KEY DEVELOPMENTS IN ALGORITHMIC RANDOMNESS

JOHANNA N. Y. FRANKLIN AND CHRISTOPHER P. PORTER

Contents

§1. Introduction. The goal of this introductory survey is to present the major developments of algorithmic randomness with an eye toward its historical development. While two highly comprehensive books [26, 81] and one thorough survey article [21] have been written on the subject, our goal is to provide an introduction to algorithmic randomness that will be both useful for newcomers who want to develop a sense of the field quickly and interesting for researchers already in the field who would like to see these results presented in chronological order.

Algorithmic Randomness: Progress and Prospects
Edited by Johanna N. Y. Franklin and Christopher P. Porter
Lecture Notes in Logic, 50

We begin in this section with a brief introduction to computability theory as well as the underlying mathematical concepts that we will later draw upon. Once these basic ideas have been presented, we will selectively survey four broad periods in which the primary developments in algorithmic randomness occurred: (1) the mid-1960s to mid-1970s, in which the main definitions of algorithmic randomness were laid out and the basic properties of random sequences were established, (2) the 1980s through the 1990s, which featured intermittent and important work from a handful of researchers, (3) the 2000s, during which there was an explosion of results as the discipline matured into a full-fledged subbranch of computability theory, and (4) the early 2010s, which we briefly discuss as a lead-in to the remaining surveys in this volume, which cover in detail many of the exciting developments in this later period.

We do not intend this to be a full reconstruction of the history of algorithmic randomness, nor are we claiming that the only significant developments in algorithmic randomness are the ones recounted here. Instead, we aim to provide readers with sufficient context for appreciating the more recent work that is described in the surveys in this volume. Moreover, we highlight those concepts and results that will be useful for our readers to be aware of as they read the later chapters.

Before we proceed with the technical material, we briefly comment upon several broader conceptual questions which may occur to the newcomer upon reading this survey: What is a definition of algorithmic randomness intended to capture? What is the aim of studying the properties of the various types of randomness? And why are there so many definitions of randomness to begin with? It is certainly beyond the scope of this survey to answer these questions in any detail. Here we note first that more recent motivations for defining randomness and studying the properties of the resulting definitions have become unmoored from the original motivation that led to early definitions of randomness, namely, providing a foundation for probability theory (see, for example, [85]).

This original motivation led to the desire for a definition of a random sequence satisfying the standard statistical properties of almost every sequence (such as the strong law of large numbers and the law of the iterated logarithm). Martin-Löf's definition was the first to satisfy this constraint. Moreover, this definition proved to be robust, as it was shown to be equivalent to definitions of randomness with a significantly different informal motivation: while Martin-Löf's definition was motivated by the idea that random sequences are statistically typical, later characterizations were given in terms of incompressibility and unpredictability.

With such a robust definition of randomness, one can inquire into just how stable it is: if we modify a given technical aspect of the definition, is the resulting notion equivalent to Martin-Löf randomness? As we will see below, the answer is often negative. As there are a number of such modifications, we now have

a number of nonequivalent definitions of randomness. Understanding the relationships between these notions of randomness, as well as the properties of the sequences that satisfy them, is certainly an important endeavor.

One might legitimately express the concern that such work amounts to simply concocting new definitions of randomness and exploring their features. However, not every new variant of every notion of randomness has proven to be significant. Typically, attention is given to definitions of randomness that have multiple equivalent formalizations, or which interact nicely with computability-theoretic notions, or which provide insight into some broader phenomenon such as the analysis of almost sure properties that hold in classical mathematics. Many such developments are outlined in the surveys in this volume.

1.1. Notation. Our notation will primarily follow [26] to make it easier to cross-reference these results. The set of natural numbers will be denoted by ω, and we will usually name elements of this set using lowercase Latin letters such as m and n. Subsets of ω will be denoted by capital Latin letters such as A and B. Without loss of generality, we may associate an element of 2^ω (that is, an infinite binary sequence) with the subset of ω consisting of the places at which the infinite binary sequence is equal to 1. Finite binary strings, or elements of $2^{<\omega}$, will be denoted by lowercase Greek letters such as σ and τ.

We will often wish to discuss the subset of 2^ω whose elements all begin with the same prefix σ; we will denote this by $[\sigma]$. We will further extend this to an arbitrary subset S of $2^{<\omega}$:

$$[S] = \{A \in 2^\omega \mid \sigma \preceq A \text{ for some } \sigma \in S\}.$$

The first n bits of a binary sequence X of length at least n, be it infinite or finite, will be denoted by $X\restriction n$, and the length of a finite binary string σ will be denoted by $|\sigma|$. The concatenation of two binary strings σ and τ will be denoted by $\sigma\tau$.

1.2. Computability theory. This section is intended for researchers in other areas of mathematics who are encountering computability theory for the first time and require an introduction to the underlying concepts; others may safely skip it. While each of Chapter 2 of [26] and Chapter 1 of [81] contains all the fundamental concepts of computability theory that will be required for this volume in more detail, researchers who wish to acquire a deeper understanding of the subject are urged to consult Cooper [18], Odifreddi [83, 84], or Soare [91].

Computability theory allows us to think about mathematical functions in an effective context. While there are several ways to formalize the notions we are about to describe, including Turing machines, register machines, the λ-calculus, and μ-recursive functions, we will not fix such a formalization and will instead encourage the reader to think of the concepts we describe below in

terms of the calculations that a computer with potentially unlimited memory is capable of carrying out in a finite but unbounded amount of time.

The most fundamental concept is that of a *partial computable function* φ, which can be thought of as an idealized computer program that accepts natural numbers as inputs and outputs natural numbers as well. We note that when a computer program is given an input (and thus when a partial computable function is), it may either return an answer at some finite point, or *stage*, or never halt. If a partial computable function halts on every natural number (that is, it is actually a total function), we simply call it a *computable function*.

We now provide some necessary notation. If φ halts on input n, we write $\varphi(n)\downarrow$; if it does not, we write $\varphi(n)\uparrow$. Furthermore, if φ halts on input n and gives the output m within s stages, we write $\varphi(n)[s] = m$ to indicate the number of stages as well as the output.

At this point, we make three observations. The first is that there are countably many partial computable functions: each partial computable function is associated with a computer program, and we can note that a computer program is a finite sequence of characters from a finite alphabet and that there are thus countably many such objects. The second is that we can list the partial computable functions in a computable way simply by generating a list of all of the "grammatically correct" programs and that thus we can speak about, for instance, the k^{th} partial computable function φ_k. While there are still only countably many (total) computable functions and thus we can list them as well, it can be shown that we cannot list them in a computable way because no computer program is capable of identifying precisely the partial computable functions that halt on every natural number. The third is that there are computable bijections between the natural numbers and the sets of finite binary strings and the rational numbers, so we may discuss partial computable functions from or to these sets without loss of generality.

Now we can define special kinds of subsets of ω. A *computably enumerable* (*c.e.*) set is one that is the range of a partial computable function, so we will often write W_e for the set that is the range of the e^{th} partial computable function φ_e. We may think of φ_e as enumerating W_e as follows: we first spend one step trying to compute $\varphi_e(0)$, then two steps trying to calculate each of $\varphi_e(0)$ and $\varphi_e(1)$, then three steps trying to calculate each of $\varphi_e(0)$, $\varphi_e(1)$, and $\varphi_e(2)$, and so on.[1] If the calculation of $\varphi_e(n)$ ever halts, we will eventually discover this through this dovetailing of computations, and when we do, we will enumerate its value into our set W_e. Now we can use this idea of set enumeration to formalize the concept of approximations to a set: for any c.e. set W_e, we say that

[1] Each of these steps may be said to make up a single stage of the computation mentioned above. These steps will be defined differently based on our formalization of computability theory: they may be the number of states a Turing machine has been in or the number of reduction rules applied in the λ-calculus, but, at a less formal level, we may think of "spending n steps" as "running the computer program for n seconds."

the approximation to it at stage s is $W_e[s] = \{n \mid (\exists k \leq s)[n = \varphi_e(k)[s]]\}$. This gives us a sequence of approximations that converge to our c.e. set; in fact, we can write $W_e = \bigcup_{s\in\omega} W_e[s]$. We quickly observe that there are several equivalent definitions of a c.e. set; the other one that will be especially useful to us is that of a c.e. set as one that is the domain of a partial computable function.

A set that is itself c.e. and has a c.e. complement is called a *computable* set. Just as a computable function halts on every input n, we can get an answer to "Is n in A?" for a computable set A for every n: to see this, we observe that if $A = W_e$ and $\overline{A} = W_i$, then we can determine whether $n \in A$ by enumerating W_e and W_i as described above; n must be in one of them, and we simply note which. These two procedures can again be dovetailed and performed by a single function that will give us the *characteristic function* of A:

$$\chi_A(n) = \begin{cases} 1 & n \in A, \\ 0 & n \notin A. \end{cases}$$

We can therefore show that the characteristic function of a computable set will be a computable function. Once again, there are countably many c.e. sets and countably many computable sets.

Often, when we discuss randomness, we will talk about a sequence of *uniformly c.e.* sets. Instead of simply requiring that each sequence in the set be c.e., we require that there be a single computable function that generates the entire sequence: $\langle A_i\rangle_{i\in\omega}$ is uniformly c.e. if there is a computable function f such that A_i is the range of the $f(i)^{th}$ partial computable function. Later, we will generalize this concept to other classes of sets that have some natural indexing: given such a class of sets $\mathcal{C}$, we can say that that we have a sequence of uniformly $\mathcal{C}$ sets $\langle A_i\rangle_{i\in\omega}$ if there is a computable function f such that $f(i)$ gives the index of the i^{th} set in the sequence.

The final topic we must consider in order to understand the concepts in algorithmic randomness we will discuss in this survey is that of oracle computation, or relativization. This requires us to consider *Turing functionals*, usually denoted by capital Greek letters such as Φ, which require not only a natural number n as input but a sequence X that serves as an oracle. These functionals can make use of the standard computational methods of partial computable functions and receive answers to finitely many queries of the sort "Is k in X?" for use in their computation, and they can be indexed as $\Phi_0, \Phi_1, \ldots$ just as the partial computable functions can be indexed as $\varphi_0, \varphi_1, \ldots$. When we use the sequence X as an oracle for the Turing functional Φ, we write Φ^X. Finally, we note that our notation for stages of computations using Turing functionals carries over directly from that for stages of computations using partial computable functions: we write $\Phi_e^X(n)[s]$ just as we would have written $\varphi_e(n)[s]$.

We say that A is *computable from*, or *Turing reducible to* B ($A \leq_T B$) if there is some Turing functional that, given B as an oracle, can compute the characteristic function of A. We then use this reducibility to form equivalence classes of sets that we call the *Turing degrees*: A and B have the same Turing degree if $A \leq_T B$ and $B \leq_T A$. This allows us to talk about properties related to a set's computational strength and not its particular members (for instance, we can talk about $\boldsymbol{0}$, the Turing degree of the computable sets, rather than "the Turing degree of the empty set"). The Turing degrees will be denoted by boldface lowercase Latin letters such as $\boldsymbol{d}$.

We will also use relativization to define new sets and Turing degrees. For instance, for each set A, we define A' to be $\{n \mid \Phi_n^A(n)\downarrow\}$ and call it the *jump* of A. The jump of the empty set, $\emptyset'$, is therefore the set of all natural numbers n such that $\{n \mid \Phi_n^{\emptyset}(n)\downarrow\}$, or, in other words, the indices of those Turing functions that halt given their own index as input and no additional information. Its Turing degree, $\boldsymbol{0}'$, is the degree of the famous Halting Problem (see Chapter II.2 of [83]). Since $A <_T A'$ for any set A, we can develop an infinite ascending chain $\boldsymbol{0} <_T \boldsymbol{0}' <_T \boldsymbol{0}'' <_T \ldots$ of Turing degrees. We note quickly that in general, the n^{th} jump of A is written as $A^{(n)}$.

Other, stronger reducibilities and their corresponding degree structures have also been found to be useful in the study of randomness. Turing reductions are not required to converge on any input and, when they do converge, the size of the elements of the oracle queried during the computation is not necessarily bounded by any reasonable function. The next type of reducibility is *weak truth-table reducibility*, or wtt-reducibility. A wtt-reduction Φ^A is a Turing reduction in which the computation of $\Phi^A(n)$, should it halt, is carried out by querying only the first $f(n)$ bits of A for a given computable function f. Finally, the last such reducibility we will discuss in this survey is *truth-table reducibility*, or tt-reducibility: a tt-reduction Φ^A is one that will converge at every input given any oracle A.[2] It can be seen that

$$A \leq_{tt} B \Longrightarrow A \leq_{wtt} B \Longrightarrow A \leq_T B$$

but that none of these implications reverse.

As with Turing reducibility, we can also create equivalence classes of mutually wtt- or tt-computable sets and study the wtt- and tt-degrees. We can then ask about the properties of all of these structures – the Turing degrees, for instance, form an upper semilattice – or types of substructures within these structures, such as ideals, which are subsets that are both downward closed and closed under join, or an interval between two degrees (for instance, the interval $[\boldsymbol{0}, \boldsymbol{0}']$ in the Turing degrees). We can also discuss relationships between individual

[2] While this was not the original definition of a tt-reduction, it is perhaps the most intuitive. The original definition of a tt-reduction explains its name and can be found in Chapter III.3 of [83].

degrees; for example, we say that two degrees $\boldsymbol{c}$ and $\boldsymbol{d}$ form a *minimal pair* in their degree structure if the only degree they both compute is $\boldsymbol{0}$.

It is often useful to characterize a subset of ω in terms of the number of unbounded quantifiers required to define it. A Σ_n set A has a membership relation that can be defined from a computable relation $R(x_1, x_2, \ldots, x_n, y)$ using n alternating quantifiers, starting with an existential one: $y \in A$ if and only if

$$\exists x_1 \forall x_2 \exists x_3 \ldots Q_n x_n (R(x_1, x_2, \ldots, x_n, y)).$$

Q_n will be an existential quantifier if n is odd and a universal quantifier if n is even. We observe that we can consider these quantifiers to be strictly alternating. For instance, if we had the membership relation

$$\exists x_1 \exists x_2 (R(x_1, x_2, y)),$$

we could use a computable pairing function $p : \omega^2 \to \omega$ and express the same relation as

$$\exists x \exists x_1 \leq x \exists x_2 \leq x (x = p(x_1, x_2) \wedge R(x_1, x_2, y))$$

instead; note that the second and third existential quantifiers in this formula are bounded and therefore that

$$\exists x_1 \leq x \exists x_2 \leq x (x = p(x_1, x_2) \wedge R(x_1, x_2, y))$$

is a computable relation.

Example 1.1. $\emptyset'$ is Σ_1: $e \in \emptyset'$ if and only if

$$\exists s (\Phi_e^{\emptyset}(e)[s]\downarrow).$$

Example 1.2. The set of all e such that W_e is finite, Fin, is Σ_2: e belongs to Fin if and only if

$$\exists m \forall s \forall k (k > m \to k \notin W_e[s]).$$

A Π_n set is defined in a similar way. It will also have n alternating quantifiers, but this time starting with a universal quantifier; we note that the complement of a Σ_n set is a Π_n set and vice versa.

Example 1.3. The set of all e such that φ_e is total, Tot, is Π_2: e belongs to Tot if and only if

$$\forall k \exists s \exists m (\varphi_e(k)[s] = m).$$

Finally, we have the Δ_n sets, which we define to be those sets that can be characterized in both a Σ_n and a Π_n way. Since we often refer to the class of Σ_n sets simply as Σ_n (and similarly for the classes of Π_n and Δ_n sets), we can write

$$\Delta_n = \Sigma_n \cap \Pi_n.$$

These classes of sets – the Σ_n, Π_n, and Δ_n sets – form the *arithmetic hierarchy*. We will note some fundamental facts relating these classes to the classes of sets we have already discussed (see Chapter IV.1 of [83]):

1. $\Sigma_0 = \Pi_0 = \Delta_0 = \Delta_1$ is simply the class of computable sets.
2. A set is Σ_1 if and only if it is c.e.
3. A set is Δ_n exactly when it is Turing computable from $\emptyset^{(n-1)}$.

Furthermore, this hierarchy is proper: as long as $n > 0$, we will always have $\Delta_n \subsetneq \Sigma_n$ and $\Delta_n \subsetneq \Pi_n$.

We may describe subsets of natural numbers in ways other than the arithmetic hierarchy, too. For instance, we have the *high* and *low* sets, which are defined based on the usefulness of the sets in question as oracles:

- a low set $A \subseteq \omega$ is one such that $A' \equiv_T 0'$, and
- a high set $A \subseteq \omega$ is one such that $A' \geq_T 0''$.

It is often more useful to define high sets in terms of the domination property discovered by Martin [65]: A set is high if and only if it Turing computes a function f that dominates all computable functions (that is, for each computable g, we have $f(n) \geq g(n)$ for all sufficiently large n).

Similarly, we may consider highness and lowness in the context of tt-reducibility: a set A is *superhigh* if $A' \geq_{tt} 0''$ and *superlow* if $A' \equiv_{tt} 0'$.

Another hierarchy of classes of sets that has proven useful is the genericity hierarchy, which we can compare to a hierarchy of randomness notions that we will see later. Instead of classifying sets based directly on the complexity of their definitions, we classify them in terms of the complexity of the sets they are forced to either meet or avoid.

Definition 1.4. Let S be a set of finite binary strings. We say that an infinite binary sequence A *meets* S if there is some $\sigma \in S$ that is an initial segment of A. Furthermore, we say that A *avoids* S if there is some initial segment of A that is not extended by any element of S.

This gives us the framework necessary to define generic sets once we have also defined a dense set of strings: a set of strings S is *dense* if for every $\tau \in 2^{<\omega}$, there is a $\sigma \in S$ that extends it.

Definition 1.5. A sequence A is *n-generic* if it either meets or avoids every Σ_n set and *weakly n-generic* if it meets every dense Σ_n set.

These classes of sequences once again form a proper hierarchy: every n generic is weakly n-generic, and every weakly $(n+1)$-generic is n-generic.

Other classes of sets whose definitions are less closely tied to the arithmetical hierarchy have also been shown to be useful. For instance, we will make use of the sets of hyperimmune degree, which are defined, once again, using a domination property [77]: a set A has *hyperimmune degree* if it computes a function that is not dominated by *any* computable function. The sets that do not have hyperimmune degree are said to be of *hyperimmune-free degree*; it is worth noting that there are continuum many such sets and that all of them (except the computable sets) are Turing incomparable to $\emptyset'$.

1.3. Core mathematical concepts. Several concepts from classical mathematics will prove useful; we summarize them here. First we will recall some fundamental facts about the Cantor space, 2^ω, as a topological space and as a probability space.

In the Cantor space, our basic open sets have the form $[\sigma]$ for $\sigma \in 2^{<\omega}$: as previously mentioned, $[\sigma]$ is the set of elements of 2^ω that extend σ. In fact, these sets are all clopen, and the clopen sets are the finite unions of these $[\sigma]$'s. Now that we have done this, we can describe the complexity of the open sets that we generate in this way in terms of their generating sets. For instance, we can say that $[S]$ is effectively open if S is c.e., and more generally, we can define the effective Borel hierarchy as we defined the arithmetic hierarchy in the previous subsection: $[S]$ is Σ^0_1 if it is the union of a c.e. sequence of basic open sets, Π^0_1 if it is the complement of a Σ^0_1 set, Σ^0_n for $n > 1$ if it is the union of a sequence of uniformly Π^0_{n-1} sets (that is, a computable sequence of Π^0_{n-1} classes), and so on. It is worth noting at this point that it is customary to refer to a subset of the Cantor space, especially one defined using this hierarchy, as a class.

We can also establish the Lebesgue measure on the Cantor space: the measure of a basic open set $[\sigma]$ is $\mu([\sigma]) = 2^{-|\sigma|}$, and the measure of any other measurable set is determined in the standard way.

We will often identify the Cantor space with the unit interval $(0,1)$ since these spaces are measure-theoretically isomorphic. Here, we make use of the interval topology on $\mathbb{R}$, and our basic open sets are intervals (a, b). We will establish the Lebesgue measure in this context as well, denoted throughout by μ once again [57]. This is the "standard" measure on $\mathbb{R}$, and the Lebesgue measure of such an interval is $\mu((a, b)) = b - a$ for finite a and b.

We can also describe elements of $\mathbb{R}$ using concepts from classical computability theory. In general, we identify a real α in the unit interval with the element A of the Cantor space such that $\alpha = 0.A$.[3] This allows us to say that such a real is computable if the corresponding $A \in 2^\omega$ is; it is equivalent to say that a real α is computable if there is a computable sequence of rationals $\langle q_i \rangle_{i \in \omega}$ converging to it such that $|q_n - \alpha| < 2^{-n}$ for every n [99, 100].

Of course, we would like to extend this to computable enumerability as well: just as a c.e. set is one which we build up from $\emptyset$ by enumerating elements into it, a *left-c.e. real* α is one that is effectively approximable from below; that is, there is a computable, increasing sequence of rationals that converges to α. Correspondingly, a *right-c.e. real* is one that is effectively approximable from above. Equivalently, we could define these in terms of Dedekind cuts: a real α

[3] While some reals may have two representations, this does not matter: such a real will be rational and therefore the corresponding possibilities for A are both computable and have the same computational strength.

is left-c.e. if and only if its left cut $\{q \in \mathbb{Q} \mid q < \alpha\}$ is a c.e. set and right-c.e. if its right cut (defined similarly) is a c.e. set.

We can extend the notions of computability and computable enumerability once more, this time to functions from ω or another computable set to $\mathbb{R}$. To do so, we need to be able to talk about a sequence of reals that is uniformly computable or left-c.e. These definitions are built directly from those of uniformly computable and uniformly c.e. sets:

Definition 1.6. A uniformly computable (left-c.e.) sequence of reals is a sequence $\langle r_i \rangle_{i \in \omega}$ such that there is a computable function $f : \omega^2 \to \mathbb{Q}$ such that for a given i, $\langle f(i,n) \rangle_{n \in \omega}$ is a computable (left-c.e.) approximation for r_i.

Definition 1.7. A function from a computable set to $\mathbb{R}$ is computable if its values are uniformly computable reals, and it is computably enumerable if its values are uniformly left-c.e. reals.

Now we turn our attention to the general mathematical ideas we will need to study randomness properly and place them in the context of computability theory. The first concept, that of a martingale, will be useful when we discuss the predictability framework for randomness. In general, a martingale is a certain type of stochastic process, but here we need only think of it as a type of betting strategy on finite binary strings.

Definition 1.8 ([59]). A function $d : 2^{<\omega} \to \mathbb{R}^{\geq 0}$ is a *martingale* if it obeys the fairness condition

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

for all $\sigma \in 2^{<\omega}$. We say that a martingale d *succeeds* on $A \in 2^{\omega}$ if

$$\limsup_n d(A \upharpoonright n) = \infty,$$

and the *success set* of d, which we will write as $S[d]$, is the set of all sequences upon which d succeeds.

We can think of $d(\sigma)$ as expressing the amount of capital that we have after betting on the initial string σ using the strategy inherent in d (so $d(\langle\rangle)$ is our capital before any bets are placed) and $S[d]$ as the set of sequences that we can make arbitrarily much money betting on if the payout is determined by d. As shown by Ville, $P \subseteq 2^{\omega}$ has Lebesgue measure zero if and only if there is some martingale d such that $P \subseteq S[d]$ [102].

A computable or c.e. martingale is simply a martingale that is, respectively, a computable or c.e. function.

To define the last of the core mathematical concepts in this section, Hausdorff dimension, we must consider a variation of Lebesgue measure on the Cantor space.