COMPUTABILITY AND RANDOMNESS

ROD DOWNEY AND DENIS R. HIRSCHFELDT

1. Historical roots

1.1. Von Mises.

Around 1930, Kolmogorov and others founded the theory of probability, basing it on measure theory. Probability theory is concerned with the distribution of outcomes in sample spaces. It does not seek to give any meaning to the notion of an individual object, such as a single real number or binary string, being random, but rather studies the expected values of random variables. How could a binary string representing a sequence of n coin tosses be random, when all strings of length n have the same probability of 2^{-n} for a fair coin?

Less well known than the work of Kolmogorov are early attempts to answer this kind of question by providing notions of randomness for individual objects. The modern theory of *algorithmic randomness* realizes this goal. One way to develop this theory is based on the idea that an object is random if it passes all relevant "randomness tests". For example, by the law of large numbers, for a random real X, we would expect the number of 1's in the binary expansion of X to have limiting frequency $\frac{1}{2}$. (That is, writing $X(j)$ for the jth bit of this expansion, we would expect to have $\lim_{n\to\infty} \frac{|\{j<n : X(j)=1\}|}{n} = \frac{1}{2}$.) Indeed, we would expect X to be *normal* to base 2, meaning that for any binary string σ of length k, the occurrences of σ in the binary expansion of X should have limiting frequency 2^{-k}. Since base representation should not affect randomness, we would expect X to be normal in this sense no matter what base it were written in, so that in base b the limiting frequency would be b^{-k} for a string σ of length k. Thus X should be what is known as *absolutely normal.*

The idea of normality, which goes back to Borel (1909), was extended by von Mises (1919), who suggested the following definition of randomness for individual binary sequences.[1] A *selection function* is an increasing function $f : \mathbb{N} \to \mathbb{N}$. We think of $f(i)$ as the ith place selected in forming a subsequence of a given sequence. (For the definition of normality above, where we consider the entire sequence, $f(i) = i$.) Von Mises suggested that a sequence $a_0a_1\ldots$ should be random if any selected subsequence $a_{f(0)}a_{f(1)}\ldots$ is normal.

There is, of course, an obvious problem with this approach. For any sequence X with infinitely many 1's we could let f select the positions where 1's occur, and X would fail the test determined by f. However, it does not seem reasonable to be able to choose the testing places *after* selecting an X. The question is then: What kinds of selection functions should be allowed, to capture the intuition that

Downey wishes to thank the Marsden Fund of New Zealand. Hirschfeldt is partially supported by NSF Grant DMS-1600543.

[1] Due to space limitations, we omit historical citations and those found in the books Downey and Hirschfeldt [9], Li and Vitányi [18], or Nies [24] from the list of references. In some sections below, we also cite secondary sources where additional references can be found.

we ought not to be able to sample from a random sequence and get the wrong frequencies? It is reasonable to regard prediction as a computational process, and hence restrict ourselves to *computable* selection functions. Indeed, this suggestion was eventually made by Church (1940), though von Mises' work predates the definition of computable function, so he did not have a good way to make his definition mathematically precise.

As we will see, von Mises' approach had a more significant flaw, but we can build on its fundamental idea: Imagine that we are judges deciding whether a sequence X should count as random. If X passes all tests we can (in principle) devise given our computational power, then we should regard X as random since, as far as we are concerned, X has all the expected properties of a random object. We will use this intuition and the apparatus of computability and complexity theory to describe notions of *algorithmic* randomness.

Aside from the intrinsic interest of such an approach, it leads to useful mathematical tools. Many processes in mathematics are computable, and the expected behavior of such a process should align itself with the behavior obtained by providing it with an algorithmically random input. Hence, instead of having to analyze the relevant distribution and its statistics, we can simply argue about the behavior of the process on a single input. For instance, the expected number of steps of a sorting algorithm should be the same as that for a single algorithmically random input. We could also be more fine-grained and seek to understand exactly "how much" randomness is needed for certain typical behaviors to arise. (See Section 4.)

As we will discuss, algorithmic randomness also goes hand in hand with other parts of algorithmic information theory, such as Kolmogorov complexity, and has ties with notions such as Shannon entropy and fractal dimension.

1.2. **Some basic computability theory.** In the 1930's, Church, Gödel, Kleene, Post, and most famously Turing (1937) gave equivalent mathematical definitions capturing the intuitive notion of a computable function, leading to the *Church-Turing Thesis*, which can be taken as asserting that a function (from $\mathbb{N}$ to $\mathbb{N}$, say) is computable if and only if it can be computed by a Turing machine. (This definition can easily be transferred to other objects of countable mathematics. For instance, we think of infinite binary sequences as functions $\mathbb{N} \to \{0,1\}$, and identify sets of natural numbers with their characteristic functions.) Nowadays, we can equivalently regard a function as computable if we can write code to compute it in any given general-purpose programming language (assuming the language can address unlimited memory). It has also become clear that algorithms can be treated as data, and hence that there is a *universal Turing machine*, i.e., there are a listing $\Phi_0, \Phi_1, \dots$ of all Turing machines and a single algorithm that, on input $\langle e, n\rangle$ computes the result $\Phi_e(n)$ of running Φ_e on input n.[2]

[2] The realization that such universal machines are possible helped lead to the development of modern computers. Previously, machines had been purpose-built for given tasks. In a 1947 lecture on his design for the Automated Computing Engine, Turing said, "The special machine may be called the universal machine; it works in the following quite simple manner. When we have decided what machine we wish to imitate we punch a description of it on the tape of the universal machine ... The universal machine has only to keep looking at this description in order to find out what it should do at each stage. Thus the complexity of the machine to be imitated is concentrated in the tape and does not appear in the universal machine proper in any way. ... [D]igital computing machines such as the ACE ... are in fact practical versions of the universal

It is important to note that a Turing machine might not halt on a given input, and hence the functions computed by Turing machines are in general *partial.* Indeed, as Turing showed, the *halting problem* "Does the eth Turing machine halt on input n?" is algorithmically unsolvable. Church and Turing famously showed that Hilbert's *Entscheidungsproblem* (the decision problem for first-order logic) is unsolvable, in Turing's case by showing that the halting problem can be coded into first-order logic. Many other problems have since been shown to be algorithmically unsolvable by similar means.

We write $\Phi_e(n)\downarrow$ to mean that the machine Φ_e eventually halts on input n. Then $\emptyset' = \{\langle e, n\rangle : \Phi_e(n)\downarrow\}$ is a set representing the halting problem. This set is an example of a noncomputable *computably enumerable* (*c.e.*) set, which means that the set can be listed (not necessarily in numerical order) by some algorithm.

Another important notion is that of *Turing reducibility* (which we define for sets of natural numbers but is similarly defined for functions), where A is Turing reducible to B, written as $A \leqslant_{\mathrm{T}} B$, if there is an algorithm for computing A when given access to B. That is, the algorithm is allowed access to answers to questions of the form "Is n in B?" during its execution. This notion can be formalized using Turing machines with oracle tapes. If $A \leqslant_{\mathrm{T}} B$, then we regard A as no more complicated than B from a computability-theoretic perspective. We also say that A is *B-computable* or *computable relative to B*. Turing reducibility naturally leads to an equivalence relation, where A and B are *Turing equivalent* if $A \leqslant_{\mathrm{T}} B$ and $B \leqslant_{\mathrm{T}} A$. The *(Turing) degree* of A is its equivalence class under this notion. (There are several other notions of reducibility and resulting degree structures in computability theory, but Turing reducibility is the central one.)

In general, the process of allowing access to an oracle in our algorithms is known as *relativization.* As in the unrelativized case, we can list the Turing machines $\Phi_0^B, \Phi_1^B, \dots$ with oracle B, and let $B' = \{\langle e, n\rangle : \Phi_e^B(n)\downarrow\}$ be the relativization of the halting problem to B. This set is called the *(Turing) jump* of B. The jump operation taking B to B' is very important in computability theory, one reason being that B' is the most complicated set that is still c.e. relative to B, i.e., B' is c.e. relative to B and every set that is c.e. relative to B is B'-computable. There are several other important classes of sets that can be defined in terms of the jump. For instance, A is *low* if $A' \leqslant_{\mathrm{T}} \emptyset'$ and *high* if $\emptyset'' \leqslant_{\mathrm{T}} A'$ (where $\emptyset'' = (\emptyset')'$). Low sets are in certain ways "close to computable", while high ones partake of some of the power of $\emptyset'$ as an oracle. These properties are invariant under Turing equivalence, and hence are also properties of Turing degrees.

1.3. **Martin-Löf randomness.** As mentioned above, Church suggested that a sequence should count as algorithmically random if it is random in the sense of von Mises with selection functions restricted to the computable ones. However, in 1939, Ville showed that von Mises' approach cannot work in its original form, no matter what countable collection of selection functions we choose. Let $X \upharpoonright n$ denote the first n bits of the binary sequence X.

Theorem 1.1 (Ville (1939))**.** *For any countable collection of selection functions, there is a sequence X that passes all von Mises tests associated with these functions, such that for every n, there are more 0's than 1's in $X \upharpoonright n$.*

machine." From our contemporary point of view, it may be difficult to imagine how novel this idea was.

Clearly, Ville's sequence cannot be regarded as random in any reasonable sense.

We could try to repair von Mises' definition by adding further tests, reflecting statistical laws beyond the law of large numbers. But which ones? Ville suggested ones reflecting the law of iterated logarithms, which would take care of his specific example. But how could we know that further examples along these lines—i.e., sequences satisfying both von Mises' and Ville's tests, yet failing to have some other property we expect of random sequences—would not arise?

The situation was finally clarified in the 1960's by Martin-Löf (1966). In probability theory, "typicality" is quantified using measure theory, leading to the intuition that random objects should avoid null sets. Martin-Löf noticed that tests like von Mises' and Ville's can be thought of as *effectively* null sets. His idea was that, instead of considering specific tests based on particular statistical laws, we should consider *all* possible tests corresponding to some precisely defined notion of effectively null set. The restriction to such a notion gets around the problem that no sequence can avoid being in *every* null set.

To give Martin-Löf's definition, we work for convenience in Cantor space 2^ω, whose elements are infinite binary sequences. (We associate a real number with its binary expansion, thought of as a sequence, so we will also obtain a definition of algorithmic randomness for reals. The choice of base is not important. For example, all of the notions of randomness we consider are enough to ensure absolute normality.) The basic open sets of Cantor space are the ones of the form $[\sigma] = \{X \in 2^\omega : X \text{ extends } \sigma\}$ for $\sigma \in 2^{<\omega}$, where $2^{<\omega}$ is the set of finite binary strings. The uniform measure λ on this space is obtained by defining $\lambda([\sigma]) = 2^{-|\sigma|}$. We say that a sequence $T_0, T_1, \ldots$ of open sets in 2^ω is *uniformly c.e.* if there is a c.e. set $G \subseteq \mathbb{N} \times 2^{<\omega}$ such that $T_n = \bigcup\{[\sigma] : (n, \sigma) \in G\}$.

Definition 1.2. A *Martin-Löf test* is a sequence $T_0, T_1, \ldots$ of uniformly c.e. open sets such that $\lambda(T_n) \leqslant 2^{-n}$. A sequence X *passes* this test if $X \notin \bigcap_n T_n$. A sequence is *Martin-Löf random* (*ML-random*) if it passes all Martin-Löf tests.

The intersection of a Martin-Löf test is our notion of effectively null set. Since there are only countably many Martin-Löf tests, and each determines a null set in the classical sense, the collection of ML-random sequences has measure 1. It can be shown that Martin-Löf tests include all the ones proposed by von Mises and Ville, in Church's computability-theoretic versions. Indeed they include all tests that are "computably performable", which avoids the problem of having to adaptively introduce more tests as more Ville-like sequences are found.

Martin-Löf's effectivization of measure theory allowed him to consider the laws a random sequence should obey from an abstract point of view, leading to a mathematically robust definition. As Jack Lutz said in a talk at the *7th Conference on Computability, Complexity, and Randomness* (Cambridge, 2012), "Placing computability constraints on a nonconstructive theory like Lebesgue measure seems *a priori* to weaken the theory, but it may strengthen the theory for some purposes. This vision is crucial for present-day investigations of individual random sequences, dimensions of individual sequences, measure and category in complexity classes, etc."

1.4. **The three approaches.** ML-randomness can be thought of as the *statistician's approach* to defining algorithmic randomness, based on the intuition that random sequences should avoid having statistically rare properties. There are two other major approaches:

- The *gambler's approach*: random sequences should be unpredictable.
- The *coder's approach*: random sequences should not have regularities that allow us to compress the information they contain.

The gambler's approach may be the most immediately intuitive one to the average person. It was formalized in the computability-theoretic setting by Schnorr (1971), using the idea that we should not be able to make arbitrarily much money when betting on the bits of a random sequence. The following notion is a simple special case of the notion of martingale from probability theory. (See [9, Section 6.3.4] for further discussion of the relationship between these concepts.)

Definition 1.3. A *martingale* is a function $f : 2^{<\omega} \to \mathbb{R}^{\geqslant 0}$ such that

$$f(\sigma) = \frac{f(\sigma 0) + f(\sigma 1)}{2}$$

for all σ. We say that f *succeeds* on X if $\limsup_{n\to\infty} f(X \upharpoonright n) = \infty$.

We think of f as the capital we have when betting on the bits of a binary sequence according to a particular betting strategy. The displayed equation ensures that the betting is fair. Success then means that we can make arbitrarily much money when betting on X, which should not happen if X is random. By considering martingales with varying levels of effectivity, we get various notions of algorithmic randomness, including ML-randomness itself, as it turns out.

For example, X is *computably random* if no computable martingale succeeds on it, and *polynomial-time random* if no polynomial-time computable martingale succeeds on it. (For the purposes of defining these notions we can think of the martingale as rational-valued.) Schnorr (1971) showed that X is ML-random iff no left-c.e. martingale succeeds on it, where a function $f : 2^{<\omega} \to \mathbb{R}^{\geqslant 0}$ is *left-c.e.* if it is computably approximable from below, i.e., there is a computable function $g : 2^{\omega} \times \mathbb{N} \to \mathbb{Q}^{\geqslant 0}$ such that $g(\sigma, n) \leqslant g(\sigma, n+1)$ for all σ and n, and $f(\sigma) = \lim_{n\to\infty} g(\sigma, n)$ for all σ. One way to think of a left-c.e. martingale is that initially we might have no idea what to bet on some string σ, but as we learn more about the universe, we might discover that σ seems more unlikely to be an initial segment of a random sequence, and are then prepared to bet more of our capital on it.

The coder's approach builds on the idea that a random string should have no short descriptions. For example, in describing 010101 . . . (1000 times) by the brief description "print 01 1000 times", we are using regularities in this string to compress it. For a more complicated string, say the first 2000 bits of the binary expansion of e^{π}, the regularities may be harder to perceive, but are still there and can still lead to compression. A random string should have no such exploitable regularities (i.e., regularities that are not present in most strings), so the shortest way to describe it should be basically to write it out in full. This idea can be formalized using the well-known concept of Kolmogorov complexity. We can think of a Turing machine M with inputs and outputs in $2^{<\omega}$ as a description system. If $M(\tau) = \sigma$ then τ is a description of σ relative to this description system. The *Kolmogorov complexity* $C_M(\sigma)$ of σ relative to M is the length of the shortest τ such that $M(\tau) = \sigma$. We can then take a universal Turing machine U, which emulates any given Turing machine with at most a constant increase in the size of programs, and define the *(plain) Kolmogorov complexity* of σ as $C(\sigma) = C_U(\sigma)$. The value of $C(\sigma)$ depends on U, but only up to an additive constant independent of σ. We think of a string as random if its Kolmogorov complexity is close to its length.

For an infinite sequence X, a natural guess would be that X should be considered random if every initial segment of X is incompressible in this sense, i.e., if $C(X \upharpoonright n) \geqslant n - O(1)$. However, plain Kolmogorov complexity is not quite the right notion here, because the information in a description τ consists not only of the bits of τ, but also its length, which can provide another $\log_2 |\tau|$ many bits of information. Indeed, Martin-Löf (see [18]) showed that it is not possible to have $C(X \upharpoonright n) \geqslant n - O(1)$: Given a long string ρ, we can write $\rho = \sigma\tau\nu$, where $|\tau|$ is the position of σ in the length-lexicographic ordering of $2^{<\omega}$. Consider the Turing machine M that, on input η, determines the $|\eta|$th string ξ in the length-lexicographic ordering of $2^{<\omega}$ and outputs $\xi\eta$. Then $N(\tau) = \sigma\tau$. For any sequence X and any k, this process allows us to compress some initial segment of X by more than k many bits.

There are several ways to get around this problem by modifying the definition of Kolmogorov complexity. The best-known one is to use prefix-free codes, that is, to restrict ourselves to machines M such that if $M(\tau)$ is defined (i.e., if the machine eventually halts on input τ) and μ is a proper extension of τ, then $M(\mu)$ is not defined. There are universal prefix-free machines, and we can take such a machine U and define the *prefix-free Kolmogorov complexity* of σ as $K(\sigma) = C_U(\sigma)$. The roots of this notion be found in the work of Levin, Chaitin, and Schnorr, and in a certain sense—like the notion of Kolmogorov complexity more generally—even earlier in that of Solomonoff (see [9, 18]). As shown by Schnorr (see Chaitin (1975)), it is indeed the case that X is Martin-Löf random if and only if $K(X \upharpoonright n) \geqslant n - O(1)$.

There are other varieties of Kolmogorov complexity, but C and K are the main ones. For applications, it often does not matter which variety is used. The following surprising result establishes a fairly precise relationship between C and K. Let $C^{(1)}(\sigma) = C(\sigma)$ and $C^{(n+1)}(\sigma) = C(C^{(n)}(\sigma))$.

Theorem 1.4 (Solovay (1975)). *$K(\sigma) = C(\sigma) + C^{(2)}(\sigma) \pm O(C^{(3)}(\sigma))$, and this result is tight in that we cannot extend it to $C^{(4)}(\sigma)$.*

There is a vast body of research on Kolmogorov complexity and its applications. We will discuss some of these applications below; much more on the topic can be found in Li and Vitányi [18].

2. Goals

There are several ways to explore the ideas introduced above. First, there are natural internal questions, such as: How do the various levels of algorithmic randomness interrelate? How do calibrations of randomness relate to the hierarchies of computability and complexity theory, and to relative computability? How should we calibrate partial randomness? Can a source of partial (algorithmic) randomness be amplified into a source that is fully random, or at least more random? The books Downey and Hirschfeldt [9] and Nies [24] cover material along these lines up to about 2010.

We can also consider applications. Mathematics has many theorems that involve "almost everywhere" behavior. Natural examples come from ergodic theory, analysis, geometric measure theory, and even combinatorics. Behavior that occurs almost everywhere should occur at sufficiently random points. Using notions from algorithmic randomness, we can explore exactly *how much* randomness is needed in a given case. For example, the set of reals at which an increasing function is differentiable is null. How complicated is this null set, and hence, what level of algorithmic randomness is necessary for a real to avoid it (assuming the function is itself computable in some sense)? Is Martin-Löf randomness the right notion here?

We can also use the idea of assigning levels of randomness to individual objects to prove new theorems or give simpler proofs of known ones. Early examples of this method tended to use Kolmogorov complexity and what is called the "incompressibility method". For instance, Chaitin (1971) (see also [17]) famously used Kolmogorov complexity to give a proof of a version of Gödel's First Incompleteness Theorem, by showing that for any sufficiently strong, computably axiomatizable, consistent theory T, there is a number c such that T cannot prove that $C(\sigma) > c$ for any given string σ (which also follows by interpreting an earlier result of Barzdins; see [18, Section 2.7]). More recently, Kritchman and Raz [17] used these methods to give a proof of the Second Incompleteness Theorem as well.[3] As we will see below, a more recent line of research has used notions of effective dimension based on partial randomness to give new proofs of classical theorems in ergodic theory and obtain new results in geometric measure theory.

3. Some interactions with computability

3.1. **Halting probabilities.** A first question we might ask is how to generate "natural" examples of algorithmically random reals. A classic example is Chaitin's halting probability. Let U be a universal prefix-free machine and let

$$\Omega = \sum_{U(\sigma)\downarrow} 2^{-|\sigma|}.$$

This number is the measure of the set of sequences X such that U halts on some initial segment of X, which we can interpret as the halting probability of U, and was shown by Chaitin (1975) to be ML-random (where, as mentioned above, we identify Ω with its binary expansion, thought of as an infinite binary sequence).

For any prefix-free machine M in place of U we can similarly define a halting probability. In some ways, halting probabilities are the analogs of computably enumerable sets in the theory of algorithmic randomness. Every halting probability α is a *left-c.e. real*, meaning that there is a computable increasing sequence of rationals converging to it. Calude, Hertling, Khoussainov, and Wang (1998) showed that every left-c.e. real is the halting probability of some prefix-free machine.

We should perhaps write Ω_U instead of Ω, to stress the dependence of its particular value on the choice of universal machine, but the fundamental properties of Ω do not depend on this choice, much as those of the halting problem do not depend on the specific choice of enumeration of Turing machines. In particular, Kučera and Slaman (2001) showed that every left-c.e. real is reducible to every Ω_U up to a strong notion of reducibility known as Solovay reducibility, and hence all such Ω_U's are equivalent modulo this notion. (The situation is analogous to that of versions of the halting problem, where the relevant notion is known as 1-reducibility.)

[3] Other recent work has explored the effect of adding axioms asserting the incompressibility of certain strings in a probabilistic way. Bienvenu, Romashchenko, Shen, Taveneaux, and Vermeeren [4] have shown that this kind of procedure does not help to prove new interesting theorems, but that the situation changes if we take into account the sizes of the proofs: randomly chosen axioms (in a sense made precise in their paper) can help to make proofs much shorter under the reasonable complexity-theoretic assumption that NP $\neq$ PSPACE.

Left-c.e. and right-c.e. reals (those of the form $1 - \alpha$ for a left-c.e. α) occur naturally in mathematics. Braverman and Yampolsky [7] showed that they arise in connection with Julia sets, and there is a striking example in symbolic dynamics: A d-dimensional *subshift of finite type* is a certain kind of collection of A-colorings of $\mathbb{Z}^d$, where A is a finite set, defined by local rules (basically saying that certain coloring patterns are illegal) invariant under the shift action

$$(S^g x)(h) = x(h+g) \text{ for } g, h \in \mathbb{Z}^d \text{ and } x \in A^{\mathbb{Z}^d}.$$

Its *(topological) entropy* is an important invariant measuring the asymptotic growth in the number of legal colorings of finite regions. It has been known for some time that entropies of subshifts of finite type for dimensions $d \geqslant 2$ are in general not computable, but the following result gives a precise characterization.

Theorem 3.1 (Hochman and Meyerovitch [15]). *The values of entropies of subshifts of finite type over $\mathbb{Z}^d$ for $d \geqslant 2$ are exactly the nonnegative right-c.e. reals.*

3.2. **Algorithmic randomness and relative computability.** Solovay reducibility is stronger than Turing reducibility, so Ω can compute the halting problem $\emptyset'$. Indeed Ω and $\emptyset'$ are Turing equivalent, and in fact Ω can be seen as a "highly compressed" version of $\emptyset'$. Other computability-theoretically powerful ML-random sequences can be obtained from the following remarkable result.

Theorem 3.2 (Gács (1986), Kučera (1985)). *For every X there is an ML-random Y such that $X \leqslant_{\mathrm{T}} Y$.*

This theorem and the Turing equivalence of Ω with $\emptyset'$ do not seem to accord with our intuition that random sets should have low "useful information". This phenomenon can be explained by results showing that, for certain purposes, the benchmark set by ML-randomness is too low. A set A has *PA degree* if it can compute a $\{0,1\}$-valued function f with $f(n) \neq \Phi_n(n)$ for all n. (The reason for the name is that this property is equivalent to being able to compute a completion of Peano Arithmetic.) Such a function can be seen as a weak version of the halting problem, but while $\emptyset'$ has PA degree, there are sets of PA degree that are low, in the sense of Section 1.2, and hence are far less powerful than $\emptyset'$.

Theorem 3.3 (Stephan (2006)). *If an ML-random sequence has PA degree then it computes $\emptyset'$.*

Thus there are two kinds of ML-random sequences. Ones that are complicated enough to somehow "simulate" randomness, and "truly random" ones that are much weaker. It is known that the class of sequences that can compute $\emptyset'$ has measure 0, so almost all ML-random sequences are in the second class. One way to ensure that a sequence is in that class is to increase the complexity of our tests by relativizing them to noncomputable oracles. It turns out that iterates of the Turing jump are particularly natural oracles to use. Let $\emptyset^{(0)} = \emptyset$ and $\emptyset^{(n+1)} = (\emptyset^{(n)})'$. We say that X is *n-random* if it passes all Martin-Löf tests relativized to to $\emptyset^{(n-1)}$. Thus the 1-random sequences are just the ML-random ones, while the 2-random ones are the ones that are ML-random relative to the halting problem. These sequences have low computational power in several ways. For instance, they cannot compute any noncomputable c.e. set, and in fact the following holds.

Theorem 3.4 (Kurtz (1981)). *If X is 2-random and Y is computable relative both to $\emptyset'$ and to X, then Y is computable.*

A precise relationship between tests and the dichotomy mentioned above was established by Franklin and Ng [11].

In general, among ML-random sequences, computational power (or "useful information") is inversely proportional to level of randomness. The following is one of many results attesting to this heuristic.

Theorem 3.5 (Miller and Yu (2008))**.** *Let $X \leqslant_{\mathrm{T}} Y$. If X is ML-random and Y is n-random, then X is also n-random.*

There are many other interesting levels of algorithmic randomness. Schnorr (1971) argued that his martingale characterization of ML-randomness shows that this is an intrinsically *computably enumerable* rather than *computable* notion, and defined a notion now called *Schnorr randomness*, which is like the notion of computable randomness mentioned below Definition 1.3 but with an extra effectiveness condition on the rate of success of martingales. He also showed that X is Schnorr random iff it passes all Martin-Löf tests $T_0, T_1, \dots$ such that the measures $\lambda(T_n)$ are uniformly computable (i.e., the function $n \mapsto \lambda(T_n)$ is computable in the sense of Section 4.4 below). It follows immediately from their definitions in terms of martingales that ML-randomness implies computable randomness, which in turn implies Schnorr randomness. It is more difficult to prove that none of these implications can be reversed. In fact, these levels of randomness are close enough that they agree for sets that are somewhat close to computable, as shown by the following result, where highness is as defined in Section 1.2.

Theorem 3.6 (Nies, Stephan, and Terwijn (2005))**.** *Every high Turing degree contains a set that is computably random but not ML-random and a set that is Schnorr random but not computably random. This fact is tight, however, because every nonhigh Schnorr random set is ML-random.*

As we will discuss, various notions of algorithmic randomness arise naturally in applications.

3.3. **Randomness-theoretic weakness.** As mentioned above, X is ML-random iff $K(X \upharpoonright n) \geqslant n - O(1)$, i.e., X's initial segments have very high complexity. There are similar characterizations of other notions of algorithmic randomness, as well as of notions arising in other parts of computability theory, in terms of high initial segment complexity. For instance, Downey and Griffiths (2004) showed that X is Schnorr random iff $C_M(X \upharpoonright n) \geqslant n - O(1)$ for every prefix-free machine M with computable halting probability, while Kjos-Hanssen, Merkle, and Stephan (2006) showed that X can compute a diagonally noncomputable function, that is, a function h with $h(e) \neq \Phi_e(e)$ for all e, iff there is an X-computable function f such that $C(X \upharpoonright f(n)) \geqslant n$ for all n. But what if the initial segments of a sequence have *low* complexity? Such sequences have played an important role in the theory of algorithmic randomness, beginning with the following information-theoretic characterization of computability.

Theorem 3.7 (Chaitin (1976))**.** *$C(X \upharpoonright n) \leqslant C(n) + O(1)$ iff X is computable.*

It is also true that if X is computable then $K(X \upharpoonright n) \leqslant K(n) + O(1)$. Chaitin (1977) considered sequences with this property, which are now called *K-trivial.* He showed that every K-trivial sequence is $\emptyset'$-computable, and asked whether they are all in fact computable. Solovay (1975) answered this question by constructing a noncomputable K-trivial sequence.

The class of K-trivials has several remarkable properties. It is a naturally definable *countable* class, contained in the class of low sets (as defined in Section 1.2, where we identify a set with its characteristic function, thought of as a sequence), but with stronger closure properties. (In technical terms, it is what is known as a *Turing ideal.*) Post's problem asked whether there are computably enumerable sets that are neither computable nor Turing equivalent to the halting problem. Its solution in the 1950's by Friedberg and Muchnik introduced the somewhat complex priority method, which has played a central technical role in computability theory since then. Downey, Hirschfeldt, Nies, and Stephan (2003) showed that K-triviality can be used to give a simple priority-free solution to Post's problem.

Most significantly, there are many natural notions of randomness-theoretic weakness that turn out to be equivalent to K-triviality.

Theorem 3.8 (Nies (2005), Nies and Hirschfeldt for (1) → (3))**.** *The following are equivalent.*

(1) *A is K-trivial.*
(2) *A is computable relative to some c.e. K-trivial set.*
(3) *A is* low for K*, meaning that A has no compression power as an oracle. i.e., that $K^A(\sigma) \geqslant K(\sigma) - O(1)$, where K^A is the relativization of prefix-free Kolmogorov complexity to A.*
(4) *A is* low for ML-randomness*, meaning that A does not have any derandomization power as an oracle, i.e., any ML-random set remains ML-random when this notion is relativized to A.*

There are now over a dozen other characterizations of K-triviality. Some appear in [9, 24], and several others have emerged more recently. These have been used to solve several problems in algorithmic randomness and related areas. Lowness classes have also been found for other randomness notions. For Schnorr randomness, for instance, lowness can be characterized using notions of traceability related to concepts in set theory, as first explored by Terwijn and Zambella (2001).

4. Some applications

4.1. **Incompressibility and information content.** This article focuses on algorithmic randomness for infinite objects, but we should mention that there have been many applications of Kolmogorov complexity under the collective title of the *incompressibility method*, based on the observation that algorithmically random strings should exhibit typical behavior for computable processes. For example, this method can be used to give average running times for sorting, by showing that if the outcome is not what we would expect then we can compress a random input. See Li and Vitányi [18, Chapter 6] for applications of this technique to areas as diverse as combinatorics, formal languages, compact routing, and circuit complexity, among others. Many results originally proved using Shannon entropy or related methods also have proofs using Kolmogorov complexity. For example, Messner and Thierauf [22] gave a constructive proof of the Lovász Local Lemma using Kolmogorov complexity.

Other applications come from the observation that in some sense Kolmogorov complexity provides an "absolute" measure of the intrinsic complexity of a string. We can define a notion of conditional Kolmogorov complexity $C(\sigma \mid \tau)$ of a string

σ given another string τ. Then, for example, $C(\sigma \mid \sigma) = O(1)$, and σ is "independent of τ" if $C(\sigma \mid \tau) = C(\sigma) - O(1)$. Researchers comparing two sequences σ, τ representing, say, two DNA sequences, or two phylogenetic trees, or two languages, or two pieces of music, have invented many distance metrics, such as the maximum parsimony distance on phylogenetic trees, but it is also natural to use a content-neutral measure of "information distance" like $\max\{C(\sigma \mid \tau), C(\tau \mid \sigma)\}$. There have been some attempts to make this work in practice for solving classification problems, though results have so far been mixed. Of course, C is not computable, but it can be replaced in applications by measures derived from practical compression algorithms. See [18, Sections 8.3 and 8.4].

4.2. **Effective dimensions.** If $X = x_0x_1\dots$ is random, then we might expect a sequence such as $x_000x_100x_200\dots$ to be "$\frac{1}{3}$-random". Making precise sense of the idea of partial algorithmic randomness has led to significant applications. Hausdorff used work of Carathéodory on s-dimensional measures to generalize the notion of dimension to possibly nonintegral values, leading to concepts such as Hausdorff dimension and packing dimension. Much like algorithmic randomness can make sense of the idea of individual reals being random, notions of partial algorithmic randomness can be used to assign dimensions to individual reals.

The measure-theoretic approach, in which we for instance replace the uniform measure λ on 2^ω by a generalized notion assigning the value $2^{-s|\sigma|}$ to $[\sigma]$ (where $0 < s \leqslant 1$), was translated by Lutz (2000, 2003) into a notion of *s-gale*, where the fairness condition of a martingale is replaced by $f(\sigma) = 2^{-s}(f(\sigma 0) + f(\sigma 1))$. We can view s-gales as modeling betting in a hostile environment (an idea due to Lutz), where "inflation" is acting so that not winning means that we automatically lose money. Roughly speaking, the effective fractal dimension of a sequence is then determined by the most hostile environment in which we can still make money betting on this sequence.

Mayordomo (2002) and Athreya, Hitchcock, Lutz, and Mayordomo (2007) found equivalent formulations in terms of Kolmogorov complexity, which we take as definitions. (Here it does not matter whether we use plain or prefix-free Kolmogorov complexity.)

Definition 4.1. Let $X \in 2^\omega$. The *effective Hausdorff dimension* of X is

$$\dim(X) = \liminf_{n\to\infty} \frac{K(X \upharpoonright n)}{n}.$$

The *effective packing dimension* of X is

$$\mathrm{Dim}(X) = \limsup_{n\to\infty} \frac{K(X \upharpoonright n)}{n}.$$

It is not hard to extend these definitions to elements of $\mathbb{R}^n$, yielding effective dimensions between 0 and n. They can also be relativized to any oracle A to obtain the effective Hausdorff and packing dimensions $\dim^A(X)$ and $\mathrm{Dim}^A(X)$ of X relative to A.

It is of course not immediately obvious why these notions are effectivizations of Hausdorff and packing dimension, but crucial evidence of their correctness is provided by *point to set principles*, which allow us to express the dimensions of sets of reals in terms of the effective dimensions of their elements. The most recent and powerful of these is the following, where we denote the classical Hausdorff dimension of $E \subseteq \mathbb{R}^n$ by $\dim_{\mathrm{H}}(E)$, and its classical packing dimension by $\dim_{\mathrm{p}}(E)$.

Theorem 4.2 (Lutz and Lutz [19])**.**

$$\dim_{\mathrm{H}}(E) = \min_{A \subseteq \mathbb{N}} \sup_{X \in E} \dim^A(X).$$

$$\dim_{\mathrm{p}}(E) = \min_{A \subseteq \mathbb{N}} \sup_{X \in E} \mathrm{Dim}^A(X).$$

For certain well-behaved sets E, relativization is actually not needed, and the classical dimension of E is the supremum of the effective dimensions of its points. In the general case, it is of course not immediately clear that the minima mentioned in Theorem 4.2 should exist, but they do. Thus, for example, to prove a lower bound of α for $\dim_{\mathrm{H}}(E)$ it suffices to prove that, for each $\varepsilon > 0$ and each A, the set E contains a point X with $\dim^A(X) > \alpha - \varepsilon$. In several applications, this argument turns out to be easier than ones directly involving classical dimension. This fact is somewhat surprising given the need to relativize to arbitrary oracles, but in practice this issue has so far turned out not to be an obstacle.

For example, Lutz and Stull [21] obtained a new lower bound on the Hausdorff dimension of generalized sets of Furstenberg type; Lutz [20] showed that a fundamental intersection formula, due in the Borel case to Kahane and Mattila, is true for arbitrary sets; and Lutz and Lutz [19] gave a new proof of the two-dimensional case (originally proved by Davies) of the well-known Kakeya conjecture, which states that, for all $n \geqslant 2$, if a subset of $\mathbb{R}^n$ has lines of length 1 in all directions, then it has Hausdorff dimension n.

There had been earlier applications of effective dimension, for instance in symbolic dynamics, whose iterative processes are naturally algorithmic. For example, Simpson [25] generalized a result of Furstenberg as follows. Let A be finite and G be either $\mathbb{N}^d$ or $\mathbb{Z}^d$. A closed set $X \subseteq A^G$ is a *subshift* if it is closed under the shift action of G on A^G (see Section 3.1).

Theorem 4.3 (Simpson [25])**.** *Let A be finite and G be either $\mathbb{N}^d$ or $\mathbb{Z}^d$. If $X \subseteq A^G$ is a subshift then the topological entropy of X is equal both to its classical Hausdorff dimension and to the supremum of the effective Hausdorff dimensions of its elements.*

In currently unpublished work, Day has used effective methods to give a new proof of the Kolmogorov-Sinai theorem on entropies of Bernoulli shifts.

There are other applications of sequences of high effective dimension, for instance ones involving the interesting class of *shift complex* sequences. While initial segments of ML-random sequences have high Kolmogorov complexity, not all segments of such sequences do. Random sequences must contain arbitrarily long strings of consecutive 0's, for example. However, for any $\varepsilon > 0$ there are *ε-shift complex* sequences Y such that for any string σ of consecutive bits of Y, we have $K(\sigma) \geqslant (1-\varepsilon)|\sigma| - O(1)$. These sequences can be used to create tilings with properties such as certain kinds of pattern-avoidance, and have found uses in symbolic dynamics. See for instance Durand, Levin, and Shen (2008) and Durand, Romashchenko, and Shen [10].

4.3. **Randomness amplification.** Many practical algorithms use random seeds. For example, the important *Polynomial Identity Testing* (*PIT*) problem takes as input a polynomial $P(x_1, \ldots, x_n)$ with coefficients from a large finite field and determines whether it is identically 0. Many practical problems can be solved using a reduction to this problem. There is a natural fast algorithm to solve it randomly: Take a random sequence of values for the variables. If the polynomial is not 0 on these values, "no" is the correct answer. Otherwise, the probability that the answer is "yes" is very high. It is conjectured that PIT has a polynomial-time deterministic algorithm,[4] but no such algorithm is known.

Thus it is important to have good sources of randomness. Some (including Turing) have believed that randomness can be obtained from physical sources, and there are now commercial devices claiming to do so. At a more theoretical level, we might ask question such as:

(1) Can a weak source of randomness always be amplified into a better one?
(2) Can we in fact always recover full randomness from partial randomness?
(3) Are random sources truly useful as computational resources?

In our context, we can consider precise versions of such questions by taking randomness to mean algorithmic randomness, and taking all reduction processes to be computable ones. One way to interpret the first two questions then is to think of partial randomness as having nonzero effective dimension. For example, for packing dimension, we have the following negative results.

Theorem 4.4 (Downey and Greenberg (2008))**.** *There is an X such that* $\operatorname{Dim}(X) = 1$ *and X computes no ML-random sequence. (This X can be built to be of* minimal degree*, which means that every X-computable set is either computable or has the same Turing degree as X. It is known that such an X cannot compute an ML-random sequence.)*

Theorem 4.5 (Conidis [8])**.** *There is an X such that* $\operatorname{Dim}(X) > 0$ *and X computes no Y with* $\operatorname{Dim}(Y) = 1$.

On the other hand, we also have the following strong positive result.

Theorem 4.6 (Fortnow, Hitchcock, Pavan, Vinochandran, and Wang (2006))**.** *If* $\varepsilon > 0$ *and* $\operatorname{Dim}(X) > 0$ *then there is an X-computable Y such that* $\operatorname{Dim}(Y) > 1-\varepsilon$. *(In fact, Y can be taken to be equivalent to X via polynomial-time reductions.)*

For effective Hausdorff dimension, the situation is quite different. Typically, the way we obtain an X with $\dim(X) = \frac{1}{2}$, say, is to start with an ML-random sequence and somehow "mess it up", for example by making every other bit a 0. This kind of process is reversible, in the sense that it easy to obtain an X-computable ML-random. However, Miller [23] showed that it is possible to obtain sequences of fractional effective Hausdorff dimension that permit no randomness amplification at all.

Theorem 4.7 (Miller [23])**.** *There is an X such that* $\dim(X) = \frac{1}{2}$ *and if* $Y \leqslant_{\mathrm{T}} X$ *then* $\dim(Y) \leqslant \frac{1}{2}$.

That is, effective Hausdorff dimension *cannot* in general be amplified. (In this theorem, the specific value $\frac{1}{2}$ is only an example.) Greenberg and Miller [13] also showed that there is an X such that $\dim(X) = 1$ and X does not compute any ML-random sequences. Interestingly, Zimand (2010) showed that for *two* sequences X and Y of nonzero effective Hausdorff dimension that are in a certain technical sense

[4] This conjecture comes from the fact that PIT belongs to a complexity class known as BPP, which is widely believed to equal the complexity class P of polynomial-time solvable problems, since Impagliazzo and Wigderson showed in the late 1990's that if the well-known Satisfiability problem is as hard as generally believed, then indeed BPP = P.

sufficiently independent, X and Y together can compute a sequence of effective Hausdorff dimension 1.

In some attractive recent work, it has been shown that there is a sense in which the intuition that every sequence of effective Hausdorff dimension 1 is close to an ML-random sequence is correct. The following is a simplified version of the full statement, which quantifies how much randomness can be extracted at the cost of altering a sequence on a set of density 0. Here $A \subseteq \mathbb{N}$ has *(asymptotic) density* 0 if $\lim_{n\to\infty} \frac{|A\upharpoonright n|}{n} = 0$.

Theorem 4.8 (Greenberg, Miller, Shen, and Westrick [14])**.** *If* $\dim(X) = 1$ *then there is an ML-random* Y *such that* $\{n : X(n) \neq Y(n)\}$ *has density* 0*.*

The third question above is whether sources of randomness can be useful oracles. Here we are thinking in terms of complexity rather than just computability, so results such as Theorem 3.2 are not directly relevant. Allender and others have initiated a program to investigate the speedups that are possible when random sources are queried efficiently. Let R be the set of all random finite binary strings for either plain or prefix-free Kolmogorov complexity (e.g., $R = \{x : C(x) \geqslant |x|\}$). For a complexity class $\mathcal{C}$, let $\mathcal{C}^R$ denote the relativization of this class to R. So, for instance, for the class P of polynomial-time computable functions, P^R is the class of functions that can be computed in polynomial time with R as an oracle. (For references to the articles in this and the following theorem, see [1].)

Theorem 4.9 (Buhrman, Fortnow, Koucký, and Loff (2010); Allender, Buhrman, Koucký, van Melkebeek, and Ronneburger (2006); Allender, Buhrman, and Koucký (2006))**.**

(1) $\mathrm{PSPACE} \subseteq \mathrm{P}^R$.
(2) $\mathrm{NEXP} \subseteq \mathrm{NP}^R$.
(3) $\mathrm{BPP} \subseteq \mathrm{P}^R_{tt}$ *(where the latter is the class of functions that are reducible to* R *in polynomial time via truth-table reductions, a more restrictive notion of reduction than Turing reduction).*

The choice of universal machine does have some effect on efficient computations, but we can quantify over all universal machines. In the result below, U ranges over universal prefix-free machines, and R_{K_U} is the set of random strings relative to Kolmogorov complexity defined using U.

Theorem 4.10 (Allender, Friedman, and Gasarch (2013); Cai, Downey, Epstein, Lempp, and Miller (2014))**.**

(1) $\bigcap_U \mathrm{P}^{R_{K_U}}_{tt} \subseteq \mathrm{PSPACE}$.
(2) $\bigcap_U \mathrm{NP}^{R_{K_U}} \subseteq \mathrm{EXPSPACE}$.

We can also say that sufficiently random oracles will always accelerate *some* computations in the following sense. Say that X is *low for speed* if for any computable set A and any function t such that A can be computed in time $t(n)$ using X as an oracle, there is a polynomial p such that A can be computed (with no oracle) in time bounded by $p(t(n))$. That is, X does not significantly accelerate any computation of a computable set. Bayer and Slaman (see [3]) constructed noncomputable sets that are low for speed, but these cannot be very random.

Theorem 4.11 (Bienvenu and Downey [3]). *If X is Schnorr random, then it is not low for speed, and this fact is witnessed by an exponential-time computable set A.*

4.4. **Analysis and Ergodic Theory.** Computable analysis is an area that has developed tools for thinking about computability of objects like real-valued functions by taking advantage of separability. Say that a sequence of rationals $q_0, q_1, \dots$ *converges fast* to x if $|x - q_n| \leqslant 2^{-n}$ for all n. A function $f : \mathbb{R} \to \mathbb{R}$ is *(Type 2) computable* if there is an algorithm Φ that, for every $x \in \mathbb{R}$ and every sequence $q_0, q_1, \dots$ that converges fast to x, if Φ is given $q_0, q_1, \dots$ as an oracle, then it can compute a sequence that converges fast to $f(x)$. We can extend this definition to similar separable spaces. We can also relativize it, and it is then not difficult to see that a function is continuous iff it is computable relative to some oracle, basically because to define a continuous function we need only to specify its action on a countable collection of balls.

Mathematics is replete with results concerning almost everywhere behavior, and algorithmic randomness allows us to to turn such results into "quantitative" ones like the following.

Theorem 4.12 (Brattka, Miller, and Nies [5], also Demuth (1975, see [5]) for (2)).

(1) *The reals at which every computable increasing function $\mathbb{R} \to \mathbb{R}$ is differentiable are exactly the computably random ones.*

(2) *The reals at which every computable function $\mathbb{R} \to \mathbb{R}$ of bounded variation is differentiable are exactly the ML-random ones.*

Ergodic theory is another area that has been studied from this point of view. A measure-preserving transformation T on a probability space is *ergodic* if all measurable subsets E such that $T^{-1}(E) = E$ have measure 1 or 0. Notice that this is an "almost everywhere" definition. We can make this setting computable (and many systems arising from physics will be computable). One way to proceed is to work in Cantor space without loss of generality, since Hoyrup and Rojas [16] showed that any computable metric space with a computable probability measure is isomorphic to this space in an effective measure-theoretic sense. Then we can specify a computable transformation T as a computable limit of computable partial maps $T_n : 2^{<\omega} \to 2^{<\omega}$ with certain coherence conditions. We can also transfer definitions like that of ML-randomness to computable probability spaces other than Cantor space.

The following is an illustrative result. A classic theorem of Poincaré is that if T is measure-preserving, then for all E of positive measure and almost all x, we have $T^n(x) \in E$ for infinitely many n. For a class $\mathcal{C}$ of measurable subsets, x is a *Poincaré point* for T with respect to $\mathcal{C}$ if for every $E \in \mathcal{C}$ of positive measure, $T^n(x) \in E$ for infinitely many n. An *effectively closed* set is one whose complement can be specified as a computably enumerable union of basic open sets.

Theorem 4.13 (Bienvenu, Day, Mezhirov, and Shen [2]). *Let T be a computable ergodic transformation on a computable probability space. Every ML-random element of this space is a Poincaré point for the class of effectively closed sets.*

In general, the condition that the element be ML-random is not just sufficient but necessary, even in a simple case like the shift operator on Cantor space.

The non-ergodic case has also been analyzed, by Franklin and Towsner [12], who also studied the Birkhoff ergodic theorem. In these and several other cases, similar correspondences with various notions of algorithmic randomness have been found. While many theorems of ergodic theory have been analyzed in this way, including the Birkhoff, Poincaré, and von Neumann ergodic theorems, some, like Furstenberg's ergodic theorem, have yet to be understood from this point of view.

Regarding the physical interpretation of some of the work in this area, Braverman, Grigo, and Rojas [6] have obtained results that they argue show that, while random noise makes predicting the short term behavior of a system difficult, it may in fact allow prediction to be easier in the long term.

4.5. **Normality revisited.** Borel's notion of normality, with which we began our discussion, is a very weak kind of randomness. Polynomial-time randomness implies absolute normality, and Schnorr and Stimm (1971/72) showed that a sequence is normal to a given base iff it satisfies a martingale-based notion of randomness defined using certain finite state automata, a much weaker model of computation than Turing machines. Building examples of absolutely normal numbers is another matter, as Borel already noted. While it is conjectured that e, π, and all irrational algebraic numbers such as $\sqrt{2}$ are absolutely normal, *none* of these have been proved to be normal to *any* base. In his unpublished manuscript "A note on normal numbers", believed to have been written in 1938, Turing built a computable absolutely normal real, which is in a sense the closest we have come so far to obtaining an explicitly-described absolutely normal real. (His construction was not published until his Collected Works in 1992, and there was uncertainty as to its correctness until Becher, Figueira, and Picchi (2007) reconstructed and completed it, correcting minor errors.[5]) There is a sense in which Turing anticipated Martin-Löf's idea of looking at a large collection of effective tests, in this case ones sufficiently strong to ensure that a real is normal for all bases, but sufficiently weak to allow some computable sequence to pass them all. He took advantage of the correlations between blocks of digits in expansions of the same real in different bases.

This approach can also be thought of in terms of effective martingales, and its point of view has brought about a great deal of progress in our understanding of normality recently. For instance, Becher, Heiber, and Slaman (2013) showed that absolutely normal numbers can be constructed in low-level polynomial time, and Lutz and Mayordomo (`arXiv:1611.05911`) constructed them in "nearly linear" time. Much of the work along these lines has been number-theoretic, connected to various notions of well-approximability of irrational reals, such as that of a *Liouville number*, which is an irrational α such that for every natural number $n > 1$, there are $p, q \in \mathbb{N}$ for which $|\alpha - \frac{p}{q}| < q^{-n}$. For example, Becher, Heiber, and Slaman (2015) have constructed computable absolutely normal Liouville numbers. This work has also produced results in the classical theory of normal numbers, for instance by Becher, Bugeaud, and Slaman (2016).

References

[1] E. Allender. The complexity of complexity. In *Computability and Complexity*, volume 10010 of *Lecture Notes in Comput. Sci.*, pages 79–94. Springer, Cham, 2017.

[2] L. Bienvenu, A. Day, I. Mezhirov, and A. Shen. Ergodic-type characterizations of algorithmic randomness. In *Programs, Proofs, Processes*, volume 6158 of *Lecture Notes in Comput. Sci.*, pages 49–58. Springer, Berlin, 2010.

[5] See `https://www-2.dc.uba.ar/staff/becher/publications.html` for references to the papers cited here and below.

[3] L. Bienvenu and R. Downey. On Low for Speed Oracles. In R. Niedermeier and B. Vallée, editors, *35th Symposium on Theoretical Aspects of Computer Science (STACS 2018)*, volume 96 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 15:1–15:13, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

[4] L. Bienvenu, A. Romashchenko, A. Shen, A. Taveneaux, and S. Vermeeren. The axiomatic power of Kolmogorov complexity. *Annals of Pure and Applied Logic*, 165(9):1380–1402, 2014.

[5] V. Brattka, J. S. Miller, and A. Nies. Randomness and differentiability. *Transactions of the American Mathematical Society*, 368(1):581–605, 2016.

[6] M. Braverman, A. Grigo, and C. Rojas. Noise vs computational intractability in dynamics. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, pages 128–141. ACM, New York, 2012.

[7] M. Braverman and M. Yampolsky. Computability of Julia sets. *Moscow Mathematical Journal*, 8(2):185–231, 399, 2008.

[8] C. J. Conidis. A real of strictly positive effective packing dimension that does not compute a real of effective packing dimension one. *The Journal of Symbolic Logic*, 77(2):447–474, 2012.

[9] R. G. Downey and D. R. Hirschfeldt. *Algorithmic Randomness and Complexity*. Theory and Applications of Computability. Springer, New York, 2010.

[10] B. Durand, A. Romashchenko, and A. Shen. Fixed-point tile sets and their applications. *Journal of Computer and System Sciences*, 78(3):731–764, 2012.

[11] J. N. Y. Franklin and K. M. Ng. Difference randomness. *Proceedings of the American Mathematical Society*, 139(1):345–360, 2011.

[12] J. N. Y. Franklin and H. Towsner. Randomness and non-ergodic systems. *Moscow Mathematical Journal*, 14(4):711–744, 827, 2014.

[13] N. Greenberg and J. S. Miller. Diagonally non-recursive functions and effective Hausdorff dimension. *Bulletin of the London Mathematical Society*, 43(4):636–654, 2011.

[14] N. Greenberg, J. S. Miller, A. Shen, and L. B. Westrick. Dimension 1 sequences are close to randoms. *Theoretical Computer Science*, 705:99–112, 2018.

[15] M. Hochman and T. Meyerovitch. A characterization of the entropies of multidimensional shifts of finite type. *Annals of Mathematics. Second Series*, 171(3):2011–2038, 2010.

[16] M. Hoyrup and C. Rojas. Computability of probability measures and Martin-Löf randomness over metric spaces. *Information and Computation*, 207(7):830–847, 2009.

[17] S. Kritchman and R. Raz. The surprise examination paradox and the second incompleteness theorem. *Notices of the American Mathematical Society*, 57(11):1454–1458, 2010.

[18] M. Li and P. Vitányi. *An Introduction to Kolmogorov Complexity and its Applications*. Texts in Computer Science. Springer, New York, third edition, 2008.

[19] J. H. Lutz and N. Lutz. Algorithmic information, plane Kakeya sets, and conditional dimension. *ACM Transactions on Computation Theory*, 10(2):Art. 7, 22, 2018.

[20] N. Lutz. Fractal intersections and products via algorithmic dimension. In *42nd International Symposium on Mathematical Foundations of Computer Science*, volume 83 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 58, 12. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2017.

[21] N. Lutz and D. M. Stull. Bounding the dimension of points on a line. In *Theory and Applications of Models of Computation*, volume 10185 of *Lecture Notes in Comput. Sci.*, pages 425–439. Springer, Cham, 2017.

[22] J. Messner and T. Thierauf. A Kolmogorov complexity proof of the Lovász local lemma for satisfiability. *Theoretical Computer Science*, 461:55–64, 2012.

[23] J. S. Miller. Extracting information is hard: a Turing degree of non-integral effective Hausdorff dimension. *Advances in Mathematics*, 226(1):373–384, 2011.

[24] A. Nies. *Computability and Randomness*, volume 51 of *Oxford Logic Guides*. Oxford University Press, Oxford, 2009.

[25] S. G. Simpson. Symbolic dynamics: entropy = dimension = complexity. *Theory of Computing Systems*, 56(3):527–543, 2015.

School of Mathematics and Statistics, Victoria University, PO Box 600, Wellington, New Zealand
E-mail address: rod.downey@vuw.ac.nz

Department of Mathematics, The University of Chicago, 5734 S. University Ave., Chicago, IL 60637, USA
E-mail address: drh@math.uchicago.edu