BOOK REVIEW: KOLMOGOROV COMPLEXITY AND ALGORITHMIC RANDOMNESS BY ALEXANDER SHEN, VLADIMIR A. USPENSKIĬ, AND NIKOLAY K. VERESHCHAGIN

MATHEMATICAL SURVEYS AND MONOGRAPHS, VOL. 220, AMERICAN MATHEMATICAL SOCIETY, 2017, ISBN 9781470431822, 511 PAGES, $124.00

1. Introduction

As young children, we all quickly develop a rudimentary intuition for gravity. But it is not until we've played games of chance – and perhaps not until we've lost in a rigged game – that we have some inkling of what randomness is. To be fair, one eventually learns in school that gravity has at least as many mysteries as the definition of randomness. But the question remains: What is randomness?

A mathematically satisfying answer can be found in the book under review (henceforth [SUV], from the authors' initials). [SUV] presents a rich account of numerous earlier definitions of randomness, their interconnection with algorithmic complexity, and their applications. Before exploring [SUV] further, let us first see some concrete examples of the boundary between randomness and non-randomness.

2. Adversaries and Pseudorandom Generators

If one makes a yes/no decision via a coin flip, while standing in the middle of a gusting windstorm, then one may be inclined to believe that this decision method is random, and that either outcome is equally likely. On the other hand, if an experienced gambler picks your coin, and tosses it for you in his private casino, then you would be less inclined to believe that the outcome will be random. The key difference between these two coin-tossing scenarios is *predictability*: One can argue that the necessary measurements and calculations to predict the coin flip in the first scenario make prediction completely impractical. On the other hand, making predictable coin tosses in controlled settings has been known long enough in the gambling world that mathematicians have finally written papers on it (see, e.g., [DHM07]). Our two toy scenarios (inspired by a similar discussion in [BM84]) thus suggest that a suitable definition of randomness should include some notion of unpredictability.

Our two toy scenarios also hint at the deep connection between randomness and physics. (See [Rue93] for a beautiful exposition on the connections

Date: June 21, 2019.

2010 *Mathematics Subject Classification.* Primary 68, 03, 60, 97.

Partially supported by NSF grants CCF-1409020 and CCF-1900881, and NSF REU grant DMS-1757872.

between dynamics, physics, and randomness.) Furthermore, it becomes apparent that computational power is a factor behind predicting (or *derandomizing*) a sequence of events that is intended to be random. More to the point, we can use computational complexity to define *pseudorandomness*, and perhaps sidestep the question of how to reliably extract randomness from some physical process.

The algorithmic construction of unpredictable sequences of bits has numerous applications, especially if certain natural efficiency and equidistribution properties are also guaranteed. For instance, in cryptography, such sequences can be used[1] to safely send messages that need to be kept secret from an adversary. Also, in some statistical applications, the only way to practically estimate the mean of a quantity over a large population is to average over a smaller, well-distributed sample. Pseudorandomness attempts to capture the most useful aspects of randomness in an algorithmically efficient way, but its definition is best preceded with an explicit example.

2.1. Pseudorandom Bit Streams from the Discrete Logarithm Problem.

Consider the following famous construction, by Blum and Micali [BM84], of a family of (putatively) unpredictable sequences of bits. In what follows, n is a positive integer, $p \in \{2^{n-1}+1, \ldots, 2^n-1\}$ is an odd prime (so p has exactly n bits in its binary expansion), g is a generator for the multiplicative group $\mathbb{F}_p^*$ of nonzero integers mod p, and $x_0 \in \{0, \ldots, p-2\}$ is called the *seed* for our bit stream. Letting $M_p : \{1, \ldots, p-1\} \longrightarrow \{0, 1\}$ denote the function satisfying $M_p(a) = 1$ if and only if $a \geq (p-1)/2$ (so $M_p(a)$ is akin to the most significant bit of a), we then define a sequence $(x_0, x_1, \ldots, x_N)$ with, say, $N := n^{10}$, via the recurrence $x_{j+1} := g^{x_j} \bmod p$, valid for all $j \geq 0$.[2] Our pseudorandom sequence of bits — an instance of the *Blum-Micali pseudorandom generator (PRG)* — is then

$$B(p, g, x_0) := (M_p(x_N), M_p(x_{N-1}), \ldots, M_p(x_1)).$$

Letting $(b_1, \ldots, b_N) := B(p, g, x_0)$, a remarkable property of the Blum-Micali PRG is that it can be *proved* that b_{i+1} is unpredictable from b_i (and even x_{N-i+1}), provided a well-known number-theoretic problem is hard to solve. To properly define unpredictability and hardness, let us first recall the following problem:

The Discrete Logarithm Problem (DLP) *Given an n-bit prime p, a generator g for the multiplicative group $\mathbb{F}_p^*$, and an integer $h \in \{1, \ldots, p-1\}$, find an $\ell \in \{0, \ldots, p-2\}$ with $g^\ell = h$ mod p.* $\diamond$

We call the ℓ in the definition above the *mod p base-g discrete logarithm of h*, and think of n as a rough measure of the size of an instance (p, g, h) of

[1] via the ancient idea of a *one-time pad*

[2] The length $N = n^{10}$ is somewhat arbitrary: One can in fact use $N = n^k$ for any positive integer k [Blu19], although this was not explicitly stated in [BM84].

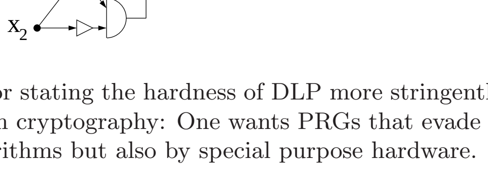
DLP. In what follows, we will use the usual number-theoretic/computer science notations $O(\cdot)$, $o(\cdot)$, $\Omega(\cdot)$, and $\omega(\cdot)$ for asymptotic growth comparisons (see, e.g., [AB09, pp. 3–4]). For instance, if we say that that a function $f:\mathbb{N}\longrightarrow\mathbb{N}$ satisfies $f(n)=n^{\omega(1)}$ then this means that for any real c, there is an $n_0(c)\in\mathbb{N}$ such that $f(n)\geq n^c$ for all $n\geq n_0(c)$, i.e., f ultimately grows faster than any polynomial.

It is easy to compute $h=g^\ell \bmod p$, given (p,g,ℓ) as above, in near-quadratic time: $n^{2+o(1)}$ bit operations suffice (see, e.g., [BS96, pp. 43 & 102–104]). However, computing the discrete logarithm ℓ from (p,g,h) in time $n^{O(1)}$ remains an open problem: The best general complexity bound is $e^{O(n^{1/3}(\log n)^{2/3})}$, via refinements of the index calculus method (see, e.g., [HPS14, Sec. 3.8] and [A+19]), and the underlying algorithm uses randomization. In particular, DLP is conjectured to be hard in the following sense:

DLP Hardness Assumption *For any $k,n\in\mathbb{N}$, let $C_{n,k}$ be any boolean circuit that solves DLP for at least $\frac{1}{n^k}$ of the n-bit primes p. Then the size of $C_{n,k}$ is $n^{\omega(1)}$.* $\diamond$

Basic complexity theory (see, e.g., [AB09, Ch. 6]) tells us that the existence of an algorithm for solving DLP in time polynomial in n would imply that there is a *family* of boolean circuits $\{C_n\}_{n\in\mathbb{N}}$ such that C_n solves DLP for any input (p,g,h), with p an n-bit prime, and C_n has size $n^{O(1)}$. So the DLP Hardness Assumption rules out such a family of circuits, *as well as* polynomial-time algorithms for DLP in the classical Turing model.[3]

To better understand this hardness assumption, recall that a *boolean circuit* C is an acyclic directed graph with nodes (also called *gates*) consisting of one of five types: (0) *input* nodes (having in-degree 0 and finite out-degree), (1) *output* nodes (having in-degree 1 and out-degree 0), (2) a *not* gate (having in-degree and out-degree 1), (3) an *and* gate (having in-degree 2 and out-degree 1, and (4) an *or* gate (having in-degree 2 and out-degree 1). The *size* of C is simply the number of vertices in the underlying graph, and the *number of inputs* is the number of input nodes. Identifying 1 and 0 respectively with "True" and "False", and labelling the input nodes with variables, we then obtain a natural interpretation of boolean circuits as implementations of logical formulae. For instance, the circuit below (with 2 input nodes and size 8) computes the well-known XOR function $(x_1\vee\neg x_2)\wedge(\neg x_1\vee x_2)$:



[3]One motivation for stating the hardness of DLP more stringently, in terms of families of circuits, comes from cryptography: One wants PRGs that evade prediction not just by polynomial-time algorithms but also by special purpose hardware.

Abusing notation slightly, we then say that a function

$$F : \{0, \ldots, 2^n - 1\} \longrightarrow \{0, \ldots, 2^m - 1\}$$

is *computed by a boolean circuit* if and only if there are boolean circuits computing each of the m bits of $F(x)$ from the binary expansion of $x \in \{0, \ldots, 2^n - 1\}$.

The (conditional) unpredictability of the Blum-Micali PRG can then be phrased as follows:

Blum-Micali Theorem [BM84] *Assume the DLP Hardness Assumption is true, and let $\mathcal{C} := \{C_n\}$ be any family of boolean circuits, with $n \in \mathbb{N}$ and C_n having exactly $3n$ inputs and size $n^{O(1)}$. Considering the uniform distribution over the set of all (p, g, h) with p an n-bit prime, g a generator for $\mathbb{F}_p^*$, and $h \in \mathbb{F}_p^*$, let $\rho_n^{\mathcal{C}}$ be the probability that $C_n(p, g, h) = M_p(\ell)$, where $h = g^\ell$ mod p. Then $\rho_n^{\mathcal{C}} < \frac{1}{2} + \frac{1}{n^{\omega(1)}}$.*

In other words, if DLP is hard, then no (classical) polynomial-time algorithm can do much better than coin-tossing to guess the $(i+1)$st bit of an instance of the Blum-Micali PRG, even if one knows p, g, and the element of $\mathbb{F}_p^*$ whose discrete logarithm yielded the ith bit. Curiously, as of June 2019, there is still no known polynomial-time construction that attains unpredictability in the preceding sense *without* unproven hypotheses.

2.2. **A Rigorous Definition of Pseudorandomness.** The Blum-Micali PRG, under the DLP Hardness Assumption, is actually an instance of a more general construction by Blum and Micali [BM84]. Following [BM84], and a refinement from [Nis92a], one can define a more general class of PRGs as follows:

Definition 2.1. [4] *A* pseudorandom generator (PRG) for class Γ *is a family of functions $G := \left\{G_n : \{0,1\}^n \longrightarrow \{0,1\}^{Q(n)}\right\}_{n \in \mathbb{N}}$ such that $Q : \mathbb{N} \longrightarrow \mathbb{N}$ is an increasing function and, writing $(y_1, \ldots, y_{Q(n)}) := G_n(x_1, \ldots, x_n)$, satisfies the following property: For each $i \in \{0, \ldots, Q(n)-1\}$ and every algorithm A in Γ, we have:*

$$\left| \operatorname*{Prob}_{(x_1,\ldots,x_n)\in\{0,1\}^n} [A(y_1, \ldots, y_i) = y_{i+1}] - \frac{1}{2} \right| = \frac{1}{n^{\omega(1)}}. \qquad \diamond$$

Remark 2.1. *The function $\frac{Q(n)}{n}$ (which is bounded from below by 1, thanks to our definition) is sometimes called the* stretch-factor *of the PRG and is a useful measure of how the PRG* amplifies *randomness, i.e., simulates a long stream of uniformly random bits from just a small number of "truly" uniformly random bits. Turning this around, we can also view a PRG as a means of* compressing *the sequence $(y_1, \ldots, y_{Q(n)})$ into the shorter representation $(x_1, \ldots, x_n)$. This leads to the notion of* Kolmogorov Complexity, *which is a central theme in the book under review.* $\diamond$

[4]To be more precise, the definition from [BM84] took $\Gamma = \mathbf{P}_{/\text{poly}}$ and assumed that each function G_n is computable in time polynomial in n. Here we make no restriction on the complexity of computing the G_n.

The *existence* of PRGs (for the complexity classes $\mathbf{P}_{/\text{poly}}$ or $\mathbf{P}$) with polynomial stretch-factor can be proved quite easily. Technically, this is a rephrasing of the classical containment of complexity classes $\mathbf{BPP} \subseteq \mathbf{P}_{/\text{poly}}$ (see, e.g., [Adl78, BG81] and [AB09, Sec. 7.5]). Much how probabilistic methods yield non-constructive proofs in combinatorics, there are still no proven *explicit* constructions of such PRGs, unless one uses unproved hypotheses (like our earlier Blum-Micali example). The subtlety of finding explicit PRGs is also revealed by how the famous $\mathbf{P} \overset{?}{=} \mathbf{BPP}$ question from complexity theory [IW97, IK04] can be reformulated as the construction of a PRG for $\mathbf{P}$ with exponential stretch-factor (and the complexity of G_n, say, exponential in n) [Gol11]. One should also be aware of the fact that the existence of PRGs for $\mathbf{P}$, with stretch-factor $n^{\omega(1)}$ and G_n computable in time polynomial in n, implies an even more famous conjecture: $\mathbf{P} \neq \mathbf{NP}$ (see, e.g., [Vad11, Sec. 7.2]).

In closing our discussion of pseudorandomness, we point out that there has been important recent progress on explicit constructions of PRGs (for complexity classes *lower* than $\mathbf{P}$) with exponential stretch: see, e.g., [Nis92b, MZ13, GKM18]. Further background on pseudorandomness can be found in [Yao82, Kab04, Vad11].

3. The Book of Shen, Uspensky, and Vereshchagin

3.1. **Two Initial Definitions for Randomness.**

Since *pseudo*random bitstreams can be defined by unpredictability against an adversary with computational power determined by a complexity class Γ, why not try to define randomness by letting $\Gamma = \mathbf{RE}$?[5] Alternatively, why not define randomness in terms of equidistribution of the 0s and 1s within subsequences? These two approaches, which we'll respectively call $\mathbf{U}$ and $\mathbf{S}$, for *unpredictability* and *stochasticity*, correspond to early to mid-20th century approaches detailed in [SUV].

Stochasticity, which has roots in 1919 work of von Mises, turns out to be the looser notion. In particular, as detailed in Appendix 2 of [SUV], one can show that approach $\mathbf{S}$ strictly includes the sequences defined by approach $\mathbf{U}$. However, a complication with approach $\mathbf{S}$ is defining the kinds of subsequences one should examine for limiting frequencies. For instance, while the 1s in the sequence $0101010101010101010101\ldots$ appear to have a limiting frequency of $\frac{1}{2}$, the limiting frequency of 1s for the entries of *even* index is 1. (That this sequence also does not "look random" to most mathematicians is more than a coincidence.) More to the point, what kinds of increasing functions $f : \mathbb{N} \longrightarrow \mathbb{N}$ should we consider when we examine the sequence $(b_i)_{i \in \mathbb{N}}$ and insist that

[5]**RE** is the class of decision problems where a "Yes" answer can be verified by a Turing machine in finite time. (So the machine might never halt if the answer is "No.") In other words, a PRG for class RE has the property that even unlimited computational time makes the next bit no more predictable than a fair coin-toss.

$$\lim_{n\to\infty} \frac{|\{i \mid b_{f(i)}=1\ ,\ i\in\{1,\ldots,n\}\}|}{n} = \frac{1}{2}?$$

Church (in 1940) and Kolmogorov (in 1963) proposed formalizations that led to separate definitions of **S**, with Kolmogorov's definition strictly more general.

Approach **U** can be defined via a gambling metaphor, reminiscent of our description of pseudorandomness through "fooling" adversaries: Imagine a bit-stream $(b_i)_{i\in\mathbb{N}}$ corresponding to a sequence of coin-flips in a casino, where a player can bet v_i dollars just before seeing the value of b_i. After the ith coin-flip, a player receives v_i dollars if he/she guesses b_i correctly, and loses v_i dollars if he/she errs. If there is *no* strategy that allows one to start with 1 dollar and approach infinite winnings as $i \longrightarrow \infty$, then we declare the sequence to be unpredictable. The notion of strategy is also formalized in Appendix 2 of [SUV] and, in Chapter 9 of [SUV], one sees that approach **U** is known as *Kolmogorov-Loveland randomness* (which goes back to 1963–1966).

3.2. **Kolmogorov Complexity.** Approaches **S** and **U** have their appeal, but the most influential approach to randomness — independently discovered by Solomonoff, Kolmogorov, and Chaitin in the 1960s — is based on *optimal compression.* More precisely, just as pseudorandomness stretches short sequences of "truly" random bits into long sequences that "look random" to observers (and algorithms) with low computational power, randomness for a *finite* bit-stream can be measured by its *lack of compressibility.*[6] For instance, the trait common to the following 3 examples of bit-streams of length $41,943,042$:

$$\begin{array}{l} 000000000000000000000\cdots 00 \\ 101010101010101010101 0\cdots 10 \\ 0110111001011101111000\cdots 11 \end{array}$$

is that each can be easily generated by a very short `C++` program.[7] (We will soon see that the underlying language will not matter.)

Now suppose we encode programs into bit-streams (assuming one fixes the underlying encoding of programs and the underlying programming language), and we try to *compress* our sequence $(b_i)_{i\in\{1,\ldots,n\}}$ by replacing it with a program *of minimal length* that generates $(b_i)_{i\in\{1,\ldots,n\}}$. One can then consider the finite sequence $(b_i)_{i\in\mathbb{N}}$ to be far from random if and only if it is not compressible in the preceding sense. The Linux command `gzip` is an example of this kind of compression: One can check on any recent version of Linux that the first two examples can be compressed by a factor of

[6]One must be careful: Strictly speaking, randomness for a sequence of bits only makes sense for *infinite* sequences. What one can do with finite sequences is define quantities that lead to suitable definitions of randomness as the sequence length tends to infinity.

[7]The last sequence is simply the concatenation of the binary expansions of 0, 1, 2, 3, ..., $2^{21}-1$.

around one million, thus suggesting that our first two examples are far from random.[8]

Needless to say, our preceding description is far from rigorous. However, the key insight derived independently by Solomonoff, Kolmogorov, and Chaitin is that one *can* formalize the notion of compressibility, and do so in an *invariant* way. More precisely, consider any partial computable function $D : \{0,1\}^* \longrightarrow \{0,1\}^*$, where $\{0,1\}^*$ denotes the set of all finite binary strings, and *partial computable* means that D is computable by a Turing machine that may not terminate on certain inputs. The *Kolmogorov complexity* of an $x \in \{0,1\}^*$, with respect to the *decompressor* D, is then defined to be

$$C_D(x) := \min\{L(y) \mid D(y) = x \ , \ y \in \{0,1\}^*\},$$

where $L(y)$ denotes the length of the string $y \in \{0,1\}^*$. In particular, Solmonoff and Kolmogorov proved independently that there are *optimal* decompressors D in the sense that for all $x \in \{0,1\}^*$, and all decompressors D', we have $C_D(x) \leq C_{D'}(x) + O(1)$. In other words, we can fix some optimal decompressor D once and for all, and write Kolmogorov complexity without mentioning any decompressor, by setting $C(x) := C_D(x)$.

A surprising aspect of Kolmogorov complexity is that it is simultaneously hard to compute but possesses numerous natural functorial properties. For example, while it is easy to show that $C(x) \leq n + O(1)$ for all $x \in \{0,1\}^n$, the length of the *shortest* bit-stream x with $C(x) = n$ is a Turing-uncomputable function of n (see Theorem 15 of [SUV]). On the other hand, one can naturally conjecture that Kolmogorov complexity is sub-additive with respect to concatenation of strings. And, indeed, writing xy for the concatenation of two bit-streams x and y, one has a property close to sub-additivity: $C(xy) \leq C(x) + C(y) + 2\log_2(C(x)) + O(1)$ (see Theorem 4 of [SUV]).

More importantly, Kolmogorov complexity leads us to new perspectives not just on randomness but also on information theory, algorithmic complexity, and probability theory. Indeed, Chapter 8 is a tasty dessert in the middle of [SUV] highlighting important applications of Kolmogorov complexity in several other areas of mathematics.

3.3. **Final Comments.** The account of Kolmogorov Complexity and randomness in [SUV] is masterful. Perhaps the first question that comes to mind, if deciding whether to purchase [SUV], is how it compares to the classic text by Li and Vitányi [LV08]. My answer would be to buy both books: While [LV08] has the advantage of additional polishing (having gone through 3 editions in 14 years), [SUV] maintains amazing clarity while geting quickly to the heart of almost everything one needs from Kolmogorov complexity and its variants. The exercises in [SUV] are also elegantly designed, well accented by hints, and nicely amplify the development.

[8]The same can be said for our third example, and we leave the precise compression factor as an exercise.

Acknowledgements

I am indebted to Manuel Blum, Qi Cheng, Eviatar Procaccia, Salil Vadhan, and David Zuckerman for their patience with, and answers to, my many naive questions.

References

[Adl78] Leonard M. Adleman, *"Two theorems on random polynomial time,"* Proceedings of the Nineteenth Annual IEEE Symposium on Foundations of Computer Science, pp. 75–83, 1978.

[A+19] David Adrian, Karthikeyan Bhargavan, Zakir Durumeric, Pierrick Gaudry, Matthew Green, J. Alex Halderman, Nadia Heninger, Drew Springall, Emmanuel Thomé, Luke Valenta, Benjamin VanderSloot, Eric Wustrow, Santiago Zanella-Béguelin, Paul Zimmermann, *"Imperfect Forward Secrecy: How Diffie-Hellman Fails in Practice,"* Communications of the ACM, January 2019, Vol. 62, No. 1, pp. 106–114. doi 10.1145/3292035

[AB09] Sanjeev Arora and Boaz Barak, *Computational complexity. A modern approach.* Cambridge University Press, Cambridge, 2009.

[BS96] Eric Bach and Jeff Shallit, *Algorithmic Number Theory, Vol. I: Efficient Algorithms,* MIT Press, Cambridge, MA, 1996.

[BG81] Charles H. Bennett and John Gill, *"Relative to a Random Oracle A,* $\mathbf{P}^A \neq \mathbf{NP}^A \neq \mathbf{coNP}^A$ *with Probability* 1*,"* SIAM J. Comput., Vol. 10, No. 1, February 1981.

[Blu19] Manuel Blum, *e-mail communication*, March 23, 2019.

[BM84] Manuel Blum and Silvio Micali, *"How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits,"* SIAM J. Comput., Vol. 13, No. 4, Nov. 1984.

[DHM07] Persi Diaconis, Susan Holmes, and Richard Montgomery, *"Dynamical Bias in the Coin Toss,"* SIAM Review, Vol. 49, No. 2 (Jun., 2007), pp. 211–235.

[Gol11] Oded Goldreich, *"In a World of* $\mathbf{P} = \mathbf{BPP}$*,"* in Studies in Complexity and Cryptography, Miscellanea on the Interplay between Randomness and Computation, (Oded Goldreich, et. al., eds.), Lectures in Computer Science 6650, pp. 191–232, Springer, 2011.

[GKM18] Parikshit Gopalan, Daniel M. Kane, and Raghu Meka, *"Pseudorandomness via the Discrete Fourier Transform,"* SIAM J. Comput., Vol. 47, No. 6, pp. 2451–2487 (2018).

[HPS14] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman, *An Introduction to Mathematical Cryptography,* Undergraduate Texts in Mathematics, Springer-Verlag, 2014.

[IK04] Russell Impagliazzo and Valentine Kabanets, *"Derandomizing Polynomial Identity Tests Means Proving Circuit Lower Bounds,"* Computational Complexity, 13(1-2), pp. 1–6, 2004.

[IW97] Russell Impagliazzo and Avi Wigderson, "$\mathbf{P} = \mathbf{BPP}$ *unless E has Subexponential Circuits: Derandomizing the XOR Lemma,"* Proceedings of the 29th STOC, pp. 220–229, 1997.

[Kab04] Valentine Kabanets, *"Derandomization: A brief overview,"* in Current Trends in Theoretical Computer Science: The Challenge of the New Century, Vol 1: Algorithms and Complexity, edited by G. Paun, G. Rosenberg, and A. Salomaa, World Scientific Publishing Company, 2004.

[LV08] Ming Li and Paul Vitányi, *An Introduction to Kolmogorov Complexity and Its Applications,* third edtion, Texts in Computer Science, Springer, 2008.

[MZ13] Raghu Meka and David Zuckerman, *"Pseudorandom Generators for Polynomial Threshold Functions,"* SIAM J. Comput. Vol. 42, No. 3, pp. 1275–1301 (2013).

[Nis92a] Noam Nisan, *Using Hard Problems to Create Pseudorandom Generators,* ACM Doctoral Dissertation Award series, MIT Press, 1992.

[Nis92b] Noam Nisan, *"Pseudorandom Generators for space-bounded computation,"* Combinatorica, 12 (1992), pp. 449–461.

[Rue93] David Ruelle, *Chance and Chaos,* Princeton University Press, 1993.

[Vad11] Salil P. Vadhan, *Pseudorandomness,* Foundations and Trends in Theoretical Computer Science (2011), 7(1-3). http://dx.doi.org/10.1561/0400000010 .

[Yao82] Andrew C. Yao, *"Theory and Applications of Trapdoor Functions,"* in proceedings of FOCS (IEEE Symposium on the Foundations of Computer Science), pp. 80–91, 1982.

J. Maurice Rojas, Texas A&M University, TAMU 3368, College Station, TX 77843-3368

Email address: `rojas@math.tamu.edu`