

Computational Depth and Reducibility ^{*}

David W. Juedes, James I. Lathrop, and Jack H. Lutz
Department of Computer Science
Iowa State University
Ames, IA 50011

Abstract

This paper reviews and investigates Bennett's notions of strong and weak computational depth (also called logical depth) for infinite binary sequences. Roughly, an infinite binary sequence x is defined to be *weakly useful* if every element of a non-negligible set of decidable sequences is reducible to x in recursively bounded time. It is shown that every weakly useful sequence is strongly deep. This result (which generalizes Bennett's observation that the halting problem is strongly deep) implies that every high Turing degree contains strongly deep sequences. It is also shown that, in the sense of Baire category, almost every infinite binary sequence is weakly deep, but not strongly deep.

Contents

1	Introduction	2
2	Preliminaries	5
3	Measure and Category	7
4	Algorithmic Information and Randomness	14
5	Strong Computational Depth	22
6	Weak Computational Depth	37
7	Conclusion	40

^{*}This research was supported in part by National Science Foundation Grant CCR-9157382, with matching funds from Rockwell International and Microware Systems Corporation.

1 Introduction

Algorithmic information theory, as developed by Solomonoff [51], Kolmogorov [21, 22, 23], Chaitin [9, 10, 11, 12], Martin-Löf [39, 40], Levin [26, 27, 28, 29, 30, 31, 55], Schnorr [47], Gács [15], Shen' [48, 49], and others, gives a satisfactory, quantitative account of the information content of individual binary strings (finite) and binary sequences (infinite). However, a given quantity of information may be organized in various ways, rendering it more or less useful for various computational purposes. In order to quantify the degree to which the information in a computational, physical, or biological object has been organized, Bennett [4, 5] has extended algorithmic information theory by defining and investigating the *computational depth* of binary strings and binary sequences.

Roughly speaking, the computational depth (called “logical depth” by Bennett [4, 5]) of an object is the amount of time required for an algorithm to derive the object from its shortest description. (Precise definitions appear in the sections to follow.) Since this shortest description contains all the information in the object, the depth thus represents the amount of “computational work” that has been “added” to this information and “stored in the organization” of the object. (Depth is closely related to Adleman’s notion of “potential” [1] and Koppel’s notion of “sophistication” [24, 25].)

One way to investigate the computational usefulness of an object is to investigate the class of computational problems that can be solved efficiently, given access to the object. When the object is an infinite binary sequence, i.e., a sequence $x \in \{0, 1\}^\infty$, this typically amounts to investigating the class of binary strings $y \in \{0, 1\}^\infty$ that are Turing reducible to x in some recursive time bound $s : \mathbf{N} \rightarrow \mathbf{N}$. This condition, written $y \leq_T^{\text{DTIME}(s)} x$, means that there is an oracle Turing machine M that, on input $n \in \mathbf{N}$ with oracle x , computes $y[n]$, the n^{th} bit of y , in at most $s(l)$ steps, where l is the number of bits in the binary representation of n . For example, consider the *diagonal halting problem* $\chi_K \in \{0, 1\}^\infty$, whose n^{th} bit $\chi_K[n]$ is 1 if and only if M_n , the n^{th} Turing machine, halts on input n . It is well-known that χ_K is useful, in the sense that every recursive sequence (in fact, every recursively enumerable sequence) $y \in \{0, 1\}^\infty$ is Turing reducible to χ_K in polynomial time.

An interesting feature of this example is that χ_K has relatively low information content. In fact, an n -bit prefix of χ_K , denoted $\chi_K[0..n-1]$, contains only $O(\log n)$ bits of algorithmic information [3]. Intuitively, this is because $\chi_K[0..n-1]$ is completely specified by the *number* of indices

$i \in \{0, \dots, n-1\}$ such that the i^{th} Turing machine M_i halts on input i . Once this $O(\log n)$ -bit number is known, direct simulation of M_0, M_1, \dots, M_{n-1} on inputs $0, 1, \dots, n-1$, respectively, will eventually determine all n bits of $\chi_K[0..n-1]$.

In contrast, consider a sequence $z \in \{0, 1\}^\infty$ that is *algorithmically random* in the equivalent senses of Martin-Löf [39], Levin [26], Schnorr [47], Chaitin [11], Solovay [52], and Shen' [48, 49]. (See section 4 below for a precise definition and basic properties of algorithmic randomness.) An n -bit prefix $z[0..n-1]$ of an algorithmically random sequence z contains approximately n bits of algorithmic information [39], so the information content of z is exponentially greater than that of χ_K . On the other hand, z is much less useful than χ_K , in the following sense. While *every* recursive sequence is Turing reducible to χ_K in polynomial time, a recursive sequence $y \in \{0, 1\}^\infty$ is Turing reducible to z in polynomial time if and only if y is in the complexity class BPP [5, 8]. (The class BPP, defined by Gill [17], consists of those sequences $y \in \{0, 1\}^\infty$ such that there is a randomized algorithm that decides $y[n]$, the n^{th} bit of y , with error probability less than $\frac{1}{n}$, using time that is at most polynomial in the number of bits in the binary representation of n .) Since BPP contains only the simplest recursive sequences, this means that, for the purpose of efficiently deciding recursive sequences, χ_K is much more useful than an algorithmically random sequence z .

Bennett has argued that the computational usefulness of χ_K derives not from its algorithmic information content (which is relatively low), but rather from its computational depth. In support of this thesis, Bennett [5] has proven that χ_K is *strongly deep*, while no algorithmically random sequence can even be *weakly deep*. (Precise definitions of these terms appear in sections 5 and 6 below.)

This paper furthers Bennett's investigation of the computational depth of infinite binary sequences. We pay particular, quantitative attention to interactions between computational depth and time-bounded Turing reductions.

In order to further investigate the above-discussed notion of the computational usefulness of a sequence $x \in \{0, 1\}^\infty$, we quantify the *size* of the set of recursive sequences that are Turing reducible to x within some recursive time bound. For this purpose, let REC be the set of all recursive (i.e., decidable) sequences, and, for a recursive time bound $s : \mathbf{N} \rightarrow \mathbf{N}$, let $\text{DTIME}^x(s)$ be the set of all sequences $y \in \{0, 1\}^\infty$ such that $y \leq_{\text{T}}^{\text{DTIME}^x(s)} x$. We are interested in the size of $\text{DTIME}^x(s) \cap \text{REC}$ as a subset of REC. To quantify

this, we use a special case of the *resource-bounded measure theory* of Lutz [37, 36]. (A detailed description of the relevant special case appears in section 3 below.) Intuitively, this theory, a generalization of classical Lebesgue measure theory, defines a set X of infinite binary sequences to have *measure 0 in REC* if $X \cap \text{REC}$ is a *negligibly small* subset of REC.

In this paper, we define a sequence $x \in \{0, 1\}^\infty$ to be *weakly useful* if there exists a recursive time bound $s : \mathbf{N} \rightarrow \mathbf{N}$ such that $\text{DTIME}^x(s)$ does *not* have measure 0 in REC. Returning to the two examples discussed earlier, χ_K is weakly useful because *every* element of REC is in $\text{DTIME}^{\chi_K}(s)$, provided that s is superpolynomial, e.g. if $s(n) = n^{\log n}$. On the other hand, if z is algorithmically random, then z is *not* weakly useful, by the following two facts.

- (i) For every recursive time bound $s : \mathbf{N} \rightarrow \mathbf{N}$ there exists a recursive time bound $\hat{s} : \mathbf{N} \rightarrow \mathbf{N}$ such that, for all algorithmically random sequences z , $\text{DTIME}^z(s) \cap \text{REC} \subseteq \text{DTIME}(\hat{s})$ [5, 8, 7].
- (ii) For every recursive time bound $\hat{s} : \mathbf{N} \rightarrow \mathbf{N}$, $\text{DTIME}(\hat{s})$ has measure 0 in REC [37].

Our main result, Theorem 5.11 below, establishes that *every* weakly useful sequence is strongly deep. This implies that every high Turing degree contains strongly deep sequences (Corollary 5.15). Since the Turing degree of χ_K is one of many high Turing degrees, our main result thus generalizes Bennett's result [5] that χ_K is strongly deep.

More importantly, our main result rigorously confirms Bennett's intuitive arguments relating the computational usefulness of χ_K to its depth. The fact that the useful sequence χ_K is strongly deep is no coincidence. *Every* sequence that is even weakly useful *must* be strongly deep.

Bennett [5] also defines the class of *weakly deep* binary sequences. (As noted by Bennett, this class has been investigated in other guises by Levin and V'jugin [28, 31, 32, 53, 54, 55].) A sequence $x \in \{0, 1\}^\infty$ is weakly deep if there do *not* exist a recursive time bound $s : \mathbf{N} \rightarrow \mathbf{N}$ and an algorithmically random sequence z such that $x \leq_{\text{T}}^{\text{DTIME}(s)} z$. Bennett [5] notes that every strongly deep sequence is weakly deep, but that there exist weakly deep sequences that are not strongly deep. In section 6 below we strengthen the separation between these two notions by proving that, in the sense of Baire category, *almost every* sequence $x \in \{0, 1\}^\infty$ is weakly deep, but not strongly deep. (A self-contained discussion of Baire category appears in

section 3.) Intuitively, this means that weakly deep sequences are “topologically abundant.” (They “cannot be avoided” by one player in a two-person game described in section 3.) In contrast, weakly deep sequences are “probabilistically scarce,” in the sense that, with respect to Lebesgue measure, almost every sequence $x \in \{0, 1\}^\infty$ is algorithmically random [39], hence not weakly deep.

In order to provide a basis for further investigation of Bennett’s fundamental ideas, this paper also includes a self-contained mathematical treatment of the weak and strong computational depth of infinite sequences. In section 2 we introduce our basic terminology and notation. In section 3 we review fundamental ideas of Baire category and measure that are used in our work. In section 4 we give a similar review of algorithmic information and randomness. Section 5 is the main section of the paper. In this section, we present the strong computational depth of infinite binary sequences in a unified, self-contained framework using a convenient family of parametrized depth classes, D_g^t . This framework is used to prove our main result (Theorem 5.11), that every weakly useful sequence is strongly deep. In the course of our development, we prove several results, some of which were already proven by Bennett [5], giving precise, quantitative relationships among depth, randomness, and recursiveness. We also prove (Theorem 5.16) that strongly deep sequences are extremely rare, in that they form a meager, measure 0 subset of $\{0, 1\}^\infty$. In section 6 we give a brief discussion of weak computational depth, including a proof that, in the sense of Baire category, almost every sequence is weakly deep, but not strongly deep. In section 7 we mention possible directions for further research.

2 Preliminaries

We work primarily in the set $\{0, 1\}^\infty$ of all (infinite, binary) *sequences*. We also use the set $\{0, 1\}^*$ of all (finite, binary) *strings*. We write $|x|$ for the length of a string x , and λ for the empty string. The *standard enumeration* of $\{0, 1\}^*$ is the sequence s_0, s_1, \dots , in which shorter strings precede longer ones and strings of the same length are ordered lexicographically.

Given a sequence $x \in \{0, 1\}^\infty$ and $m, n \in \mathbf{N}$ with $m \leq n$, we write $x[m..n]$ for the string consisting of the m^{th} through n^{th} bits of x . In particular, $x[0..n-1]$ is the string consisting of the first n bits of x . We write $x[n]$ for $x[n..n]$, the n^{th} bit of x .

We write $\llbracket \varphi \rrbracket$ for the Boolean value of a condition φ , i.e.,

$$\llbracket \varphi \rrbracket = \begin{cases} 1 & \text{if } \varphi \text{ is true} \\ 0 & \text{if } \varphi \text{ is false} \end{cases}$$

The *characteristic sequence* of a set $A \subseteq \mathbf{N}$ is then the sequence $\chi_A \in \{0, 1\}^\infty$ defined by $\chi_A[n] = \llbracket n \in A \rrbracket$ for all $n \in \mathbf{N}$.

We say that a condition $\varphi(n)$ holds *infinitely often (i.o.)* if it holds for infinitely many $n \in \mathbf{N}$. We say that a condition $\varphi(n)$ holds *almost everywhere (a.e.)* if it holds for all but finitely many $n \in \mathbf{N}$.

All logarithms in this paper are base-2 logarithms.

Given a function $f : \mathbf{N}^n \times \{0, 1\}^* \rightarrow Y$ and an n -tuple $\vec{k} \in \mathbf{N}^n$, we define the function $f_{\vec{k}} : \{0, 1\}^* \rightarrow Y$ by $f_{\vec{k}}(x) = f(\vec{k}, x)$ for all $x \in \{0, 1\}^*$. This enables us to regard the function f as a “uniform enumeration” of the functions $f_{\vec{k}}$.

Although we introduce a very specific Turing machine model to define algorithmic information, algorithmic probability, and algorithmic depth in sections 4 and 5, we assume that the reader is already familiar with the general ideas of Turing machine computation, including computation by oracle Turing machines. (Discussion of such machines may be found in many texts, e.g., [2, 19, 44, 50].)

Given a recursive time bound $s : \mathbf{N} \rightarrow \mathbf{N}$, we say that an oracle Turing machine M is *s-time-bounded* if, given any input $n \in \mathbf{N}$ and oracle $y \in \{0, 1\}^\infty$, M outputs a bit $M^y(n) \in \{0, 1\}$ in at most $s(l)$ steps, where l is the number of bits in the binary representation of n . In this case, if $x \in \{0, 1\}^\infty$ satisfies $x[n] = M^y(n)$ for all $n \in \mathbf{N}$, then we say that x is *Turing reducible to y in time s via M* , and we write $x \leq_T^{\text{DTIME}(s)} y$ via M . We say that x is *Turing reducible to y in time s* , and we write $x \leq_T^{\text{DTIME}(s)} y$, if there is some oracle Turing machine M such that $x \leq_T^{\text{DTIME}(s)} y$ via M . For $y \in \{0, 1\}^\infty$ and $s : \mathbf{N} \rightarrow \mathbf{N}$, we write

$$\text{DTIME}^y(s) = \left\{ x \in \{0, 1\}^\infty \mid x \leq_T^{\text{DTIME}(s)} y \right\}.$$

(Note that the time bound here is “sharp”; there is no “big-O.”) The unrelativized complexity class $\text{DTIME}(s)$ is then defined to be $\text{DTIME}^{0^\infty}(s)$, where 0^∞ is the sequence consisting entirely of 0’s.

A sequence $x \in \{0, 1\}^\infty$ is *truth-table reducible* to a sequence $y \in \{0, 1\}^\infty$, and we write $x \leq_{\text{tt}} y$, if there exists a recursive time bound $s : \mathbf{N} \rightarrow \mathbf{N}$ such that $x \leq_T^{\text{DTIME}(s)} y$. (This definition is easily seen to be equivalent to

standard textbook definitions of truth-table reducibility [44, 50].) Given a set $Y \subseteq \{0, 1\}^\infty$, we write

$$\begin{aligned} \text{REC}_{\text{tt}}(Y) &= \{x \in \{0, 1\}^\infty \mid (\exists y \in Y) x \leq_{\text{tt}} y\} \\ &= \bigcup_{\text{recursive } s} \bigcup_{y \in Y} \text{DTIME}^y(s). \end{aligned}$$

We write REC for the set of all recursive (i.e., decidable) sequences $x \in \{0, 1\}^\infty$. Note that $\text{REC} \cup Y \subseteq \text{REC}_{\text{tt}}(Y)$ for all sets $Y \subseteq \{0, 1\}^\infty$. A sequence $x \in \{0, 1\}^\infty$ is *Turing reducible* to a sequence $y \in \{0, 1\}^\infty$, and we write $x \leq_{\text{T}} y$, if there is an oracle Turing machine M such that $M^y(n) = x[n]$ for every $n \in \mathbb{N}$. Two sequences $x, y \in \{0, 1\}^\infty$ are *Turing equivalent*, and we write $x \equiv_{\text{T}} y$, if $x \leq_{\text{T}} y$ and $y \leq_{\text{T}} x$. A *Turing degree* is an equivalence class of $\{0, 1\}^\infty$ under the equivalence relation \equiv_{T} .

The *complement* of a set $X \subseteq \{0, 1\}^\infty$ is $X^c = \{0, 1\}^\infty - X$.

3 Measure and Category

Three different senses in which a set X of binary sequences may or may not be “small” are used in this paper. A set $X \subseteq \{0, 1\}^\infty$ may have *measure 0*, in which case it is small “in the sense of Lebesgue measure.” A set $X \subseteq \{0, 1\}^\infty$ may have *measure 0 in REC*, in which case $X \cap \text{REC}$ is a small subset of REC , “in the sense of resource-bounded measure.” Finally, a set $X \subseteq \{0, 1\}^\infty$ may be *meager* (also known as *first category*), in which case it is small “in the sense of Baire category.” This section reviews the basic ideas from Lebesgue measure, resource-bounded measure, and Baire category that are involved in our use of these three notions of “smallness.” The interested reader may consult [6, 18, 36, 37, 43, 45] for further discussion of these notions, but the material in the present section is sufficient for following the arguments of this paper.

Resource-bounded measure [36, 37] is a generalization of classical Lebesgue measure. As such it has classical Lebesgue measure and measure in REC as special cases. We use this fact to present the notions “measure 0” and “measure 0 in REC ” more or less simultaneously.

Consider the random experiment in which a binary sequence $x \in \{0, 1\}^\infty$ is chosen probabilistically, using an independent toss of a fair coin to decide each bit of x . Intuitively, a set $X \subseteq \{0, 1\}^\infty$ has (Lebesgue) *measure 0*—a condition defined precisely below—if $\Pr[x \in X] = 0$, where $\Pr[x \in X]$ is the probability that x , the outcome of the coin-tossing experiment, is an

element of X . In this case, we write $\mu(X) = 0$ (“ X has measure 0”). We now develop the necessary definitions.

A string $w \in \{0, 1\}^*$ is a *prefix* of a string or sequence $x \in \{0, 1\}^* \cup \{0, 1\}^\infty$, and we write $w \sqsubseteq x$, if there exists $y \in \{0, 1\}^* \cup \{0, 1\}^\infty$ such that $x = wy$. The *cylinder generated by* a string $w \in \{0, 1\}^*$ is

$$\mathbf{C}_w = \{x \in \{0, 1\}^\infty \mid w \sqsubseteq x\},$$

i.e., the set of all *infinite* binary sequences beginning with the string w .

Definition [37]. A *density function* is a function $d : \{0, 1\}^* \rightarrow [0, \infty)$ satisfying

$$d(w) = \frac{d(w0) + d(w1)}{2} \quad (3.1)$$

for all $w \in \{0, 1\}^*$. The *global value* of a density function d is $d(\lambda)$. The *set covered by* a density function d is

$$S[d] = \bigcup_{\substack{w \in \{0, 1\}^* \\ d(w) \geq 1}} \mathbf{C}_w. \quad (3.2)$$

An *n -dimensional density system* (*n -DS*) is a function

$$d : \mathbf{N}^n \times \{0, 1\}^* \rightarrow [0, \infty)$$

such that, for all $\vec{k} \in \mathbf{N}^n$, the function $d_{\vec{k}}$ is a density function. (Recall that $d_{\vec{k}}(w) = d(\vec{k}, w)$ for all $\vec{k} \in \mathbf{N}^n$ and $w \in \{0, 1\}^*$.)

Taken together, parts (3.1) and (3.2) of the above definition imply that

$$\Pr[x \in S[d]] \leq d(\lambda)$$

in our coin-tossing random experiment. We thus intuitively regard d as a “detailed verification” that $\Pr[x \in X] \leq d(\lambda)$ for all $X \subseteq S[d]$. With this intuition in mind, we present the central idea of resource-bounded measure 0 sets.

Definition [37]. A *null cover* of a set $X \subseteq \{0, 1\}^\infty$ is a 1-DS d that satisfies the following two conditions for all $k \in \mathbf{N}$.

- (i) $X \subseteq S[d_k]$.

(ii) $d_k(\lambda) \leq 2^{-k}$.

Definition [37]. A set $X \subseteq \{0, 1\}^\infty$ has (*Lebesgue*) *measure 0*, and we write $\mu(X) = 0$, if it has a null cover. A set $X \subseteq \{0, 1\}^\infty$ has (*Lebesgue*) *measure 1*, and we write $\mu(X) = 1$, if $\mu(X^c) = 0$. In this latter case, we say that X contains *almost every* sequence $x \in \{0, 1\}^\infty$.

It is a routine exercise to check that this definition is equivalent to “standard textbook” definitions [6, 18, 43, 45] of measure 0 and measure 1 sets.

The main advantage of the above definition is that it naturally yields analogous notions of measure in REC and various complexity classes. To specify the analogous measure in REC, we need to define the computability of density systems. Since density systems are real-valued, they must be computed via approximations. For this purpose, it is natural to use the set

$$\mathbf{D} = \{m2^{-n} \mid m \in \mathbf{Z}, n \in \mathbf{N}\}$$

of *dyadic rationals*. These are real numbers whose standard binary representations are finite.

Definition [37]. An n -DS d is *computable* if there is a total recursive function $\hat{d} : \mathbf{N}^{n+1} \times \{0, 1\}^* \rightarrow \mathbf{D}$ such that, for all $\vec{k} \in \mathbf{N}^n$, $r \in \mathbf{N}$, and $w \in \{0, 1\}^*$,

$$\left| \hat{d}_{\vec{k}, r}(w) - d_{\vec{k}}(w) \right| \leq 2^{-r}.$$

Note that the above definition is uniform, in the sense that it requires a single total recursive function \hat{d} to compute approximations for all the density functions $d_{\vec{k}}$ (given \vec{k} , a precision parameter r , and the input to $d_{\vec{k}}$ as inputs to \hat{d}).

Definition [37]. A *recursive null cover* of a set $X \subseteq \{0, 1\}^\infty$ is a null cover of X that is computable. A set $X \subseteq \{0, 1\}^\infty$ has *recursive measure 0*, and we write $\mu_{\text{rec}}(X) = 0$, if X has a recursive null cover. A set $X \subseteq \{0, 1\}^\infty$ has *recursive measure 1*, and we write $\mu_{\text{rec}}(X) = 1$, if $\mu_{\text{rec}}(X^c) = 0$. A set $X \subseteq \{0, 1\}^\infty$ has *measure 0 in REC*, and we write $\mu(X \mid \text{REC}) = 0$, if $\mu_{\text{rec}}(X \cap \text{REC}) = 0$. A set $X \subseteq \{0, 1\}^\infty$ has *measure 1 in REC*, and we write $\mu(X \mid \text{REC}) = 1$, if $\mu(X^c \mid \text{REC}) = 0$. In this latter case, we say that X contains *almost every* recursive sequence $x \in \text{REC}$.

Note that the implications

$$\begin{array}{ccc}
& \mu_{\text{rec}}(X)=0 & \\
\swarrow & & \searrow \\
\mu(X)=0 & & \mu(X|\text{REC})=0
\end{array}
\quad \text{and} \quad
\begin{array}{ccc}
& \mu_{\text{rec}}(X)=1 & \\
\swarrow & & \searrow \\
\mu(X)=1 & & \mu(X|\text{REC})=1
\end{array}$$

all follow immediately from the above definitions. It is easy to see that every subset of a recursive measure 0 set has recursive measure 0, that every finite subset of REC has recursive measure 0, and that every finite union of recursive measure 0 sets has recursive measure 0. In fact, the recursive measure 0 sets enjoy a stronger closure property, which we now define.

Definition [37]. Let $Z, Z_0, Z_1, \dots \subseteq \{0, 1\}^\infty$. Then Z is a *recursive union of the sets* Z_0, Z_1, \dots of measure 0 in REC if $Z = \bigcup_{j=0}^{\infty} Z_j$ and there exists a computable 2-DS d such that, for all $j \in \mathbf{N}$, d_j is a recursive null cover of $Z_j \cap \text{REC}$.

Theorem 3.1 (Lutz [37]). If $Z \subseteq \{0, 1\}^\infty$ is a recursive union of sets of measure 0 in REC, then Z has measure 0 in REC.

On the other hand, the following result shows that not every set has measure 0 in REC.

Theorem 3.2 (Lutz [37]). No cylinder \mathbf{C}_w has measure 0 in REC. In particular, REC does not have measure 0 in REC.

Taken together, the above facts justify the intuition that, if X has measure 0 in REC, then $X \cap \text{REC}$ is a *negligibly small* subset of REC. Further discussion of this intuition may be found in [37, 43].

Other formulations of measure in REC have been investigated by Freidzon [14], Mehlhorn [41], and others. The advantage of the formulation here is that it uniformly yields Lebesgue measure, measure in REC, and measure in various complexity classes [37]. It is easy to show that, if X has “measure 0 in REC” in the sense of [14], then X has measure 0 in REC in our sense.

We now turn to the fundamentals of Baire category. Baire category gives a topological notion of smallness, usually defined in terms of “countable unions of nowhere dense sets” [42, 43, 45]. Here it is more convenient to define Baire category in terms of certain two-person, infinite games of perfect information, called Banach-Mazur games.

Informally, a Banach-Mazur game is an infinite game in which two players construct a sequence $x \in \{0, 1\}^\infty$ by taking turns extending a prefix of x . There is a “payoff set” $X \subseteq \{0, 1\}^\infty$ such that Player I wins a play of the game if $x \in X$ and Player II wins otherwise.

More formally, a *strategy* for a Banach-Mazur game is a function $\sigma : \mathbf{N} \times \{0, 1\}^* \rightarrow \{0, 1\}^*$ with the property that $w \sqsubset \sigma_m(w)$, i.e., w is a proper prefix of $\sigma_m(w)$ for all $m \in \mathbf{N}$ and $w \in \{0, 1\}^*$. A *play* of a Banach-Mazur game is an ordered pair (σ, τ) of strategies. The *result* of the play (σ, τ) is the unique sequence $R(\sigma, \tau) \in \prod_{k=0}^{\infty} \mathbf{C}_{w_k}$, where the strings w_0, w_1, \dots are defined by the following recursion.

- (i) $w_0 = \lambda$.
- (ii) For all $m \in \mathbf{N}$, $w_{2m+1} = \sigma_m(w_{2m})$.
- (iii) For all $m \in \mathbf{N}$, $w_{2m+2} = \tau_m(w_{2m+1})$.

Intuitively, Player I uses strategy σ , Player II uses strategy τ , and w_k is the prefix of $R(\sigma, \tau)$ that has been constructed when the two players have moved a total of k times. For example, if σ and τ are defined by

$$\sigma_m(w) = w0^{m+1}, \tau_m(w) = w1,$$

then

$$w_0 = \lambda, w_1 = 0, w_2 = 01, w_3 = 0100, \dots,$$

so

$$R(\sigma, \tau) = 01001000100001000001 \dots$$

We write $G[X]$ for the Banach-Mazur game with payoff set $X \subseteq \{0, 1\}^\infty$. A *winning strategy for Player I* in $G[X]$ is a strategy σ such that, for all strategies τ , $R(\sigma, \tau) \in X$. A *winning strategy for Player II* in $G[X]$ is a strategy τ such that, for all strategies σ , $R(\sigma, \tau) \notin X$.

Definition. A set $X \subseteq \{0, 1\}^\infty$ is *meager* if there exists a winning strategy for Player II in the Banach-Mazur game $G[X]$. A set $X \subseteq \{0, 1\}^\infty$ is *comeager* if X^c is meager. (A meager set is sometimes called a “set of first category.”)

As an easy example, let FIN be the set of all characteristic sequences of finite subsets of \mathbf{N} , i.e.,

$$\text{FIN} = \{x \in \{0, 1\}^\infty \mid x \text{ has only finitely many 1's}\}.$$

Then the strategy τ defined by $\tau_m(w) = w1$ is a winning strategy for Player II in $G[\text{FIN}]$, so FIN is meager.

The proof that the above definition is equivalent to the “standard textbook” definition of the meager sets is due to Banach and may be found in [42] or [43]. It is clear that every subset of a meager set is meager and that every countable set $X \subseteq \{0, 1\}^\infty$ is meager. In fact, it is well-known that every countable union of meager sets is meager [43]. On the other hand, for every $w \in \{0, 1\}^*$, the strategy

$$\sigma_m(u) = \begin{cases} w & \text{if } u \not\sqsubseteq w \\ u0 & \text{otherwise} \end{cases}$$

is a winning strategy for Player I in $G[\mathbf{C}_w]$, so no cylinder is meager. (This is the Baire Category Theorem [43].) These facts justify the intuition that meager sets are “topologically small,” or (*negligibly*) *small in the sense of Baire category*. Thus, if a set $X \subseteq \{0, 1\}^\infty$ is comeager, we say that its elements are “topologically abundant,” or that X is *large in the sense of Baire category*, or that X contains *almost every sequence in the sense of Baire category*.

The proofs of our Baire category results, Theorems 5.16 and 6.2 below, are easy, given some elementary properties of the Cantor topology on the set $\{0, 1\}^\infty$. For completeness, we review these properties. Further details may be found in a number of texts, e.g., [20, 42].

A set $X \subseteq \{0, 1\}^\infty$ is *open*, or Σ_1^0 , if it can be expressed as a (countable) union of cylinders. A set $X \subseteq \{0, 1\}^\infty$ is *closed*, or Π_1^0 , if X^c is open. For each positive integer k , a set $X \subseteq \{0, 1\}^\infty$ is Σ_{k+1}^0 if it can be expressed as a countable union of Π_k^0 sets. For each positive integer k , a set $X \subseteq \{0, 1\}^\infty$ is Π_{k+1}^0 if X^c is Σ_{k+1}^0 . (The “boldface” classes $\Sigma_1^0, \Pi_1^0, \Sigma_2^0, \Pi_2^0, \dots$ are collectively known as the *finite Borel hierarchy*. This hierarchy is closely analogous to the “lightface” arithmetical hierarchy $\Sigma_1^0, \Pi_1^0, \Sigma_2^0, \Pi_2^0, \dots$ of recursion theory [42].)

A *finite variation* of a sequence $x \in \{0, 1\}^\infty$ is a sequence $y \in \{0, 1\}^\infty$ such that $y[n] = x[n]$ for all but finitely many $n \in \mathbf{N}$. A set $X \subseteq \{0, 1\}^\infty$ is *closed under finite variations* if every finite variation of every element of X is an element of X .

A function $f : \{0, 1\}^\infty \rightarrow \{0, 1\}^\infty$ is *continuous* if, for every $x \in \{0, 1\}^\infty$ and $n \in \mathbf{N}$, there exists $k \in \mathbf{N}$ such that $f(\mathbf{C}_{x[0..k-1]}) \subseteq \mathbf{C}_{f(x)[0..n-1]}$.

We use the following two facts. For completeness, we sketch proofs. Further details may be found in standard texts, e.g., [20, 42].

Fact 3.3.

1. Let X and Y be disjoint subsets of $\{0, 1\}^\infty$. If X is Σ_2^0 , $Y \neq \emptyset$, and Y is closed under finite variations, then X is meager.

2. If $X \subsetneq \{0, 1\}^\infty$ is Σ_2^0 and closed under finite variations, then X is meager.

Proof. To prove part 1, assume the hypothesis and fix a sequence $z \in Y$. Since X is Σ_2^0 , there exist closed sets $X_0, X_1, \dots \subseteq \{0, 1\}^\infty$ such that $X = \bigcup_{k=0}^{\infty} X_k$. To see that X is meager, it suffices to exhibit a winning strategy for Player II in the Banach-Mazur game $G[X]$. Player II's strategy uses z as a source of bits. To specify this strategy, let $w_k \in \{0, 1\}^*$ be the string constructed by the game play prior to move k of Player II, where $k \in \mathbf{N}$. Let $w_k//z$ be the sequence obtained from z by putting w_k in place of the first $|w_k|$ bits of z . Since $z \in Y$ and $w_k//z$ is a finite variation of z , it must be the case that $w_k//z \in Y$. In particular, this implies that $w_k//z \notin X_k$. Since X_k is closed, it follows that there exists $n > |w_k|$ such that $\mathbf{C}_{(w_k//z)[0..n-1]} \cap X_k = \emptyset$. Player II's strategy in move k is to extend w_k to $(w_k//z)[0..n-1]$ for this value of n . The final sequence $x \in \{0, 1\}^\infty$ constructed by the game play is now guaranteed to satisfy $x \notin X_k$. Since Player II eventually establishes this for every $k \in \mathbf{N}$, it follows that $x \notin X$. Hence this is a winning strategy for Player II in $G[X]$, so X is meager.

To prove part 2, take $Y = X^c$ in part 1. \square

Fact 3.4. If $X \subseteq \{0, 1\}^\infty$ is Σ_2^0 and $f : \{0, 1\}^\infty \rightarrow \{0, 1\}^\infty$ is continuous, then the image $f(X)$ is also Σ_2^0 .

Proof. Assume the hypothesis. Then there exist closed sets $Y_0, Y_1, \dots \subseteq \{0, 1\}^\infty$ such that $X = \bigcup_{k=0}^{\infty} Y_k$. Each Y_k is a closed subset of the compact Hausdorff space $\{0, 1\}^\infty$, so each Y_k is compact. Since f is continuous, it follows that each $f(Y_k)$ is compact, hence closed. Since $f(X) = \bigcup_{k=0}^{\infty} f(Y_k)$, this implies that $f(X)$ is Σ_2^0 . \square

We have described three notions of smallness in this section. It should be noted that no two of them coincide. Although some sets (e.g. finite sets) are small in all three senses, it is possible for a set to be small in any one of these senses without being small in the other two. For example, in section 4 below, we define the set RAND, consisting of all algorithmically

random sequences. Consider also the set REC of all recursive sequences. It is well-known [39] that $\text{REC} \cap \text{RAND} = \emptyset$, that RAND is meager, and that RAND has measure 1. (See also Theorems 4.7 and 6.2 below.) Also, since REC is countable, REC is meager and has measure 0. The following three things follow easily from these observations.

- (a) $\text{RAND} \cup \text{REC}$ is meager, but has measure 1 and measure 1 in REC .
- (b) REC^c has measure 0 in REC but is comeager and has measure 1.
- (c) RAND^c has measure 0, but is comeager and has measure 1 in REC .

As Oxtoby [43] has noted, “There is of course nothing paradoxical in the fact that a set that is small in one sense may be large in some other sense.”

4 Algorithmic Information and Randomness

In this section we review some fundamentals of algorithmic information theory that are used in this paper. We are especially concerned with self-delimiting Kolmogorov complexity and algorithmic randomness. The interested reader is referred to [33, 35] for more details, discussion, and proofs.

Kolmogorov complexity, also called program-size complexity, was discovered independently by Solomonoff [51], Kolmogorov [21], and Chaitin [9]. Self-delimiting Kolmogorov complexity is a technical improvement of the original formulation that was developed independently, in slightly different forms, by Levin [26, 27], Schnorr [47], and Chaitin [11]. The advantage of the self-delimiting version is that it gives precise characterizations of algorithmic probability and randomness.

Self-delimiting Kolmogorov complexity employs a slightly restricted model of (deterministic) Turing machine computation. In this model, a Turing machine M has a program tape, an output tape, and some number k of worktapes. (For some purposes it is also advantageous to have a special input tape, but we do not need one here.) Only 0’s, 1’s and blanks can ever appear on a tape. The program tape and the output tape are infinite to the right, while the worktapes are infinite in both directions. Each tape has a scanning head. The program and output tape heads cannot move left, but the worktape heads can move left or right. The program tape is read-only, the output tape is write-only, and the worktapes are read/write. The output tape head can only write 0’s and 1’s; it cannot write blanks.

A Turing machine M starts in the initial state with a program $\pi \in \{0, 1\}^*$ on its program tape, the output tape blank, and the worktapes blank. The leftmost cell of the program tape is blank, with the program tape head initially scanning this cell. The program π lies immediately to the right of this cell. The rest of the program tape is blank. The output tape head initially scans the leftmost cell of the output tape.

If, after finitely many steps, M halts with the program tape head scanning the last bit of π , then the computation is deemed to be a *success*, we write $M(\pi)\downarrow$, and the *output* of the computation is the string $M(\pi) \in \{0, 1\}^*$ that has been written on the output tape. Otherwise, the computation is a *failure*, we write $M(\pi)\uparrow$, and there is no output (i.e., we disregard the contents of the output tape). If $M(\pi)\downarrow$, then $time_M(\pi)$ denotes the number of steps executed in this computation. If $M(\pi)\uparrow$, then we write $time_M(\pi) = \infty$.

It should be emphasized that a successful computation must end with the program tape head scanning the last bit of the program. Since the program tape head is read-only and cannot move left, this implies that, for every Turing machine M , the set

$$\text{PROG}_M = \{\pi \in \{0, 1\}^* \mid M(\pi)\downarrow\}$$

must be an *instantaneous code*, i.e., must be a set of nonempty strings, no one of which is a prefix of another. (It is this feature of the model that the adjective “self-delimiting” describes.) It follows by Kraft’s inequality (see [13], for example) that, for all Turing machines M ,

$$\sum_{\pi \in \text{PROG}_M} 2^{-|\pi|} \leq 1.$$

It is well-known that there are Turing machines U that are *universal*, in the sense that, for every Turing machine M , there exists a program prefix $\pi_M \in \{0, 1\}^*$ such that, for all $\pi \in \{0, 1\}^*$,

$$U(\pi_M\pi) = M(\pi).$$

(This condition means that $M(\pi)\downarrow$ if and only if $U(\pi_M\pi)\downarrow$, in which case $U(\pi_M\pi) = M(\pi)$.) Furthermore, there are universal Turing machines U that are *efficient*, in the sense that, for each Turing machine M there is a constant $c \in \mathbf{N}$ (which depends on M) such that, for all $\pi \in \{0, 1\}^*$,

$$time_U(\pi_M\pi) \leq c(1 + time_M(\pi) \log time_M(\pi)).$$

Notational Convention. Throughout this paper, U is a fixed, efficient, universal Turing machine.

The *set of programs* for a string $x \in \{0, 1\}^*$ relative to a Turing machine M is

$$\text{PROG}_M(x) = \{\pi \in \{0, 1\}^* \mid M(\pi) = x\}.$$

Similarly, given a time bound $t : \mathbf{N} \rightarrow \mathbf{N}$, the *set of t -fast programs* for x relative to M is

$$\text{PROG}_M^t(x) = \{\pi \in \text{PROG}_M(x) \mid \text{time}_M(\pi) \leq t(|x|)\}.$$

(Note that the time bound here is computed in terms of the output length.) We write PROG , $\text{PROG}(x)$, and $\text{PROG}^t(x)$ for PROG_U , $\text{PROG}_U(x)$, and $\text{PROG}_U^t(x)$, respectively.

We define the *probability* of an instantaneous code $I \subseteq \{0, 1\}^*$ to be

$$\Pr(I) = \sum_{w \in I} 2^{-|w|}.$$

Intuitively, if we choose a sequence $x \in \{0, 1\}^\infty$ probabilistically, using an independent toss of a fair coin to decide each bit of x , then $\Pr(I)$ is the probability that $x \in \bigcup_{w \in I} \mathbf{C}_w$, i.e., the probability that some element of I is a prefix of x .

We now come to the central ideas of algorithmic information theory. (See [33] for a history of the development of these definitions.)

Definition. Let $x \in \{0, 1\}^*$, let $t : \mathbf{N} \rightarrow \mathbf{N}$ be a time bound, and let M be a Turing machine.

1. The (*self-delimiting*) *Kolmogorov complexity* of x relative to M is

$$K_M(x) = \min \left\{ |\pi| \mid \pi \in \text{PROG}_M(x) \right\}.$$

(Here we use the convention that $\min \emptyset = \infty$.) The (*self-delimiting*) *Kolmogorov complexity* of x is

$$K(x) = K_U(x)$$

The quantity $K(x)$ is also called the *algorithmic entropy*, or *algorithmic information content*, of x .

2. The *t-time-bounded (self-delimiting) Kolmogorov complexity* of x relative to M is

$$K_M^t(x) = \min \left\{ |\pi| \mid \pi \in \text{PROG}_M^t(x) \right\}.$$

The *t-time-bounded (self-delimiting) Kolmogorov complexity*, or *t-time-bounded algorithmic entropy*, of x is

$$K^t(x) = K_U^t(x).$$

3. The *algorithmic probability* of x relative to M is

$$\mathbf{m}_M(x) = \Pr(\text{PROG}_M(x)).$$

The *algorithmic probability* of x is

$$\mathbf{m}(x) = \mathbf{m}_U(x).$$

4. The *t-time-bounded algorithmic probability* of x relative to M is

$$\mathbf{m}_M^t(x) = \Pr(\text{PROG}_M^t(x)).$$

The *t-time-bounded algorithmic probability* of x is

$$\mathbf{m}^t(x) = \mathbf{m}_U^t(x).$$

In general, we omit the adjective “self-delimiting”, since this is the only type of Kolmogorov complexity in this paper.

We now present some basic properties of Kolmogorov complexity and algorithmic probability that are used in this paper. The first is obvious, well-known, and useful.

Lemma 4.1. There is a constant $c_0 \in \mathbf{N}$ such that, for all $x \in \{0, 1\}^*$ and all $\pi \in \text{PROG}(x)$,

$$K(x) \leq K(\pi) + c_0.$$

The next two important theorems express the fundamental relationship between Kolmogorov complexity and algorithmic probability.

Theorem 4.2 (Levin [26, 27], Chaitin [11]). There is a constant $\tilde{c} \in \mathbf{N}$ such that, for all $x \in \{0, 1\}^*$,

$$-\log \mathbf{m}(x) \leq K(x) < -\log \mathbf{m}(x) + \tilde{c}.$$

A straightforward modification of the proof of Theorem 4.2 yields the following time-bounded version. (This result also follows immediately from Lemma 3 of [34].)

Theorem 4.3. Let $t : \mathbf{N} \rightarrow \mathbf{N}$ be recursive.

1. For all $x \in \{0, 1\}^*$,

$$-\log \mathbf{m}^t(x) \leq K^t(x).$$
2. There exist a recursive function $t_1 : \mathbf{N} \rightarrow \mathbf{N}$ and a constant $c_1 \in \mathbf{N}$ such that, for all $x \in \{0, 1\}^*$,

$$K^{t_1}(x) < -\log \mathbf{m}^t(x) + c_1.$$

In addition to the above facts, we need the following lemma and corollary, due to Bennett. For the lemma, say that a string $\pi \in \{0, 1\}^*$ *computes* a finite instantaneous code I if $U(\pi) = [x_0, \dots, x_{n-1}]$ is a binary string that encodes an enumeration of the elements x_0, \dots, x_{n-1} of I in some standard fashion.

Lemma 4.4 (Bennett [5]). There is a constant $c' \in \mathbf{N}$ such that, for all $\pi \in \{0, 1\}^*$, if π computes a finite instantaneous code I , then for all $x \in I$,

$$K(x) \leq |x| + \log \Pr(I) + |\pi| + c'.$$

(Note that $-|x| \leq \log \Pr(I) \leq 0$, so that the bound becomes tighter as $\Pr(I)$ becomes smaller.)

Proof. Let M be a Turing machine that performs as indicated in Figure 1 with program $\pi\hat{\pi}$, where π computes a finite instantaneous code and $\hat{\pi} \in \{0, 1\}^*$. (If the program for M is not of this form, then the computation is a failure.) Since U is a universal Turing machine, there is a program prefix $\pi_M \in \{0, 1\}^*$ such that, for all $\pi \in \{0, 1\}^*$, $U(\pi_M\pi) = M(\pi)$. Let

$$c' = |\pi_M| + 1.$$

To see that c' has the desired property, let $\pi \in \{0, 1\}^*$ compute a finite instantaneous code I . If $I = \emptyset$, then the lemma is affirmed vacuously, so assume that $I \neq \emptyset$. Let x_0, \dots, x_{n-1} and k_0, \dots, k_{n-1} be as in Figure 1. Define real numbers $r_0 < \dots < r_n$ by the recursion

$$r_0 = 0, \quad r_{i+1} = r_i + 2^{-k_i},$$

```

begin
  simulate  $U(\pi)$  to obtain  $I$  (on a worktape) in the form
     $I = \{x_0, \dots, x_{n-1}\}$ ,
  where  $x_0, \dots, x_{n-1}$  are in standard order;
   $\pi' := \lambda$ ;
  for  $0 \leq i < n$  do
    begin
      if  $i = 0$  then  $w := 0^{k_i}$  else  $w := next(w, k_i)$ ,
      where  $k_i = |x_i| - \lfloor -\log \Pr(I) \rfloor$  and  $next(w, k_i)$  is the
      immediate lexicographic successor of the string  $w1^{k_i-|w|}$ ;
      while  $\pi' \sqsubseteq w$  do
        if  $\pi' = w$  then output  $x_i$  and halt
        else  $\pi' := \pi'b$ , where  $b$  is the
          next bit on the program tape
      end
    end
  end  $M(\pi\hat{\pi})$ .

```

Figure 1: The Turing Machine M used in the proof of Lemma 4.4.

and note that

$$r_n = \sum_{i=0}^{n-1} 2^{-k_i} = \sum_{x \in I} 2^{\lfloor -\log \Pr(I) \rfloor - |x|} \leq \Pr(I)^{-1} \sum_{x \in I} 2^{-|x|} = 1.$$

Define strings $\hat{\pi}, \dots, \hat{\pi}_{n-1} \in \{0, 1\}^*$ by

$$\hat{\pi}_0 = \mathbf{0}^{k_0}, \quad \hat{\pi}_{i+1} = \text{next}(\hat{\pi}_i, k_{i+1}),$$

where the function next is defined as in Figure 1. A routine induction on i shows that each $\hat{\pi}_i$ is the standard k_i -bit binary representation of the natural number $r_i \cdot 2^{k_i}$. (The key point in the induction step is that, for $0 \leq i < n-1$, we have $r_i + 2^{-k_i} = r_{i+1} \leq r_{n-1} < r_n \leq 1$, so $r_i \cdot 2^{k_i} < 2^{k_i} - 1$. By the induction hypothesis, this means that $\hat{\pi}_i$ does not consist entirely of 1's, so $\hat{\pi}_{i+1} = \text{next}(\hat{\pi}_i, k_{i+1})$ contains only k_{i+1} bits.) Moreover, it is easily checked that, for all $0 \leq i < n$, $\hat{\pi}_i$ is the value assigned to w by M during iteration i of the for-loop, and that

$$U(\pi_M \pi \hat{\pi}_i) = M(\pi \hat{\pi}_i) = x_i,$$

whence

$$\begin{aligned} K(x_i) &\leq |\pi_M \pi \hat{\pi}_i| = k_i + |\pi| + c' - 1 \\ &\leq |x_i| + \log \Pr(I) + |\pi| + c'. \end{aligned}$$

□

Corollary 4.5. For every recursive function $t : \mathbf{N} \rightarrow \mathbf{N}$ there exists a constant $c^* \in \mathbf{N}$ such that, for all $y \in \{0, 1\}^*$ and all $\pi \in \text{PROG}^t(y)$,

$$K(\pi) \leq |\pi| + \log \mathbf{m}^t(y) + K(y) + c^*.$$

(Note that $-|\pi| \leq \log \mathbf{m}^t(y) \leq 0$, so the bound becomes tighter as the time-bounded algorithmic probability of y becomes smaller.)

Proof. Let $t : \mathbf{N} \rightarrow \mathbf{N}$ be recursive. Let M be a Turing machine that, with program $\pi \in \{0, 1\}^*$, does the following. First M simulates $U(\pi)$. If this computation does not succeed, then $M(\pi) \uparrow$. Otherwise, if $U(\pi) = y$, then M simulates $U(\pi')$ for $t(|y|)$ steps for every string $\pi' \in \{0, 1\}^{\leq t(|y|)}$, and uses the result of this simulation to output an (encoded) enumeration $[\pi_0, \dots, \pi_{n-1}]$ of the finite instantaneous code $\text{PROG}^t(y)$.

Since U is a universal Turing machine, there is a program prefix $\pi_M \in \{0, 1\}^*$ such that, for all $\pi \in \{0, 1\}^*$, $U(\pi_M\pi) = M(\pi)$. Let

$$c^* = |\pi_M| + c',$$

where c' is the constant given by Lemma 4.4. For $y \in \{0, 1\}^*$, let π_y be a shortest element of $\text{PROG}(y)$. Then, for all y , the string $\pi_M\pi_y$ computes the finite instantaneous code $\text{PROG}^t(y)$. It follows by Lemma 4.4 that, for all $y \in \{0, 1\}^*$ and $\pi \in \text{PROG}^t(y)$,

$$\begin{aligned} K(\pi) &\leq |\pi| + \log \Pr(\text{PROG}^t(y)) + |\pi_M\pi_y| + c' \\ &= |\pi| + \log \mathbf{m}^t(y) + K(y) + c^*. \end{aligned}$$

□

In this paper we are especially interested in the Kolmogorov complexities of initial segments of infinite binary sequences. In this regard, given a function $g : \mathbf{N} \rightarrow [0, \infty)$ and a recursive time bound $t : \mathbf{N} \rightarrow \mathbf{N}$, we define the classes

$$\mathbf{K}_{\text{i.o.}}[< g(n)] = \{x \in \{0, 1\}^\infty \mid K(x[0..n-1]) < g(n) \text{ i.o.}\}$$

and

$$\mathbf{K}_{\text{i.o.}}^t[< g(n)] = \left\{x \in \{0, 1\}^\infty \mid K^t(x[0..n-1]) < g(n) \text{ i.o.}\right\}.$$

Thus we are using $g(n)$ as a “threshold value” for the Kolmogorov complexity of the n -bit prefix of a sequence $x \in \{0, 1\}^\infty$. These classes contain those sequences for which this Kolmogorov complexity is below the threshold value for infinitely many prefixes.

The following theorem, which is used in proving our main result, says that almost every recursive sequence has very high time-bounded Kolmogorov complexity almost everywhere.

Theorem 4.6 (Lutz [37]). For every recursive bound $t : \mathbf{N} \rightarrow \mathbf{N}$ and every real number $0 < \alpha < 1$,

$$\mu(\mathbf{K}_{\text{i.o.}}^t[< \alpha n] \mid \text{REC}) = 0.$$

(In fact, Corollary 4.9 of [37] is stronger than this in several respects.)

We conclude this section with a brief discussion of the algorithmic randomness of infinite binary sequences. Algorithmic randomness was originally defined by Martin-Löf [39], using constructive versions of ideas from measure theory. Subsequently, Levin [26, 27], Schnorr [47], and Chaitin [11] showed that algorithmic randomness could be characterized in terms of self-delimiting Kolmogorov complexity. (Indeed, this was an important motivation for developing the self-delimiting formulation.) For the purposes of the present paper, it is convenient to use this characterization as the definition.

Definition. A sequence $x \in \{0, 1\}^\infty$ is *algorithmically random*, and we write $x \in \text{RAND}$, if there is a constant $k \in \mathbf{N}$ such that $K(x[0..n-1]) > n - k$ a.e. That is,

$$\text{RAND} = \bigcup_{k=0}^{\infty} \mathbf{K}_{\text{i.o.}}[< n - k]^c.$$

The following theorem summarizes some elementary properties of RAND that are used in this paper.

Theorem 4.7 (Martin-Löf [39]). RAND is a Σ_2^0 , measure 1 subset of $\{0, 1\}^\infty$ that is closed under finite variations and does not contain the characteristic sequence of any recursively enumerable set.

5 Strong Computational Depth

In this section, we investigate Bennett's notion of strong computational depth for infinite binary sequences. This notion can be defined in several equivalent ways. We start with the definition most convenient for our purposes. Subsequently, in Theorem 5.4 below, we prove the equivalence of this definition with others that have appeared in the literature.

Definition. For $t, g : \mathbf{N} \rightarrow \mathbf{N}$ and $n \in \mathbf{N}$, we define the sets

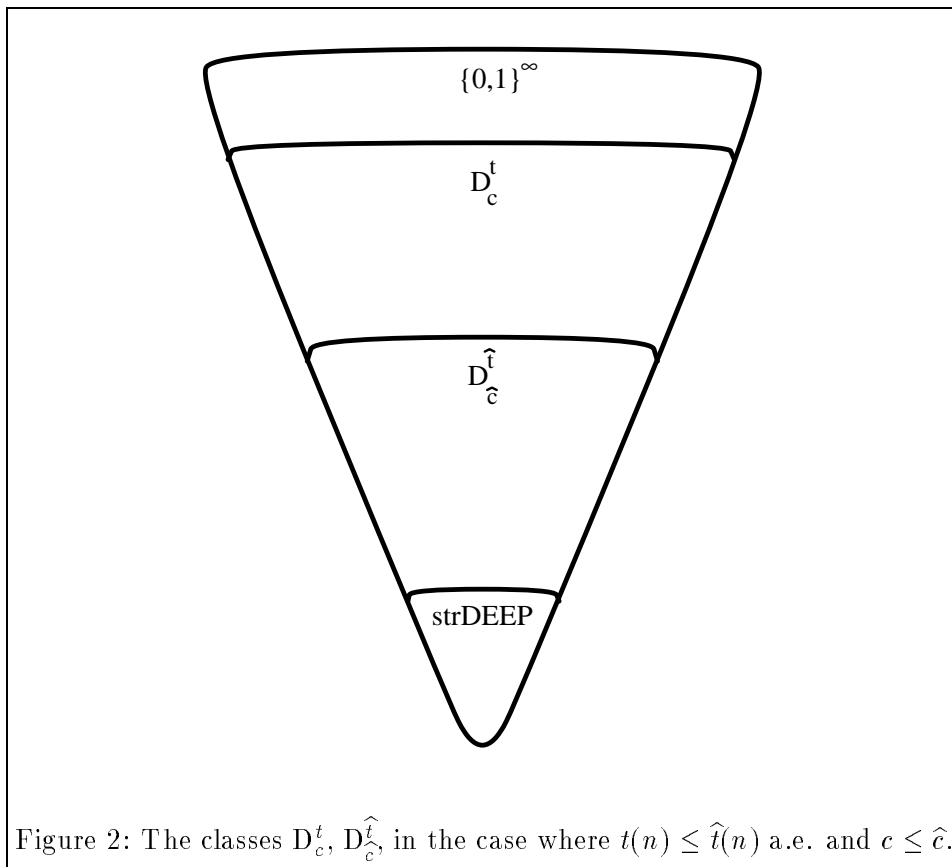
$$D_g^t(n) = \{x \in \{0, 1\}^\infty \mid (\forall \pi \in \text{PROG}^t(x[0..n-1])) K(\pi) \leq |\pi| - g(n)\}$$

and

$$\begin{aligned} D_g^t &= \bigcup_{m=0}^{\infty} \bigcap_{n=m}^{\infty} D_g^t(n) \\ &= \{x \in \{0, 1\}^\infty \mid x \in D_g^t(n) \text{ a.e.}\}. \end{aligned}$$

A sequence $x \in \{0, 1\}^\infty$ is *strongly deep*, and we write $x \in \text{strDEEP}$, if for every recursive time bound $t : \mathbf{N} \rightarrow \mathbf{N}$ and every constant $c \in \mathbf{N}$, it is the case that $x \in D_c^t$.

Intuitively, then, a sequence $x \in \{0, 1\}^\infty$ is in $D_g^t(n)$ if every t -fast program π for $x[0..n-1]$ can be compressed by at least $g(n)$ bits. Note that, if $t(n) \leq \hat{t}(n)$ and $g(n) \leq \hat{g}(n)$, then $D_{\hat{g}}^{\hat{t}}(n) \subseteq D_g^t(n)$. Thus, if $t(n) \leq \hat{t}(n)$ a.e. and $g(n) \leq \hat{g}(n)$ a.e., then $D_{\hat{g}}^{\hat{t}} \subseteq D_g^t$. In particular, if $g(n) = c$ and $\hat{g}(n) = \hat{c}$ are constant, then we have the situation depicted in Figure 2.



We start by examining the relationship between randomness and strong depth. We use the following technical lemma.

Lemma 5.1. If $x \in \text{RAND}$, then there exist a sequence k_0, k_1, \dots of natural

numbers and a sequence π_0, π_1, \dots of programs satisfying the following three conditions for all $i \in \mathbf{N}$.

- (1) For all $n \geq k_i$, $K(x[0..k_i - 1]) - k_i \leq K(x[0..n - 1]) - n$.
- (2) $U(\pi_i) = x[0..k_i - 1]$ and $|\pi_i| = K(x[0..k_i - 1])$.
- (3) $k_{i+1} > k_i + \text{time}_U(\pi_i)$.

Proof. Let $x \in \text{RAND}$. Define $f : \mathbf{N} \rightarrow \mathbf{Z}$ by $f(n) = K(x[0..n - 1]) - n$. For each $i \in \mathbf{N}$, fix the least argument $n_i \geq i$ such that $f(n_i) \leq f(n)$ for all $n \geq i$. (Since $x \in \text{RAND}$, f is bounded below, so n_i exists.) Define the sequences k_0, k_1, \dots and π_0, π_1, \dots recursively as follows. Let $k_0 = n_0$ and let π_0 be a minimal program for $x[0..k_0 - 1]$. Given k_i and π_i , let $k_{i+1} = n_{k_i + \text{time}_U(\pi_i) + 1}$ and let π_{i+1} be a minimal program for $x[0..k_{i+1} - 1]$. It is easily verified that the sequences k_0, k_1, \dots and π_0, π_1, \dots satisfy conditions (1), (2), and (3). \square

Bennett [5] has noted that no algorithmically random sequence is strongly deep. We now prove this fact. Moreover, we show that it holds in a very strong way. Intuitively, we show that every algorithmically random sequence lies “very near the top” of the diagram in Figure 2.

Theorem 5.2 (Bennett [5]). $\text{RAND} \cap \text{strDEEP} = \emptyset$. In fact, there exist a recursive function $t(n) = O(n \log n)$ and a constant $c \in \mathbf{N}$ such that $\text{RAND} \cap D_c^t = \emptyset$.

Proof. Let M be a Turing machine that, with program πy , does the following. The machine M simulates $U(\pi)$, recording $\text{time}_U(\pi)$ while doing so. If the simulated computation succeeds, M then reads and outputs the first $\text{time}_U(\pi)$ bits of y (appended to the string $U(\pi)$ already produced as output) and halts. Note that if $|y| = \text{time}_U(\pi)$, then the computation of $M(\pi y)$ succeeds, with $M(\pi y) = U(\pi)y$. Otherwise, the computation of $M(\pi y)$ is a failure.

On successful computations, the Turing machine M takes $O(|y|)$ steps to produce $U(\pi)y$. Thus there exist a program prefix π_M and a recursive, nondecreasing time bound $t(n) = O(n \log n)$ such that, for all successful computations $U(\pi)$ and all strings y with $|y| = \text{time}_U(\pi)$, the following two conditions hold.

- (i) $U(\pi_M \pi y) = U(\pi)y$.

$$(ii) \text{ time}_U(\pi_M \pi y) \leq t(|y|).$$

Let $c = |\pi_M| + c_0$, where c_0 is the constant from Lemma 4.1. We prove that $\text{RAND} \cap D_c^t = \emptyset$.

Let $x \in \text{RAND}$. Fix sequences k_0, k_1, \dots and π_0, π_1, \dots as in Lemma 5.1. For each $i \in \mathbf{N}$, let $n_i = k_i + \text{time}_U(\pi_i)$. Note that the sequence n_0, n_1, \dots is strictly increasing. We prove that $x \notin D_c^t$ by showing that, for all $i \in \mathbf{N}$, $x \notin D_c^t(n_i)$.

Conditions (i) and (ii) above imply that the following conditions hold for all $i \in \mathbf{N}$.

$$(iii) U(\pi_M \pi_i x[k_i..n_i - 1]) = x[0..n_i - 1].$$

$$(iv) \text{ time}_U(\pi_M \pi_i x[k_i..n_i - 1]) \leq t(n_i - k_i) \leq t(n_i).$$

Then, for all $i \in \mathbf{N}$

$$\pi_M \pi_i x[k_i..n_i - 1] \in \text{PROG}^t(x[0..n_i - 1])$$

and Lemma 5.1 tells us that

$$\begin{aligned} K(x[0..k_i - 1]) &\leq K(x[0..n_i - 1]) - n_i + k_i \\ &= K(x[0..n_i - 1]) - \text{time}_U(\pi_i), \end{aligned}$$

whence

$$\begin{aligned} K(\pi_M \pi_i x[k_i..n_i - 1]) &\geq K(x[0..n_i - 1]) - c_0 \\ &\geq K(x[0..k_i - 1]) + \text{time}_U(\pi_i) - c_0 \\ &= |\pi_i| + n_i - k_i - c_0 \\ &= |\pi_i x[k_i..n_i - 1]| - c_0 \\ &= |\pi_M \pi_i x[k_i..n_i - 1]| - c. \end{aligned}$$

Thus $x \notin D_c^t(n_i)$ for all $i \in \mathbf{N}$, so $x \notin D_c^t$. \square

We next show that strong computational depth can be characterized in several equivalent ways. For this, we need some notation and a lemma. We first recall Bennett's definition of the computational depth of finite strings.

Definition [5]. Let $w \in \{0, 1\}^*$ and $c \in \mathbf{N}$. Then the *computational depth of w at significance level c* is

$$\text{depth}_c(w) = \min\{t \in \mathbf{N} \mid (\exists \pi \in \text{PROG}^t(w)) |\pi| < K(\pi) + c\}.$$

That is, the depth of a finite string at significance level c is the minimum time required to compute w from a program that is not compressible by c or more bits.

Our alternate characterizations of strong depth also use the following classes.

Definition. For $t, g : \mathbf{N} \rightarrow \mathbf{N}$ and $n \in \mathbf{N}$, we define the sets

$$\begin{aligned}\widehat{D}_g^t(n) &= \{x \in \{0, 1\}^\infty \mid K(x[0..n-1]) \leq K^t(x[0..n-1]) - g(n)\}, \\ \widetilde{D}_g^t(n) &= \{x \in \{0, 1\}^\infty \mid \mathbf{m}(x[0..n-1]) \geq 2^{g(n)} \mathbf{m}^t(x[0..n-1])\}, \\ \widehat{D}_g^t &= \bigcup_{m=0}^{\infty} \bigcap_{n=m}^{\infty} \widehat{D}_g^t(n), \\ \widetilde{D}_g^t &= \bigcup_{m=0}^{\infty} \bigcap_{n=m}^{\infty} \widetilde{D}_g^t(n).\end{aligned}$$

The following lemma shows that the classes \widehat{D}_g^t and \widetilde{D}_g^t are, in a quantitative sense, “minor variants” of the classes D_g^t . This result was proven in a slightly different form in [5].

Lemma 5.3 (Bennett [5]). If $t : \mathbf{N} \rightarrow \mathbf{N}$ is recursive, then there exist constants $c_0, c_1, c_2 \in \mathbf{N}$ and a recursive function $t_1 : \mathbf{N} \rightarrow \mathbf{N}$ such that the following six conditions hold for all $g : \mathbf{N} \rightarrow \mathbf{N}$ and all $n \in \mathbf{N}$.

- | | |
|---|---|
| <ol style="list-style-type: none"> 1. $D_{g+c_0}^t(n) \subseteq \widehat{D}_g^t(n)$ 2. $\widehat{D}_{g+c_1}^{t_1}(n) \subseteq \widetilde{D}_g^t(n)$ 3. $\widetilde{D}_{g+c_2}^t(n) \subseteq D_g^t(n)$ | <ol style="list-style-type: none"> 4. $D_{g+c_0}^t \subseteq \widehat{D}_g^t$ 5. $\widehat{D}_{g+c_1}^{t_1} \subseteq \widetilde{D}_g^t$ 6. $\widetilde{D}_{g+c_2}^t \subseteq D_g^t$ |
|---|---|

Proof. It suffices to prove 1, 2, and 3, since 4, 5, and 6 then follow immediately.

1. Let c_0 be as in Lemma 4.1 and assume that $x \in D_{g+c_0}^t(n)$. Let π be a shortest element of $\text{PROG}^t(x[0..n-1])$. Since $x \in D_{g+c_0}^t(n)$, we have $K(\pi) \leq |\pi| - g(n) - c_0$. It follows that

$$\begin{aligned}K(x[0..n-1]) &\leq K(\pi) + c_0 \\ &\leq |\pi| - g(n) \\ &= K^t(x[0..n-1]) - g(n),\end{aligned}$$

whence $x \in \widehat{\mathbf{D}}_g^t(n)$.

2. Choose c_1 and t_1 for t as in Theorem 4.3 and assume that $x \in \widehat{\mathbf{D}}_{g+c_1}^{t_1}(n)$. Then $K(x[0..n-1]) \leq K^{t_1}(x[0..n-1]) - g(n) - c_1$. It follows by Theorems 4.2 and 4.3 that

$$\begin{aligned} \mathbf{m}(x[0..n-1]) &\geq 2^{-K(x[0..n-1])} \\ &\geq 2^{g(n)+c_1-K^{t_1}(x[0..n-1])} \\ &> 2^{g(n)} \mathbf{m}^t(x[0..n-1]), \end{aligned}$$

whence $x \in \widetilde{\mathbf{D}}_g^t(n)$.

3. Let \tilde{c} be as in Theorem 4.2, choose c^* for t as in Corollary 4.5, let $c_2 = \tilde{c} + c^*$, and assume that $x \in \widetilde{\mathbf{D}}_{g+c_2}^t(n)$. Then

$$\begin{aligned} K(x[0..n-1]) &\leq -\log \mathbf{m}(x[0..n-1]) + \tilde{c} \\ &\leq -\log \mathbf{m}^t(x[0..n-1]) - g(n) - c_2 + \tilde{c} \\ &= -\log \mathbf{m}^t(x[0..n-1]) - g(n) - c^*. \end{aligned}$$

Thus, for all $\pi \in \text{PROG}^t(x[0..n-1])$,

$$\begin{aligned} K(\pi) &\leq |\pi| + K(x[0..n-1]) + \log \mathbf{m}^t(x[0..n-1]) + c^* \\ &\leq |\pi| - g(n), \end{aligned}$$

whence $x \in \mathbf{D}_g^t(n)$. □

We now prove the equivalence of several characterizations of strong computational depth.

Theorem 5.4 (Bennett [5]). For $x \in \{0, 1\}^\infty$, the following four conditions are equivalent.

- (1) x is strongly deep.
- (2) For every recursive time bound $t : \mathbf{N} \rightarrow \mathbf{N}$ and every constant $c \in \mathbf{N}$, $\text{depth}_c(x[0..n-1]) > t(n)$ a.e.
- (3) For every recursive time bound $t : \mathbf{N} \rightarrow \mathbf{N}$ and every constant $c \in \mathbf{N}$, $x \in \widehat{\mathbf{D}}_c^t$.
- (4) For every recursive time bound $t : \mathbf{N} \rightarrow \mathbf{N}$ and every constant $c \in \mathbf{N}$, $x \in \widetilde{\mathbf{D}}_c^t$.

Proof. The equivalence of (1) and (2) follows immediately from the definitions. The equivalence of (1), (3), and (4) follows immediately from Lemma 5.3. \square

In [5], Bennett uses condition (2) of Theorem 5.4 above as the definition of strong computational depth. As noted above, this is trivially equivalent to condition (1), i.e., to our definition in terms of the classes D_c^t . Bennett [5] also considers definitions in terms similar to those used in defining the classes \widehat{D}_c^t and \widetilde{D}_c^t and implicitly proves the equivalence of conditions (1), (3), and (4). The discussions of depth by Li and Vitányi in the *Handbook of Theoretical Computer Science* [33] and their recent book [35] essentially use condition (4) as the definition. In any case, a sequence x is strongly deep if, for every recursive t and constant c , almost every prefix $x[0..n-1]$ is “more than t deep at significance level c ,” in the sense that more than $t(n)$ time is required to derive $x[0..n-1]$ from any description whose length is within c bits of the minimum possible length.

We next prove a technical lemma on the quantitative relationship between computational depth and time-bounded Turing reducibility. This can be regarded as a quantitative, infinitary version of Bennett’s deterministic slow-growth law [5]. We need two special notations for this lemma. First, for any function $s : \mathbf{N} \rightarrow \mathbf{N}$, we define the function $s^* : \mathbf{N} \rightarrow \mathbf{N}$ by

$$s^*(n) = 2^{s(\lceil \log n \rceil)+1}.$$

Second, for any unbounded, nondecreasing function $f : \mathbf{N} \rightarrow \mathbf{N}$, we define the special-purpose “inverse” function $f^{-1} : \mathbf{N} \rightarrow \mathbf{N}$ by

$$f^{-1}(n) = \max\{m \mid f(m) < n\}.$$

Also, for this lemma, say that a function $s : \mathbf{N} \rightarrow \mathbf{N}$ is *time-constructible* if there exist a constant $c_s \in \mathbf{N}$ and a Turing machine that, given the standard binary representation w of a natural number n , computes the standard binary representation of $s(n)$ in at most $c_s \cdot s(|w|)$ steps. Using standard techniques [2, 19], it is easy to show that, for every recursive function $r : \mathbf{N} \rightarrow \mathbf{N}$, there is a strictly increasing, time-constructible function $s : \mathbf{N} \rightarrow \mathbf{N}$ such that, for all $n \in \mathbf{N}$, $r(n) \leq s(n)$.

Lemma 5.5. Let $s : \mathbf{N} \rightarrow \mathbf{N}$ be strictly increasing and time-constructible, with the constant $c_s \in \mathbf{N}$ as witness. For each s -time-bounded oracle Turing machine M , there is a constant $c_M \in \mathbf{N}$ with the following property. Given

nondecreasing functions $t, g : \mathbf{N} \rightarrow \mathbf{N}$, define the functions $\tau, \hat{t}, \hat{g} : \mathbf{N} \rightarrow \mathbf{N}$ by

$$\begin{aligned}\tau(n) &= t(s^*(n+1)) + 4s^*(n+1) + 2(n+1)c_s s(l) + 2ns^*(n+1)s(l), \\ \hat{t} &= c_M(1 + \tau(n)\lceil \log \tau(n) \rceil), \\ \hat{g} &= g(s^*(n+1)) + c_M,\end{aligned}$$

where l is the number of bits in the binary representation of n . For all $x, y \in \{0, 1\}^\infty$, if $y \leq_T^{\text{DTIME}(s)}$ x via M and $y \in D_g^{\hat{t}}$, then $x \in D_g^{\hat{t}}$.

Proof. Let s and M be as in the statement of the lemma. Let M' be a Turing machine that, with program $\pi \in \{0, 1\}^*$, operates as in Figure 3. Since U is an efficient universal Turing machine, there exist a program prefix $\pi_{M'} \in \{0, 1\}^*$ and a constant $c_{M'} \in \mathbf{N}$ such that, for all $\pi \in \{0, 1\}^*$,

$$U(\pi_{M'}\pi) = M'(\pi)$$

and

$$\text{time}_U(\pi_{M'}\pi) \leq c_{M'}(1 + \text{time}_{M'}(\pi) \log \text{time}_{M'}(\pi)).$$

Let M'' be a Turing machine that, with program $\pi^* \in \{0, 1\}^*$, simulates $U(\pi^*)$ and outputs π if and only if $U(\pi^*) = \pi_{M'}\pi$. Since U is universal, there is a program prefix $\pi_{M''} \in \{0, 1\}^*$ such that, for all $\pi^* \in \{0, 1\}^*$, $U(\pi_{M''}\pi^*) = M''(\pi^*)$. Let

$$c_M = \max \{c_{M'}, |\pi_{M'}| + |\pi_{M''}|\}.$$

Fix $m_0 \in \mathbf{N}$ such that $(s^*)^{-1}(m) > 0$ for all $m \geq m_0$.

Now define τ, \hat{t} , and \hat{g} as in the statement of the lemma and assume that $x, y \in \{0, 1\}^\infty$ satisfy $y \leq_T^{\text{DTIME}(s)}$ x via M and $y \in D_g^{\hat{t}}$. Fix $n_0 \in \mathbf{N}$ such that $y \in D_g^{\hat{t}}(n)$ for all $n \geq n_0$ and let

$$m_1 = \max \{m_0, s^*(n_0) + 1\}.$$

The following two claims are verified at the end of this proof.

Claim 1. For all $m \geq m_0$ and $\pi \in \{0, 1\}^*$, if $\pi \in \text{PROG}^t(x[0..m-1])$, then $\pi_{M'}\pi \in \text{PROG}^{\hat{t}}(y[0..n-1])$, where $n = (s^*)^{-1}(m)$.

Claim 2. For all $m \geq m_1$ and all $\pi \in \text{PROG}^t(x[0..m-1])$,

$$K(\pi) \leq |\pi| - \hat{g}(n) + c_M,$$

```

begin
   $u := U(\pi)$ ;
   $n := (s^*)^{-1}(|u|)$ ;
  for  $0 \leq i < n$  do
    append the bit  $M^{u0^\infty}(i)$  to the output;
  halt;
end  $M'(\pi)$ .

```

Figure 3: The Turing machine M' used in the proof of Lemma 5.5.

where $n = (s^*)^{-1}(m)$.

To finish proving the lemma, let $m \geq m_1$ and let $\pi \in \text{PROG}^t(x[0..m-1])$. Then, by Claim 2 and the monotonicity of g ,

$$\begin{aligned}
K(\pi) &\leq |\pi| - \widehat{g}((s^*)^{-1}(m)) + c_M \\
&= |\pi| - g(s^*((s^*)^{-1}(m) + 1)) \\
&\leq |\pi| - g(m).
\end{aligned}$$

Thus $x \in D_g^t(m)$. Since this holds for all $m \geq m_1$, it follows that $x \in D_g^t$, affirming the lemma. All that remains, then, is to prove the two claims.

To prove Claim 1, assume that $m \geq m_0$ and $\pi \in \text{PROG}^t(x[0..m-1])$. Let $u = x[0..m-1]$ and $n = (s^*)^{-1}(m)$. Since $m \geq m_0$, we must have $s^*(n) < m$. Since M is s -time-bounded, this implies that $M^{u0^\infty}(i) = M^x(i) = y[i]$ for all $0 \leq i < n$. (All queries in these computations must be made to bits $x[j]$ for $j < |u|$.) Thus

$$U(\pi_{M'}\pi) = M'(\pi) = y[0..n-1].$$

With program π , M' requires at most $t(m)$ steps to compute u , at most $4m$ additional steps to compute $|u|$ in binary, at most $2(n+1)c_s s(l)$ steps to compute n , and at most $2nms(l)$ steps to execute the for-loop. Since $s^*(n+1) \geq m$, and t is nondecreasing, it follows that $\text{time}_{M'}(\pi) \leq \tau(n)$, so

$$\text{time}_U(\pi_{M'}\pi) \leq \widehat{t}(n).$$

Thus $\pi_{M'}\pi \in \text{PROG}^{\widehat{t}}(y[0..n-1])$. This proves Claim 1.

Finally, to prove Claim 2, let $m \geq m_1$, let $\pi \in \text{PROG}^t(x[0..m-1])$, and let $n = (s^*)^{-1}(m)$. Since $m > s^*(n_0)$, it must be the case that $n = (s^*)^{-1}(m) \geq n_0$, whence $y \in D_g^{\hat{t}}(n)$. Since $m \geq m_0$, Claim 1 tells us that $\pi_{M'}\pi \in \text{PROG}^{\hat{t}}(y[0..n-1])$. Since $y \in D_g^{\hat{t}}$, it follows that

$$K(\pi_{M'}\pi) \leq |\pi_{M'}\pi| - \hat{g}(n) = |\pi| - \hat{g}(n) + |\pi_{M'}|.$$

Now let π^* be a shortest element of $\text{PROG}(\pi_{M'}\pi)$. Then $U(\pi^*) = \pi_{M'}\pi$, so

$$U(\pi_{M''}\pi^*) = M''(\pi^*) = \pi,$$

so

$$\begin{aligned} K(\pi) &\leq |\pi_{M''}\pi^*| \\ &= K(\pi_{M'}\pi) + |\pi_{M''}| \\ &\leq |\pi| - \hat{g}(n) + c_M. \end{aligned}$$

This proves Claim 2 and completes the proof of Lemma 5.5 □

Using Lemma 5.5, we prove that a strongly deep sequence cannot be truth-table reducible (equivalently, reducible in recursively bounded time) to a sequence that is not also strongly deep. This implies the fact, noted by Bennett [5], that strong depth is invariant under truth-table equivalence.

Theorem 5.6. Let $x, y \in \{0, 1\}^\infty$. If $y \leq_{\text{tt}} x$ and y is strongly deep, then x is strongly deep.

Proof. Assume the hypothesis. To see that x is strongly deep, fix a recursive function $t : \mathbf{N} \rightarrow \mathbf{N}$ and a constant $c \in \mathbf{N}$. It suffices to prove that $x \in D_c^t$.

Since $y \leq_{\text{tt}} x$, there exist a strictly increasing time-constructible function $s : \mathbf{N} \rightarrow \mathbf{N}$ and an s -time-bounded oracle Turing machine M such that $y \leq_{\text{T}}^{\text{DTIME}(s)} x$ via M . Choose a constant c_M for M as in Lemma 5.5 and define $g : \mathbf{N} \rightarrow \mathbf{N}$ by $g(n) = c$ for all $n \in \mathbf{N}$. Then, in the notation of Lemma 5.5, \hat{t} is recursive and \hat{g} is constant. Since y is strongly deep, it follows that $y \in D_g^{\hat{t}}$. It follows by Lemma 5.5 that $x \in D_c^t$. □

We now note that no recursive sequence is strongly deep.

Corollary 5.7 (Bennett [5]). $\text{REC} \cap \text{strDEEP} = \emptyset$.

Proof. Let $x \in \text{REC}$; it suffices to show that $x \notin \text{strDEEP}$. Fix $z \in \text{RAND}$. Then, trivially, $x \leq_{\text{tt}} z$. By Theorem 5.2, $z \notin \text{strDEEP}$, so by Theorem 5.6, $x \notin \text{strDEEP}$. \square

Up to this point, this section has largely followed the line of Bennett's work. We now build on this work to prove some new results. Our first such result says, roughly, that every recursive sequence is either somewhat deep or somewhat compressible. It is convenient to use the classes $\widehat{\text{D}}_g^t$ for this result.

Theorem 5.8. If $t : \mathbf{N} \rightarrow \mathbf{N}$ is recursive and $0 < \alpha < \beta < 1$, then

$$\text{REC} \subseteq \widehat{\text{D}}_{\alpha n}^t \cup \mathbf{K}_{\text{i.o.}}^t[< \beta n].$$

Proof. Assume the hypothesis and let

$$x \in \text{REC} - \mathbf{K}_{\text{i.o.}}^t[< \beta n].$$

It suffices to prove that $x \in \widehat{\text{D}}_{\alpha n}^t$.

Since $x \notin \mathbf{K}_{\text{i.o.}}^t[< \beta n]$, we have

$$K^t(x[0..n-1]) \geq \beta n \text{ a.e.}$$

Since x is recursive, it follows that there is a constant $c \in \mathbf{N}$ such that, for all sufficiently large n ,

$$\begin{aligned} K(x[0..n-1]) &< 2 \log n + c \\ &< \beta n - \alpha n \\ &\leq K^t(x[0..n-1]) - \alpha n, \end{aligned}$$

whence $x \in \widehat{\text{D}}_{\alpha n}^t$. \square

Corollary 5.9. For every recursive function $t : \mathbf{N} \rightarrow \mathbf{N}$ and every $0 < \gamma < 1$, the set $\text{D}_{\gamma n}^t$ has measure 1 in REC .

Proof. Let $t : \mathbf{N} \rightarrow \mathbf{N}$ be recursive and let $0 < \gamma < \alpha < \beta < 1$. Choose a recursive function $t_1 : \mathbf{N} \rightarrow \mathbf{N}$ and constants $c_1, c_2 \in \mathbf{N}$ for t as in Lemma 5.3, so that

$$\widehat{\text{D}}_{\gamma n + c_2 + c_1}^{t_1}(n) \subseteq \widetilde{\text{D}}_{\gamma n + c_2}^t(n) \subseteq \text{D}_{\gamma n}^t(n)$$

for all $n \in \mathbf{N}$. For all sufficiently large n ,

$$\widehat{\text{D}}_{\alpha n}^{t_1}(n) \subseteq \widehat{\text{D}}_{\gamma n + c_2 + c_1}^{t_1}(n),$$

so it follows that $\widehat{D}_{\alpha n}^{t_1} \subseteq D_{\gamma n}^t$.

By Theorem 4.6, $\mathbf{K}_{\text{i.o.}}^t[< \beta n]$ has measure 0 in REC. By Theorem 5.8, this implies that $\widehat{D}_{\alpha n}^{t_1}$ has measure 1 in REC. Since $\widehat{D}_{\alpha n}^{t_1} \subseteq D_{\gamma n}^t$, it follows that $D_{\gamma n}^t$ has measure 1 in REC. \square

Corollary 5.10. For every recursive function $t : \mathbf{N} \rightarrow \mathbf{N}$ and every constant $c \in \mathbf{N}$, D_c^t has measure 1 in REC.

It is instructive to compare RAND with REC in light of Theorem 5.2, Corollary 5.7, and Corollary 5.10. Neither RAND nor REC contains a strongly deep sequence. However, referring to Figure 2, Corollary 5.10 says that REC “reaches arbitrarily close to” strDEEP, in the sense that each class D_c^t (for t recursive and c constant) contains almost every sequence in REC. In contrast, if t and c are fixed as in Theorem 5.2, then every element of RAND lies above (i.e. outside of) D_c^t in Figure 2. In this sense, intuitively, REC is much deeper than RAND.

We have now developed enough machinery to examine the computational depth of computationally useful sequences. We use the following definition.

Definition. A sequence $x \in \{0, 1\}^\infty$ is *weakly useful* if there is a recursive time bound $s : \mathbf{N} \rightarrow \mathbf{N}$ such that $\text{DTIME}^x(s)$ does not have measure 0 in REC.

That is, x is weakly useful if it can be used to “efficiently” (i.e., in some recursive time s) solve all the problems in a non-negligible subset of REC.

If $x \in \text{REC}$, then for every recursive time bound s , there is a recursive time bound t such that $\text{DTIME}^x(s) \subseteq \text{DTIME}(t)$. Since every such set $\text{DTIME}(t)$ has measure 0 in REC by Theorem 4.6, this shows that no recursive sequence is weakly useful.

The following result, which is the main theorem of this paper, shows that much more is true.

Theorem 5.11. Every weakly useful sequence is strongly deep.

Proof. Let $x \in \{0, 1\}^\infty$ be weakly useful. To see that x is strongly deep, let $t : \mathbf{N} \rightarrow \mathbf{N}$ be a recursive time bound, and let $c \in \mathbf{N}$. It suffices to prove that $x \in D_c^t$.

Since x is weakly useful, there is a recursive time bound $s : \mathbf{N} \rightarrow \mathbf{N}$ such that $\text{DTIME}^x(s)$ does not have measure 0 in REC. Since every recursive

function is bounded above by a strictly increasing, time-constructible function, we can assume without loss of generality that s is strictly increasing and time-constructible.

Let $\tilde{t}(n) = n \cdot (1 + \tau(n) \lceil \log \tau(n) \rceil)$, where τ is defined from t and s as in Lemma 5.5, and let $\gamma = \frac{1}{2}$. Since \tilde{t} is recursive, Corollary 5.9 tells us that $D_{\gamma n}^{\tilde{t}}$ has measure 1 in REC. Since $\text{DTIME}^x(s)$ does not have measure 0 in REC, it follows that $D_{\gamma n}^{\tilde{t}} \cap \text{DTIME}^x(s) \neq \emptyset$. Fix a sequence $y \in D_{\gamma n}^{\tilde{t}} \cap \text{DTIME}^x(s)$. Then there is an s -time-bounded oracle Turing machine M such that $y \leq_T^{\text{DTIME}(s)} x$. Fix a constant c_M for M as in Lemma 5.5. Define $g(n) = c$ for all $n \in \mathbf{N}$ and define the functions τ, \hat{t} , and \hat{g} from t and g as in Lemma 5.5. Since \hat{g} and c_M are constant, we have $\tilde{t}(n) > \hat{t}(n)$ a.e. and $\gamma n > \hat{g}(n)$ a.e., so $y \in D_{\gamma n}^{\tilde{t}} \subseteq D_{\hat{g}}^{\hat{t}}$. It follows by Lemma 5.5 that $x \in D_c^t$. \square

Notation. Let χ_H and χ_K be the characteristic sequences of the halting problem and the diagonal halting problem, respectively. That is, the sequences $\chi_H, \chi_K \in \{0, 1\}^\infty$ are defined by

$$\begin{aligned} \chi_H[\langle i, n \rangle] = 1 &\Leftrightarrow M_i(n) \text{ halts,} \\ \chi_K[n] = 1 &\Leftrightarrow M_n(n) \text{ halts,} \end{aligned}$$

where M_0, M_1, \dots is a standard enumeration of all deterministic Turing machines and $\langle \cdot, \cdot \rangle$ is a standard pairing function, e.g., $\langle i, n \rangle = \binom{i+n+1}{2} + n$.

Corollary 5.12 (Bennett [5]). The sequences χ_H and χ_K are strongly deep.

Proof. It is well-known that H and K are polynomial-time complete for the set of all recursively enumerable subsets of \mathbf{N} , so χ_H and χ_K are weakly useful. Thus χ_H and χ_K are strongly deep by Theorem 5.11. \square

Note that Theorems 5.2 and 5.11 also provide a new proof of the fact, noted in the introduction, that no algorithmically random sequence is weakly useful.

To see that Theorem 5.11 is actually stronger than Corollary 5.12, we use two known facts concerning high Turing degrees. We first review the relevant definitions. (More detailed discussion can be found in a standard recursion theory text, e.g. [50].)

Recall from section 2 that the characteristic sequence of a set $A \subseteq \mathbf{N}$ is the sequence $\chi_A \in \{0, 1\}^\infty$ such that $A = \{n \in \mathbf{N} \mid \chi_A[n] = 1\}$. A sequence

$x \in \{0, 1\}^\infty$ is *recursively enumerable (r.e.)* if $x = \chi_A$ for some r.e. set $A \subseteq \mathbf{N}$. The *diagonal halting problem relative to* a sequence $x \in \{0, 1\}^\infty$ is the set

$$K^x = \{n \in \mathbf{N} \mid M_n^x(n) \text{ halts}\},$$

where M_n is the n^{th} oracle Turing machine in a standard enumeration. The *jump* of a sequence $x \in \{0, 1\}^\infty$ is the sequence

$$\text{jump}(x) = \chi_{K^x}.$$

A sequence $x \in \{0, 1\}^\infty$ is *high* if $x \leq_T \chi_K$ and $\text{jump}(x) \equiv_T \text{jump}(\chi_K)$. A Turing degree is *high* if it contains a high sequence. It is clear that χ_K and its Turing degree are high.

A set $X \subseteq \{0, 1\}^\infty$ is *uniformly recursive in* a sequence $x \in \{0, 1\}^\infty$ if there is a sequence $y \in \{0, 1\}^\infty$ with the following two properties.

- (i) $y \leq_T x$.
- (ii) $X \subseteq \{y_k \mid k \in \mathbf{N}\}$, where each $y_k \in \{0, 1\}^\infty$ is defined by $y_k[n] = y[\langle k, n \rangle]$ for all $n \in \mathbf{N}$. (Here we are using the standard pairing function $\langle k, n \rangle = \binom{k+n+1}{2} + n$.)

We use the following two known facts.

Theorem 5.13 (Sacks [46]). There exist r.e. sequences that are high and not Turing equivalent to χ_K .

Theorem 5.14 (Martin [38]). A sequence $y \in \{0, 1\}^\infty$ satisfies $\text{jump}(\chi_K) \leq_T \text{jump}(y)$ if and only if there exists $x \equiv_T y$ such that REC is uniformly recursive in x .

Corollary 5.15. Every high Turing degree contains a strongly deep sequence.

Proof. The key observation, pointed out to the third author by Stuart Kurtz, is that every high Turing degree contains a weakly useful sequence. To see this, let \mathbf{a} be a high Turing degree. By Theorem 5.14, there is a sequence $x \in \mathbf{a}$ such that REC is uniformly recursive in x . Then there is a sequence $y \leq_T x$ such that $\text{REC} \subseteq \{y_k \mid k \in \mathbf{N}\}$. Define $z \in \{0, 1\}^\infty$ by

$$z[k] = \begin{cases} x[\frac{k}{2}] & \text{if } k \text{ is even} \\ y[\frac{k-1}{2}] & \text{if } k \text{ is odd} \end{cases}$$

Then $z \equiv_T x$, so $z \in \mathbf{a}$. Also, there is a constant $c \in \mathbf{N}$ such that

$$\text{REC} \subseteq \{y_k \mid k \in \mathbf{N}\} \subseteq \text{DTIME}^z(cn^2 + c),$$

so z is weakly useful. This confirms that every high Turing degree contains a weakly useful sequence. By Theorem 5.11, the corollary follows immediately. \square

Taken together, Theorem 5.13 and Corollary 5.15 show that Theorem 5.11 does indeed strengthen Bennett’s result, Corollary 5.12.

We conclude this section by proving that strongly deep sequences are extremely rare, both in the sense of Lebesgue measure and in the sense of Baire category.

Theorem 5.16. The set strDEEP is meager and has measure 0. In fact, if t and c are as in Theorem 5.2, then D_c^t is meager and has measure 0.

Proof. Let t and c be as in Theorem 5.2. Then $\text{RAND} \cap D_c^t = \emptyset$. Since RAND has measure 1, it follows that D_c^t has measure 0.

For each $n \in \mathbf{N}$, the complement of $D_c^t(n)$ can be written as a (finite) union of cylinders \mathbf{C}_w , with each $|w| = n$. (This is because membership or nonmembership of a sequence x in $D_c^t(n)$ depends only upon $x[0..n-1]$.) Thus, for each $n \in \mathbf{N}$, the set $D_c^t(n)$ is closed. It follows that, for each $m \in \mathbf{N}$, the set $\bigcap_{n=m}^{\infty} D_c^t(n)$ is closed, whence the set $D_c^t = \bigcup_{m=0}^{\infty} \bigcap_{n=m}^{\infty} D_c^t(n)$ is Σ_2^0 . By Theorems 4.7 and 5.2, RAND is nonempty, closed under finite variations, and disjoint from D_c^t . It follows by Fact 3.3 that D_c^t is meager. \square

If we combine the proofs of Fact 3.3 and Theorem 5.16 to form a direct proof of Theorem 5.16, then Player II’s strategy in this proof is to play an appropriate number of “random bits” (bits from a sequence $z \in \text{RAND}$) during each turn. Intuitively, it is only the “shallowness” of these random bits that is relevant to the argument. For example, let FIN be the set of all characteristic sequences of *finite* subsets of \mathbf{N} , i.e.,

$$\text{FIN} = \{x \in \{0, 1\}^{\infty} \mid x[n] = 0 \text{ a.e.}\}$$

If t and c are as in Theorem 5.2, then it is not difficult to show that $\text{FIN} \cap D_c^t = \emptyset$. It follows that Player II could use the sequence 0^{∞} in place of z in the above strategy. That is, Player II could win by playing an appropriate number of 0’s, instead of random bits, during each turn.

6 Weak Computational Depth

In Theorem 5.16, we saw that strongly deep sequences are very rare, both in the sense of Lebesgue measure and in the sense of Baire category. In this brief section, we show that the situation is different for weakly deep sequences. We first recall the definition.

Definition (Bennett [5]). A sequence $x \in \{0, 1\}^\infty$ is *weakly deep*, and we write $x \in \text{wkDEEP}$, if there is no sequence $z \in \text{RAND}$ such that $x \leq_{\text{tt}} z$.

We use the notation

$$\text{REC}_{\text{tt}}(\text{RAND}) = \{x \in \{0, 1\}^\infty \mid (\exists z \in \text{RAND}) x \leq_{\text{tt}} z\}.$$

We thus have

$$\text{wkDEEP} = \text{REC}_{\text{tt}}(\text{RAND})^c.$$

Since $\text{REC} \cup \text{RAND} \subseteq \text{REC}_{\text{tt}}(\text{RAND})$, it follows immediately that

$$\text{wkDEEP} \cap \text{REC} = \text{wkDEEP} \cap \text{RAND} = \emptyset,$$

i.e., that no weakly deep sequence can be recursive or algorithmically random.

As the terminology suggests, every strongly deep sequence is weakly deep.

Theorem 6.1 (Bennett [5]). $\text{strDEEP} \subseteq \text{wkDEEP}$.

Proof. Assume that $x \in \text{strDEEP}$ and $x \leq_{\text{tt}} y$. To see that $x \in \text{wkDEEP}$, it suffices to show that $y \notin \text{RAND}$. But this follows immediately from Theorems 5.2 and 5.6. \square

In particular, Theorems 5.11 and 6.1 imply that weakly deep sequences exist. It should be noted that Gács [16] has proven that, for *every* sequence $x \in \{0, 1\}^\infty$, there exists a sequence $z \in \text{RAND}$ such that $x \leq_{\text{T}} z$. Thus \leq_{T} -reducibility cannot be used in place of \leq_{tt} -reducibility in the definition of wkDEEP .

We have already noted that $\text{wkDEEP} \cap \text{RAND} = \emptyset$. Since RAND has Lebesgue measure 1, it follows that wkDEEP , like strDEEP , has Lebesgue measure 0. The situation for Baire category is quite different. While strDEEP is meager by Theorem 5.16, wkDEEP is comeager by the following result.

Theorem 6.2. The set wkDEEP is comeager.

Proof. Each \leq_{tt} -reduction can be interpreted as a continuous function $f : \{0, 1\}^\infty \rightarrow \{0, 1\}^\infty$. (The condition $y = f(x)$ means that $y \leq_{tt} x$ via the \leq_{tt} -reduction f .) If we let \mathcal{F} be the set of all \leq_{tt} -reductions, then \mathcal{F} is countable and

$$\text{REC}_{tt}(\text{RAND}) = \bigcup_{f \in \mathcal{F}} f(\text{RAND}).$$

We noted in section 4 that RAND is Σ_2^0 . It follows by Fact 3.4 that $f(\text{RAND})$ is Σ_2^0 for every $f \in \mathcal{F}$. Since f is countable, this implies that $\text{REC}_{tt}(\text{RAND})$ is Σ_2^0 .

It is clear that $\text{REC}_{tt}(\text{RAND})$ is closed under finite variations. Also, by Corollary 5.12 and Theorem 6.1, $\text{REC}_{tt}(\text{RAND}) \subsetneq \{0, 1\}^\infty$. Thus, by Fact 3.3, $\text{REC}_{tt}(\text{RAND})$ is meager, whence $\text{wkDEEP} = \text{REC}_{tt}(\text{RAND})^c$ is comeager. \square

Bennett [5] noted that there exist sequences that are weakly deep, but not strongly deep. The following corollary shows that such sequences are, in the sense of Baire category, commonplace.

Corollary 6.3. The set wkDEEP – strDEEP is comeager.

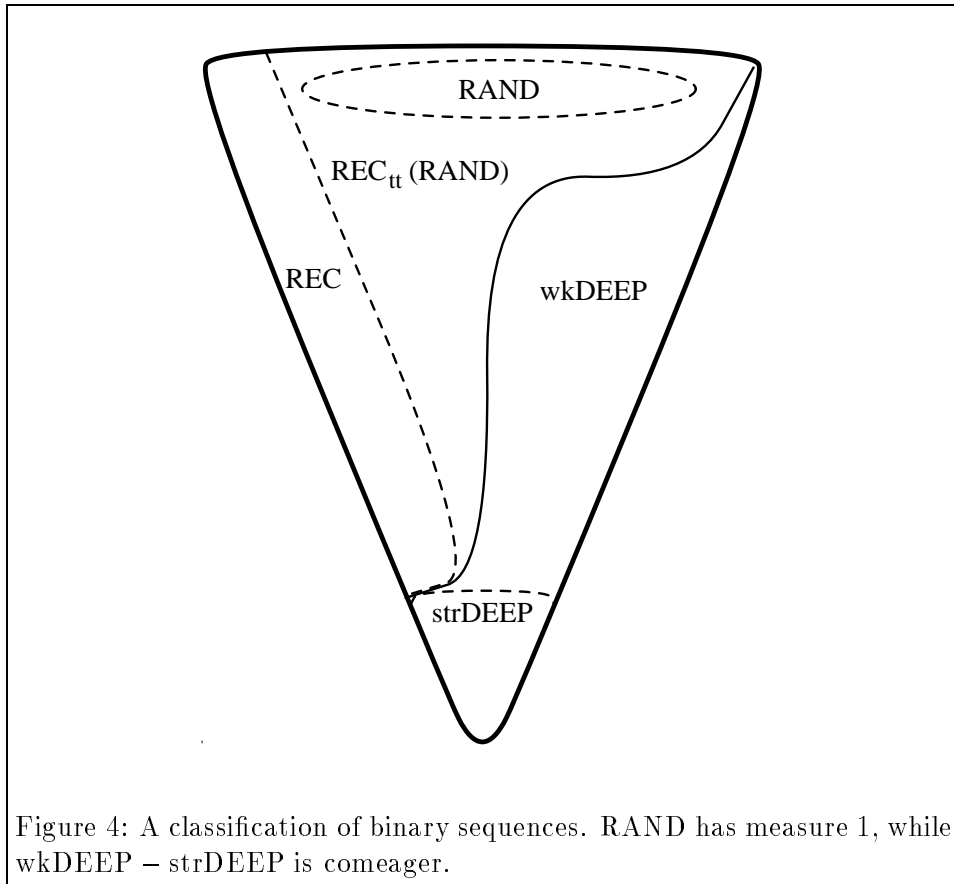
Proof. This follows immediately from Theorems 5.16 and 6.2. \square

Thus, in the sense of Baire category, almost every sequence $x \in \{0, 1\}^\infty$ is weakly deep, but not strongly deep.

Corollary 6.4 (Bennett [5]). $\text{strDEEP} \subsetneq \text{wkDEEP}$.

Proof. This follows immediately from Theorem 6.1 and Corollary 6.2. \square

Figure 4 summarizes the relationships among REC , RAND , wkDEEP , and strDEEP . In the sense of Lebesgue measure, almost every binary sequence is in RAND . On the other hand, in the sense of Baire category, almost every binary sequence is in $\text{wkDEEP} - \text{strDEEP}$.



7 Conclusion

We have shown that every weakly useful sequence is strongly deep. This result generalizes Bennett's observation that χ_K is strongly deep, and gives support to Bennett's thesis that the computational usefulness of χ_K is related to its computational depth. We mention two open questions that are suggested by this result.

Recall that a sequence $x \in \{0, 1\}^\infty$ is weakly useful if there is a recursive time bound $s : \mathbf{N} \rightarrow \mathbf{N}$ such that $\text{DTIME}^x(s)$ does not have measure 0 in REC . Define a sequence $x \in \{0, 1\}^\infty$ to be *strongly useful* if there is a recursive time bound $s : \mathbf{N} \rightarrow \mathbf{N}$ such that $\text{REC} \subseteq \text{DTIME}^x(s)$. Clearly, every strongly useful sequence is weakly useful.

Question 7.1. Do there exist sequences that are weakly useful, but not strongly useful? (We conjecture in the affirmative.)

Our main result implies that every high Turing degree contains a strongly deep sequence. A well-known generalization of high sequences and degrees defines a sequence $x \in \{0, 1\}^\infty$ to be *high_n* ($n \in \mathbf{N}$) if $x \leq_T \chi_K$ and $\text{jump}^{(n)}(x) \equiv_T \text{jump}^{(n)}(\chi_K)$, where $\text{jump}^{(n)}$ is the n -fold iteration of the jump operation. A Turing degree \mathbf{a} is then *high_n* if it contains a high_n sequence. (See [50], for example.) If a sequence or degree is high_n, then it is clearly high_{n+1}. The Turing degree of χ_K is clearly the only high₀ degree. It is also clear that a sequence or degree is high₁ if and only if it is high. Thus, by Corollary 5.15, every high₁ Turing degree contains a strongly deep sequence.

Question 7.2. For $n > 1$, is it necessarily the case that every high_n Turing degree contains a strongly deep sequence?

Answers to Question 7.1 and 7.2 may well improve our understanding of computational depth *vis-à-vis* computational usefulness. More generally, further investigation of Bennett's fundamental notions may yield profound insights into the role of depth in the organization of computational, physical, and biological information.

Acknowledgments

The third author thanks Charles Bennett for several helpful discussions, and Stuart Kurtz for pointing out Theorem 5.14. We also thank Ron Book,

Josep Díaz, and two anonymous referees for suggestions that have improved the exposition of this paper.

References

- [1] L. Adleman. Time, space, and randomness. Technical Report MIT/LCS/79/TM-131, Massachusetts Institute of Technology, Laboratory for Computer Science, March 1979.
- [2] J. L. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I*. Springer-Verlag, 1988.
- [3] Y. M. Barzdin'. Complexity of programs to determine whether natural numbers not greater than n belong to a recursively enumerable set. *Soviet Mathematics Doklady*, 9:1251–1254, 1968.
- [4] C. H. Bennett. Dissipation, information, computational complexity and the definition of organization. In D. Pines, editor, *Emerging Syntheses in Science, Proceedings of the Founding Workshops of the Santa Fe Institute*, pages 297–313, 1985.
- [5] C. H. Bennett. Logical depth and physical complexity. In R. Herken, editor, *The Universal Turing Machine: A Half-Century Survey*, pages 227–257. Oxford University Press, 1988.
- [6] P. Billingsley. *Probability and Measure*, second edition. John Wiley and Sons, 1986.
- [7] R. V. Book. On languages reducible to algorithmically random languages. *SIAM Journal on Computing*, 1993. to appear.
- [8] R. V. Book, J. H. Lutz, and K. W. Wagner. An observation on probability versus randomness with applications to complexity classes. *Mathematical Systems Theory*. to appear.
- [9] G. J. Chaitin. On the length of programs for computing finite binary sequences. *Journal of the Association for Computing Machinery*, 13:547–569, 1966.
- [10] G. J. Chaitin. On the length of programs for computing finite binary sequences: statistical considerations. *Journal of the ACM*, 16:145–159, 1969.

- [11] G. J. Chaitin. A theory of program size formally identical to information theory. *Journal of the Association for Computing Machinery*, 22:329–340, 1975.
- [12] G. J. Chaitin. Incompleteness theorems for random reals. *Advances in Applied Mathematics*, 8:119–146, 1987.
- [13] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. John Wiley & Sons, Inc., 1991.
- [14] R. I. Freidzon. Families of recursive predicates of measure zero. translated in *Journal of Soviet Mathematics*, 6(1976):449–455, 1972.
- [15] P. Gács. On the symmetry of algorithmic information. *Soviet Mathematics Doklady*, 15:1477, 1974.
- [16] P. Gács. Every sequence is reducible to a random one. *Information and Control*, 70:186–192, 1986.
- [17] J. Gill. Computational complexity of probabilistic Turing machines. *SIAM Journal on Computing*, 6:675–695, 1977.
- [18] P. R. Halmos. *Measure Theory*. Springer-Verlag, 1950.
- [19] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.
- [20] J. L. Kelley. *General Topology*. Van Nostrand, 1955.
- [21] A. N. Kolmogorov. Three approaches to the quantitative definition of ‘information’. *Problems of Information Transmission*, 1:1–7, 1965.
- [22] A. N. Kolmogorov. Logical basis for information theory and probability theory. *IEEE Transactions on Information Theory*, IT-14:662–664, 1968.
- [23] A. N. Kolmogorov and V. A. Uspenskii. Algorithms and randomness. translated in *Theory of Probability and its Applications*, 32:389–412, 1987.
- [24] M. Koppel. Complexity, depth, and sophistication. *Complex Systems*, 1:1087–1091, 1987.

- [25] M. Koppel. Structure. In R. Herken, editor, *The Universal Turing Machine: A Half-Century Survey*, pages 435–452. Oxford University Press, 1988.
- [26] L. A. Levin. On the notion of a random sequence. *Soviet Mathematics Doklady*, 14:1413–1416, 1973.
- [27] L. A. Levin. Laws of information conservation (nongrowth) and aspects of the foundation of probability theory. *Problems of Information Transmission*, 10:206–210, 1974.
- [28] L. A. Levin. On the principle of conservation of information in intuitionistic mathematics. *Soviet Mathematics Doklady*, 17:601–605, 1976.
- [29] L. A. Levin. Uniform tests of randomness. *Soviet Mathematics Doklady*, pages 337–340, 1976.
- [30] L. A. Levin. Various measures of complexity for finite objects (axiomatic description). *Soviet Mathematics Doklady*, 17:522–526, 1976.
- [31] L. A. Levin. Randomness conservation inequalities; information and independence in mathematical theories. *Information and Control*, 61:15–37, 1984.
- [32] L. A. Levin and V. V. V’jugin. Invariant properties of informational bulks. *Proceedings of the Sixth Symposium on Mathematical Foundations of Computer Science*, pages 359–364, 1977.
- [33] M. Li and P. M. B. Vitányi. Kolmogorov complexity and its applications. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume A*, pages 187–254. Elsevier, 1990.
- [34] M. Li and P. M. B. Vitányi. Learning simple concepts under simple distributions. *SIAM Journal on Computing*, 20:911–935, 1991.
- [35] M. Li and P. M. B. Vitányi. *An Introduction to Kolmogorov Complexity and its Applications*. Springer, 1993.
- [36] J. H. Lutz. Resource-bounded measure. in preparation.
- [37] J. H. Lutz. Almost everywhere high nonuniform complexity. *Journal of Computer and System Sciences*, 44:220–258, 1992.

- [38] D. A. Martin. Classes of recursively enumerable sets and degrees of unsolvability. *Z. Math. Logik Grundlag. Math.*, 12:295–310, 1966.
- [39] P. Martin-Löf. On the definition of random sequences. *Information and Control*, 9:602–619, 1966.
- [40] P. Martin-Löf. Complexity oscillations in infinite binary sequences. *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete*, 19:225–230, 1971.
- [41] K. Mehlhorn. The “almost all” theory of subrecursive degrees is decidable. In *Proceedings of the Second Colloquium on Automata, Languages, and Programming*, pages 317–325. Springer Lecture Notes in Computer Science, vol. 14, 1974.
- [42] Y. N. Moschovakis. *Descriptive Set Theory*. North-Holland, 1980.
- [43] J. C. Oxtoby. *Measure and Category*. Springer-Verlag, 1980. second edition.
- [44] H. Rogers, Jr. *Theory of Recursive Functions and Effective Computability*. McGraw - Hill, 1967.
- [45] H. L. Royden. *Real Analysis*, third edition. Macmillan Publishing Company, 1988.
- [46] G. E. Sacks. *Degrees of Unsolvability*. Princeton University Press, 1966.
- [47] C. P. Schnorr. Process complexity and effective random tests. *Journal of Computer and System Sciences*, 7:376–388, 1973.
- [48] A. Kh. Shen'. The frequency approach to defining a random sequence. *Semiotika i Informatika*, 19:14–42, 1982. (In Russian.).
- [49] A. Kh. Shen'. On relations between different algorithmic definitions of randomness. *Soviet Mathematics Doklady*, 38:316–319, 1989.
- [50] R. I. Soare. *Recursively Enumerable Sets and Degrees*. Springer-Verlag, 1987.
- [51] R. J. Solomonoff. A formal theory of inductive inference. *Information and Control*, 7:1–22, 224–254, 1964.
- [52] R. M. Solovay, 1975. reported in [12].

- [53] V. V. V'jugin. On Turing invariant sets. *Soviet Mathematics Doklady*, 17:1090–1094, 1976.
- [54] V. V. V'jugin. The algebra of invariant properties of finite sequences. *Problems of Information Transmission*, 18:147–161, 1982.
- [55] A. K. Zvonkin and L. A. Levin. The complexity of finite objects and the development of the concepts of information and randomness by means of the theory of algorithms. *Russian Mathematical Surveys*, 25:83–124, 1970.