two distinct domain elements to the same image equals 2^{-m}. Such (families of) functions are called universal (cf. [50]), but we will focus on the stronger condition of t-wise independence.

D.2.2 Constructions

The following constructions are merely a re-interpretation of the constructions presented in §8.5.1.1. (Alternatively, one may view the constructions presented in §8.5.1.1 as a re-interpretation of the following two constructions.)

Construction D.2 (t-wise independent hashing): *For $t, m, n \in \mathbb{N}$ such that $m \leq n$, consider the following family of hashing functions mapping n-bit strings to m-bit strings. Each t-sequence $\overline{s} = (s_0, s_1, ..., s_{t-1}) \in \{0,1\}^{t \cdot n}$ describes a function*

$h_{\overline{s}} : \{0,1\}^n \rightarrow \{0,1\}^m$ *such that* $h_{\overline{s}}(x)$ *equals the* m*-bit prefix of the binary representation of* $\sum_{j=0}^{t-1} s_j x^j$*, where the arithmetic is that of* $\mathrm{GF}(2^n)$*, the finite field of* 2^n *elements.*

Proposition 8.24 implies that Construction D.2 constitutes a family of t-wise independent hash functions. Typically, we will use either $t = 2$ or $t = \Theta(n)$. To make the construction totally explicit, we need an explicit representation of $\mathrm{GF}(2^n)$; see comment following Proposition 8.24. An alternative construction for the case of $t = 2$ may be obtained analogously to the pairwise independent generator of Proposition 8.25. Recall that a Toeplitz matrix is a matrix with all diagonals being homogeneous; that is, $T = (t_{i,j})$ is a Toeplitz matrix if $t_{i,j} = t_{i+1,j+1}$, for all i, j.

Construction D.3 (alternative pairwise independent hashing): *For* $m \leq n$*, consider the family of hashing functions in which each pair* (T, b)*, consisting of a* n*-by-*m *Toeplitz matrix* T *and an* m*-dimensional vector* b*, describes a function* $h_{T,b} : \{0,1\}^n \rightarrow \{0,1\}^m$ *such that* $h_{T,b}(x) = Tx + b$*.*

Proposition 8.25 implies that Construction D.3 constitutes a family of pairwise independent hash functions. Note that a n-by-m Toeplitz matrix can be specified by $n + m - 1$ bits, yielding a description length of $n + 2m - 1$ bits. An alternative construction (analogous to Eq. (8.23) and requiring $m \cdot n + m$ bits of representation) uses arbitrary n-by-m matrices rather than Toeplitz matrices.

D.2.3 The Leftover Hash Lemma

We now turn to the "almost uniform" cover condition (i.e., Condition 1) mentioned in Section D.2.1. One concrete interpretation of this condition is given by the following lemma (and another interpretation is implied by it: see Theorem D.5).

Lemma D.4 *Let* $m \leq n$ *be integers,* H_n^m *be a family of pairwise independent hash functions, and* $S \subseteq \{0,1\}^n$*. Then, for every* $y \in \{0,1\}^m$ *and every* $\varepsilon > 0$*, for all but at most an* $\frac{2^m}{\varepsilon^2 |S|}$ *fraction of* $h \in H_n^m$ *it holds that*

$$(1-\varepsilon) \cdot \frac{|S|}{2^m} \; < \; |\{x \in S : h(x) = y\}| \; < \; (1+\varepsilon) \cdot \frac{|S|}{2^m}. \tag{D.7}$$

Note that by pairwise independence (or rather even by 1-wise independence), the expected size of $\{x \in S : h(x) = y\}$ is $|S|/2^m$, where the expectation is taken uniformly over all $h \in H_n^m$. The lemma upper bounds the fraction of h's that deviate from the expected behavior (i.e., for which $|h^{-1}(y) \cap S| \neq (1 \pm \varepsilon) \cdot |S|/2^m$). Needless to say, the bound is meaningful only in case $|S| > 2^m/\varepsilon^2$. Focusing on the case that $|S| > 2^m$ and setting $\varepsilon = \sqrt[3]{2^m/|S|}$, we infer that *for all but at most an* ε *fraction of* $h \in H_n^m$ *it holds that* $|\{x \in S : h(x) = y\}| = (1 \pm \varepsilon) \cdot |S|/2^m$. Thus, each range element has approximately the right number of h-preimages in the set S, under almost all $h \in H_n^m$.

Proof: Fixing an arbitrary set $S \subseteq \{0,1\}^n$ and an arbitrary $y \in \{0,1\}^m$, we estimate the probability that a uniformly selected $h \in H_n^m$ violates Eq. (D.7). We define random variables ζ_x, over the aforementioned probability space, such that $\zeta_x = \zeta_x(h)$ equal 1 if $h(x) = y$ and $\zeta_x = 0$ otherwise. The expected value of $\sum_{x\in S} \zeta_x$ is $\mu \stackrel{\text{def}}{=} |S| \cdot 2^{-m}$, and we are interested in the probability that this sum deviates from the expectation. Applying Chebyshev's Inequality, we get

$$\mathsf{Pr}\left[\left|\mu - \sum_{x\in S} \zeta_x\right| \geq \varepsilon \cdot \mu\right] \; < \; \frac{\mu}{\varepsilon^2 \mu^2}$$

because $\mathsf{Var}[\sum_{x\in S} \zeta_x] < |S| \cdot 2^{-m}$ by the pairwise independence of the ζ_x's and the fact that $\mathsf{E}[\zeta_x] = 2^{-m}$. The lemma follows. ■

A generalization (called mixing). The proof of Lemma D.4 can be easily extended to show that *for every set* $T \subset \{0,1\}^m$ *and every* $\varepsilon > 0$*, for all but at most an* $\frac{2^m}{|T|\cdot|S|\varepsilon^2}$ *fraction of* $h \in H_n^m$ *it holds that* $|\{x \in S : h(x) \in T\}| = (1 \pm \varepsilon) \cdot |T| \cdot |S|/2^m$. (Hint: redefine $\zeta_x = \zeta(h) = 1$ if $h(x) \in T$ and $\zeta_x = 0$ otherwise.) This assertion is meaningful provided that $|T| \cdot |S| > 2^m/\varepsilon^2$, and in the case that $m = n$ it is called a mixing property.

An extremely useful corollary. The aforementioned generalization of Lemma D.4 asserts that, for any fixed set of preimages $S \subset \{0,1\}^n$ and any fixed sets of images $T \subset \{0,1\}^m$, most functions in H_n^m behave well with respect to S and T (in the sense that they map approximately the adequate fraction of S (i.e., $|T|/2^m$) to T). A seemingly stronger statement, which is (non-trivially) implied by Lemma D.4 itself, reverses the order of quantification with respect to T; that is, for all adequate sets S, most functions in H_n^m map S to $\{0,1\}^m$ in an almost uniform manner (i.e., assign each set T approximately the adequate fraction of S, where here the approximation is up to an additive deviation). As we shall see, this is a consequence of the following theorem.

Theorem D.5 (a.k.a Leftover Hash Lemma): *Let* H_n^m *and* $S \subseteq \{0,1\}^n$ *be as in Lemma D.4, and define* $\varepsilon = \sqrt[3]{2^m/|S|}$. *Consider random variables* X *and* H *that are uniformly distributed on* S *and* H_n^m*, respectively. Then, the statistical distance between* $(H, H(X))$ *and* (H, U_m) *is at most* 2ε.

It follows that, *for* X *and* ε *as in Theorem D.5 and any* $\alpha > 0$*, for all but at most an* α *fraction of the functions* $h \in H_n^m$ *it holds that* $h(X)$ *is* $(2\varepsilon/\alpha)$*-close to* U_m.[2] (Using the terminology of the subsequent Section D.4, we may say that Theorem D.5 asserts that H_n^m yields a strong extractor (with parameters to be spelled out there).)

Proof: Let V denote the set of pairs (h, y) that violate Eq. (D.7), and $\overline{V} \stackrel{\text{def}}{=} (H_n^m \times \{0,1\}^m) \setminus V$. Then for every $(h, y) \in \overline{V}$ it holds that

$$\begin{aligned}\mathsf{Pr}[(H, H(X)) = (h, y)] &= \mathsf{Pr}[H = h] \cdot \mathsf{Pr}[h(X) = y] \\ &= (1 \pm \varepsilon) \cdot \mathsf{Pr}[(H, U_m) = (h, y)].\end{aligned}$$

[2]This follows by defining a random variable $\zeta = \zeta(h)$ such that ζ equals the statistical distance between $h(X)$ and U_m, and applying Markov's inequality.

On the other hand, by the setting of ε and Lemma D.4 (which imply that $\mathsf{Pr}[(H,y) \in V] \le \varepsilon$ for every $y \in \{0,1\}^m$), we have $\mathsf{Pr}[(H,U_m) \in V] \le \varepsilon$. It follows that

$$\begin{aligned} \mathsf{Pr}[(H,H(X)) \in V] &= 1 - \mathsf{Pr}[(H,H(X)) \in \overline{V}] \\ &\le 1 - \mathsf{Pr}[(H,U_m)) \in \overline{V}] + \varepsilon \ \le\ 2\varepsilon. \end{aligned}$$

Using all these upper-bounds, we upper-bounded the statistical difference between $(H,H(X))$ and (H,U_m), denoted Δ, by separating the contribution of V and $\overline{V}$. Specifically, we have

$$\begin{aligned} \Delta &= \frac{1}{2} \cdot \sum_{(h,y) \in H_n^m \times \{0,1\}^m} |\mathsf{Pr}[(H,H(X)) = (h,y)] - \mathsf{Pr}[(H,U_m) = (h,y)]| \\ &\le \frac{\varepsilon}{2} + \frac{1}{2} \cdot \sum_{(h,y) \in V} |\mathsf{Pr}[(H,H(X)) = (h,y)] - \mathsf{Pr}[(H,U_m) = (h,y)]|\ , \end{aligned}$$

where the first term upper-bounds the contribution of all pairs $(h,y) \in \overline{V}$. Hence,

$$\begin{aligned} \Delta &\le \frac{\varepsilon}{2} + \frac{1}{2} \cdot \sum_{(h,y) \in V} (\mathsf{Pr}[(H,H(X)) = (h,y)] + \mathsf{Pr}[(H,U_m) = (h,y)]) \\ &\le \frac{\varepsilon}{2} + \frac{1}{2} \cdot (2\varepsilon + \varepsilon)\ , \end{aligned}$$

where the first inequality is trivial (i.e., $|\alpha - \beta| \le \alpha + \beta$ for any non-negative α and β), and the second inequality uses the foregoing upper-bounds (i.e., $\mathsf{Pr}[(H,H(X)) \in V] \le 2\varepsilon$ and $\mathsf{Pr}[(H,U_m) \in V] \le \varepsilon$). The theorem follows. ■

An alternative proof of Theorem D.5. Define the collision probability of a random variable Z, denote $\mathsf{cp}(Z)$, as the probability that two independent samples of Z yield the same result. Alternatively, $\mathsf{cp}(Z) \stackrel{\text{def}}{=} \sum_z \mathsf{Pr}[Z = z]^2$. Theorem D.5 follows by combining the following two facts:

1. A general fact: *If $Z \in [N]$ and $\mathsf{cp}(Z) \le (1+4\epsilon^2)/N$ then Z is ϵ-close to the uniform distribution on $[N]$.*

   We prove the contra-positive: Assuming that the statistical distance between Z and the uniform distribution on $[N]$ equals δ, we show that $\mathsf{cp}(Z) \ge (1+4\delta^2)/N$. This is done by defining $L \stackrel{\text{def}}{=} \{z : \mathsf{Pr}[Z = z] < 1/N\}$, and lower-bounding $\mathsf{cp}(Z)$ by using the fact that the collision probability is minimized on uniform distributions. Specifically, considering the uniform distributions on L and $[N] \setminus L$ respectively, we have

   $$\mathsf{cp}(Z) \ge |L| \cdot \left(\frac{\mathsf{Pr}[Z \in L]}{|L|} \right)^2 + (N - |L|) \cdot \left(\frac{\mathsf{Pr}[Z \in [N] \setminus L]}{N - |L|} \right)^2_. \qquad \text{(D.8)}$$

   Using $\delta = \rho - \mathsf{Pr}[Z \in L]$, where $\rho = |L|/N$, the r.h.s of Eq. (D.8) equals $\frac{(\rho-\delta)^2}{\rho N} + \frac{(1-(\rho-\delta))^2}{(1-\rho)N} = \left(1 + \frac{\delta^2}{(1-\rho)\rho}\right) \cdot \frac{1}{N} \ge \left(1 + 4\delta^2\right) \cdot \frac{1}{N}$.

2. *The collision probability of* $(H, H(X))$ *is at most* $(1 + (2^m/|S|))/(|H_n^m| \cdot 2^m)$. (Furthermore, this holds even if H_n^m is only universal.)

   The proof is by a straightforward calculation. Specifically, note that $\mathsf{cp}(H, H(X)) = |H_n^m|^{-1} \cdot \mathsf{E}_{h \in H_n^m}[\mathsf{cp}(h(X))]$, whereas $\mathsf{E}_{h \in H_n^m}[\mathsf{cp}(h(X))] = |S|^{-2} \sum_{x_1, x_2 \in S} \mathsf{Pr}[H(x_1) = H(x_2)]$. The sum equals $|S| + (|S|^2 - |S|) \cdot 2^{-m}$, and so $\mathsf{cp}(H, H(X)) < |H_n^m|^{-1} \cdot (2^{-m} + |S|^{-1})$.

It follows that $(H, H(X))$ is $2\sqrt{2^m/|S|}$-close to (H, U_m), which is actually a stronger bound than the one asserted by Theorem D.5.

Stronger uniformity via higher independence. Recall that Lemma D.4 asserts that for each point in the range of the hash function, with high probability over the choice of the hash function, *this fixed point* has approximately the expected number of preimages in S. A stronger condition asserts that, with high probability over the choice of the hash function, *every point in its range* has approximately the expected number of preimages in S. Such a guarantee can be obtained when using n-wise independent hash functions (rather than using pairwise independent hash functions).

Lemma D.6 *Let $m \leq n$ be integers, H_n^m be a family of n-wise independent hash functions, and $S \subseteq \{0,1\}^n$. Then, for every $\varepsilon \in (0,1)$, for all but at most an $2^m \cdot (n \cdot 2^m/\varepsilon^2 |S|)^{n/2}$ fraction of the functions $h \in H_n^m$, it is the case that Eq.* (D.7) *holds for every $y \in \{0,1\}^m$.*

Indeed, the lemma should be used with $2^m < \varepsilon^2 |S|/4n$. In particular, using $m = \log_2 |S| - \log_2(5n/\varepsilon^2)$ guarantees that with high probability (i.e., $1 - 2^m \cdot 5^{-n/2} \geq 1 - (4/5)^{n/2}$) each range elements has $(1 \pm \varepsilon) \cdot |S|/2^m$ preimages in S. Under this setting of parameters $|S|/2^m = 5n/\varepsilon^2$, which is $\mathrm{poly}(n)$ whenever $\varepsilon = 1/\mathrm{poly}(n)$. Needless to say, this guarantee is stronger than the conclusion of Theorem D.5.

Proof: The proof follows the footsteps of the proof of Lemma D.4, taking advantage of the fact that here the random variables (i.e., the ζ_x's) are n-wise independent. For $t = n/2$, this allows using the so-called $2t^{\text{th}}$ *moment analysis*, which generalizes the second moment analysis of pairwise independent sampling (presented in §D.1.2.2). As in the proof of Lemma D.4, we fix any S and y, and define $\zeta_x = \zeta_x(h) = 1$ if and only if $h(x) = y$. Letting $\mu = \mathsf{E}[\sum_{x \in S} \zeta_x] = |S|/2^m$ and $\overline{\zeta}_x = \zeta_x - \mathsf{E}(\zeta_x)$, we start with Markov's inequality:

$$\begin{aligned} \mathsf{Pr}\left[\left|\mu - \sum_{x \in S} \zeta_x\right| \geq \varepsilon \cdot \mu\right] &\leq \frac{\mathsf{E}[(\sum_{x \in S} \overline{\zeta}_x)^{2t}]}{\varepsilon^{2t} \mu^{2t}} \\ &= \frac{\sum_{x_1, \ldots, x_{2t} \in S} \mathsf{E}[\prod_{i=1}^{2t} \overline{\zeta}_{x_i}]}{\varepsilon^{2t} \cdot (|S|/2^m)^{2t}} \end{aligned} \tag{D.9}$$

Using $2t$-wise independence, we note that only the terms in Eq. (D.9) that do not vanish are those in which each variable appears with multiplicity. This mean that only terms having less than t distinct variables contribute to Eq. (D.9). Now, for every $j \leq t$, we have less than $\binom{|S|}{j} \cdot (2t!) < (2t!/j!) \cdot |S|^j$ terms with j distinct variables, and each such term contributes less than $(2^{-m})^j$ to the sum (because for every $e > 1$ it holds that $\mathsf{E}[\overline{\zeta}_{x_i}^e] < \mathsf{E}[\zeta_{x_i}] = 2^{-m}$). Thus, Eq. (D.9) is upper-bounded by

$$\frac{2t!}{(\varepsilon|S|/2^m)^{2t}} \cdot \sum_{j=1}^{t} \frac{(|S|/2^m)^j}{j!} \;<\; 2 \cdot \frac{2t!/t!}{(\varepsilon^2|S|/2^m)^t} \;<\; \left(\frac{2t \cdot 2^m}{\varepsilon^2|S|}\right)^t$$

where the first inequality assumes $|S| > n2^m$ (which is justified by the fact that the claim hold vacuously otherwise). This upper-bounds the probability that a random $h \in H_n^m$ violates Eq. (D.7) with respect to a fixed y. Using a union bound on all $y \in \{0,1\}^m$, the lemma follows. ■

D.3 Sampling

In many settings repeated sampling is used to estimate the average (or other statistics) of a huge set of values.[3] Namely, given a "value" function $\nu : \{0,1\}^n \to \mathbb{R}$, one wishes to approximate $\bar{\nu} \stackrel{\rm def}{=} \frac{1}{2^n}\sum_{x\in\{0,1\}^n} \nu(x)$ without having to inspect the value of ν at each point of the domain. The obvious thing to do is sampling the domain at random, and obtaining an approximation to $\bar{\nu}$ by taking the average of the values of ν on the sample points. It turns out that certain "pseudorandom" sequences of sample points may serve almost as well as truly random sequences of sample points, and thus the foregoing problem is indeed related to Section 8.5.

D.3.1 Formal Setting

It is essential to have the range of the function ν be bounded (since otherwise no reasonable approximation is possible). For simplicity, we adopt the convention of having $[0,1]$ be the range of ν, and the problem for other (predetermined) ranges can be treated analogously. Our notion of approximation depends on two parameters: accuracy (denoted ε) and error probability (denoted δ). We wish to have an algorithm that, with probability at least $1-\delta$, gets within ε of the correct value. This leads to the following definition.

Definition D.7 (sampler): *A* sampler *is a randomized oracle machine that on input parameters n* (length)*, ε* (accuracy) *and δ* (error)*, and oracle access to* any *function $\nu : \{0,1\}^n \to [0,1]$, outputs, with probability at least $1-\delta$, a value that is at most ε away from $\bar{\nu} \stackrel{\rm def}{=} \frac{1}{2^n}\sum_{x\in\{0,1\}^n} \nu(x)$. Namely,*

$$\mathsf{Pr}[|\mathrm{sampler}^{\nu}(n,\varepsilon,\delta) - \bar{\nu}| > \varepsilon] \;<\; \delta$$

where the probability is taken over the internal coin tosses of the sampler.
A non-adaptive sampler *is a sampler that consists of two deterministic algorithms: a* sample generating *algorithm, G, and a* evaluation algorithm*, V. On input n, ε, δ*

[3] Indeed, this problem was already mentioned in §D.1.2.4.

and a random seed *of adequate length, algorithm G generates a sequence of queries, denoted $s_1, ..., s_m \in \{0,1\}^n$. Algorithm V is given the corresponding sequence of ν-values (i.e., $\nu(s_1), ..., \nu(s_m)$) and outputs an estimate to $\bar{\nu}$.*

We are interested in "the complexity of sampling" quantified as a function of the parameters n, ε and δ. Specifically, we will consider three complexity measures: The sample complexity (i.e., the number of oracle queries made by the sampler); the randomness complexity (i.e., the length of the random seed used by the sampler); and the computational complexity (i.e., the running-time of the sampler). We say that a sampler is efficient if its running-time is polynomial in the total length of its queries (i.e., polynomial in both its sample complexity and in n). We will focus on efficient samplers. Furthermore, we will be most interested in efficient samplers that have optimal (up-to a constant factor) sample complexity, and will seek to minimize the randomness complexity of such samplers. Note that minimizing the randomness complexity without referring to the sample complexity makes no sense.

D.3.2 Known Results

We note that all the following positive results refer to non-adaptive samplers, whereas the lower bound hold also for general samplers. For more details on these results, see [90, Sec. 3.6.4] and the references therein.

The naive sampler. The straightforward method (a.k.a the naive sampler) consists of *uniformly and independently* selecting sufficiently many sample points (queries), and outputting the average value of the function on these points. Using Chernoff Bound it follows that $O(\frac{\log(1/\delta)}{\varepsilon^2})$ sample points suffice. As indicated next, the naive sampler is optimal (up-to a constant factor) in its sample complexity, but is quite wasteful in randomness.

It is known that $\Omega(\frac{\log(1/\delta)}{\varepsilon^2})$ samples are needed in any sampler, and that any sampler that makes $s(n, \varepsilon, \delta)$ queries must have randomness complexity at least $n + \log_2(1/\delta) - \log_2 s(n, \varepsilon, \delta) - O(1)$. These lower bounds are tight (as demonstrated by non-explicit and inefficient samplers). The foregoing facts guide our quest for improvements, which is aimed at finding more randomness-efficient ways of *efficiently* generating sample sequences that can be used in conjunction with an appropriate evaluation algorithm V. (We stress that V need not necessarily take the average of the values of the sampled points.)

The pairwise-independent sampler. Using a pairwise-independence generator (cf. §8.5.1.1) for generating sample points, along with the natural evaluation algorithm (which outputs the average of the values of these points), we can obtain a great saving in the randomness complexity: In particular, using a seed of length $2n$, we can generate $O(1/\delta\varepsilon^2)$ pairwise-independent sample points, which (by Eq. (D.4)) suffice for getting accuracy ε with error δ. Thus, this (Pairwise-Independent) sampler uses $2n$ coin tosses rather than the $\Omega((\log(1/\delta))\varepsilon^{-2} \cdot n)$ coin tosses used by the naive sampler. Furthermore, for constant $\delta > 0$, the Pairwise-Independent Sampler is optimal up-to a constant factor in both its sample and randomness complexities. However, for small δ (i.e., $\delta = o(1)$), this sampler is wasteful in sample complexity.

The Median-of-Averages sampler. A new idea is required for going further, and a relevant tool – random walks on expander graphs (see Sections 8.5.3 and E.2) – is needed too. Specifically, we combine the Pairwise-Independent Sampler with the Expander Random Walk Generator (of Proposition 8.29) to obtain a new sampler. The new sampler uses a t-long random walk on an expander with vertex set $\{0,1\}^{2n}$ for *generating a sequence of* $t \stackrel{\text{def}}{=} O(\log(1/\delta))$ *related seeds for* t *invocations of the Pairwise-Independent Sampler*, where each of these invocations uses the corresponding $2n$ bits to generate a sequence of $O(1/\varepsilon^2)$ samples in $\{0,1\}^n$. The new sampler, called the Median-of-Averages Sampler, outputs the median of the t values obtained in these t invocation of the Pairwise-Independent Sampler. In analyzing this sampler, we first note that each of the foregoing t invocations returns a value that, with probability at least 0.9, is ε-close to $\bar{\nu}$. By Theorem 8.28 (see also Exercise 8.44), with probability at least $1-\exp(-t) = 1-\delta$, *most* of these t invocations return an ε-close approximation. Hence, *the median among these* t *values is an* (ε, δ)*-approximation to the correct value*. The resulting sampler has sample complexity $O(\frac{\log(1/\delta)}{\varepsilon^2})$ and randomness complexity $2n+O(\log(1/\delta))$, which is optimal up-to a constant factor in both complexities.

Further improvements. The randomness complexity of the Median-of-Averages Sampler can be decreased from $2n + O(\log(1/\delta))$ to $n + O(\log(1/\delta\varepsilon))$, while maintaining its (optimal) sample complexity (of $O(\frac{\log(1/\delta)}{\varepsilon^2})$). This is done by replacing the Pairwise Independent Sampler by a sampler that picks a random vertex in a suitable expander, samples all its neighbors, and outputs the average value seen.

Averaging Samplers. Averaging (a.k.a. "Oblivious") samplers are non-adaptive samplers in which the evaluation algorithm is the natural one: that is, it merely outputs the average of the values of the sampled points. Indeed, the Pairwise-Independent Sampler is an averaging sampler, whereas the Median-of-Averages Sampler is not. Interestingly, averaging samplers have applications for which ordinary non-adaptive samplers do not suffice. Averaging samplers are closely related to randomness extractors, defined and discussed in the subsequent Section D.4.

An odd perspective. Recall that a non-adaptive sampler consists of a sample generator G and an evaluator V such that for every $\nu : \{0,1\}^n \to [0,1]$ it holds that

$$\mathsf{Pr}_{(s_1,...,s_m)\leftarrow G(U_k)}[|V(\nu(s_1),...,\nu(s_m)) - \bar{\nu}| > \varepsilon] \;<\; \delta, \tag{D.10}$$

where k denotes the length of the sampler's (random) seed. Thus, we may view G as a pseudorandom generator that is subjected to a class of distinguishers that is determined by a fixed algorithm V and an arbitrary function $\nu : \{0,1\}^n \to [0,1]$. Specifically, assuming that V works well when the m samples are distributed uniformly and independently (i.e., $\mathsf{Pr}[|V(\nu(U_n^{(1)}),...,\nu(U_n^{(m)})) - \bar{\nu}| > \varepsilon] \;<\; \delta$), we require G to generate sequences that satisfy the corresponding condition (as stated in Eq. (D.10)). What is a bit odd about the foregoing perspective is that, except for the case of averaging samplers, the class of distinguishers considered here is effected by a component (i.e., the evaluator V) that is potentially custom-made to help the generator G fool the distinguisher.[4]

D.3.3 Hitters

Hitters may be viewed as a relaxation of samplers. Specifically, considering only Boolean functions, hitters are required to generate a sample that contains a point evaluating to 1 whenever at least an ε fraction of the function values equal 1. That is, a hitter is a randomized algorithm that on input parameters n (length), ε (accuracy) and δ (error), outputs a list of n-bit strings such that, for every set $S \subseteq \{0,1\}^n$ of density greater than ε, with probability at least $1-\delta$, the list contains at least one element of S. Note the correspondence to the (ε,δ)-hitting problem defined in Section 8.5.3.

Needless to say, any sampler yields a hitter (with respect to essentially the same parameters n, ε and δ).[5] However, hitting is strictly easier than evaluating the density of the target set: $O(1/\varepsilon)$ (pairwise independent) random samples suffice to hit any set of density ε with constant probability, whereas $\Omega(1/\varepsilon^2)$ samples are needed for approximating the average value of a Boolean function up to accuracy ε (with constant error probability). Indeed, adequate simplifications of the samplers discussed in Appendix D.3.2 yield hitters with sample complexity proportional to $1/\varepsilon$ (rather than to $1/\varepsilon^2$).

D.4 Randomness Extractors

Extracting almost-perfect randomness from sources of weak (i.e., defected) randomness is crucial for the actual use of randomized algorithms, procedures and protocols. The latter are analyzed assuming that they are given access to a perfect random source, while in reality one typically has access only to sources of weak (i.e., highly imperfect) randomness. This gap is bridged by using randomness extractors, which are efficient procedures that (possibly with the help of little extra randomness) convert any source of weak randomness into an almost-perfect random source. Thus, randomness extractors are devices that greatly enhance the quality

[4] Another aspect in which samplers differ from the various pseudorandom generators discussed in Chapter 8 is in the aim to minimize, rather than maximize, the number of "blocks" (denoted here by m) in the output sequence. However, also in the case of samplers the aim is to maximize the block-length (denoted here by n).

[5] Specifically, any sampler with respect to the parameters n, ε and δ, yields a hitter with respect to the parameters n, 2ε and δ. (The need for slackness is easily demonstrated by noting that estimating the average with accuracy $\varepsilon = 1/2$ is trivial, whereas hitting is non-trivial for any accuracy (density) $\varepsilon < 1$.) The claim is obvious for non-adaptive samplers, but actually holds also for adaptive samplers. Note that adaptivity does not provide any advantage in the context of hitters, because one may assume (without loss of generality) that all prior samples missed the target set S.

of random sources. In addition, randomness extractors are related to several other fundamental problems, to be further discussed later.

One key parameter, which was avoided in the foregoing discussion, is the class of weak random sources from which we need to extract almost perfect randomness. Needless to say, it is preferable to make as little assumptions as possible regarding the weak random source. In other words, we wish to consider a wide class of such sources, and require that the randomness extractor (often referred to as the extractor) "works well" for any source in this class. A general class of such sources is defined in §D.4.1.1, but first we wish to mention that even for very restricted classes of sources no deterministic extractor can work.[6] To overcome this impossibility result, two approaches are used:

Seeded extractors: The first approach consists of considering randomized extractors that use a relatively small amount of randomness (in addition to the weak random source). That is, these extractors obtain two inputs: a short truly random seed and a relatively long sequence generated by an arbitrary source that belongs to the specified class of sources. This suggestion is motivated in two different ways:

1. The application may actually have access to an almost-perfect random source, but bits from this high-quality source are much more expensive than bits from the weak (i.e., low-quality) random source. Thus, it makes sense to obtain few high-quality bits from the almost-perfect source and use them to "purify" the cheap bits obtained from the weak (low-quality) source. Thus, combining many cheap (but low-quality) bits with few high-quality (but expensive) bits, we obtain many high-quality bits.
2. In some applications (e.g., when using randomized algorithms), it may be possible to invoke the application multiple times, and use the "typical" outcome of these invocations (e.g., rule by majority in the case of a decision procedure). For such applications, we may proceed as follows: first we obtain an outcome r of the weak random source, then we invoke the application multiple times such that for every possible seed s we invoke the application feeding it with $\texttt{extract}(s, r)$, and finally we use the "typical" outcome of these invocations. Indeed, this is analogous to the context of derandomization (see Section 8.3), and likewise this alternative is typically not applicable to cryptographic and/or distributed settings.

Few independent sources: The second approach consists of considering deterministic extractors that obtain samples from a few (say two) *independent* sources of weak randomness. Such extractors are applicable in any setting (including in cryptography), provided that the application has access to the required number of independent weak random sources.

[6] For example, consider the class of sources that output n-bit strings such that no string occurs with probability greater than $2^{-(n-1)}$ (i.e., twice its probability weight under the uniform distribution).

In this section we focus on the first type of extractors (i.e., the *seeded extractors*). This choice is motivated both by the relatively more mature state of the research of seeded extractors and by the closer connection between seeded extractors and other topics in complexity theory.

D.4.1 Definitions and various perspectives

We first present a definition that corresponds to the foregoing motivational discussion, and later discuss its relation to other topics in complexity.

D.4.1.1 The Main Definition

A very wide class of weak random sources corresponds to sources in which no specific output is too probable. That is, the class is parameterized by a (probability) bound β and consists of all sources X such that for every x it holds that $\mathsf{Pr}[X = x] \leq \beta$. In such a case, we say that X has min-entropy[7] at least $\log_2(1/\beta)$. Indeed, we represent sources as random variables, and assume that they are distributed over strings of a fixed length, denoted n. An (n,k)-source is a source that is distributed over $\{0,1\}^n$ and has min-entropy at least k.

An interesting special case of (n,k)-sources is that of sources that are uniform over some subset of 2^k strings. Such sources are called (n,k)-flat. A useful observation is that *each (n,k)-source is a convex combination of (n,k)-flat sources.*

Definition D.8 (extractor for (n,k)-sources):

1. *An algorithm* $\mathrm{Ext}:\{0,1\}^d \times \{0,1\}^n \to \{0,1\}^m$ *is called an* extractor with error ε for the class $\mathcal{C}$ *if for every source X in $\mathcal{C}$ it holds that* $\mathrm{Ext}(U_d, X)$ *is ε-close to U_m. If $\mathcal{C}$ is the class of (n,k)-sources then* Ext *is called a* (k,ε)-extractor.

2. *An algorithm* Ext *is called a* strong extractor with error ε for $\mathcal{C}$ *if for every source X in $\mathcal{C}$ it holds that* $(U_d, \mathrm{Ext}(U_d, X))$ *is ε-close to* (U_d, U_m)*. A* strong (k,ε)-extractor *is defined analogously.*

Using the aforementioned "decomposition" of (n,k)-sources into (n,k)-flat sources, it follows that Ext *is a (k,ε)-extractor if and only if it is an extractor with error ε for the class of (n,k)-flat sources.* (A similar claim holds for strong extractors.) Thus, much of the technical analysis is conducted with respect to the class of (n,k)-flat sources. For example, by analyzing the case of (n,k)-flat sources it is easy to see that, for $d = \log_2(n/\varepsilon^2) + O(1)$, there exists a (k,ε)-extractor $\mathrm{Ext} : \{0,1\}^d \times \{0,1\}^n \to \{0,1\}^k$. (The proof employs the Probabilistic Method and uses a union bound on the (finite) set of all (n,k)-flat sources.)[8]

[7] Recall that the entropy of a random variable X is defined as $\sum_x \mathsf{Pr}[X = x] \cdot \log_2(1/\mathsf{Pr}[X = x])$. Indeed the min-entropy of X equals $\min_x\{\log_2(1/\mathsf{Pr}[X = x])\}$, and is always upper-bounded by its entropy.

[8] Indeed, the key fact is that the number of (n,k)-flat sources is $N \stackrel{\text{def}}{=} \binom{2^n}{2^k}$. The probability that a random function $\mathrm{Ext} : \{0,1\}^d \times \{0,1\}^n \to \{0,1\}^k$ is not an extractor with error ε for a

We seek, however, explicit extractors; that is, extractors that are implementable by polynomial-time algorithms. We note that the evaluation algorithm of any family of pairwise independent hash functions mapping n-bit strings to m-bit strings constitutes a (strong) (k, ε)-extractor for $\varepsilon = 2^{-\Omega(k-m)}$ (see Theorem D.5). However, these extractors necessarily use a long seed (i.e., $d \geq 2m$ must hold (and in fact $d = n + 2m - 1$ holds in Construction D.3)). In Section D.4.2 we survey constructions of efficient (k, ε)-extractors that obtain logarithmic seed length (i.e., $d = O(\log(n/\varepsilon))$). But before doing so, we provide a few alternative perspectives on extractors.

An important note on logarithmic seed length. The case of logarithmic seed length (i.e., $d = O(\log(n/\varepsilon))$) is of particular importance for a variety of reasons. Firstly, when emulating a randomized algorithm using a defected random source (as in Item 2 of the motivational discussion of seeded extractors), the overhead is exponential in the length of the seed. Thus, the emulation of a generic probabilistic polynomial-time algorithm can be done in polynomial time only if the seed length is logarithmic. Similarly, the applications discussed in §D.4.1.2 and §D.4.1.3 are feasible only if the seed length is logarithmic. Lastly, we note that logarithmic seed length is an absolute lower-bound for (k, ε)-extractors, whenever $k < n - n^{\Omega(1)}$ (and the extractor is non-trivial (i.e., $m \geq 1$ and $\varepsilon < 1/2$)).

D.4.1.2 Extractors as averaging samplers

There is a close relationship between extractors and averaging samplers (which are defined towards the end of Section D.3.2). We shall first show that any averaging sampler gives rise to an extractor. Let $G : \{0,1\}^n \to (\{0,1\}^m)^t$ be the sample generating algorithm of an averaging sampler having accuracy ε and error probability δ. That is, G uses n bits of randomness and generates t sample points in $\{0,1\}^m$ such that, for every $f : \{0,1\}^m \to [0,1]$ with probability at least $1 - \delta$, the average of the f-values of these t pseudorandom points resides in the interval $[\overline{f} \pm \varepsilon]$, where $\overline{f} \stackrel{\text{def}}{=} \mathsf{E}[f(U_m)]$. Define $\mathrm{Ext} : [t] \times \{0,1\}^n \to \{0,1\}^m$ such that $\mathrm{Ext}(i, r)$ is the i^{th} sample generated by $G(r)$. We shall prove that Ext is a $(k, 2\varepsilon)$-extractor, for $k = n - \log_2(\varepsilon/\delta)$.

Suppose towards the contradiction that there exists a (n, k)-flat source X such that for some $S \subset \{0,1\}^m$ it is the case that $\mathsf{Pr}[\mathrm{Ext}(U_d, X) \in S] > \mathsf{Pr}[U_m \in S] + 2\varepsilon$, where $d = \log_2 t$ and $[t] \equiv \{0,1\}^d$. Define

$$B = \{x \in \{0,1\}^n : \mathsf{Pr}[\mathrm{Ext}(U_d, x) \in S] > (|S|/2^m) + \varepsilon\}.$$

Then, $|B| > \varepsilon \cdot 2^k = \delta \cdot 2^n$. Defining $f(z) = 1$ if $z \in S$ and $f(z) = 0$ otherwise, we have $\overline{f} \stackrel{\text{def}}{=} \mathsf{E}[f(U_m)] = |S|/2^m$. But, for every $r \in B$ the f-average of the sample

fixed (n, k)-flat source is upper-bounded by $p \stackrel{\text{def}}{=} 2^{2^k} \cdot \exp(-\Omega(2^{d+k}\varepsilon^2))$, because p bounds the probability that when selecting 2^{d+k} random k-bit long strings there exists a set $T \subset \{0,1\}^k$ that is hit by more than $((|T|/2^k) + \varepsilon) \cdot 2^{d+k}$ of these strings. Note that for $d = \log_2(n/\varepsilon^2) + O(1)$ it holds that $N \cdot p \ll 1$. In fact, the same analysis applies to the extraction of $m = k + \log_2 n$ bits (rather than k bits).

$G(r)$ is greater than $\overline{f}+\varepsilon$, in contradiction to the hypothesis that the sampler has error probability δ (with respect to accuracy ε).

We now turn to show that extractors give rise to averaging samplers. Let $\mathrm{Ext}:\{0,1\}^d\times\{0,1\}^n\to\{0,1\}^m$ be a (k,ε)-extractor. Consider the sample generation algorithm $G:\{0,1\}^n\to(\{0,1\}^m)^{2^d}$ define by $G(r)=(\mathrm{Ext}(s,r))_{s\in\{0,1\}^d}$. We prove that G corresponds to an averaging sampler with accuracy ε and error probability $\delta=2^{-(n-k-1)}$.

Suppose towards the contradiction that there exists a function $f:\{0,1\}^m\to[0,1]$ such that for $\delta 2^n=2^{k+1}$ strings $r\in\{0,1\}^n$ the average f-value of the sample $G(r)$ deviates from $\overline{f}\stackrel{\rm def}{=}\mathsf{E}[f(U_m)]$ by more than ε. Suppose, without loss of generality, that for at least half of these r's the average is greater than $\overline{f}+\varepsilon$, and let B denote the set of these r's. Then, for X that is uniformly distributed on B and is thus a (n,k)-source, we have

$$\mathsf{E}[f(\mathrm{Ext}(U_d,X))]>\mathsf{E}[f(U_m)]+\varepsilon,$$

which (using $|f(z)|\leq 1$ for every z) contradicts the hypothesis that $\mathrm{Ext}(U_d,X)$ is ε-close to U_m.

D.4.1.3 Extractors as randomness-efficient error-reductions

As may be clear from the foregoing discussion, extractors yield randomness-efficient methods for error-reduction. This is the case because *error-reduction is a special case of the sampling problem*, obtained by considering Boolean functions. Specifically, for a two-sided error decision procedure A, consider the function $f_x:\{0,1\}^{\rho(|x|)}\to\{0,1\}$ such that $f_x(r)=1$ if $A(x,r)=1$ and $f_x(r)=0$ otherwise. Assuming that the probability that A is correct is at least $0.5+\varepsilon$ (say $\varepsilon=1/6$), error reduction amounts to providing a sampler with accuracy ε and any desired error probability $\delta\ll\varepsilon$ for the Boolean function f_x. Thus, by §D.4.1.2, any (k,ε)-extractor $\mathrm{Ext}:\{0,1\}^d\times\{0,1\}^n\to\{0,1\}^{\rho(|x|)}$ with $k=n-\log(1/\delta)-1$ yields the desired error-reduction, provided that 2^d is feasible (e.g., $2^d=\mathrm{poly}(\rho(|x|))$, where $\rho(\cdot)$ represents the randomness complexity of the original algorithm A). The question of interest here is how does n (which represents the randomness complexity of the corresponding sampler) grow as a function of $\rho(|x|)$ and δ.

Error-reduction using the extractor $\mathrm{Ext}:[\mathrm{poly}(\rho(|x|))]\times\{0,1\}^n\to\{0,1\}^{\rho(|x|)}$

	error probability	randomness complexity
original algorithm	$1/3$	$\rho(\|x\|)$
resulting algorithm	δ (may depend on $\|x\|$)	n (function of $\rho(\|x\|)$ and δ)

Needless to say, the answer to the foregoing question depends on the quality of the extractor that we use. In particular, using Part 1 of the forthcoming Theorem D.10, we note that for every $\alpha>1$, one can obtain $n=O(\rho(|x|))+\alpha\log_2(1/\delta)$, for any $\delta>2^{-\mathrm{poly}(\rho(|x|))}$. Note that, for $\delta<2^{-O(\rho(|x|))}$, this bound on the randomness-complexity of error-reduction is better than the bound of $n=\rho(|x|)+O(\log(1/\delta))$ that is provided (for the reduction of one-sided error) by the Expander Random

Walk Generator (of Section 8.5.3), albeit the number of samples here is larger (i.e., $\mathrm{poly}(\rho(|x|)/\delta)$ rather than $O(\log(1/\delta))$).

Mentioning the reduction of *one-sided* error-probability brings us to a corresponding relaxation of the notion of an extractor, which is called a disperser. Loosely speaking, a (k,ε)-disperser is only required to hit (with positive probability) any set of density greater than ε in its image, rather than produce a distribution that is ε-close to uniform.

Definition D.9 (dispersers): *An algorithm* $\mathrm{Dsp} : \{0,1\}^d \times \{0,1\}^n \to \{0,1\}^m$ *is called a* (k,ε)-disperser *if for every* (n,k)*-source* X *the support of* $\mathrm{Dsp}(U_d, X)$ *covers at least* $(1-\varepsilon)\cdot 2^m$ *points. Alternatively, for every set* $S \subset \{0,1\}^m$ *of size greater than* $\varepsilon 2^m$ *it holds that* $\mathsf{Pr}[\mathrm{Dsp}(U_d, X) \in S] > 0$.

Dispersers can be used for the reduction of one-sided error analogously to the use of extractors for the reduction of two-sided error. Specifically, regarding the aforementioned function f_x (and assuming that $\mathsf{Pr}[f_x(U_{\ell(|x|)})=1] > \varepsilon$), we may use any (k,ε)-disperser $\mathrm{Dsp} : \{0,1\}^d \times \{0,1\}^n \to \{0,1\}^{\ell(|x|)}$ towards finding a point z such that $f_x(z) = 1$. Indeed, if $\mathsf{Pr}[f_x(U_{\ell(|x|)})=1] > \varepsilon$ then there are less than 2^k points z such that $(\forall s \in \{0,1\}^d)\, f_x(\mathrm{Dsp}(s,z)) = 0$, and thus the one-sided error can be reduced from $1-\varepsilon$ to $2^{-(n-k)}$ while using n random bits. (Note that dispersers are closely related to hitters (cf. Appendix D.3.3), analogously to the relation of extractors and averaging samplers.)

D.4.1.4 Other perspectives

Extractors and dispersers have an appealing interpretation in terms of bipartite graphs. Starting with dispersers, we view any (k,ε)-disperser $\mathrm{Dsp} : \{0,1\}^d \times \{0,1\}^n \to \{0,1\}^m$ as a bipartite graph $G = ((\{0,1\}^n, \{0,1\}^m), E)$ such that $E = \{(x, \mathrm{Dsp}(s,x)) : x \in \{0,1\}^n, s \in \{0,1\}^d\}$. This graph has the property that *any* subset of 2^k vertices on the left (i.e., in $\{0,1\}^n$) has a neighborhood that contains at least a $1-\varepsilon$ fraction of the vertices of the right, which is remarkable in the typical case where d is small (e.g., $d = O(\log n/\varepsilon)$) and $n \gg k \geq m$ whereas $m = \Omega(k)$ (or at least $m = k^{\Omega(1)}$). Furthermore, if Dsp is efficiently computable then this bipartite graph is strongly constructible in the sense that, given a vertex on the left, one can efficiently find each of its neighbors. Any (k,ε)-extractor $\mathrm{Ext} : \{0,1\}^d \times \{0,1\}^n \to \{0,1\}^m$ yields an analogous graph with an even stronger property: the neighborhood multi-set of *any* subset of 2^k vertices on the left covers the vertices on the right in an almost uniform manner.

An odd perspective. In addition to viewing extractors as averaging samplers, which in turn may be viewed within the scope of the pseudorandomness paradigm, we mention here an even more odd perspective. Specifically, randomness extractors may be viewed as randomized algorithms (distinguishers) designed on purpose such that to be fooled by any weak random source (but not by an even worse source). Specifically, for any (k,ε)-extractor $\mathrm{Ext} : \{0,1\}^d \times \{0,1\}^n \to \{0,1\}^m$, where $\varepsilon \leq 1/100$, $m = k = \omega(\log n/\varepsilon)$ and $d = O(\log n/\varepsilon)$, consider the following class of distinguishers (or tests), parameterized by subsets of $\{0,1\}^m$: for $S \subset \{0,1\}^m$, the test T_S satisfies $\mathsf{Pr}[T_S(x)=1] = \mathsf{Pr}[\mathrm{Ext}(U_d, x) \in S]$ (i.e., on input $x \in \{0,1\}^n$, the test uniformly selects $s \in \{0,1\}^d$ and outputs 1 if and only if $\mathrm{Ext}(s,x) \in S$). Then, as shown next, any (n,k)-source is "pseudorandom" with respect to this class of distinguishers, but sufficiently "non-(n,k)-sources" are not "pseudorandom" with respect to this class of distinguishers.

1. For every (n,k)-source X and every $S \subset \{0,1\}^m$, the test T_S does not distinguish X from U_n (i.e., $\mathsf{Pr}[T_S(X) = 1] = \mathsf{Pr}[T_S(U_n) = 1] \pm 2\varepsilon$), because $\mathrm{Ext}(U_d, X)$ is 2ε-close to $\mathrm{Ext}(U_d, U_n)$ (since each is ε-close to U_m).
2. On the other hand, for every $(n, k-d-4)$-flat source Y there exists a set S such that T_S distinguish Y from U_n with gap at least 0.9 (e.g., for S that equals the support of $\mathrm{Ext}(U_d, Y)$, it holds that $\mathsf{Pr}[T_S(Y)=1] = 1$ but $\mathsf{Pr}[T_S(U_n)=1] \leq \mathsf{Pr}[U_m \in S] + \varepsilon = 2^{d+(k-d-4)-m} + \varepsilon < 0.1$). Furthermore, any source that has entropy below $(k/4) - d$ will be detected as defected by this class (with probability at least 2/3).[9]

Thus, this weird class of tests deems each (n,k)-source as "pseudorandom" while deeming sources of significantly lower entropy (e.g., entropy lower than $(k/4) - d$) as non-pseudorandom. Indeed, this perspective stretches the pseudorandomness paradigm quite far.

D.4.2 Constructions

Recall that we seek explicit constructions of extractors; that is, functions $\mathrm{Ext} : \{0,1\}^d \times \{0,1\}^n \to \{0,1\}^m$ that can be computed in polynomial-time. The question, of course, is of parameters; that is, having explicit (k,ε)-extractors *with m as large as possible and d as small as possible*. We first note that, except in "pathological" cases[10], both $m \leq k+d-(2\log_2(1/\varepsilon) - O(1))$ and $d \geq \log_2((n-k)/\varepsilon^2) - O(1)$ must hold, regardless of the explicitness requirement. The aforementioned bounds are in fact tight; that is, there exists (non-explicit) (k,ε)-extractors with $m = k + d - 2\log_2(1/\varepsilon) - O(1)$ and $d = \log_2((n-k)/\varepsilon^2) + O(1)$. The obvious goal is meeting these bounds via explicit constructions.

D.4.2.1 Some known results

Despite tremendous progress on this problem (and occasional claims regarding "optimal" explicit constructions), the ultimate goal was not reached yet. Nevertheless, the known explicit constructions are pretty close to being optimal.

Theorem D.10 (explicit constructions of extractors): *Explicit (k,ε)-extractors of the form $\mathrm{Ext} : \{0,1\}^d \times \{0,1\}^n \to \{0,1\}^m$ exist for the following cases (i.e., settings of the parameters d and m):*

[9] For any such source Y, the distribution $Z = \mathrm{Ext}(U_d, Y)$ has entropy at most $k/4 = m/4$, and thus is 0.7-far from U_m (and 2/3-far from $\mathrm{Ext}(U_d, U_n)$). The lower-bound on the statistical distance between Z and U_m can be proved by the contra-positive: if Z is δ-close to U_m then its entropy is at least $(1-\delta) \cdot m - 1$ (e.g., by using Fano's inequality, see [63, Thm. 2.11.1]).

[10] That is, for $\varepsilon < 1/2$ and $m > d$.

1. *For* $d = O(\log n/\varepsilon)$ *and* $m = (1-\alpha)\cdot(k-O(d))$*, where* $\alpha > 0$ *is an arbitrarily small constant and provided that* $\varepsilon > \exp(-k^{1-\alpha})$*.*

2. *For* $d = (1+\alpha)\cdot\log_2 n$ *and* $m = k/\text{poly}(\log n)$*, where* $\varepsilon, \alpha > 0$ *are arbitrarily small constants.*

Proofs of Part 1 and Part 2 can be found in [113] and [201], respectively. We note that, for sake of simplicity, we did not quote the best possible bounds. Furthermore, we did not mention additional incomparable results (which are relevant for different ranges of parameters).

We refrain from providing an overview of the proof of Theorem D.10, but rather review the proof of a weaker result that provides *explicit* $(n^\gamma, \text{poly}(1/n))$*-extractors for the case of* $d = O(\log n)$ *and* $m = n^{\Omega(\gamma)}$*, where* $\gamma > 0$ *is an arbitrarily small constant.* Indeed, in §D.4.2.2, we review the conceptual insight that underlies this result (as well as much of the subsequent developments in the area).

D.4.2.2 The pseudorandomness connection

We conclude this section with an overview of a fruitful connection between extractors and certain pseudorandom generators. The connection, discovered by Trevisan [222], is surprising in the sense that it goes in a non-standard direction: it transforms certain pseudorandom generators into extractors. As argued throughout this book (most conspicuously at the end of Section 7.1.2), computational objects are typically more complex than the corresponding information theoretical objects. Thus, if pseudorandom generators and extractors are at all related (which was not suspected before [222]) then this relation should not be expected to help in the construction of extractors, which seem an information theoretic object. Nevertheless, the discovery of this relation did yield a breakthrough in the study of extractors.[11]

> **Teaching note:** The current text assumes familiarity with pseudorandom generators and in particular with the Nisan–Wigderson Generator (presented in §8.3.2.1).

But before describing the connection, let us wonder for a moment. Just looking at the syntax, we note that pseudorandom generators have a single input (i.e., the seed), while extractors have two inputs (i.e., the n-bit long source and the d-bit long seed). But taking a second look at the Nisan–Wigderson Generator (i.e., the combination of Construction 8.17 with an amplification of worst-case to average-case hardness), we note that this construction can be viewed as taking two inputs: a d-bit long seed and a "hard" predicate on d'-bit long strings (where $d' = \Omega(d)$).[12] Now, an appealing idea is to use the n-bit long source as a (truth-table) description of a (worse-case) hard predicate (which indeed means setting $n = 2^{d'}$). The key observation is that *even if the source is only weakly random then it is likely to represent a predicate that is hard on the worst-case.*

[11] We note that once the connection became better understood, influence started going in the "right" direction: from extractors to pseudorandom generators.

[12] Indeed, to fit the current context, we have modified some notations. In Construction 8.17 the length of the seed is denoted by k and the length of the input for the predicate is denoted by m.

Recall that the aforementioned construction is supposed to yield a pseudorandom generator whenever it starts with a hard predicate. In the current context, where there are no computational restrictions, pseudorandomness is supposed to hold against any (computationally unbounded) distinguisher, and thus here pseudorandomness means being statistically close to the uniform distribution (on strings of the adequate length, denoted ℓ). Intuitively, this makes sense only if the observed sequence is shorter that the amount of randomness in the source (and seed), which is indeed the case (i.e., $\ell < k + d$, where k denotes the min-entropy of the source). Hence, there is hope to obtain a good extractor this way.

To turn the hope into a reality, we need a proof (which is sketched next). Looking again at the Nisan–Wigderson Generator, we note that the proof of indistinguishability of this generator provides a black-box procedure for computing the underlying predicate when given oracle access to any potential distinguisher. Specifically, in the proofs of Theorems 7.19 and 8.18 (which holds for any $\ell = 2^{\Omega(d')}$)[13], this black-box procedure was implemented by a *relatively small circuit* (which depends on the underlying predicate). Hence, this procedure contains relatively little information (regarding the underlying predicate), on top of the observed ℓ-bit long output of the extractor/generator. Specifically, for some fixed polynomial p, the amount of information encoded in the procedure (and thus available to it) is upper-bound by $b \stackrel{\text{def}}{=} p(\ell)$, while the procedure is suppose to compute the underlying predicate correctly on each input. That is, b bits of information are supposed to fully determine the underlying predicate, which in turn is identical to the n-bit long source. However, if the source has min-entropy exceeding b, then it cannot be fully determine using only b bits of information. It follows that the foregoing construction constitutes a $(b + O(1), 1/6)$-extractor (outputting $\ell = b^{\Omega(1)}$ bits), where the constant $1/6$ is the one used in the proof of Theorem 8.18 (and the argument holds provided that $b = n^{\Omega(1)}$). Note that this extractor uses a seed of length $d = O(d') = O(\log n)$. The argument can be extended to obtain $(k, \text{poly}(1/k))$-extractors that output $k^{\Omega(1)}$ bits using a seed of length $d = O(\log n)$, provided that $k = n^{\Omega(1)}$.

We note that the foregoing description has only referred to two abstract properties of the Nisan–Wigderson Generator: (1) the fact that this generator uses any worst-case hard predicate as a black-box, and (2) the fact that its analysis uses any distinguisher as a black-box. In particular, we viewed the amplification of worst-case hardness to inapproximability (performed in Theorem 7.19) as part of the construction of the pseudorandom generator. An alternative presentation, which is more self-contained, replaces the amplification step of Theorem 7.19 by a direct argument in the current (information theoretic) context and plugs the resulting predicate directly into Construction 8.17. The advantages of this alternative include using a simpler amplification (since amplification is simpler in the information theoretic setting than in the computational setting), and deriving transparent construction and analysis (which mirror Construction 8.17 and Theorem 8.18, respectively).

[13] Recalling that $n = 2^{d'}$, the restriction $\ell = 2^{\Omega(d')}$ implies $\ell = n^{\Omega(1)}$.

The alternative presentation. The foregoing analysis transforms a generic distinguisher into a procedure that computes the underlying predicate correctly on each input, which fully determines this predicate. Hence, an upper-bound on the information available to this procedure yields an upper-bound on the number of possible outcomes of the source that are bad for the extractor. In the alternative presentation, we transforms a generic distinguisher into a procedure that only approximates the underlying predicate; that is, the procedure yields a function that is relatively close to the underlying predicate. If the potential underlying predicates are far apart, then this yields the desired bound (on the number of bad source-outcomes that correspond to such predicates). Thus, the idea is to encode the n-bit long source by an error correcting code of length $n' = \text{poly}(n)$ and relative distance $0.5 - (1/n)^2$, and use the resulting codeword as a truth-table of a predicate for Construction 8.17.[14] Such codes (coupled with efficient encoding algorithms) do exist (see §E.1.2.5), and the benefit in using them is that each n'-bit long string (determined by the information available to the aforementioned approximation procedure) may be $(0.5 - (1/n))$-close to at most $O(n^2)$ codewords[15] (which correspond to potential predicates). Thus, each approximation procedure rules out at most $O(n^2)$ potential predicates (i.e., source outcomes). In summary, *the resulting extractor converts the n-bit input x into a codeword $x' \in \{0,1\}^{n'}$, viewed as a predicate over $\{0,1\}^{d'}$* (where $d' = \log_2 n'$), *and evaluates this predicate at the ℓ projections of the d-bit long seed, where these projections* (to d' bits) *are determined by the corresponding set system* (i.e., the ℓ-long sequence of d'-subsets of $[d]$ that is used in Construction 8.17). The analysis mirrors the proof of Theorem 8.18, and yields a bound of $2^{O(\ell^2)} \cdot O(n^2)$ on the number of bad outcomes for the source, where $O(\ell^2)$ upper-bounds the amount of information encoded in (and available to) the approximation procedure, and $O(n^2)$ upper-bounds the number of source-outcomes that correspond to codewords that are each $(0.5 - (1/n))$-close to any fixed approximation procedure.

D.4.2.3 Recommended reading

The interested reader is referred to a survey of Shaltiel [200]. This survey contains a comprehensive introduction to the area, including an overview of the ideas that underly the various constructions. In particular, the survey describes the approaches used before the discovery of the pseudorandomness connection, the connection itself (and the constructions that arise from it), and the "third generation" of constructions that followed.

The aforementioned survey predates the most recent constructions (of extractors) that extract a constant fraction of the min-entropy using a logarithmically long seed (cf. Part 1 of Theorem D.10). Such constructions were first presented in [159] and improved (using different ideas) in [113]. Indeed, we refer to reader to [113], which provides a self-contained description of the best known extractor (for almost all settings of the relevant parameters).

[14] Indeed, the use of this error correcting code replaces the hardness-amplification step of Theorem 7.19.

[15] See Appendix E.1.4.