2019

# K-Triviality and Computable Measures

William Joseph Franczak
*Lehigh University*, wjf212@lehigh.edu

# $K$-Triviality and Computable Measures

by

William Franczak

A Dissertation
Presented to the Graduate Committee
of Lehigh University
in Candidacy for the Degree of
Doctor of Philosophy
in
Mathematics

Lehigh University
January 2019

Approved and recommended for acceptance as a dissertation in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

William Franczak
$K$-triviality and Computable Measures

---

**Date**

---

**Dr. Lee Stanley**, Dissertation Director, Chair

---

**Accepted Date**

Committee Members

---

**Dr. Vincent Coll**

---

**Dr. Daniel Conus**

---

**Dr. Cynthia Curtis**

---

**Dr. Garth Isaak**

# Contents

# List of Figures

## Abstract

Algorithmic randomness is primarily concerned with quantifying the degree of randomness of infinite binary strings, and is usually carried out in the setting of Cantor space. One characterization of randomness involves prefixes "being as hard as possible to describe". Also of interest are the infinite binary strings whose prefixes are as easy as possible to describe i.e., the $K$-trivial strings. We will study these strings in the setting of computable metric spaces, and investigate several definitions which attempt to correctly generalize $K$-triviality. We describe some of the difficulties inherent in a natural-seeming approach, and offer partial results where new definitions relate to a more established definition of $K$-triviality under the right conditions.

# Chapter 1

# Introduction

This dissertation investigates topics in *algorithmic randomness* related to the notion of $K$-triviality, in settings more general than the standard setting of Cantor space, equipped with Lebesgue measure. Initially, the theory of algorithmic randomness was developed in this standard setting. Starting in roughly 2005-6, the focus of attention started to shift to the more general ones, and this has become the predominant line of development. We present a more detailed overview of the period prior to 2005-6 in Section 1.2, and present its main results in Section 2.3. In Chapter 3, we do the same for the later period, concentrating on presenting those notions and results most directly relevant for our work.

From its very inception, algorithmic randomness has featured (and attempted to elucidate) the dichotomy between objects (mainly infinite binary strings) that should be regarded as random and those that should be regarded as far from random. The $K$-triviality notion (see Definition 2.3.26, for its formulation in the standard setting) is a precise mathematical characterization of the latter half of that dichotomy. However, in order to provide context and intuition, we will also discuss the "randomness side", but more briefly, since none of our own work directly involves those notions.

The interplay between another pair of notions looms large in developments since 2005, though here the notions are not dichotomous. Instead, they represent two

different ways of "coming to grips with the same underlying reality." The first of these notions is that of a *computable metric space*, viz. Definition 3.1.1. The second is that of *computable measure*, viz. Definitions 3.2.5, 3.2.13. In full generality (Definition 3.2.5), the notion of computable measure depends on a "background" computable metric space, but for an important family of such "background" spaces, the details of the metric structure are not critical, and only the metric topology really matters: this is the context of Definition 3.2.13. More detailed discussion of the issues involved is provided in Section 3.2 and to a lesser extent in the last three paragraphs of Section 2.1, where we establish the needed notation and look ahead to the fuller discussion.

The point of view of computable measures provided the natural approach to generalizing randomness notions, and indeed the ideas developed in Section 3.2 were developed for this very purpose. Somewhat later, Melnikov and Nies, [15] and [14], viz. Definition 3.1.4, put forward a generalization of $K$-triviality in terms of the computable metric space notions. Later still, J. Rute, [21], viz. Definition 3.2.22, proposed another generalization, framed in terms of computable measures.

In the initial stages of work towards this dissertation, we focused on understanding the similarities and differences between these two proposed generalizations of $K$-triviality notion. For reasons discussed in connection with Theorem 3.1.5, below, below, a consensus developed that the Melnikov-Nies generalization of $K$-triviality was successful. Nevertheless, one possible critique of their generalization is that non-specialists in computability theory might be dissuaded from working directly with computable metric spaces because of the technicalities of some its specialized apparatus (special points, Cauchy names) and would therefore welcome a more familiar setting, such as that of computable measures, closer to a purely measure theoretic one.

Rute posed but did not investigate the question of whether his proposed generalization of the $K$-triviality notion was equivalent to the one proposed by Melnikov

and Nies. Rute was very likely motivated in part by the successful generalization of the notion of Martin-Löf randomness for general computable measures, and in particular, by the attractive simplicity of its characterization given by a result of Hoyrup and Rojas, [9] (see Proposition 3.2.21, below). This is in full analogy with the characterization of $K$-triviality in the standard setting provided by Schnorr's Theorem, Proposition 2.3.25, and involves the very natural looking move of replacing the length of a finite bitstring by the negative (base 2) logarithm of the measure of the corresponding basic open subset of Cantor space. Note that when the measure is Lebesgue measure, this negative logarithm is exactly the length. This is exactly the approach taken by Rute in [21]. We do not know whether Rute was also motivated by the possible critique of the Melnikov-Nies generalization noted at the end of the previous paragraph, but that critique would certainly be mooted if Definition 3.2.22 turned out to be equivalent to Definition 3.1.4. Unfortunately (or perhaps *fortunately*, for the development of this dissertation) things did not turn out that way.

One of our earliest results, Proposition 4.3.1, showed this not to be the case. Shortly thereafter, we formulated two "improved" versions of Rute's proposed definition, Definitions 4.3.2, 4.3.3 and noted that they "block" the particular form of pathlogical behavior of Proposition 4.3.1. We considered the possibility that either or both of these improvements might be equivalent to the Melnikov-Nies definition, but we were unable to prove this; indeed this remains an open question. From that point on, it became one of the main objectives of our work to formulate a generalization of $K$-triviality in the language of computable measures which is (as close as possible to being) purely measure-theoretic and prove its equivalence to Definition 3.1.4. Our pursuit of that objective has been only partially successful, and is embodied in Definitions 5.2.1 and 5.2.4, in Proposition 5.2.5, and in Theorem 5.2.3. It only became clear to us much later that it should not have come as a surprise that we were only partially successful. As we shall discuss, in what follows, in the light of some of our other results, and the understanding we have gained along the way, the goal we set for ourselves was extremely difficult to achieve.

Returning to the chronological development of our results, we took a step in the direction of our main objective in proving Theorem 5.1.1; there, for a restricted class of "tame" computable measures, we were able to show that any infinite binary string that satisfies Definition 3.1.4 must also satisfy Definition 3.2.22. The "tameness" involved is formulated in terms of the notions of the *granularity* of a computable measure (introduced in [8]), and the dual notion of its *coarseness* which we introduce in Section 4.1.

Several other of our principal results also involve the notion of coarseness. In Theorem 4.1.5 we construct (in analogy to a result of [8] dealing with granularity) a computable measure whose coarseness function is not computable. Our remaining result involving coarseness shifts the focus somewhat from the issue of generalizations of $K$-triviality itself to the issue of generalizations of the so-called *Machine Existence Theorem* (MET in what follows) Theorem 2.2.19, [18].

The MET is one of the principal tools for proving results about $K$-triviality in the standard setting, and is an "effective version" of an earlier result, Kraft's Theorem (KFT in what follows), Theorem 2.2.17, [18]. In Theorem 4.1.6, we show that for computable measures which are "tame" *in terms of their coarseness alone, without reference to their granularity*, the most direct (and strongest) generalization of the MET holds (and therefore so does its consequence, the most direct (and strongest) generalization of KFT). A counterpoint to this result is Theorem 4.2.1 in which we construct a computable measure for which a weaker (but arguably more natural) generalization of KFT *fails*.

Though we defer further discussion of these last two results until Chapter 4 and Chapter 6, we do regard them as indications of just how rich and complex is the setting of computable measures. This is especially true if, unlike the computable measures discussed so far which are all *continuous*, viz. Definition 3.2.12, we also consider computable measures which have atoms (or "point-masses"). The recent

[19] undertakes a systematic study of such measures and their pathological properties. Reinforcing the picture that emerges from [19], in Proposition 4.3.4, we construct a computable measure with atoms for which all infinite binary strings which take on the value 0 only finitely many times satisfy Definition 3.2.22 (as they "should"), while all those which take on the value 1 only finitely often are atoms and "appear random" to this measure. In a suitable sense made precise in Section 4.3, all other infinite binary strings are "far from being random".

Thus, the big picture that emerges from our results of Chapter 4 is that there is a wide range of computable measures associated with any computable metric space. The picture becomes even more complex if one takes into account an additional technical notion that figures in Section 3.2 and in Definitions 3.2.22, 5.2.1, and 5.2.4: that of a *generator for a computable measure.*

Despite its importance, very little is known about the question of which computable measures have generators with "nice properties" and this question appears to be quite difficult and murky. On the other hand, if a generalization of $K$-triviality framed in terms of computable measures (and generators) were to be fully equivalent to Definition 3.1.4, then for each computable metric space, the definition in terms of measures and generators would have to be invariant across the whole range of computable measures (and generators) associated with the computable metric space. This seems quite difficult to achieve even for specific, well-understood computable metric spaces, let alone in full generality.

## 1.1 Organization and Summary

Chapter 2 is devoted to preliminaries: we establish our notation and other conventions, recall the basic notions from computability theory to which we will appeal, and develop some of the main results about algorithmic randomness in the standard setting. In Chapter 3, we give an overview of the frameworks of computable

metric spaces (including the Melnikov-Nies generalization of $K$-triviality) and computable measures (including Rute's proposed generalizaion of $K$-triviality in this framework).

Our results appear in Chapters 4 and 5, while Chapter 6 is devoted to concluding remarks, including discussion of open problems and future directions. In particular:

1. in Section 4.1, we introduce our notion of coarseness and in Subsection 4.1.1 we prove Theorem 4.1.5,
2. in Subsection 4.1.2, we prove Theorems 4.1.6,
3. in Section 4.2, we prove Theorem 4.2.1,
4. in Section 4.3, we prove Propositions 4.3.1 and 4.3.4,
5. in Section 5.1, we prove Theorem 5.1.1
6. in Section 5.2, we prove Theorem 5.2.3 and Proposition 5.2.5.

Starting with Chapter 2, each Chapter begins with a Chapter overview, laying out its organization as well as which other results are proved and where. Results from the literature are given with attribution, but most often without proof, mainly in Chapters 2 and 3. Occasionally, a proof is given for a result from the literature, when the proof is accessible, and sheds light on issues to be dealt with later. Results given without attribution are our results, and where our proof is modeled on the proof of an analogous result from the literature this is noted.

## 1.2 Historical Development of Algorithmic Randomness: Highlights

We give a short survey of impotant stages in the historical development of algorithmic randomness in the standard setting of Cantor space equipped with Lebesgue

measure, focusing on the period of the early/mid 1960's through 2005-6. The appearance of one pair of papers ushers in this period, while the appearance of a second pair of papers marks its close, as well as a shift in the main focus of activity from the standard setting to the other settings briefly mentioned above.

The first two papers are [11] and [13], by Kolmogorov and Martin-Löf, respectively, and the second two papers are [17] and [5], by Nies and Gács, respectively. Martin-Löf's paper opened the way to a systematic mathematical approach to the notion of randomness. This was also one of Kolmogorov's objectives, but his paper also introduced the notions of descriptive complexity of finite binary strings, and led most directly to the notion of $K$-triviality, a property formulated to characterize (via the descriptive complexity of their finite initial segments or *prefixes*) those elements of Cantor space which are as far from random as possible. The interplay between descriptive complexity and notions of randomness is one of the main threads of development throughout the entire period, and Nies' [17] is the culmination of this thread. The nearly simultaneous appearance of Gács' [5] was met with intense interest and triggered a flurry of activity (that persists to the present) seeking to extend the reach of the notions and methods developed to settings other than the standard one.

The survey that follows is culled from the more detailed treatments found in [1], [27] and especially in [4], [18]. The last two of these references are excellent "snapshots" of the state of the art at the close of our period on which we draw heavily in our account of some of the main notions and results in Section 2.3, below.

Algorithmic randomness approaches the problem of how to quantify and formalize the concept of randomness in terms of computability theory. The main question is: given an element of Cantor space, how do we determine if it exhibits the properties one would expect of a random element of Cantor space? It should certainly satisfy, for example, the strong law of large numbers, and other well known laws of probability. Thus, the strong law of large numbers can be viewed as a test - if an element

of Cantor space does not satisfy the strong law of large numbers, it should not be considered to be random. However, there are certainly elements of Cantor space, e.g., 010101 . . ., that should not be considered random even though they satisfy the strong law of large numbers. A natural approach would be to require elements of Cantor space to "pass" additional tests: in this example, a test that requires there to be long runs of 0's (and of 1's). The question then arises: how many tests and which ones should an element of Cantor space be required to "pass" in order to be considered random. An appealing idea is to impose any test that can be expressed in a sufficiently effective way (and is "failed" only on a null set). But how should we make precise the notion of "sufficiently effective"? There are many reasonable potential answers.

In 1966, Per Martin-Löf made the first successful *published* attempt at answering this question in [13], but it should be noted that similar ideas underlie the notion of random-real forcing used in Solovay's celebrated paper [25]. Despite its publication date of 1970, it is well-known that its main theorems were proved considerably earlier, possibly even before 1966. Martin-Löf formalized the idea of a test as a computably enumerable (c.e., in what follows) sequence of nested (effectively) open sets whose measures converge 0, with rate of convergence specified to be $2^{-n}$.

Elements of Cantor space lying in this intersection fail the test; those which fail no such test are *Martin-Löf random*. A rich hierarchy of different notions of randomness arose, mainly in reaction to Schnorr's 1971 paper [22] which offered dual criticisms of Martin-Löf's definition. On one hand, Schnorr suggested that c.e. but not outright computable tests were too powerful. This led him to his notion of Schnorr Randomness. On the other hand, he suggested that Martin-Löf random elements of Cantor space could actually still have some properties which seemed not-so-random, e.g., for some, the corresponding real number will have a property known as being *left c.e.* (see Definition 2.2.9). Many of the more prominent proposed definitions of randomness are surveyed in [26]. It is worth noting that every notion considered there (other than Martin-Löf randomness itself, of course) is either strictly stronger or

strictly weaker than Martin-Löf randomness. A notable exception to this pattern is the centrally important notion of Kolmogorov-Loveland randomness. An element, $x$, of Cantor space is said to be partially computably random if no computable betting strategy succeeds when bidding sequentially on the values of $x$. Then $x$ is said to be Kolmogorov-Loveland random if no computable betting strategy succeeds even if the order in which the values are bid on is not fixed. Kolmogorov-Loveland randomness is known to be no stronger than Martin-Löf randomness, but whether it is the same is unknown, and this is in fact one of the main open questions in algorithmic randomness today.

As noted in [4], most approaches to randomness can be put into one of three categories:

1. Computability-Theoretic - using computability notions, such as complexity of initial segments of an element of Cantor space, to determine whether or not it is random.

2. Measure Theoretic - as in Martin-Löf's approach, wherein he attempts to capture nonrandom strings in "effective" sets of small measure

3. Based on Unpredictability - as Kolmogorov-Loveland randomness, where randomness is characterized in terms of the failure of "effective" betting strategies, such as martingales.

In addition to studying concepts of randomness, much attention has also been devoted to attempts to characterize those elements of Cantor space that should be considered to be highly non-random. The approaches to such questions mostly fall under the heading of Computability-Theoretic, and Kolmogorov appealed to computability theoretic notions in his development of descriptive complexity, in [11], as a way of measuring how much information is inherent in a finite binary string.

In his 1972 paper [23], Schnorr provided a characterization of Martin-Löf randomness in terms of Kolmogorov complexity, showing that an element of Cantor space

is Martin-Löf random iff the Kolmogorov complexity of its prefixes grows asymptotically as fast as possible. Of course, having such a nice characterization for elements of Cantor space whose prefix-complexity function grows as quickly as possible, it was natural to also investigate the class of those whose prefix-complexity function grows as slowly as possible, a property which would eventually be called $K$-triviality. The $K$-trivial elements of Cantor space are at the opposite end of the spectrum from the Martin-Löf random strings - they exhibit patterns and are "easy to describe".

One of the first important results about this class of strings was due to Solovay in 1975 (unpublished notes, see [4]), who showed that there are non-computable $K$-trivial elements of Cantor space, thereby significantly increasing the degree of interest in studying the the class of $K$-trivials. Results showing that the $K$-trivials are not so far from being computable followed quickly. In 1977, Chaitin showed that all $K$-trivials were also $\Delta_2^0$ [3]. In 1990, Zambella [28] would adapt Solovay's proof to give a requirement free solution to Post's problem.

At the same time, Zambella began studying the notion of an element, $x$ of Cantor space being *low for randomness*, meaning that if $x$ is allowed as an oracle in all definitions of effectiveness (cf. the final paragraph of Subsection 2.2.1, all random elements of Cantor space remain random *relative to* $x$. Muchnik (unpublished, see [4]) proposed a similar idea for Kolmogorov complexity. In 1999, Kucera and Terwijn showed in [12] that each string low for Kolmogorov Complexity was also low for Martin-Löf randomness. In 2006, Nies showed, [17], that being low for Kolmogorov complexity was equivalent to being $K$-trivial and also equivalent to being low for a different notion of randomness, known as 1-randomness, as well as showing that being low for Kolmogorov complexity was equivalent to being $K$-trivial.

Thus, the notion of $K$-triviality has emerged as being closely linked to various notions of randomness and its investigation has given rise to useful techniques such as Solovay functions/cost functions, and the so-called "decanter method". In our

work towards a suitable generalization of $K$-triviality in the framework of computable measures, we were naturally led to supplement the "classical" computability-theoretic methods with more measure-theoretic ones.

# Chapter 2

# Preliminaries

## Chapter Overview

In Section 2.1, we set out our notation and establish other conventions and terminology that will be used in the rest of the disseration. In Section 2.2, we survey some basic and standard material from computability as well as some more specialized topics, e.g. KFT and the MET in subsection 2.2.2. Finally, in Section 2.3, we present the key notions and results about algorithmic randomness in the "standard setting" of Cantor space equipped with Lebesgue measure, including plain and prefix-free complexity (Definitions 2.3.4 and 2.3.14), Martin-Löf randomness (Definition 2.3.2), and $K$-triviality (Definition 2.3.26).

All results are from the literature and most are given without proof. We mainly follow the references [4] and [18], which, as mentioned in the discussion at the end of Section 1.2, constitute an excellent "snapshot", circa 2009-2010, of the state of the art in the standard setting, just as the main focus of development was starting to shift to generalizations to other frameworks.

## 2.1 Notation, Conventions, and Terminology

For the most part, our notation and terminology are standard or intended to be so. What follows is designed to cover all possible exceptions. The next table summarizes some of our most important notation. The paragraphs that follow complete the table.

| | |
|---|---|
| $\Sigma^*$ | The set of all finite binary strings |
| $2^{\mathbb{N}}$ | The set of all infinite binary strings |
| $\sigma\tau$ | concatenation of $\sigma$ and $\tau$ |
| $\sigma^n$ | $\sigma$ concatenated with itself $n$ times |
| $XY$ | $\{\sigma\tau \vert \sigma \in X, \tau \in Y\}$ |
| $\sigma \preceq \tau$ | $\sigma$ is a prefix of $\tau$ |
| $[\sigma]$ | $\{x \in 2^{\mathbb{N}} \vert \, \sigma \preceq x\}$ |
| $\vert\sigma\vert$ | The length of $\sigma$ |
| $\sigma \upharpoonright n$ | The first $n$ bits of the string $\sigma$ |
| $\sigma(n)$ | The $n^{th}$ bit of $\sigma$ |
| $\epsilon$ | The empty string |
| $\#X$ | The cardinality of the set $X$ |
| $x.f(x)$ | notation for the function $f$ |
| $[n]$ | $\{0, \cdots, n-1\}$ |
| $\phi$ | The empty set |
| $\delta A$ | The boundary of $A$ |
| $B(a,r)$ | $\{x \vert d(a,x) < r\}$ |
| $\overline{B}(a,r)$ | $\{x \vert d(a,x) \leq r\}$ |

**Figure 2.1:** Basic Notation

When carrying out recursive definitions/constructions, we adopt the standard practice of referring to the *stages* of the definition/construction and prior to carrying out

the recursion step (say at stage $i$), we will often explicitly list as "*recursion hypotheses*" the properties assumed for the objects defined/constructed at previous stages. Then, in defining/constructing the needed object or objects at stage $i$, we will do so in such a way that these recursion hypotheses are preserved. The argument for this then completes the recursive construction.

Lower case letters from the middle of the Latin alphabet $(i, j, \ldots, n)$ will typically denote natural numbers, i.e. elements of $\mathbb{N}$. As usual, a function, $f$, is said to be $1:1$ if and only if $f$ is an injection, and said to be $m:1$ if and only if whenever $f(x_0) = \ldots = f(x_m)$, there are natural numbers $i < j \leq m$ such that $x_i \neq x_j$. If each of $f,\ g$ is a total function from $\mathbb{N}$ to $\mathbb{R}$, we will write $f \leq^+ g$ to mean that there is $b$ such that for all $n,\ f(n) \leq g(n) + b$, while writing $f =^+ g$ will mean that both $f \leq^+ g$ and $g \leq^+ f$ are true. For sets, $A, B$, we write $A \subset B$ to mean that $A$ is a proper subset of $B$, whereas, as usual, $A \subseteq B$ allows $A = B$.

We fix a standard (computable) pairing function, i.e., a computable bijection from $\mathbb{N} \times \mathbb{N}$ to $\mathbb{N}$, which we denote by $<,>$. One such computable bijection is given by $< m, n >= \frac{(m+n)(m+n+1)+n}{2}$. The corresponding coordinate inverses, or projection functions, $\pi_1$, $\pi_2$, are those (computable) functions satisfying $\pi_1(< m, n >) = m$ and $\pi_2(< m, n >) = n$. By nesting a pairing function, for each $k > 2$ we obtain a (computable) bijection from $\mathbb{N}^k$ to $\mathbb{N}$. We will abuse notation by using $<>$ to denote all such bijections. Most often we mean the 2 place pairing function, and when we do not, it is clear from context.

We will let $\mathbb{Q}$ denote the rational numbers as usual, and log will always mean logarithm in base 2. We fix a standard enumeration without repetitions (where the denominators are non-decreasing and relatively prime to the numerators), $\{q_i\}_{i \in \mathbb{N}}$ of the positive rationals and then encode a positive rational as a natural number via its index. Much of the rest of our notational conventions will involve finite binary strings (i.e., elements of $\Sigma^*$) or infinite ones (i.e. elements of $2^{\mathbb{N}}$). We lay out these conventions in the next few paragraphs.

Typically, we will use: $\sigma$, $\tau$ and $\omega$ to denote finite binary strings, while $x$ and $y$ will be used to denote infinite binary strings. For finite strings, $\sigma \in \Sigma^*$, $[\sigma]$ is often called the $\sigma$-cell, or the cylinder of or about $\sigma$; it is a basic open subset of Cantor space, but under a dif and only iferent topology on $2^{\mathbb{N}}$, its status might be problematical. However, when considering measures on Cantor space, or some other other separable metric space with ambient set $2^{\mathbb{N}}$ , we will often blur the distinction between $\sigma$ and $[\sigma]$, writing $\mu(\sigma)$ or occasionally even $\mu\sigma$ when $\mu([\sigma])$ is intended.

All measures considered will be probability measures on the $\sigma$-algebra of Borel sets (for the metric topology of the separable metric space under consideration), unless otherwise noted.

We fix $a : \mathbb{N} \to \Sigma^*$ to be the bijection with the property that if $i < j$, then $a(i)$ precedes $a(j)$ in lexicographic order. This allows us to identify finite binary strings with natural numbers. We may also conflate sets, $A$, of natural numbers with infinite binary strings by identifying $A$ with its characteristic (or indicator) function. Note that this gives us two different views of $A \subseteq \mathbb{N}$: either as a set of finite binary strings, or as an infinite binary string. We shall attempt to make it clear from context which one is intended, any time the issue might arise.

In a similar vein, when discussing descriptive complexity (in Section 2.3) we will sometimes use a natural number, $n$, in place of the formally correct $a(n)$. Thus, for example, in discussing prefix-free (or Kolmogorov) complexity (viz. Definition 2.3.17), we will sometimes write $K(n)$ rather than $K(a(n))$, and similarly for plain descriptive complexity and $C$.

On a number of occasions in Chapter 1 we have spoken (somewhat loosely, but always accurately) of Cantor space. Our usage, so far, has been accurate either because we were really only referring to its ambient set, $2^{\mathbb{N}}$, or because we carefully

specified something like 'equipped with Lebesgue measure', or because this was implicit in the fact that we were speaking about developments *in the standard setting of Cantor space equipped with Lebesgue measure*. Going forward, we will have to be more careful, and the following notation and terminology is designed to allow us to do so without being too cumbersome. As we have just mentioned and set out in our table, $2^{\mathbb{N}}$ is the set of infinite binary strings, and is the ambient set of Cantor space.

Going forward, in what will become the formalism of Definition 3.1.1, when we get there, when we refer to Cantor space, we will mean the following metric space with ambient set $2^{\mathbb{N}}$, together with the enumeration, in lexicographic order, of the elements of $2^{\mathbb{N}}$ which take on the value 1 only finitely often. For distinct $x, y \in 2^{\mathbb{N}}$, we take the distance between $x$ and $y$ to be $2^{-i(x,y)}$, where $i(x, y)$ is their least difference coordinate, i.e., the least $i$ such that $x(i)$ (the $i^{\text{th}}$ bit of $x$) differs from $y(i)$. But we do not necessarily equip this metric space with Lebesgue measure. We will use $\mathfrak{C}$ to denote Cantor space.

In fact, we could replace this metric by any equivalent one, where the equivalence resides in the fact that the metric topologies coincide. This leads us to the notion of *topological Cantor space*, for which we shall use the notation $\mathfrak{C}_T$. By this we will usually mean *any* computable metric space (in the sense of Definition 3.1.1 whose metric is equivalent to the metric we have just specified for $\mathfrak{C}$, with any *reasonable* enumeration of any *reasonable* choice of a countable dense subset (the precise sense of both instances of 'reasonable' being supplied by Definition 3.1.1. Sometimes, however, we mean the family of all such computable metric spaces. Finally, looking ahead to Section 5.2, it will be important to note that (in the second sense of our usage) $\mathfrak{C}_T$ is a much more restrictive family of computable metric spaces than the family of all computable metric spaces whose ambient set happens to be $2^{\mathbb{N}}$.

## 2.2 Computability

### 2.2.1 Basic Concepts

We survey the basic notions and results from computability theory needed in what follows. For the most part, we follow the treatments of [18] and [4].

Any of the various equivalent well-known formal models of computation would provide an adequate setting for what follows, and for the most part we will not be explicit about the technicalities inherent in any such model except to say that the models best adapted to our purposes all involve some idealized computing device (e.g. one of the many developments of the Turing Machine model or the Register Machine model) and *programs* for this device. The idealized computing device has a countably infinite memory/workspace. Programs consist of a finite sequence of instructions in a simple effectively presented countable language. On any initial input (or inputs), instructions are executed sequentially, most often resulting in a simple (deterministic) update to memory, but possibly, in the case of a "control instruction", changing the internal state of the computing device (Turing Machine model) or specifying the label of the next instruction to be executed (Register Machine model). We shall follow [18] and [4] in blurring the hardware/software distinction by referring to each combination of computing device and program as its own "*machine*".

One important instruction (Turing Machine model) is the HALT instruction. In the Register Machine model this is "simulated" by a control instruction specifying a value for the label of the next instruction to be executed that is larger than the label of any program instruction. If the HALT instruction is reached, the computation terminates, and an output value is specified according to the current state of memory and the I/O conventions of the model. Of course, there are combinations of programs and inputs which lead to non-terminating computations: the HALT instruction is never reached.

For each positive integer $n$, each program, $P$, determines an $n$-place partial function, $\varphi_P^{(n)}$, whose domain consists of those ordered $n$-tuples of inputs, $(i_1, \ldots i_n)$ for which the HALT instruction is reached; for such $(i_1, \ldots i_n)$ , $\varphi_P^{(n)}(i_1, \ldots i_n)$ is defined to be output value of the terminating computation.

The usual models typically take inputs and outputs to be natural numbers, but via the effective bijection $a$ of Section 2.1, they can be taken to be finite bitstrings. Accordingly, we shall often view the $\varphi_P^{(n)}$ as partial functions from $\mathbb{N}^n$ to $\Sigma^*$, or from $(\Sigma^*)^n$ to $\Sigma^*$, etc..

**Definition 2.2.1.** A partial function $\varphi : \mathbb{N}^n \to \mathbb{N}$ (or $(\Sigma^*)^n \to \Sigma^*$) is called *computable* if there is some program $P$ such that $\varphi = \varphi_P^{(n)}$. A set $A \subseteq \mathbb{N}^n$ (or $(\Sigma^*)^n$ is called *computable* (aka decidable, computably decidable, effective) if its characteristic function $\chi_A$ is computable and it is called *computably enumerable* or *c.e.* (aka semi-decidable, computably semi-decidable, etc) if it is the domain of some $n$-place computable function.

There are various ways of extending the notions of computability to other domains, and some of these will be discussed in Chapter 3, but for now we note that the encoding provided by our standard enumeration of $\mathbb{Q}$ gives us a natural and easy extension of computability notions to the setting of (partial) functions from $\mathbb{N} \times \mathbb{N}$ to $\mathbb{Q}$. We omit the standard (and easy) details, but note that this will enable us to introduce, here, a literature notion that figures in Definition 3.2.5, below. In the literature, the definition is typically given for partial functions, but we shall only need it in the case of total functions. Further, in the literature, typically the domain of the function is (a subset of) $\mathbb{Q}$ rather than (a subset of) $\mathbb{N}$.

**Definition 2.2.2.** A (total) function $f : \mathbb{N} \to \mathbb{R}$ is said to be *lower semicomputable* if there is a (total) computable function $\tilde{f} : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ such that for all $n \in \mathbb{N}$, the

sequence $\{\tilde{f}(k,n)\}_{k\in\mathbb{N}}$ is monotone non-decreasing and :

$$\lim_{k\to\infty} \tilde{f}(k,n) = f(n).$$

Similar notions are implicit in Definition 3.2.13 and Propositions 2.3.18, 3.2.15 and 4.1.3. In the case of Propositions 2.3.18, and 3.2.15, the similarity is really to the dual notion of upper semicomputable.

It is well known and easy to see that finite sets are computable, that computable sets are c.e., that a set is computable if and only if both it and its complement are c.e., and that every non-empty c.e. subset of $\mathbb{N}$ (or $\Sigma^*$) is also the range of some total computable function (which was the initial definition and the motivation for the terminology). There is, however, an important difference between computable sets and c.e. but non-computable sets in this regard. An infinite subset of $\mathbb{N}$ (or $\Sigma^*$) is computable if and only if it is the range of a *monotone* increasing total computable function, i.e., if and only if it can be computably enumerated in increasing order (in the case of subsets of $\Sigma^*$, this means in increasing order for lexicographic order). The idea is that in an enumeration in increasing order, once a potential member of the set has "been skipped", we know that it is not in the set. It is worth noting that (necessarily infinite) c.e. but not computable sets can nevertheless be computably enumerated without repetitions.

Since there are only countably many programs, these can be enumerated as $(P_e|e \in \mathbb{N})$. Indeed there are effective such enumerations, but even more will be true, as we shall soon see. Each such enumeration induces, for each positive integer $n$, an enumeration of the $n$-place computable functions, $\left(\varphi_e^{(n)}|e \in \mathbb{N}\right)$ by taking $\varphi_e^{(n)}$ to be $\varphi_{P_e}^{(n)}$. In such an enumeration, $e$ is called an *index of* $\varphi_e^{(n)}$. Each computable function has infinitely many indices. This is easily seen in an device/program model, since any program can be "padded" by adding useless (but harmlesss) instructions, but in fact, this is an essential feature of any effective enumeration of computable functions

in any model of computation, and not merely a quirk of the device/program models. When no positive integer $n$ is specified, it is understood that $n = 1$: we are working with one-place functions.

Each such enumeration also gives rise to a family of functions that are *universal* for computable functions: for each positive integer $n$, we have the $n+1$-place partial function $U^{(n)}$ whose domain consists of those $(e, m_1, \ldots, m_n)$ for which $(m_1, \ldots, m_n)$ is in the domain of $\varphi_{P_e}^{(n)}$, and when this occurs, $U^{(n)}(e, m_1, \ldots, m_n) = \varphi_{P_e}^{(n)}(m_1, \ldots, m_n)$. Of course, these universal functions depend strongly on the enumeration. In one of the cornerstone results of the early development of the subject, Turing showed that Gödel's approach to producing an enumeration of programs yields computable universal functions. This is the content of the next Proposition and Definition.

**Proposition 2.2.3.** There is an enumeration of programs which is strongly effective in the sense that each of its universal functions is computable. □

In device/program models, universal functions are computed by universal programs: the universal program first decodes $P_e$ from $e$ and then simulates its operation on the remaining inputs. This strongly exploits the fact that the enumeration of programs is achieved in some standard (and informally effective) fashion. But similar results (the existence of a uniformly computable family of universal functions) hold for all models of computation, not only for the device/program ones. In the standard developments of computability theory, a comprehensive and precise treatment of these issues culminates in the construction of the Kleene Normal Form predicates. On the way to this construction, typically the so-called $s-m-n$ Theorem (sometimes called the Parametrization Theorem) is proved. This Theorem is mentioned, with details left to any of the standard treatments, in the proof of Theorem 4.1.6.

**Definition 2.2.4.** We fix an enumeration of programs, $(P_e | e \in \mathbb{N})$, with the property of Proposition 2.2.3, and for each positive integer, $n$, we denote its $n+1$-place computable universal function by $U^{(n)}$. We also denote its induced enumeration of $n$-place computable function by $\left(\varphi_e^{(n)} | e \in \mathbb{N}\right)$.

A consequence of the preceding Proposition and Definition is given in [4] and [18], and in what follows we shall actually use this variant. It illustrates a useful feature of taking our functions as having finite bitstrings as inputs and outputs: we can "fuse" index and input in a way that allows us to know where one ends and the other begins.

**Definition 2.2.5.** The one-place partial function $\mathbb{U} : \Sigma^* \to \Sigma^*$ is defined by taking its domain to consist of those members of $\Sigma^*$ of the form $0^i 1\sigma$ such that $\sigma$ is in the domain of $\varphi_i$, and when this occurs, $\mathbb{U}(0^i 1\sigma) = \varphi_i(\sigma)$.

Of course, by construction, $\mathbb{U}$ is universal for one-place computable functions.

**Proposition 2.2.6.** [4], [18] The one place partial function $\mathbb{U}$ is computable. □

**Definition 2.2.7.** If the program $P_e$ halts on input $n$ in at most $s$ steps, we write $P_{e,s}(n)\downarrow$. Otherwise, we write $P_{e,s}(n)\uparrow$. Of course, if $P_{e,s}(n)\downarrow$, then also $P_{e,s+1}(n)\downarrow$, while if $P_{e,s}(n)\uparrow$, both $P_{e,s+1}(n)\uparrow$ and $P_{e,s+1}(n)\downarrow$ are possible. If there is some $s$ such that $P_{e,s}(n)\downarrow$, we have a terminating computation on input $n$ and we write $P_e(n)\downarrow$; if there is no such $s$, we have a non-terminating computation on input $n$ and we write $P_e(n)\uparrow$. One of the reasons why this is such an important notion is that:

$$\{(s, e, n) | P_{e,s}(n)\downarrow\} \text{ is } \textit{decidable} \tag{2.1}$$

and not merely semi-decidable; here too, the underlying issues are addressed in the construction of the Kleene Normal Form predicates.

We use (e.g.) $\varphi_{e,s}(n)\downarrow$ to mean the same thing as $P_{e,s}(n)\downarrow$, and similarly for the other notions in the previous paragraph. We will abuse notation by writing $\mathbb{U}_s(n)\downarrow$ (resp. $\mathbb{U}_s(n)\uparrow$) to mean that *fixing in advance some particular index $e^*$ for* $\mathbb{U}$, $P_{e^*,s}(n)\downarrow$ (resp. $P_{e^*,s}(n)\uparrow$). This is an abuse of notation since, of course, which of these actually occurs depends on the choice of the index $e^*$.

For any (one-place) computable function $\varphi$ (and in particular when $\varphi$ is $\mathbb{U}$), we will write $\varphi(n)\downarrow$ to mean that $\varphi(n)$ is defined and $\varphi(n)\uparrow$ to mean that $\varphi(n)$ is undefined.

There is no abuse of notation here, since this independent of the choice of index for $\varphi$.

Finally, as usual, we define the (diagonal) *Halting Problem*, $0'$, to be $\{n|\varphi_n(n)\downarrow\}$.

We then have the following classical ground-breaking result, also due to Turing As usual, proofs can be found in [4], [18].

**Proposition 2.2.8.** The set $0'$ is c.e. but not computable. □

One property which is weaker than outright computability, but still allows for some effective applications is the following:

**Definition 2.2.9.** A real number $\gamma$ is called *left c.e.* if the left cut of $\gamma$, $L(\gamma) = \{q \in \mathbb{Q}|q < \gamma\}$ is c.e..

**Definition 2.2.10.** A function $f$ with domain included in $\Sigma^*$ is called *prefix-free* if for any $\sigma, \tau \in \mathrm{Dom}(f)$, $\sigma \preceq \tau \Rightarrow \sigma = \tau$.

We now have the analogue of Proposition 2.2.6 for prefix-free computable functions.

**Proposition 2.2.11.** [4], [18] There is a single one-place prefix-free (and so necessarily partial) computable function $\mathbb{V} : \Sigma^* \to \Sigma^*$ which is *universal* for one-place prefix-free computable functions, in that the following enumeration is an enumeration of *all* one-place prefix-free computable functions: given $i$, take $\psi_i$ to be the function $\sigma.\mathbb{V}(0^i1\sigma)$. □

It is worth giving a brief indication of how $\mathbb{V}$ is obtained from $\mathbb{U}$. This is by modifying, in the following way, each $\mathbb{U}_s$ to get what will turn out to be $\mathbb{V}_s$. When computing $\mathbb{V}_s(0^i1\sigma)$, we first check to see whether or not:

$$\text{for some } \tau \prec \sigma,\ \mathbb{U}_s\left(0^i1\tau\right). \tag{2.2}$$

As noted above, this is a decidable condition, and there are only finitely many $\tau$ to check. If this condition holds, then $\mathbb{V}_s\left(0^i1\tau\right)$ will be undefined; otherwise, $\mathbb{V}_s\left(0^i1\tau\right) := \mathbb{U}_s\left(0^i1\tau\right)$. Of course, if $\tau \prec \sigma$ and $\mathbb{U}_s\left(0^i1\tau\right)$ then for any $t \geq s$ it is also true that $\mathbb{U}_t\left(0^i1\tau\right)$.

**Definition 2.2.12.** We fix a function, $\mathbb{V}$, with the properties of Proposition 2.2.11. The enumeration it provides will be called the *standard prefix-free enumeration*, and it will be denoted by $(\psi_i | i \in \mathbb{N})$.

## Oracle Computations

Another important contribution of Turing was to consider (idealized) computations which have access to an element, $x$, of $2^{\mathbb{N}}$ which acts as an "oracle". In a register machine or similar model, we add to our list of program instructions an "oracle instruction", $O(n)$, which, when executed with $x$ "in the oracle", returns the $n^{\text{th}}$ bit of $x$, storing it in a specified location in memory so that it is then available for use in subsequent stages of the computation. All of the notions (and notations) developed above can be relativized to $x$, providing the notion of computability relative to $x$. Non-computable sets may become computable relative to some choices of $x$: e.g. $0'$ is trivially computable relative to its characteristic function (and similarly for any other set).

We do not enter into the details of their development, but we have the suitably reformulated analogues of all of the numbered items above, starting with Definition 2.2.1. As usual, a complete development is given in [4] and [18]. We do note that, in particular (with a different but equally standard (and still informally outright effective) enumeration of "oracle programs") we have, uniformly in $x$, the $x$-analogues of Propositions 2.2.3, 2.2.6, and 2.2.11 and their companion Definitions 2.2.4, 2.2.5, and 2.2.12. We adopt the following related notational conventions:

**NOTATIONAL CONVENTIONS:** For any $x \in 2^{\mathbb{N}}$:

1. $\varphi_e^x$ denotes the $x$-computable one-place partial function with standard oracle index $e$,

2. $\psi_e^x$ denotes the $x$-computable prefix-free one-place partial function with standard prefix-free oracle index $e$,

3. $\mathbb{U}^x$ denotes the $x-$computable one-place partial function analogous to $\mathbb{U}$:

$$\mathbb{U}^x(0^i1\sigma) \text{ is defined if and only if } \varphi_i^x(\sigma) \text{ is and } \mathbb{U}^x(0^i1\sigma) = \varphi_i^x(\sigma).$$

4. $\mathbb{V}^x$ denotes the $x-$computable one-place prefix-free partial function analogous to $\mathbb{V}$:

$$\mathbb{V}^x\left(0^i1\sigma\right) \text{ is defined if and only if } \psi_i^x(\sigma) \text{ is and } \mathbb{V}^x\left(0^i1\sigma\right) = \psi_i^x(\sigma).$$

Notions related to relative computability dominated the development of computability theory from the early 1940's, when Post formulated the celebrated problem that bears his name, through the 1990's when the remarkable collection of results on the structure of the Turing degrees was completed. It may also be worth pointing out that the idea of using random noise from outside systems to augment random number generation is actually quite similar in spirit to the ideas behind relative computability.

### 2.2.2 KFT and the MET

**Definition 2.2.13.** If $I \subseteq \mathbb{N}$ is an initial segment of $\mathbb{N}$ (i.e., either $I = \mathbb{N}$ or for some $n \in \mathbb{N}$, $I = [n]$), by a *request set on* $I$ we mean a (finite or infinite) sequence $\{< r_i, \zeta_i >\}_{i \in I}$ of (codes of) pairs $< r_i, \zeta_i >$, where each $r_i$ is a positive integer and each $\zeta_i \in \Sigma^*$.

The motivation for the terminology "request set" will only become clear when we arrive at the statement of the Machine Existence Theorem, Theorem 2.2.19, below, at which point the role of the $\zeta_i$ will also become clear. For now, we'll say just that the "request set asks us" to associate to each $i$, an element $\sigma_i$ of $\Sigma^*$ with $|\sigma_i| = r_i$. Typically, the set of $\sigma_i$'s is also required to be prefix-free, in which case a sequence $\{\sigma_i\}_{i \in I}$ that results from "meeting each request" will also be called a *prefix-free code*.

**Definition 2.2.14.** If $A \subseteq \Sigma^*$, define the *weight* of $A$ by $\text{wgt}A = \sum_{\sigma \in A} 2^{-|\sigma|}$. If $g$ is a function to $\mathbb{N}$ with domain $I \subseteq \mathbb{N}$, we define the *weight of* $g$ by $\text{wgt}g = \sum_{n \in I} 2^{-g(n)}$.

If $f$ is a function to $\Sigma^*$ with domain $I$ (with $I$ as above) we define the *weight of* $f$ by $\mathrm{wgt}f = \sum_{n\in I} 2^{-|f(n)|}$.

Then we say that $A$ (resp. $g$, resp. $f$ satisfies the *weight condition* if and only if weight of $A$ (resp. weight of $g$, resp. weight of $f$) is at most 1. A request set $\{< r_i, \zeta_i >\}_{i\in I}$ satisfies the weight condition if and only if $i.r_i$ satisfies the weight condition.

**Definition 2.2.15.** In the event that the $W = \{< r_i, \zeta_i >\}_{i\in I}$ is c.e., and satisfies the weight condition, we call $W$ a *bounded request set*.

**Lemma 2.2.16.** If a set $A$ is prefix-free, $\mathrm{wgt}A \leq 1$. □

Many standard arguments dealing with the notion of $K$-triviality (viz. Definition 2.3.26 in the next Section) involve constructing a request set and showing this set has certain properties. We are now in a position to present the MET, and its precursor, KFT. The latter says that given a request set with small enough total weight, we can find a prefix-free code that satisfies its requests. The MET says that for a bounded request set, this can be done effectively, uniformly in an index of the request set. Though both are proved in [4] and [18], we next present a proof of KFT, and in subsection 4.1.2, Theorem 4.1.6, we present a proof of a generalization of the MET. Both proofs are modelled on the ones given in the second of the preceding references. The proof of the generalization of the MET subsumes a proof of the MET itself, with an additional argument. In the oposite direction, in subsection 4.2.1, Theorem 4.2.1, we show that if, in Theorem 4.1.6, a key hypothesis there is dropped, then not only can the generalization of the MET fail, but even a related generalization of KFT can fail.

**Theorem 2.2.17. Kraft's Theorem** [18] A request set $W = \{< r_i, \zeta_i >\}_{i\in\mathbb{N}}$ satisfies the weight condition if and only if there is a prefix-free code $\{\sigma_i\}_{i\in\mathbb{N}}$ such that for all $i$, $|\sigma_i| = r_i$.

*Proof.* Necessity of the weight condition is clear - any prefix-free code has weight at most 1. For sufficiency, we will assume, WLOG, that the sequence $\{r_i\}_{i\in\mathbb{N}}$ is

nondecreasing; if it were not, just reindex it to make it so. Note that this reordering cannot be carried out effectively, unless the request set is outright computable. In the case of the MET, where the effectiveness matters, we will not be able to make this assumption, and this will require a somewhat different approach. On the other hand, in that setting, we will have that the request set is at least c.e.

We will construct the $\sigma_i$ by recursion on $i$. The basis of the recursion is to simply take $\sigma_0$ to be $0^{r_0}$. For the recursion step, we let $i > 0$, and we assume that the $\sigma_j$ defined at stages $j < i$ satisfy the following properties.

1. The set $\{\sigma_j | j < i\}$ is prefix-free and for all $j < i$, $|\sigma_j| = r_j$,
2. We let $U_i$ be the set of strings $\tau$ such that $|\tau| = r_i$ and such that for some $j < i$, either $\tau = \sigma_j$ or $\sigma_j$ is a prefix of $\tau$. We then assume that $U_i$ is an initial segment, for lexicographic order, of the set of all strings whose length is $r_i$.

Let $k := \#U_i$. Note that:

$$\frac{k}{2^{r_i}} = \sum_{j<i} 2^{-r_j} \text{ and by the weight condition } \sum_{j<i} 2^{-r_j} < 1. \tag{2.3}$$

Thus, the weight condition guarantees that there is some string $\tau$ such that $|\tau| = r_i$ and $\tau \notin U_i$. We simply take $\sigma_i$ to be the lexicographically least such $\tau$.

It follows (by construction of $\sigma_i$ and the definition of $U_i$) that we have preserved Property (1). The choice of $\sigma_i$ as lexicographically least (together with the definition of $U_i$) guarantees that we have also preserved Property (2). This completes the recursion and the proof.

□

**Remark 2.2.18.** Note that the $\zeta_i$'s do not have any role in the proof. They are only used in refinements of KFT (such as the MET).

In order to shed some light on the purpose of the preliminary re-ordering, consider the situation that arises if we had $r_1 = 2$, $r_2 = 2$, $r_3 = 1$ and had taken $\sigma_1 = 00$

and $\sigma_2 = 10$. Then the requirement that the set of $\sigma_i$'s be prefix-free means that we have no possible choice for $\sigma_3$. While any finite number of such conflicts can be anticipated and avoided, we cannot "look ahead infinitely often" in this way. But if the $r_i$ are reordered so as to appear in non-decreasing order, this sort of difficulty is avoided. We next state the extremely important MET.

**Theorem 2.2.19. Machine Existence Theorem** [18] Let $W = \{< r_i, \zeta_i >\}_{i \in \mathbb{N}}$ be a bounded request set, and let $e$ be an index of $W$ (i.e., let $e$ be such that $\varphi_e$ is the function $i. < r_i, \zeta_i >$). Effectively (and uniformly) in $e$, we can find a c.e. prefix-free code, $\{\sigma_i\}_{i \in \mathbb{N}}$, such that for all $i$, $|\sigma_i| = r_i$, and a prefix-free index, $d(e)$, of the prefix-free machine $M = \sigma_i.\zeta_i$ (meaning, among other things, that $M$ is computable, the domain of $M$ is precisely the set of $\sigma_i$'s, $\psi_{d(e)} = M$, and the function $e.d(e)$ is computable). □

The MET is an extremely powerful and useful tool in the standard setting and it figures prominently in many important results, such as Nies' Theorem, [17], that the $K$-trivial strings are low for $K$. As noted prior to the statement, above, of KFT, the status of KFT and the MET becomes rather complicated once we leave the standard setting. In the computable metric space setting, this difficulty is dodged since there, in virtue of the result of Melnikov and Nies, [15], given below as Theorem 3.1.5, the $K$-triviality notion can be viewed in terms of the (standard) $K$-triviality notion for a $K$-trivial Cauchy name.

## 2.3 Algorithmic Randomness In Cantor Space

In this Section, we flesh out our historical account, in Section 1.2, by giving precise definitions of and results for many of the basic notions of algorithmic randomness in the standard setting of Cantor Space equipped with Lebesgue (aka fair coin) measure that were disscussed there. First, we will give Martin-Löf's original definition of the notion of randomness that he introduced and that has come to bear his name. This is done in Definitions 2.3.1 and 2.3.2, after which we state, without

proof Proposition 2.3.3 which establishes that there is a single *ML-test* (viz. Definition 2.3.1) which alone suffices to detect every failure of Martin-Löf randomness.

We then turn our attention to presenting the basic theory of the two notions of the descriptive complexity of elements of $\Sigma^*$. These seek to measure the amount of information inherent in a finite binary string. The first notion is the plain descriptive complexity of such a string, $\sigma$, and it is denoted by $C(\sigma)$. The second notion is the prefix free descriptive complexity, or Kolmogorov complexity, denoted by $K(\sigma)$. There is a strong parallelism in the development of the two notions; compare, e.g., the numbered items 2.3.4 through 2.3.8 and 2.3.10 for $C$ with the numbered items 2.3.14 through 2.3.17 and 2.3.20, 2.3.24 for $K$.

Nevertheless, the development must be sequential, first for $C$ and then for $K$, except for Proposition 2.3.18. This is because all of the above-referenced numbered items for $C$ are needed for the proofs of Propositions 2.3.11 and 2.3.12 and their consequence, Corollary 2.3.13, and it is these results, in turn, which provide the explanation for why (in virtue of the unpleasant property of $C$ which they establish) the somewhat more delicate notion of prefix-free complexity has supplanted in importance the apparently simpler one of plain complexity. In other words, in order to "make the case for " $K$, we need to first establish much of the theory of $C$ (and only then re-establish much of it for $K$ itself).

The development of the theory of $K$ has some items not mirrored in the theory of $C$ (e.g. Proposition 2.3.21 and Lemma 2.3.23) and culminates with Schnorr's Theorem, Proposition 2.3.25, discussed in Section 1.2. This Section then concludes with the definition of $K$-triviality, Definition 2.3.26, completing the development in the standard setting.

**Definition 2.3.1. Martin-Löf Tests** We say $\{G_m\}_{m\in\mathbb{N}}$ is an *ML-test* if it is a uniformly c.e. sequence of open sets such that for all $m$, $\lambda(G_m) < 2^{-m}$.

**Definition 2.3.2.** We say that $x \in 2^{\mathbb{N}}$ is *Martin-Löf Random* if there is no ML-test

$\{G_m\}_{m\in\mathbb{N}}$ such that for all $m$, $x \in G_m$. If there exists $\{G_m\}_{m\in\mathbb{N}}$ such that for all $m$, $x \in G_m$, we say $x$ fails the ML-test $\{G_m\}_{m\in\mathbb{N}}$.

**Proposition 2.3.3.** There is a universal ML-test, $\{U_m\}_{m\in\mathbb{N}}$, in the sense that every $x \in 2^{\mathbb{N}}$ which fails some ML-test also fails $\{U_m\}_{m\in\mathbb{N}}$.

We now turn to the development of the theory of descriptive complexity. Recall that, as foreshadowed in Section 2.1, when we write things like $C(n)$ or $K(n)$, what we really mean is $C(a(n))$ or $K(a(n))$.

**Definition 2.3.4.** If $\tau \in \Sigma^*$ and $M$ is a universal computable function, we define $C_M(\tau)$ to be he smallest $\ell$ such that for some $\sigma$, $|\sigma| = \ell$ and $M(\sigma) = \tau$. We call $\sigma$ an *$M$-description of $\tau$*.

**Definition 2.3.5.** A universal computable function $R$ is called optimal if and only if for all $M$, $C_R \leq^+ C_M$.

**Proposition 2.3.6.** [4], [18] $\mathbb{U}$, as defined above, is optimal. □

**Definition 2.3.7.** We define $C := C_{\mathbb{U}}$.

Since neither $C$ nor $K$ (defined below in Definition 2.3.17) is computable, they are difficult to work with. However, we do have some basic bounds and properties which will be useful in what follows. For the reasons indicated above, we first develop most of the theory of $C$ before turning to $K$.

**Proposition 2.3.8.** [18], [24] If $f$ is total and computable, then $C(f(n)) \leq^+ C(n)$ □

**Remark 2.3.9.** This essentially says that from $n$, and an index for $f$, it is no harder to describe $f(n)$ than $n$.

We also have some useful general bounds:

**Proposition 2.3.10.** [4], [18] If $x \in 2^{\mathbb{N}}$ then $C(n) \leq^+ C(x \upharpoonright n) \leq^+ n$ □

As motivation for the first inequality, assume, to the contrary that $C(x \upharpoonright n) < C(n)$. The function $\sigma.|\sigma|$ is certainly computable, so we would essentially be obtaining less-than-minimal descriptions for $n$. The second inequality can be thought of intuitively as the idea that each string provides a description for itself. For a fixed optimal machine (one which gives the copying machine $\sigma.\sigma$ a small index), we may improve this bound to $C(\sigma \upharpoonright n) \leq n+1$. We will assume that our standard optimal machine has this property.

We are now in a position to prove some results that pinpoint an undesirable property of $C$: the failure of subadditivity, viz. Corollary 2.3.13, below. This is a consequence of the next two Propositions that preceed the Corollary. These Propositions involve growth rates and "dips in complexity".

**Proposition 2.3.11.** [4] For all $n \in \mathbb{N}$ there exists $\omega$ such that $|\omega| = n$ and $C(\omega) > |\omega|$. □

**Proposition 2.3.12.** [18] There exists $c$ such that for all $d \in \mathbb{N}$ and all $\tau$ such that $|\tau| \geq 2^{d+1} + d$, there exists $\sigma \preceq \tau$ such that $C(\sigma) \leq |\sigma| - d + c$. □

**Corollary 2.3.13.** [18] There exist $\sigma$, $\tau$ such that $C(<\sigma, \tau>) > C(\sigma) + C(\tau)$

*Proof.* Note that $C(\sigma\tau) < C(<\sigma, \tau>)$ (since we don't know where $\sigma$ ends and $\tau$ starts), so we prove something even stronger:
Let $\omega$ be as in 2.3.11, and $\sigma \preceq \omega$ be as in 2.3.12. Let $\tau$ be such that $\omega = \sigma\tau$. Then $C(\sigma) < |\sigma| + c - d$, and as for all $\tau$, $C(\tau) \leq |\tau| + 1$, hence

$$C(\omega) = C(\sigma\tau) > |\omega| > |\sigma| + |\tau| + 1 + c - d \geq C(\sigma) + C(\tau)$$

as long as $d$ is large enough. □

Because of this undesirable property of $C$, the prefix-free version, $K$, to which we now turn, has emerged as the "correct" notion of the descriptive complexity of a finite binary string.

**Definition 2.3.14.** If $\tau \in \Sigma^*$ and $M$ is a universal prefix-free computable function, define $K_M(\tau)$ to be the smallest $\ell$ such that for some $\sigma$, $|\sigma| = \ell$ and $M(\sigma) = \tau$. As before, we call $\sigma$ an *$M$-description of $\tau$*.

**Definition 2.3.15.** A universal prefix-free computable function $R$ is called optimal if and only if for all $M, K_R \leq^+ K_M$.

**Proposition 2.3.16.** [4], [18] $\mathbb{V}$, as defined above, is optimal. □

**Definition 2.3.17.** We define $K := K_{\mathbb{V}}$.

$K$ has properties very similar to those of $C$, but without the failure of subadditivity:

**Proposition 2.3.18.** [4], [18] The functions $C$ and $K$ are computably approximable from above, i.e., there are computable functions $C_s$, $K_s$ such that $C_s(\sigma) \leq C_{s-1}(\sigma)$, $K_s(\sigma) \leq K_{s-1}(\sigma)$, and $C_s(\sigma) \to C(\sigma)$, $K_s(\sigma) \to K(\sigma)$ as $s \to \infty$. □

**Remark 2.3.19.** The key point is that we cannot computably determine, given $s$, whether $C_s(\sigma) = C(\sigma)$ (resp. whether $K_s(\sigma) = K(\sigma)$), even though this will be true for sufficiently large $s$.

In the next few items we continue to develop the theory of $K$, paralleling the development for $C$.

**Proposition 2.3.20.** [18], [24] If $f$ is total and computable, then $K(f(n)) \leq^+ K(n)$ □

**Proposition 2.3.21.** [18], [24] If $f$ is total, computable and $1:1$, then $K(f(n)) =^+ K(n)$ □

**Remark 2.3.22.** In virtue of this Proposition, it is not unreasonable to make the notation $K(n)$ mentioned above in Section 2.1 do double duty by taking it to also denote $K(0^n)$. This is justified because the function $a(n).0^n$ is computable and 1:1, so $K(a(n)) =^+ K(0^n)$.

**Lemma 2.3.23.** Let $f$ be total, computable, and $m : 1$. Further, suppose $\#\{x : x \in f^{-1}(n)\}$ is computable in $n$ (or equivalently that there is a computable bound on the largest element of $f^{-1}[n]$). Then $K(f(n)) =^+ K(n)$.

*Proof.* $K(f(n)) \leq^+ K(n)$ is immediate. For the other inequality, let $g_i(n)$ denote the $i$th element of $f^{-1}[f(n)]$, or some default value if there are fewer than $i$ elements. Then for all $n$, $K(g_i(f(n))) \leq K(f(n)) + b_i$ is true for some $b_i$, for all $n$ and all $i$ less than $m+1$. Also, $n = g_i(f(n))$ for some $i \leq m$. Hence, $K(n) \leq K(f(n)) + \max\{b_i\}$, so $K(f(n)) =^+ K(n)$. □

**Proposition 2.3.24.** [4], [18] If $x \in 2^{\mathbb{N}}$ then $K(n) \leq^+ K(x \upharpoonright n) \leq^+ n + K(n)$. □

The intuition here is the same as for $C$, but the restriction to prefix-free functions adds extra overhead for the second inequality.

**Proposition 2.3.25. Schnorr's Theorem**,[18]: For $x \in 2^{\mathbb{N}}$, $x$ is *Martin-Löf Random* if and only if its complexity grows asymptotically as quickly as possible, i.e., if $K(x \upharpoonright n) \geq^+ n$.

A major factor in the emergence of Martin-Löf Randomness as the "preferred" notion of randomness is its characterization in terms of Kolmogorov complexity provided by Schnorr's Theorem. Martin-Löf Randomness also has many other characterizations, attesting to the robustness of the notion. Another major contribution of Schnorr's Theorem was to suggest an approach to characterizing the highly non-random strings in terms of the opposite behavior of $K$ (leading to the next Definition).

**Definition 2.3.26.** If $x \in 2^{\mathbb{N}}$, $x$ is *K-trivial* if and only if its complexity grows asymptotically as slowly as possible, i.e., if $K(x \upharpoonright n) \leq^+ K(n)$.

We shall also want a somewhat weaker notion: $x$ is *infinitely often K-trivial* if there is $b$ such that for infinitely many $n$, $K(x \upharpoonright n) \leq K(n) + b$.

The definition of $K$-triviality extends naturally to functions from $\mathbb{N}$ to $\mathbb{N}$, for example. If $f$ is such a function, we identify $f$, first with its coded graph, i.e. with the set $\{< n, f(n) >_{n\in\mathbb{N}}\}$. We can then idenitfy the coded graph with a single member of $2^{\mathbb{N}}$, via the characteristic function, and then apply the preceding definition to the characteristic function of the coded graph.

**Definition 2.3.27.** For$x \in 2^{\mathbb{N}}$, $x$ is called *low for* $K$ if and only if

$$\text{for all } y,\ K^x(y \upharpoonright n) \geq^+ K(y \upharpoonright n).$$

The set of such $x$ is denoted *Low(K)*.

It is clear that having the extra computational power of $x$ can never hurt, so we always have that for all $y$, $K^x(y \upharpoonright n) \leq^+ K(y \upharpoonright n)$. Computing complexity using $x$ as an oracle can provide insight into the computational power of $x$. For some strings $\sigma$, the information contained in $x$ will help us compute shorter descriptions of $\sigma$. For example, if $x \upharpoonright n$ had relatively high complexity, having access to the bits of $x$ in the oracle reduces the problem of describing $x \upharpoonright n$ to the problem of describing $n$. Some strings may not get shorter descriptions in this way. A string $x$ in *Low(K)* is considered to be computationally weak, since the bits of $x$ do not offer any utility in computing more efficient descriptions for any string. The following proposition says that being computationally weak in this way is the same as being easy to describe, in terms of $K$.

**Theorem 2.3.28.** [17] For $x \in 2^{\mathbb{N}}$, $x \in \text{Low}(K)$ if and only if $x$ is $K$-trivial. □

In Chapter 3 we will lay out how these notions were generalized to the related frameworks of computable metric spaces and computable measures which emerged in the wake of Gács' groundbreaking paper, [5]. This, in turn, sets the stage for the presentation of the results of this dissertation in Chapters 4 and 5.

# Chapter 3

# Beyond the Standard Setting

## Chapter Overview

Each of the two Sections of this Chapter is devoted to one of the two points of view that grew out of Gács' [5]. In Section 3.1, we will introduce the idea of a computable metric space. In Definition 3.1.4, we give Melnikov and Nies' successful generalization of the notion of $K$-triviality to this setting. Section 3.2 focuses on that of computable measures, where our development is considerably more in-depth.

In the next few paragraphs, but also to some extent, at the start of each Section, we will discuss the reasons for this asymmetry in parallel with a discussion of the important relationships between these two points of view. For the most part, at least early on, the main goal was to provide generalizations of the notion of Martin-Löf randomness. In view of the inherently measure-theoretic flavor of many of the notions (the ML-tests, mainly), the point of view of the computable measures provided the most natural setting.

Nevertheless, in order to achieve full generality, one needs a background computable metric space in which to work. Oddly, this has proven to be the more fruitful point of view for generalizing the $K$-triviality notion: it was in the language of computable metric spaces that Melnikov and Nies framed their Definition 3.1.4, which

has come to be regarded as the correct generalization for reasons already discussed. So the computable metric space point of view is already sufficient for generalizing the $K$-triviality notion and it provides the needed background for developing the computable measure notions, in terms of which the randomness notion is most naturally generalized.

Before turning to a quasi-historical account, a few more observations are in order. If one is content to consider computable measures on $\mathfrak{C}_T$, a more direct approach is possible, and indeed was taken even earlier by Zvonkin and Levin, [29]. It turns out that this already allows for full generality, but this could only be seen a posteriori, thanks to the results of Gács, Hoyrup, and Rojas, [6], and in particular, their notion of cellularization which provides a natural map to $\mathfrak{C}_T$: the map $w_{\mathcal{C}}$ whose properties are developed starting with Definition 3.2.3. The Zvonkin-Levin approach proceeds by direct construction of what we call a *cell-measure*, viz. Definition 3.2.13. The particular computable measures we construct in Chapter 4 are all constructed as cell-measures.

Finally, while the setting of $\mathfrak{C}_T$ provides full generality, there is a question that we regard as important that has not been investigated: given a computable metric space, $\mathcal{X}$, and a computable measure, $\nu$, on $\mathfrak{C}_T$, is there a computable measure, $\mu$, on $\mathcal{X}$ and a cellularization, $\mathcal{C}$ of $\mathcal{X}$ for which $\nu$ is the pushforward measure for $\mu$, obtained via $w_{\mathfrak{C}}$? In Definition 3.2.10, we construct a cellularization for any computable metric space which, we conjecture, allows us to "hit" any computable measure on $\mathfrak{C}_T$.

Soon after Martin-Löf's seminal definition of randomness, Zvonkin and Levin [29] began pushing the concept into the realm of general computable measures on $\mathfrak{C}_T$. For the most part, they were able to replicate the results from the standard setting; in particular, an analogue to Schnorr's Theorem was found, reinforcing the conviction that the theory was robust.

The general approach to computable measures really "took off" after Gács, Hoyrup, and Rojas [6] (to whom we will refer to jointly as GHR) developed their method of "cellularizing" a computable metric space. The cellularizations, in effect, allowed techniques typically applied to binary strings in Cantor space to be applied more broadly. Section 3.2 is somewhat ahistorical in that we start with the later general approach and only after discussion of the cellularizations and related questions indicated above do we turn to the historically earlier Zvonkin-Levin approach. The Section commences with a more detailed discussion of its organization and subsections.

## 3.1 Computable Metric Spaces

In 1937, Banach and Mazur gave a definition for computable real functions (unpublished, see [7]), but it was not considered widely successful. In 1955, Grzegorczyk [7], extending some of Kleene's work on computable functionals, introduced the concept of "type II effectiveness" - a slight modification of effectiveness in the traditional sense, which allows a machine to continually accept input bits, and produce output bits. In the 1960's, Kleene's work [10] on computable functionals, and Moschavakis' work [16] on effectivizing metric spaces pushed computable analysis forwards for some time, but it was not until 2005 the framework was suitably advanced to be able to study randomness by Gács in [5]. In 2007 ([9]) Hoyrup and Rojas use their cellularization technique to associate metrics with measures and prove some of the standard randomness results for general computable metric spaces: e.g., the existence of a universal test, and a characterization of randomness in terms of prefix complexity. In [14] and [15], Melnikov and Nies turn this metric space approach towards $K$-triviality. We present the background of this approach here.

The first step from a general separable metric space to the notion of a computable metric involves fixing an enumeration, without repetitions, of a countable dense subset.

**Definition 3.1.1.** Let $(X, d)$ be a separable metric space, and fix an enumeration without repetitions, $\{\alpha_p\}_{p\in\mathbb{N}}$, of a countable dense subset of $X$. The $\alpha_p$ will be called *special points*. Let $\mathcal{X} = (X, d, \{\alpha_p\}_{p\in\mathbb{N}})$. Then, $\mathcal{X}$ is a *computable metric space* if and only if the function $<i, j>.d(\alpha_i, \alpha_j)$ is computable (this means, in particular that for all $i, j \in \mathbb{N},\ d(\alpha_i, \alpha_j)$ is a computable real number, but it also requires that it is obtained uniformly and effectively from $i$ and $j$).

When $\mathcal{X}$ is a computable metric space, for $q \in \mathbb{Q}^+$ and $p \in \mathbb{N}$, we call the open balls of the form $B(\alpha_p, q)$ *computable balls*, and we enumerate the family of computable balls of $\mathcal{X}$ as $\{B_i\}$, where we take $B_i$ to be the open ball with center $\alpha_{\pi_1(i)}$ and radius $q_{\pi_2(i)}$.

Throughout the rest of this dissertation, $\mathcal{X}$ **will denote a fixed but arbitrary computable metric space, as in the previous Definition.** Additional hypotheses on $\mathcal{X}$ will be introduced as needed.

**Definition 3.1.2.** If $x \in X$ and $c : \mathbb{N} \to \mathbb{N},\ c$ is a *Cauchy name for* $x$ if $\alpha_{c(n)} \to x$ and $d(\alpha_{c(s)}, \alpha_{c(t)}) \leq 2^{-s}$, for all $t > s$.

**Definition 3.1.3.** If $x \in X,\ x$ is called *computable* if and only if (for our fixed effective enumeration of $\mathbb{Q}$), there is a computable function $n.p(n)$ such that for all $n,\ d(\alpha_{p(n)}, x) < q_n$.

### 3.1.1 Randomness in Computable Metric Spaces

In [9], Hoyrup and Rojas define randomness for computable metric spaces. However, the notions of randomness are not given in terms of a metric structure. Instead, they can apply their cellularization technique given in Section 3.2. Thus, randomness in metric spaces is really dealt with in subsection 3.2.3, when we talk about randomness for computable measures.

### 3.1.2 $K$-triviality in Computable Metric Spaces

Melnikov and Nies, [15] propose the following generalization of $K$-triviality to computable metric spaces. We will refer to this generalization as $K_{\text{MN}}$-triviality. This definition agrees with the standard definition in Cantor Space. Most of the important properties of $K$-trivial strings carry over to the $K_{\text{MN}}$ ones.

**Definition 3.1.4.** Let $x \in X$. Then we define $x$ to be $K_{\text{MN}}(\mathcal{X})$-trivial if and only if there exists $b$ such that for all $n \in \mathbb{N}$ there exists $p$ such that $d(x, \alpha_p) < q_n$, and $K(< p, n >) \leq K(n) + b$. By analogy with Definition 2.3.26, we have the notion of *infinitely often* $K_{\text{MN}}(\mathcal{X})$-trivial; we omit the obvious details.

When $\mathcal{X}$ is clear from context, as it often will be, we may omit mention of $\mathcal{X}$.

We will also use the following notation: $KT_{\text{MN}}(\mathcal{X}) :=$ the set of $x \in X$ such that $x$ is $K_{\text{MN}}(\mathcal{X})$-trivial.

The next result is quite important, and was a major factor in cementing the conviction that $K_{\text{MN}}$-triviality is the correct generalization of $K$-triviality: since $K$-trivial Cauchy names are available, most of the (most) important results from the standard setting carry over, via these $K$-trivial Cauchy names.

**Theorem 3.1.5.** [15] For $x \in X,\ x \in KT_{\text{MN}}(\mathcal{X})$ if and only if $x$ has a $K$-trivial (in the standard sense) Cauchy name. □

**Remark 3.1.6.** The left-to-right implication is the significant one and was much more difficult to prove. Of course, $K_{\text{MN}}(\mathcal{X})$-trivial elements of $X$ can also have non-$K$-trivial Cauchy names, and there may not be a canonical way of choosing a $K$-trivial Cauchy name.

## 3.2 Computable Measures: The GHR framework

We begin this Section by presenting the general approach to computable measures on a computable metric space, developed, jointly, by Gács, Hoyrup, and Rojas [6].

We will refer to this approach to computable measures as the GHR framework. This is carried out in Definitions 3.2.1 through 3.2.5 and Proposition 3.2.4.

In particular, the cellularization notion is introduced in Definition 3.2.2. The ultimate goal is to define the map, $w_{\mathcal{C}}$, introduced in Definition 3.2.3. This is a map from a subset of $X$ (the ambient set of the background computable metric space), the set of points represented in the cellularization, to their representations by elements of $2^{\mathbb{N}}$. The rough idea is to break down the ambient set, $X$, into a sequence of pairs of disjoint subsets. For each represented point, $x$, and each $n$, the $n^{\text{th}}$ bit of $w_{\mathcal{C}}(x)$ tells us in which subset in the $n^{\text{th}}$ pair $x$ lies.

Additional properties of a cellularization are introduced in Definition 3.2.6. These are designed to guarantee that, among other things, almost all points of $X$ are represented. A cellularization satisfying these additional properties (one of which refers to a computable measure, $\mu$) is called a generator (for $\mu$) with a corresponding change in notation from the more general $\mathcal{C}$ (for arbitrary cellularizations) to $\mathcal{A}$, for generators (of some computable measure $\mu$). A specific generator, $\mathcal{A}_d$, (for any computable measure $\mu$) was constructed by Hoyrup and Rojas, [9]. Their construction is presented in Definition 3.2.9. In Definition 3.2.10 we give a more intricate construction of a generator, $\mathcal{A}'_d$, which we conjecture (based on a detailed proof strategy) has significantly better properties, but which is also "always available". Additional properties of an arbitrary generator are developed in Proposition 3.2.8 and the properties we conjecture hold for $\mathcal{A}'_d$ are presented following its construction.

Starting with Definition 3.2.13, we specialize to $\mathfrak{C}_T$ and cell-measures, rejoining the earlier Zvonkin-Levin approach. This leads naturally to subsections 3.2.1 and 3.2.2 which deal with this more special situation. The first of these subsections presents the notion of *granularity*, due to Hölzl and Porter, [8]. In subsection 3.2.2, we discuss how KFT and the MET may be generalized to the setting of computable measures on $\mathfrak{C}_T$. Finally, in subsection 3.2.3, we return to the a more general setting to discuss

various approaches to generalizing the notion of Martin-Löf randomness. The discussion is very brief because, as we've already noted, the main focus of our work is on the "far from random side", and the discussions of randomness are mainly for the purposes of motivation, perspective and analogy. A notable exception, in this subsection, is that, at its close, we introduce J. Rute's suggestion, in [21], presented as our Definition 3.2.22, for an approach to a generalization of $K$-triviality in measure-theoretic terms. The point is that the motivation came from one of the approaches to generalizing the randomness notion. Rute also asked whether his notion might be equivalent to the approach of Melnikov and Nies, but he did not pursue this question.

**Definition 3.2.1.** If $A \subseteq X$, $A$ is called $\Sigma_1^0$ if $A$ can be expressed as $A = \bigcup_{i \in I} B_i$, where the $B_i$ are the computable balls of $\mathcal{X}$, as in Definition 3.1.1 and $I$ is c.e.. An index of any such $I$ will be called *an index of A*.

**Definition 3.2.2.** By a *cellularization* for $\mathcal{X}$, we mean a sequence $\mathcal{C} = \{(A_i^0, A_i^1)\}_{i \in \mathbb{N}}$, where for each $i$, $A_i^0$, $A_i^1$ are disjoint $\Sigma_1^0$ sets.

We can then represent certain elements of $X$ by infinite binary strings, using a cellularization as follows.

**Definition 3.2.3.** [21] Let $\mathcal{C}$ be a cellularization for $\mathcal{X}$. For a finite bitstring $\sigma$ of length $n$, define $[\sigma]_{\mathcal{C}} = A_0^{\sigma(0)} \cap A_1^{\sigma(1)} \cap \cdots \cap A_{|\sigma|-1}^{\sigma(|\sigma|-1)}$. This will be referred to as a *cell*. Define $x \upharpoonright_{\mathcal{C}} n$ to be the unique $\sigma$ of length $n$ such that $x \in [\sigma]_{\mathcal{C}}$ if there is such a $\sigma$; otherwise $x \upharpoonright_{\mathcal{C}} n$ is undefined. We say $x$ is a represented point under $\mathcal{C}$ or $x \in \mathrm{rep}_{\mathcal{C}}$ if and only if $x \upharpoonright_{\mathcal{C}} n$ is defined for all $n$. When $x \in \mathrm{rep}_{\mathcal{C}}$, let $w = w_{\mathcal{C}}(x)$ in $2^{\mathbb{N}}$ be such that for all $n$, $[w \upharpoonright n] = x \upharpoonright_{\mathcal{C}} n$.

The next Proposition gives an important property of $w_{\mathcal{C}}$. The proof is both standard and straighforward, and it is omitted.

**Proposition 3.2.4.** Let $\mathcal{T}_1$ be the topological space $rep_{\mathcal{C}}$, equipped with the relative topology induced by the metric topology of $\mathcal{X}$, and let $\mathcal{T}_2$ be $2^{\mathbb{N}}$ equipped with the metric topology of $\mathfrak{C}$. Then, $w_{\mathcal{C}}$ is 1:1 and continuous with continuous inverse, viewed as a map from $\mathcal{T}_1$ to $\mathcal{T}_2$. □

Hoyrup and Rojas [9] (their Definition 4.1.2) call a measure on $X$ *computable for* $\mathcal{X}$ if and only if it is a *constructive point* of the space, $\mathcal{M}(\mathcal{X})$, of all (probability) measures on $\mathcal{X}$, equipped with the Prokhorov metric. However, they proceed to show this to be equivalent to a more useful condition (their Theorem 4.2.1). We take this condition to be our definition of a computable measure.

**Definition 3.2.5.** [9] *A* measure $\mu$ on the Borel sets of $(X, d, \{\alpha_i\}_{i\in\mathbb{N}})$ is *a computable measure on* $\mathcal{X}$ if and only if the function $<i_1, \cdots, i_k>.\mu(B_{i_1} \cup \cdots \cup B_{i_k})$ is lower semicomputable, where $B_i$ is the open ball around $\alpha_{\pi_1(i)}$ of radius $q_{\pi_2(i)}$. If $\mu$ is a computable measure on $\mathcal{X}$, we say that $\mu$ is $\mathcal{X}$-*orderly* if and only if for all open balls, $B$ of $\mathcal{X}$, $\mu(\delta\, B) = 0$.

Suppose $\mu$ is a computable measure on $\mathcal{X}$. It is natural and desirable to seek cellularizations $\mathcal{C}$ for which $\mu$-almost all points of $X$ are represented. This leads us to impose two additional conditions on $\mathcal{C}$. The first is directly designed to achieve this goal. For the second condition, let $\mathcal{L}$ be the lattice of sets generated by $\{A_i^0\}_{i\in\mathbb{N}} \cup \{A_i^1\}_{i\in\mathbb{N}}$, i.e. the closure of $\{A_i^0\}_{i\in\mathbb{N}} \cup \{A_i^1\}_{i\in\mathbb{N}}$ under finite unions and intersections, and let $\{L_i\}_{i\in\mathbb{N}}$ be an enumeration of $\mathcal{L}$ (obtained in some effective fashion from $\mathcal{C}$). The second condition will guarantee that $\mathcal{L}$ is a basis for the metric topology, and (in the sense made precise by the condition) effectively so. In particular, the second condition will guarantee that the cells (or cylinders), $\sigma_{\mathcal{C}}$, of the previous Definition can be uniformly and effectively expressed as effective unions of computable open balls.

**Definition 3.2.6.** Let $\mu$ be a computable measure on $\mathcal{X}$, and $\mathcal{C} = \{(A_i^0, A_i^1)\}_{i\in\mathbb{N}}$ be a cellularization for $\mathcal{X}$. We call $\mathcal{C}$ a *generator* for $(\mathcal{X}, \mu)$ if the following two conditions are met:

1. for all $i$, $\mu(A_i^0 \cup A_i^1) = 1$, and
2. if $U \subseteq X$ is a $\Sigma_1^0$ set, then there is a c.e. set, $J$, of indices such that, $\mu$-a.e., $U = \bigcup_{j\in J} L_j$; further, we can find an index for $J$, uniformly and effectively, from an index for $U$.

When condition 1. of the previous Definition is satisfied, the pairs $(A_i^0, A_i^1)$ are commonly referred to *a.e. decidable pairs*. Though being orderly may not, strictly speaking, be a necessary condition for condition 1. to hold, it is nevertheless a natural condition to impose. Henceforth we adopt the following convention:

Going forward, all computable measures are assumed to be orderly.

Note that condition 2. of the previous Definition makes no reference to $\mu$. In the case that $\mathcal{C}$ is a generator, it is more common to denote it by $\mathcal{A}$. We adopt this convention in what follows. Even for generators, $\mathcal{A}$, we may have $x \notin \text{rep}_{\mathcal{A}}$. Such unrepresented points $x$ have special computability properties which suggest they are far from being random. As such, they do not present any problems in generalizing randomness notions, but may be problematic in trying to generalize notions like $K$-triviality.

**Definition 3.2.7.** If $\mathcal{A}$ is a generator for $(\mathcal{X}, \mu)$ we define $\mu^*$ to be the "push-forward" measure on $2^{\mathbb{N}}$ obtained from $\mu$ via $w_{\mathcal{A}}$ in the standard way: $S \subseteq 2^{\mathbb{N}}$ is $\mu^*$-measurable if and only if $w_{\mathcal{A}}^{-1}[S]$ is, and when this is true, $\mu^*(S) := \mu\left(w_{\mathcal{A}}^{-1}[S]\right)$.

We then have a correspondingly stronger form of Proposition 3.2.4. The proof is, once again, routine, standard and omitted.

**Proposition 3.2.8.** If $\mathcal{A}$ is a generator for $(\mathcal{X}, \mu)$ then $\mu^*$ (as in the previous Definition) is a computable measure on $\mathfrak{C}_T$. Further, the continuity of $w_{\mathcal{A}'_d}$ and $w_{\mathcal{A}'_d}^{-1}$ is effective in that inverse and forward images of $\Sigma_1^0$ sets are $\Sigma_1^0$. □

In fact, the compact statement of the previous Proposition exploits the convention established in the final paragraph of Section 2.1: more formally, the conclusion is properly stated as: $\mu^*$ is a computable measure on $\mathcal{X}^* = \left(2^{\mathbb{N}}, d^*, \{\alpha_i^*\}_{i\in\mathbb{N}}\right)$, whenever $\mathcal{X}$ is a computable metric space for which the metric topology of $d^*$ coincides with

the usual metric topology of $\mathfrak{C}$.

Hoyrup and Rojas, [9], give a standard way to get a cellularization from the open balls which will be a generator for $(\mathcal{X}, \mu)$ *whenever* $\mu$ is a computable measure on $\mathcal{X}$.

**Definition 3.2.9.** Let $\left(X, d, \{\alpha_i\}_{i\in\mathbb{N}}\right)$ be a computable metric space. Define $\mathcal{A}_d$ by

$$A^1_{<i,j>} = B(\alpha_i, q_j),\ A^0_{<i,j>} = X - \overline{B}(\alpha_i, q_j),$$

Certainly we can write $A^1_{<i,j>}$ effectively as a union of computable balls (just $B(\alpha_i, q_j)$). We can also effectively enumerate $A^0_{<i,j>}$, by searching through all computable balls and enumerating those computable balls $B(\alpha_m, q_n)$ such that $d(\alpha_i, \alpha_m) > q_j + q_n$. However, this enumeration also has an unpleasant property - namely that there will always be $\sigma$ for which $[\sigma]_{\mathcal{A}_d} = \emptyset$. To see this, simply take disjoint balls $B(\alpha_{i_1}, q_{j_1})$ and $B(\alpha_{i_2}, q_{j_2})$, and let $\sigma\left(< i_1, j_1 >\right) = \sigma\left(< i_2, j_2 >\right) = 1$. Nothing in the relevant definitions rules this out, but it severely restricts what the push-forward measures, $\mu^*$, obtained via $w_{\mathcal{A}_d}$, can be like: they must assign measure 0 to any $\sigma$ for which (as in the previous paragraph) $[\sigma]_{\mathcal{A}_d} = \emptyset$.

We now undertake the construction of a cellularization, $\mathcal{A}'_d$ of $\mathcal{X}$ which, like $\mathcal{A}_d$, will be a generator for $(\mathcal{X}, \mu)$ whenever $\mu$ is a computable measure on $\mathcal{X}$, but without the unpleasant property just noted for $\mathcal{A}_d$. Indeed, rather than any unpleasant properties, $\mathcal{A}'_d$ will have some extremely useful properties.

**Definition 3.2.10.** We will denote by $\mathcal{A}'_d$ the cellularization of $\mathcal{X}$ constructed in what follows.

**CONSTRUCTION OF $\mathcal{A}'_d$:**

Let $\left(X, d, \{\alpha_i\}_{i\in\mathbb{N}}\right)$ be a computable metric space. We will construct a generator $\mathcal{A}'_d$ in stages. We fix an enumeration $\{B_i\}_{i\in\mathbb{N}}$ of computable balls to aid in the construction (for concreteness, we fix $B_{<i,j>} = B(\alpha_i, q'_j)$, where $\{q'_j\}$ is an enumeration of all dyadic rational numbers such that for no $i, j$ does $B_{<i,j>}$ cover all of $X$). We will also maintain an array, $B_{(i,j,k)}$, of balls, defined for all $i$, for all $j \geq i$, and for all $k < 2^{j-i-1}$. In determining $A^1_i$ and $A^0_i$, we will first construct auxiliary $\Sigma^0_1$ sets, $A'^1_{i,X}$ and $A'^0_{i,X}$. It will be helpful to note that the following operations are effective:

1. determining whether $B_{m_1}$ and $B_{m_2}$ are disjoint is computable. This can be done by checking that

$$d(\alpha_{\pi_1(m_1)}, \alpha_{\pi_1(m_2)}) > q_{\pi_2(m_1)} + q_{\pi_2(m_2)}.$$

2. determining whether $B_{m_2} \subseteq B_{m_1}$ is computable. This can be done by checking that

$$d(\alpha_{\pi_1(m_1)}, \alpha_{\pi_1(m_2)}) + q_{\pi_2(m_2)} < q_{\pi_2(m_1)}.$$

3. determining whether $B_{m_2} \subset B_{m_1}$, i.e., $B_{m_2}$ is a proper subset of $B_{m_1}$. This is not fully computable, but enumerable. First determine if $B_{m_2} \subset B_{m_1}$, and then enumerate special points, $\alpha_m$, until one is found such that $d(\alpha_m, \alpha_{\pi_1(m_1)}) < q_{\pi_2(m1)}$, but $d(\alpha_m, \alpha_{\pi_1(m_2)}) > q_{\pi_2(m2)}$. This technically fails to detect the possibility that $B_{m_2} \cup \{x\} = B_{m_1}$, where $x$ has no Cauchy name, but such $x$ cannot be referenced by the computable framework of special points at all, and so will be disregarded.

Construction of $A'^1_{i,X}$ and $A'^0_{i,X}$:

Stage $n = 0$: Pick $B_{(0,0,0)} = B_0$, and set $A'^1_{0,X} = B_{(0,0,0)}$. Then $A'^0_{0,X}$ is enumerated by enumerating balls $B_m$, and placing them into $A^0_0$ if they are disjoint from $A'^1_{0,X}$. We will thus always have that $A'^0_{0,X}$ (and, in fact, $A'^0_{n,X}$, for all $n$) is an infinite union of balls, but we can also give an index for the enumeration effectively.

Stage $n > 0$: For all $i \in [0, n)$, all $j \in [i, n)$, and $k \in [0, 2^{j-i-1})$, $B_{(i,j,k)}$ is defined. For all $i < n$, We define $B_{(i,n,k)}$ as follows:

Given $k \in [0, 2^{n-i})$, define

$$j'_k = \text{ the least } s \text{ such that } k \leq \sum_{r=0}^{s} 2^r,$$

and

$$k' = k - \sum_{r=0}^{j'_k} 2^r.$$

Now, to define $B_{(i,n,k)}$, enumerate all computable balls, $B_m$, until one is found such that $B_m \subset B_{(i,j'_k,k')}$ (we will actually require that $B_m \subset B(\alpha_{\pi_1(m')}, \frac{q_{\pi_2(m')}}{4})$, where$B_{m'} = B_{(i,j'_k,k')}$, but the reason why won't become apparent until later), and also $B_m$ is disjoint from all

$$\bigcup_{r=j'_k+1}^{n-1} \bigcup_{s=0}^{2^r-1} B_{(i,r,s)}.$$

Since this still a finite union of balls, detecting disjointness is still effective. Now, define $B_{n,n,0}$ by enumerating balls until one is found to be disjoint from all other $B_{i,i,0}$ for all $i < n$. Finally, define

$$A'^{1}_{n,X} = \bigcup_{i=0}^{n} \bigcup_{0}^{2^r-1} B_{(i,n,s)},$$

and, as before, $A'^{0}_{n,X}$ is the enumeration of open computable balls which are disjoint from $A'^{1}_{n,X}$, which we can effectively give an index for.

We note that if $A'^{1}_{i,X}$, and $A'^{0}_{i,X}$, were taken to be the a.e. decidable pairs of a generator, it would guarantee that all cells are nonempty, but, it is possible that there is an open set $U$ such that for all $x \in U$, $w_{\mathcal{A}}(x) = 0^\infty$. This is clearly an undesirable property for a representation to have. We also define $A'^{1}_{n,B_m}$ and $A'^{0}_{n,B_m}$ exactly as we defined $A'^{1}_{n,X}$ and $A'^{0}_{n,X}$ above, except that all balls used must be proper subsets of $B_m$. This will allow us to preserve the property of our generator producing no

null cells.

Now, to remedy the fact that many points may not be represented, we will incorporate all $B_m$ into our generator. At stage $n$, if $B_n$ does not cover any of the balls $B_{(i,j,k)}$ which have been defined up to stage $n$, then we let $A^1_n = B_n \cup {A'}^1_{n,X}$, and define $A^0_n$ to be an enumeration of the balls in the complement of $A^1_n$. On the other hand, if for some $i, j, k$, $B_{(i,j,k)} \subset B_n$, we let $\{m_s\}_{s\in I}$ be an enumeration of indices of balls such that $B_{m_s} \in B_n$, and then define $\{C_t\}_{t\in\mathbb{N}}$ be an enumeration of the balls which are contained within $B_n$, but are disjoint from $B(\alpha_{\pi_1(m_s)}, \frac{q_{\pi_2(m_s)}}{2})$. Now, we define

$$A^1_n = \left(\bigcup_{t\in\mathbb{N}} C_t\right) \cup {A'}^1_{n,X}.$$

$A^0_n$ to be an enumeration of the balls in the complement of $A^1_n$, as before.

This completes the construction of $\mathcal{A}'_d$.

**Conjecture 3.2.11.** The cellularization $\mathcal{A}'_d$ just constructed has the following properties.

1. For any (orderly) computable measure $\mu$ on $\mathcal{X}$, $\mathcal{A}'_d$ is a generator for $(\mathcal{X}, \mu)$.
2. For all finite binary strings $\sigma$, $[\sigma]_{\mathcal{A}'_d}$ is nonempty.
3. If $\nu$ is a continuous computable measure on $\mathfrak{C}_T$, and $\mu$ is the push-forward measure for $\nu$ obtained from $w^{-1}_{\mathcal{A}'_d}$, then $\nu = \mu^*$.

The idea of the proof is as follows: property (1) is routine. Property (2) is essentially seen in Figure 3.1, except the case of overlap from $B_n$ at stage $n$. For property (3), we need to be clear about exactly how the balls and cells relate via $w$ (which cells are in the pushforward of a ball, which balls are in the pullback of a cell). Once this is done, the proof should be routine.

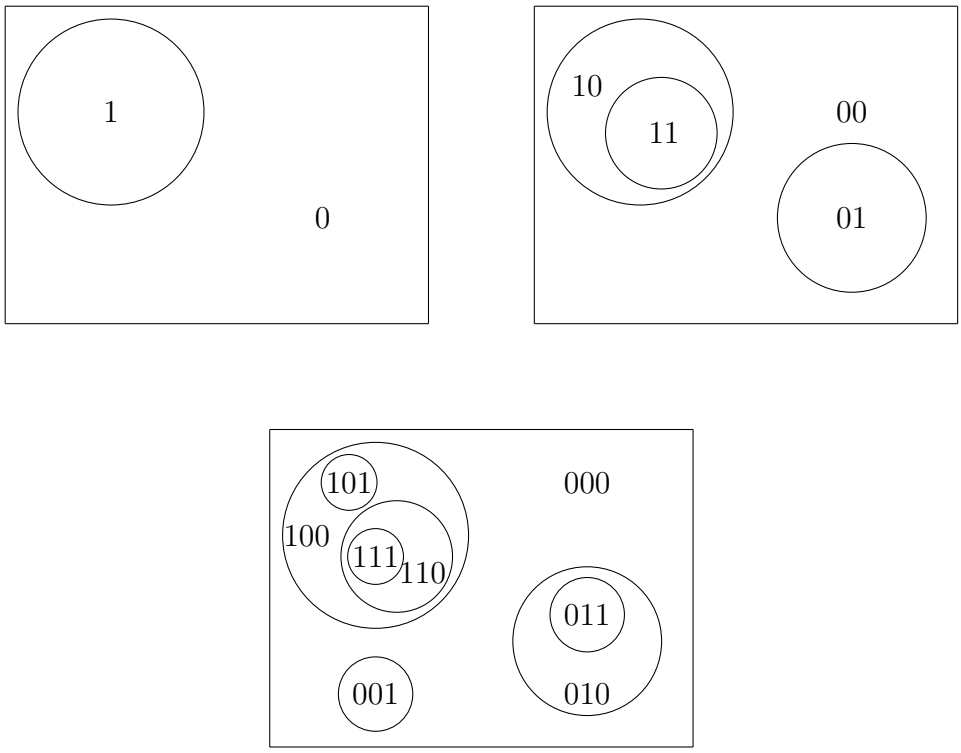


**Figure 3.1:** $\mathcal{A}'_d$ emulates Cantor space via $A'^0_{i,X}$, $A'^1_{i,X}$ in a general metric space

The previous Proposition provides a complete connection between the GHR approach to computable measures embodied in Definitions 3.2.1 through 3.2.7, and Propositions 3.2.4 and 3.2.8 on the one hand, and $\mathcal{M}(\mathcal{C}_T)$, on the other. This is because, in view of property 3. of the Proposition, and unlike the situation for $\mathcal{A}_d$, above, every continuous computable measure on $\mathcal{C}_T$ arises as the push-forward measure of some computable measure on $\mathcal{X}$.

We proceed, now, to give the more direct and historically earlier approach to computable measures on $\mathfrak{C}_T$ used by Zvonkin and Levin, [29]. This approach will be used in our work in Chapter 4.

**Definition 3.2.12.** A function $\rho : \Sigma^* \to [0,1]$ is called a *cell-semi-measure on* $2^{\mathbb{N}}$ if

it satisfies:

$$\rho(\epsilon) = 1, \text{ and } \rho(\sigma 0) + \rho(\sigma 1) \leq \rho(\sigma).$$

If $\rho$ is a cell-semi-measure where equality holds, i.e., $\rho(\sigma 0) + \rho(\sigma 1) = \rho(\sigma)$, then $\rho$ is called a *cell-measure on* $2^{\mathbb{N}}$. Such a cell-measure is called *continuous* if and only if for all $x \in 2^{\mathbb{N}}$, $\lim_{n\to\infty} \rho(x \upharpoonright n) = 0$. Otherwise, it is called a *cell-measure with atoms*.

**Definition 3.2.13.** A cell-measure, $\rho$, on $2^{\mathbb{N}}$ is called *computable* if and only if there exists total, computable $\tilde{\rho} : \mathbb{N} \times \Sigma^* \to \mathbb{Q}_2$ such that

$$|\rho(\sigma) - \tilde{\rho}(i, \sigma)| \leq 2^{-i}.$$

Where $\mathbb{Q}_2$ denotes the dyadic rationals. Such a cell-measure is called *exactly computable* if $\tilde{\rho}$ is constant in its first argument, i.e., the approximation is exact.

We can obtain, canonically from a cell-measure, $\rho$, a measure, $\mu^\rho$, on $2^{\mathbb{N}}$, in the usual sense, by setting $\mu^\rho([\sigma]) := \rho(\sigma)$, and extending this definition to Borel subsets (for the usual topology) of $2^{\mathbb{N}}$ in the usual way. Further, any measure on the Borel sets of $2^{\mathbb{N}}$ arises as $\mu^\rho$, for the obvious $\rho$, since the measure is completely determined by its restriction to the cells $[\sigma]$.

Further, it is easily seen that a cell-measure $\rho$ satisfies Definition 3.2.13 if and only if its $\mu^\rho$ is a computable measure on $\mathfrak{C}_T$. These observations fully justify the typical abuse of notation/terminology involved in identifying a cell-measure, $\rho$ with its $\mu^\rho$, and via that identification, taking Definition 3.2.13 as defining the notion of computable measure on $\mathfrak{C}_T$. We shall proceed in exactly this fashion in Chapter 4, by constructing computable cell-measures, and taking them to have constructed the associated computable measures. In fact, with the exception of the construction in Theorem 4.1.5, the computable measures constructed in Chapter 4 will all be exactly computable.

It will be useful to be able to visually represent some of the measures we construct as depicted in Figure 3.2 which follows. The size of each cell corresponds to its

measure, and cells are stacked on top of each other so that extensions are above their prefixes.

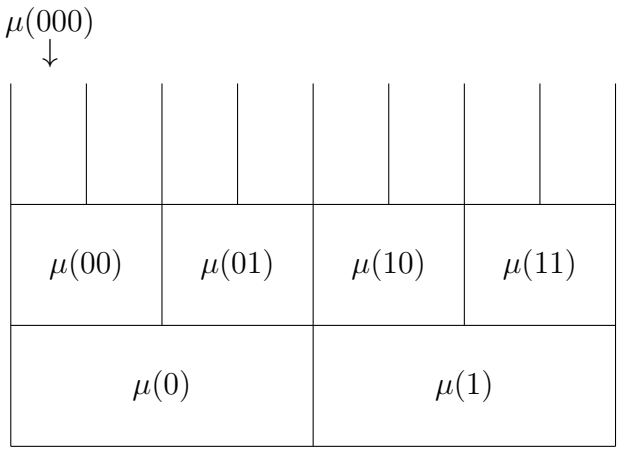


**Figure 3.2:** A visual representation of $\mu$

For example, if we have $\mu(0) = .7$, $\mu(00) = .1$, we may visualize this in Figure 3.3 as follows.

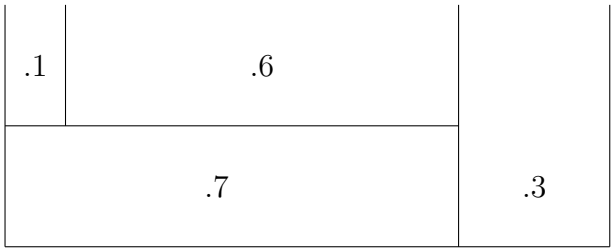


**Figure 3.3:** An example visualization

### 3.2.1 Granularity

In [8], Hölzl and Porter define a useful notion related to randomness.

**Definition 3.2.14.** If $\mu$ is a computable measure (on $\mathfrak{C}_T$) define $g_\mu$ for all $n > 0$ by $g_\mu(n) =$ the smallest $m$ such that all strings of length $m$ have $\mu$-measure less than $2^{-n}$.

It is obvious that $g_\mu(n)$ is always larger than $n$. Lebesgue measure has the slowest possible granularity function, $g_\lambda(n) = n+1$. More irregular measures generally have faster growing granularity, for example, $g_{\mu^\dagger}$ as defined below in 4.3.1 has at least exponential growth. The following is a useful fact about $g_\mu$.

**Proposition 3.2.15.** [8] For any computable measure $\mu$, its granularity function, $g_\mu$, has a computable upper bound $g^*_\mu$.

*Proof.* Let $\tilde{\mu}$ be as guaranteed for $\mu$ by Definition 3.2.13, and for finite bitstrings, $\sigma$, and natural numbers, $j$, we use $\mu_j(\sigma)$ in place of $\tilde{\mu}(j, \sigma)$. Let $m, n$ be positive integers $n$. We let $\{\sigma_i\}_{i<2^n}$ be any enumeration without repetitions of the set of all strings of length $n$. Note that Definition 3.2.13 guarantees that for sufficiently large $j$, $\sum_i \mu_j(\sigma_i) > 1-2^{-m}$, and let $s(m, n)$ be the least $j$ for which this holds. Note that $s$ is computable and defined for all $m, n > 0$. We define the computable predicate $P(m, n)$ by letting this hold if and only if $m, n > 0$ and for all $i < 2^n$, $\mu_{s(m,n)}(\sigma_i) < 2^{-m}$. Note that if $P(m+1, n)$ holds, then $g_\mu(m) \leq n$, so finally, taking

$$g^*(m) := \text{the least } n \text{ such that } P(m+1, n) \text{ holds},$$

we have that $g^*$ is the required computable upper bound. □

If $\mu$ is exactly computable, then $g_\mu$ is always computable. If $\mu$ is computable, but not exactly computable, it is possible that $g_\mu$ is not computable. In Proposition 5.2 of [8], Hölzl and Porter construct a computable measure for which

$$g_\mu(n) = \begin{cases} 2n & \text{if } \phi_n(n) \downarrow \\ 2n+1 & \text{if } \phi_n(n) \uparrow \end{cases}$$

so that $g_\mu$ is not computable.

We will also consider a localized version of granularity, $g_{x,\mu}$, which we will use in Proposition 5.2.5.

**Definition 3.2.16.** We define the local granularity of a measure $\mu$, $g_{x,\mu}$, by $g_{x,\mu}(n) =$ the least $m$ such that $\mu(x \upharpoonright m) < 2^{-n}$.

It is the local version of granularity since we clearly have that for all $n$:

$$g_\mu(n) = \max\{g_{x,\mu}(n) | x \text{ is a string with } |x| = n\}.$$

### 3.2.2 KFT and MET for computable measures on $\mathfrak{C}_T$

Here we enumerate notation for potential generalizations of KFT and the MET.

**Definition 3.2.17.** Let $\mu$ be a computable measure. If $\vec{W} = (r_i, \zeta_i)_{i\in\mathbb{N}}$ is a bounded request set we will use $\mathrm{KFT}(\mu, \vec{W})$ to denote the assertion:

$$\text{there exists a prefix-free sequence } \{\sigma_i\}_{i\in\mathbb{N}} \text{ such that for all } i,\ -\log \mu\sigma_i = r_i.$$

We will use $\mathrm{KFT}(\mu)$ to denote the statement:

$$\text{for all bounded request sets, } \vec{W},\ \mathrm{KFT}(\mu, \vec{W}).$$

There is a reasonable argument that this is not an appropriate attempt at a generalization: applications of KFT in the standard setting go through when the descriptions, $\sigma_i$, are allowed to be longer than $r_i$ by a fixed constant $b$. Indeed the entire concept of complexity is about growth rates, and not specific values. This suggests an attempt at a somewhat weaker generalization.

**Definition 3.2.18.** Let $\mu$ be a computable measure and $b$ a non-negative integer. If $\vec{W} = (r_i, \zeta_i)_{i\in\mathbb{N}}$ is a bounded request set we will use $\mathrm{KFT}^*(\mu, \vec{W}, b)$ to denote the assertion:

$$\text{there exists a prefix-free sequence } \{\sigma_i\}_{i\in\mathbb{N}} \text{ such that for all } i,\ -\lceil\log \mu\sigma_i\rceil \leq r_i + b.$$

We will use $\mathrm{KFT}^*(\mu)$ to denote the statement:

$$\text{for all bounded request sets, } \vec{W} \text{ for some } b,\ \mathrm{KFT}^*(\mu, \vec{W}, b).$$

Similarly, we will give two potential versions of a generaliztion of the MET. The effectiveness and uniformity in the index, $e$ of the bounded request set is as in the statement of Theorem 2.2.19, so we omit the parenthetical explanations given there.

**Definition 3.2.19.** Let $\mu$ be a computable measure, let $\vec{W} = \{< r_i, \zeta_i >\}_{i\in\mathbb{N}}$ be a bounded request set, and let $e$ be an index of $\vec{W}$. We use $\mathrm{MET}(\mu, \vec{W})$ to denote the following assertion:
effectively and uniformly in $e$, we can find a c.e. prefix-free code, $\{\sigma_i\}_{i\in\mathbb{N}}$ and a prefix-free index, $d(e)$ of the prefix-free machine $M = \sigma_i.\zeta_i$ such that

$$\text{for all } i,\ -\log \mu\sigma_i = r_i.$$

By $MET(\mu)$, we mean the assertion that for all $\vec{W}:\ MET(\mu, \vec{W})$ holds.

**Definition 3.2.20.** Similarly, let $\mu$ be a computable measure, let $\vec{W} = \{< r_i, \zeta_i >\}_{i\in\mathbb{N}}$ be a bounded request set, let $b \in \mathbb{N}$, and let $e$ be an index of $\vec{W}$. We use $\mathrm{MET}^*(\mu, \vec{W}, b)$ to denote the assertion which differs from $\mathrm{MET}(\mu, \vec{W})$ only in that the final displayed formula is changed to what follows:

$$\text{for all } i,\ -\lceil \log \mu\sigma_i \rceil \leq r_i + b.$$

By $\mathrm{MET}^*(\mu)$, we mean the assertion that for all $\vec{W}$ there exists $b \in \mathbb{N}$ such that $\mathrm{MET}^*(\mu, \vec{W}, b)$ holds.

### 3.2.3 Randomness in computable measure spaces and $K_{\mathbf{R}}$-triviality

Various generalizations of Martin-Löf randomness were proposed, all in the natural setting of computable measures, since they all involved, in one form or another, some

sort of generalization of ML-test, with its underlying measure-theoretic character. Attesting to the robustness of the notion, they all turned out to be equivalent in that, for each computable measure, the different approaches yielded the same set of Martin-Löf random elements of the ambient space. We briefly mention a few of the more prominent approaches, but omit the details, here, since, except as explicitly noted below, they do not directly impact our work.

One approach was taken, variously, in [1], [4], and [19] and in Chapter 6 we will discuss some questions related to this approach. Another approach was taken by Gács [5] and Hoyrup and Rojas [9], and their approach led to the following generalization of Schnorr's Theorem. We omit the detailed development of their notion of *computable measure space* which dif and only ifers in some details from the above account of their approach to computable measures on computable metric spaces.

**Proposition 3.2.21.** [9] If $(X, \mu)$ is a computable measure space and $x \in X$, then $x$ is $\mu$-Martin-Löf random if and only if

$$K(\rho(x) \upharpoonright n) \geq^{+} -\log \mu(\rho(x) \upharpoonright n).$$

□

Essentially, this replaces $n$, in Proposition 2.3.25, by $-\log \mu(\rho(x) \upharpoonright n)$. It is then reasonable to wonder if the same type of replacement is sensible for $K$-triviality.
Later still, Rute, [21], developed a approach to generalizing the notion of Martin-Löf randomness that was even closer to the spirit of the GHR development, sketched above, of the theory of computable measures on computable metric spaces, and proved the equivalence of his approach to the others. What will be important for us in what follows is that the previous Proposition made it natural for him to consider formulating a generalization of the $K$-triviality notion in the language of computable measures. This led him to suggest the notion of $K_{\mathrm{R}}$-triviality which we now present.

**Definition 3.2.22.** Let $\mathcal{X}$ be a computable metric space, let $\mu$ be a computable measure on $\mathcal{X}$, and let $\mathcal{A}$ be a generator for $(\mathcal{X}, \mu)$. We say $x \in X$ is $K_{\mathrm{R}}(X, \mu, \mathcal{A})$-trivial if

$$K(x \upharpoonright_{\mathcal{A}} n) \leq^{+} K(-\lceil \log \mu(x \upharpoonright_{\mathcal{A}} n) \rceil)$$

We let $KT_{\mathrm{R}}$ denote the set of $x \in X$ such that $x$ is $K_{\mathrm{R}}$-trivial.

By analogy with Definition 2.3.26, we have the notion of *infinitely often* $K_{\mathrm{R}}(X, \mu, \mathcal{A})$-trivial; we omit the obvious details. As before, we will omit sometimes omit $\mu$ and even more frequently omit $\mathcal{A}$ when they are clear from context.

In the case of Cantor space, this new definition agrees with the standard formulation $K(x \upharpoonright n) \leq^{+} K(n)$ since $-\log \mu(x \upharpoonright n) = -\log 2^{-n} = n$. The idea is that, in the general setting the negative logarithm of the measure is the analogue of length in Cantor space, and it seems quite natural, especially given its aptness in the previous Proposition. As already noted in Chapter 1, looking more deeply into the notion of $K_{\mathrm{R}}$-triviality and comparing it with Definition 3.1.4 was the initial motivation for the work of this dissertation. This sets the stage for the next two Chapters, where we will carry this out.

# Chapter 4

# Computable Measures: The Good, The Bad, and the Atomic

## Chapter Overview

In this Chapter, we present our results on computable measures on $\mathfrak{C}_T$. In Section 4.1, we introduce the notion of coarseness, a notion dual to granularity, present some basic facts, and prove two basic results (Proposition 4.1.3 and Theorem 4.1.5) about coarseness for computable measures. In subsection 4.1.2, we invoke a "tameness" hypothesis on the coarseness to prove, Theorem 4.1.6, that the generalization of the MET introduced in Defintion 3.2.20 holds for computable measures satisfying this hypothesis. A counterpoint is provided in Section 4.2, where we explore the difficulties in establishing generalizations of the MET, and even of KFT itself in the absence of additional hypotheses on the computable measure. In Section 4.3, we explore some additional pathologies that arise for computable measures on $\mathfrak{C}_T$, especially regarding Rute's suggested generalization of $K$-triviality, Definition 3.2.22.

## 4.1 Coarseness

In 3.2.14, we defined the notion of granularity . Here, we define the dual notion of coarseness. In order to see that this notion is well defined, note that if $m > 0$, then all strings of length at most $n$ have $\mu$-measure greater than $2^{-m}$ and that some string of length $m$ must have $\mu$-measure no greater than $2^{-m}$.

**Definition 4.1.1.** If $\mu$ is a computable measure (on $\mathfrak{C}_T$), we define $c_\mu$, the *coarseness* of $\mu$, by setting $c_\mu(0) = 0$, and for $m > 0$, taking $c_\mu(m)$ to be the largest $n$ such that all strings of length at most $n$ have $\mu$-measure greater than $2^{-m}$ (and 0 if no such $m$ exists).

The next remark records a few obvious, but nevertheless helpful facts about coarseness.

**Remark 4.1.2.** For all measures, $\mu$:

1. for all $n > 0,\ c_\mu(n) < n$,
2. for all $n > 0,\ c_\mu(n)$ is the least $m < n$ such that some string of length $m + 1$ has $\mu$-measure at most $2^{-n}$.

In analogy to the situation for granularity, Lebesgue measure has the largest possible coarseness function: $c_\lambda(0) = 0$ and for $n > 0,\ c_\lambda(n) = n-1$. More irregular measures generally have slower growing coarseness. For example, the measure, $\nu$, constructed in Proposition 4.2.1 has logarithmic coarseness.

### 4.1.1 Basic Results About Coarseness

The notion of coarseness will figure in several of our results about computable measures. In analogy with Proposition 3.2.15 (and building on its proof), the next Proposition shows that if $\mu$ has no null cells then there is always a (nontrivial) computable lower bound on $c_\mu$ (and as a corollary to the proof, that if $\mu$ is exactly computable, then $c_\mu$ itself is computable). In Theorem 4.1.5, we obtain an

analogue of another result from [8]: there is a computable (but necessarily not exactly computable) $\mu$ such that $c_\mu$ is not computable. In subsection 4.1.2, we show, Theorem 4.1.6, that if $c_\mu$ has the asymptotically fastest possible growth rate, then $\mathrm{MET}^*(\mu)$ holds in a strong way: there is a single $b$ such that for bounded request sets, $\vec{W}$, $\mathrm{MET}^*(\mu, \vec{W}, b)$ holds.

**Proposition 4.1.3.** If $\mu$ is a continuous computable measure with no null cells, then $c_\mu$ has a computable lower bound such that

$$\lim_{m\to\infty} c_\mu(m) = \infty.$$

*Proof.* We proceed much as in the proof of Proposition 3.2.15; in particular, we adopt the same notation $\mu_j$ as there and we take $m, n, \{\sigma_i\}$, the total computable function $s$, the computable predicate $P$ and $g^*_\mu$ to be as in that proof. We define the computable predicate $Q(m, n)$ by taking it to hold if and only if for all $i$, $\mu_{s(m,n)}(\sigma_i) > 2^{-m}$.

If $m = 0$, we simply take $c^*_\mu(m) := 0$, so assume that $m > 0$. Note that this clearly implies that $Q(m, 0)$ holds. Also note that $Q(m, g^*_\mu(m))$ fails and that if $n_1 < n_2$ and $Q(m, n_2)$ holds, then so does $Q(m, n_1)$. Further, appealing to the definition of $s$, it follows that $n_2 \leq c_\mu(m)$. So, we take

$$c^*_\mu(m) := \text{the largest } n \text{ such that } Q(m, n) \text{ holds},$$

and then $c^*_\mu$ is a computable lower bound for $c_\mu$, as required. It follows easily from the continuity of $\mu$ that $\lim_{m\to\infty} c_\mu(m) = \infty$.

If $\mu$ is exactly computable, then for all $s$, $\mu_s = \mu$, and then the construction of $c^*_\mu$ simply yields $c_\mu$, which is, therefore, computable. □

**Remark 4.1.4.** The limit condition is included because $i.0$ is a trivial lower bound.

We now construct a computable (but necessarily not exactly computable) measure (necessarily without null cells) whose coarseness function is not computable. Our construction is similar to that of [8], where the analogous result for granularity was proved.

**Theorem 4.1.5.** There exists a computable measure $\mu$ (on $\mathcal{C}_T$) such that $c_\mu$ is not computable.

*Proof.* The idea is to construct a measure so that at level $k+2$, only one string can possibly have measure as small as $2^{-(2k+3)}$. This string will also be able to have measure larger than $2^{-(2k+3)}$. In this way, the coarseness will depend solely on what we do with this string. We will use this to force

$$c_\mu(2k+3) = \begin{cases} k+2, & \text{if } \mathbb{U}(k)\downarrow \\ k+1, & \text{if } \mathbb{U}(k)\uparrow . \end{cases} \tag{4.1}$$

This will complete the proof, since then we will have shown that $c_\mu$ encodes the Halting Problem. We will define the computable function $\tilde{\mu}$ required by Definition 3.2.13. As in the proof of Proposition 3.2.15, for natural numbers, $s$ and strings, $\omega$, we use $\mu_s(\omega)$ in place of $\tilde{\mu}(s,\omega)$. We will define the $\mu_s$ by recursion on $s$. Though this is not required by our Definition 3.2.13 (it is required in some equivalent definitions), we will construct the $\mu_s$ so that for each string $\omega$, the sequence $\{\mu_s(\omega)\}_{s\in\mathbb{N}}$ will be monotone non-decreasing; thus, our $\mu$ will, in fact, be lower semicomputable.

We begin by defining the "backbone" of our measure. On the backbone, $\tilde{\mu}$ will be constant in $s$, so what follows constitutes the basis of our recursive definition of the $\mu_s$ as well. For all $s$ and for $\star = 0$ or $\star = 1$:

$$\mu_s(\epsilon) = \mu(\epsilon) := 1 \text{ and for } k \geq 0,\ \mu_s\left(1^k\star\right) = \mu\left(1^k\star\right) := 2^{-(k+1)}.$$

For the one-bit extensions, $1^k0\star$, of the $1^k0$, this will no longer be true. The definition of $\mu_s$ for these strings will involve a genuine recursion on $s$. In view of their important role in what follows, we introduce special notation for these strings: we will use $\tau_k$ to denote the string $1^k0$, and refer to such strings as *backbone strings*.

We will refer to the one-bit extensions of backbone strings, the $\tau_k\star$, as *special strings*.

We will define the $\mu_s(\tau_k\star)$ according to whether or not $\mathbb{U}_s(k)\downarrow$, and we will do this in such a way that $c_\mu(\tau_k 0)$ will encode whether or not $\mathbb{U}(k)\downarrow$. If this is achieved, then $c_\mu$ will encode the Halting Problem and so will not be computable.
Before carrying out the recursion on $s$ that will define the $\mu_s(\tau_k\star)$, we mention that the situation above the special strings will be very simple: for all $s$ and for any string, $\eta$, of positive length, we will define:

$$\mu_s(\tau_k \star \eta) := \mu_s(\tau_k\star)\cdot 2^{-|\eta|}, \tag{4.2}$$

i.e, above the special strings, each $\mu_s$ "distributes the measure evenly." Since, for every string $\omega$ we will have that $\mu(\omega) = \sup_s \mu_s(\omega)$, the previous equation also holds for $\mu$ in place of $\mu_s$: for any string $\eta$, of positive length, we will have that:

$$\mu(\tau_k \star \eta) := \mu(\tau_k\star)\cdot 2^{-|\eta|}, \tag{4.3}$$

Thus, while there is formally a recursion on $s$ involved in the definition of $\mu_s$ for (non-trivial) extensions of special strings, the role of the recursion is limited to its role in determining the value of $\mu_s$ for its special prefix. It is an immediate consequence of the previous equation that if $\tau$ is a special string and $\tau \preceq \omega$ then, for $\star = 0$ or $\star = 1$:

$$\mu(\omega\star) = \frac{1}{2}\mu(\omega), \tag{4.4}$$

so that it is also true that above special strings, $\mu$ "distributes the measure evenly."

The last two equations will play a key role in verifying an important property of the construction. We will have that for all $k$:

$$\text{for all } \sigma \text{ with } |\sigma| = k+2,\ \mu(\sigma) \geq 2^{-(2k+3)}, \tag{4.5}$$

with equality only possibly holding if $\sigma = \tau_k 0$.

Once we have given the details of the recursive definition of the $\mu_s$, we will prove, by induction on $k$, that this equation holds. Once we do this, we will satisfy 4.1 as desired.

With all of this in place, we proceed to the actual definition by recursion on $s$ of $\mu_s$ for the special strings. For $s = 0$, we define:

$$\mu_0(\tau_k 0) = 2^{-(2k+4)},\ \mu_0(\tau_k 1) = 2^{-(k+1)} - 2^{-(2k+2)} + 2^{-(2k+4)}. \tag{4.6}$$

If $s > 0$, we define:

$$\mu_s(\tau_k 1) = \begin{cases} 2^{-(k+1)} - 2^{-(2k+2)} + \sum_{i=1}^{s+1} 2^{-(2k+3+i)}, & \text{if } \mathbb{U}_s(k)\uparrow \\ \mu_{s-1}(\tau_k 1), & \text{if } \mathbb{U}_s(k)\downarrow \end{cases} \tag{4.7}$$

Finally, when $s > 0$, in the "$\star = 0$ case", we define:

$$\mu_s(\tau_k 0) = \begin{cases} \sum_{i=1}^{s+1} 2^{-(2k+3+i)}, & \text{if } \mathbb{U}_s(k)\uparrow \\ \mu_{s-1}(\tau_k 0), & \text{if } \mathbb{U}_{s-1}(k)\downarrow \\ 2^{-(k+1)} - \mu_s(\tau_k 1), & \text{if } \mathbb{U}_s(k)\downarrow \text{ but } \mathbb{U}_{s-1}(k)\uparrow. \end{cases} \tag{4.8}$$

This completes the recursive definition. As already pointed out, in Equation 4.2, we have "predefined" all of the $\mu_s$ on all of the extensions of the special strings, so this also completes the definition of the $\mu_s$. We now begin the inductive proof of Equation 4.5.

Assume that Equation 4.5 holds for all strings of length $k+2$. Let $\sigma$ be such that $|\sigma| = k+3$.

1. Case $\sigma = \tau_{k+1}1$: It is clear by construction that $\mu(\sigma) > 2^{-(k+2)} - 2^{-(2k+4)} > 2^{-(2k+5)}$.

2. Case $\sigma = \tau_{k+1}0$: If $\mathbb{U}(k+1)$ ever halts, we know that $\mu(\sigma) = 2^{-(2k+4)} - \sum_{i=1}^{s'+1} 2^{-(2k+5+i)}$, where $s'$ is the least $s$ such that $\mathbb{U}_s(k)\downarrow$. This is greater

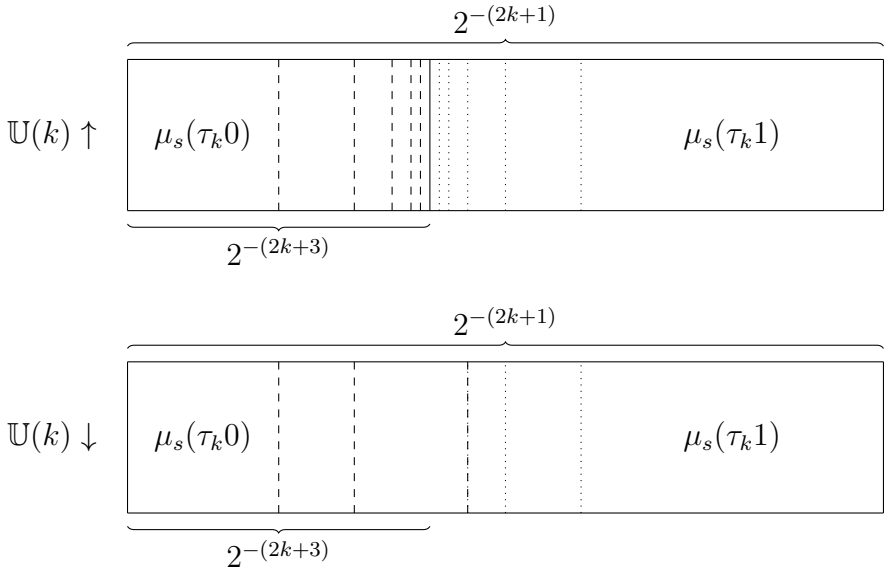


**Figure 4.1:** $\mu_s$ on the special points $\tau_k 0$ and $\tau_k 1$

than $2^{-(2k+5)}$ as desired. In the case that $\mathbb{U}(k+1)$ never halts, we instead see $\lim_{s\to\infty} \mu_s(\sigma) = 2^{-(2k+5)} = \mu(\sigma)$ - this is the only way that equality in 4.5 can hold.

3. Case $\sigma$ is not a special string: By the induction hypothesis, $\mu(\sigma \upharpoonright (k+2)) \geq 2^{-(2k+3)}$, and $\mu(\sigma) = \frac{1}{2}\mu(\sigma \upharpoonright (k+2))$, so it follows that $\mu(\sigma) > 2^{-(2k+5)}$.

Hence, 4.5 holds, and we can see that

$$c_\mu(2k+3) = \begin{cases} k+2 & \text{if } \mathbb{U}(k)\downarrow \\ k+1 & \text{if } \mathbb{U}(k)\uparrow \end{cases}$$

and so, it is not computable.

□

### 4.1.2 MET for Tame Measures

The MET plays such an important role in the standard setting that it is natural to investigate to what it extent it holds for computable measures (on $\mathfrak{C}_T$). While the

results of the next Section place important limitations on the range of measures for which it can hold, our next result gives a characterization, in terms of coarseness, of a family of computable measures $\mu$, for which $\mathrm{MET}^*(\mu)$ will hold in a strong fashion.

**Theorem 4.1.6.** If $\mu$ is a computable measure (on $\mathfrak{C}_T$) and $b$ is a positive integer for which $c_\mu(n) > n - b$, then for all bounded request sets, $\vec{W}$, $\mathrm{MET}^*\left(\mu, \vec{W}, b\right)$ holds.

*Proof.* We follow the proof given in [18] and there are strong analogies to the proof of Theorem 2.2.17, above. Once again, we work by recursion on $i$, but we can no longer asssume that $r_i \geq r_j$ whenever $j < i$. This leads us to consider a more complicated set, $R_{i-1}$ of "candidates" to be $\sigma_i$ or a prefix of $\sigma_i$. In the proof of Theorem 2.2.17, the analogue of $R_{i-1}$ was implicitly just the set of strings of length $r_i$ which were not in $U_i$.

We begin by conventionally taking $R_{-1}$ to be $\{\epsilon\}$. The recursion then proceeds as follows. At stage $i \geq 0$, we will assume we have defined $R_{i-1}$ and the $\sigma_j$ for $0 \leq j < i$, with the properties we shall give shortly, and we define $\sigma_i$ and $R_i$ in such a way as to preserve the properties assumed for the construction prior to stage $i$. We let $r_i^* := \max_{j \leq i} r_j$, and note, once again, that possibly $r_i < r^*$. We also let $A_i := R_{i-1} \cup \{\sigma_j | 0 \leq j < i\}$ and $R'_{i-1} := \{\tau \in R_{i-1} | \, |\tau| \leq r_i\}$.

1. $A_i$ is a prefix-free set of strings, all of length at most $r_i^*$,
2. distinct elements of $R_{i-1}$ have different lengths,
3. $R'_{i-1}$ is non-empty,
4. Every string of length $r_i^*$ is either an element of $A_i$ or has an element of $A_i$ as prefix.

Note that by properties (1) and (4) we easily have that:

$$\sum_{0\leq j<i} 2^{-r_j} + \sum_{\tau\in R_{i-1}} 2^{-|\tau|} = 1. \tag{4.9}$$

To define $\sigma_i$, let $z_i$ be that element of $R'_{i-1}$ of the largest length (of elements of $R'_{i-1}$). Clearly $z_i$ is well-defined by properties (2) and (3). Let $m_i := r_i - |z_i|$. We take $\sigma_i := z_i 0^m$ (so that if $m = 0, \sigma_i = z_i$). Finally, we set $R_i := R_{i-1}\backslash\{z_i\} \cup \{z_i 0^\ell 1 | 0 \leq \ell < m\}$. Note that if $\sigma_i = z_i$, then there are no new strings in $R_i$; we have just removed $z_i$.

We next argue that we have preserved properties (1) - (4). Of course $r^*_{i+1} = \max(r^*_i, r_{i+1})$, and properties (1), (4) for $A_{i+1}$ are clear by construction and the fact that these properties hold $A_i$. It is also clear by construction (especially the choice of $z_i$) that property (2) holds for $A_{i+1}$. Thus, it remains to verify that we have property (3) for $A_{i+1}$. As above, it follows from properties (1), (4) for $A_{i+1}$ that:

$$1 = \sum_{0\leq j\leq i} 2^{-r_j} + \sum_{\tau\in R_i} 2^{-|\tau|} = \sum_{0\leq j\leq i} 2^{-r_j} + \sum_{\tau\in R'_i} 2^{-|\tau|} + \sum_{\tau\in R_i\backslash R'_i} 2^{-|\tau|}. \tag{4.10}$$

It also follows from construction and property (2) for $A_{i+1}$ that $\sum_{\tau\in R_i\backslash R'_i} 2^{-|\tau|} < 2^{-r_{i+1}}$, and therefore, it is impossible for $R'_i$ to be empty, so we have also preserved property (3).

This completes the construction of the prefix-free code $\{\sigma_i\}_{i\in\mathbb{N}}$, which, we note was carried out effectively in the sequence of lengths $\{r_i\}_{i\in\mathbb{N}}$. The index $e$ provides us "effective access" to this sequence and so to the construction of $\{\sigma_i\}_{i\in\mathbb{N}}$. Thus, we have outlined an effective procedure to obtain $\sigma_i$ from $i$ and $e$. Since the set $\{\sigma_i\}_{i\in I}$ is prefix-free, it can also be effectively inverted to find $i$ from $\sigma_i$. We can then appeal to $e$ again, to obtain $\zeta_i$ effectively from $\sigma_i$, which gives us our desired prefix-free"machine" $M$. Since this was done uniformly and effectively in $e$, by the $s-m-n$ Theorem, starting from $e$ we effectively obtain $d$, a prefix-free index for this "machine" $M$ with $M(\sigma_i) = \zeta_i$.

It remains to verify that for all $i$, $-\lceil \log \mu\sigma_i \rceil \leq r_i + b$, as required for the definition of $\mathrm{MET}^*(\mu, \vec{W}, b)$, cf. the second displayed formula of Definition 3.2.20. By hypothesis, $c_\mu(r_i + b) > r_i$. Since $|\sigma_i| = r_i$, it follows that $\mu(\sigma_i) > 2^{-(r_i+b)}$, and hence $-\lceil \log \mu\sigma_i \rceil \leq r_i + b$, as required. □

If we sought to weaken the coarseness hypothesis to be only that $c_\mu(n) > an - b$ where $a < 1$, the proof given above would only guarantee $\mu\sigma_i > 2^{-\frac{r_i+b}{a}}$, so the descriptions could be smaller than desired by up to a factor of $\frac{1}{a}$, which is not good enough for our purposes. However, if we tighten the weight condition from $\sum_{n=1}^{\infty} 2^{-r_i} \leq 1$ to $\sum_{n=1}^{\infty} 2^{-\lceil ar_i \rceil} \leq 1$, and this would allow us to adapt the proof so that it still works - we define a new request sequence $r_i^* = \lceil ar_i \rceil$, and choose $\sigma_i$ for this new request sequence as above. Now, $c_\mu(n) > an - b$ gives us $\mu\sigma_i > 2^{-\frac{\lceil ar_i \rceil + b}{a}} > 2^{-(r_i + 1 + \frac{b}{a})}$, so $-\lceil \log \mu\sigma_i \rceil \leq r_i + b^*$, where $b^* = 1 + \frac{b}{a}$.

## 4.2 Negative Results for Computable Measures on $\mathfrak{C}_T$

### 4.2.1 KFT may fail

In Chapter 2, we introduced KFT as a partial result on the path to the proof of the MET, which is an important tool in many proofs related to $K$-Triviality in the standard setting. We will consider several attempts to generalize this theorem.

For a computable measure $\mu$, $\mathrm{KFT}(\mu)$ as given in 3.2.17 is the simplest and most direct attempt at a generalization, since KFT in the standard setting is the assertion that $\mathrm{KFT}(\lambda)$ holds. Unfortunately, it is not difficult to find computable measures, $\mu$, for which $\mathrm{KFT}(\mu)$ fails, e.g., the following. Let $\mu$ be any computable measure such that:

$$\mu(0) = .75, \quad \mu(1) = .25, \quad \mu(00) = .5$$

and let $\vec{W}$ be any bounded request set such that $r_0 = r_1 = 1$. Clearly there is no way to find a prefix free code $\{\sigma_i\}_{i\in\mathbb{N}}$ with the desired measures.

It is reasonable to take the ease with which counterexamples to KFT($\mu$) can be found as further evidence that, as argued following Definition 3.2.17, despite its simplicity, KFT($\mu$) is not an appropriate attempt at a generalization.

**Theorem 4.2.1.** There exists a computable measure $\nu$ and bounded request set $\vec{W}$ such that for all b $\mathrm{KFT}^*(\nu, \vec{W}, b)$ fails.

*Proof.* We explicitly construct such a measure, $\nu$ and bounded request set $\vec{W}$, working by recursion on stages, $i$. We will define $\nu$ specifically so that after satisfying a certain number of requests, the only cells that remain will have measure too small to satisfy the remaining requests. We will let $\nu^{(i)}$ denote the partial sub-measure of $\nu$ consisting of those strings whose measure has been defined by or before stage $i$, and we will let $A_i$ denote the set of strings whose measure has been defined at stage $i$, but not earlier. Thus, the domain of $\nu^{(i)}$ is $\bigcup_{i'\leq i} A_{i'}$. We will have that the domain of $\nu^{(i)}$ is prefix-closed, i.e. if $\nu^{(i)}(\tau)$ is defined and $\sigma$ is a prefix of $\tau$, then $\nu^{(i)}(\sigma)$ is also defined. We will have that $A_i$ is finite, and we let $A_i^*$ be the set of terminal strings of $A_i$, i.e., $\sigma \in A_i^*$ if and only if $\sigma \in A_i$ and for no proper extension, $\tau$, of $\sigma$ do we have $\tau \in A_i$. Thus, for any string $\tau$, $\nu^{(i)}(\tau)$ is undefined if and only if there is a proper prefix of $\tau$ which is in $A_i^*$. We will construct $\nu$ so as to be *monotonic*, i.e. for any strings $\sigma, \tau$, if $\sigma$ is a proper prefix of $\tau$, then $\nu(\tau) < \nu(\sigma)$.

In defining $\vec{W}$, we will simply take $r_i = i$, and for simplicity and concreteness we will take $\zeta_i = \epsilon$ for all $i$, though any reasonable choice of the $\zeta_i$ would also work. For $\sigma \in A_i$, we define $\nu^{(i)}(\sigma)$ so as to guarantee that $\mathrm{KFT}^*(\nu, \vec{W}, i)$ will fail, or, as we will say, "to defeat $b = i$".

Before undertaking the construction of the $\nu^{(i)}$, we introduce some auxiliary notions. The first is a pair of integer sequences, $\{j_i\}$ and $\{k_i\}$ defined by

$$j_0 = k_0 = 0,\ j_1 = 1, \text{ and for } i > 0,\ k_i = j_i + i + 2,\ j_{i+1} = 2^{k_i}.$$

The $k_i$ will figure in the definition of $\nu^{(i)}(\sigma)$ for $\sigma \in A_i$, telling us how many/how small the divisions in $\nu^{(i)}$ will need to be in order to defeat $b = i$.
For positive integers, $k$, let let $C_k$ be the "$k$-comb": $C_k = C_{k.0} \cup C_{k,1}$, where $C_{k,0} = \left\{0^\ell | \ell = 0, 1, \ldots, 2^k - 1\right\}$ and $C_{k,1} = \left\{\tau 1 | \tau \in C_{k,0},\ \tau \neq 0^{2^k-1}\right\}$. We also let $T_i := C_{k_i,1} \cup \left\{0^{2^k-1}\right\}$, and $C'_k := C_k \setminus \{\epsilon\}$.

We also have the partial measure, $M_k$ defined on $C_k$ by:

$$M_k\left(0^\ell\right) = 1 - \frac{\ell}{2^k} \text{ and for } \ell < 2^k - 1,\ M_k\left(0^\ell 1\right) = M_k\left(0^\ell\right) - M_k\left(0^{\ell+1}\right).$$

Note that:

$$\text{for } \sigma \in T_i,\ M_{k_i}(\sigma) = 2^{-k_i}. \tag{4.11}$$

Recall that for sets, $X,\ Y$ of strings, we use $XY$ to denote $\{\sigma\tau | \sigma \in X,\ \tau \in Y\}$. Everything is now in place for the definition of the $A_i$ and $\nu^{(i)}$:

$$\text{we set } A_0 := \{\epsilon\} \text{ and } \nu^{(0)}(\epsilon) := 1. \tag{4.12}$$

and having defined $A_i,\ \nu^{(i)}$ we set:

$$A_{i+1} := A_i^* C'_{k_{i+1}}; \text{ for } \sigma \in A_i^*,\ \tau \in C'_{k_{i+1}},\ \nu^{(i+1)}(\sigma\tau) := \nu^{(i)}(\sigma) M_{k_{i+1}}(\tau). \tag{4.13}$$

This completes the construction of $\nu$. It is clear from this construction that, as promised, $\nu$ is monotonic.

Let $k_i^*$ denote $\sum_{i'=0}^{i} k_{i'}$. Note that it follows easily from 4.11 (and a small inductive argument) that:

$$\text{for } \eta \in A_i^*,\ \nu(\eta) = 2^{-k_i^*} = \left(\prod_{i'=1}^{i+1} j_{i'}\right)^{-1}. \tag{4.14}$$

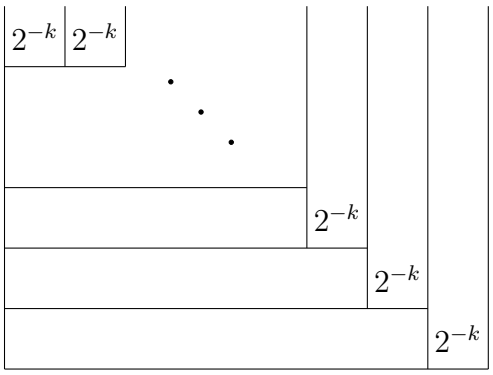


**Figure 4.2:** The partial measure $M_k$

As an easy consequence of Equation 4.14 and the monotonicity of $\nu$, we have:

$$\text{for } \eta \in A_{i+1},\ \nu(\eta) < 2^{-k_i^*} = \left(\prod_{i'=1}^{i+1} j_{i'}\right)^{-1}, \tag{4.15}$$

and

$$\text{for } \eta \notin A_i C_{k_{i+1}-1,0},\ \nu(\eta) \leq 2^{-k_i^*} = \left(\prod_{i'=1}^{i+2} j_{i'}\right)^{-1}. \tag{4.16}$$

A few more simply observations will make it straightforward to verify that our construction of $\nu^{(i)}$ has indeed defeated "$b = k_i$".

For all $i$,

$$\left|A_{i+1}^*\right| = 2^{k_{i+1}} \left|A_i^*\right|, |A_{i+1}| = 2\left|A_{i+1}^*\right| - 1, \text{ therefore:}$$

For all $i$,

$$\left|A_{i+1}^*\right| = 2^{\sum_{i'=0}^{i} k_{i'}} = 2^{k_i^*},$$

Suppose that $X$ is a prefix-free subset of $(A_0 \cup \cdots \cup A_i)\, C_{i-1,0}$. Then $|X| \leq 2^{k_i^*}$.

Now suppose $b = i$. At this point in our construction, there are $k_i^*$ cells with measure at least $2^{-k_i^*}$. Suppose we are able to find valid assignments for $\sigma_1, \cdots, \sigma_{k_i^*}$. For all $i \leq k_i^*$, we must have $\sigma_i \in A_i C_{k_{i+1}-1,0}$, since any other cells have insufficient

measure - less than $2^{-(k_i^*+b+2)}$. However, after assigning $\sigma_1, \cdots, \sigma_{k_i^*}$, necessarily all cells with measure greater than $2^{-(k_i^*+b+2)}$ were used, leaving no compatible choice for $\sigma_{k_i^*+1}$. □

### 4.2.2 KFT can hold when MET fails

Next we will show that there exists a computable measure, $\nu$, on $\mathcal{C}_T$, and a bounded request set, $\vec{W}$, for which KFT$(\nu, \vec{W})$ holds but MET$(\nu, \vec{W})$, the strongest possible generalization of the Machine Existence Theorem, in which requests must be hit exactly, fails. First, however, we define an auxiliary computable measure $\mu$.

**Definition 4.2.2.** We define the computable measure $\mu$ which will be helpful in proving Proposition.

$$\begin{array}{cc} \mu(0) = .5 & \mu(1) = .5 \\ \mu(00) = .25 & \mu(01) = .25 \\ \mu(10) = .125 & \mu(11) = .375 \\ \mu(110) = .1875 & \mu(111) = .1875 \end{array}$$

The following figure and discussion will illustrate the role that $\mu$ will play in the proof of Proposition .

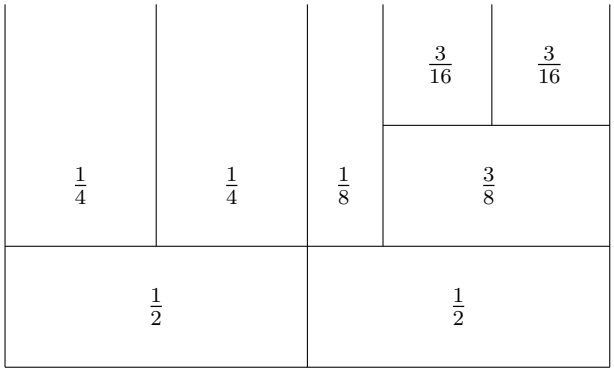


**Figure 4.3:** The basis for conflicts in choosing $\sigma_i$

The idea here is that we are forced to make a choice up front that will cut off either our ability to give one large measure description, or two medium measure descriptions - and we cannot know at the time which is correct. Suppose we are given a request $B$ set with $r_1$=1. There are potential values for $\sigma_1$ to realize the request set: 0 or 1. However, if we choose $\sigma_1 = 0$, and later on see $r_i = 2$ for some value of $i$, we cannot satisfy that request Likewise if we choose $\sigma_1 = 1$, and later on see $r_i = -\log(.375)$. We cannot look ahead to see which choice is correct, as it is possible neither value actually occurs, so any time we make such an assignment (and we must eventually), we must acknowledge the possibility of conflict with future values (possibly infinitely often), and so give up on effectively being able to find an index. This does not rule out the possibility of finding such an index semi-computably.

**Proposition 4.2.3.** There exist a computable measure, $\nu$, on $\mathcal{C}_T$, and bounded request set, $\vec{W}$, such that KFT$(\nu, \vec{W})$ holds, but MET$(\nu, \vec{W})$ does not.

*Proof.* Let the measure $\mu$ be defined as above, with the measure of extensions of these strings being split evenly, i.e. $\mu(\tau 0) = \mu(\tau 1)$ for all $\tau \neq 0, 1,$ or 11 . Now, using $\mu$ to assist us, we define the desired $\nu$ and request set $\vec{W}$. Let $p_i$ denote the $i^{\text{th}}$ prime number, and $p$ denote $\sum_{i=0}^{\infty} \frac{1}{p_i^2}$. We use $p_i$ to make sure requests can only be satisfied in specific ways.

For $i \geq 0$, define:

$$\nu(1^i 0 \tau) = \mu(\tau) * \frac{p_i^{-2}}{p}.$$

Let $(\vec{\sigma}^j | j \in \mathbb{N})$ be an enumeration without repetitions of all finite prefix-free codes, $\vec{\sigma}$, and let $\vec{\sigma}^j = \left\{\sigma_k^j\right\}_{k \in [\ell_j]}$. Remember that $\sigma_k^j$ will be the $k^{\text{th}}$ string in the prefix-free code $\vec{\sigma}^j$, not the $k^{\text{th}}$ bit of the string $\sigma^i$. We will enumerate our request set while also simulating the standard universal machine $\mathbb{U}$, and whenever we see a computation halt, we will enter extra requests. At stage $i$, enter request $(-\log \frac{1}{2p_i^2}, 0)$ into $\vec{W}$ at the first unused index. At the same time, if for any $j \leq i$, we see that $\mathbb{U}_i(j)\downarrow$, let $j'$

be the index such that $r_{j'} = (-\log \frac{1}{2p_j^2}, 0)$. We consider cases, depending on whether or not $j' < \ell_j$, and if so, depending on the value of $\sigma^j_{j'}$.

First, if $j' \geq \ell_j$, we enter no additional requests, so assume that $j' < \ell_j$. Then, if $\sigma^j_{j'} \neq 1^j01, 1^j00$, we also enter no additional requests. If $\sigma^j_{j'} = 1^j01$, add a request of the form $(\frac{3}{8p_j^2}, 0)$ to $\vec{W}$. Finally, if $\sigma^j_{j'} = 1^j00$, add a request of the form $(\frac{1}{4p_j^2}, 0)$ to $\vec{W}$.

We argue that no computable prefix free code satisfies the request set $\vec{W}$, but KFT still holds. Note that:

$$\sigma_s = \begin{cases} 1^j00 & \text{if } r_s = (\frac{1}{2p_j^2}, 0) \text{ s.t } \ \phi_{e_j}(j') = 1^j01 \\ 1^j01 & \text{if } r_s = (\frac{1}{2p_j^2}, 0) \text{ s.t } \ \phi_{e_j}(j') \neq 1^j01 \\ 1^j011 & \text{if } r_s = (\frac{3}{8p_j^2}, 0) \\ 1^j001 & \text{if } r_s = (\frac{1}{4p_j^2}, 0) \end{cases}$$

realizes $\vec{W}$. However, it is not computable, since it computes the halting problem, as for all $j$, either $1^j00$ or $1^j01$ is in $\{\sigma_s\}_{s\in\mathbb{N}}$. If $1^j00 \in \{\sigma_s\}_{s\in\mathbb{N}}$, then $\mathbb{U}(j)\downarrow$, and if $1^j01 \in \{\sigma_s\}_{s\in\mathbb{N}}$, then $\mathbb{U}(j)\uparrow$.

□

Even with the weaker condition embodied in the displayed formula of Definition 3.2.20, there is still the potential for difficulties. While we do not show, here, that there exist $(\nu, \vec{W})$ and $b$ for which $\text{KFT}^*(\nu, \vec{W})$ holds but $\text{MET}^*(\mu, \vec{W}, b)$ fails, we will give an argument to show that at least one conflict, as above, can occur. This issue is discussed in more detail in Chapter 6.

Let $b$ be given. Now, let $\gamma$ be such that $2^{-(b+1)} = \frac{1}{2} - \gamma \cdot 2^{-(b+5)}$, $\delta$ be such that $2^{-(b+2)} + 2^{-(b+4)} = \frac{1}{2} - \delta \cdot 2^{-(b+5)}$, and let $\epsilon$ be such that $2^{-(b+2)} = \frac{1}{2} - \epsilon \cdot 2^{-(b+5)}$. Note $\gamma, \delta, \epsilon$ are integers. Define:

$$m_k(0) = \frac{1}{2}$$

$$m_k(1) = \frac{1}{2}$$

$$m_k(00^i) = \frac{1}{2} - i \cdot 2^{-(b+5)} \text{ for } i \leq \delta$$
$$m_k(00^\delta 0) = 2^{-(b+2)}$$
$$m_k(00^\delta 1) = 2^{-(b+4)}$$

$$m_k(10^i) = \frac{1}{2} - i \cdot 2^{-(b+5)} \text{ for } i \leq \epsilon$$
$$m_k(10^\delta 0) = 2^{-(b+3)}$$
$$m_k(10^\delta 1) = 2^{-(b+3)}$$

Then, make the request $r_1 = 1$. We must choose either $\sigma_1 = 00^\delta$ or $\sigma_1 = 10^\delta$ (choosing prefixes of these also satisfies $r_i \geq -\lceil \log m_k \sigma_i \rceil - b$, but exacerbates the problems that follow). In the latter case, if we later see two requests $r_i = 3$, $r_{i+j} = 3$, we will not be able to realize them. In the former case, if we later see two requests $r_i = 2$, $r_{i+j} = 4$, we will not be able to realize them.

## 4.3 Pathological Behavior of $KT_{\mathbf{R}}$ on $\mathfrak{C}_T$

### 4.3.1 Continuous Measures

The previous examples showcase the ways in which some important tools from the standard setting fail to generalize.

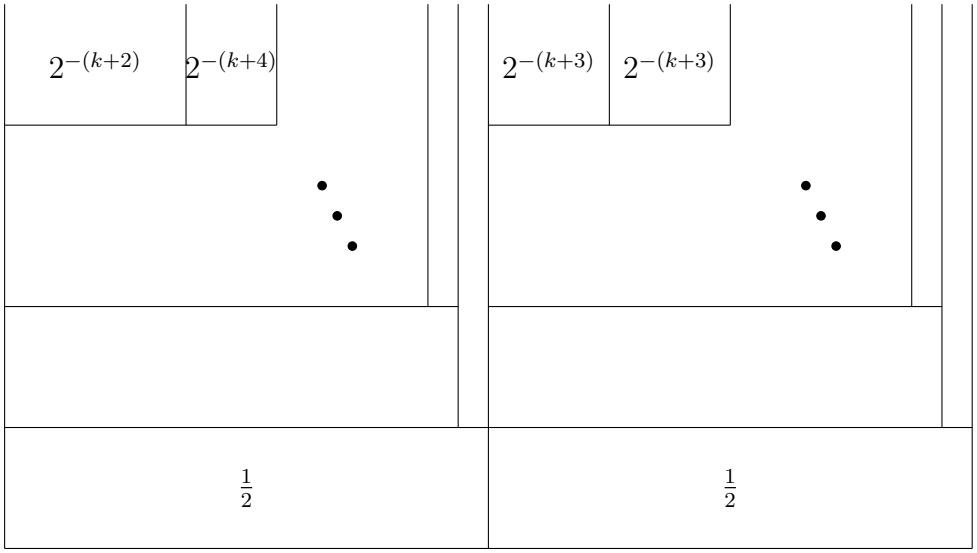


**Figure 4.4:** The partial measure $m_k$

In what follows we will see some ways in which the definition of $\mathrm{KT_R}$ violates our intuition from the standard setting. In Cantor Space, the complexity function on prefixes of computable strings grows as slowly as possible, as the are easily encoded by a fixed constant index and length. Now, when we look at the standard definition of $K$-Trivial, we note that, naturally, we expect longer strings to have naturally higher complexity than shorter strings. When we stop thinking about string length, and start thinking about cell measure, this breaks down. Our definition of $K_{\mathrm{R}}$-triviality is based on the intuition that strings with small measure should have higher complexity, while those with larger measure should have relatively lower complexity. The problem arises when the measure stays large on prefixes of a string for quite some time - we eventually run out of small descriptions for strings with large measure. In this way, even computable strings ... such as $0^\infty$, for exammple, may turn out to fail to be $K_{\mathrm{R}}$-trivial.

**Proposition 4.3.1.** There is a computable measure $\mu^\dagger$ such that $0^\infty \notin \mathrm{KT}_R$.

*Proof.* Using a combinatorial argument, we will satisfy requirements $P_e$ which say there exists $n$ such that $K(0^n) > K(-\log \mu^\dagger(0^n)) + e$. We will let $l_e$ denote the place

$P_e$ is satisfied. The marker $l_e$ will not move once assigned. For each e, simulate the universal machine until we find $\sigma_e$ such that $\mathbb{V}(\sigma_e) = e$ (this will provide a bound for $K(-\lceil \log \mu^\dagger(0^n))\rceil)$. Since there are only $2^{e+|\sigma_e|} - 1$ possible descriptions of length less than $e + |\sigma_e|$, we set $l_e = l_{e-1} + 2^{e+|\sigma_e|}$, $l_0 = 0$.

Now we define $\mu^\dagger$ in terms of the position of the markers: if $l_{i-1} < k \leq l_i$, then $\mu^\dagger(0^k) = 2^{-(i-1)} - \frac{k2^{-i}}{l_i}$, $\mu^\dagger(0^{k-1}1) = \mu^\dagger(0^{k-1}) - \mu^\dagger(0^k)$. and $\mu^\dagger(0^k 1\sigma) = 2^{-|\sigma|}\mu^\dagger(0^k 1)$. Clearly each $P_e$ will be satisfied, so $0^\infty \notin \mathrm{KT}_R$. □

Note that since the largest measure cell along each level is always the one of all 0's. $\mu(0^{l_i}) = 2^{-i}$, so $g_\mu(n) = l_n$, which has exponential growth. On the other hand, $c_\mu$ grows essentially linearly.

Note that since at the first level, $\mu^\dagger(1) = \frac{1}{2l_1}$, so for all $n < \lceil \log 2l_1 \rceil$, $c_{\mu^\dagger} = 0$. Since $\mu^\dagger(1) = \frac{1}{2l_1}$, we can safely say $c_{\mu^\dagger}(\lceil \log 2l_1 \rceil) = 1$. Since the measure is distributed evenly on extensions of 1, coarseness will continue to increase linearly until we get to the marker $l_1$, we have $c_{\mu^\dagger}(\lceil \log 2l_1 \rceil + n) = n + 1$ for $n \leq l_1$. Once we reach $\lceil \log 2l_1 \rceil + l_1$, the coarseness will stay constant for a time, since at level $l_1 + 1$, we have a cell with measure $\frac{1}{2^2 l_2}$. In general,

$$c_{\mu^\dagger}(n) = \begin{cases} 0 & : 0 \leq n < \lceil \log 2l_2 \rceil \\ l_i + n - \lceil \log 2^{i+1} l_{i+1} \rceil + 1 & : \lceil \log 2^{i+1} l_{i+1} \rceil \leq n < \lceil \log 2^{i+1} l_{i+1} \rceil + l_{i+1} \\ l_i & : \lceil \log 2^i l_i \rceil + l_i \leq n < \lceil \log 2^{i+1} l_{i+1} \rceil \end{cases}$$

However, the intervals $[\lceil \log 2^i l_i \rceil + l_i, \lceil \log 2^{i+1} l_{i+1} \rceil)$ eventually become empty, after which, $c_\mu$ is linear with slope 1 forever. To see that the intervals become empty:

$$\log(2^i l_i) + l_i = \log(l_i) + l_i + i >^+ \log(l_i) - 1 + i + |\sigma_i|$$

This is true since $|\sigma_i|$ is bounded above by $\log(a(i)) + d$, where $d$ is an index for the copy machine. Now,

$$\log(2^i l_i) + l_i >^+ \log(l_i) - 1 + i + |\sigma_i| = \frac{\log(l_i) + \log(2^{i+1+|\sigma_i|})}{2} > \log(2^{i+1} l_{i+1})$$

holds by concavity of the logarithm.

We suggest two ways to remedy this particular pathology:

**Definition 4.3.2.** We say $x \in 2^{\mathbb{N}}$ is $K_{R1}(\mu)$-trivial if

$$K(x \upharpoonright n) \leq^+ K(\max\{n, \lceil -\log \mu(x \upharpoonright n) \rceil\})$$

**Definition 4.3.3.** We say $x \in 2^{\mathbb{N}}$ is $K_{R2}(\mu)$-trivial if

$$K(x \upharpoonright n) \leq^+ K(< n, \lceil -\log \mu(x \upharpoonright n) \rceil >)$$

These alternative definitions are different approaches to prevent the pathological behavior of repeated requests seen in $\mu^\dagger$. Note that 4.3.2, contains all strings which are $K$-trivial in Cantor space by definition.

### 4.3.2 Measures with Atoms

With $\mu^\dagger$, we showed that a pathological measure can make a string which one would expect to be trivial look non-trivial. Now we will show the opposite is also possible: elements which one would expect to be random (for example, noncomputable strings) can be made to look non-random (in fact infinitely often $K_{\mathrm{R}}$ trivial) under this definition. For this, we relax the assumption that our measure be continuous. It is worth noting that for any computable measure, strings which are atoms are trivially random (eventually $-\lceil \log \mu(\sigma \upharpoonright n) \rceil$ becomes constant).

**Proposition 4.3.4.** There is a computable measure with atoms, $\mu$, such that all strings with infinitely many 1's will appear non-random to $\mu$.

*Proof.* We define $\mu(\epsilon) = 1$, $\mu(\sigma 1) = 2^{-a^{-1}(\sigma 1)}$, and $\mu(\sigma 0) = \mu(\sigma) - \mu(\sigma 1)$. This clearly defines a measure (on $\mathfrak{C}_T$), since $a$ was chosen to be order-preserving between the natural ordering of $\mathbb{N}$ and the lexicographic ordering on $\Sigma^*$, and so if $\omega \preceq \tau$, then $a^{-1}(\omega) < a^{-1}(\tau)$. Thus, for all $n$ such that $\sigma \upharpoonright n$ ends in 1, we have $K(\sigma \upharpoonright n) = K(-\lceil \log \mu(\sigma \upharpoonright n) \rceil)$, hence all strings with infinitely many 1's appear to be non-random. In fact, they are infinitely often $K_{\mathrm{R}}$-trivial. □

By the note above, we also see that all strings with only finitely many 1's are atoms, and, in fact, random according to $\mu$. Also, all strings with finitely many 0's will be $K_{\mathrm{R}}(\mu)$-trivial, and strings with both infinitely many 0's and 1's could possibly be trivial, or just trivial infinitely often. Of course the fact that strings with finitely many 1's are random is a direct consequence of the fact that the measure has atoms, but as highlighted in [19], measures with atoms also have unique properties in places other than the actual atoms.

# Chapter 5

# Generalizations to Metric and Measure Spaces

### Chapter Overview

Here we will examine a natural way to compare the random/trivial elements of metric spaces with those of measure spaces where the ambient set is $2^{\mathbb{N}}$ (though this is merely a convenience). We use it to compare $K_{\mathrm{MN}}$-triviality with $K_{\mathrm{R}}$-triviality, note where differences occur, and finally propose alternative definitions to triviality in a computable measure space, in Definition 5.2.1 and Definition 5.2.4.

## 5.1 $KT_{\mathbf{MN}} \subseteq KT_{\mathbf{R}}$ for Tame Measures

Throughout this Chapter, the following notational conventions will apply.

1. $\mathcal{X} = \left(X, d, \{\alpha_i\}_{i\in\mathbb{N}}\right)$ is a computable metric space, $\mu$ is a computable measure on $\mathcal{X}$, and $\mathcal{A} := \mathcal{A}'_d$ is a generator for $(\mathcal{X}, \mu)$,

2. Let $\mu^*$ be the push-forward measure on $\mathfrak{C}_T$, given for $\mu$ by Definition 3.2.7.

**Theorem 5.1.1.** With $\mathcal{X},\ \mathcal{A},\ \mu,\ \mu^*$ as above, suppose that $M$ is a positive integer

(necessarily at least 3) such that $g_{\mu^*}(n) - c_{\mu^*}(n) < d$ where $g$ is granularity, and $c$ is coarseness. If $x \in rep_{\mathcal{A}}$ and $x \in KT_{\mathrm{MN}}(\mathcal{X})$ then $x \in KT_{\mathrm{R}}(\mu, \mathcal{A})$.

*Proof.* Let $x \in rep_{\mathcal{A}}$, $x \in KT_{\mathrm{MN}}$, and $\gamma$ be a $K$-trivial Cauchy name for $x$. Since $\gamma$ is $K$-trivial, we have that $K(w_{\mathcal{A}}(x) \upharpoonright n) \leq^+ K^{\gamma}(w_{\mathcal{A}}(x) \upharpoonright n)$. We will determine the first $n$ bits of $w_{\mathcal{A}}(x)$ from $\gamma$ as follows:

To determine the $i^{th}$ bit of $w_{\mathcal{A}}(x)$, enumerate the computable balls $B(b_k, r_k)$ which are either included in $A_i^0$ or included in $A_i^1$. Search for a pair $(j, k)$ such that

$$j > -\log(r_k - d(b_k, \gamma_j)) + 1$$

so that we have

$$d(b_k, x) \leq d(b_k, \gamma_j) + d(x, \gamma_j) < r_k$$

and so $x \in B(b_k, r_k)$. Since $\gamma$ is a Cauchy name for $x$, we will eventually find such a pair; further, this procedure is clearly effective in $\gamma$. Taking $(j, k)$ to be such a pair for which $< j, k >$ is smallest possible, either $B(b_k, r_k) \subseteq A_i^0$ (and so is disjoint from $A_i^1$), or vice-versa. And, of course, in the first case, $x \in A_i^0$, and in the second $x \in A_i^1$, which determines the $i^{\text{th}}$ bit of $w_{\mathcal{A}}(x)$.

Now let $h(n)$ be the max of $h(n-1)+1$ and the max (over $i < n$) of the $j$'s found, as above, for determining the first $i^{\text{th}}$ bit of $w_{\mathcal{A}}(x)$ (we include $h(n-1)+1$, to ensure that $h$ is monotone increasing. Then knowing $\gamma \upharpoonright h(n)$, we can effectively carry out the above procedure for determining $w_{\mathcal{A}}(x) \upharpoonright n$. Thus, since $n.\gamma \upharpoonright h(n)$ is $\gamma$ computable, we see that:

$$K^{\gamma}(x \upharpoonright_{\mathcal{A}} n) \leq^+ K^{\gamma}(\gamma \upharpoonright h(n)) \leq^+ K^{\gamma}(n).$$

Notice that in this way, any number of bits of $x$ can be used in a $\gamma$ computation, and so knowing $\lceil -\log \mu(x \upharpoonright_{\mathcal{A}} n) \rceil$, since $\mu$ is computable and we have access to bits of $x$, we can determine $n$ is one of $M$ possible values. Hence by 2.3.23

$$K^{\gamma}(n) \leq^+ K^{\gamma}(\lceil -\log \mu(x \upharpoonright_{\mathcal{A}} n)_{\mathcal{A}} \rceil)$$

and $x \in KT_{R\mathcal{A}}$.

□

## 5.2 When $KT_{\mathrm{MN}}(\mathcal{X}) = KT_m(\mu, \mathcal{A})$

.

**Definition 5.2.1.** Let $\mu$ be a computable measure on $\mathfrak{C}_T$. An element $x$ of $2^{\mathbb{N}}$ is $K_m(\mu)$-trivial if and only if there exists $b$ such that for all $n$ there exists $m$ such that $0 < \mu(x \upharpoonright m) \leq 2^{-n}$ and $K(< a^{-1}(x \upharpoonright m, n >) < K(n) + b$. As usual, we write $x \in KT_m(\mu)$ to mean that $x$ is $K_m(\mu)$-trivial.

Now, let $\mathcal{X}$ be a computable metric space, $\mu$ be a computable measure for $\mathcal{X}$, $\mathcal{A}$ be a generator of $(\mathcal{X}, \mu)$, and $\mu^*$ be, as usual, the pushforward measure for $\mu$, obtained via $w_{\mathcal{A}}$. For $x \in X$:

$$x \text{ is } K_m(\mu, \mathcal{A})\text{-trivial if and only if } w_{\mathcal{A}}(x) \text{ is } K_m\left(\mu^*\right)\text{-trivial.}$$

As usual, we write $x \in KT_m(\mu, \mathcal{A})$ to mean that $x$ is $K_m\left(\mu, \mathcal{A}\right)$-trivial.

**Definition 5.2.2.** If $\mathcal{X} = (X, d, \{\alpha_i\}_{i\in\mathbb{N}})$ is a computable metric space, $\mu$ is a computable measure on $\mathcal{X}$, and $\mathcal{A}$ is a generator for $(\mathcal{X}, \mu)$, (on $X$), we say that $(\mu, \mathcal{A})$ *meshes* with $\mathcal{X}$ if and only if there exist total, computable 1:1 functions $h_1$, $h_2$ such that

$$\text{for all } x \in X \text{ and for all } n,\ \mu(B(x, 2^{-h_1(n)})) < 2^{-n}$$

and, as usual, letting $\mu^*$ be the push-forward measure for $\mu$ obtained via $w_{\mathcal{A}}$:

$$\text{for all } n,\ \text{ if } x, y \in \sigma,\ \text{and } \mu^*(\sigma) < 2^{-h_2(n)},\ \text{ then } d(x, y) < 2^{-n}$$

**Theorem 5.2.3.** Let $\mathcal{X} = (X, d, \{\alpha_i\}_{i\in\mathbb{N}})$ be a computable metric space, let $\mu$ be a computable measure on $\mathcal{X}$ which meshes with $\mathcal{X}$, and let $\mathcal{A}$ be a generator for $(\mathcal{X}, \mu)$ with the property that if $x \in rep_{\mathcal{A}}$ then $w_{\mathcal{A}}(x)$ takes on value 1 infinitely often. Then, for $x \in rep_{\mathcal{A}}$:

$$x \in KT_{\mathrm{MN}}(\mathcal{X}) \text{ if and only if } x \in KT_m\left(\mu, \mathcal{A}\right).$$

*Proof.* Suppose $x \in rep_{\mathcal{A}}$ and $x \in KT_m(\mu, \mathcal{A})$, and let $n \in \mathbb{N}$. By our hypothesis on $w_{\mathcal{A}}$, there exists $\sigma$ such that $x \in [\sigma 1]_{\mathcal{A}}$, and $\mu(\sigma 1) < 2^{-h_2(n)}$.

Thus, by assumption,

$$K(< \sigma 1, h_2(n) >) \leq^+ K(h_2(n))$$

Since $h_2$ is 1:1 and computable, $K(h_2(n)) =^+ K(n)$. Further, from $< \sigma 1, h_2(n) >$, we can effectively determine $< \pi_1(|\sigma 1|), n >$, so

$$K(< \pi_1(|\sigma 1)|, n >) \leq^+ K(< \sigma 1, h_2(n) >) \leq^+ K(h_2(n)) =^+ K(n)$$

Since the $|\sigma 1|^{th}$ bit of $x \upharpoonright_{\mathcal{A}} n = 1$, $x \in B(\alpha_{\pi_1(|\sigma 1|)}, n)$, and so $x \in KT_{\mathrm{MN}}(\mathcal{X})$.

Now, let $x \in KT_{\mathrm{MN}}(\mathcal{X})$. Thus, $x$ has a $K$-trivial Cauchy name $\gamma$. Now, we can use the method of the proof of Theorem 5.1.1 to obtain a cylinder of small measure containing $x$.

Let $h(n)$ be the max of the number of bits of $\gamma$ in determining the first $n$ bits of $x$, $h_1(n)$, and also $h(n-1)+1$ (to ensure $h$ is $1:1$). Thus, by assumption,

$$K(< \alpha_{\gamma(h(n))}, h(n) >) \leq^+ K(h(n)) =^+ K(n)$$

From $< \alpha_{\gamma(h(n))}, h(n) >$, we can determine $< x \upharpoonright_{\mathcal{A}} n, n >$, and since $d(\alpha_{\gamma(h(n))}, x) < 2^{-h_1(n)}$, $\mu(x \upharpoonright_{\mathcal{A}} n) < 2^{-n}$ as desired, so $x \in KT_m(\mu, \mathcal{A})$.

□

We now give a variant of Definition 5.2.1 that highlights the role of the local version the granularity function from Definition 3.2.16. It will turn out the variant is equivalent to the original. That will be the content of the next and final proposition, Proposition 5.2.5.

**Definition 5.2.4.** For $x \in 2^{\mathbb{N}}$, $x \in KT'_m$ if and only if

$$K(< n, \sigma \upharpoonright g_{x,\mu}(n) >) \leq^+ K(n)$$

.

**Proposition 5.2.5.** Definition 5.2.1 and Definition 5.2.4 are equivalent.

*Proof.* If $x \in 2^{\mathbb{N}}$ satisfies Definition 5.2.4, then given $n$, $x \upharpoonright g_{x,\mu}(n)$ is the required witness with $K(< n, x \upharpoonright g_{x,\mu}(n) >) \leq^+ K(n)$. If $x$ satisfies Definition 5.2.1, then given $n$, there is a witness $K(< n, x \upharpoonright m_n >) \leq^+ K(n)$. Since $x \upharpoonright g_{x,\mu}(n) \prec x \upharpoonright m_n$, we have $K(x \upharpoonright g_{x,\mu}(n)) \leq K(x \upharpoonright m_n)$. Note that the function $x. < |x|, x >$ is computable and $1:1$. Applying it to both sides of the previous inequality we get

$$K(< n, x \upharpoonright g_{x,\mu}(n) >) \leq^+ K(< n, x \upharpoonright m_n >)$$

and thus Definition 5.2.4 is satisfied.

□

Using the same function $f$ as above, we see that in the standard setting of Cantor space with Lebesgue measure, $g_{x,\mu}(n) = n$ and $K(< n, \sigma \upharpoonright g_{x,\mu}(n) >) =^+ K(f(\sigma \upharpoonright h(n)) =^+ K(\sigma \upharpoonright n)$, so that Definition 5.2.4 agrees with the standard definition of $K$-triviality. This definition also parallels Definition 4.3.3 very strongly, essentially inverting the argument of the right hand side. The main difference is that in Definition 5.2.4, we don't care about every prefix of $\sigma$ satisfying an inequality - we care about the complexity of prefixes dipping low enough at about the same rate as that of the measure shrinking.

# Chapter 6

# Concluding Remarks, Open Problems, and Future Directions

Here we discuss possible future directions to pursue.

## 6.1 Machine Existence Theorem

In Chapter 3, we saw both a partial positive and a partial negative result about the Machine Existence Theorem, in that it holds for tame measures, and fails for the strictest possible generalization to computable measure spaces. The exact standing of this theorem in a generalized setting is still unknown. It would be interesting to expand both sides of this question: is there a wider class of measures for which the Machine Existence Theorem holds? If so, does this class have a natural characterization? Can it be shown that the Machine Existence Theorem fails in a more general way (i.e. $MET(\mu, \vec{R}, b)$)? One possibility is that the theorem may hold, not in a computable way, but in a semi-computable way. That is, instead of having $i.\sigma_i$ being a computable function, we may instead have that $\sigma_{i,s}$ is a computable function of $s$ and $i$ that that $\sigma_{i,s} \to \sigma_i$ as $s \to \infty$. We conjecture that this is the case.

## 6.2 Computable measures and Generators

We showed that for the right measure / generator, we have $KT_{\mathrm{MN}} \subseteq K_{\mathrm{R}}$. Is there a natural class of measures / generators for which the reverse inclusion holds?

One fundamental idea about generators which is currently unknown is whether for a fixed measure, different generators preserve various properties, such as $K_{\mathrm{R}}$

## 6.3 Measures and Their K-Trivials

In [20], the following question is investigated: given a real number (or element of Cantor space), which measures can it appear random with respect to? The answer to which is that any non-computable real can be random with respect to some probability measure. However, some reals $x$ can only be random with respect to atomic measures. At first this is not surprising, since atoms are always random for trivial reasons, but it turns out such $x$'s can be random with respect to an atomic measure within which they are not atoms, and not withing any continuous measure.

This question would be interesting to investigate with the dual notion of triviality: Given a real number, which measures does it appear trivial with respect to? Presumably, all $x \in 2^{\mathbb{N}}$ look trivial with respect to some measure, provided the measure is allowed to be highly incomputable, to the point that it contains lots of information about $x$. Other variants seem less obvious: Which reals can be trivial with respect to a computable measure? An exactly computable measure?

## 6.4 Noncomputable measures

In [1], an approach of using representations of noncomputable measures as oracles leads to a definition of randomness. This has not been investigated for $K$-triviality. As mentioned in 6.3, if done successfully, this would likely lead to instances where computationally powerful measures give a large class of $K$-trivials.

## 6.5 Other Possible Settings

In [1], a topological approach is used to define Martin-Lof tests on spaces of closed sets. He proves that this approach is equivalent (at least under Lebesgue measure on $3^\omega$) to an approach used in [2], which encodes closed sets by trees, which are in turn encoded by ternary strings. The tree approach is of particular interest, because the use of ternary strings opens up the possibility of a prefix definition of triviality in the same spaces.

# Bibliography


[1] Logan M. Axon, *Martin-löf randomness in spaces of closed sets*, The Journal of Symbolic Logic **80** (2015), no. 2, 359–383.

[2] George Barmpalias, Paul Brodhead, Douglas Cenzer, Seyyed Dashti, and Rebecca Weber, *Algorithmic randomness of closed sets*, Journal of Logic and Computation **17** (2007), no. 6, 1041–1062.

[3] Gregory J. Chaitin, *Nonrecursive infinite strings with simple initial segments*, IBM Journal of Research and Development **21** (1977), 350–359.

[4] Rodney G. Downey and Denis R. Hirschfeldt, *Algorithmic randomness and complexity*, Springer Science & Business Media, 2010.

[5] Peter Gács, *Uniform test of algorithmic randomness over a general space*, Theoretical Computer Science **341** (2005), no. 1-3, 91–137.

[6] Peter Gács, Mathieu Hoyrup, and Cristóbal Rojas, *Randomness on computable probability spaces—a dynamical point of view*, Theory of Computing Systems **48** (2011), no. 3, 465–485.

[7] Andrzej Grzegorczyk, *Computable functionals*, Fund. Math **42** (1955), no. 19553, 168–202.

[8] Rupert Hölzl and Christopher P. Porter, *Randomness for computable measures and initial segment complexity*, Annals of Pure and Applied Logic **168** (2017), no. 4, 860–886.

[9] Mathieu Hoyrup and Cristóbal Rojas, *Computability of probability measures and Martin-Löf randomness over metric spaces*, arXiv preprint arXiv:0709.0907 (2007).

[10] Stephen Cole Kleene, *Countable functionals*, Constructivity in Mathematics (1962), 81–100.

[11] Andrei N. Kolmogorov, *On tables of random numbers*, Sankhyā: The Indian Journal of Statistics, Series A (1963), 369–376.

[12] Antonín Kučera and Sebastiaan A. Terwijn, *Lowness for the class of random sets*, The Journal of Symbolic Logic **64** (1999), no. 4, 1396–1402.

[13] Per Martin-Löf, *The definition of random sequences*, Information and control **9** (1966), no. 6, 602–619.

[14] Alexander Melnikov, *Computability and structure*, Ph.D. thesis, ResearchSpace@ Auckland, 2012.

[15] Alexander Melnikov and André Nies, *K-triviality in computable metric spaces*, Proceedings of the American Mathematical Society **141** (2013), no. 8, 2885–2899.

[16] Yiannis Nicholas Moschovakis, *Recursive metric spaces*, Fundamenta Mathematicae **55** (1966).

[17] André Nies, *Lowness properties and randomness*, Advances in Mathematics **197** (2005), no. 1, 274–305.

[18] ———, *Computability and randomness*, vol. 51, Oxford University Press, 2009.

[19] Jan Reimann and Theodore Slaman, *Measures and their random reals*, Transactions of the American Mathematical Society **367** (2015), no. 7, 5081–5097.

[20] Jan Reimann and Theodore A. Slaman, *Measures and their random reals*, arXiv preprint arXiv:0802.2705 (2008).

[21] Jason Rute, *Computable randomness and betting for computable probability spaces*, Mathematical Logic Quarterly **62** (2016), no. 4-5, 335–366.

[22] Claus-Peter Schnorr, *A unified approach to the definition of random sequences*, Mathematical systems theory **5** (1971), no. 3, 246–258.

[23] ———, *The process complexity and effective random tests.*, Proceedings of the fourth annual ACM symposium on Theory of computing, ACM, 1972, pp. 168–176.

[24] Alexander Shen, *Around Kolmogorov complexity: basic notions and results*, Measures of Complexity, Springer, 2015, pp. 75–115.

[25] Robert M. Solovay, *A model of set-theory in which every set of reals is lebesgue measurable*, Mathematical Logic In The 20th Century, World Scientific, 2003, pp. 480–535.

[26] Antoine Taveneaux, *Randomness zoo*, https://hal.archives-ouvertes.fr/hal-00850992/document (2012).

[27] M. van Lambalgen, *Random sequences. PhD thesis*, Universiteit van Amsterdam (1987).

[28] Domenico Zambella, *On sequences with simple initial segments*, Institute for Language, Logic and Information, 1990.

[29] Alexander K. Zvonkin and Leonid A. Levin, *The complexity of finite objects and the development of the concepts of information and randomness by means of the theory of algorithms*, Russian Mathematical Surveys **25** (1970), no. 6, 83–124.

# Bill Franczak | Curriculum Vitae

(732) 859-0053 • billfranczak@gmail.com

## Education

- **Lehigh University** — **Bethlehem, PA**
  ***Doctor of Philosophy***, *Mathematics, January 2019* — *2012–2018*
- **Lehigh University** — **Bethlehem, PA**
  ***Master of Sciences***, *Mathematics* — *2012–2014*
- **The College of New Jersey** — **Ewing, NJ**
  ***Bachelor of Sciences***, *Computer Science* — *2008–2012*
- **The College of New Jersey** — **Ewing, NJ**
  ***Bachelor of Sciences***, *Mathematics* — *2008–2012*

## Research Interests

Logic, specifically computability theory and algorithmic randomness. Cryptography. Knot Theory.

## Teaching Experience

### Instructor, Lehigh University

- **Survey of Calculus II** — **Bethlehem, PA**
  *Math 052* — *Summer Session 2016 & 2017, Spring 2018*
- **Preparation for Calculus** — **Bethlehem, PA**
  *Math 000* — *Fall Semester 2017*

### Teaching Assistant, Lehigh University

**Courses Taught:** Calculus I, II, & III (Math 021, 022, & 023), Calculus with business applications (Math 081), Survey of Calculus I & II (Math 051 & 052), Statistics (Math 012, Spring 2018)

## Talks

### Invited Talks

- **Graduate Student Intercollegiate Mathematics Seminar, Lehigh University** — **Bethlehem, PA**
  *Shor's Quantum Factoring Algorithm* — *October 2017*
- **Graduate Student Intercollegiate Mathematics Seminar, Lehigh University** — **Bethlehem, PA**
  *Algorithmic Randomness in General Settings* — *October 2016*

### Contributed Talks

- **Joint Mathematics Meetings, AMS** — **San Diego, CA**
  *Algorithmic Randomness in General Settings* — *January 2018*

## Computer Skills

- **Operating Systems:** Windows, UNIX, Linux
- **Software:** Microsoft Office, MATLAB, Mathematica, SAS, MyLab, CourseSite, WebAssign, SourceTree
- **Programming Languages:** C++/C#, Java, Scala, LaTeX, Metapost

### Notable Projects

- **Quadratic Number Field Sieve:** Solo project where I implemented a powerful algorithm to factor large numbers.
- **Mano's 8-bit Computer:** Two-Person project where I implemented a full simulation of an 8-bit computer, including machine level code and logic circuits.
- **Astronomy Quiz:** Team project where we created interactive educational software for 3rd-grade students at a local elementary school.

## Publications

- Curtis, Cynthia L., et al. "Repeated boundary slopes for 2-bridge knots." *Journal of Knot Theory and Its Ramifications* 24.14 (2015): 1550068.

## Memberships

- **American Mathematical Society**

## Awards

- **Strohl Graduate Research Fellowship**
  *Funded by Lehigh University's College of Arts and Sciences* — *Summer 2015*

## Service Activities

- **Graduate Student Mentor, Lehigh University**
  *Mentor to incoming graduate students* — *2016-Present*
- **Mathematics Help Center, Lehigh University**
  *Drop-in tutor for undergraduate students* — *2012-Present*