KOLMOGOROV COMPLEXITY AND ALGORITHMIC RANDOMNESS

HENRY STEINITZ

Abstract. This paper aims to provide a minimal introduction to algorithmic randomness. In particular, we cover the equivalent 1-randomness and Martin-Löf randomness. After a brief review of relevant concepts in computability, we develop the basic theory of Kolmogorov complexity, including the KC theorem and information content measures. We then build two natural definitions of randomness for infinite strings and prove their equivalence. Finally, we show that almost all reals are 1-random when we identify them with infinite strings.

Contents

1. Introduction

The study of algorithmic randomness is concerned with determining which infinite sequences are random. There are several approaches to solving this problem. We look at two: 1-randomness based on Kolmogorov Complexity and the measure-theoretic Martin-Löf Randomness. Interestingly these (and other) characterizations are equivalent despite significant differences in their perspective. This supports algorithmic randomness as a fundamental topic in computability.

In section 2, we discuss the essential topics in computability necessary for the exposition. First, we formally define strings and reals. We then use an informal definition of Turing machines to introduce computably enumerable languages and partial computable functions. In its final subsection, the prefix-free variants of languages, functions and Turing machines are defined.

In section 3, Kolmogorov Complexity is introduced. Both plain and prefixfree varieties are discussed, along with minimality of prefix-free complexity among information content measures.

Finally, the two definitions of algorithmic randomness are built in section 4. The paper concludes by proving they are equivalent and that almost all reals are algorithmically random. This paper follows the structure laid out in Downey and Hirschfeldt's *Algorithmic Randomness and Complexity* [1]. All of the results and their corresponding proofs are taken from this text. Our exposition is naturally less detailed. Instead, it covers the minimum amount of material required to prove the equivalence of 1- randomness and Martin-Löf randomness.

2. Foundations in Computability

2.1. **Strings and Reals.** Strings are finite sequences of symbols. We will always assume that the symbols come from a finite set called the alphabet.

Definition 2.1. A string s over finite alphabet Σ is a finite sequence $s_1 s_2 \dots s_n$ with each $s_i \in \Sigma$. In this case, we say that s has length n or $|s| = n$. The set of all strings over the alphabet $\{0, 1\}$is denoted $2^{<\omega}$.

For example, english words are strings over the alphabet $\{a, b, ..., z\}$. The most common operation performed on a string is concatenation. If $s = s_1 \dots s_n$ and $t = t_1 \dots t_m$ the concatenation of s with t is $st = s_1 \dots s_n t_1 \dots t_m$. We define s^n to mean s concatenated with itself n times. Lastly, if $k \leq n$, we denote $s_1 \dots s_k$ by $s \restriction k$.

Next, we define an infinite string over the alphabet $\{0, 1\}$.

Definition 2.2. A real S is an infinite sequence $s_1 s_2 \dots$, with each s_i in $\{0, 1\}$. The set of all reals is denoted by 2^ω. Again, $s_1 \dots s_k = s \restriction k$ for all $k \in \omega$.

The word real is used to illustrate the natural correspondence between infinite strings and real numbers. One can think of our infinite strings as the binary expansion of some number. This view is useful because it allows us to assign a measure to a set of infinite strings.

Definition 2.3. Let $\mathscr{S}$ be a set of reals (infinite strings) and let X be the set of all $x \in \mathbb{R}$ such that there exists an $S = s_1 s_2 \dots$ where $.s_1 s_2 \dots$ is a binary representation of x. Then, the measure $\mu(\mathscr{S})$ is defined to be $\mu(X)$.

There is a small subtlety here. First, note that the correspondence from 2^ω to $\mathbb{R}$ is not injective. This is because many rationals have two infinite string representations. However, no irrational number has two representations, and thus the set of real numbers with multiple representations has measure 0.

Lastly, note that it makes sense to define the concatenation st for $s \in 2^{<\omega}$ and $t \in 2^\omega$.

2.2. **Turing Machines.** Turing machines are theoretical computers. They were introduced by Alan Turing in the 1930's to mathematically formalize what it means for a function to by "effectively calculable", or computable by algorithm. They are one of many formalizations that were all shown to be equivalent, but Turing's definition remains the most prominent for its striking clarity.

Definition 2.4. A *Turing machine* is a 7-tuple, $M = (Q, \Gamma, b, \Sigma, \delta, q_0, F)$ where

- Q is a finite set of *states*.
- Γ is a finite set called the *tape alphabet*.
- $b \in \Gamma$ is called the *blank symobl*.
- $\Sigma \subset \Gamma \setminus b$ is called the *input alphabet*.
- $\delta : \Sigma \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is the *transition function*.
- $q_0 \in Q$ is the *initial state*.
- $F \subset Q$ is the set of *halting states*

A Turing machine contains an infinite, one-dimensional tape divided into squares. Each square contains a symbol from the tape alphabet, and at any given time, there are infinitely many blanks and finitely many other symbols. It also contains a finite-state machine that traverses and edits the tape according to the transition function. The tape begins with some string over the input alphabet, surrounded by an infinite array of blank characters on each side. The aforementioned finite-state machine, or tape head, begins pointed to the first character of the input string and begins in state q_0. It then interacts with the tape. For example, if the input string was

100 and $\delta(q_0, 1) = (q_1, 0, R)$, the tape head would take the tape from $\ldots bb\mathbf{1}00bb\ldots$ to $\ldots bb0\mathbf{0}0bb\ldots$, where the bolded character where the tape head is pointed. It would also change its state from q_0 to q_1. Finally, when the tape head enters a halting state, the machine freezes, and the finite sequence of characters from the input alphabet form the output of the machine. This is denoted $M(s)$, where M is the Turing machine and s is the input string. Note that $M(s)$ is not defined for all strings over the input alphabet since a Turing machine doesn?t necessarily enter a halting state. This leads us to the following important definition.

Definition 2.5. Let M be a Turing machine. The domain of M $\mathrm{dom}(M)$ is the set of strings over the input alphabet for which $M(s)$ is defined.

Precisely defining the output of a turing machine isn?t difficult, but it is somewhat tedious. For this reason, we omit it. It is covered in any introduction to computability.

In this paper, we will consider only Turing machines whose input alphabet is $\{0, 1\}$. The following are the essential objects studied in computability.

Definition 2.6. A langauge is a subset of 2^ω. A language $\mathscr{L}$ is computably enumerable if there exists a turing machine M such that $\mathrm{dom}(M) = \mathscr{L}$.

Definition 2.7. A partial function $f : 2^{<\omega} \to 2^{<\omega}$ is *partial computable* if there exists a Turing machine M such that $M(s) = f(s)$ for all $s \in 2^{<\omega}$. We required that $M(s)$ is defined if and only if $f(s)$ is defined.

Turing machines quickly become complex and describing them in detail would obscure central ideas. We will embrace the Church-Turing thesis and casually identify Turing machines with algorithms.

We will soon utilize the equivalence of ordinary Turing machines and multi-tape Turing machines. These are turing machines that have more than one tape, each paired with a unique tape head.

Lastly, we discuss the existence of a universal Turing machine. This is Turing machine that can calculate any partial computable function via the placement of a code before the input.

Proposition 2.8. *There exists a Turing machine U such that. for all partial computable f, there exists an $s_f \in 2^{<\omega}$ with*

$$\mathscr{U}(s_f t) = f(t)$$

for all $t \in 2^{<\omega}$. We require that $f(t)$ is defined exactly when $U(s_f t)$ is defined.

We call any such U a *universal machine*. Note that there are infinitely many universal machines. We will leave this proposition unproved since the construction is technical. Nevertheless, universal machines are very important and will play a critical role in building Kolmogorov complexity.

2.3. **Prefix-free Languages.** It will be useful to have a concise way of referring to the initial segments of strings.

Definition 2.9. A string s is a *prefix* of $u \in 2^{<\omega} \cup 2^\omega$ if there exists a $t \in 2^{<\omega} \cup 2^\omega$ such that $u = st$.

In order to build a suitable definition of string complexity, we need to introduce languages that are *prefix-free*.

Definition 2.10. A language $\mathscr{L}$ is prefix-free if for all $s \in \mathscr{L}$ and $t \in 2^{<\omega}$, $st \notin \mathscr{L}$.

In other words, a language $\mathscr{L}$ is prefix-free if no $s \in L$ is a prefix of some $t \in L$. A simple example of a prefix-free set is the set of all phone numbers. No phone number can be a prefix of a second phone number. Otherwise, the second phone number could never be dialed.

Next, we define prefix-free Turing machines and prefix-free partial computable functions.

Definition 2.11. A Turing machine M is prefix-free if its domain $\text{dom}(M)$ is prefix-free. Similarly, a partial computable function is prefix-free if its domain is prefix free.

Let's momentarily return to multi-tape Turing machines to informally define *self-delimitng* Turing machines. These are turing machines with a designated read-only input tape. At any point in time, the only information on this tape is the input string. The machine is self-delimiting if the input tape?s tape head never exceeds the bounds of the input string. In other words, the machine is self-delimiting if its input tape head is incapable of pointing to a blank. The most important self-delimiting machine is a prefix-free, universal Turing machine.

Proposition 2.12. *There exists a self-delimiting prefix-free Turing machine $\mathscr{U}$ such that, for all prefix-free partial computable f, there exists an $s_f \in 2^{<\omega}$ with*

$$\mathscr{U}(s_f t) = f(t)$$

for all $t \in 2^{<\omega}$. Again, We require that $f(t)$ is defined exactly when $\mathscr{U}(s_f t)$ is defined.

Finally, we introduce a useful notation that shows up often when dealing with prefix-free sets, and use it to define what it means for a set of reals to be computably enumerable.

Definition 2.13. Let $X \subset 2^{<\omega}$. Then

$$[[X]] = \{r \in 2^\omega \mid r \upharpoonright n \in X \text{ for some } n \in \omega\}$$

In English, $[[X]]$ is the set of all reals with a prefix in X.

Definition 2.14. A set of reals $\mathscr{R}$ is computably enumerable if there exists a c.e. language $\mathscr{L}$ such that $\mathscr{R} = [[\mathscr{L}]]$.

3. Kolmogorov Complexity

3.1. **Plain Kolmolgorov Complexity.** In formulating a precise definition of randomness, it is useful to consider how difficult a string is to describe. This notion is captured by Kolmogorov complexity, which measures the length of the smallest computer program whose output is the desired string.

Definition 3.1. Let S be a finite string and $f : 2^{<\omega} \to 2^{<\omega}$ be a partial computable function. Then, the plain Kolmogrov complexity of S with respect to f is

$$C_f(S) = \min_{x \in S}\{|x| \mid f(x) = S\}$$

Of course, we want Kolmogorov complexity to account for *any* description. To accomplish this we specify f in the above definition to be a universal Turing machine.

Definition 3.2. Let S be a finite string and $U : 2^{<\omega} \to 2^{<\omega}$ be a universal Turing machine. Then, the plain Kolmogrov complexity of S is $C(S) = C_U(S)$.

Although the exact Kolmogorov complexity of a string depends on the chosen U, all universal machines produce asymptotically equivalent complexities in the following sense:

Proposition 3.3. *For every partial computable f, there exists a constant c_f such that $C(s) \leq C_f(s) + c_f$ for all strings s.*

Examples 3.4. Let's look at $s_n = (01)^n$. The string s_n is very easy to describe in ordinary language; it is just the string "01" repeated n times. For exactly the same reason, the Kolmogorov complexity of s_n is small relative to $|s_n|$. Consider the computable function $f : 2^{<\omega} \to 2^{<\omega}$ such that $f(n) = (01)^n$. If m_f is the universal code of f, and u is a binary representation of n, then $U(m_f u) = s_n$. It follows that

$$C(s_n) \leq |m_f u| \leq O(log(n)),$$

since $|u| = \lfloor log(n) \rfloor + 1$. In contrast, if we consider the string

$$t = \text{"0101001101100110011100110011000111010110101100010010 1"},$$

we will probably not find a simple description. The simplest universal code of a partial computable function whose output is t probably contains t explicitly, and so $C(t)$ is probably around $|t|$.

The next theorem tells us that seemingly random strings exist, and it is the consequence of an elementary counting argument. These are strings with high complexity relative to their length.

Theorem 3.5. *For every $n \in \omega$, there exists a string $s \in 2^{<\omega}$ such that $|s| = n$ and $C(s) \geq n$.*

Proof. Let $n \in \omega$. Note that there are 2^n strings of length n, but only

$$\sum_{i=0}^{n-1} 2^i = 2^n - 1$$

strings of length less than n. It follows that there exists at least 1 string s of length n with the following property: For all $x \in \omega$ of length less than n, $U(x) \neq s$. It follows that $C(s) \geq n$. □

The next theorem tells us that all long strings have compressible prefixes. It will also demonstrate why it is difficult to define algorithmic randomness on reals using plain complexity.

Theorem 3.6. *Let $k \in \omega$. Then, there exists some $N \in \omega$ such that, if $s \in 2^{<\omega}$ with $|s| \geq N$, there exists a prefix μ of s with $C(\mu) < |\mu| - k$.*

Proof. Consider the computable function f with code s_f that takes v to $v'v$, where v' is $|v|$ in binary. We require that $N = |s_f| + k + 2^{|s_f|+k}$. Now let $s \in 2^{<\omega}$ such that $s \geq n$, let t be a prefix of s with $|t| = s_f + k$, and let n be the number that t represents in binary. Then let u be the next n characters of s after t, and $v = tu$. Now we can compute v from u. Then, $U(s_f u) = tu$ and

$$\begin{aligned} C(v) &\leq |s_f u| \\ &= |s_f| + |u| \\ &= |t| - k + |u| \\ &= |v| - k. \end{aligned}$$

□

3.2. **Prefix-free Kolmogorov Complexity.** In order to avoid the consequences of the previous theorem and other undesirable characteristics of plain complexity, we will replace our normal universal machine U with a self-delimitating prefix-free universal machine $\mathscr{U}$.

Definition 3.7. Fix a self-delimiting prefix-free $\mathscr{U}$. The *prefix-free Kolmogorov complexity*, or *K-complexity*, of a string $s \in 2^{<\omega}$ is

$$K(s) = C_{\mathscr{U}}(s)$$

Let's return to the phone number example. Suppose for a moment that the set of all phone numbers were not prefix-free. Then, for ever number to be dialable, one symbol must designate the end of the number. This would be the analog of a blank space in the standard universal machine. This tells us the length of the string, yet is not factored into the calculation of plain complexity. However, it is accounted for in prefix-free complexity, since $\mathscr{U}$ is self-delimiting.

3.3. **The KC Theorem.** The following theorem is a very useful tool for building prefix-free machines and will be vital for the results that follow. Essentially, it says we can define a prefix-free machine by only specifying a computable sequence of outputs and lengths of their corresponding inputs. However, not every sequence of lengths is acceptable. This is because $\sum_{s \in S} 2^{-|s|} \leq 1$.

Theorem 3.8. *Let $(d_i, s_i)_{i \in \omega}$ be a computable sequence with each $d_i \in \omega, s_i \in 2^{<\omega}$ such that $\sum_{i \in \omega} 2^{-d_i} \leq 1$. Then, there exists a prefix free Turing machine M such that for each $s_i \in 2^{<\omega}$ there exists a t_i such that $M(t_i) = s_i$ and $|t_i| = d_i$.*

Proof. To prove the theorem, we define the computable prefix-free language $\{t_i\}$ such that $|t_i| = d_i$. First, for each $n \in \omega$ we build $x^n = x^n_1 \dots$ so that

$$0.x^n_1 \dots x^n_k = 1 - \sum_{m \leq n} 2^{-d_m}$$

and x^n ends in an infinite string of zeros (we only ever comput the finite, nonzero prefix). We will inductively associate to each x^n_m equal to 1 a u^n_m such that

$$X_n := \{u^n_m\}_{x^n_m = 1} \cup \{t_i\}_{i \leq n}$$

is prefix-free at each $n \in \omega$. The $\{u^n_m\}_{x^n_m = 1}$ acts as a bank from which we extract and modify t_{n+1}. Then, we isolate the $\{t_i\}$ and obtain our desired set.

To begin, let $t_0 = 0^{d_0}$ and $u^0_m = 0^{m-1}1$, for each m such that $x^0_m = 1$. This is precisely the set of m such that $1 \leq m \leq d_0$. Because $x^0_m = 0$ for all $m \geq d_0$, $X_0 = \{t_0\} \cup \{u^0_m\}_{x^0_m = 1}$ is prefix-free.

We now proceed to the inductive step. Suppose X_n is prefix-free. If $x^n_{d^{n+1}} = 1$, then $x^{n+1} = x^n - 2^{-d_{n+1}}$ is identical to x^n, except at the d_{n+1} position. So we

define $t_{n+1} = u^n_{d_{n+1}}$ and $u^{n+1}_m = u^n_m$ for all m such that $x^{n+1}_m = 1$. We are left with exactly the same set, so $X_{n+1} = X_n$ is prefix-free.

If $x^n_{d_{n+1}} = 0$, then there exists some $j < d_{n+1}$ such that $x^n_j = 1$. Otherwise,

$$1 - \sum_{m \leq n+1} 2^{-d_m} < 2^{-(d_{n+1})}$$

which implies

$$\sum_{m \in \omega} 2^{-d_m} > 1.$$

We also have that $x^{n+1}_j = 0$ and $x^{n+1}_m = 1$ when $j < m \leq d_{n+1}$. So define $t_{n+1} = t_n 0^{d_{n+1}-j}$ and $u^{n+1}_m = u^n_m$ for all m such that $x^{n+1}_m = 1$. If $j < m < d_{n+1}$, let $u^{n+1}_m = u^n_j 0^{m-j-1}1$. It follows X_{n+1} is again prefix-free.

So we have algorithmically defined the prefix-free $\{t_i\}_{i \in \omega}$ with $|t_i| = d_i$.

□

3.4. **Information Content Measures.** We now develop a generalization that will be useful in our discussion of algorithmically random reals.

Theorem 3.9. $\sum_{s \in 2^{<\omega}} 2^{-K(s)} \leq 1$

Proof. First note that

$$\sum_{s \in 2^{<\omega}} 2^{-K(s)} \leq \sum_{s \in \mathrm{dom}(\mathscr{U})} 2^{-|s|}$$

since each description only computes a single string. Because $\mathscr{U}$ is a prefix-free machine, it remains to show that $\sum_{s \in \mathscr{L}} 2^{-|s|} \leq 1$ for any prefix-free $\mathscr{L}$. This follows immediately from the following equality:

$$\sum_{s \in \mathscr{L}} 2^{-|s|} = \sum_{s \in \mathscr{L}} \mu([[s]]) \leq 1.$$

The first equality holds because the $[[s]]$ are disjoint when $\mathscr{L}$ is prefix-free, so we are done. □

Definition 3.10. An information content measure is a function $F : A \to \omega$, where $A \subset 2^{<\omega}$ such that

- $\sum_{s \in A} 2^{-F(s)} \leq 1$
- $\{(s, n) \mid F(s) \leq n\}$ is c.e.

Theorem 3.11. *For every information content measure F and string $s \in 2^{<\omega}$,*

$$K(s) \leq F(s) + O(1).$$

Proof. Let $\{F_k\}_{k \in \omega}$ be an enumeration of all information content measures such that

$$\{\{(s, n) \mid F_k(s) \leq n\}_{n \in \omega}\}_{k \in \omega}$$

is uniformly c.e.. Then, define

$$\hat{K}(s) := \min_{k \in \omega}\{F_k(s) + k + 1\}.$$

It follows that $\hat{K}$ is an information content measure, and that $\hat{K}(s) \leq F(s) + O(1)$ for all information content measures F and $s \in 2^{<\omega}$. Furthermore, if F is an information content measure, the set

$$\{(K + 1, s) \mid F(s) \leq k\}_{s \in \mathrm{dom}(F), k \in \omega}$$

satisfies the conditions of the KC theorem. Taking $T = \hat{K}$, there is a prefix-free machine M such that for all $s \in 2^{<\omega}$, there exists $t \in 2^{<\omega}$ with $M(t) = s$ and $|t| \leq \hat{K} + 1$. Thus,

$$K(s) \leq \hat{K}(s) + O(1) \leq F(s) + O(1)$$

for all F. □

4. Algorithmic Randomness

4.1. **1-Randomness.** Finally, we are ready to define what it means for a real to be random. Note that the most natural formulation of randomness using plain complexity results in a vacuous definition. More sophistication is needed to define randomness using plain complexity. This is why we have developed prefix-free complexity.

Definition 4.1. A real S is 1-random if there exists a $k \in \omega$ such that

$$K(S \upharpoonright n) > n - k$$

for all $n \in \omega$.

As the name suggests it is possible to easily generalize 1-randomness to nrandomness. It is also possible to naturally extend the measure-theoretic definition which will be discussed in the next section, and the generalizations turn out to be equivalent. However, this is beyond our scope.

4.2. **Martin-Löf Tests.** The next approach to defining randomness abandons Kolmolgorov complexity altogether. It instead uses special c.e. sets of real numbers and there measure. For a real to be random, it must satisfy a particular property of random things. For example, the number of 1?s should approach the number 0?s as larger and larger prefixes of the real are taken. But many infinite strings satisfy this condition that are not random, such as "101010…". Abstracting away from this, we notice that if, for some real r, $r \upharpoonright n$ contains many or more 1's as 0's for all $n \in \omega$, r cannot be random. Note that we can construct a Turing machine to test this up to a given n. But this is just one of many testable conditions that would, intuitively, render a string not random. Abstracting away from any specific example gives us *Martin-Löf tests*

Definition 4.2. A Martin-Löf test is a sequence $\{U_k\}_{k\in\omega}$ of uniformly Σ^0_1 classes of reals such that $\mu(U_k) \leq 2^{-k}$ for all k. A cass of reals, X is called Martin-Löf null if $X \subset \cap_{n\in\omega} U_k$, for some Martin-Löf test $\{U_k\}_{k\in\omega}$.

Lemma 4.3. *Let M be a prefix-free machine, $k \in \omega$, and*

$$X = \{s \in 2^{<\omega} \mid C_M(s) \leq |s| - k\}.$$

Then, $\mu([[X]]) \leq 2^{-k}\mu([[dom(M)]])$

Proof. First we find the appropriate descriptions descriptions for each $s \in X$. For each $s \in X$, there exists $t_s \in \mathrm{dom}(M)$ such that $M(t_s) = s$ and $|t_s| \leq |s| - k$. Then,

$$\begin{aligned}\mu([[X]]) \leq \sum_{s \in X} 2^{-|s|} &\leq \sum_{s \in X} 2^{-(|t_s|+k)} \\ &= 2^{-k} \sum_{s \in X} 2^{-|t_s|} \\ &\leq 2^{-k} \sum_{t \in \mathrm{dom}(M)} 2^{-|t|} \\ &= 2^{-k} \mu([[\mathrm{dom}(M)]])\end{aligned}$$

□

We are now ready to prove the main equivalence.

Theorem 4.4. *Let $X \in 2^\omega$. Then X is 1-random if and only if $\{X\}$ is not Martin-Löf null.*

Proof. Suppose $\{X\}$ is not Martin-Löf null. Then consider the sets

$$\begin{aligned}U_k :&= \bigcup_{n \in \omega} \{r \in 2^\omega \mid K(r \upharpoonright n) \leq n - k\} \\ &= [[\{s \in 2^{<\omega} \mid K(s) \leq |s| - k\}]].\end{aligned}$$

Because we can enumerate the set of prefix-free machines and simulate them on $\mathscr{U}$, the collection $\{U_k\}$ is uniformly c.e.. Furthermore, by the previous lemma, $\mu(U_k) \leq 2^{-k}$. Thus $\{u_k\}$ is a valid Martin-Löf test. Then $X \notin \cap U_k$ and X is 1-random.

Next suppose $\{S\}$ is Martin-Löf null. By definition, there exists a uniformly c.e. class $\{U_k\}$ such that $X \in \cap_{k \in \omega} U_k$. We can then build uniformly c.e. prefix-free sets $\{R_k\}_{k \in \omega}$ such that $U_k = [[R_k]]$. Next define the information content measures $F_{k^2} : r_{k^2} \to \omega$ with $F_{k^2}(s) = |s| - k$ for all $k \in \omega, k \geq 2$. These functions are really information content measures, since we can effectively list all strings whose size is less than $n + k$ for all $n, k \in \omega$ and

$$\begin{aligned}\sum_{k \geq 2} \sum_{s \in U_{k^2}} 2^{-F_{k^2}(s)} &= \sum_{k \geq 2} \sum_{s \in U_{k^2}} 2^{-|s|+k} \\ &= \sum_{k \geq 2} 2^k \mu(U_{k^2})) \\ &\leq \sum_{k \geq 2} 2^{k-k^2} \\ &< 1.\end{aligned}$$

Now let $k \geq 2$. By invoking the minimality of K, we have that for all $s \in R_{k^2}$ there exists some $c \in \omega$ such that $K(s) \leq |s| - k + c$. But that means that for all $k \in \omega$ there exists an $n \in \omega$ such that $K(S \upharpoonright n) \leq n - k + c$. Thus S is not 1random, and we are done. □

It is worth noting that the Martin-Löf test used in the first part of the above proof filters only the random reals. We call such a test *universal*.

Definition 4.5. A Martin-Löf test $\{U_n\}_{n\in\omega}$ is universal if $\cap_{n\in\omega}U_n$ contains every Martin-Löf null class.

Corollary 4.6. *There exists a universal Martin-Löf test.*

Proof. Let $\{U_k\}$ be as described in the previous proof. Suppose for contradiction that there exists a Martin-Löf null class of reals X such that $X \not\subset \cap U_k$. Next pick some $r \in X$ such that $r \notin \cap U_k$. Because any subset of X must also be Martin-Löf null, the singleton $\{r\}$ is Martin-Löf null. By 4.4, r is not 1-random. But then there exists no $k \in \omega$ such that $K(S \upharpoonright n) \geq n - k$ for all $n \in \omega$. Thus r must be in $\cap U_k$, which directly contradicts how it was chosen. □

We conclude with an interesting consequence of the measure-theoretic paradigm: Almost all reals are 1-random.

Corollary 4.7. *The set of reals that are not 1-random has measure 0.*

Proof. By definition, all intersections of Martin-Löf tests have measure 0. So we let $\{U_k\}_{k\in\omega}$ be a universal test and notice that a real r is not 1-random if and only if $r \in \cap U_k$. Therefore

$$\mu(\{r \mid r \text{ is not 1-random}\}) = \mu(\cap U_k) = 0.$$

□

Acknowledgments. It is a pleasure to thank my mentor, Jonathan Stephenson, for suggesting Algorithmic Randomness as a topic and guiding my research.

References

[1] Rodney G. Downey, Denis R. Hirschfeldt. Algorithmic Randomness and Complexity. Springer New York. 2010.