

The Cornerstones of Algorithmic Randomness

Author: Caleb Gill
Instructor: Dr. Rachel Epstein

May 11, 2021

Abstract

Algorithmic Randomness is a research area of mathematics dedicated to determining if something is random or not. There are several definitions and interpretations of randomness, but for the purposes of this paper we will be focusing on the most widely accepted definitions and intuitions.

1 Introduction

What does it mean for a mathematical object to be considered random? Let's look at some examples.

Example 1: $S_1 = 1010101010 \dots$

Example 2: $S_2 = 1001011001 \dots$

If we look at S_1 , we can see that it is clearly not random because it is just 10 repeated indefinitely. But what about S_2 ? It definitely seems like it would be random but in actuality it isn't! We will explore why later.

Before we dive into this topic, we will have to go over some preliminary knowledge. Note that the sections preceding Section 6 are meant to be used as a reference as needed.

2 Notation

Here we will explain some of the common notations that will be used throughout this paper. Anything else will be explained when it is brought up.

In this paper we will be primarily looking at what are called sequences and strings. *Strings* are finite sequences of 0s and 1s while *sequences* are infinite. The sets of strings and sequences are denoted by $2^{<\omega}$ and 2^ω respectively.

Often we will wish to look at the length of a string, σ , so we denote it by $|\sigma|$. As for sequences we will also wish to look at its initial segments. So if $A \in 2^\omega$, then $A \upharpoonright n$ represents A restricted to its first n -bits. One important string that we will occasionally bring up is λ which is the empty string or the string of length 0. Lastly we will often talk about extending strings. So suppose that we have two strings σ and ρ , and $\rho = \sigma\tau$ for some τ . Then we say that ρ extends σ and write $\rho \succ \sigma$. If ρ has the potential to equal σ , then we write $\rho \succeq \sigma$.

We will also bring up the notion of concatenation or the combining of strings. Let $\sigma, \tau \in 2^{<\omega}$. Then $\sigma\tau$ is simply the string σ followed by the string τ . For example if $\sigma = 110$ and $\tau = 101$, then $\sigma\tau = 110101$. (Downey & Hirschfeldt, 2010)

Lastly we can also represent real numbers from the interval $[0, 1]$ as binary sequences as well. We simply start with a decimal point and then add the binary sequence after. We will go more into this when we talk about Martingales.

2.1 Length-Lexicographic Ordering

Let σ and τ be strings. If either $|\sigma| < |\tau|$ or else both $|\sigma| = |\tau|$ and $\sigma(n) = 0$ for the least n such that $\sigma(n) \neq \tau(n)$, then we can say σ is less than τ . We call this the *Length-Lexicographic Ordering* of $2^{<\omega}$. (Downey & Hirschfeldt, 2010)

3 Important notes from Computability Theory

In this section we will give a brief overview over the area of Computability Theory since the concept of Algorithmic Randomness will be based around this logic. There are many different ways to handle computability. For example we could look at every possible computer program. Many people have attempted to define computability and there are many definitions that are equivalent. In this paper we will

use the Turing Machine definition.¹

3.1 Functions

Let's define what a function is for the purposes of this paper. The first type of function is what we call a *total function*. These are functions whose domain is \mathbb{N} and outputs elements from \mathbb{N} . The next type of functions are *partial functions*. These have domains that are subsets of \mathbb{N} who also output elements from \mathbb{N} . Therefore we can say that all functions are partial functions but not necessarily the other way around.

Suppose that x is in the domain of some function f . Then we write $f(x) \downarrow$ and say that the function *halts* or *converges*. If x isn't in f 's domain, then we say that f *diverges* and write $f(x) \uparrow$.

Definition 3.1.1 For a set A , the *characteristic function* of A is the following total function.

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

Often we will simply represent χ_A with A . So we will write $A(n)$ to mean $\chi_A(n)$. Therefore $A(n) = 1$ implies that $n \in A$ and $A(n) = 0$ implies that $n \notin A$.

3.2 Turing Machines

One of the more significant and rigorous definitions of computability is given by Alan Turing. He thought about how people do computations on paper and abstracted these actions into 4 parts: i) they read the symbols, ii) they write the symbols, iii) they hold these symbols in their mind, and iv) they look at other symbols on the paper. With these ideas in mind, we will introduce the next topic.

A *Turing Machine*, sometimes denoted by TM, is an idealized computer that has a tape it can read and write on, a head that does the actual reading and writing, and moves back and forth along the tape, and an internal state which changes depending on what

¹Note that unless stated otherwise, the information in this section will come from Weber's book on Computability Theory.

happens during the computation. It also has a list of rules that are applied based on the current internal state and symbol that the head is reading.

The tape in the Turing Machine is divided into squares, each with a symbol on it, and the head rests on a single square at a time. The head will move left or right one square depending on the internal state. We specify Turing Machines with a finite set of quadruples, $\langle a, b, c, d \rangle$. Note that some authors use finite sets of quintuples, but for the purposes of this paper we will stick to the quadruples. These are sets of instructions that are decoded as follows:

- a is the state the TM is currently in
- b is the symbol the TM's head is currently reading
- c is an instruction to tell the head to either write or move
- d is the state the TM is in at the after the instruction is done

Let's look at an example, $\langle q_5, 1, 0, q_3 \rangle$ means "I am in state q_5 and I am currently reading 1. I must change 1 to 0 and now I am in state q_3 ." If we wish to move Left or Right, typically we will write L or R respectively. Sometimes we will have the symbol $(*)$ in position c which tells the machine to delete whatever it is reading.

Any computation that yields an output must only use a finite number of squares on the tape and finitely many steps for the computation. Since Turing Machines represent idealized computers we must allow them unlimited time and memory to complete their computation. It is important to differentiate this from infinite time and memory. Rather this means that we allow the TM as much time and memory as it needs to finish its computation. However we cannot bound the tape at the beginning. What if the computation is one square short? Thus the tape of the TM is infinite even though the computation will only use a finite length of it.

But how does the TM know when to stop? There are states that we call halting states, and they can be described as moments in the computation when the machine can no longer do any more instructions.

So far we've explained how Turing Machines operate, but we have yet to explain how they actually take in values and give outputs. Turing Machines take in elements from \mathbb{N} and convert them either into their binary forms or $n+1$ 1s. We say $n+1$ vs n to allow 0 a place on the tape. We will focus primarily on the $n+1$ 1s representation.

So the machine takes in natural numbers represented as $n+1$ 1s, and after the computation ends the output will be the number of 1s on the tape. Note that TMs can have a variety of inputs and outputs. For example they can have strings for their inputs and outputs as we will discuss later.

Definition 3.2.1: Let $A \subset \mathbb{N}$. A partial function, $f : A \rightarrow \mathbb{N}$, is partial computable if there is a TM that can compute it. In other words, if M is a TM that computes f , then $\forall x \in A, f(x) \in \mathbb{N}$ is the output computed by M .

A similar definition can be made for total computable functions, but we will just call them computable functions.

The Church Turing Thesis: Something important to bring up is the Church-Turing Thesis which essentially states that every intuitively computable function is a Turing-computable function (a function that can be computed using a Turing Machine). Therefore if we have an algorithm that describes a function, we may assume that there is a TM that computes it, and we don't necessarily need to know its instructions. (Brodkorb, 2019)

3.3 Coding and Countability

Before we can talk about a Universal Turing Machine, we must first introduce the notions of countability and coding. How could we define computability outside the domains of \mathbb{N} ? If the domain we want is countable, then we may be able to code its members as elements of \mathbb{N} . For example, we could code \mathbb{Z} into \mathbb{N} using the following function.

$$f(x) = \begin{cases} 2x & x \geq 0 \\ -2x - 1 & x < 0 \end{cases}$$

Observe that any even numbers would represent positive integers while any odd numbers would be negative integers. The important property we need in order to treat elements from \mathbb{N} as codes of elements from another set S is that we need a computable bijection between S and \mathbb{N} . Thus we can have a computable inverse with this as well. Sets with this property are called *effectively countable*, and the image of the element of S under this bijection is its *code*. The main reason we wish for there to be a bijection between \mathbb{N} and

S is that this will form a unique mapping between every pair of elements between S and \mathbb{N} . Lastly, the effectiveness of this function is important since bijection cannot be used in a Turing Machine unless it is computable.

Often coding is ignored in research papers since one simply assumes a coding function between some countable set S and \mathbb{N} is fixed. Now we wish to move into higher dimensions of the naturals. Let's consider \mathbb{N}^2 . We will bring up the idea of *pairing functions*. Pairing functions are those that send n-tuples from \mathbb{N}^n to \mathbb{N} . There is a standard pairing function indicated below that pairs 2-tuples to \mathbb{N} . It is defined as

$$\langle x, y \rangle := \frac{1}{2}(x^2 + 2xy + y^2 + 3x + y).$$

If we have higher tuples of naturals, we can still use this function. For example, $\langle x, y, z \rangle := \langle \langle x, y \rangle, z \rangle$. This can be extended to n-tuples. We will not go into too much detail where the standard pairing function comes from as it is not necessary for the rest of this paper, but it is not difficult to derive.

One interesting observation about the set of Turing Machines is that since each TM is defined by a finite set of information, it is effectively countable. Just like the standard pairing function we will not go into too much detail with this, but do note that since Turing Machines are finite sets of 4-tuples we can code them with multiple instances of said pairing function. Another thing to keep in mind is that they will also include what we call “junk machines”, or machines that don't really do anything. We will also get machines that will compute the same functions.

Definition 3.3.1: We call the code of a Turing Machine its *index*. When we choose a particular coding for a Turing Machine, we say we *fix an enumeration* of the Turing Machines. Often we use φ for partial function and φ_e represents the e^{th} machine in the enumeration or the machine with index e .

Definition 3.3.2 (Downey & Hirschfeldt, 2010): We say that a set $A \subset \mathbb{N}$ is *computably enumerable* (c.e.) if there is some Turing Machine, M , such that $A = \text{dom}(M)$. Since Turing Machines can represent partial functions, we could also say for some e that $A = \text{dom}(\varphi_e)$.

3.4 Universal Turing Machines

Here we will define a *Universal Turing Machine* U . This is a machine that represents every other machine.

$$U(\langle e, x \rangle) = \varphi_e(x).$$

U decodes $\langle e, x \rangle$ into the pair (e, x) , decodes e into the appropriate set of quadruples, and uses x as the input.

3.5 Kleene's Recursion Theorem

One significant result from Computability Theory is Kleene's Recursion Theorem. It allows us take the index from a computable function that we are creating and use it in the construction of the very thing we are trying to construct.

Theorem 3.5.1: (Nies, 2009): If f is a computable function, then there is some number n such that

$$\varphi_n = \varphi_{f(n)}.$$

The index n is also called a *fixed point* of f .

3.6 Reals

Here we will bring up the notion of reals. Reals for the purposes of this topic are irrational numbers on the interval $[0, 1]$ and will have unique binary expansions. We identify them with elements from 2^ω and subsets of \mathbb{N} . In other words if $A \subset \mathbb{N}$, then a real α can be defined as

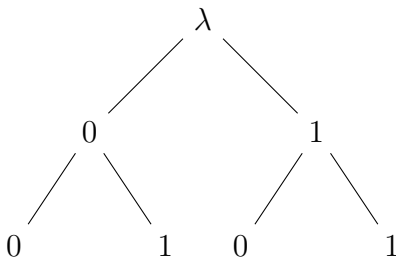
$$\alpha := 0.\chi_A(1)\chi_A(2)\chi_A(3)\chi_A(4)\dots$$

Definition 3.6.1: Let D be a countable set. A function $f : D \rightarrow \mathbb{R}$ is c.e. if there is a computable sequence of computable functions that approximate f from below; that is, $f_s : D \rightarrow \mathbb{Q}$ such that for every s and $x \in D$ we have that $f_s(x) \leq f_{s+1}(x)$ and $\lim_{s \rightarrow \infty} f_s(x) = f(x)$. (Downey & Hirschfeldt, 2010)

4 Some Basics on Measure Theory

A measure on a set is a way to describe the size or length of said set. When we wish to define the measure of a particular set,

S, we used the notation $\mu(S)$. However the measure of sets of binary sequences can be very tricky to think about. Let's look at the following diagram to see how we measure binary sequences.



When talking about measuring binary sequences, we look at the *extensions* of finite strings. Let $\sigma \in 2^{<\omega}$. Then the extensions of σ , denoted by $[[\sigma]]$, is the set of infinite sequences whose initial subsequence is σ , that is, $[[\sigma]] = \{\sigma X : X \in 2^\omega\}$. For example, if $\tau = 10011101$, then $[[\tau]]$ is the set of all sequences whose initial part is 10011101.

If we wish to measure $[[\sigma]]$, then we find the probability of each sequential bit being either a 1 or 0. Then the measure of $[[\sigma]]$, defined as $\mu([[\sigma]])$, is $\mu([[\sigma]]) = 2^{-|\sigma|}$. For example suppose $\sigma = 101$. Then $\mu([[\sigma]]) = 2^{-|101|} = 2^{-3} = \frac{1}{8}$.

Let $A \subset 2^{<\omega}$. Then $[[A]] := \bigcup_{\sigma \in A} [[\sigma]]$. In words this means that the extensions of some subset of the finite binary strings, A, is the union of the extensions of σ for every $\sigma \in A$. A sequence of binary basic open sets $\{[[\sigma]] : \sigma \in A\}$ is said to *cover* a set $C \subset 2^\omega$ if $C \subset [[A]]$. The *outer measure* of C is

$$\mu^*(C) = \inf \left\{ \sum_n 2^{-|\sigma_n|} : \{[[\sigma_n]] : n \in \mathbb{N}\} \text{ covers } C \right\}.$$

In words, first we look at all the ways we can cover C. Then we sum up the measures of every extension and take the infimum of this. Next let \bar{C} be the complement of C, that is, $\bar{C} = 2^\omega \setminus C$. The *inner measure* of C is $\mu_*(C) = 1 - \mu^*(\bar{C})$. If C is measurable, then $\mu^*(C) = \mu_*(C)$, and this quantity is the *measure* $\mu(C)$. (Downey & Hirschfeldt, 2010)

5 Kolmogorov Complexity

² Here we will introduce some notions of Complexity, specifically plain and prefix-free Kolmogorov Complexity. Note that TMs from here will take strings as their inputs and give strings as their outputs.

5.1 Plain Kolmogorov Complexity

Let M be a machine and $M(\sigma) = \tau$. Then we say that σ is an M -description of τ . Note that an M -description of τ can have shorter length than τ . Thus we can think of σ as a compressed form of τ sort of how we described in the introduction paragraph of this section. The main reason why we find this interesting is to see how well τ is being compressed. In other words, we want the length of the shortest description. Hence we introduce a new definition.

Definition 5.1.1 Let M be a machine. The *Kolmogorov Complexity* of a string $\tau \in 2^{<\omega}$ with respect to M is the smallest length of $\sigma \in 2^{<\omega}$ that yields τ in M , that is,

$$C_M(\tau) = \min\{|\sigma| : M(\sigma) = \tau\},$$

where the minimum is taken to be ∞ if the set is empty.

Observe that our value for complexity is dependent on our machine M . Hence we would like to remove this dependence. Recall back to our previous definition of Universal Turing Machines. For any machine φ_e , there is a Turing Machine such that $U(\langle e, x \rangle) = \varphi_e(x)$. Hence $U(\langle e, \sigma \rangle) = M_e(\sigma)$ where M_e is the e^{th} machine that takes in strings. Here we will alter this notion. For the U described above, there is a string ρ for every machine M such that

$$U(\rho\sigma) = M(\sigma).$$

These strings are called *coding strings* and their respective lengths are the *coding constants*. Then for every string σ ,

$$C_U(\sigma) \leq C_M(\sigma) + |\rho|.$$

Now that we have a universal description for any machine, we can now define the (plain) Kolmogorov Complexity as

$$C(\sigma) = C_U(\sigma)$$

²The source for this information comes from section 3 in Downey and Hirschfeldt's book on Algorithmic Randomness and Complexity.

for every $\sigma \in 2^{<\omega}$.

Next let's observe the following theorem as we will need it later.

Theorem 5.1.2 Let k be a fixed constant. If μ is sufficiently long enough, then there is an initial segment, σ , of μ such that $C(\sigma) < |\sigma| - k$.

Proof: Let μ be a sufficiently long enough string and k be a fixed constant. Let ν be an initial segment of μ with n being such that ν is the n^{th} string in the Length-Lexicographic Ordering of $2^{<\omega}$. Now let ρ be such that $|\rho| = n$ and it takes up the next n bits of μ . Lastly let $\sigma = \nu\rho$. Note that by the Church-Turing Thesis we can build a machine M such that $M(\rho) = \sigma$. Let c be the coding constant of M in U . Thus

$$C(\sigma) \leq C_M(\sigma) + c = |\rho| + c,$$

However $|\sigma| = |\nu| + |\rho|$ and if we choose ν such that $|\nu| > c + k$, then we have that

$$C(\sigma) \leq |\rho| + c = |\sigma| - |\nu| + c < |\sigma| - |\nu| + |\nu| - k = |\sigma| - k.$$

Hence $C(\sigma) < |\sigma| - k$.

5.2 Prefix-Free Kolmogorov Complexity

Here we will introduce the notion prefix-free Kolmogorov Complexity. The main reason we talk about this concept is to deal with algorithmic randomness of infinite sequences. Many people believe that this notion of Kolmogorov Complexity is the correct notion vs plain Kolmogorov Complexity.

One argument for this is that with plain Kolmogorov Complexity, if τ is a description of σ , then the bits of τ contains all the information necessary to yield σ . But a Turing Machine M might produce σ by not only using the bits of τ , rather it may also use the length as well. If M is allowed to use the length of τ , then there must be some termination symbol T on M 's input tape following the bits of τ . Therefore the output that is based on the output alphabet $\{0, 1\}$ is dependent on the following input alphabet $\{0, 1, T\}$ which many view as cheating.

Definition 5.2.1: A set $A \subset 2^{<\omega}$ is *prefix-free* if it is an antichain with respect to the natural partial ordering of $2^{<\omega}$; that is, for every $\sigma \in A$ and every τ properly extending σ , then $\tau \notin A$.

Prefix-free functions and *prefix-free Turing machines* are those whose domains are prefix-free. It is usual to consider a machine such as this as *self-delimiting* which basically means that it has a one-way read head that halts when the machine accepts the string described by the bits read so far. The reason why is so that the machine is forced to accept the strings without knowing whether or not there are more bits on the input tape.

Definition 5.2.2: Let U be a prefix-free universal TM. Then the *prefix-free Kolmogorov Complexity* of σ is said to be

$$K(\sigma) = C_U(\sigma).$$

6 Cornerstones of Algorithmic Randomness

³ Here will look at three major approaches to defining algorithmic randomness for infinite sequences.

The first is the *computational paradigm*. The idea behind this paradigm is that the initial segments of random sequences are hard to describe.

The second is the *measure-theoretic paradigm*. With this notion, random sequences are those with no “rare” properties, that is, a random sequence should pass all effective (or computable) statistical tests.

The last is the *unpredictability paradigm*. This approach is based on the idea that we should not be able to predict the next bit in a random sequence even if we know what every previous bit is, hence the name “unpredictability”. One can look at this like coin tosses where each output of flipping a coin infinitely many times will be unpredictable.

We will use all of these approaches to define three equivalent definitions of randomness which we will call Martin-Löf Randomness or 1-Randomness.

6.1 The Computational Paradigm

As mentioned previously, random sequences in the sense of the computational paradigm are those whose initial segments are hard

³Unless stated otherwise, the source for the information in this section comes primarily from chapter 6 of Downey and Hirschfeldt’s book on Algorithmic Randomness.

to describe.

A natural first attempt to defining randomness for some set A would be to say for every n , $C(A \upharpoonright n) \geq n - d$ for some d . However this contradicts theorem 5.1.2 so no such set exists. However we can correct randomness for sequences with complexity by using prefix-free complexity.

Definition 6.1.1 A sequence A is *1-random* if there is some d such that $K(A \upharpoonright n) \geq n - d$ for all n .

What this is saying is if for every n the length of the shortest universal prefix-free description of $A \upharpoonright n$ is greater than or equal to $n - d$, then we cannot describe any initial segment of A in a shorter/nicer way. Hence every initial segment is hard to describe. Next we will introduce a particular number that we will prove is 1-random.

Definition 6.1.2: Let U be a universal prefix-free machine. The *halting probability* Ω (sometimes called *Chaitin's Ω*) of U is

$$\Omega = \mu(\llbracket \text{dom } U \rrbracket) = \sum_{\sigma \in \text{dom } U} 2^{-|\sigma|}.$$

Let $\Omega_s = \sum_{\sigma \in \text{dom } U[s]} 2^{-|\sigma|}$. Note that $U[s]$ is the universal machine up to it's s^{th} stage, so $\text{dom } U[s]$ is the set of every string included up to this point. We care about Ω_s because it is computable whereas Ω is not. Note $\lim_{s \rightarrow \infty} \Omega_s = \Omega$. Observe that the value of Ω is dependent on the choice of U but each Ω is closely related to each other.

We will not prove this result but note that Ω is irrational. Hence $\lim_{s \rightarrow \infty} \Omega_s \upharpoonright n = \Omega \upharpoonright n$. What this means is given an n after some stage t for all $s > t$, $\Omega_s \upharpoonright n = \Omega \upharpoonright n$. The same cannot be said if Ω was taken to be rational. In order to better understand this let's suppose Ω could be rational and let $\Omega = 0.101$. Then a sequence that approaches this could be 0.1 , 0.10 , 0.1001 , 0.10011, 0.1001111, etc. Because Ω_s can never equal Ω there will never be an s such that $\Omega_s \upharpoonright 3 = 0.101$. However because Ω is irrational for every n , there will always be an s such that $\Omega_s \upharpoonright n = \Omega \upharpoonright n$.

But why do we care about this? Specifically why do we care about Ω_s ? Recall that this is actually computable whereas Ω isn't, and we wish to show that Ω is 1-random. The definition of 1-randomness uses the concept of Turing Machines and we will need something

that is computable. Hence we will look Ω 's approximations from below to prove this. In particular we need our proof to be such that for every n , $\lim_{s \rightarrow \infty} \Omega_s \upharpoonright n = \Omega \upharpoonright n$, which would not be the case if Ω was taken to be rational.

Theorem 6.1.3: Ω is 1-random

Proof: First we build a prefix-free TM dependent on some n , so that $M = \Phi_{f(n)}$. By the Recursion Theorem there is a c such that $\Phi_{f(c)} = \Phi_c$. Hence $\Phi_c = M$, so we may assume that we know the index c of this machine that we are building in advance. However this is for machines that take in numbers as their inputs. Let's adapt this to machines that take in strings. We may assume to know the coding string, ρ , of M in some universal machine U and its length, $|\rho| = c$. Then for every $\sigma \in \text{dom}(M)$, $U(\rho\sigma) = M(\sigma)$.

Next we run U until we have a potential compression for some initial segment of Ω_s . So we wait for some stage s , a string τ , and an n such that $U(\tau)[s] = \Omega_s \upharpoonright n$ with $|\tau| < n - c$. Hence $K(\Omega_s \upharpoonright n) < n - c$. Let δ be such that $\delta \notin \text{rng } U[s]$. Then by the Church-Turing Thesis we can define $M(\tau) = \delta$. Since M is coded in U by ρ , there must be a $\nu = \rho\tau$ such that $U(\rho\tau) = M(\tau)$. So $|\nu| = |\tau| + |\rho| = |\tau| + c < n$ and $U(\nu) = M(\tau) = \delta$. Since $\delta \notin \text{rng } U[s]$, it follows that $\nu \notin \text{dom } U[s]$. Since $\nu \in \text{dom } U$ but $\nu \notin \text{dom } U[s]$, we have that $2^{-|\nu|}$ contributes to the value of Ω but not Ω_s . In other words, $2^{-|\nu|}$ contributes to the value of Ω after the s^{th} stage; that is, $2^{-|\nu|}$ contributes to $\Omega - \Omega_s$. Thus $\Omega - \Omega_s \geq 2^{-|\nu|} > 2^{-n}$ which implies that $2^{-n} + \Omega_s < \Omega$. Then in order to get Ω we must add a constant greater than 2^{-n} thus changing the first n bits of Ω_s . Note that we do this for every initial segment of Ω_s that may have a potential compression.

Therefore if we ever have for any n and s , $\Omega_s \upharpoonright n$ is compressible by more than c bits, then $\Omega \upharpoonright n \neq \Omega_s \upharpoonright n$. Thus if $|\tau| < n - c$, then $U(\tau)$ cannot be $\Omega \upharpoonright n$. Hence $K(\Omega \upharpoonright n)$ cannot be less than $n - c$ as stated above for $K(\Omega_s \upharpoonright n)$. Therefore $\forall n (K(\Omega \upharpoonright n) \geq n - c)$ which would make Ω 1-random.

6.2 The Measure-Theoretic Paradigm

The main idea behind the Measure-Theoretic paradigm is that random sequences should have no nice, rare properties. In other words, if we have some way of looking at every possible nice, rare

property then a random sequence should not satisfy any of them. Martin-Löf introduced the test concept; that is, tests that determine the properties described above.

Definition 6.2.1 (Downey & Reimann, 2007): A *Martin-Löf Test* (*ML-Test*) is a computably enumerable set $W \subset \mathbb{N} \times 2^{<\omega}$ such that, with $W_n := \{\sigma : (n, \sigma) \in W\}$, it holds true for every n

$$\mu(\llbracket W_n \rrbracket) = \sum_{\sigma \in W_n} 2^{-|\sigma|} \leq 2^{-n}.$$

Observe that the intersection of these sets forms a measure 0 set. So the niceness of these tests comes from the fact that they are c.e. while the rareness is because their intersection has measure 0. Now let $A \in 2^\omega$. A passes the test if

$$A \notin \bigcap_n \llbracket W_n \rrbracket.$$

In other words, A passes an ML-test if there is an n such that $\forall \sigma \in W_n, A \notin \llbracket \sigma \rrbracket$. To get a better idea, let's look at an example if A was not random.

Example 6.2.2: Let $A = 10100110100101100110\dots$. At first this seems random, but let's look at A a little differently!

$$A = 10\ 10\ 01\ 10\ 10\ 01\ 01\dots$$

Notice that every 2 bits has both a 1 and a 0. Clearly this isn't random. If we we're to translate this to a test concept, we could say let $W \subset \mathbb{N} \times 2^{<\omega}$ be a ML test with

$$\begin{aligned} W_1 &= \{10, 01\} \\ W_2 &= \{1010, 0101, 1001, 0110\} \\ &\vdots \end{aligned}$$

Taking the measure of each set of extensions

$$\begin{aligned} \mu(\llbracket W_1 \rrbracket) &= 2^{-|10|} + 2^{-|01|} = \frac{1}{4} + \frac{1}{4} = \frac{1}{2} \leq 2^{-1} \\ \mu(\llbracket W_2 \rrbracket) &= 2^{-|1010|} + \dots + 2^{-|0110|} = \frac{1}{16} + \dots + \frac{1}{16} = \frac{1}{4} \leq 2^{-2} \\ &\vdots \end{aligned}$$

So each set satisfies the measure condition. Notice that no matter the pattern of 1's and 0's for A if every 2 bits has both a 1 and a 0, there is surely a $\sigma \in W_n$ for every n such that $A \in \bigcap_n \llbracket W_n \rrbracket$. Thus A has this nice property which makes it nonrandom.

Fact 6.2.3 (Nies, 2009): One may uniformly in an index n for a c.e. open set W obtain a computable antichain X such that $\llbracket X \rrbracket = W_n$. Moreover, X is given in the form $X = \{\sigma_i\}_{i < N}$ for $N \in \mathbb{N} \cup \{\infty\}$ where $\sigma_i \neq \sigma_j$ for $i \neq j$ and $|\sigma_i| \leq |\sigma_{i+1}|$.

Next we present a Lemma.

Lemma 6.2.4 (Downey & Hirschfeldt, 2010): Let M be a prefix-free machine, let $k \in \mathbb{N}$, and $S_k = \{\sigma : K_M(\sigma) \leq |\sigma| - k\}$. Then $\mu(\llbracket S_k \rrbracket) \leq 2^{-k} \mu(\llbracket \text{dom}(M) \rrbracket)$.

Proof: Note that for every $\sigma \in S_k$, there is a string τ such that $M(\tau) = \sigma$ and $|\tau| \leq |\sigma| - k$. So $|\sigma| \geq |\tau| + k$. Therefore

$$\mu(\llbracket S_k \rrbracket) = \sum_{\sigma \in S} 2^{-|\sigma|} \leq \sum_{\tau \in \text{dom}(M)} 2^{-k} 2^{-|\tau|} = 2^{-k} \mu(\llbracket \text{dom}(M) \rrbracket).$$

Note that if M were universal, then $\mu(\llbracket S_k \rrbracket) = 2^{-k} \mu(\llbracket \text{dom}(M) \rrbracket) = 2^{-k} \Omega \leq 2^{-k}$ since $\Omega \in [0, 1]$.

Requests

Before we begin our proof to show that 1-randomness is equivalent to ML-randomness, we must briefly cover a few small topics before hand so it suffices to bring it up here. We will introduce a new method of building machines. A *request* is a pair $\langle r, \sigma \rangle$ from $\mathbb{N} \times 2^{<\omega}$ that asks for a machine M that produces σ with a string of length r . For the purposes of this topic we will be dealing with c.e. sets of requests rather than computable lists since those could allow repeats. (Nies, 2009)

For example let $W = \{\langle 1, 101 \rangle, \langle 2, 1101 \rangle\}$. Then we are requesting for a machine M where there are strings $|\tau| = 1$ and $|\rho| = 2$ such that $M(\tau) = 101$ and $M(\rho) = 1101$.

Definition 6.2.5 (Nies, 2009): Let $W \subset \mathbb{N} \times 2^{<\omega}$ be a c.e. set of requests. If

$$\sum_{\langle r, \sigma \rangle \in W} 2^{-r} \leq 1$$

for every request in W , then we call W a *bounded request set*. Furthermore let $S \subset 2^{<\omega}$. The *weight* of S given by W is said to be

$$\text{wgt}_W(S) = \sum_{\langle r, \sigma \rangle \in W} 2^{-r}$$

provided that σ is also in S . The *total weight* of W is $\text{wgt}_W(2^{<\omega})$.

Theorem 6.2.6 (Nies, 2009): Machine Existence Theorem
For every bounded request set W , one can computably obtain a prefix-free machine M_d for $d > 1$ such that $\forall r, \sigma, \langle r, \sigma \rangle \in W$ if and only if there is a string $|\rho| = r$ such that $M_d(\rho) = \sigma$.

Theorem 6.2.7 [(Downey & Hirschfeldt, 2010), (Nies, 2009)]: A sequence is ML-random if and only if it is 1-random.

Proof: (\Rightarrow) Let $W_k = \{\sigma : K(\sigma) \leq |\sigma| - k\}$ and define W to be the c.e. set of pairs (k, σ) such that $\sigma \in W_k$. By Lemma 6.2.4 we have that

$$\mu(\llbracket W_k \rrbracket) \leq 2^{-k} \Omega \leq 2^{-k}$$

Therefore W is a ML-test.

Let $A \in 2^\omega$ be ML-random. Then $A \notin \bigcap_k \llbracket W_k \rrbracket$. So there is a i such that for every n , $A \upharpoonright n \notin W_i$. Hence $K(A \upharpoonright n) > n - i$. Thus A is 1-random.

(\Leftarrow) Suppose A isn't ML-random. Then there is a ML-test Y such that $A \in \bigcap_n \llbracket Y_n \rrbracket$. Note that Y_n can be replaced with Y_{2n} so by definition of ML-tests $\mu(\llbracket Y_n \rrbracket) \leq 2^{-2n}$. By fact 6.2.3 we can uniformly in n obtain a computable antichain $X = \{\sigma_i^n\}_{i < N_n}$ for $N_n \in \mathbb{N} \cap \{\infty\}$ such that $\llbracket Y_n \rrbracket = \llbracket X \rrbracket$.

Let $L = \{(|\sigma_i^n| - n + 1, \sigma_i^n) : n \in \mathbb{N}, i < N_n\}$. Then the weight of $\llbracket Y_n \rrbracket$ given by L

$$\sum_{\sigma_i^n \in Y_n} 2^{-|\sigma_i^n| + n - 1} = 2^{n-1} \sum_{\sigma_i^n \in Y_n} 2^{-|\sigma_i^n|} \leq 2^{n-1} 2^{-2n} = 2^{-n-1}.$$

Then we can say that

$$\begin{aligned}
\sum_{n \in \mathbb{N}} \text{wgt}_L(Y_n) &= \text{wgt}_L(Y_1) + \text{wgt}_L(Y_2) + \text{wgt}_L(Y_3) + \dots \\
&\leq \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \dots \\
&= \frac{1}{4} \left(\frac{1}{2} \right)^{n-1} \\
&= \frac{1/4}{1 - 1/2} \\
&= \frac{1}{2} < 1.
\end{aligned}$$

Since the weight of L is less than 1, by definition 6.2.5 L is a bounded request set.

By the Machine Existence Theorem we can build a machine M with coding constant d for L . Fix $b \in \mathbb{N}$ and let $n = b + d + 1 \implies -b = -n + 1 + d$. Hence

$$K(\sigma_i^n) \leq (|\sigma_i^n| - n + 1) + d = |\sigma_i| - b.$$

Since $A \in \llbracket Y_n \rrbracket$, for all n , A extends σ_i^n for some i . Thus A isn't 1-random.

6.3 The Unpredictability Paradigm

The third cornerstone to algorithmic randomness is the idea that random sequences should not be predictable. The way most people look at this is in the sense of betting. In other words, one should not be able to get rich off of something random. Let's explain with an example.

Example 6.3.1: Suppose we have the string $\sigma = 0011110110101$ and we are betting on whether or not the next bit will be 0 or 1. Let $B(\sigma) \geq 0$ be the current capital gained so far and α , $0 \leq \alpha \leq B(\sigma)$ be the amount that we wish to bet. Let's say we wish to bet α on the next bit being 0. Then $B(\sigma 0) = B(\sigma) + \alpha$ and $B(\sigma 1) = B(\sigma) - \alpha$. Hence we have that

$$B(\sigma 0) + B(\sigma 1) = B(\sigma) + \alpha + B(\sigma) - \alpha = 2B(\sigma).$$

Definition 6.3.2: A *martingale* is a function $B : 2^{<\omega} \rightarrow \mathbb{R}^{\geq 0}$ that satisfies the equality

$$B(\sigma 0) + B(\sigma 1) = 2B(\sigma)$$

for every $2^{<\omega}$.

What we have just done is formalize the concept of a betting strategy. One advantage to using martingales over Complexity or ML-tests is that they have algebraic properties as we have just seen.

Let B be a martingale and $Z \in 2^\omega$. Then we say that B *succeeds* on Z if

$$B(Z) := \limsup_n B(Z \upharpoonright n) = \infty.$$

What this means is that there is no upper bound to our capital gains i.e. we can get very rich by betting on Z . Let $\text{Succ}(B) = \{Z : B(Z) = \infty\}$. So intuitively we may think that if some sequence $A \in \text{Succ}(B)$, then A surely isn't random. We will show this in a later proof. Next we will define a notion that is a little broader than martingales.

Definition 6.3.3: A *supermartingale* is a function $S : 2^{<\omega} \rightarrow \mathbb{R}^{\geq 0}$ that satisfies the inequality

$$2S(\sigma) \geq S(\sigma 0) + S(\sigma 1)$$

for all $\sigma \in 2^{<\omega}$.

The idea behind supermartingales is that they are martingales but with some extra money that the gambler can waste in addition to gambling with it. Similar to the above results S succeeds on a sequence Z if $S(Z) = \limsup_n (Z \upharpoonright n) = \infty$. Likewise $\text{Succ}(S)$ is the set of sequences that S succeeds on. Note that in a lot of situations we will say (super)martingales to mean that this applies to both martingales and supermartingales.

Proposition 6.3.4: Let B be a (super)martingale.

(i) Let $\sigma \in 2^{<\omega}$ and $S \subset \llbracket \sigma \rrbracket$ be prefix-free. Then

$$\sum_{\tau \in S} 2^{-|\tau|} B(\tau) \leq 2^{-|\sigma|} B(\sigma).$$

(ii) Let $U_n = \{\sigma : B(\sigma) \geq n\}$. Then

$$\mu(\llbracket U_n \rrbracket) \leq \frac{B(\lambda)}{n}.$$

Proof of part (i): We will use mathematical induction. Let S be a prefix-free finite set of extensions of some string σ . Note that it is enough to consider S to be finite. If we were to consider S to be infinite, then all we need to show is that the limit of the finite cases holds true.

Base Case: Suppose $|S| = 1$. If $\tau \in S$, then there is some $\rho \in 2^{<\omega}$ such that $\tau = \sigma\rho$. Hence $|\tau| \geq |\sigma|$. Thus

$$\sum_{\tau \in S} 2^{-|\tau|} B(\tau) = 2^{-|\tau|} B(\tau) = 2^{-|\sigma|} 2^{-|\rho|} B(\sigma\rho)$$

Recall by definition of (super)martingales

$$B(\sigma) \geq 2^{-1}[B(\sigma 0) + B(\sigma 1)].$$

For example,

$$B(\sigma) \geq 2^{-1}B(\sigma 0) \geq 2^{-2}B(\sigma 00) \dots$$

So at some point $2^{-|\rho|}B(\sigma\rho) \leq B(\sigma)$. Hence $2^{-|\sigma|}2^{-|\rho|}B(\sigma\rho) \leq 2^{-|\sigma|}B(\sigma)$. Thus $\sum_{\tau \in S} 2^{-|\tau|}B(\tau) \leq 2^{-|\sigma|}B(\sigma)$ if S has only 1 element.

Inductive Hypothesis: Let's assume it is true that if $|S| = n$, then

$$\sum_{\tau \in S} 2^{-|\tau|} B(\tau) \leq 2^{-|\sigma|} B(\sigma).$$

Inductive Step: Suppose $|S| = n + 1$. Let ρ be the longest string such that every element in S extends ρ . So if $\tau \in S$, then τ extends ρ . Thus $|\tau| \geq |\rho|$. Now let $S_i = \{\tau \in S : \tau \text{ extends } \rho i\}$ with $i \in \{0, 1\}$. Then $|S_i| \leq |S|$ since S_i is restricted to less extensions. Thus by the inductive hypothesis

$$\sum_{\tau \in S_i} 2^{-|\tau|} B(\tau) \leq 2^{-|\rho i|} B(\rho i).$$

Note that $2^{-|\rho i|} B(\rho i) = 2^{-(|\rho|+1)} B(\rho i) \leq 2^{-(|\rho|+1)} (B(\rho 0) + B(\rho 1)) = 2^{-1} 2^{-|\rho|} (B(\rho 0) + B(\rho 1))$. We know this to be true since $B(\rho i)$ is either $B(\rho 0)$ or $B(\rho 1)$. Thus $B(\rho i) \leq B(\rho 0) + B(\rho 1)$. By the definition of (super)martingale

$$2^{-1} 2^{-|\rho|} (B(\rho 0) + B(\rho 1)) \leq 2^{-|\rho|} B(\rho).$$

Hence we have that $2^{-|\rho i|}B(\rho i) \leq 2^{-|\rho|}B(\rho)$. Lastly we have that $|\rho| \geq |\sigma|$. Thus by the base case, we have that

$$2^{-|\rho|}B(\rho) \leq 2^{-|\sigma|}B(\sigma).$$

Thus $\sum_{\tau \in S} 2^{-|\tau|}B(\tau) \leq 2^{-|\rho i|}B(\rho i) \leq 2^{-|\rho|}B(\rho) \leq 2^{-|\sigma|}B(\sigma)$.

Proof of part (ii): Let $U_n = \{\sigma : B(\sigma) \geq n\}$. Let $P \subset U_n$ be a prefix-free set such that $\llbracket P \rrbracket = \llbracket U_n \rrbracket$. Then

$$\mu(\llbracket P \rrbracket) = \sum_{\tau \in P} 2^{-|\tau|} \leq \frac{1}{n} \sum_{\tau \in P} 2^{-|\tau|}B(\tau).$$

We know this to be true by the way U_n is defined above. By the result in part (i) with $\sigma = \lambda$ we have that

$$\frac{1}{n} \sum_{\tau \in S} 2^{-|\tau|}B(\tau) \leq \frac{2^{-|\lambda|}B(\lambda)}{n} = \frac{B(\lambda)}{n}.$$

Thus $\mu(\llbracket P \rrbracket) = \mu(\llbracket U_n \rrbracket) \leq \frac{B(\lambda)}{n}$.

In order to define randomness in the sense of martingales, we must first restrict our class of martingales to c.e. martingales. A (super)martingale is computably enumerable in the sense of definition 3.6.1.

Theorem 6.3.5: A set is ML-random if and only if no c.e. (super)martingale succeeds on it.

Proof: (\Rightarrow) Let B be a c.e. (super)martingale. Without loss of generality we may assume $B(\lambda) = 1$. Let $W_n = \{\sigma : B(\sigma) \geq 2^{-n}\}$ with $W \subset \mathbb{N} \times 2^{<\omega}$ being the c.e. set of tuples (n, σ) for $\sigma \in W_n$. Then by the second part of proposition 6.3.4

$$\mu(\llbracket W_n \rrbracket) \leq \frac{B(\lambda)}{2^n} = \frac{1}{2^n}.$$

So $\mu(\llbracket W_n \rrbracket) \leq 2^{-n}$. Hence W is a ML-test.

Suppose $A \in 2^\omega$ is ML-random. Then $A \notin \bigcap_n W_n$. Therefore $A \notin \text{Succ}(B)$. Since B could be any c.e. (super)martingale, A isn't in the success set of any c.e. (super)martingale.

(\Leftarrow) Let $U \subset \mathbb{N} \times 2^{<\omega}$ be a ML-test with $U_n = \{\sigma : (n, \sigma) \in U\}$. We will assume that each U_n is prefix-free. Define B_n as follows.

Whenever σ enters U_n add 1 to the value of $B_n(\tau)$ if $\tau \succeq \sigma$, and add $2^{|\tau|-|\sigma|}$ if $\sigma \succ \tau$. So

$$B_n(\tau) = \sum_{\substack{\sigma \in U_n \\ \tau \succeq \sigma}} 1 + \sum_{\substack{\sigma \in U_n \\ \sigma \succ \tau}} 2^{|\tau|-|\sigma|}.$$

Do note that only one of these sums will ever happens and if the left sum happens, then it only happens once since we are assuming each U_n is prefix-free. So a better way to define B_n would be as a piece-wise function.

$$B_n(\tau) = \begin{cases} 1 & \exists \sigma \in U_n (\tau \succeq \sigma) \\ \sum_{\substack{\sigma \in U_n \\ \sigma \succ \tau}} 2^{|\tau|-|\sigma|} & \text{otherwise} \end{cases}$$

Claim: B_n is a martingale

Proof of claim: Let B_n be defined as above.

Case 1: Suppose there is a $\sigma \in U_n$ such that τ extends σ . Then $B_n(\tau) = 1$. Note that $\tau 0$ and $\tau 1$ also extend σ . So $B_n(\tau 0) = 1$ and $B_n(\tau 1) = 1$. Therefore by definition of martingales

$$B_n(\tau 0) + B_n(\tau 1) = 1 + 1 = 2 = 2 * 1 = 2B_n(\tau).$$

Case 2: Suppose there is a $\sigma \in U_n$ such that $\sigma \succ \tau$. Then

$$B_n(\tau) = \sum_{\substack{\sigma \in U_n \\ \sigma \succ \tau}} 2^{|\tau|-|\sigma|}.$$

Since σ is a proper extension of τ , σ has the potential to be $\tau 0$ or $\tau 1$. So we can separate the above summation as follows.

$$\sum_{\substack{\sigma \in U_n \\ \sigma \succ \tau}} 2^{|\tau|-|\sigma|} = \sum_{\substack{\sigma \in U_n \\ \sigma \succ \tau 0 \\ \text{or } \sigma \succ \tau 1}} 2^{|\tau|-|\sigma|} + \sum_{\substack{\tau 0 \in U_n \\ \tau 1 \in U_n}} 2^{-1}.$$

Note that we have 2^{-1} for the right summation since $-1 = |\tau| - |\tau 0|$ and $-1 = |\tau| - |\tau 1|$ depending on whichever may be in U_n .

Next observe that

$$B_n(\tau 0) = \sum_{\substack{\sigma \in U_n \\ \sigma \succ \tau 0}} 2^{1+|\tau|-|\sigma|} + \sum_{\tau 0 \in U_n} 1$$

and

$$B_n(\tau 1) = \sum_{\substack{\sigma \in U_n \\ \sigma \succ \tau 1}} 2^{1+|\tau|-|\sigma|} + \sum_{\tau 1 \in U_n} 1$$

by the way we defined B_n above. Therefore

$$\begin{aligned} B_n(\tau 0) + B_n(\tau 1) &= \sum_{\substack{\sigma \in U_n \\ \sigma \succ \tau 0 \\ \text{or } \sigma \succ \tau 1}} 2^{1+|\tau|-|\sigma|} + \sum_{\substack{\tau 0 \in U_n \\ \text{or } \tau 1 \in U_n}} 1 \\ &= 2 \sum_{\substack{\sigma \in U_n \\ \sigma \succ \tau 0 \\ \text{or } \sigma \succ \tau 1}} 2^{|\tau|-|\sigma|} + 2 \sum_{\substack{\tau 0 \in U_n \\ \text{or } \tau 1 \in U_n}} 2^{-1} \\ &= 2B_n(\tau). \end{aligned}$$

Thus B_n is a martingale.

We have not proved this result but the function $B = \sum_{n \in \mathbb{N}} B_n$ is also a martingale. It is also easy to see that the B_n are c.e. in the sense of definition 3.6.1 since they are approximated from below. From this we can also see that B is a c.e. martingale as well. So we have that B is a c.e. martingale.

Suppose $A \in 2^\omega$ isn't ML-random. Then $A \in \bigcap_n \llbracket U_n \rrbracket$. So for every k , there is an m such that U_k contains $A \upharpoonright m$. So for the martingale B built up from said test we have that $B_k(A \upharpoonright m) = 1$. Since for every n , $\llbracket U_{n+1} \rrbracket \subset \llbracket U_n \rrbracket$, we have that for any $n < k$, $\llbracket U_k \rrbracket \subset \llbracket U_n \rrbracket$. Therefore there is a $\sigma \in U_n$ such that $A \upharpoonright m \succeq \sigma$. Thus $B_n(A \upharpoonright m) = 1$. So we have that

$$\sum_{n=1}^k B_n(A \upharpoonright m) = k.$$

Therefore $B(A \upharpoonright m) \geq k$ and hence $\limsup_m B(A \upharpoonright m) = \infty$.

7 Summary

What we have just shown is that the three most accepted understandings of randomness are all logically equivalent. So we have the following theorem.

Theorem 7.1.1: Let $A \in 2^\omega$. The following statements are equivalent:

- i) A is 1-random,
- ii) A is ML-random,
- iii) No c.e. (super)martingale succeeds on A .

There are many other definitions and interpretations of randomness, but none can be comparative like these three. However there is one drawback to defining randomness. Recall back to Chaitin's Ω . Intuitively it doesn't seem random since there is a pretty nice way to describe it. Yet it satisfies these definitions of randomness. Well as it turns out the more we try to define and parameterize randomness, the further we get from true randomness.

8 Bibliography

- Weber, Rebecca. *Computability Theory*. American Mathematical Society, 2012.
- Downey, R.G. & Hirschfeldt, D.R. *Algorithmic Randomness and Complexity*. Springer, 2010.
- Nies, André. *Computability and Randomness*. Oxford University Press. 2009.
- Brodkorb, Laurel. *The Entscheidungsproblem and Alan Turing*. gcsu.edu. 2019.
- Downey, R.G. & Reimann, Jan. *Algorithmic Randomness*. scholarpedia.org. 2007.