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 more general notion of information content measures. We then build two natural definitions of randomness for infinite strings (or reals) and, using the preceding results, prove their equivalence. Finally, we cover an interesting consequence of this equivalence, namely, that almost all reals are 1-random.

Contents

1. Introduction

The study of algorithmic randomness is concerned with classifying which infinite sequences of characters from a finite set look random. As one might expect, 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 striking differences in their development. This supports algorithmic randomness as a fundamental topic in computability, rather than an isolated definition.

Date: DEADLINES: Draft AUGUST 19 and Final version AUGUST 30, 2013.

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 c.e. 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 prefix-free varieties are discussed, along with minimality of prefix-free complexity among information content measures.

Finally, the two equivalent definitions of algorithmic randomness are built in section 4. The paper concludes by proving their equivalence, and that, according to these definitions, 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. Of course, our exposition is less in-depth. It, instead, 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.** In general, a string is any finite sequence of characters from a finite set, known as the alphabet. However, for the purposes of this exposition, we're really only interested in strings consisting of the characters 0 and 1. Thus, we introduce the following definition and set. Again s

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

So strings, as we've defined, are just finite lists consisting 0s and 1s. Of course, strings can be formed over other alphabets. For example, english words are strings over the alphabet $\{a, b, \dots, z\}$. Adding punctuation to that set gives you the ability to express English sentences. 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 use the notation s^n to mean s concatenated with itself n times. Lastly, if $k \leq n$, we denote $s_1 \dots s_k$ by $s \upharpoonright k$.

Next, we define an infinite string, known as a real.

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, $s \upharpoonright k = s_1 \dots s_k$ 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 merely the binary expansion of some number. When we discuss algorithmic randomness of the reals, this paradigm becomes incredibly useful. In particular, it allows us to assign a measure to a set of these infinite strings.

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

$$\mu(\mathscr{S}) = \mu(X).$$

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.

2.2. **Turing Machines.**

Turing machines are theoretical computers. They were introduced by Alan Turing in the 1930's to provide a mathematical formalization of what it means for a function to by "effectively calculable", or computable by algorithm. In fact, they're one of many formalizations that were all shown to be equivalent and was by no means the first. Turing's definition remains the most prominent for its striking clarity. The well-known Church-Turing thesis conjectures that the set of effectively calculable functions and those computed by Turing Machines are precisely the same. Such a result cannot be proved, as it relies on one's intuitive notion of an algorithm. Yet, many would argue that the Church-Turing thesis is the obvious consequence of the Turing's definition, while other definitions leave them unconvinced. In the same vein as this historical unfolding of events, we construct the following lengthy, but clear, definition.

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

Intuitively, one pictures a Turing machine as an infinite, one-dimensional tape with infinitely many blank symbols and finitely many other tape symbols, along with 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 infinite arrays of blank characters on both sides. 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 the 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 gives the following important definition.

Definition 2.5. Let M be a Turing machine. Then, the domain of M, 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. Luckily, it is covered in any introduction to computability.

In this paper, we will consider only Turing machines whose input alphabet is $\{0,1\}$. Thus, we will now assume that, for any Turing machine, $\Sigma = \{0,1\}$. The following are computable versions of basic mathematical objects .

Definition 2.6. A *language* is a subset of $2^{<\omega}$. 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, for all $s \in 2^{<\omega}$, $M(s)$ is defined if and only if $f(s)$ is defined and, when both $M(s)$ and $f(s)$ are defined, $M(s) = f(s)$.

Because of the Church-Turing thesis, the existence of a clear algorithm implies the existence of a Turing machine performing the same task. Since Turing machines easily become incredibly complex, we will use the Church-Turing thesis to assert a language is Note that a computable language is different from a c.e. language, but that distinction is not used in the following results. It's also interesting to note that there is also a natural correspondence between languages and reals. However, this too is beyond the scope of this paper.

Later on, we will 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*

$$U(s_f t) = f(t)$$

for all $t \in 2^{<\omega}$ (If $f(t)$ is undefined, $U(s_f t)$ is undefined).

We will leave this proportion unproved since the construction of a specific universal Turing machine is extraordinarily tedious. Nevertheless, universal machines are very important and will play a critical role in building Kolmogorov complexity. Also note that we will sometimes refer to s_f as the *universal code* of f.

2.3. **Prefix-free Languages.**

As we develop complex arguments about the structures we have defined, It will be very useful to build a concise way of referring to the initial segments of strings. The following is that very convenient definition.

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

In building a suitable definition of string complexity, we will need to introduce computational objects that are prefix-free. These objects are exactly what one would expect.

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 $s \in \mathscr{L}$ is a prefix of some $t \in \mathscr{L}$. 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 $\mathrm{dom}(M)$ is prefix-free. Similarly, a partial function f is prefix-free if $\mathrm{dom}(f)$ 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 $t \in 2^{<\omega}$ (If $f(t)$ is undefined, $\mathscr{U}(s_f t)$ is undefined).

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. Kolmolgorov 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 Kolmogorov complexity of S with respect to f is

$$C_f(S) = \min{}_{x \in 2^{<\omega}}\{|x| \mid f(x) = S\}$$

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

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

$$C(S) = C_U(S).$$

The easiest way to digest this definition is to consider a few simple examples.

Examples 3.3. Let's look at $S_n = (01)^n$. The string s_1 is very easy to describe verbally — it's just the string 01 repeated n times. For exactly the same reason, the Kolmogorov complexity of s_n is very 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

$$t = 01010011011001100111001100110001110101101011000100101,$$

we'll probably not find a simple description. The simplest universal code of a partial computable function whose output is t probably contains t explicitly within itself. So, $C(t)$ is probably around $|t|$.

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

Theorem 3.4. *For every $n \in \omega$, there exists a string $s \in 2^{<\omega}$ such that $|s| = n$ and $C(s) \geq n$.*

Proof. Let $n \in \omega$. First, note that there exists 2^n strings of length n, but only

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

strings with length less than n. Then, there is at least 1 string of length n— call it s— with the following property: For all $x \in \omega$ with 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's difficult to define algorithmic randomness on reals using plain complexity, but this will be discussed in the next section.

Theorem 3.5. *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 of s, μ, with $C(\mu) < |\mu| - k$.*

Proof. Consider the computable function f that takes v to $v'v$, where v' is $|v|$ in binary, and its universal Turing machine code s_f. We require that $N = |s_f| + k + 2^{|s_f|+k}$. Now, let $s \in 2^{<\omega}$ such that $|s| \geq n$, t be a prefix of s with $|t| = s_f + k$, and n be the number of 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

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

and the theorem is proved. □

3.2. **Prefix-free Kolmogorov Complexity.** Next, we utilize the notion of prefix-free machines to construct a definition of complexity more useful when dealing with infinite strings. And, in some sense, the following notion of complexity is actually more correct, albeit less intuitive.

Definition 3.6. 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 $\sum_{s\in S} 2^{-|s|} \leq 1$ for any prefix-free S.

Theorem 3.7. *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 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 $x^n = x_1^n \dots$ so that $0.x_1^n \dots x_k^n = 1 - \sum_{m\leq n} 2^{-d_m}$ and x^n ends in an infinite string of zeros (we only ever compute the finite, nonzero prefix). We will inductively associate with each x_m^n equal to 1 a u_m^n such that $\{u_m^n\}_{x_m^n=1}$ along with $\{t_i\}_{i\leq n}$ is prefix free at each $n \in \omega$. The $\{u_m^n\}_{x_m^n=1}$ acts as a bank from which 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 $u_m^0 = 0^{m-1}1$, for each m such that $x_m^0 = 1$. This is precisely the set of m such that $1 \leq m \leq d_0$. So because $x_m^0 = 0$ for all $m \geq d_0$, $X_0 = \{t_0\} \cup \{u_m^0\}_{x_m^0=1}$ is prefix free. We now proceed to our inductive step.

Suppose $X_n = \{u_m^n\}_{x_m^n=1} \cup \{t_i\}_{i\leq n}$ is prefix-free. If $x_{d_{n+1}}^n = 1$, 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 $t_{n+1} = u_{d_{n+1}}^n$ and $u_m^{n+1} = u_m^n$ for all m such that $x_m^{n+1} = 1$. We are left with exactly the same set, so $X_{n+1} = \{u_m^{n+1}\}_{x_m^{n+1}=1} \cup \{t_i\}_{i\leq n+1} = X_n$ is prefix-free.

If $x_{d_{n+1}}^n = 0$, then there exists some $j < d_{n+1}$ such that $x_j^n = 1$. 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 $x_j^{n+1} = 0$ and $x_m^{n+1} = 1$ when $j < m \leq d_{n+1}$. So, define $t_{n+1} = t_n 0^{d_{n+1}-j}$ and $u_m^{n+1} = u_m^n$ for $m < j$ or $m > d_{n+1}$ if $x_m^{n+1} = 1$. If $j < m < d_{n+1}$, let $u_m^{n+1} = u_j^n 0^{m-j-1}1$. It follows that $X_{n+1} = \{u_m^{n+1}\}_{x_m^{n+1}=1} \cup \{t_i\}_{i\leq n+1}$ is again prefix free.

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

3.4. **Information Content Measures.**

We now develop a bit of a generalization that will be useful in our discussion of algorithmically random reals.

Theorem 3.8. $\sum_{s\in 2^{<\omega}} 2^{-K(s)} \leq 1$

Proof. First, note that

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

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 following the equality:

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

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

Definition 3.9. An information content measure is a function $F : A \to \omega$, where $A \subset 2^{<\omega}$ such that

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

Theorem 3.10. *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 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 $F = \hat{K}$, 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 $|t| \leq \hat{K}(s) + 1$. Thus, $K(s) \leq \hat{K}(s) + O(1) \leq F(s) + O(1)$ for all F. □

4. Definitions of Algorithmic Randomness

4.1. 1-Randomness.

Finally, we are ready do 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've 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 n-randomness. 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-lof Tests.

The next approach to defining randomness abandons Kolmogorov complexity altogether. Instead, it very cleverly uses c.e sets of real numbers and there measure. One may believe that, 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 strings satisfy this condition that aren't random, such as $101010\ldots$. Abstracting away from this, we notice that if, for some real r, $r \upharpoonright n$ contains as many or more 1's as 0's for all $n \in \omega$, r cannot be random. Note that we can find construct a Turing machine to test this up to a given n. But of course, this is just one of many testable conditions that would, intuitively, render a string not random. Abstracting away from any specific example births 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 class of reals, X, is called Martin-Löf null if $X \subset \bigcap_{n\in\omega} U_k$, for some Martin-Lof 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 for each $s \in X$. Precisely, for each $s \in X$, there exists a $t_s \in dom(M)$ 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 in a position to state and prove the first equivalence.

Theorem 4.4. *Let $X \in 2^\omega$. 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

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

Because we can enumerate the set of prefix-free machines and simulate them on $\mathscr{U}$, $\{U_k\}$ are uniformly c.e. Furthermore, by the previous lemma, $\mu(U_k) \leq 2^{-k}$. Thus, $\{U_k\}$ is a valid Martin-Lof 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 $X \in \bigcap_{k\in\omega} U_k$. We can then build uniformly c.e. prefix-free sets $\{R_k\}_{k\in\omega}$ such that $U_k = [[R_k]]$ (cite something from computability chapter, and figure out how to make those brackets). Next, define the information content measures $F_{k^2} : R_{k^2} \to \omega$ with $F_{k^2}(s) = |s| - k$, for all $k \in \omega, k \geq 2$. Such functions are indeed information content measures, since we can effectively list all strings whose size is less than $n + k$ for all $n, k \in \omega$ and

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

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

It's worth noting that the Martin-Lof 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-Lof test, $\{U_n\}_{n\in\omega}$, is universal if $\bigcap_{n\in\omega} U_n$ contains every Martin-Lof null class.

Corollary 4.6. *There exists a universal Martin-Lóf 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 $X \not\subset \bigcap U_k$. Next, pick some $r \in X$ such that $r \notin \bigcap U_k$. Because any subset of X must also be Martin-Löf null, $\{r\}$ is Martin-Löf null. By 4.4, r is not 1-random. But then exists no $k \in \omega$ such that $K(S \upharpoonright n) \geq n - k$ for all $n \in \omega$. Thus, r must be in $\bigcap U_k$, which directly contradicts how it was chosen. □

This leads to another interesting consequence of the measure-theoretic paradigm: Almost all reals are 1-random.

Corollary 4.7. *The set of all 1-random reals has measure 1.*

Proof. By definition, all intersections of Martin-Löf tests have measure 0. We can then let $\{U_k\}_{k\in\omega}$ be a universal test and notice that a real, r, is 1-random if and only if $r \notin \bigcap U_k$. Therefore, $\mu(\{r \mid r \text{ is 1-random}\}) = 1 - \mu(\bigcap U_k) = 1$. □

Acknowledgments. It is a pleasure to thank my mentor, Jonathan Stephenson, for suggesting Algorithmic Randomness as a topic, guiding my research, and correcting my mistakes. He's just a really cool dude.

References

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