Aspectos de aleatoriedad

Figueira, Santiago Daniel

2006

Tesis Doctoral

Universidad de Buenos Aires
Facultad de Ciencias Exactas y Naturales
Departamento de Computación

Aspectos de aleatoriedad

Tesis presentada para optar al título de
Doctor de la Universidad de Buenos Aires
en el área de Computación

Santiago Daniel Figueira

Directora de tesis: Verónica Andrea Becher

Lugar de trabajo: Departamento de Computación, FCEyN, UBA

Buenos Aires, 2006

A Coni

CONTENTS

ASPECTOS DE ALEATORIEDAD

Resumen. En esta tesis, investigamos algunos aspectos de aleatoriedad y trivialidad definidos por la teoría de largo de programa.

Primero abordamos la *aleatoriedad* y la *absoluta normalidad* de números reales. Ambos conjuntos de reales tienen medida de Lebesgue 1 y son nociones que implican varias propiedades de estocasticidad. A pesar de esto, no ha sido fácil dar ejemplos concretos en estas clases. Probamos que existen números absolutamente normales que son computables y damos dos algoritmos para construirlos. El primero está basado en una reformulación computable de un resultado de Sierpinski de 1916. El segundo es parte de nuestra reconstrucción de un manuscrito inédito de Turing sobre números normales. En cuanto a ejemplos de aleatoriedad, generalizamos la probabilidad de detención Ω de Chaitin y analizamos la probabilidad de que una máquina universal se detenga y devuelva un resultado en un conjunto dado X. Estudiamos la relación entre las propiedades de X provenientes de la teoría de la computabilidad y las propiedades de aleatoriedad de la probabilidad inducida.

El segundo aspecto de aleatoriedad que tratamos es el estudio de una variante de la complejidad clásica de largo de programa que no involucra oráculos, y nos preguntamos si esta noción conduce a una definición más estricta de aleatoriedad. Definimos nuestra función de complejidad en base a máquinas de Turing monótonas que realizan cómputos infinitos. Investigamos algunas propiedades de esta función y consideramos las definiciones inducidas de aleatoriedad y trivialidad. Con esta última noción caracterizamos a los reales computables.

El último aspecto se vincula con la *anti-aleatoriedad* y la posibilidad de caracterizar a los reales llamados K-triviales con nociones que no involucren directamente a la complejidad de largo de programa libre de prefijos. Proponemos e investigamos dos nociones de *lowness* que tienen sus raíces puramente en la teoría de la computabilidad, reforzando otras ya existentes en la literatura. Relacionamos la complejidad de largo de programa plana C y libre de prefijos K con estas nociones, considerando variaciones de K-trivialidad y C-trivialidad.

Concluimos con una lista de las principales preguntas que quedaron abiertas.

Palabras clave. Teoría algorítmica de la información, teoría de la computabilidad, complejidad de largo de programa, complejidad de Kolmogorov, números normales, números absolutamente normales, aleatoriedad, número Omega de Chaitin, probabilidad de detención, jerarquía aritmética, cómputos infinitos, máquina de Turing, máquina monótona, K-trivialidad, noción de *lowness* (bajura), *traceability* (rastreabilidad).

ASPECTS OF RANDOMNESS

Summary. In this thesis we investigate some aspects of randomness and triviality defined by the theory of program-size.

We first deal with *randomness* and *absolute normality* of real numbers. Both sets of reals have Lebesgue measure 1 and they are notions that imply several properties of stochasticity. Despite that fact, it has not been easy to give concrete examples in such classes. We prove that there are absolutely normal numbers which are computable and we give two algorithms for constructing such numbers. The former is a computable reformulation of a result of Sierpinski of 1916. The latter is part of our reconstruction of an unpublished manuscript of Turing on normal numbers. For examples of randomness, we generalize Chaitin's halting probability Ω and we analyze the probability that a universal machine halts and gives an output in a given set X. We study the relationship between the computability theoretic properties of X and the randomness properties of the induced probability.

The second aspect of randomness that we tackle is the study a variant of the classical definition of program-size complexity which does not involve oracles, and we ask whether it leads to a stronger notion of randomness. We define our complexity function based on monotone Turing machines performing unending computations. We investigate some properties of this function and we consider the induced definitions of randomness and triviality. With this last notion we characterize the computable reals.

The last aspect deals with *anti-randomness* and the possibility to characterize the so called K-trivial reals in terms of notions that do not directly involve the prefix-free program-size complexity. We propose and investigate two computability theoretical combinatorial lowness notions by strengthening other notions already existing in the literature. We relate the plain C and the prefix-free program-size complexity K with these notions by considering variations of K-triviality and C-triviality.

We conclude with a list of the main questions that remain open.

Key words. Algorithmic information theory, computability theory, program-size complexity, Kolmogorov complexity, normal numbers, absolutely normal numbers, randomness, Chaitin's Omega number, halting probability, arithmetical hierarchy, infinite computation, Turing machine, monotone machine, K-triviality, lowness notion, traceability.

ACKNOWLEDGEMENTS

This thesis is the result of about four years of work whereby I have been accompanied and supported by many people. I am very happy to express now my gratitude to all of them.

I would like to express my deep and sincere thanks to my supervisor Verónica Becher for her continuous support and friendship, for her enthusiasm in my work and for the motivating and critical atmosphere during the many discussions we had. I am also indebted to her for giving me the chance to spread my work and to meet prominent researchers at several meetings.

The teaching and guidance of my co-supervisor André Nies was essential for my learning. Furthermore, I want to thank him for his generosity, his deep explanations and for being such a good friend of mine. I also want to express my gratitude to André, Bakhadyr Khoussainov and the Marsden Fund of New Zealand for inviting me to The University of Auckland from October 2004 to January 2005. The quantity of knowledge I learnt in that travel was incomparable. During that trip, Rod Downey invited me to Victoria University of Wellington. I am indebted to him for the interesting discussions we had and for being such a kind host. I also want to thank Cristian Calude for his hospitality during my visit to Auckland.

I am especially grateful to Frank Stephan for sharing with me his brilliant ideas, for his clear explanations and for fruitful conversations. My discussions with Frank have been of great value for me. I also thank him and Gouhua Wu for their generosity for inviting me to work with them in an article. I also thank Joe Miller, who kindly taught me a lot of tricks on randomness.

Besides the people already mentioned I would like to thank the following persons for discussions on various subjects: Max Dickmann, Joos Heintz, Serge Grigorieff, Denis Hirschfeldt, Bjørn Kjos-Hanssen, Guillermo Martínez, Elvira Mayordomo, Wolfgang Merkle, Silvana Picchi, Jan Reimann, Ted Slaman, Ludwig Staiger, Sebastiaan Terwijn, and Paul Vitanyi.

I had the pleasure to co-supervise and work with Alejo Capparelli, Mariano Moscato, Rafel Picchi and Martín Urtasun, all students who did their graduation work in this topic. I also thank Benjamín R. Callejas Bedregal, Daniel Gorín and Rafael Grimson, with whom I have done some research not directly connected to this thesis but which was very enlightening for me.

I would like to thank the Group in Logic and Computaility (GLyC) for the interest in my work and for the many enriching discussions on different topics. In particular I would like to thank Carlos Areces, Diego Figueira, Sergio Mera, Mariano Pérez Rodríguez and Ricardo Rodríguez. I am also grateful to the following students and professors of the Department of Computer Science at the University of Buenos Aires for providing a nice atmosphere: Santiago Bazerque, Diego Bendersky, Flavia Bonomo, Victor Braverman, Sergio Daicz, Pablo Factorovich, Diego Fernández Slezak, Andrés Ferrari, Juan Pablo Galeotti, Diego Garbervetsky, Nicolás Kicillof, Carlos Lopez Pombo, Javier Marenco, Esteban Mocskos, Yuri Poveda, Fernando Schapachnik, Enrique Tobis, Pablo Turjanski and Demian Wasserman.

This research has been supported by various organizations. Besides the Department of Computer Science at the University of Buenos Aires, which provided me an excellent working environment, I also acknowledge Fundación Antorchas, Fundación YPF and CONICET.

1. INTRODUCTION

In this thesis we investigate and discuss some aspects of randomness and triviality defined by the theory of program size.

We first deal with randomness and absolute normality. Both are definitions that imply several properties of stochasticity and both classes contain almost all real numbers, in the sense of Lebesgue measure. Despite that fact, it has not been easy to give concrete examples in such classes, that is, to identify single elements with distinctive proper names.

Absolutely normal numbers were first defined by Borel in 1909 and he proved that almost all reals are absolutely normal [16]. The idea of a real being *normal* to a given scale t is that every digit and block of digits appears equally frequent in its t-scale fractional expansion. *Absolutely normal* numbers are those that are normal to *every* scale. The problem of giving concrete examples of such numbers was raised by Borel as soon as he introduced the definition of normality.

The first example of an absolutely normal number was given by Sierpinski in 1916 [69], using generating functions and working with a sequence of sets with limiting measure 0 covering the reals that are candidates not to be absolutely normal. It is not clear whether his example is a *computable* number or not: his work appeared twenty years before the concept of computability was formalized. In the beginning of chapter 2 we reformulate Sierpinski's construction in an effective way and in Theorem 2.4.9 we exhibit an algorithm to compute an absolutely normal number, which becomes the first known example of an absolutely normal number that is also computable. The proof techniques we use in this part of the thesis is to computably (and carefully) bound the error at each stage of the construction, using some facts and methods of measure theory.

While we were looking for other examples of absolutely normal numbers, we found a Turing's manuscript on normal numbers which remained unpublished until 1992, when it was transcribed in the *Collected works of Alan Turing* [76]. This manuscript is incomplete, unclear and hard to read. Our motivation was to explore and make explicit the techniques used by Turing in relation to normal numbers, and our challenge was to try to reconstruct it in the most accurate possible way. The second part of chapter 2 deals with the reconstruction of Turing's manuscript. In particular, in Turing's work there is an important unproved lemma (transcribed in this document as Lemma 2.7.2) that turns to be essential for his construction. We replaced this unproved lemma with our Lemma 2.7.7, whose statement is very close to the original, but for which we were able to give a proof. Along these sections, we use elementary techniques from number theory and measure theory. As an application to our computable reconstruction of Sierpinski's or Turing's work, in Theorem 2.9.4 we show that there are not only absolutely normal numbers in the least Turing degree, but in *every* 1-degree. To prove this, we modify our algorithms and use the Lebesgue density theorem.

The formal definition of a *random* real, as conceived first by Martin-Löf [56] and later by Levin [52], Schnorr [67] and Chaitin [23] appeared in the late 1960s, once the computability theory was strongly developed. All these definitions have been proved to be equivalent. Roughly, Martin-Löf random reals are those which can avoid certain effectively presented measure 0 sets; Levin-Chaitin random reals are those which are algorithmically incompressible, in the sense that the prefix-free program-size complexity K is high in every initial segment of the real. The class of random reals is much more restrictive than the class of absolutely normal reals. In fact, no computable real can be random. However, it is still a huge class, since almost all reals are random. As with absolute normality, the problem of finding particular examples of random reals, i.e. to give a specific symbolic definitions, has been very difficult. Martin-Löf showed that almost all reals are random, but the first explicit real which was random was due to Chaitin [23]. He defines the number Ω, which represents the probability that a certain universal Turing machine halts when the input is generated by tossing a coin. In fact there is no single Ω, but a whole class of Ω numbers; one $\Omega_{\mathbf{U}}$ for each universal machine $\mathbf{U}$. So we have at our disposal the Ω numbers; all of them are *left-c.e.*, that is, they have the property that can be computably approximated from below via a sequence of rationals. Using the standard notion of computability relative to an oracle, one obtains new Ω numbers that are halting probabilities of Turing machines endowed with oracles. These examples are *more random* and also at higher levels of the arithmetical hierarchy, but still left-c.e. in their degree. But we sought for other examples of *significative* random reals, possibly not left-c.e. nor right-c.e.; we looked for a *new* source of examples of random reals. With this in mind, we started studying the following conjecture of Grigorieff, stated in 2002: "The probability $\Omega_{\mathbf{U}}[X]$ that a universal Turing machine $\mathbf{U}$ halts and outputs a string in a fixed non-empty set X is random. Moreover the harder the set X, the more random $\Omega_{\mathbf{U}}[X]$ will be." Grigorieff's conjecture is a wide generalization of Chaitin's Ω, which, if true, would lead to a lot of new examples of random reals. This study comprises chapter 3 and shows that in many ways the situation is much more complicated than the elegant conjecture suggests. What can be said about the relationship between the computability theoretic properties of X and the randomness properties of the induced probability? It turned out that the conjecture was not true in its most general form but still is true with some refinements and adjustments. For the classical halting computations we found some positive instances but unfortunately also many negative examples. On the other hand, using infinite computations, much more positive examples have been discovered in [14, 12], at the same time this thesis was carried on. Our main positive example is Theorem 3.6.1, which states that for any $n \geq 2$, if X is Σ^0_n-complete or Π^0_n-complete then $\Omega_{\mathbf{U}}[X]$ is random. The proof of this result uses an application of the Recursion Theorem over the existing technique employed to prove that Ω is random. A further interesting part of chapter 3 is devoted to the study of the topological structure of the set of reals that can be obtained as the halting probability for some X, that is reals of the form $\Omega_{\mathbf{U}}[X]$. Here the main result is Theorem 3.7.2, saying that for any universal machine $\mathbf{U}$, this set is of a rather simple structure, namely, it is the union of finitely many closed intervals. As a consequence of this theorem, it follows that any real small enough is $\Omega_{\mathbf{U}}[X]$ for some X and it also yields a proof that there are Δ^0_2 sets X for which $\Omega_{\mathbf{U}}[X]$ is rational, hence far from random.

The notion of randomness defined by Levin-Chaitin involves the prefix-free program-size complexity K and this gives the classical definition of randomness that coincides with the notion of Martin-Löf. The use of machines endowed with oracles yields a complexity function relative to these machines and leads to stronger notions of randomness. In this way, by relativizing K to oracles, we obtain a whole hierarchy of random reals. The second aspect of randomness that we tackle in this thesis is the study another version of the program-size complexity, which does not appeal to oracles in its definition, and we ask whether it leads to a stronger notion of randomness. In chapter 4 we define K^∞ as the prefix-free program-size complexity associated to possibly infinite computations in monotone machines. These unending computations have proved to be useful for giving new examples of random reals [6, 5]. How does this new complexity function associated to programs that never halt behave? What are the classes of random and anti-random reals induced by K^∞? These are some of the questions that we attack in this chapter. We prove that the function K^∞ is between K and $K^{\emptyset'}$ (this last is the prefix-free program-size complexity relative to the Halting Problem $\emptyset'$) except for additive constants, but K^∞ is inherently different from the relativization of K to any oracle. That is, K^∞ does not behave like K^A, no matter what is A. Surprisingly, although this new complexity function K^∞ is very different from K, the definition of randomness induced by K^∞ exactly coincides with the classical Levin-Chaitin's notion of randomness induced by K. The situation is completely different regarding the notion of anti-randomness. A real A is K-trivial if A is highly algorithmically compressible, in the sense that the prefix-free program-size complexity of every prefix of A is as low as it can be. It turns out that K-triviality, a notion opposite to randomness, is very different when working with K or K^∞. In Theorem 4.5.6 it is proved that a real is K^∞-trivial if and only if it is computable. This is a big difference with respect to K-triviality, where it is known [73, 34] that there exist non-computable K-trivial reals. In this aspect, K^∞ behaves more like C, the *plain* program-size complexity, where C-trivial reals coincide with the computable ones [24]. Our result partially answers an open question of Chaitin about the possibility to characterize the computable reals in terms of the *prefix-free* complexity K instead of the *plain* C: using K^∞, our variant of K for infinite computations, we do so. In this chapter we use repeatedly a proof technique which consists of exploiting the Shoenfield's Limit Lemma from computability theory. We also use some other techniques from this theory like approximations by effective trees.

A *lowness property* of a real A says that A is computational weak when used as an oracle, and hence A is close to being computable. The class of K-trivial reals was characterized by using different lowness notions. A real is *low for K* when it is not useful as an oracle to reduce the prefix-free program-size complexity of a string; a real is *low for random* when it is not useful as an oracle to detect regularities in a random real. Nies [62] proved that the classes of low for K, low for random and K-trivial reals coincide. Even though the three mentioned classes represent the same notion, they express very different aspects of it. However, the definition of the three classes involve in one way or another the prefix-free program-size complexity K. In [74] it was shown that the class of low for Schnorr tests (sets which are not useful as oracles for detecting patterns in a Schnorr random real) coincides with the class of *recursively traceable* sets, a combinatorial notion which is purely computably theoretical, rather than measure theoretical. The third aspect we deal with in chapter 5 is the possibility to characterize the K-trivial reals in terms of a purely computably theoretical combinatorial lowness notion which does not involve K in the definition. Two new notions are proposed: *strong jump-traceability* and *well-approximability.* They are strong variants of the notions of jump-traceability and ω-c.e. for sets of natural numbers, studied in [60]. A special emphasis is given to the case where the *jump* of A is ω-c.e. Roughly speaking, a set A is jump-traceable if one may effectively enumerate a set of possible values for the jump $J^A(e)$ and number of values enumerated is computably bounded; a set is ω-c.e. if it can be computably approximated with a $\{0,1\}$-valued function and the number of changes of this approximation is also computable bounded. Our notions are much stronger versions, in the sense that the computable bounds introduced in the above definitions may be forced to grow arbitrarily slow. There is a tight connection between strong jump-traceability and well-approximability: we proved that if A' is well-approximable then A is strongly jump-traceable, and that the converse holds as well in case A is c.e. We could not prove nor disprove that any of these two new notions lead to a characterization of K-trivial reals, but in Theorem 5.4.3 it is given a characterization of strongly jump-traceable reals as a variant of low for C reals (reals that, when used as oracles, does not help much in reducing the C complexity of strings). Furthermore, in Corollary 5.5.4 it is given a characterization of the jump-traceable reals as a variant of the low for K reals. Among the proof techniques used in this chapter, we can mention the finite injury priority method, strategies from algorithmic information theory for compressing information and some counting arguments. In particular, in Theorem 5.2.5 there is an interesting application of the finite injury method with *dynamic* requirements.

In the rest of this chapter we introduce some definitions and results that will be needed in subsequent chapters. In section 1.1 we describe concisely the field to which this thesis belongs. In section 1.2 we fix some notation and give some basic definitions that we will use. In section 1.3 we mention some results needed from computability theory. In section 1.4 we introduce program-size complexity, one of the main objects of study of this thesis. Sections 1.5 and 1.6 deal with some needed material related with randomness and triviality.

1.1 Algorithmic information theory

The theory of program size complexity, also known as algorithmic information theory, was initiated independently in the 1960s by Kolmogorov [46], Solomonoff [72] and Chaitin [22]. This theory defines a notion of complexity of strings taking into account the length of the shortest program which computes the string. From among all the descriptions of a string we can take the length of the shortest program as a measure of the string's complexity. In this way, programs are regarded as algorithmic descriptions of strings. A string σ is simple, that is, has low complexity, if its complexity is substantially smaller than the length of σ; and a string is complex if its algorithmic description is as large as the length of σ. The pigeonhole principle shows that the large majority of the strings have high complexity.

The function which associates to each string σ the length of the shortest program which outputs σ is called *program-size complexity* or *Kolmogorov complexity* and was introduced first by Kolmogorov [46] and studied by Levin [50, 51], Schnorr [66, 67], Gács [40] and Chaitin [23].

It should be noticed that in this definition there is an implicit underlying *universal machine*, that is, a particular Turing machine that is able to simulate any other Turing machine. This universal machine is used as reference to execute the mentioned programs. When the domain of all the Turing machines is prefix-free (i.e. there are no two halting programs such that one is a proper extension of the other) we talk about *prefix-free* program-size complexity (here denoted by K) and when no restriction on the domain is imposed, we speak of *plain* program-size complexity (here denoted by C). Although these complexities depend on the underlying universal machine, they are asymptotically independent of it [19, 54].

The roots of the study of algorithmic randomness go back to von Mises' work of the early 20th century [78], where he argued that random sequences should have several properties of stochasticity from classical probability theory. For example, in a random binary sequence of 0s and 1s, the proportion between the number of 1s in the first n digits of the sequence and the number n should be $1/2$ when n tends to infinity. Intuitively a sequence is random when it lacks structure or regularity, in other words, it does not have recognizable patterns. One would like to define random sequences as those which are indistinguishable from an infinite output of tossing a coin and writing 0 if it comes out a head or 1 if it comes out a tail. Thus the sequences

$$00000000000000000000\ldots \quad \text{or} \quad 10101010101010101010\ldots$$

do not seem to be random because they have a very strong pattern. In contrast, one feels that a sequence like

$$10010111010111100101\ldots$$

is random because it is difficult to discover patterns in it. Clarifications of precisely what constitutes *randomness* were only made in the late 20th century, when Church [28] and others argued that the notion of algorithmic randomness is closely connected to the notion of computable function. The reader may refer to the PhD thesis of Michael van Lambalgen [77] for the evolution and philosophical insights of many of these notions.

Martin-Löf [56] introduced a notion of randomness based on statistical tests. The idea is that a random sequence has to pass every conceivable and reasonable statistical test. He formalizes this notion of *reasonable statistical test* as a kind of effective sets of measure zero. He defines a sequence to be random if it avoids all such effectively presented measure zero sets.

Levin [51, 52] (involving the *monotone complexity*) and Schnorr [67] (using a variant of monotone complexity, which he called *process complexity*) and later Chaitin [23] (using the *prefix-free complexity*) introduced a notion of randomness in terms of program-size complexity. The essential idea of *prefix-free* program-size complexity was implicit in all these works. Levin-Chaitin random sequences are those whose initial segments are *algorithmically incompressible*. That is, an infinite sequence A of 0s and 1s is Levin-Chaitin random if the prefix-free program-size complexity of the first n digits of A is greater than n minus a fixed constant. Hence, we regard an infinite sequence as random if the only way to get initial segments of it is to essentially hardwire those segments into a program generating them. Thus

$$1010101010101010101010101010101010101010\ldots$$

is not random as we could describe the first $2n$ bits with a short program saying "repeat n times 10".

It is interesting that the idea of *effectiveness* is involved in both Martin-Löf and Levin-Chaitin definitions. We had to wait till the development of computability theory to come up with a formal definition of randomness. In fact, the whole algorithmic information theory is closely intertwined with computability theory.

Schnorr [66] proved that Martin-Löf randomness and Levin-Chaitin randomness notions coincide. So Levin-Chaitin randomness and Martin-Löf randomness reflect two different aspects of the same fact. This notion of randomness became quite accepted in the field and is known as *Martin-Löf randomness* or *Chaitin randomness* or *Chaitin-Kolmogorov randomness*. After this proof, the notion is generally called just *randomness*.

Any sequence of 0s and 1s can be regarded as a real number by identifying the binary fractional expansion of the real with such sequence. Chaitin [23] found a natural example of a random real number, called Ω, which represents the probability that an arbitrary program halts. In fact, this is a whole class of numbers, since the definition of Ω depends on the chosen underlying universal machine. But the property of being random is independent of this choice: for any universal machine the halting probability associated to it is always random. The value of Ω can be computably approximated from below and the word *left-c.e.* is used to denote this property. Further research [21, 48] provided the following equivalence: a real is left-c.e. and random if and only if it is the halting probability of some prefix-free universal machine.

In the opposite corner of the random sequences, are the K-trivial sequences, which are those whose initial segments are highly compressible in terms of prefix-free program-size complexity [34]. Thus, this notion is contrary to randomness, where this complexity is high. More formally, an infinite sequence A is K-trivial when the prefix-free program-size complexity of the first n bits of A is as low as it can be, that is, less than $K(n)$ plus a fixed constant.

In the remaining of this chapter we will present formally the notion of program-size complexity and two definitions that will be used along the thesis: randomness and triviality. We will limit ourselves to introduce the main definitions and results that are necessary in this document. More information on program-size complexity and applications can be found in [54] and [19]. In [32] there is a whole picture of the field, gathering both classical and very new results in the area.

1.2 Basic definitions

$\mathbb{N}$ denotes the set of natural numbers. As usual we identify a set $A \subseteq \mathbb{N}$ with its characteristic function $\chi_A \colon \mathbb{N} \to \{0,1\}$. That is, $A(x) = 1$ if $x \in A$ and $A(x) = 0$ if $x \notin A$. $\mathbb{Q}$ the set of rationals and $\mathbb{R}$ the set of reals. $\mathbb{N}^+$, $\mathbb{Q}^+$ and $\mathbb{R}^+$ are the described sets restricted to positive values. $\mathcal{P}(A)$ is the power set of A. For a real r, $\lfloor r \rfloor$ is the floor of r and $\lceil r \rceil$ is the ceiling of r.

$2^{<\omega}$ is the set of all finite strings of 0s and 1s; 2^n denotes the set of strings in $2^{<\omega}$ of length n and $2^{\leq n}$ all the words of length up to n. λ denotes the empty string For a string $\sigma \in 2^{<\omega}$, $|\sigma|$ denotes the length of σ. For $i \in \{0, \ldots, |\sigma| - 1\}$, $\sigma(i)$ denotes the i-th bit of σ. Sometimes we will consider natural numbers as strings. In this case, we use the string 0 to represent the natural number 0 and for any $n > 0$, we use the string that represents n in binary notation, starting with 1. Observe that when interpreting a number $n > 0$ as a string we have $|n| = 1 + \lfloor \log_2 n \rfloor$. We fix a recursive pairing function $\langle \cdot, \cdot \rangle \colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$. We will use the same symbol to recursively codify pairs in $\mathbb{N} \times \mathbb{N}$ or $\mathbb{N} \times 2^{<\omega}$. Let $\mathrm{str} \colon \mathbb{N} \to 2^{<\omega}$ be the standard enumeration of the strings: the string $\mathrm{str}(n)$ is that binary sequence $b_0 b_1 \ldots b_m$ for which the binary number $1 b_0 b_1 \ldots b_m$ has the value $n + 1$. Thus, $\mathrm{str}(0) = \lambda$, $\mathrm{str}(1) = 0$, $\mathrm{str}(2) = 1$, $\mathrm{str}(3) = 00$, $\mathrm{str}(4) = 01$ and so on. For $\sigma, \tau \in 2^{<\omega}$, we write $\sigma \preceq \tau$ when σ is a prefix of τ, that is $|\sigma| \leq |\tau|$ and $\sigma(i) = \tau(i)$ for all $i \in \{0, \ldots, |\sigma| - 1\}$. We write $\sigma \prec \tau$ when σ is a proper prefix of τ, that is $\sigma \preceq \tau$ and $\sigma \neq \tau$. A set of strings A is *prefix-free* if there are no two strings σ and τ in A such that $\sigma \prec \tau$. A set $T \subseteq 2^{<\omega}$ is a *tree* if for all $\sigma, \tau \in 2^{<\omega}$ if $\sigma \in T$ and $\tau \prec \sigma$ then $\tau \in T$.

The Cantor space, denoted 2^ω is the set of all infinite sequences of 0s and 1s. The infinite sequence $A \in 2^\omega$ can also be regarded as the enumeration $A(0)A(1)A(2)\ldots$ of the characteristic function of a set $A \subseteq \mathbb{N}$. Furthermore, A can also be seen as the real number in $[0,1]$ defined by $\sum_{n \geq 0} A(n) \cdot 2^{-n-1}$.

We denote by $A \upharpoonright n$ the string of length n which consists of the first n bits of A, that is, $A(0) \ldots A(n-1)$. The relation $A < B$ is transferred from numbers to sets and sequences with the additional convention, that for the two representations of numbers of the from $n \cdot 2^{-m}$ the one ending with $011111\ldots$ is below the one ending with $100000\ldots$ so that $<$ becomes a linear ordering on sets and sequences. The number of elements of the set A is denoted by $\|A\|$. For any two sets A and B we define $A \oplus B = \{2x \colon x \in A\} \cup \{2x + 1 \colon x \in B\}$. If T is a tree then $[T] \subseteq 2^\omega$ denotes the infinite branches of a tree, that is $[T] = \{X \colon (\forall \sigma \prec X)\, \sigma \in T\}$.

For any two strings σ and τ, $\sigma\tau$ is the concatenation of σ and τ. Also if $Z \in 2^\omega$ then σZ is the infinite sequence that starts with σ and follows with Z. For a string σ, we denote by $\sigma 2^\omega$ the class $\{\sigma Z \colon Z \in 2^\omega\}$. If A is a set of strings then $A2^\omega$ denotes the class $\{\sigma 2^\omega \colon \sigma \in A\}$. If $\mathcal{C}$ is a set of real numbers, the Lebesgue measure of $\mathcal{C}$ is notated $\mu(\mathcal{C})$. Hence, $\mu(\sigma 2^\omega) = 2^{-|\sigma|}$.

We usually notate strings in lowercase Greek letters. Usually, σ, τ, χ, for outputs and ρ, γ, ν for programs. Numbers will be denoted in lowercase Roman letters such as x, y, n, m, k, etc. Usually s, t, u are used for steps in computations and c, d for coding constants. Sets of natural numbers or sets of strings will be notated in uppercase Roman letters, such as A, B, X, etc. Machines will be noted in uppercase boldface Roman letters, typically, $\mathbf{M}$, $\mathbf{N}$ for specific ones, and $\mathbf{U}$, $\mathbf{V}$, $\mathbf{W}$ for universal ones. In general we use f, g, h for functions and calligraphic uppercase letters like $\mathcal{C}$, $\mathcal{T}$ for sets of reals.

1.3 Computability theory

We will work with Turing machines introduced by Turing [75] as a precise definition capturing the intuitive notion of *algorithmically computable function*. Our Turing machine architecture has a read-only input tape, a read-write working tape and a write-only output tape. Both programs and outputs will be represented with strings of 0s and 1s. A Turing machine is just a computer program used to perform a specific task: it takes a binary word ρ as input and either gets undefined (notated $\mathbf{M}(\rho)\uparrow$) or it halts (notated $\mathbf{M}(\rho)\downarrow$) and produces a certain binary string σ as output. In this last case we say that $\mathbf{M}(\rho) = \sigma$ or that ρ is an $\mathbf{M}$-*description* of σ. The *domain* of $\mathbf{M}$, notated $\operatorname{dom}\mathbf{M}$, is the set of all strings σ for which $\mathbf{M}(\sigma)\downarrow$. If $\operatorname{dom}\mathbf{M} = 2^{<\omega}$ we say that $\mathbf{M}$ is *total*, otherwise $\mathbf{M}$ is *partial*.

Nowadays the fact that programs can be treated as strings and that all of them can be listed is quite standard for a computer scientist. Moreover, there are special programs that take strings representing programs as input, and simulate them. This is a consequence of the Enumeration Theorem, which says that we can algorithmically enumerate all the Turing machines

$$\mathbf{T}_1, \mathbf{T}_2, \mathbf{T}_3, \ldots \tag{1.1}$$

and that there is a universal machine $\mathbf{V}$ such that

$$\mathbf{V}(0^{i-1}1\sigma) = \mathbf{T}_i(\sigma), \tag{1.2}$$

so $\mathbf{V}$ can simulate every other machine. Each Turing machine $\mathbf{N}$ corresponds to a number in the list (1.1) and that number is called the *Gödel number* of $\mathbf{N}$.

Any Turing machine $\mathbf{M}$ can be approximated step by step. By $\mathbf{M}_s(\rho) = \sigma$ we denote that the machine $\mathbf{M}$ on input ρ halts within s computational steps and outputs σ; by $\mathbf{M}_s(\rho)\uparrow$ we denote that $\mathbf{M}$ has not reached a halting state by stage s. At each stage s we can algorithmically determine if $\mathbf{M}$ has reached a halting state or not: if $\mathbf{M}_s(\rho)\downarrow$ then $\mathbf{M}_t(\rho)\downarrow = \mathbf{M}(\rho)$ for all $t \geq s$. We write $\mathbf{T}_{e,s}$ for $(\mathbf{T}_e)_s$.

The class of recursive functions was introduced by Kleene [44] as another definition for capturing the *algorithmically computable function* idea. The Turing machine model and the recursive functions formalism are equivalent. However recursive functions are defined from A to B, where A and B are subsets of $\mathbb{N}$, and then this equivalence is true modulo an appropriate conversion from strings to natural numbers and vice versa (Turing machines work with *strings* but recursive functions with *numbers*). When working with functions from A to B, where A and B are subsets of $\mathbb{N}$ we will generally use the symbol φ_e to denote the function computed by the e-th Turing machine which *only takes numbers and outputs numbers* (these numbers shall be represented, for example, as stated in section 1.2). When working with functions from A to B, where A and B are subsets of $2^{<\omega}$, we will use the same symbol $\mathbf{T}_e$ for denoting the e-th Turing machine (regarded as a computing agent) and the (mathematical) function it computes. The distinction between the enumeration $(\varphi_i)_{i\in\mathbb{N}}$ and $(\mathbf{T}_i)_{i\in\mathbb{N}}$ is not important for the facts that we will discuss in this section, since all the results hold for both $(\varphi_i)_{i\in\mathbb{N}}$ and $(\mathbf{T}_i)_{i\in\mathbb{N}}$.

The Recursion Theorem due to Kleene [45] says that for every computable function f we can compute an n such that $\varphi_n = \varphi_{f(n)}$. This n is usually called the *fixed point* of f. This result (together with the *s-m-n* Theorem, see [71] for more details) allows us to define computable functions which know in advance its own Gödel number. For example, when defining a machine $\mathbf{M}$ we may assume that we already know the number e such that $\mathbf{M} = \mathbf{T}_e$.

A set of natural numbers A is *computably enumerable* (c.e.) if A is the domain of φ_e for some e. W_e denotes the e-th c.e. set, that is, $W_e = \operatorname{dom}\varphi_e$ and $W_{e,s} = \{x\colon x \leq s \wedge \varphi_{e,s}(x)\downarrow\}$. The name *computably enumerable* comes from the fact that we can computably approximate the set step by step: A is c.e. if and only if there is a total computable function ψ such that $\psi(\mathbb{N}) = A$. Hence ψ enumerates all the elements of A, not necessarily in order. A c.e. approximation of the set A is a sequence $(A_s)_{s\in\mathbb{N}}$ such that $A_s \subseteq A_{s+1}$ and $\bigcup_s A_s = A$. A set A is co-c.e. if $\mathbb{N} \setminus A$ is c.e. and A is *computable* if A is c.e. and co-c.e. It is not difficult to see that A is computable if and only if there is a total computable function ψ such that $\psi(x) = A(x)$ for all $x \in \mathbb{N}$.

The famous Halting Problem due to Turing [75] says that the set

$$\{e\colon \varphi_e(e)\downarrow\} \tag{1.3}$$

is c.e. but non-computable. The notions of partial computable functions and computably enumerable sets relativize to an arbitrary set B which acts as a piece of information to use for free by Turing machines and it is called *oracle.* In the architecture of Turing machines, we may think that the information of the oracle B is coded in an extra infinite tape with a read-only head where we codify B as an infinite binary sequence. We write $\varphi_e^B(x)$ for the output of φ_e when the input is x and the oracle is B. When $B = \emptyset$ we write $\varphi_e(x)$ instead of $\varphi_e^B(x)$. If ψ is a function computable by an oracle machine and $\psi^B(x)$ is defined then this computation should have made a finite number of questions to the oracle. The *use* of ψ when the oracle is B and the input is x is defined as one plus the maximum number queried to the oracle in the computation if $\psi^B(x)$, and ∞ otherwise.

A set A is *B-c.e.* or *c.e. in B* if A is the domain of φ_e^B for some e; and A is *B-computable* or *computable in B* if there is function computable by an oracle machine ψ such that $\psi^B(x) = A(x)$ for all $x \in \mathbb{N}$. When A is B-computable we say that A is *Turing reducible* to A, denoted $A \leq_T B$. In case $A \leq_T B$ and $B \leq_T A$ we say A is Turing equivalent to B and we notate $A \equiv_T B$. The Turing reducibility is a pre-ordering of the sets of natural numbers. The *Turing degree* of A is $\deg A = \{B\colon B \equiv_T A\}$. We define, for any set A, the *jump of* A as $A' = \{e\colon J^A(e)\downarrow\}$, where $J^A(e) = \varphi_e^A(e)$. The Halting Problem (1.3), which under this notation is $\emptyset'$, can be relativized to oracles and says that A' is c.e. relative to A but $A' \not\leq_T A$. For n iterations of the jump of A we write $A^{(n)}$. From a partial function ψ computable by an oracle machine, one can effectively obtain a primitive recursive (*primitive recursive* functions are a very simple class of computable functions; see [71] for a definition) and strictly increasing function α, called a *reduction function* for ψ, such that $(\forall B)(\forall e)\, \psi^B(e) = J^B(\alpha(e))$.

The jump operator gives us a hierarchy of degrees

$$\deg \emptyset \subset \deg \emptyset' \subset \deg \emptyset'' \ldots$$

The Arithmetical Hierarchy gives a different hierarchy according to the quantifier complexity in the syntactic definition of A. For $n \geq 1$, a set A is Σ_n^0 if there is a computable relation $R(x_1, \ldots x_n, y)$ such that $y \in A$ if and only if

$$(\exists x_1)(\forall x_2)(\exists x_3)\ldots(Qx_n)\ R(x_1, \ldots x_n, y)$$

and in $(\exists x_1)(\forall x_2)(\exists x_3)\ldots(Qx_n)$ there are $n-1$ alternations of quantifiers (starting with $\exists$, and Q is $\exists$ if n is odd and $\forall$ if n is even). A set A is Π_n^0 if the same happens but starting with a leading $\forall$ and letting Q be $\exists$ or $\forall$ depending on whether n is even or odd. We say that a set is Δ_n^0 if it is both Σ_n^0 and Π_n^0. The Post Theorem says that A is Σ_{n+1}^0 if and only if A is c.e. relative to $\emptyset^{(n)}$, A is Π_{n+1}^0 if and only if A is co-c.e. relative to $\emptyset^{(n)}$ and A is Δ_{n+1}^0 if and only if $A \leq_T \emptyset^{(n)}$. The arithmetical hierarchy can be relativized to an arbitrary oracle B and then we get similar results with classes $\Sigma_n^{0,B}$, $\Pi_n^{0,B}$ and $\Delta_n^{0,B}$ which are defined in the same way as Σ_n^0, Π_n^0 and Δ_n^0 but now the relation R is B-computable (see [71, 63] for further details).

Shoenfield's Limit Lemma states that a set A is Δ_2^0 if and only if there is a total computable $f\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ such that for all x, $\lim_{s\to\infty} f(x,s)$ exists (i.e. $\{s\colon f(x,s) \neq f(x,s+1)\}$ is finite) and $\lim_{s\to\infty} f(x,s) = A(x)$. This f is usually called an *effective approximation* or *computable approximation* of A, and it is generally written $A_s(x)$ for $f(x,s)$. If A is given by an effective approximation and ψ is a function computable by an oracle machine, we write $\psi^A(e)[s]$ for $\psi_s^{A_s}(e)$.

There are weaker reducibilities besides the Turing reducibility. We say that A is *1-reducible* to B, notated $A \leq_1 B$, if there is a computable injective function f such that $x \in A \Leftrightarrow f(x) \in B$ for all x. In case $A \leq_1 B$ and $B \leq_1 A$ we say A is 1-equivalent to B and we notate $A \equiv_1 B$. The set $\{B\colon B \equiv_1 A\}$ is the 1-degree of A. We say that A is *weak truth-table reducible* to B, notated $A \leq_{wtt} B$, if there is partial function ψ computable with an oracle machine such that $\psi^B(x) = A(x)$ for all x and we can computably bound the number of queries to the oracle B, that is, there is a computable function f such that the use of $\psi^B(x)$ is at most $f(x)$. When the above ψ is total for all oracles (i.e., $\psi^Z(x)\downarrow$ for all $Z \subseteq \mathbb{N}$) we say that A is *truth-table reducible* to B, notated $A \leq_{tt} B$.

A set is Σ^0_n-*complete* if A is Σ^0_n and $B \leq_1 A$ for every Σ^0_n set B; a set is Π^0_n-*complete* if A is Π^0_n and $B \leq_1 A$ for every Π^0_n set B. $\emptyset^{(n)}$ is Σ^0_n-complete for all $n \geq 1$.

A real number r is left-c.e. if there is a computable non-decreasing sequence of rationals which converges to r, and it is right-c.e. if there is a non-increasing sequence of rationals which converges to r. A real is *computable* if it is computable when it is regarded as a set. Also a real is computable if it is right-c.e. and left-c.e. Therefore every computable real is left-c.e., but the converse is not true. It is possible that a real number r be approximated from below by a computable non-decreasing sequence of rationals but that there is no function which effectively gives each of the fractional digits of r. This happens when it is not possible to give an effective bound of the error of approximating the number by any computable sequence of rationals.

Some classical references in computability theory are [30, 43, 63, 71, 80].

1.4 Program-size complexity

In this section we introduce the plain and prefix-free program-size complexity and we mention some known results and standard notation that will be used repeatedly in the rest of this thesis. Some other important results will be introduced when necessary.

In algorithmic information theory, programs are regarded as algorithmic descriptions of strings. As we mentioned in section 1.3, we say that a program ρ $\mathbf{M}$-*describes* a string σ when it is executed in a Turing machine $\mathbf{M}$ and produces σ as output. In general there is more than one program describing the same output σ. The idea of the program-size complexity introduced in [46, 72, 23] is to take the length of a shortest description as a measure of the complexity of the string.

Definition 1.4.1 (Plain program-size complexity). $C_{\mathbf{M}}\colon 2^{<\omega} \to \mathbb{N}$, the *plain program-size complexity with respect to the Turing machine* $\mathbf{M}$, is defined as

$$C_{\mathbf{M}}(\sigma) = \begin{cases} \min\{|\rho| \colon \mathbf{M}(\rho) = \sigma\} & \text{if } \sigma \text{ is in the range of } \mathbf{M};\\ \infty & \text{otherwise.}\end{cases}$$

If we restrict ourselves to partial computable functions with prefix-free domain, we can think of *prefix-free Turing machines* (independently introduced by Levin [50, 51], Schnorr [66] and Chaitin [23]) as the formalism to compute them. Chaitin thought of a convenient device to compute exactly all prefix-free machines. This device is just like a classical Turing machines but the input tape is a bit primitive: the input head moves only to the right and there is no blank end-marker delimiting the end of the input. So a prefix-free machine starting with the input head in the first bit of the input ρ may eventually try to read beyond the end of ρ and if this happens, it simply crashes (i.e. it becomes undefined) and it stays in that error state forever. Therefore, for a prefix-free machine $\mathbf{M}$ to succeed in the computation, it needs to somehow figure out by itself where the end of the input is. This is the reason why this device is sometimes called *self-delimiting* machine. In this model, if a given input is defined, all its extensions are undefined. It can be shown that this restricted architecture of Turing machines compute exactly all partial computable functions with prefix-free domain. The *the prefix-free program-size complexity* [52, 40, 23] is defined in the same way as in Definition 1.4.1 but relative to prefix-free machines:

Definition 1.4.2 (Prefix-free program-size complexity). $K_{\mathbf{M}}\colon 2^{<\omega} \to \mathbb{N}$, the *prefix-free program-size complexity with respect to the prefix-free machine* $\mathbf{M}$, is defined as

$$K_{\mathbf{M}}(\sigma) = \begin{cases} \min\{|\rho|\colon \mathbf{M}(\rho) = \sigma\} & \text{if } \sigma \text{ is in the range of } \mathbf{M}; \\ \infty & \text{otherwise.} \end{cases}$$

The two complexity functions C and K have been used for different purposes (for more details, see [32] and [54]). In this thesis we will mainly use prefix-free program-size complexity, but in some sections we will also work with the plain version.

There are two traditions of notation in the program-size complexity community as regards the plain/prefix-free versions: the C/K tradition and the K/H tradition. In recent years it seems that the C/K notation appears more often in both the complexity and the recursion theory literature. In this document, we follow the C/K notation.

Observe that it does not matter if the list in (1.1) is an enumeration of all classical machines or all prefix-free machines (both have a finite table of instructions, so they can be effectively listed). The machine $\mathbf{V}$ defined in (1.2) exists in both cases and in the second, $\mathbf{V}$ is also prefix-free.

The classical notion of universality from computability theory is redefined in the algorithmic information theory. We will use the word *universal* for denoting machines like $\mathbf{V}$ defined in (1.2) that somehow simulate every other machine at no noticeable extra cost in the length.

Definition 1.4.3 (Universality). $\mathbf{U}$ is a *universal* prefix-free (resp. classical) Turing machine if and only if

$$(\forall e)(\exists c_e)(\forall \rho)(\exists \gamma_{e,\rho})\,[\mathbf{U}(\gamma_{e,\rho}) = \mathbf{T}_e(\rho)\ \wedge |\gamma_{e,\rho}| \leq |\rho| + c_e],$$

where $\mathbf{T}_0, \mathbf{T}_1, \mathbf{T}_2, \ldots$ is an enumeration of all the prefix-free (resp. classical) Turing machines

When working with oracles, $\mathbf{U}$ will be universal if it may simulate *any* other machine with *any* oracle, so it is *universally universal.* In the case of an prefix-free universal machine, $\mathbf{U}^A$ will be prefix-free for all $A \subseteq \mathbb{N}$.

In general we use the same letter $\mathbf{U}$ for denoting both a classical universal machine and a prefix-free universal machine; it will always be clear from the context which one we refer to. In [23] Chaitin called these machines *optimal universal*, because they permit us to define the program-size complexity in an *optimal* way:

Theorem 1.4.4 (Invariance)**.** *If* $\mathbf{U}$ *is universal prefix-free (resp. classical) Turing machine then for any prefix-free (resp. classical) Turing machine* $\mathbf{M}$ *there is a constant* c *such that for all* $\sigma \in 2^{<\omega}$, $K_{\mathbf{U}}(\sigma) \leq K_{\mathbf{M}}(\sigma) + c$ *(resp.* $C_{\mathbf{U}}(\sigma) \leq C_{\mathbf{M}}(\sigma) + c$*).*

Hence for any two prefix-free universal machines $\mathbf{U}$ and $\mathbf{V}$, the prefix-free program-size complexity with respect to $\mathbf{U}$ and $\mathbf{V}$ is the same up to an additive constant, i.e. there is some constant c such that $|K_{\mathbf{U}}(\sigma) - K_{\mathbf{V}}(\sigma)| \leq c$ for any $\sigma \in 2^{<\omega}$. Of course the same is true for classical machines and C. If there is no need to refer to the underlying universal machine $\mathbf{U}$, one just writes K for $K_{\mathbf{U}}$ and C for $C_{\mathbf{U}}$. In this way, K and C become an *absolute* measure of the complexity of the strings.

To illustrate some useful application of the Invariance Theorem, let us see some upper bounds for K and C. Imagine a classical machine $\mathbf{M}$ that reads the input σ and writes σ as its own output. Then $C_{\mathbf{M}}(\sigma) = |\sigma|$ for all $\sigma \in 2^{<\omega}$ and by the Invariance Theorem 1.4.4 this shows that for all σ, $C(\sigma) \leq |\sigma| + c$ for some constant c. With prefix-free machines the situation is a bit different because the machine has to discover by itself where the end of the input is. Suppose $\mathbf{U}$ is any universal prefix-free machine and suppose ρ_σ is a minimal $\mathbf{U}$-description of $|\sigma|$. This just means that $\mathbf{U}(\rho_\sigma) = |\sigma|$ and $|\rho_\sigma| = K(|\sigma|)$. Then there is a prefix-free machine $\mathbf{N}$ that with input $\rho_\sigma\sigma$ can do the following: fist simulate $\mathbf{U}$ step by step. Each time $\mathbf{U}$ asks for reading one more bit, $\mathbf{N}$ obeys and reads from its own input. Eventually $\mathbf{U}$ will read the whole ρ_σ and will terminate with $\mathbf{U}(\rho_\sigma) = |\sigma|$ as output. Now $\mathbf{N}$ (which is still running) knows that exactly $|\sigma|$ more bits need to be read from the input. So $\mathbf{N}$ reads $|\sigma|$ bits from the input and recovers σ. Afterwards, $\mathbf{N}$ writes σ in the output and terminates. This shows that $\mathbf{N}(\rho_\sigma\sigma) = \sigma$. Of course, if the input is wrong, the computation may go wrong, for example $\mathbf{N}$ can try to read beyond the end of the input and then crashes. However, when the input is of the form $\rho_\sigma\sigma$, $\mathbf{N}$ always outputs σ. By the invariance Theorem 1.4.4 this shows that $K(\sigma) \leq |\rho_\sigma\sigma| + d = K(|\sigma|) + |\sigma| + d$ for some constant d. These upper bounds on C and K will be repeatedly used in the thesis. We will also use that for any n there is a string σ of length n such that $K(\sigma) \geq n$. This is true because there are at most $2^n - 1$ programs of length less than n but 2^n strings of length n. The same holds for C. In fact, Chaitin [23, 26] showed that there is a constant c such that for all $d \in \mathbb{N}$ and all n

$$\|\{\sigma\colon |\sigma| = n \wedge K(\sigma) \leq n + K(n) - d\}\| \leq 2^{n-d+c}$$

This result is usually called Counting Theorem, tells us that only a small fraction of all the strings of length n have prefix-free program-size complexity below $n+K(n)-d$, when we take d much bigger than c. Observe that if we let $d = n - b$, we obtain

$$\|\{\sigma\colon |\sigma| = n \wedge K(\sigma) \leq K(n) + b\}\| \leq 2^{b+c}$$

and this last upper bound depends on b and c, but not on n.

As we explained in the last paragraph, one way to bound the program-size complexity is to explicitly design specific machines, like $\mathbf{M}$ or $\mathbf{N}$ and explain their behavior. Another powerful method to implicitly build prefix-free machines and upper bound the prefix-free program-size complexity is via a Kraft-Chaitin set. This method consists of an effective interpretation of an inequality of Kraft [47]:

Definition 1.4.5 (Kraft-Chaitin set)**.** A c.e. set

$$W = \{\langle n_0, \sigma_0\rangle, \langle n_1, \sigma_1\rangle, \langle n_2, \sigma_2\rangle, \dots\},$$

where $n_i \in \mathbb{N}$ and $\sigma_i \in 2^{<\omega}$, is a *Kraft-Chaitin set* if $\mathrm{wt}\,(W) \leq 1$, where $\mathrm{wt}\,(W) = \sum_{i\in\mathbb{N}} 2^{-n_i}$ is the *weight* of W. The pairs enumerated into such a set W are called *axioms*.

The Kraft-Chaitin Theorem can be found in Levin's Thesis [50] and in [52], Schnorr also included a version of it in [67] and Chaitin [23] gave the first proof explicitly for prefix-free program-size complexity. This theorem states that from a Kraft-Chaitin set W like the one described in the above Definition 1.4.5, we can effectively obtain a prefix-free machine $\mathbf{M}$ such that for each i there is a ρ_i of length n_i with $\mathbf{M}(\rho_i) \downarrow = \sigma_i$, and $\mathbf{M}(\gamma) \uparrow$ unless $\gamma = \rho_i$ for some i. Observe that the machine $\mathbf{M}$ is built in an implicit way: we only need to specify the lengths of the programs we want. In particular, the Kraft-Chaitin Theorem states that there is a constant c such that for all i, $K(\sigma_i) \leq \min\{m\colon \langle m, \sigma_i\rangle \in W\} + c$.

The complexity K is related with the probability that a prefix-free machine $\mathbf{M}$ outputs a given string σ. This last magnitude is defined as

$$\begin{aligned} P_{\mathbf{M}}(\sigma) &= \mu\left(\{\rho\colon \mathbf{M}(\rho) = \sigma\}2^{\omega}\right) \\ &= \sum_{\rho\colon \mathbf{M}(\rho)=\sigma} 2^{-|\rho|}. \end{aligned} \tag{1.4}$$

Clearly, for all σ, $2^{-K(\sigma)} \leq P_{\mathbf{U}}(\sigma)$, since a shorter $\mathbf{M}$-description of σ already contributes to the sum in equation (1.4). The following result, due to Levin [52], Gács [40] and Chaitin, [23] known as the *Coding Theorem*, states that the other inequality is also true, except for a multiplicative constant

Theorem 1.4.6 (Coding Theorem)**.** *For each prefix-free machine* $\mathbf{M}$*, one may effectively find a constant* c *such that* $(\forall\sigma \in 2^{<\omega})\ 2^{c-K(\sigma)} \geq P_{\mathbf{M}}(\sigma)$.

For more information in the Coding Theorem, see [54] and [32].

Although K and C are not computable functions, they can be computably approximated from above using the step by step approximation of $\mathbf{U}$. $K_s(\sigma)$ is the approximation at stage s of $K(\sigma)$ defined by $K_s(\sigma) = \min\{|\rho|\colon \mathbf{U}_s(\rho) = \sigma\} \cup \{\infty\}$. Observe that $K_s(\sigma) \to K(\sigma)$ when $s \to \infty$. The same holds for C_s.

Machines can be relativized to oracles. For any $\mathbf{M}^A$, a prefix-free machine $\mathbf{M}$ relativized to an oracle A, $\operatorname{dom}\mathbf{M}^A$ has to be prefix-free. The Definition 1.4.2 of program-size complexity can be relativized to oracles:

Definition 1.4.7 (Relativized prefix-free program-size complexity)**.** $K^A_{\mathbf{M}}\colon 2^{<\omega} \to \mathbb{N}$, the *prefix-free program-size complexity with respect to the prefix-free machine* $\mathbf{M}$ *and relative to oracle* A, is defined as

$$K^A_{\mathbf{M}}(\sigma) = \begin{cases} \min\{|\rho|\colon \mathbf{M}^A(\rho) = \sigma\} & \text{if } \sigma \text{ is in the range of } \mathbf{M}^A; \\ \infty & \text{otherwise.} \end{cases}$$

Definition 1.4.1 of $C_{\mathbf{M}}$ also adapts straightforwardly to $C^A_{\mathbf{M}}$. Again, when there is no need to refer to the underlying universal machine, we just write C^A and K^A. Having an oracle A gives more power for compressing strings, so K^A is always smaller than K up to an additive constant. The same holds for C^A and C.

1.5 Randomness

In this section we introduce the equivalent definitions of randomness due to Martin-Löf and Levin-Chaitin that we will use all over the present research.

A collection of reals $\mathcal{A}$ that can be effectively enumerated is a Σ^0_1 class. More formally, $\mathcal{T}$ is a Σ^0_1 class if and only if $\mathcal{T} = \{\sigma 2^\omega \colon \sigma \in W\}$ for some c.e. prefix-free set $W \subseteq 2^{<\omega}$. Martin-Löf [56] gave a definition of a random real using constructive measure theory.

Definition 1.5.1 (Martin-Löf test and Martin-Löf randomness [56])**.** A *Martin-Löf test* is a sequence $(\mathcal{U}_n)_{n\in\mathbb{N}}$ of uniformly c.e. Σ^0_1 classes such that $\mu(\mathcal{U}_n) \leq 2^{-n}$. We say that a real A *passes* the Martin-Löf test $(\mathcal{U}_n)_{n\in\mathbb{N}}$ if $A \notin \bigcap_{n\in\mathbb{N}} \mathcal{U}_n$. A is *Martin-Löf random* if and only if A passes every Martin-Löf test.

The intuitive idea is that Martin-Löf random reals cannot be covered by any effectively presented set of measure 0. A sequence is Martin-Löf random when it avoids all conceivable effectively presented properties of stochasticity.

The first characterization of randomness using algorithmic complexity was due to Levin [51, 52] and involved what are called *monotone machines* and *monotone complexity*. His idea was to assign a complexity to the real itself, working with infinite computations (that is, computations that output an infinite sequence of 0s and 1s). This monotone complexity was related to Schnorr's *process complexity* [67], which used a different kind of monotone machines. Independently, Chaitin [23] defined random sequences using the prefix-free program-size complexity. All this three notions coincide and in this section we mention the prefix-free complexity that will be used all along this thesis. for detailed historical remarks, see refer to [54].

Definition 1.5.2 (Levin-Chaitin randomness [52, 23])**.** A real A is *Levin-Chaitin random* if and only if there is a constant c such that $(\forall n)\, K(A \upharpoonright n) > n - c$.

The idea of a Levin-Chaitin random real A is that the prefixes of A are hard to describe with a program. We need at least n bits to describe the first n bits of A, and then A is algorithmically incompressible. From the Invariance Theorem 1.4.4 it follows that the above definition of Levin-Chaitin randomness does not depend on the underlying universal machine. Martin-Löf showed that we cannot replace K by C in the above Levin-Chaitin's definition of randomness because we end up with an empty definition: there are no reals A such $(\forall n)\, C(A \upharpoonright n) > n - c$.

Schnorr [66] proved that A is Levin-Chaitin random if and only if A is Martin-Löf random. We will call this notion just *randomness*.

In [23] Chaitin proposed the following experiment: given a prefix-free machine $\mathbf{M}$, execute it step by step. Each time the machine $\mathbf{M}$ requests a new bit from the input, we give it a 0 or 1 bit depending on the result of tossing a coin. Now consider the probability that the machine $\mathbf{M}$ halts:

$$\Omega_{\mathbf{M}} = \sum_{\mathbf{M}(\rho)\downarrow} 2^{-|\rho|} = \mu\,(\operatorname{dom}\mathbf{M}2^{\omega})\,. \tag{1.5}$$

Chaitin called this magnitude the *halting probability of* $\mathbf{M}$. For example, if machine $\mathbf{M}$ halts only on strings of the form 0^i1, for i even, then $\Omega_{\mathbf{M}} = 0.1010101\cdots = 2/3$. Observe that, since the domain of $\mathbf{M}$ is c.e., $\Omega_{\mathbf{M}}$ can be computably approximated from below by a sequence of rationals, so $\Omega_{\mathbf{M}}$ is a left-c.e. real.

Martin-Löf has shown in a constructive measure theoretical sense that almost all sequences are random, that is, the set $\{Z \in 2^{\omega} \colon Z \text{ is random}\}$ has Lebesgue measure 1. However, to give examples of single random reals has not been easy. As a consequence of the definition of randomness, no computable real can be random. In [23], Chaitin gives the first example: the halting probability of any universal prefix-free machine is random. This means that $\Omega_{\mathbf{U}}$ is random regardless of the universal prefix-free machine $\mathbf{U}$. When the underlying universal prefix-free machine is fixed, we simply write Ω. Chaitin also proved that the computational power of Ω is equivalent to the Halting Problem, that is $\Omega \equiv_T \emptyset'$.

The notion of randomness can easily be relativized to oracles: A is *random relative to* B if and only if there is a constant c such that $(\forall n)\, K^B(A \upharpoonright n) > n - c$. Since there is a constant d such that $(\forall B)(\forall \sigma)\, K^B(\sigma) \leq K(\sigma) + d$, if A is a random real relative to B then A is already random. We say that A is n-random when it is random relative to $\emptyset^{(n-1)}$.

1.6 K-triviality

If randomness means algorithmically incompressible, the anti-random sequences are those which are highly compressible, in the sense that the prefix-free program-size complexity of $A \upharpoonright n$ is about the same as the program-size complexity of n (regarded as a string). These sequences have been studied in [24, 73] and are called K-trivial:

Definition 1.6.1 (K-triviality)**.** A is K-trivial if and only if there is a constant c such that $(\forall n)\, K(A \upharpoonright n) \leq K(n) + c$.

All computable sequences are K-trivial because given n one can compute the first n bits of A. If instead of considering K, in Definition 1.6.1 we consider C then Chaitin [24] proved that such sequences coincide with the computable sequences. Chaitin [24] also showed that if a sequence is K-trivial then it is Δ^0_2; and Downey, Hirschfeldt, Nies and Stephan [34] (after the first construction Solovay [73], later adapted by Zambella [81] and Calude and Coles [20]) showed that there are K-trivial reals which are not computable.

A is low for K when it cannot be used to further compress strings:

Definition 1.6.2 (Low for K)**.** The real A is *low for* K if and only if there is a constant c such that $(\forall \sigma)\, K(\sigma) \leq K^A(\sigma) + c$.

Hence, if A is low for K then K and K^A differ at most by a constant. Nies [62] showed that A is K-trivial if and only if A is low for K. This means that being computationally weak as an oracle for compressing strings is the same as being algorithmically incompressible.

For more information on K-triviality, see [34, 31] and for more characterizations of K-triviality (like being low for K), see [62].

2. ABSOLUTELY NORMAL NUMBERS

In this chapter we give a computable reformulation of Sierpinski's example of an absolutely normal number of 1916. We also reconstruct an unpublished manuscript of Turing on normal numbers, which is incomplete and quite hard to read. Both computable versions of Sierpinski's and Turing's works lead lo examples of absolutely normal numbers that are also computable. Based in these constructions, we prove that there is an absolutely normal number in each 1-degree.

This chapter comprises joint work [7] with Verónica Becher and joint work [11] with Verónica Becher and Rafael Picchi.

2.1 Introduction

Random sequences (and other variants of randomness) have various properties of stochasticity from classical probability theory. For example, every random real is *normal* to the scale of t, in the sense that no block of digits is more frequent than another of the same size, when the sequence is regarded as a real number between 0 and 1 written to some scale (or base) t. One can write the same real number to other scale t' and check normality to the new scale t'. A real number is *absolutely normal* when it is normal to every scale $t \geq 2$.

Examples of random reals are automatically examples of non-computable absolutely normal numbers. Surprisingly, almost all (in the sense of Lebesgue measure) real numbers are absolutely normal. This result without a constructive proof dates back to 1909, and was proven by Émile Borel [16]. However there are very few known examples of absolutely normal numbers (see [17, 7, 49, 41]). The problem of giving concrete examples was raised by Borel in [16] as soon as he introduced the definition of normality.

The first example of an absolutely normal number was given by Sierpinski in 1916 [69], twenty years before the concept of computability was formalized. Sierpinski determines such a number using a construction of infinite sets of intervals and using the minimum of an uncountable set. Thus, it is a priori unclear whether his number is computable or not. In the following sections we give a computable reformulation of Sierpinski's construction and obtain a computable absolutely normal number. Unfortunately, the algorithm to produce this number is highly exponential. We also present a reconstruction of an unpublished manuscript of Turing on normal numbers, which also leads to an exponential algorithm.

The problem of giving examples of numbers that are *normal to a given scale* has been more successful. There are fast algorithms to produce several examples, and these examples have analytic formulations such as a conveniently defined series. For instance, Champernowne's number [27] and its generalization given by Copeland and Erdös [29], or the Stoneham class of normals to given scales and its generalization by Bailey and Crandall [4] to reals of the form $\sum_i \frac{1}{b^{m_i} c^{n_i}}$, for certain sequences $(m_i), (n_i)$.

In section 2.2 we give the formal definitions of normality and absolute normality. In section 2.3 we present Sierpinski's original construction and in section 2.4 we introduce our algorithmic version of his construction. We discuss Sierpinski's number and we consider other variants defining absolutely normal numbers in section 2.5. In sections 2.6, 2.7 and 2.8 we include the reconstruction of Turing's unpublished manuscript, leading to another algorithm for computing absolutely normal numbers. Finally in section 2.9 we mention some applications of our constructions: we give a simple proof of the fact that all Schnorr random reals are absolutely normal and we prove that there is an absolutely normal number in every Turing degree.

Whenever possible, we try to keep the notation used in Sierpinski's and Turing's original works. This is why some letters used to refer to scales, digits, words, etc. change from section 2.6 with respect to the previous one.

2.2 Definition of normality and absolute normality

The idea of normal numbers is that every digit and block of digits appears equally frequent in its fractional expansion. Of course, this definition is dependent of a given scale. Absolute normal numbers are normal in every scale. In the following we give the precise definitions of normality and absolute normality.

Let q be an integer greater than or equal to 2. The elements in $\{0,\dots,q-1\}$ are referred to as *digits in the scale of* q. A word in the scale of q is a finite sequence of digits in the scale of q. The set of all words of length k in the scale of q is denoted by $\{0,\dots,q-1\}^k$. We say that a word γ in the scale of q *occurs* in a word σ in the scale of q *at position* i, $0 \leq i < |p|$, if

$$\sigma(i)\ \sigma(i+1)\ \dots\ \sigma(i+|\gamma|-1) = \gamma;$$

a word γ *occurs* in σ if it occurs at some position.

Definition 2.2.1. Let α be any real in $(0,1)$. We denote by $Q(\alpha,q,\gamma,n)$ the number of occurrences of the word γ in the first n digits after the fractional point in the expansion of α in the scale of q.

Definition 2.2.2 (Normality and absolute normality)**.** α is *normal* in the scale of q if for every word γ in the scale of q,

$$\lim_{n\to\infty} \frac{Q(\alpha,q,\gamma,n)}{n} = \frac{1}{q^{|\gamma|}}.$$

α is *absolutely normal* if it is normal to every scale $t \geq 2$.

For example, the rational number

$$0.1010101010101010101\dots \qquad \text{(written in the scale of 2)},$$

is not normal in the scale of 2 because although the probability to find "1" is 1/2 and so is the probability to find "0", the probability to find "11" is not 1/4. In fact, no rational is absolutely normal: a/b with $a < b$ may be written in the scale of b as

$$0.a00000000000\ldots$$

This follows from the fact that $a/b = a \cdot b^{-1} + 0 \cdot b^{-2} + 0 \cdot b^{-3} \ldots$ Moreover, every rational r is not normal in any scale $q \geq 2$ [55]: the fractional expansion of r written in the scale of q will eventually repeat, say with a period of k digits, in which case the number r is about as far as being normal in the scale of q^k as it can be.

Of course, there are also irrational numbers that are not normal in some scale, for example

$$0.101001000100001000001\ldots \qquad \text{(written in the scale of 2)},$$

is not normal in the scale of 2. Champernowne's number [27]

$$0.123456789101112131415\ldots \qquad \text{(written in the scale of 10)},$$

(which has all natural numbers in their natural order, written in the scale of 10) is normal in the scale of 10, but not in some other scales.

A famous example of an absolutely normal but not computable real number is Chaitin's random number Ω, the halting probability of a universal machine [23] defined in (1.5). As explained in sections 1.4 and 1.5, we can formalize the notion of lack of structure and unpredictability in the fractional expansion of a real number, obtaining a definition of randomness stronger than statistical properties of randomness. Although the definition of Ω is known there is no algorithm to exhibit its fractional digits. That is, Ω is not computable.

The fundamental constants, like π, $\sqrt{2}$ and e, are computable and it is widely conjectured [70, 3] that they are absolutely normal. However, none of these has even been proven to be normal in the scale of 10, much less in all scales. The same has been conjectured of the irrational algebraic numbers [3]. In general, we lack an algorithm that decides on absolute normality.

Definition 2.2.2 is not the only one we can find in literature. In [16] Borel defines normality of real numbers as follows:

Definition 2.2.3 (Absolute normality as defined by Borel)**.** The real $\alpha \in (0, 1)$ is *simply normal* in the scale of t if for every digit $p \in \{0, \ldots, q-1\}$,

$$\lim_{n\to\infty} \frac{Q(\alpha, q, p, n)}{n} = \frac{1}{t},$$

and we say that α is *absolutely normal à la Borel* if it is simply normal in every scale $t \geq 2$.

Based just on symbols instead of words, Definition 2.2.3 does not allow any overlapping. Although the condition imposed in Definition 2.2.3 seems to be weaker than that of Definition 2.2.2, the are known to be equivalent:

Theorem 2.2.4. *α is absolutely normal if an only if it is absolutely normal à la Borel.*

A nice proof of the equivalence of the two definitions can be read in [42, Theorem 1.3, pp. 5–7]. Sierpinki's work is based on this last Definition 2.2.3 of normality, while Turing's manuscript (studied from section 2.6 of this chapter) is based in the first Definition 2.2.2.

Stochasticity, as conceived by von Mises [78], Church [28] and Wald [79], is related with randomness. If the reader is interested in this topic, refer to the work of van Lambalgen [77], Ambos-Spies [1], Merkle [57], Ambos-Spies, Mayordomo, Wang and Zheng [2] and Durand and Vereshchagin [36] . For a complete compendium of the results in this area, see [32].

2.3 Sierpinski's result of 1916

Sierpinski [69] achieves an elementary proof of an important proposition proved by E. Borel that states that almost all real numbers are absolutely normal [16]. At the same time he gives a way to effectively determine one such number. Following his notation, he defines $\Delta(\varepsilon) \subseteq \mathcal{P}(\mathbb{R})$ as a set of certain open intervals with rational end points. Although the set $\Delta(\varepsilon)$ contains countably many intervals, they do not cover the whole $(0, 1)$ interval. Sierpinski proves that every real number in $(0, 1)$ that is external to every interval of $\Delta(\varepsilon)$ is absolutely normal.

$\Delta(\varepsilon)$ is defined as the union of infinitely many sets of intervals $\Delta_{q,m,n,p}$.

Definition 2.3.1. The set of reals $\Delta(\varepsilon)$ is

$$\Delta(\varepsilon) = \bigcup_{q\geq 2} \bigcup_{m\geq 1} \bigcup_{n\geq n_{m,q}(\varepsilon)} \bigcup_{p=0}^{q-1} \Delta_{q,m,n,p}$$

The parameter ε is a number in $(0, 1]$ used to bound the measure of $\Delta(\varepsilon)$. In the above definition the variable q ranges over all possible scales, n ranges over the lengths of fractional expansions, p ranges between 0 and $q - 1$, m allows for arbitrarily small differences in the rate of appearance of the digit p in the fractional expansions.

Definition 2.3.2. $\Delta_{q,m,n,p} \subseteq \mathcal{P}(\mathbb{R})$ is the set of all open intervals of the form

$$\left(\frac{b_1}{q} + \frac{b_2}{q^2} + \cdots + \frac{b_n}{q^n} - \frac{1}{q^n}, \frac{b_1}{q} + \frac{b_2}{q^2} + \cdots + \frac{b_n}{q^n} + \frac{2}{q^n}\right)$$

such that

$$\left|\frac{Q_p(b_1, b_2, \ldots, b_n)}{n} - \frac{1}{q}\right| \geq \frac{1}{m} \tag{2.1}$$

where $0 \leq b_i \leq q - 1$ for $1 \leq i \leq n$ and $Q_p(b_1, b_2, \ldots, b_n)$ represents the number of times that the digit p appears amongst $b_1, b_2, \ldots, b_n$.

The idea is that $\Delta_{q,m,n,p}$ contains all numbers that are not normal in the scale of q. If a number is normal in the scale of q we expect that the rate of appearance of the digit p in a prefix of length n to be as close as possible to $1/q$. Each interval in $\Delta_{q,m,n,p}$ contains all numbers that written in the scale of q start with $0.b_1\, b_2 \ldots b_n$ and the digit p appears in $0.b_1\, b_2 \ldots b_n$ at a rate far from $1/q$ -here *far from* means with a difference greater than or equal to $1/m$. Let us observe that the right end of the intervals in $\Delta_{q,m,n,p}$ add $2/q^n$: having added only $1/q^n$ would leave the number

$$0.b_1\, b_2 \ldots b_n\, (q-1)\,(q-1)\,(q-1) \cdots = \frac{b_1}{q} + \frac{b_2}{q^2} + \cdots + \frac{b_n}{q^n} + \frac{1}{q^n}$$

outside the open interval, and this number must be included in the interval. Clearly, each interval in $\Delta_{q,m,n,p}$ has measure $3/q^n$ and for fixed q, m, n, $\Delta_{q,m,n,p}$ is a finite set of intervals.

From Sierpinski's proof, it follows that $n_{m,q}(\varepsilon)$ must be large enough as to imply $\mu\left(\Delta(\varepsilon)\right) < \varepsilon$; $n_{m,q}(\varepsilon) = \lfloor 24m^6q^2/\varepsilon \rfloor + 2$ suffices. In order to bound the measure of $\Delta(\varepsilon)$, Sierpinski works with the sum of the measures of each interval of $\Delta_{q,m,n,p}$:

Definition 2.3.3.

$$s(\varepsilon) = \sum_{q\geq 2} \sum_{m\geq 1} \sum_{n\geq n_{m,q}(\varepsilon)} \sum_{p=0}^{q-1} \sum_{\mathcal{I}\in\Delta_{q,m,n,p}} \mu\left(\mathcal{I}\right)$$

He proves that

$$\mu\left(\Delta(\varepsilon)\right) \leq s(\varepsilon) < \varepsilon$$

for every $\varepsilon \in (0,1]$. We are abusing notation here, because $\Delta(\varepsilon)$ is not a set of reals but a set of intervals. For simplicity, for any set of intervals $\mathcal{C}$ we will write $\mu\left(\mathcal{C}\right)$ for $\mu\left(\bigcup_{\mathcal{I}\in\mathcal{C}} \mathcal{I}\right)$. Of course, $\mu\left(\mathcal{C}\right) \leq \sum_{\mathcal{I}\in\mathcal{C}} \mu\left(\mathcal{I}\right)$.

$E(\varepsilon)$ is defined as the set of all real numbers in $(0,1)$ external to every interval of $\Delta(\varepsilon)$, i.e. $E(\varepsilon) = (0,1) \setminus \Delta(\varepsilon)$ and he proves that -thanks to the definition of $\Delta(\varepsilon)$- for every $\varepsilon \in (0,1]$, every real in $E(\varepsilon)$ is absolutely normal. Since $\mu\left(E(\varepsilon)\right) > 1-\varepsilon$, for every ε in $(0,1]$, every real in $(0,1)$ is absolutely normal with probability 1. This was E. Borel's result that almost all (in the sense of measure theory) real numbers are absolutely normal.

Although the measure of $E(\varepsilon)$ is greater than $1-\varepsilon$, no interval can be completely included in $E(\varepsilon)$. If this happened there would be infinitely many rationals belonging to $E(\varepsilon)$, contradicting absolute normality. Thus, $E(\varepsilon)$ is a set of infinitely many isolated irrational points. Sierpinski defines $\xi = \min(E(1))$, and in this way he determines a particular absolutely normal number.

2.4 Computing an absolutely normal number

Our work is based on one essential observation: we can give a computable enumeration of Sierpinski's set $\Delta(\varepsilon)$, and we can bound the error measure in each step. To simplify notation, we fix a rational $\varepsilon \in (0,1/2]$ (in fact, ε can be any computable real in $(0,1/2]$) and we rename $\Delta = \Delta(\varepsilon)$; $s = s(\varepsilon)$; $n_{m,q} = n_{m,q}(\varepsilon)$. We define the following computable sequence $(\Delta_k)_{k\in\mathbb{N}^+}$:

Definition 2.4.1. For any k, The set of reals Δ_k is

$$\Delta_k = \bigcup_{q=2}^{k+1} \bigcup_{m=1}^{k} \bigcup_{n=n_{m,q}}^{kn_{m,q}} \bigcup_{p=0}^{q-1} \Delta_{q,m,n,p}.$$

We also define the natural upper bound to the measure of each term in the sequence:

Definition 2.4.2. For any k, the real s_k is

$$s_k = \sum_{q=2}^{k+1} \sum_{m=1}^{k} \sum_{n=n_{m,q}}^{kn_{m,q}} \sum_{p=0}^{q-1} \sum_{\mathcal{I}\in\Delta_{q,m,n,p}} \mu\left(\mathcal{I}\right).$$

It is clear that $\lim_{k\to\infty} s_k = s$ and $\bigcup_{k\in\mathbb{N}} \Delta_k = \Delta$. Since s_k is the sum of the measures of all intervals belonging to Δ_k, we have $\mu\left(\Delta_k\right) \leq s_k$ and similarly we have $\mu\left(\Delta\right) \leq s$. Furthermore, for every natural k, $s_k \leq s$. Let us observe that for every pair of natural numbers k and l such that $k \leq l$, we have $\Delta_k \subseteq \Delta_l$, and for any k, $\Delta_k \subseteq \Delta$. Finally, we define the error of approximating s by s_k, $r_k = s - s_k$. We can give an upper bound on r_k, a result that makes our construction computable.

Theorem 2.4.3. *For every $k \geq 1$, $r_k < \frac{5\varepsilon}{2k}$.*

Proof. Let us define

$$S_{q,m,n} = \sum_{p=0}^{q-1} \sum_{\mathcal{I}\in\Delta_{q,m,n,p}} \mu\left(\mathcal{I}\right).$$

By splitting the sums in the definition of s, we obtain

$$\begin{aligned} s \;=\; & \sum_{q=2}^{k+1}\sum_{m=1}^{k}\sum_{n=n_{m,q}}^{kn_{m,q}} S_{q,m,n} + \sum_{q=2}^{k+1}\sum_{m=1}^{k}\sum_{n\geq kn_{m,q}+1} S_{q,m,n} + \\ & \sum_{q=2}^{k+1}\sum_{m\geq k+1}\sum_{n\geq n_{m,q}} S_{q,m,n} + \sum_{q\geq k+2}\sum_{m\geq 1}\sum_{n\geq n_{m,q}} S_{q,m,n}. \end{aligned} \tag{2.2}$$

But the first term of (2.2) is s_k, so the rest is r_k.

$$\begin{aligned} r_k \;=\; & \sum_{q=2}^{k+1}\sum_{m=1}^{k}\sum_{n\geq kn_{m,q}+1} S_{q,m,n} + \sum_{q=2}^{k+1}\sum_{m\geq k+1}\sum_{n\geq n_{m,q}} S_{q,m,n} + \\ & \sum_{q\geq k+2}\sum_{m\geq 1}\sum_{n\geq n_{m,q}} S_{q,m,n}. \end{aligned} \tag{2.3}$$

We bound each of the three terms that appear in the above equation. From Sierpinski's proof we know that $S_{q,m,n} < 12m^4/n^2$. Hence, the third term of equation (2.3) can be bounded by

$$12\sum_{q\geq k+2}\sum_{m\geq 1}\left(m^4 \sum_{n\geq n_{m,q}} 1/n^2\right). \tag{2.4}$$

Let us now find a bound for $\sum_{n\geq n_{m,q}} 1/n^2$ of equation (2.4). For any $i \geq 1$, we know that $\sum_{n\geq i+1} 1/n^2 < 1/i$, and by the definition of $n_{m,q}$ we have, $n_{m,q} - 1 = \lfloor 24m^6q^2/\varepsilon\rfloor + 1 > 24m^6q^2/\varepsilon$. Then, $\sum_{n\geq n_{m,q}} 1/n^2 < \frac{\varepsilon}{24m^6q^2}$. Applying this last result to (2.4) we have

$$\begin{aligned}\sum_{q\geq k+2}\sum_{m\geq 1}\sum_{n\geq n_{m,q}} S_{q,m,n} &< \frac{\varepsilon}{2}\left(\sum_{q\geq k+2} 1/q^2\right)\left(\sum_{m\geq 1} 1/m^2\right)\\ &< \frac{\varepsilon}{k}.\end{aligned}$$

Similarly, the second term of equation (2.3) can be bounded by

$$\frac{\varepsilon}{2}\left(\sum_{q\geq 2} 1/q^2\right)\left(\sum_{m\geq k+1} 1/m^2\right) < \frac{\varepsilon}{2k}. \tag{2.5}$$

Finally, the first term of equation (2.3) can be bounded by

$$12\sum_{q\geq 2}\sum_{m\geq 1}\left(m^4 \sum_{n\geq kn_{m,q}+1} 1/n^2\right) < \frac{\varepsilon}{k}. \tag{2.6}$$

Replacing in (2.3) the bounds found in (2.4), (2.5) and (2.6) we conclude

$$r_k < \frac{\varepsilon}{k} + \frac{\varepsilon}{2k} + \frac{\varepsilon}{k} = \frac{5\varepsilon}{2k}.$$

and this finishes the proof. □

We now give an overview of our construction of the binary number

$$Z = 0.Z(0)\,Z(1)\,Z(2)\ldots \tag{2.7}$$

To determine the first digit of Z we divide the $[0,1]$ interval in two halves, $\mathcal{I}_0^0 = [0,1/2]$ and $\mathcal{I}_0^1 = [1/2,1]$, each one of measure $1/2$. Thinking in the scale of 2, in $\mathcal{I}_0^0$ there are only numbers whose first fractional digit is 0 while in $\mathcal{I}_0^1$ there are only numbers whose first fractional digit is 1. By Sierpinski's result we know that neither Δ nor any of the Δ_k cover the whole segment $[0,1]$. Of course, all reals external to Δ must either be in $\mathcal{I}_0^0$ or in $\mathcal{I}_0^1$. The idea now is to determine a subset of Δ, Δ_{p_0}, big enough (i.e. sufficiently similar to Δ) as to ensure that, whenever Δ_{p_0} does not completely cover a given interval, then Δ does not either. We can guarantee this because we have an upper bound on the error of approximating Δ at every step. We pick the interval $\mathcal{I}_0^0$ or $\mathcal{I}_0^1$, the least covered by Δ_{p_0}. If we select $\mathcal{I}_0^0$ then there will be real numbers external to every interval of Δ whose first digit in the binary expansion is 0; therefore, we define $Z(0) = 0$. Similarly, if we select $\mathcal{I}_0^1$ we define $Z(0) = 1$.

To define the rest of the digits we proceed recursively. We will divide $\mathcal{I}_{n-1}^{Z(n-1)}$ in two halves defining the intervals $\mathcal{I}_n^0$ and $\mathcal{I}_n^1$, each of measure $1/2^n$. At least one of the two will not be completely covered by Δ. The n-th digit of Z will be determined by comparing the measure of a suitable set Δ_{p_n} restricted to the intervals $\mathcal{I}_n^0$ and $\mathcal{I}_n^1$, where the index p_n is computably obtained from n. If we select $\mathcal{I}_n^0$, then we will define $Z(n)$ to be 0, otherwise $Z(n)$ will be 1. Since these measures are computable we have obtained an algorithm to define a real number Z, digit by digit, such that Z is external to every interval of Δ. By Sierpinski's result, Z is absolutely normal.

Before proceeding we shall prove some results, concerning the error we make when approximating Δ with Δ_k. The following proposition gives us a bound on the measure of the sets that have not been enumerated at step k.

Proposition 2.4.4. *For every k, $\mu(\Delta \setminus \Delta_k) \leq r_k$.*

Proof. Since Δ_k is included in Δ, the measure of $\Delta \setminus \Delta_k$ is less than or equal to the sum of the measures of those intervals in Δ but not in Δ_k. Hence $\mu(\Delta \setminus \Delta_k) \leq s - s_k = r_k$. □

We are also able to bound the measure of the difference between two sets enumerated at different steps.

Proposition 2.4.5. *For every k and l such that $k \leq l$, $\mu(\Delta_l \setminus \Delta_k) \leq r_k - r_l$.*

Proof. Since Δ_k is included in Δ_l, the measure of $\Delta_l \setminus \Delta_k$ is less than or equal to the sum of the measures of those intervals in Δ_l but not in Δ_k. Hence $\mu(\Delta_l \setminus \Delta_k) \leq s_l - s_k = r_k - r_l$. □

Let $\mathcal{C}$ be a set of intervals and let $\mathcal{I}$ be an interval. We will denote with $\mathcal{C} \cap \mathcal{I}$ the restriction of $\mathcal{C}$ to $\mathcal{I}$, i.e. $\mathcal{I} \cap \bigcup_{\mathcal{J} \in \mathcal{C}} \mathcal{J}$.

Lemma 2.4.6. *For any interval $\mathcal{I}$ and any k, $\mu(\Delta \cap \mathcal{I}) \leq \mu(\Delta_k \cap \mathcal{I}) + r_k$.*

Proof. Since $\Delta_k \subseteq \Delta$, for any k we have that $\Delta \cap \mathcal{I} \subseteq (\Delta_k \cap \mathcal{I}) \cup \left(\bigcup_{\mathcal{I} \in \Delta \setminus \Delta_k} \mathcal{I}\right)$. Taking measure we obtain $\mu(\Delta \cap \mathcal{I}) \leq \mu(\Delta_k \cap \mathcal{I}) + \mu(\Delta \setminus \Delta_k)$. By Proposition 2.4.4 we have $\mu(\Delta \cap \mathcal{I}) \leq \mu(\Delta_k \cap \mathcal{I}) + r_k$. □

Lemma 2.4.7. *For any interval $\mathcal{I}$ and any k and l such that $k \leq l$, $\mu(\Delta_l \cap \mathcal{I}) \leq \mu(\Delta_k \cap \mathcal{I}) + r_k - r_l$.*

Proof. The argument is similar to the one in the proof of Lemma 2.4.6, using Proposition 2.4.5 and the fact that $\Delta_l \cap \mathcal{I} \subseteq (\Delta_k \cap \mathcal{I}) \cup \left(\bigcup_{\mathcal{I} \in \Delta_l \setminus \Delta_k} \mathcal{I}\right)$. □

Lemma 2.4.8. *The function $f \colon \mathbb{N}^+ \to \mathbb{Q}$ such that $f(k) = \mu(\Delta_k)$ is computable. Also the function $g \colon \mathbb{N}^+ \times \mathbb{Q} \times \mathbb{Q} \to \mathbb{Q}$ such that $g(k, a, b) = \mu(\Delta_k \cap (a, b))$ is computable.*

Proof. Δ_k is a finite set of known intervals with rationals endpoints. Moreover, since condition (2.1) is clearly computable, and the limits in the unions of Definition 2.4.1 also are, the function which given k produces a list of rationals

$$a_1, b_1, a_2, b_2, \ldots, a_m, b_m$$

(m depending on k) such that $a_i < b_i < a_{i+1}$ and $\Delta_k = (a_1, b_1) \cup \cdots \cup (a_m, b_m)$ is computable. Hence $\mu(\Delta_k) \in \mathbb{Q}$ and $\mu(\Delta_k \cap (a, b)) \in \mathbb{Q}$. An algorithm for calculating these magnitudes from k can be easily given. □

Theorem 2.4.9. *There is a real Z which is computable and absolutely normal.*

Proof. We give an algorithm to determine every bit of Z (see (2.7)). We prove that for all m, if we define

$$\begin{aligned} p_m &= 5 \cdot 2^{2m}, && (2.8) \\ u_m &= \varepsilon + \sum_{j=0}^{m} 2^j r_{p_j}, && (2.9) \end{aligned}$$

$$\begin{aligned} \mathcal{I}_m^0 &= [0.Z(0)\,Z(1)\ldots Z(m-1)\ ,\ 0.Z(0)\,Z(1)\ldots Z(m-1)\,1]\,, && (2.10) \\ \mathcal{I}_m^1 &= [0.Z(0)\,Z(1)\ldots Z(m-1)1\ ,\ 0.Z(0)\,Z(1)\ldots Z(m-1)\,111\ldots]\,, \end{aligned}$$

$$Z(m) = \begin{cases} 0 & \text{if } \mu\left(\Delta_{p_m} \cap \mathcal{I}_m^0\right) \leq \mu\left(\Delta_{p_m} \cap \mathcal{I}_m^1\right); \\ 1 & \text{otherwise.} \end{cases} \qquad (2.11)$$

then we obtain

$$\mu\left(\Delta \cap \mathcal{I}_m^{Z(m)}\right) < 1/2^{m+1} = \mu\left(\mathcal{I}_m^{Z(m)}\right) \qquad (2.12)$$

and

$$\mu\left(\Delta_{p_m} \cap \mathcal{I}_m^{Z(m)}\right) + r_{p_m} < u_m/2^{m+1}. \qquad (2.13)$$

For the determination of the first digit, take $m = 0$ in definitions (2.8), (2.9), (2.10) and (2.11). Clearly, $\mathcal{I}_0^0 = [0, 1/2]$ and $\mathcal{I}_0^1 = [1/2, 1]$ and

$$\mu\left(\Delta_{p_0} \cap \mathcal{I}_0^0\right) + \mu\left(\Delta_{p_0} \cap \mathcal{I}_0^1\right) = \mu\left(\Delta_{p_0}\right) \leq s_{p_0}.$$

Adding $r_{p_0} + r_{p_0}$ at each side of this inequality and using the definition of r_{p_0}, we have

$$\begin{aligned} \left(\mu\left(\Delta_{p_0} \cap \mathcal{I}_0^0\right) + r_{p_0}\right) + \left(\mu\left(\Delta_{p_0} \cap \mathcal{I}_0^1\right) + r_{p_0}\right) &\leq s_{p_0} + r_{p_0} + r_{p_0} \\ &= s + r_{p_0} \\ &< \varepsilon + r_{p_0}. \end{aligned}$$

It is impossible that both terms $\mu\left(\Delta_{p_0} \cap \mathcal{I}_0^0\right) + r_{p_0}$ and $\mu\left(\Delta_{p_0} \cap \mathcal{I}_0^1\right) + r_{p_0}$ be greater than or equal to $(\varepsilon + r_{p_0})/2$. Thus, either $\mu\left(\Delta_{p_0} \cap \mathcal{I}_0^0\right)$ is less than $(\varepsilon + r_{p_0})/2 - r_{p_0}$ or $\mu\left(\Delta_{p_0} \cap \mathcal{I}_0^1\right)$ is. Following (2.11),

$$Z(0) = \begin{cases} 0 & \text{if } \mu\left(\Delta_{p_0} \cap \mathcal{I}_0^0\right) \leq \mu\left(\Delta_{p_0} \cap \mathcal{I}_0^1\right); \\ 1 & \text{otherwise.} \end{cases}$$

and then we proved (2.13) for $m = 0$:

$$\mu\left(\Delta_{p_0} \cap \mathcal{I}_0^{Z(0)}\right) + r_{p_0} < \frac{\varepsilon + r_{p_0}}{2}.$$

The idea is that the value of p_0 is large enough so that the error r_{p_0} is sufficiently small to guarantee that even if all the remaining intervals that have not yet been enumerated at step p_0 fall in $\mathcal{I}_0^{Z(0)}$, the whole $\mathcal{I}_0^{Z(0)}$ will not be completely covered by Δ. Theorem 2.4.3 states that $r_{p_0} < 5\varepsilon/2p_0$. Following (2.8), $p_0 = 5$ and so $r_{p_0} < \varepsilon/2$. Then,

$$\begin{aligned} \mu\left(\Delta_{p_0} \cap \mathcal{I}_0^{Z(0)}\right) + r_{p_0} &< \frac{\varepsilon + r_{p_0}}{2} \\ &< \varepsilon \leq 1/2 \end{aligned}$$

and using Lemma 2.4.6 we obtain

$$\mu\left(\Delta \cap \mathcal{I}_0^{Z(0)}\right) < 1/2 = \mu\left(\mathcal{I}_0^{Z(0)}\right),$$

which is exactly what we wanted, i.e. (2.12) specializing for $m = 0$.

This means that the union of all the intervals belonging to Δ will never cover the whole interval $\mathcal{I}_0^{Z(0)}$, whose measure is 1/2. Thus, there exist real numbers belonging to no interval of Δ that fall in the interval $\mathcal{I}_0^{Z(0)}$. These have their first digit equal to $Z(0)$.

For the determination of the n-th digit, $n > 1$, assume definitions (2.8), (2.9), (2.10) and (2.11) and also assume (2.12) and (2.13) are true for all $m \leq n-2$. We prove that if we follow the definitions (2.8), (2.9), (2.10) and (2.11) for $m = n-1$, we arrive to (2.12) and (2.13) for $m = n-1$.

By (2.8), $p_{n-1} = 5 \cdot 2^{2n-2}$. Following (2.10), we split the interval $\mathcal{I}_{n-2}^{Z(n-2)}$ in two halves of measure $1/2^n$ each:

$$\begin{aligned} \mathcal{I}_{n-1}^0 &= [0.Z(0)\,Z(1)\ldots Z(n-2)\ ,\ 0.Z(0)\,Z(1)\ldots Z(n-2)\,1]\,; \\ \mathcal{I}_{n-1}^1 &= [0.Z(0)\,Z(1)\ldots Z(n-2)1\ ,\ 0.Z(0)\,Z(1)\ldots Z(n-2)\,111\ldots] \end{aligned}$$

(written in the scale of 2). As they partition the interval $\mathcal{I}_{n-2}^{Z(n-2)}$, we have

$$\mu\left(\Delta_{p_{n-1}} \cap \mathcal{I}_{n-1}^0\right) + \mu\left(\Delta_{p_{n-1}} \cap \mathcal{I}_{n-1}^1\right) = \mu\left(\Delta_{p_{n-1}} \cap \mathcal{I}_{n-2}^{Z(n-2)}\right).$$

Since $p_{n-1} \geq p_{n-2}$ and using Lemma 2.4.7 we obtain

$$\mu\left(\Delta_{p_{n-1}} \cap \mathcal{I}_{n-1}^0\right) + \mu\left(\Delta_{p_{n-1}} \cap \mathcal{I}_{n-1}^1\right) \leq \mu\left(\Delta_{p_{n-2}} \cap \mathcal{I}_{n-2}^{Z(n-2)}\right) + r_{p_{n-2}} - r_{p_{n-1}}.$$

Adding $r_{p_{n-1}} + r_{p_{n-1}}$ to both sides of this inequality we have

$$\begin{aligned} \left(\mu\left(\Delta_{p_{n-1}} \cap \mathcal{I}_{n-1}^0\right) + r_{p_{n-1}}\right) + \left(\mu\left(\Delta_{p_{n-1}} \cap \mathcal{I}_{n-1}^1\right) + r_{p_{n-1}}\right) \leq \\ \mu\left(\Delta_{p_{n-2}} \cap \mathcal{I}_{n-2}^{Z(n-2)}\right) + r_{p_{n-2}} + r_{p_{n-1}} \end{aligned}$$

and by (2.9) and (2.13),

$$\left(\mu\left(\Delta_{p_{n-1}} \cap \mathcal{I}_{n-1}^0\right) + r_{p_{n-1}}\right) + \left(\mu\left(\Delta_{p_{n-1}} \cap \mathcal{I}_{n-1}^1\right) + r_{p_{n-1}}\right) < u_{n-1}/2^{n-1}.$$

Hence, one of the two terms, $\mu\left(\Delta_{p_{n-1}} \cap \mathcal{I}^0_{n-1}\right) + r_{p_{n-1}}$ or $\mu\left(\Delta_{p_{n-1}} \cap \mathcal{I}^1_{n-1}\right) + r_{p_{n-1}}$, must be less than $u_{n-1}/2^n$. Following (2.11), we define

$$Z(n-1) = \begin{cases} 0 & \text{if } \mu\left(\Delta_{p_{n-1}} \cap \mathcal{I}^0_{n-1}\right) \leq \mu\left(\Delta_{p_{n-1}} \cap \mathcal{I}^1_{n-1}\right); \\ 1 & \text{otherwise.} \end{cases}$$

By Theorem 2.4.3 and (2.8) we have

$$u_{n-1} = \varepsilon + \sum_{j=0}^{n-1} 2^j r_{p_j} < \varepsilon + \varepsilon \sum_{j=0}^{n-1} 2^{-j-1} < 2\varepsilon.$$

From the above inequality and from the definition of $Z(n-1)$ we obtain

$$\begin{aligned} \mu\left(\Delta_{p_{n-1}} \cap \mathcal{I}^{n-1}_{Z(n-1)}\right) + r_{p_{n-1}} &< u_{n-1}/2^n \\ &< 2\varepsilon/2^n \leq 1/2^n, \end{aligned}$$

which proves (2.13) for $m = n-1$. Furthermore, using Lemma 2.4.6 we deduce

$$\mu\left(\Delta \cap \mathcal{I}^{n-1}_{Z(n-1)}\right) < 1/2^n = \mu\left(\mathcal{I}^{n-1}_{Z(n-1)}\right).$$

which proves (2.12) for $m = n-1$. Hence, the set Δ does not cover the interval $\mathcal{I}^{n-1}_{Z(n-1)}$. There must be real numbers in the interval $\mathcal{I}^{n-1}_{Z(n-1)}$ that belong to no interval of Δ.

Lemma 2.4.10. *The number Z is computable and absolutely normal.*

Proof. In our construction we need only to compute the measure of the sets $\Delta_{p_m} \cap \mathcal{I}^m_{Z(m)}$. Then, by Lemma 2.4.8, Z is computable.

Let us prove that Z is external to every interval of Δ. Suppose not. Then, there must be an open interval $\mathcal{I} \in \Delta$ such that $Z \in \mathcal{I}$. Consider the intervals $\mathcal{I}^{Z(0)}_0, \mathcal{I}^{Z(1)}_1, \mathcal{I}^{Z(2)}_2, \ldots$ By our construction, Z belongs to every $\mathcal{I}^n_{Z(n)}$. Let us call $\mathcal{J}$ the first interval $\mathcal{I}^{Z(n)}_n$ of the sequence such that $\mathcal{I}^{Z(n)}_n \subseteq \mathcal{I}$. Such an interval exists because the measure of $\mathcal{I}^{Z(n)}_n$ goes to 0 as n increases. But then the interval $\mathcal{J}$ is covered by Δ. This contradicts that in our construction at each step n we choose an interval $\mathcal{I}^{Z(n)}_n$ not fully covered by Δ. Thus, Z belongs to no interval of Δ, so by Sierpinski's result, Z is absolutely normal. □

This completes the proof of the whole result. □

Finally, it follows from Theorem 2.4.3 that the bound s on $\mu(\Delta)$ is also computable.

Corollary 2.4.11. *The real s is computable.*

Proof. Let us define the sequence of rationals in the scale of 2, $a_n = s_{5\varepsilon 2^{n-1}}$. By Theorem 2.4.3 we have $|s - a_n| = s - s_{5\varepsilon 2^{n-1}} = r_{5\varepsilon 2^{n-1}} < 2^{-n}$. Then, it is possible to approximate s by a computable sequence of rationals a_n such that the first n digits of a_n coincide with the first n digits of s. Therefore, s is computable. □

2.5 About Sierpinski's and other examples

Sierpinski takes $\xi = \min(E(1))$: this is his example of an absolutely normal number of [69]. We can consider the family of numbers that are definable using Sierpinski's idea for different values of ε. Fix ε to be any rational real in $(0,1]$ and let $\tilde{Z} = \min(E(\varepsilon))$. Following the idea of the construction of Theorem 2.4.9, we can define

$$\tilde{Z} = \tilde{Z}(0)\,\tilde{Z}(1)\dots$$

in the following way:

$$\tilde{Z}(n) = \begin{cases} 0 & \text{if } \mu\left(\Delta \cap \mathcal{I}_n^0\right) < 1/2^{n+1}; \\ 1 & \text{otherwise.} \end{cases}$$

where, as in the proof of Theorem 2.4.9, the intervals $\mathcal{I}_n^0$ and $\mathcal{I}_n^1$ are defined recursively as are the two halves of $\mathcal{I}_{n-1}^{\tilde{Z}(n-1)}$ and they have measure $1/2^{n+1}$.

Since $\mu\left(\Delta \cap \mathcal{I}_n^0\right)$ can be computably approximated from below (for example, via the computable approximation Δ_k of Definition 2.4.1), the number $\tilde{Z}$ (an in particular Sierpinski's ξ) turns out to be left-c.e. Here is the sketch of that fact. Defining the computable sequence of rationals

$$\tilde{Z}_i = \tilde{Z}_i(0)\,\tilde{Z}_i(1)\dots\tilde{Z}_i(i)$$

where

$$\tilde{Z}_i(n) = \begin{cases} 0 & \text{if } \mu\left(\Delta_i \cap \mathcal{I}_{n,i}^0\right) < 1/2^{n+1}; \\ 1 & \text{otherwise.} \end{cases}$$

and where the intervals $\mathcal{I}_{n,i}^0$ and $\mathcal{I}_{n,i}^1$ are defined as before but relative to the approximation Δ_i instead of Δ. If ever $\mu\left(\Delta_i \cap \mathcal{I}_{n,i}^0\right)$ grows too much and becomes greater than or equal to $1/2^{n+1}$, then we have to change the bit at position n from 0 to 1 and maybe all the bits at positions greater than n. This just means that $\tilde{Z}_i \leq \tilde{Z}_{i+1}$ and since $\tilde{Z}_i \to \tilde{Z}$ when $i \to \infty$, we have that $\tilde{Z}$ is left-c.e..

The construction we presented in section 2.4 defines Z, a computable absolutely normal number in the scale of 2. We can adapt the construction to define numbers in any other scale: to compute a number in the scale of $q \geq 2$, at each step we must divide the interval selected in the previous step in q parts. In the n-th step we determine the n-th digit defining the intervals $\mathcal{I}_n^0, \mathcal{I}_n^1, \dots, \mathcal{I}_n^{q-1}$ (of measure $1/q^n$), where

$$\mathcal{I}_i^n = \left[\frac{i}{q^n} + \sum_{j=1}^{n-1} \frac{Z(j-1)}{q^j} \;,\; \frac{i+1}{q^n} + \sum_{j=1}^{n-1} \frac{Z(j-1)}{q^j} \right]$$

for $0 \leq i < q$. We will choose $p_m = 5\,(q-1)\,2^{2m}$ and following the same steps as in the construction of a number in the scale of 2, there must be an index i such that

$$\mu\left(\Delta_{p_{n-1}} \cap \mathcal{I}_{n-1}^i\right) < \frac{1}{q^n} \left(\varepsilon + (q-1) \sum_{j=0}^{n-1} 2^j r_{p_j} \right) - r_{p_{n-1}}.$$

As before, we define $Z(n-1)$ as the first index corresponding to the interval least covered by Δ_{p_m}: $Z(n-1)$ is the least $i \in \{0, \ldots, q-1\}$ such that $\mu\left(\Delta_{p_{n-1}} \cap \mathcal{I}^i_{n-1}\right)$ is least amongst all $\mu\left(\Delta_{p_{n-1}} \cap \mathcal{I}^j_{n-1}\right)$ for $j \in \{0, \ldots, q-1\}$.

In principle, for different scales the numbers will be distinct (they will not be Z expressed in different scales), while they will all be examples of computable absolutely normal numbers. The definition of absolute normality is asymptotic, that is, it states a property that has to be true in the limit. Thus, given an absolutely normal number, we can alter it by adding or removing a finite number of digits of its fractional expansion to obtain an absolutely normal number. For example, we could fix an arbitrary number of digits of the fractional expansion and complete the rest with the digits of Z.

2.6 Turing's unpublished manuscript

The aim of the next three sections is to reconstruct Alan Turing's manuscript entitled *A note on normal numbers* which remained unpublished until 1992, when it was included in the "Collected works of Alan Turing" edited by J.L. Britton [76, pp. 117–119, with notes of the editor in pp. 263–265]. Britton remarks the difficulties to make the transcript because the originals are incomplete and quite hard to read. The original manuscript is in Turing's archive in King's College, Cambridge, and a scanned version of it is available on the Web from `www.turingarchive.org`. Our motivation for this work was to explore and make explicit the techniques used by Turing in relation to normal numbers especially because there are still no known general methods to prove normality of given real numbers nor there are fast algorithms to construct absolutely normal numbers (see [3, 4, 17]).

In his manuscript Turing presents two results (here transcribed as Theorems 2.6.1 and 2.6.2) with incomplete proofs. The first states that there is a computable construction showing that almost all real numbers are absolutely normal.

Theorem 2.6.1 (Turing's first theorem)**.** *There is a computable function $c : \mathbb{N} \times \mathbb{N} \to \mathcal{P}((0,1))$, such that*

1. *$c(k,n)$ is a finite union of intervals with rational endpoints;*
2. *$c(k,n+1) \subseteq c(k,n)$;*
3. *$\mu(c(k,n)) > 1 - 1/k$.*

and for each k, $E(k) = \bigcap_n c(k,n)$ has measure $1 - 1/k$ and consists entirely of absolutely normal reals.

The function c is *computable* in the sense that given k and n we can compute

$$a_1 < b_1 < a_2 < b_2 < \cdots < a_m < b_m$$

(m depending on n and k) such that a_i, b_i are rationals in $(0,1)$ and

$$c(k,n) = (a_1,b_1) \cup \cdots \cup (a_m,b_m).$$

Turing's second theorem is an algorithm to produce absolutely normal numbers.

Theorem 2.6.2 (Turing's second theorem)**.** *There is an algorithm that given an infinite sequence $\theta \in \{0,1\}^\omega$, produces an absolutely normal real number $\alpha \in (0,1)$ in the scale of* 2*, in such a way that the first n digits of θ determine α to within 2^{-n}. In case θ is computable, so is α.*

In the next two sections we successfully reconstruct Turing's first and second theorems. Our proof of the first theorem is indeed a completion of Turing's proof, except that some actual bounds we obtain do not fully coincide with those in the manuscript. To reconstruct the second theorem we introduce more dramatic changes. Both, Turing's intended algorithm and our reconstruction of it, are highly exponential, as with the algorithm of Theorem 2.4.9.

Whenever possible, we keep the notation used by Turing. Throughout the next two sections we will consistently use the following convention:

Convention 2.6.3. $R \in \mathbb{N}$ will be used for denoting the length of prefixes after the fractional point; n will be a natural number, generally between 0 and R; $t \in \mathbb{N}$, $t \geq 2$ will denote a scale; γ will denote a word in the scale of t; r will be the length of γ; $\varepsilon \in \mathbb{R}$ will denote a (small) real used to bound certain deviations from expected values.

We shall fix a bijection between words of length r in the scale of t with digits in the scale of t^r.

2.7 Turing's first theorem

Turing gives a uniform method to, given $k \in \mathbb{N}$ large enough, construct a set $E(k)$ of points in $(0,1)$ that are absolutely normal such that $\mu(E(k)) = 1 - 1/k$. $E(k)$ is an infinite countable intersection of certain computably defined sets of intervals $c(k,n)$ that contain the reals that are candidates to be absolutely normal. Given k, n, the notion of candidate is a computable property on a real α. It says that in the initial segment of the fractional expansion of α of size R expressed in each scale up to T, every word with length up to L, occurs the expected number of times plus or minus Δ, where R, T, L, and Δ are computable functions of k, n (see section 2.8). The sets $c(k,n)$ are defined as a finite boolean combination of intervals with rational endpoints, and they are defined as to have Lebesgue measure equal to $1 - 1/k + 1/(k+n)$.

In Turing's manuscript the proof that the sets $c(k,n)$ have this desired measure depends on an unproved statement: Lemma 2.7.2. This lemma gives an upper bound for the number of words of a given length in which a given word occurs too often or too seldom.

We have not been able to prove nor disprove Turing's assertion. Instead, in Lemma 2.7.7 we provide an alternative bound, less sharp than Turing's but still allowing for the same construction.

Definition 2.7.1. Let t, γ and r as in Convention 2.6.3.

- $Q(w,\gamma)$ is the number of occurrences of γ in w;

- $P(t,\gamma,n,R) = \{w \in \{0,\dots t-1\}^R \colon Q(w,\gamma) = n\}$;
- $N(t,\gamma,n,R) = \|P(t,\gamma,n,R)\|$.

The symbolic expression of the function N is not a simple one because of the possible *overlapping* of different occurrences of γ when $|\gamma| > 1$; for instance, the word $\gamma = 00$ occurs once in 1100, twice in 1000 and three times in 0000. However, in any scale t, the symbolic expression for the function N considering the exact number of occurrences of a given *digit* is simple: the number of words of length R in the scale of t with exactly n occurrences of the *digit* d in *assigned* places is $(t-1)^{R-n}$. Hence, the number of words of length R in the scale of t with exactly n occurrences of the digit d in *some* place is

$$N(t,d,n,R) = \binom{R}{n}(t-1)^{R-n} \tag{2.14}$$

and hence

$$\sum_{0\le n\le R} N(n) = t^R. \tag{2.15}$$

Lemma 2.7.2 (Turing, unproved)**.** *Let t, γ and r be as in Convention 2.6.3, and let $\delta \in \mathbb{R}$ be such that $\delta\frac{t^r}{R} < 0.3$. Then,*

$$\sum_{|n-R/t^r|>\delta} N(t,\gamma,n,R) < 2t^R e^{-\frac{\delta^2 t^r}{4R}}.$$

We will substitute it with Lemma 2.7.7 below. We first need some auxiliary results.

Lemma 2.7.3 (adapted from Harman's [42, Lemma 1.1])**.** *Let d be a digit in the scale of t, $t \ge 2$. Assuming $R > 6t$ and with ε such that $6/R \le \varepsilon \le 1/t$, both*

$$\sum_{n\ge R/t+\varepsilon R} N(t,d,n,R) \quad \textit{and} \quad \sum_{n\le R/t-\varepsilon R} N(t,d,n,R)$$

are at most $t^R e^{-t\varepsilon^2 R/6}$.

Proof. Since t, d and R are fixed, we write $N(n)$ for $N(t,d,n,R)$. Recalling from (2.14) the symbolic expression for $N(n)$, it is easy to see that

$$\frac{N(n)}{N(n+1)} = \frac{(n+1)(t-1)}{R-n}. \tag{2.16}$$

For all $n \le R/t$ we have $N(n) > N(n-1)$ and for all $n > R/t$, $N(n) \le N(n-1)$. It is not difficult to see that the quotients in (2.16) increase as n increases.

Let $a = R/t - \varepsilon R$ and $b = R/t + \varepsilon R$. The strategy is to *shift* the first sum to the right by $m = \lfloor \varepsilon R/2 \rfloor$ positions, and the second sum to the left by $m+1$ positions.

Let us compute the stated upper bound for the first sum. For any n

$$N(n) = \frac{N(n)}{N(n+1)} \cdot \frac{N(n+1)}{N(n+2)} \cdot \ldots \cdot \frac{N(n+m-1)}{N(n+m)} \cdot N(n+m) \tag{2.17}$$

and for each i such that

$$i \leq \lfloor a \rfloor + m - 1 \tag{2.18}$$

we have

$$\begin{aligned} \frac{N(i)}{N(i+1)} &\leq \frac{N(\lfloor a \rfloor + m - 1)}{N(\lfloor a \rfloor + m)} \\ &= \frac{(\lfloor a \rfloor + m)(t-1)}{R - \lfloor a \rfloor - m + 1} \\ &< \frac{(R/t - \varepsilon R/2)(t-1)}{R - R/t + \varepsilon R/2} \\ &= 1 - \frac{\varepsilon t/2}{1 - 1/t + \varepsilon/2}. \end{aligned}$$

Since $\varepsilon \leq 1/t$ we conclude

$$\begin{aligned} \frac{N(i)}{N(i+1)} &< 1 - \varepsilon t/2 \\ &< e^{-t\varepsilon/2}. \end{aligned} \tag{2.19}$$

If $n \leq a$ then $n \leq \lfloor a \rfloor$ and hence $i = n + m - 1$ satisfies condition (2.18). Since the greatest quotient among the ones which appear in equation (2.17) is the last one, we can apply (2.19) to each factor in (2.17) to obtain

$$\begin{aligned} N(n) &< e^{-t\varepsilon m/2}\, N(n+m) \\ &\leq e^{-t\varepsilon(\varepsilon R/2-1)/2}\, N(n+m) \\ &= e^{-t\varepsilon^2 R/4 + t\varepsilon/2}\, N(n+m) \\ &\leq e^{-t\varepsilon^2 R/6}\, N(n+m) \end{aligned} \tag{2.20}$$

where we use the definition of m, and in the last inequality (2.20) we have $\varepsilon^2 tR/6 \leq \varepsilon^2 tR/4 - \varepsilon t/2$, since $\varepsilon R \geq 6$. Hence by (2.15) we have

$$\begin{aligned} \sum_{n \leq a} N(n) &< e^{-t\varepsilon^2 R/6} \sum_{n \leq a} N(n+m) \\ &\leq t^R e^{-t\varepsilon^2 R/6}. \end{aligned}$$

To bound the second sum, we use the same strategy, but now we shift the sum to the left by $m+1$ positions. For any n,

$$N(n) = \frac{N(n)}{N(n-1)} \cdot \frac{N(n-1)}{N(n-2)} \cdot \ldots \cdot \frac{N(n-m)}{N(n-m-1)} \cdot N(n-m-1) \tag{2.21}$$

(with these ratios increasing as $n - i$ decreases), and for each i such that

$$i \geq \lceil b \rceil - m \tag{2.22}$$

we have

$$\begin{aligned}\frac{N(i)}{N(i-1)} &\leq \frac{N(\lceil b\rceil - m)}{N(\lceil b\rceil - m - 1)}\\ &= \frac{R-\lceil b\rceil + m + 1}{(\lceil b\rceil - m)(t-1)}\\ &\leq \frac{R - R/t - \varepsilon R/2 + 1}{(R/t + \varepsilon R/2)(t-1)}\\ &< 1 - \varepsilon t/3.\end{aligned}$$

The last inequality is just equivalent to $\varepsilon t - 2/t - \varepsilon < 1 - \frac{6}{\varepsilon t R}$ and since $\varepsilon t \leq 1$ and $\varepsilon > 0$ it is sufficient to prove that $1 - 2/t < 1 - \frac{6}{\varepsilon t R}$, which clearly holds for $\varepsilon > 3/R$. Therefore,

$$\frac{N(i)}{N(i-1)} < e^{-t\varepsilon/3}. \tag{2.23}$$

If $n \geq b$, then $n \geq \lceil b\rceil$ and hence $i = n - m$ satisfies condition (2.22). Since the greatest quotient among those which appear in equation (2.21) is the last one, we can apply (2.23) to each factor in (2.21) to obtain, as in (2.20)

$$\begin{aligned}N(n) &< e^{-t\varepsilon(m+1)/3} N(n-m-1)\\ &\leq e^{-\frac{t\varepsilon^2 R}{6}} N(n-m-1)\end{aligned}$$

and from this and (2.15),

$$\sum_{n \geq b} N(n) < t^R e^{-t\varepsilon^2 R/6}.$$

This completes the proof. □

Definition 2.7.4. Let t, γ and r be as in Convention 2.6.3. For $j \in \{0, \ldots, r-1\}$ we define

- $Q_j(w,\gamma)$ as the number of occurrences of γ in w at positions of the form $r \cdot q + j$ (i.e. congruent to j modulo r);
- $P_j(t,\gamma,n,R) = \{w \in \{0 \ldots t-1\}^R \colon Q_j(w,\gamma) = n\}$.

Lemma 2.7.5. *Let t, γ and r as in Convention 2.6.3 and let $w \in P(t,\gamma,n,R)$. There is $j \in \{0,\ldots,r-1\}$ such that $w \in P_j(t,\gamma,m,R)$ for some $m \leq n/r$ and there is $j \in \{0,\ldots,r-1\}$ such that $w \in P_j(t,\gamma,m,R)$ for some $m \geq n/r$.*

Proof. Suppose $w \in P(t,\gamma,n,R)$, i.e., γ has n occurrences in w. For each $j \in \{0,\ldots,r-1\}$ let $n_j \geq 0$ be the number of occurrences of γ in w at positions congruent to j modulo r. Then, $w \in P_j(t,\gamma,n_j,R)$, and clearly $\sum_{0 \leq n_j \leq r-1} n_j = n$. This equality implies that $n_j \leq n/r$ for some j, and not all n_js can be strictly smaller than n/r. □

The following lemma makes a correspondence between sums of $N(t,\gamma,n,R)$ and sums of $N(t^r, d, n, \lfloor R/r \rfloor)$, where d is the digit in the scale of t^r corresponding to the word γ.

Lemma 2.7.6. *Let t, γ and r be as in Convention 2.6.3 and let d be the digit corresponding to the word γ in the scale of t^r. Then*

$$\sum_{n\leq a} N(t,\gamma,n,R) \leq t^{r-1} r \sum_{m\leq a/r} N(t^r,d,m,\lfloor R/r\rfloor)$$

and

$$\sum_{n\geq a} N(t,\gamma,n,R) \leq t^{r-1} r \sum_{m\geq a/r} N(t^r,d,m,\lfloor R/r\rfloor).$$

Proof. For any $j = 0,\dots,r-1$ we define a translation f_j which transforms words w, of length R written in the scale of t into words of length $\lfloor R/r\rfloor$ written in the scale of t^r.

Here is the translation f_j. Let $w \in \{0,\dots,t-1\}^R$ and let $k = \lfloor R/r\rfloor$. We consider the $k-1$ blocks

$$\begin{aligned} b_1 &= w(j)\dots w(j+r-1);\\ b_2 &= w(j+r)\dots w(j+2r-1);\\ &\vdots\\ b_{k-1} &= w(j+r(k-2))\dots w(j+(k-1)r-1); \end{aligned}$$

and a last k-th block

$$b_k = w(j+r(k-1))\dots w(l)\ u$$

where $l = \min(j+rk-1, R-1)$ and u is the least word (in lexicographic order) such that $|b_k| = l + |u| = r$ and b_k is different from γ. Each b_i has length r and so it can be seen as a single digit in the scale of t^r.

Observe that any word v of length R and written in the scale of t and such that

$$v(j)\dots v(j+kr-1) = b_1\dots b_k$$

has the same translation than w, i.e. $f_j(v) = f_j(w)$. The only digits that v may differ from w are in positions $0,\dots,j-1$ and $j+kr,\dots,R-1$, so in total there are $r-1$ such digits. Hence there are at most t^{r-1} such words v with $f_j(v) = f_j(w)$. This implies that

$$\|P_j(t,\gamma,m,R)\| \leq t^{r-1} N(t^r,p,m,k).$$

Suppose w has exactly n occurrences of γ. By the first part of Lemma 2.7.5 we know that for all n there is $j \in \{0,\dots,r-1\}$ and $m \leq \lfloor n/r\rfloor$ such that $w \in P_j(t,\gamma,m,R)$. Therefore,

$$\bigcup_{n\leq a} P(t,\gamma,n,R) \ \subseteq \ \bigcup_{0\leq j<r}\ \bigcup_{m\leq a/r} P_j(t,\gamma,m,R)$$

and hence

$$\begin{aligned}\sum_{n\le a} N(t,\gamma,n,R) &= \|\bigcup_{0\le n\le a} P(t,\gamma,n,R)\| \\ &\le \sum_{j<r}\sum_{m\le a/r} \|P_j(t,\gamma,m,R)\| \\ &\le t^{r-1} r \sum_{m\le a/r} N(t^r,d,m,\lfloor R/r\rfloor).\end{aligned}$$

This completes the proof of the first part. The second part is similar, applying the last assertion in Lemma 2.7.5. □

Lemma 2.7.7. *Let t, γ and r be as in Convention 2.6.3 and let ε such that $6/\lfloor R/r\rfloor \le \varepsilon \le 1/t^r$. Then*

$$\sum_{|n-R/t^r|\ge\varepsilon R} N(t,\gamma,n,R) < 2t^{R+2r-2} r\ e^{-\frac{t^r\varepsilon^2 R}{6r}}.$$

Proof. Lemma 2.7.3 ensures that, for any digit p in the scale of t, whenever $6/R \le \varepsilon \le 1/t$,

$$\sum_{n\ge R/t+\varepsilon R} N(t,p,n,R) < t^R e^{-t\varepsilon^2 R/6}. \tag{2.24}$$

The idea is to use (2.24) with $\tilde t = t^r$, $\tilde R = R/r$, and the digit d corresponding to γ in the scale of $\tilde t$. By the second part of Lemma 2.7.6 we know

$$\sum_{n\ge R/t^r+\varepsilon R} N(t,\gamma,n,R) \le t^{r-1} r \sum_{n\ge \tilde R/\tilde t+\varepsilon\tilde R} N(\tilde t,d,n,\lfloor\tilde R\rfloor).$$

Since $\lfloor\tilde R\rfloor = \tilde R - x/r$ for some $x\in\{0,\dots,r-1\}$ and since $\lfloor\tilde R\rfloor\le\tilde R$, applying (2.24) we obtain

$$\begin{aligned}\sum_{n\ge R/t^r+\varepsilon R} N(t,\gamma,n,R) &\le t^{r-1} r \sum_{n\ge\lfloor\tilde R\rfloor/\tilde t+\varepsilon\lfloor\tilde R\rfloor} N(\tilde t,d,n,\lfloor\tilde R\rfloor) \\ &\le t^{r-1} r\ \tilde t^{\lfloor\tilde R\rfloor}\ e^{-\tilde t\varepsilon^2\lfloor\tilde R\rfloor/6} \\ &= t^{r-1} r\ \tilde t^{\tilde R - x/r} e^{-\varepsilon^2\tilde t(\tilde R - x/r)/6} \\ &= t^{R+r-1} r\ e^{\frac{-\varepsilon^2 t^r R}{6r}}\ e^{\frac{\varepsilon^2 t^r x}{6r}} t^{-x} \\ &\le t^{R+r-1} r\ e^{-\frac{\varepsilon^2 t^r R}{6r}}. \end{aligned} \tag{2.25}$$

To check the last inequality observe that, since $\varepsilon\le 1/t^r$, the expression $e^{\varepsilon^2 t^r x/(6r)} t^{-x}$ is at most 1 (indeed, $\varepsilon/(6r)\le\ln t$ because ε is at most $1/2$ and $6r\ln t$ is at least 4).

The other sum is more tricky. Lemma 2.7.3 ensures that for any digit p in the scale of t, whenever $6/R\le\varepsilon\le 1/t$,

$$\sum_{n\le R/t-\varepsilon R} N(t,p,n,R) < t^R e^{-t\varepsilon^2 R/6}. \tag{2.26}$$

By the first part of Lemma 2.7.6 and the definitions of d, $\tilde{t}$ and $\tilde{R}$ used above, we know

$$\begin{aligned}\sum_{n\le R/t^r-\varepsilon R} N(t,\gamma,n,R) &\le t^{r-1}r \sum_{n\le \tilde{R}/\tilde{t}-\varepsilon\tilde{R}} N(\tilde{t},d,n,\lfloor\tilde{R}\rfloor)\\ &\le t^{r-1}r \sum_{n\le \tilde{R}/\tilde{t}-\varepsilon\tilde{R}} N(\tilde{t},d,n,\lceil\tilde{R}\rceil). \qquad (2.27)\end{aligned}$$

Let $R=\lfloor\tilde{R}\rfloor r+x$ where $x\in\{0,\dots,r-1\}$. If $x\neq 0$, since $\lceil\tilde{R}\rceil=\tilde{R}+(r-x)/r$ there is $y\in\{1,\dots,r-1\}$ such that $\lceil\tilde{R}\rceil=\tilde{R}+y/r$, and if $x=0$ then $y=0$ also satisfies the condition. Thus

$$\begin{aligned}\frac{\lceil\tilde{R}\rceil}{\tilde{t}}-\varepsilon\lceil\tilde{R}\rceil &= \frac{\tilde{R}}{\tilde{t}}+\frac{y}{\tilde{t}r}-\varepsilon\tilde{R}-\frac{\varepsilon y}{r}\\ &\ge \frac{\tilde{R}}{\tilde{t}}-\varepsilon\tilde{R}, \qquad (2.28)\end{aligned}$$

where the last inequality holds because $y/(\tilde{t}r)\ge\varepsilon y/r$ when $\varepsilon\le 1/t^r$. From (2.27), using (2.28) and (2.26), we get

$$\begin{aligned}\sum_{n\le R/t^r-\varepsilon R} N(t,\gamma,n,R) &\le t^{r-1}r \sum_{n\le\lceil\tilde{R}\rceil/\tilde{t}-\varepsilon\lceil\tilde{R}\rceil} N(\tilde{t},d,n,\lceil\tilde{R}\rceil)\\ &\le t^{r-1}\tilde{t}^{\lceil\tilde{R}\rceil}r\ e^{-\tilde{t}\varepsilon^2\lceil\tilde{R}\rceil/6}\\ &= t^{R+r-1}r\ e^{-t^r\varepsilon^2\lceil\tilde{R}\rceil/6}\ t^y\\ &\le t^{R+2r-2}r\ e^{-\frac{t^r\varepsilon^2R}{6r}}. \qquad (2.29)\end{aligned}$$

The last inequality follows from the fact that $t^y\le t^{r-1}$ and $\lceil\tilde{R}\rceil\ge\tilde{R}$. Joining (2.25) and (2.29), we obtain the desired upper bound. □

Recall the meaning of Q from Definition 2.2.1.

Definition 2.7.8. We denote by $B(\varepsilon,\gamma,t,R)$ the set of reals $\alpha\in(0,1)$ such that

$$|Q(\alpha,t,\gamma,R)-R/t^r|<\varepsilon R.$$

Proposition 2.7.9. *Let t, γ and r be as in Convention 2.6.3 and let ε and R be such that $6/\lfloor R/r\rfloor\le\varepsilon\le 1/t^r$. Then*

$$\mu\left(B(\varepsilon,\gamma,t,R)\right)>1-2t^{2r-2}r\ e^{-\frac{t^r\varepsilon^2R}{6r}}.$$

Proof. Let $\overline{B}(\varepsilon,\gamma,t,R)=(0,1)\setminus B(\varepsilon,\gamma,t,R)$. Observe that if a real $\alpha\in(0,1)$ belongs to $\overline{B}(\varepsilon,\gamma,t,R)$ then every real $\beta\in(0,1)$ such that the first R digits of α (written in the scale of t) coincide with the first R digits of β (written in the scale of t) also belongs to $\overline{B}(\varepsilon,\gamma,t,R)$, which means that the interval

$$[0.\alpha\restriction R\ 000\dots,\ 0.\alpha\restriction R\ (t-1)(t-1)(t-1)\dots]$$

of measure t^{-R} is included in $\overline{B}(\varepsilon,\gamma,t,R)$. Here $\alpha \restriction R$ denotes the first R digits of the fractional expansion of α in the scale of t. Then

$$\overline{B}(\varepsilon,p,t,R) = \bigcup_{|n-R/t|\geq\varepsilon R} \ \bigcup_{w\in P(t,\gamma,n,R)} [0.w000\ldots, 0.w(t-1)(t-1)(t-1)\ldots]\,.$$

Since the intervals in the right hand side are disjoint for different words w, we have:

$$\begin{aligned} \mu\left(\overline{B}(\varepsilon,\gamma,t,R)\right) &= t^{-R} \sum_{|n-R/t|\geq\varepsilon R} N(t,\gamma,n,R) \\ &< 2t^{2r-2} r\ e^{-\frac{t^r\varepsilon^2 R}{6r}}\,. \end{aligned} \tag{2.30}$$

For the last equation apply Lemma 2.7.7. The proof is completed by taking complements. □

Definition 2.7.10. For any ε, T, L and R, let

$$A(\varepsilon,T,L,R) = \bigcap_{2\leq t\leq T} \ \bigcap_{1\leq r\leq L} \ \bigcap_{\gamma\in\{0,\ldots,t-1\}^r} B(\varepsilon,\gamma,t,R).$$

Since each $B(\varepsilon,\gamma,t,R)$ is a finite union of intervals with rational endpoints, then so is each of the sets $A(\varepsilon,T,L,R)$.

Proposition 2.7.11. *For any ε, T, L and R, such that $6/\lfloor R/L\rfloor \leq \varepsilon \leq 1/T^L$,*

$$\mu\left(A(\varepsilon,T,L,R)\right) > 1 - 2LT^{3L-1}\ e^{-\frac{\varepsilon^2 R}{3L}}\,.$$

Proof. Let $\overline{A}$ and $\overline{B}$ denote the complements of the sets A, B, respectively, in the interval $(0,1)$.

$$\mu\left(\overline{A}(\varepsilon,T,L,R)\right) \leq \sum_{2\leq t\leq T} \ \sum_{1\leq r\leq L} \ \sum_{\gamma\in\{0,\ldots,t-1\}^r} \mu\left(\overline{B}(\varepsilon,\gamma,t,R)\right).$$

Observe that in the third summand there are t^r many γs and that

$$\sum_{2\leq t\leq T} \sum_{1\leq r\leq L} t^r \ = \ \sum_{2\leq t\leq T} \frac{t^{L+1}-1}{t-1} \ \leq \ T^{L+1}.$$

The upper bound for $\mu\left(\overline{B}(\varepsilon,\gamma,t,R)\right)$ in (2.30) yields the following uniform upper bound in terms of the present parameters ε, T, R, L:

$$\mu\left(\overline{B}(\varepsilon,\gamma,t,R)\right) \ < \ 2T^{2L-2}L\ e^{-\frac{2\varepsilon^2 R}{3L}}$$

valid for all $2 \leq t \leq T$, $1 \leq r \leq L$ and $\gamma \in \{0,\ldots,t-1\}^r$. Indeed, from $1 \leq r \leq L$, we get $2r/L \leq 2 \leq t^r$; hence, $\varepsilon^2 R/(3L) \leq \varepsilon^2 R t^r/(6r)$, which gives

$$\mu\left(\overline{B}(\varepsilon,\gamma,t,R)\right) \ < \ 2t^{2r-2}r\ e^{-\frac{t^r\varepsilon^2 R}{6r}} \ < \ 2T^{2L-2}\ e^{-\frac{\varepsilon^2 R}{3L}}\,.$$

Hence we obtain,

$$\mu\left(\overline{A}(\varepsilon,T,L,R)\right) < 2LT^{3L-1}\ e^{-\frac{\varepsilon^2 R}{3L}}\,.$$

The proof is completed by taking complements. □

We now define $A(\varepsilon, T, L, R)$ for specific values of its parameters.

Definition 2.7.12. Let $A_k = A(\varepsilon, T, L, R)$ for $R = k$, $L = \sqrt{\ln k}/4$, $T = e^L$ and $\varepsilon = 1/T^L$.

Proposition 2.7.13. *There is k_0 such that for all $k \geq k_0$, $\mu(A_k) \geq 1 - \frac{1}{k(k-1)}$.*

Proof. Let R, T, L and ε be the functions of k given in Definition 2.7.12. Observe that $T^L = \sqrt[16]{k}$. Since $\varepsilon \geq 6/\lfloor R/L \rfloor$ for all $k \geq 2$, the hypothesis of Proposition 2.7.11 is satisfied. We now prove that

$$2LT^{3L-1}\, e^{-\frac{\varepsilon^2 R}{3L}} \leq \frac{1}{k(k-1)}$$

for large enough k. It suffices to prove $T^{3L}k^2 \leq e^{\frac{\varepsilon^2 R}{3L}}$ because $2L \leq T$. This is equivalent to

$$1/\varepsilon^2 \cdot (9L^2 \ln T + 6L \ln k) \leq k. \tag{2.31}$$

Since $1/\varepsilon^2 = T^{2L} = \sqrt[8]{k}$, $9L^2 \ln T = (9/64)(\ln k)^{3/2}$ and $6L \ln k = (3/2)(\ln k)^{3/2}$, (2.30) reduces to

$$(105/64)\sqrt[8]{k}(\ln k)^{3/2} \leq k \tag{2.32}$$

which can be proved to hold for any $k \geq 1$. □

Remark 2.7.14. Observe that the assignment of Definition 2.7.12 gives initial values of T smaller than 2 and the initial values of L smaller than 1. This implies that the initial intersections in A_k will have an empty range. However, as k increases, these variables will take greater and greater values.

One can give different assignments for $L = L(k)$, $T = T(k)$ and $\varepsilon = \varepsilon(k)$, where $\lim_k L(k) = \infty$, $\lim_k T(k) = \infty$ and $\lim_k \varepsilon(k) = 0$ and such that $L \geq 1, T \geq 2$ and Proposition 2.7.13 is verified for suitable large k.

From now on let k_0 be the value determined in Proposition 2.7.13 (or Remark 2.7.14).

Theorem 2.7.15. *The set $\bigcap_{k \geq k_0} A_k$ contains only absolutely normal numbers.*

Proof. Assume $\alpha \in \bigcap_{k \geq k_0} A_k$ and α is not normal to the scale of t. This means that

$$\lim_{R \to \infty} \frac{Q(\alpha, t, \gamma, R)}{R} \neq \frac{1}{t^r}$$

for some word γ of length r in the scale of t. Hence there is $\delta > 0$ and there are infinitely many Rs such that

$$|Q(\alpha, t, \gamma, R) - R/t^r| > R\delta. \tag{2.33}$$

Let $T(k)$, $L(k)$ and $\varepsilon(k)$ be the assignments of Definition 2.7.12 or Remark 2.7.14. Now fix $k_1 \geq k_0$ large enough such that $T(k_1) \geq t$, $L(k_1) \geq r$ and $\varepsilon(k_1) \leq \delta$. This is always possible because $T(k) \to \infty$, $L(k) \to \infty$ $\varepsilon(k) \to 0$ when $k \to \infty$.

For any $k \geq k_1$, $\alpha \in A_k$, and by Definition 2.7.10, $\alpha \in B(\varepsilon(k), \gamma, t, k)$. By Definition 2.7.8 we have

$$\begin{aligned} |Q(\alpha, t, \gamma, k) - k/t^r| &< k\varepsilon(k) \\ &\leq k\delta. \end{aligned}$$

for any $k \geq k_1$. Now, any $R \geq k_1$ satisfying (2.33) leads to a contradiction. □

Turing defines $c(k, n)$ as intersections of finitely many A_k and he restricts these sets so that they have measure exactly $1 - 1/k + 1/(k+n)$.

Definition 2.7.16. The computable function $c : \mathbb{N} \times \mathbb{N} \to \mathcal{P}((0,1))$, is defined as follows. For any $k \geq k_0$ let $c(k, 0) = (0, 1)$ and

$$c(k, n+1) = A_{k+n+1} \cap c(k, n) \cap (\beta_n, 1)$$

where $(\beta_n, 1)$ is an interval so that $\mu(c(k, n+1)) = 1 - 1/k + 1/(k+n+1)$.

Remark 2.7.17. It is worth noting that some interval $(\beta_n, 1)$ as above always exists and it is unique. This is because

$$\mu(A_{k+n+1} \cap c(k, n)) \geq 1 - 1/k + 1/(k+n+1).$$

Since $c(k, n)$ and A_{k+n+1} are finite unions of intervals with rational endpoints, their respective measures are effectively computable; β_n is rational and it can be determined effectively. Hence $c(k, n)$ may be represented by a finite union of disjoint intervals

$$(a_1, b_1) \cup \cdots \cup (a_m, b_m)$$

such that $a_i, b_i \in \mathbb{Q} \cap (0, 1)$, $a_i < b_i < a_{i+1}$ and such that $(a_1, b_1, a_2, b_2, \ldots, a_m, b_m)$ is computable from k and n.

We finally arrive to the proof of Turing's first theorem.

Proof of Theorem 2.6.1. Clearly, conditions 1, 2 and 3 follow from the definition of $c(k, n)$ (Definition 2.7.16). Since $E(k) \subseteq \bigcap_{i \geq k} A_i$, by Theorem 2.7.15 we conclude that if $k \geq k_0$, any real number in $E(k)$ is absolutely normal. By condition 2 and the fact that $\mu(c(k, n)) = 1 - 1/k + 1/(k+n)$, we get

$$\mu(E(k)) = \lim_{n \to \infty} \mu(c(k, n)) = 1 - 1/k.$$

This completes the proof. □

2.8 Turing's second theorem

We present our reconstruction of Turing's second theorem (Theorem 2.6.2). We do not maintain the same intervals and bounds that appear in the original manuscript, which seem to be wrong. We keep the strategy but we introduce new bounds to prove the correctness of the algorithm.

The idea of the algorithm is to recursively select for each integer $n > 0$ an interval $\mathcal{I}_n$ with dyadic rational endpoints such that

- $\mathcal{I}_{n+1} \subset \mathcal{I}_n$
- $\mu\left(\mathcal{I}_n\right) = 2^{-(n+1)}$
- $\mu\left(\mathcal{I}_n \cap E(k)\right) > 0.$

The intersection of these intervals, $\bigcap_n \mathcal{I}_n$, contains exactly one number which must be absolutely normal. The correctness of the algorithm relies on the fact that at each stage n the measure $\mu\left(c(k,n) \cap \mathcal{I}_n\right)$ is big enough to allow to proceed with the next stage.

We now adjust the sets $c(k,n)$ (see Definition 2.7.16) used in Theorem 2.6.1 to have measure $1 - 1/k + 1/k^{n+1}$ in order to make them suitable for the algorithm.

Let k_0 be as determined in Proposition 2.7.13 (or Remark 2.7.14).

Definition 2.8.1. We define the computable function $c : \mathbb{N} \times \mathbb{N} \to \mathcal{P}\left((0,1)\right)$, as follows. For any $k \geq k_0$ let $c(k,0) = (0,1)$ and

$$c(k,n+1) = A_{k^{n+2}} \cap c(k,n) \cap (\beta_n, 1);$$

where $(\beta_n, 1)$ is an interval so that $\mu\left(c(k,n+1)\right) = 1 - 1/k + 1/k^{n+2}$.

The reader may verify that it is always possible to find such a β_n for $k \geq k_0$, because

$$\mu\left(A_{k^{n+2}} \cap c(k,n)\right) \geq 1 - \frac{1}{k} + \frac{1}{k^{n+2}}.$$

Hence, for all $n \geq 0$, $\mu\left(c(k,n)\right) = 1 - 1/k + 1/k^{n+1}$.

Now we proceed with our reconstruction of the proof of Turing's second theorem.

Proof of Theorem 2.6.2. The following algorithm constructs a real α in $(0,1)$ in the scale of 2. It depends on an infinite sequence $\theta \in 2^\omega$ (used as an oracle to possibly determine some digits of α), and uses a fixed parameter k large enough ($k \geq k_0$ and $k \geq 4$). Here is the algorithm:

Start with $\mathcal{I}_{-1} = (0,1)$.At stage $n \geq 0$, split the interval $\mathcal{I}_{n-1}$ into two halves $\mathcal{I}_n^0$ and $\mathcal{I}_n^0$. That is, if $\mathcal{I}_{n-1} = (a_{n-1}, b_{n-1})$, then define

$$\mathcal{I}_n^0 = \left(a_{n-1}, \frac{a_{n-1} + b_{n-1}}{2}\right) \quad \text{and} \quad \mathcal{I}_n^1 = \left(\frac{a_{n-1} + b_{n-1}}{2}, b_{n-1}\right)$$

and let

$$M(k,n) = 2^{-n}\left(1/2 - 1/k - \frac{1 - (2/k)^{n-1}}{k^2 - 2k} + \frac{2^{n-1}}{k^{n+1}}\right).$$

If both $\mu\left(c(k,n) \cap \mathcal{I}_n^0\right)$ and $\mu\left(c(k,n) \cap \mathcal{I}_n^1\right)$ are greater or equal to $M(k,n)$ then define $\alpha(n) = \theta(n)$ and let

$$\mathcal{I}_n = \begin{cases} \mathcal{I}_n^0 & \text{if } \theta(n) = 0; \\ \mathcal{I}_n^1 & \text{otherwise.} \end{cases}$$

else, if $\mu\left(c(k,n)\cap\mathcal{I}_n^0\right)\geq M(k,n)$ then define $\mathcal{I}_n=\mathcal{I}_n^0$ and $\alpha(n)=0$; else, define $\mathcal{I}_n=\mathcal{I}_n^1$ and $\alpha(n)=1$.

The definition of $M(k,n)$ gives an explicit expression for a lower bound of $\mu\left(c(k,n)\cap\mathcal{I}_n\right)$, and verifies the inductive condition $M(k,n)-1/k^{n+1}+1/k^{n+2}=2M(k,n+1)$.

At each step n, $\mathcal{I}_n$ is either the left half of $\mathcal{I}_{n-1}$ (denoted $\mathcal{I}_n^0$) or the right half of it (denoted $\mathcal{I}_n^1$). As we mentioned in Remark 2.7.17, $c(k,n)$ is computable. Therefore we can compute its measure, and also compute the measures of both $c(k,n)\cap\mathcal{I}_n^0$ and $c(k,n)\cap\mathcal{I}_n^1$. All these measures are rational numbers in $(0,1)$.

The above algorithm defines $\alpha=\bigcap_n\mathcal{I}_n$ bit by bit, i.e. at stage n the n-th bit of α is defined. To prove that α is absolutely normal, we show $\alpha\in E(k)=\bigcap_n c(k,n)$. We prove, by induction on n, that for every $n\geq 0$,

$$\mu\left(c(k,n)\cap\mathcal{I}_n\right)\geq M(k,n). \tag{2.34}$$

For $n=0$, observe that by Definition 2.8.1, $c(k,0)=(0,1)$ and then

$$\mu\left(c(k,0)\cap\mathcal{I}_0\right)=1/2=M(k,0).$$

For the induction, assume (2.34) holds. Since $c(k,n+1)\subseteq c(k,n)$ we have

$$c(k,n+1)\cap\mathcal{I}_n=c(k,n)\cap\mathcal{I}_n\setminus\left(c(k,n)\setminus c(k,n+1)\right)\cap\mathcal{I}_n$$

and

$$\begin{aligned}\mu\left(c(k,n+1)\cap\mathcal{I}_n\right) &= \mu\left(c(k,n)\cap\mathcal{I}_n\right)-\mu\left(\left(c(k,n)\setminus c(k,n+1)\right)\cap\mathcal{I}_n\right)\\ &\geq \mu\left(c(k,n)\cap\mathcal{I}_n\right)-\mu\left(c(k,n)\setminus c(k,n+1)\right). \end{aligned}\tag{2.35}$$

Using (2.34) and $\mu\left(c(k,n)\setminus c(k,n+1)\right)=k^{-(n+1)}-k^{-(n+2)}$ in (2.35) we have

$$\begin{aligned}\mu\left(c(k,n+1)\cap\mathcal{I}_n\right) &\geq M(k,n)-(k^{-(n+1)}-k^{-(n+2)})\\ &= 2M(k,n+1).\end{aligned}$$

It is impossible that both $\mu\left(c(k,n+1)\cap\mathcal{I}_{n+1}^0\right)$ and $\mu\left(c(k,n+1)\cap\mathcal{I}_{n+1}^1\right)$ be less than $M(k,n+1)$. It follows that at least one of the sets $c(k,n+1)\cap\mathcal{I}_{n+1}^i$, $i\in\{0,1\}$, has measure $\geq M(k,n+1)$. The algorithm picks as $\mathcal{I}_{n+1}$ the set $\mathcal{I}_{n+1}^i$ which fulfills this condition, with the oracle used to decide in case both sets verify it.

It is not difficult to see that for all $k\geq 4$, $M(k,n)>0$, so at every step n, $c(k,n)\cap\mathcal{I}_n$ is non-empty and hence there are absolutely normal numbers in it. Furthermore by construction, all reals in $c(k,n)\cap\mathcal{I}_n$ have a fractional expansion starting with $\alpha(0)\alpha(1)\ \ldots\alpha(n)$. This completes the proof. □

In his manuscript Turing keeps for the second theorem the same sets $c(k,n)$ used in the first theorem, where $\mu\left(c(k,n)\right)=1-1/k+1/(k+n+1)$. We did not succeed in proving the correctness of the algorithm using these sets. However, our reconstruction is quite close to Turing's original idea, since for each k, $\lim_{n\to\infty}1-1/k+1/(k+n+1)=\lim_{n\to\infty}1-1/k+1/k^{n+1}=1-1/k$.

For both, Turing's intended algorithm and our reconstruction of it, we lack information about how many times the oracle θ is used to compute α. It is not proved that in case θ is not computable, so is α. However, in section 2.9 we make a more complex algorithm to guarantee this fact. In the original manuscript Turing uses the oracle under a subtler condition: in case both, $\mu(c(k,n)) \cap \mathcal{I}_n^0$ and $\mu(c(k,n)) \cap \mathcal{I}_n^1$ exceed the "minimum" measure required to proceed. But there is no justification that the given amount is in fact the minimum required, nor how many times it is reached.

Turing considers the union of all possible intervals $\mathcal{I}_n$ as we allow the first n digits of θ to run through all possibilities, and concludes that the set of numbers α that can be output by his algorithm is of measure at least $1 - 2/k$; by taking particular sequences θ (e.g. $\theta(n) = 0$ for all n) one obtains particular normal numbers.

2.9 Applications

It is known that all random reals are absolutely normal [18, 26]. As an application Theorem 2.6.1, we give a simple proof that all Schnorr random reals are absolutely normal.

Definition 2.9.1 (Schnorr test and Schnorr randomness [66])**.** A *Schnorr test* is a Martin-Löf test $(\mathcal{U}_i)_{i\in\mathbb{N}}$ such that $\mu(\mathcal{U}_i) = 2^{-i}$. The real A is *Schnorr random* if $A \notin \bigcap_i \mathcal{U}_i$ for any Schnorr test $(\mathcal{U}_i)_{i\in\mathbb{N}}$.

It is known that random reals (as in Definition 1.5.1 or 1.5.2) are Schnorr random, but the other inclusion is not true. The set $E(k)$ as defined in Theorem 2.6.1 already implies that Schnorr randomness is a stronger notion than absolute normality.

Theorem 2.9.2. *Any Schnorr random real is absolutely normal.*

Let us define

$$\begin{aligned}\mathcal{U}_i &= (0,1) \setminus E(2^i)\\ &= \bigcup_{n\in\mathbb{N}} (0,1) \setminus c(2^i, n).\end{aligned}$$

Proof. Clearly $\mu(\mathcal{U}_i) = 2^{-i}$. Since the sets $c(k,n)$ are computable from k and n (in the sense explained in Remark 2.7.17), the sequence $(\mathcal{U}_i)_{i\in\mathbb{N}}$ is a uniformly c.e. sequence of Σ_1^0-classes, hence a Schnorr test.

Since for any k, $E(k)$ consists entirely of absolutely normal numbers, if A is not absolutely normal then A is belongs to $\mathcal{U}_i$ for all i, and then A is not Schnorr random. □

Based on the algorithm obtained from either Sierpinski's or Turing's work, we can prove that there is an absolutely normal number in each 1-degree.

We identify reals in $(0,1)$ with infinite sequences of 0s and 1s. For $\mathcal{C} \subseteq 2^\omega$ and $\sigma \in 2^{<\omega}$ we define $\mathcal{C}|\sigma = \{X \in 2^\omega : \sigma X \in \mathcal{C}\}$ and $\mathcal{C} \cap \sigma = \{X \in \mathcal{C} : \sigma \prec X\}$. Notice that $\mu(\mathcal{C}|\sigma) = 2^{|\sigma|}\mu(\mathcal{C} \cap \sigma)$.

Lemma 2.9.3 (Lebesgue density theorem). *Let $\mathcal{C}$ be a measurable subset of $\mathbb{R}$ with $\mu(\mathcal{C}) > 0$, and let $\delta < 1$. Then there exists $\sigma \in 2^{<\omega}$ such that $\mu(\mathcal{C}|\sigma) \geq \delta$.*

For a proof of the above Lemma, the reader may refer to [64].

Theorem 2.9.4. *There is an absolutely normal number r and an infinite computable set $A = \{a_0, a_1, \dots\}$ $(a_i < a_{i+1})$ such that every real q which coincides with r in positions of $\mathbb{N} \setminus A$ is also absolutely normal.*

Proof. We follow some ideas about applying Lemma 2.9.3 from [37]. For every s we define $\mathcal{C}_s$ and σ_s such that

1. $\mathcal{C}_s = \bigcap_k \mathcal{C}_{s,k}$;
2. $\mathcal{C}_{s,k+1} \subseteq \mathcal{C}_{s,k} \subseteq \mathbb{R}$;
3. $\mu(\mathcal{C}_s) > 0$;
4. For all $x_1, \dots, x_s \in \{0,1\}$, $\sigma_1 x_1 \dots \sigma_s x_s \mathcal{C}_s$ contains only absolutely normal numbers;
5. There are computable functions f_s and g_s such that
$$\mathcal{C}_{s,k} = (f_s(k,1), f_s(k,2)) \cup \dots \cup (f_s(k, g_s(k)-1), f_s(k, g_s(k))).$$
6. There is a computable function $b_s \colon \mathbb{N} \to \mathbb{Q}$ such that $\mu(\mathcal{C}_{s,k} \setminus \mathcal{C}_s) \leq b_s(k)$ and $\lim_k b_s(k) = 0$.

The existence of these sets is guaranteed by the constructions introduced in sections 2.4 or 2.7.

Here is the construction:

Stage $s+1$. We have already constructed $\sigma_1, \dots, \sigma_s$ and $\mathcal{C}_s$ satisfying the above conditions. We do the following search: at stage stage n, let σ be the n-th string in the length-lexicographic order. Let m be the least number such that $b_s(m) \leq 2^{-|\sigma|-2}$. If $\mu(\mathcal{C}_{s,m}|\sigma) \leq 3/4$ then go to stage $n+1$; else terminate. This search must eventually terminate because by Lebesgue density theorem there must be a σ with $\mu(\mathcal{C}_s|\sigma) > 3/4$ and hence $\mu(\mathcal{C}_{s,m}|\sigma) > 3/4$ for all m. Suppose we find string σ such that $b_s(m) \leq 2^{-|\sigma|-2}$ and $\mu(\mathcal{C}_{s,m}|\sigma) > 3/4$. Then

$$\begin{aligned} \mu(\mathcal{C}_s \cap \sigma) &\geq \mu(\mathcal{C}_{s,m} \cap \sigma) - b_s(m) \\ &> 3 \cdot 2^{-|\sigma|-2} - 2^{-|\sigma|-2} \\ &= 2^{-|\sigma|-1} \end{aligned}$$

and hence

$$\begin{aligned} \mu(\mathcal{C}_s|\sigma 0 \cap \mathcal{C}_s|\sigma 1) &\geq \mu(\mathcal{C}_s|\sigma 0) + \mu(\mathcal{C}_s|\sigma 1) - 1 \\ &= 2\mu(\mathcal{C}_s|\sigma) - 1 \\ &> 0. \end{aligned} \tag{2.36}$$

Let us see that $\mathcal{C}_{s+1} = \mathcal{C}_s|\sigma 0 \cap \mathcal{C}_s|\sigma 1$ and $\sigma_{s+1} = \sigma$ verify the desired properties. Clearly 1 and 2 hold because $\mathcal{C}_{s+1} = \bigcap_k \mathcal{C}_{s+1,k}$ where $\mathcal{C}_{s+1,k} = \mathcal{C}_{s,k}|\sigma 0 \cap \mathcal{C}_{s,k}|\sigma 1$. Equation (2.36) shows that condition 3 is true. Since $\sigma 0 \mathcal{C}_{s+1} \subseteq \mathcal{C}_s$ and $\sigma 1 \mathcal{C}_{s+1} \subseteq \mathcal{C}_s$, 4 is also verified. It is not difficult to see that 5 is true for some functions f_{s+1} and g_{s+1} constructed uniformly form f_s and g_s. We can also computably bound $\mu\left(\mathcal{C}_{s+1,k} \setminus \mathcal{C}_{s+1}\right)$ by $b_{s+1} = 2^{|\sigma|} b_s$.

Now we define $a_n = n + \sum_{1 \leq s \leq n+1} |\sigma_s|$ and $A = \{a_0, a_1, \dots\}$. We also define $r = 0.\sigma_1 0 \sigma_2 0 \sigma_3 0 \dots$ Then for every set $B \subseteq \mathbb{N}$ we have that $0.\sigma_1 B(0) \sigma_2 B(1) \sigma_3 B(2) \dots$ is absolutely normal. □

Corollary 2.9.5. *There is an absolutely normal number in each* 1*-degree.*

3. RANDOMNESS AND HALTING PROBABILITIES

We consider the question of randomness of the probability $\Omega_{\mathbf{U}}[X]$ that a universal Turing machine $\mathbf{U}$ halts and outputs a string in a fixed set X, generalizing Chaitin's $\Omega_{\mathbf{U}}$. We study the question "is $\Omega_{\mathbf{U}}[X]$ random?" for different sets X and for different universal machines $\mathbf{U}$. If the universal machine $\mathbf{U}$ is given, we prove that $\Omega_{\mathbf{U}}[X]$ is random whenever X is Σ^0_n-complete or Π^0_n-complete for $n \geq 2$. However, for $n \geq 2$, $\Omega_{\mathbf{U}}[X]$ is not n-random when X is Σ^0_n or Π^0_n. If we can fix the universal machine, then there are more negative answers. It is shown that there are universal machines for which $\Omega_{\mathbf{U}}[\{\sigma\}]$ is just $2^{1-K(\sigma)}$. For such universal machines there exists a co-c.e. set X such that $\Omega_{\mathbf{U}}[X]$ is neither left-c.e. nor random.

We also look at the range of $\Omega_{\mathbf{U}}$ as an operator. The set $\{\Omega_{\mathbf{U}}[X] \colon X \subseteq 2^{<\omega}\}$ is proved to be a finite union of closed intervals. It follows that for any universal machine $\mathbf{U}$ and any sufficiently small real r, there is a set $X \subseteq 2^{<\omega}$ computable in $\emptyset' \oplus r$, such that $\Omega_{\mathbf{U}}[X] = r$. Then there are Δ^0_2 sets X such that $\Omega_{\mathbf{U}}[X]$ is rational, hence far from random.

The same questions are also considered in the context of infinite computations, and lead to similar results.

This chapter comprises the joint work [9] with Verónica Becher, Serge Grigorieff and Joseph S. Miller and also part of joint work [39] with Frank Stephan and Guohua Wu.

3.1 Introduction

In 2002, Serge Grigorieff has put forward the following conjecture on randomness, in the spirit of Rice's theorem for computability, which generalizes Chaitin's Ω numbers. It involves the notion of *universal* prefix-free Turing machine as defined in Definition 1.4.3.

Conjecture 3.1.1. *For any non-empty $X \subseteq 2^{<\omega}$, the probability $\Omega_{\mathbf{U}}[X]$ that a universal prefix-free Turing machine $\mathbf{U}$ on an arbitrary input halts and gives an output in X is random. Moreover, if X is Σ^0_n-hard then this probability is n-random (i.e. random in $\emptyset^{(n-1)}$).*

The formal definition of $\Omega_{\mathbf{M}}[X]$, the probability that machine $\mathbf{M}$ halts and gives an output inside a set X, is the following:

Definition 3.1.2 (Halting probability)**.** Let $\mathbf{M} \colon 2^{<\omega} \to 2^{<\omega}$ denote a prefix-free

Turing machine. For $X \subseteq 2^{<\omega}$, let $\mathbf{M}^{-1}(X) = \{\rho \in 2^{<\omega} \colon \mathbf{M}(\rho) \in X\}$ and define

$$\begin{aligned} \Omega_{\mathbf{M}}[X] &= \sum_{\rho \in \mathbf{M}^{-1}(X)} 2^{-|\rho|} \\ &= \mu\left(\mathbf{M}^{-1}(X)2^{\omega}\right) \end{aligned}$$

It turns out that the notion of universality considered is decisive in this conjecture.

The sense in which $\Omega_{\mathbf{U}}[X]$ is a genuine probability is considered in section 3.2. Section 3.3 is devoted to the notion of universal machine and introduces some particularizations. In section 3.4 we mention some known positive instances of the conjecture and we reproduce two proofs of randomness: the original one introduced by Chaitin [23] and a simpler version.

In section 3.5 we prove that the first part of the conjecture is false as stated, showing that there are Δ^0_2 sets that do not lead to randomness, whatever be the universal machine. We improve this result showing that the second part is also false. In fact, no Σ^0_n sets can be n-random.

Section 3.6 gives positive instances of Conjecture 3.1.1. We show that the conjecture holds for Σ^0_n-complete sets and Σ^0_n-complete sets with randomness, whatever be the universal machine.

In section 3.7 we prove that for any universal machine $\mathbf{U}$ and any sufficiently small real A, there is a set $X \subseteq 2^{<\omega}$ computable in $\emptyset' \oplus A$, such that $\Omega_{\mathbf{U}}[X] = A$. In particular, this result asserts that for any universal machine $\mathbf{U}$ there are Δ^0_2 sets X such that $\Omega_{\mathbf{U}}[X]$ is a rational number, the farthest to be random that it can be. This also yields that the range $\{\Omega_{\mathbf{U}}[X] \colon X \subseteq 2^{<\omega}\}$ is a finite union of closed intervals. We also show that for any given left-c.e. random real r, and for any given c.e. set X, there is a universal machine $\mathbf{U}$ such that $r = \Omega_{\mathbf{U}}[X]$.

Finally, in section 3.8 we study the version of the conjecture for infinite computations on monotone machines, a landscape where more positive instances have been obtained.

3.2 Uniform probability on $(2^{<\omega}, \preceq)$ and $\Omega_{\mathbf{U}}[X]$

As done in the above Conjecture 3.1.1, it is usual to name $\Omega_{\mathbf{U}}[X]$ as the *probability* that $\mathbf{U}$ halts and produces an output in X. In which precise sense is this real $\Omega_{\mathbf{U}}[X]$ a probability? The function $\rho \mapsto 2^{-|\rho|}$ induces the usual uniform probability on the set 2^n of words of fixed length n, for any n. However, as concerns the whole space of words $2^{<\omega}$, it induces a measure which takes value $+\infty$ on $2^{<\omega}$, hence $\Omega_{\mathbf{U}}$ is not a probability.

There are two ways to look at $\Omega_{\mathbf{U}}[X]$ as a probability. Using the fact that $\mathbf{U}^{-1}(X)$ is prefix-free (as is the domain of $\mathbf{U}$), a first simple solution is to embed finite inputs into infinite ones and to consider the usual Lebesgue measure on 2^{ω}. This amounts to the equality stated in Definition 3.1.2:

$$\Omega_{\mathbf{U}}[X] = \sum_{\rho \in \mathbf{U}^{-1}(X)} 2^{-|\rho|} = \mu\left(\mathbf{U}^{-1}(X)2^{\omega}\right).$$

Another solution, which keeps within the space of finite words, is to consider a notion of probability on ordered sets for which the additivity axiom $p(A \cup B) = p(A) + p(B)$ is not supposed for general disjoint events $A, B \subseteq 2^{<\omega}$ but only for incompatible ones with respect to the ordering. In case of $(2^{<\omega}, \preceq)$, this means $A2^{<\omega} \cap B2^{<\omega} = \emptyset$.

The next Proposition shows that $\mu(A2^\omega)$ can me seen as the probability of A defined for the discrete case, in the limit: the number of elements of length (up to) n extending some string of A divided by the total number of elements of length (up to) n.

Proposition 3.2.1. *For all $A \subseteq 2^{<\omega}$,*

$$\mu(A2^\omega) = \lim_{n\to\infty} \frac{\|A2^{<\omega} \cap 2^n\|}{2^n} = \lim_{n\to\infty} \frac{\|A2^{<\omega} \cap 2^{\leq n}\|}{2^{n+1}-1}$$

Proof. Let $\min A$ be the prefix-free set of minimal elements of A relative to the prefix ordering. Then $\mu(A2^\omega) = \sum_{\rho\in\min A} 2^{-|\rho|}$ and for every n,

$$A2^{<\omega} \cap 2^n = \bigcup_{\rho\in(\min A)\cap 2^{\leq n}} \rho 2^{n-|\rho|}$$

and

$$\frac{\|A2^{<\omega} \cap 2^n\|}{2^n} = \sum_{\rho\in(\min A)\cap 2^{\leq n}} 2^{-|\rho|}$$

This proves that $\frac{\|A2^{<\omega}\cap 2^n\|}{2^n}$ is monotone non-decreasing in n with limit $\mu(A2^\omega)$. Also it is not difficult to see that

$$\frac{\|A2^{<\omega} \cap 2^{\leq n}\|}{2^{n+1}} = a + b$$

where

$$\begin{aligned} a &= \frac{\|A2^{<\omega} \cap 2^{\leq k}\|}{2^{n+1}}; \\ b &= \sum_{k<m\leq n} \frac{\|A2^{<\omega} \cap 2^m\|}{2^m} 2^{-(n-m+1)}. \end{aligned}$$

Given $\varepsilon > 0$, fix k such that $0 \leq \mu(A2^\omega) - \frac{\|A2^{<\omega}\cap 2^m\|}{2^m} \leq \varepsilon/3$ for all $m \geq k$. Then, for $n \geq k + \log(3/\varepsilon)$,

$$a \leq \frac{2^{k+1}-1}{2^{n+1}} \leq 2^{-(n-k)} \leq \varepsilon/3$$

and

$$\begin{aligned} \mu(A2^\omega) - b &= \mu(A2^\omega)\, 2^{-(n-k)} + \mu(A2^\omega) \sum_{k<m\leq n} 2^{-(n-m+1)} \\ &\quad - \sum_{k<m\leq n} \frac{\|A2^{<\omega} \cap 2^m\|}{2^m} 2^{-(n-m+1)}. \end{aligned}$$

Hence, $|\mu(A2^\omega) - b|$ is at most

$$2^{-(n-k)} + \sum_{k<m\leq n} \left| \mu(A2^\omega) - \frac{\|A2^{<\omega} \cap 2^m\|}{2^m} \right| 2^{-(n-m+1)}$$

and this is bounded by $\varepsilon/3 + \varepsilon/3$. Hence

$$\left| \mu(A2^\omega) - \frac{\|A2^{<\omega} \cap 2^{\leq n}\|}{2^{n+1}} \right| \leq a + |\mu(A2^\omega) - b| \leq \varepsilon.$$

This proves that

$$\frac{\|A2^{<\omega} \cap 2^{\leq n}\|}{2^{n+1}}$$

also tends to $\mu(A2^\omega)$ when $n \to \infty$. □

Definition 3.2.2. We let $\pi\colon P(2^{<\omega}) \to [0,1]$ be the function such that, for all $A \subseteq 2^{<\omega}$,

$$\pi(A) = \lim_{n\to\infty} \frac{\|A2^{<\omega} \cap 2^n\|}{2^n}$$

A straightforward application of Proposition 3.2.1 shows that π is a probability on the ordered set $(2^{<\omega}, \preceq)$.

Proposition 3.2.3. *In the sense of the ordered set* $(2^{<\omega}, \preceq)$*,* π *is a probability, i.e.* $\pi(\emptyset) = 0$*,* $\pi(2^{<\omega}) = 1$ *and, for all* $A, B \subseteq 2^{<\omega}$

$$\begin{aligned} \pi(A \cup B) &\leq \pi(A) + \pi(B) \\ \pi(A \cup B) = \pi(A) + \pi(B) &\Leftrightarrow A2^{<\omega} \cap B2^{<\omega} = \emptyset \end{aligned}$$

Also, $\pi(A) = \pi(\min(A)) = \pi(A2^{<\omega})$ *and, if* A *is prefix-free then* $\pi(A) = \sum_{\rho\in A} 2^{-|\rho|}$.

Since for any prefix-free machine $\mathbf{M}$, dom($\mathbf{M}$) is prefix-free, we see that $\Omega_{\mathbf{M}}[X]$ is the probability of $\mathbf{M}^{-1}(X)$ relative to π.

Proposition 3.2.4. *Let* $\mathbf{M}\colon 2^{<\omega} \to 2^{<\omega}$ *be a prefix-free Turing machine and* $X \subseteq 2^{<\omega}$*. Then,* $\Omega_{\mathbf{M}}[X] = \pi(\mathbf{M}^{-1}(X))$*.*

3.3 On the notion of universality

For some sets the validity of Conjecture 3.1.1 depends on the machine $\mathbf{U}$ used to define $\Omega_{\mathbf{U}}$; we shall consider the usual notion of universality of Definition 1.4.3 and also a refinement that we name *universality by adjunction*.

As in section 1.4 assume $(\mathbf{T}_e)_{e\in\mathbb{N}}$ is a computable enumeration of all prefix-free Turing machines.

Definition 3.3.1 (Universality by adjunction)**.** Let $\mathbf{U}\colon 2^{<\omega} \to 2^{<\omega}$ be a prefix-free Turing machine. $\mathbf{U}$ is *universal by adjunction* if and only if

$$(\forall e)(\exists \gamma_e)(\forall \rho)\, \mathbf{U}(\gamma_e \rho) = \mathbf{T}_e(\rho).$$

Hence, in terms of Definition 1.4.3 we are in the special situation where, $c_e = |\gamma_e|$ and $\gamma_{e,\rho} = \gamma_e \rho$.

Clearly every universal by adjunction machine is universal. We say that $\mathbf{U}$ is *effectively universal* if there is a total computable function $c\colon \mathbb{N} \times 2^{<\omega} \to 2^{<\omega}$ such that we can take $\gamma_{e,p} = c(e,p)$ in Definition 1.4.3. This means that $\gamma_{e,p}$ not only exists, but it can be computably generated from e and p. The next proposition shows that, under some hypothesis being effectively universal already implies universal by adjunction.

Proposition 3.3.2. *Let* $\mathbf{V}$ *be effectively universal such that the associated* $c\colon \mathbb{N} \times 2^{<\omega} \to 2^{<\omega}$ *is injective and has computable range. Then there exists a machine* $\mathbf{U}$ *universal by adjunction such that*

$$(\forall \sigma \in 2^{<\omega})\, \Omega_{\mathbf{U}}[\{\sigma\}] = \Omega_{\mathbf{V}}[\{\sigma\}]. \tag{3.1}$$

Proof. Since $\mathbf{V}$ is universal, $\Omega_{\mathbf{V}}[2^{<\omega}]$ is random, hence less than 1 and so there exists k such that $\Omega_{\mathbf{V}}[2^{<\omega}] < 1 - 2^{-k}$. Fix such a k. The idea of the proof is as follows: first, define $\mathbf{U}$ on a prefix-free subset of $0^{k+1}2^{<\omega}$ in a way that ensures that $\mathbf{U}$ is universal by adjunction. Then define $\mathbf{U}$ on a prefix-free subset of $0^{\leq k}12^{<\omega}$ to get condition (3.1).

For (e,ρ) such that $c(e,\rho) \in \mathrm{dom}(\mathbf{V})$, and $n \in \mathbb{N}$ such that $|c(e,\rho)| \leq |\rho| + n$, we set

$$\mathbf{U}(0^{k+1+n}\ 1^{e+1}\ 0\ \rho) = \mathbf{V}(c(e,\rho)). \tag{3.2}$$

Since $\mathbf{V}(c(e,\rho)) = \mathbf{T}_e(\rho)$ we see that $\mathbf{U}(0^{k+1+n}1^{e+1}0\rho) = \mathbf{T}_e(\rho)$ for all $n \geq |c(e,\rho)| - |\rho|$. The universality of $\mathbf{V}$ ensures that there exists c_e such that $|c(e,\rho)| \leq |\rho| + c_e$ for all ρ. Then $\mathbf{U}(0^{k+1+c_e}1^{e+1}0\rho) = \mathbf{T}_e(\rho)$ for all ρ. This proves that $\mathbf{U}$ is universal by adjunction with $\sigma_e = 0^{k+1+c_e}1^{e+1}0$. Observe that for given e and ρ,

$$\begin{aligned}\sum_{n \geq \max(0,|c(e,\rho)|-|\rho|)} 2^{-|0^{k+1+n}1^{e+1}0\rho|} &= 2^{-(k+e+3)} \sum_{n \geq \max(0,|c(e,\rho)|-|p|)} 2^{-(n+|\rho|)} \\ &= 2^{-(k+2+e+\max(|\rho|,|c(e,\rho)|))}.\end{aligned}$$

Let $Q_{e,\rho}$ be the finite subset of $\mathbb{N}$ such that

$$\sum_{j \in Q_{e,\rho}} 2^{-j} = 2^{-|c(e,\rho)|} - 2^{-(k+2+e+\max(|p|,|c(e,\rho)|))}.$$

To define $\mathbf{U}$ on $\bigcup_{n \leq k} 0^n 12^{<\omega}$, we introduce the following Kraft-Chaitin set

$$\begin{aligned}W &= \{\langle j,, \mathbf{V}(c(e,\rho))\rangle \colon (e,\rho) \in \mathrm{dom}(\mathbf{V} \circ c), j \in Q_{e,\rho}\} \cup \\ &\quad \cup \{\langle |\gamma|, \mathbf{V}(\gamma)\rangle \colon \gamma \in \mathrm{dom}(\mathbf{V}) \setminus \mathrm{ran}(c)\}.\end{aligned}$$

Since the range of c is computable, there is a computable enumeration $(l_n, \sigma_n)_{n \in \mathbb{N}}$

of W. Let us show that W is indeed a Kraft-Chaitin set.

$$\begin{aligned}\mathrm{wt}\,(W) &= \sum_{(e,\rho)\in\mathrm{dom}(V\circ c)}\ \sum_{j\in Q_{e,p}} 2^{-j} + \sum_{\gamma\in\mathrm{dom}(\mathbf{V})\setminus\mathrm{ran}(c)} 2^{-|\gamma|}\\ &\leq \sum_{(e,\rho)\in\mathrm{dom}(\mathbf{V}\circ c)} 2^{-|c(e,\rho)|} + \sum_{\gamma\in\mathrm{dom}(\mathbf{V})\setminus\mathrm{ran}(c)} 2^{-|\gamma|}\\ &\leq \sum_{\gamma\in\mathrm{dom}(\mathbf{V})\cap\mathrm{ran}(c)} 2^{-|\gamma|} + \sum_{\gamma\in\mathrm{dom}(\mathbf{V})\setminus\mathrm{ran}(c)} 2^{-|\gamma|}\\ &< 1-2^{-k}.\end{aligned}$$

A straightforward extension of the Kraft-Chaitin theorem shows that there is a c.e. set $\{\rho_n\colon n\in\mathbb{N}\}$ which is a prefix-free subset of $0^{\leq k}12^{<\omega}$ and $|\rho_n| = l_n$ for all n. We complete the definition of $\mathbf{U}$ on $0^{\leq k}12^{<\omega}$ by setting for all n

$$\mathbf{U}(\rho_n) \;=\; \sigma_n. \tag{3.3}$$

Observe that $\mathbf{U}$, as defined by (3.2) and (3.3), has prefix-free domain. Also, for $\sigma\in 2^{<\omega}$, we have

$$\begin{aligned}\Omega_{\mathbf{V}}[\{\sigma\}] &= \sum\{2^{-|\gamma|}\colon \gamma\in\mathrm{dom}(\mathbf{V})\cap\mathrm{ran}(c)\wedge\mathbf{V}(\gamma)=\sigma\} &(3.4)\\ &\quad+\sum\{2^{-|\gamma|}\colon \gamma\in\mathrm{dom}(\mathbf{V})\setminus\mathrm{ran}(c)\wedge\mathbf{V}(\gamma)=\sigma\}. &(3.5)\end{aligned}$$

Take γ as in (3.4). Since c is injective, there is a unique pair (e,ρ) such that $\gamma = c(e,\rho)$. Thus, the sum (3.4) is exactly

$$\sum_{\mathbf{V}(c(e,\rho))=\sigma} 2^{-|c(e,\rho)|}.$$

The $\mathbf{U}$-descriptions of type (3.2) of σ add $2^{-(k+2+e+\max(|\rho|,|c(e,\rho)|))}$ to $\Omega_{\mathbf{U}}[\{\sigma\}]$, for any (e,ρ) such that $\mathbf{V}(c(e,\rho))=\sigma$; the $\mathbf{U}$-descriptions of type (3.3) add $2^{-|c(e,\rho)|} - 2^{-(k+2+e+\max(|\rho|,|c(e,\rho)|))}$ to $\Omega_{\mathbf{U}}[\{\sigma\}]$ for any (e,ρ) such that $\mathbf{V}(c(e,\rho))=\sigma$. So, in total, for every (e,ρ) such that $\mathbf{V}(c(e,\rho))=\sigma$ contributes $2^{-|c(e,\rho)|}$ to $\Omega_{\mathbf{U}}[\{\sigma\}]$.

For $\gamma\in\mathrm{dom}(\mathbf{V})\setminus\mathrm{ran}(c)$ we have $\mathbf{U}(\gamma)=\mathbf{V}(\gamma)$ so there are additional $\mathbf{U}$-descriptions that contributes to $\Omega_{\mathbf{U}}[\{\sigma\}]$ exactly with (3.5). Thus, $\Omega_{\mathbf{U}}[\{\sigma\}] = \Omega_{\mathbf{V}}[\{\sigma\}]$. □

Remark 3.3.3. In the proposition above, the condition that c has computable image is used to see that W is c.e. This condition can be replaced by the c.e. character of $\mathrm{dom}(V)\setminus\mathrm{ran}(c)$.

3.4 Known positive instances of the Conjecture

The first result supporting the conjecture is Chaitin's [23] random real Ω, and corresponds to the case $\Omega_{\mathbf{U}}[X]$ where $X = 2^{<\omega}$. The real Ω depends on $\mathbf{U}$, the universal machine, but independently of the universal machine $\mathbf{U}$ used in the definition, each $\Omega_{\mathbf{U}}$ is random. We now give Chaitin's original proof and a simpler version of the fact that $\Omega_{\mathbf{U}}$ is random.

Theorem 3.4.1 (Chaitin [23]). *For any universal prefix-free machine* $\mathbf{U}$, $\Omega_{\mathbf{U}}$ *is random.*

Proof. (Chaitin's version). Let $\mathbf{U}$ be given. For simplicity, let us write Ω for $\Omega_{\mathbf{U}}$. Let $(\rho_i)_{i\in\mathbb{N}}$ be a computable enumeration of the set $\{\rho\colon \mathbf{U}(\rho)\downarrow\}$ and let Ω_s be the following computable approximation of Ω:

$$\Omega_s = \sum_{|\rho_i|\leq s} 2^{-|\rho_i|}.$$

Let $\mathbf{M}$ be the following machine with input ρ:

1. Simulate $\mathbf{U}(\rho)$ until it halts. Let $\sigma = \mathbf{U}(\rho)$ and let $n = |\sigma|$.

2. Find a stage s such that $.\sigma - \Omega_s < 2^{-n}$.

3. Let $A = \{\mathbf{U}(\rho_i)\colon i \leq s \wedge |\rho_i| \leq n-1\}$.

4. Return the least string not in A.

Assume ρ is a minimal program for $\Omega \upharpoonright n$, that is $\mathbf{U}(\rho) = \Omega \upharpoonright n$ and $|\rho| = K(\Omega \upharpoonright n)$. Then at step 2, the machine $\mathbf{M}$ finds a stage s such that $.\Omega \upharpoonright n - \Omega_s < 2^{-n}$. Observe that it is always possible to find such an s because $\Omega_s \to \Omega$ from below, so there is an s with $\Omega - \Omega_s < 2^{-n}$ and $.\Omega \upharpoonright n \leq \Omega$.

Lemma 3.4.2. $A = \{\sigma\colon K(\sigma) \leq n-1\}$.

Proof. Suppose $\sigma \in A$, as defined in step 3. Then $\sigma = \mathbf{U}(\rho_i)$ for some $i \leq s$ and $|\rho_i| \leq n-1$. Hence clearly, $K(\sigma) \leq |\rho_i| \leq n-1$.

Suppose σ such that $K(\sigma) \leq n-1$ and assume by way of contradiction that $\sigma \notin A$. Since $K(\sigma) \leq n-1$ then there is a program γ such that $\mathbf{U}(\gamma) = \sigma$ and $|\gamma| \leq n-1$. Now, γ cannot be in the set $\{\rho_i\colon i \leq s\}$ because otherwise $\gamma = \rho_i$ for some $i \leq s$ and this would mean that $\sigma \in A$. Therefore, γ does not contribute to Ω_s, but it certainly contributes to Ω, as $\mathbf{U}(\gamma)\downarrow$. So we have

$$\Omega \geq \Omega_s + 2^{-|\gamma|} \geq \Omega_s + 2^{-n+1}$$

and thus

$$\begin{aligned} \Omega - .\Omega \upharpoonright n &= \Omega - \Omega_s + \Omega_s - .\Omega \upharpoonright n \\ &\geq 2^{-n+1} - (.\Omega \upharpoonright n - \Omega_s) \\ &> 2^{-n+1} - 2^{-n} = 2^{-n}. \end{aligned}$$

This is a contradiction. □

By Lemma 3.4.2, at step 4, $\mathbf{M}$ outputs a string τ such that $K(\tau) > n-1$. Hence, if c is the constant for $\mathbf{M}$, we have

$$n-1 < K(\tau) \leq |\rho| + c = K(\Omega \upharpoonright n) + c,$$

and so Ω is random. □

Observe that in step 2, what one would really like is to find some s such that $\Omega_s \restriction n = \sigma$. However, in case Ω is a dyadic rational this might not be possible. Indeed, suppose $\Omega = .01$. Then Ω_s might take values .0, .001, .0011, .00111... as s increases. In this case, there is no s such that $\Omega \restriction 2 = \Omega_s \restriction 2$.

Next, we show Nies' proof that Ω is random [61]. It is based in computation steps rather than in outputs. A similar proof can be found in [31, 32]. However in that proof the Recursion Theorem is used, instead it is not used in the following one.

Proof of Theorem 3.4.1 (another version). Let $\mathbf{U}$ be given. For simplicity, let us write Ω for $\Omega_{\mathbf{U}}$. Let $\mathbf{U}_t(\rho)$ be the simulation of $\mathbf{U}(\rho)$ by stage s with the additional restriction that if $\mathbf{U}_s(\rho) \downarrow$ then $s > |\mathbf{U}(\rho)|$. That is, in case $\mathbf{U}(\rho) \downarrow$, then the simulation of $\mathbf{U}(\rho)$ takes more than n steps to reach the halting state, where n is the length of the output. Let

$$\Omega_s = \sum_{\mathbf{U}_s(\rho)\downarrow} 2^{-|\rho|}$$

and let $\mathbf{M}$ be the following machine with input ρ:

1. Find the least stage t such that $\mathbf{U}_t(\rho) \downarrow$. Let $\sigma = \mathbf{U}_t(\rho)$ and let $n = |\sigma|$.
2. Find the least stage $s \geq t$ such that $.\sigma - \Omega_s < 2^{-n}$.
3. Return 0^{s+1}.

Let c be the constant for machine $\mathbf{M}$. Suppose that $K(\Omega \restriction n) < n - c - 1$. There is a program ρ such that $\mathbf{U}(\rho) = \Omega \restriction n$ and $|\rho| \leq n - c - 1$. Let s be as in step 2, so that $\mathbf{U}_s(\rho) = \Omega \restriction n$ and $.\Omega \restriction n - \Omega_s < 2^{-n}$. Then $\mathbf{M}(\rho) = 0^{s+1}$ and therefore there is γ such that $\mathbf{U}(\gamma) = 0^{s+1}$ and $|\gamma| \leq |\rho| + c \leq n - 1$. This computation cannot be defined by stage s because the output is too large, but it will eventually become defined later, so it contributes to Ω. So we have

$$\Omega \geq \Omega_s + 2^{-|\gamma|} \geq \Omega_s + 2^{-n+1}. \tag{3.6}$$

The proof now goes on as in the last part of the proof of Lemma 3.4.2 yielding a contradiction from (3.6). □

Chaitin observed [25] that if $\mathbf{U}$ is universal and X is infinite and c.e. then $\Omega_{\mathbf{U}}[X]$ is random. Here is a proof of this fact.

Theorem 3.4.3. *Let $\mathbf{U}$ be universal. If $X \subseteq 2^{<\omega}$ is c.e. and infinite then $\Omega_{\mathbf{U}}[X]$ is random.*

Proof. Follow Chaitin's proof of Theorem 3.4.1, replacing $\Omega_{\mathbf{U}}$ by $\Omega_{\mathbf{U}}[X]$. Observe that since X is c.e. we can computably approximate $\Omega_{\mathbf{U}}[X]$, as we did it with Ω in the cited proof. We let $(\rho_i)_{i\in\mathbb{N}}$ be a computable enumeration of the set $\{\rho\colon \mathbf{U}(\rho) \in X\}$ and at step 4, $\mathbf{M}$ outputs the least string in $X \setminus A$. Since X is infinite and c.e., this is always possible. □

Becher et al. [14] proved that in case $\mathbf{U}$ is universal by adjunction, the above Theorem also applies for finite sets $X \neq \emptyset$.

Theorem 3.4.4. *Let $\mathbf{U}$ be universal by adjunction. If $X \subseteq 2^{<\omega}$ is non-empty and c.e. then $\Omega_{\mathbf{U}}[X]$ is random.*

Proof. Let $\sigma \in X$ Let $\mathbf{M}$ be the prefix-free machine which on input ρ it executes $\mathbf{U}(\rho)$ and, if defined, it outputs σ. Observe that $\mathbf{M}(\rho) \in X$ if and only if $\mathbf{U}(\rho) \downarrow$. Since $\mathbf{U}$ is universal by adjunction, there is γ such that for all ρ, $\mathbf{U}(\gamma\rho) = \mathbf{M}(\rho)$. Now,

$$\begin{aligned}
\Omega_{\mathbf{U}}[X] &= \sum_{\mathbf{U}(\rho)\in X} 2^{-|\rho|} \\
&= \sum_{\mathbf{U}(\gamma\rho)\in X} 2^{-|\gamma\rho|} + \sum_{\mathbf{U}(\rho)\in X, \gamma \not\preceq \rho} 2^{-|\rho|} \\
&= \sum_{\mathbf{U}(\rho)\downarrow} 2^{-|\gamma\rho|} + \sum_{\mathbf{U}(\rho)\in X, \gamma \not\preceq \rho} 2^{-|\rho|} \\
&= 2^{-|\gamma|}\Omega_{\mathbf{U}} + r
\end{aligned}$$

where r is a left-c.e. real. In [21] it was proved that if a and b are both left-c.e. and a is random then $a + b$ is random. This implies that $\Omega_{\mathbf{U}}[X]$ is random. □

By Definition 1.4.3 we know that $\mathbf{U}$ is able to simulate $\mathbf{M}$ in an optimal way, in the sense that for all ρ there is γ such that $\mathbf{U}(\gamma) = \mathbf{M}(\rho)$ and $|\gamma| \leq |\rho| + c$, for some constant c depending only on $\mathbf{M}$. Although one might think that if $\mathbf{U}$ is universal then the execution of $\mathbf{U}(\gamma)$ needs to take at least as many steps as the execution of $\mathbf{M}(\rho)$, nothing is said in the definition about this constraint. Let us call a machine $\mathbf{U}$ *universal by time* if it is universal and, following the notation of Definition 1.4.3,

$$\min\{t \colon \mathbf{U}_t(\gamma_{e,\rho}) \downarrow\} \geq \min\{t \colon \mathbf{T}_{e,t}(\rho) \downarrow\}.$$

Observe that in the second version of Theorem 3.4.1 there is nothing special for returning 0^{s+1} at step 3. It would be feasible to output any string such ν for which the simulation of $\mathbf{M}$ in $\mathbf{U}$ is not defined by stage s. Following the same idea as in the second proof of Theorem 3.4.1 it is possible to prove that if $\mathbf{U}$ is universal by time and $X \neq \emptyset$ is c.e. then $\Omega_{\mathbf{U}}[X]$ is random: roughly, replace Ω_s by the approximation $\Omega_{\mathbf{U}}[X]_s$ of the left-c.e. real $\Omega_{\mathbf{U}}[X]$. At step 3, $\mathbf{M}$ makes sure to have made at least $s + 1$ steps and returns any string of X. Since the computation of $\mathbf{M}(\rho)$ takes at least $s + 1$ steps, $\mathbf{U}(\gamma)$ also does and hence this computation cannot be defined by stage s.

3.5 Negative results about the Conjecture

In this section we show some negative instances of the Conjecture. We first see that if we can freely choose the underlying universal machine then the conjecture fails.

Proposition 3.5.1. *There is a universal Turing machine* $\mathbf{U}$ *such that* $\Omega_{\mathbf{U}}[\{\sigma\}] = 2^{1-K_{\mathbf{U}}(\sigma)}$ *for all* $\sigma \in 2^{<\omega}$*.*

Proof. Let $\mathbf{V}$ be a universal prefix-free Turing machine. The new universal machine $\mathbf{U}$ is constructed by the following Kraft-Chaitin set:

$$W = \{\langle n, \sigma\rangle \colon \sigma \in 2^{<\omega} \wedge n > K_{\mathbf{V}}(\sigma)\}.$$

Notice that W is a Kraft-Chaitin set in the sense that there is a computable enumeration $\langle n_0, \sigma_0\rangle, \langle n_1, \sigma_1\rangle, \dots$ of W which satisfies the preconditions of the Kraft-Chaitin Theorem: on the one hand, $K_{\mathbf{V}}$ may be approximated from above and therefore $\langle n_0, \sigma_0\rangle, \langle n_1, \sigma_1\rangle, \dots$ exists and on the other hand, one can choose this enumeration to be one-one. Then

$$\begin{aligned} \mathrm{wt}\,(W) &= \sum_{m\geq 0} 2^{-n_m} \\ &\leq \sum_{\sigma\in 2^{<\omega}} 2^{-K_{\mathbf{V}}(\sigma)} \\ &\leq \Omega_{\mathbf{V}} < 1. \end{aligned}$$

Let $\mathbf{U}$ be the prefix-free machine whose existence is guaranteed by the Kraft-Chaitin Theorem.

Now, for each σ and each m there is exactly one program $\rho_{\sigma,m}$ of length $K_{\mathbf{V}}(\sigma) + m + 1$ such that $\mathbf{U}(\rho_{\sigma,m}) = \sigma$ and no program of length up to $K_{\mathbf{V}}(\sigma)$ generates σ. Thus, for every $\sigma \in 2^{<\omega}$, we have $K_{\mathbf{U}}(\sigma) = K_{\mathbf{V}}(\sigma) + 1$ and then $\mathbf{U}$ is universal because $\mathbf{V}$ is. Also

$$\begin{aligned} \Omega_{\mathbf{U}}[\{\sigma\}] &= \sum_{n>K_{\mathbf{V}}(\sigma)} 2^{-n} \\ &= 2^{-K_{\mathbf{V}}(\sigma)} = 2^{1-K_{\mathbf{U}}(\sigma)}, \end{aligned}$$

and this completes the proof. □

Notice that by Theorem 3.4.4, the constructed $\mathbf{U}$ cannot be universal by adjunction since $\Omega_U[\{\sigma\}]$ is rational.

Proposition 3.5.2. *Every prefix-free Turing machine* $\mathbf{M}$ *has a restriction* $\mathbf{M}'$ *to some c.e. set such that* $K_{\mathbf{M}} = K_{\mathbf{M}'}$ *(hence* $\mathbf{M}'$ *is universal whenever* $\mathbf{M}$ *is) and* $\Omega_{\mathbf{M}'}[X]$ *is rational (hence not random), for every finite set* $X \subseteq 2^{<\omega}$*.*

Proof. Let $(\rho_i, \sigma_i)_{i\in\mathbb{N}}$ be a computable enumeration of the graph of $\mathbf{M}$. Define a total computable function $f \colon \mathbb{N} \to \mathbb{N}$ such that $f(i)$ is the smallest $j \leq i$ satisfying

$$\sigma_j = \sigma_i \ , \ |\rho_j| = \min\{|\rho_k| \colon k \leq i \wedge \sigma_k = \sigma_i\}.$$

Let $\mathbf{M}'$ be the prefix-free machine with graph $\{(\rho_{f(i)}, \sigma_{f(i)}) : i \in \mathbb{N}\}$. Clearly, $\mathbf{M}'$ is a restriction of $\mathbf{M}$ to some c.e. set. Also, for every $\sigma \in 2^{<\omega}$, if j is the least such that $\sigma = \sigma_j$ and $|\rho_j| = K_{\mathbf{M}}(\sigma)$ then $f(i) = j$ for all $i \geq j$ such that $\sigma_i = \sigma$. Therefore, $\mathbf{M}'^{-1}(\{\sigma\})$ is finite, so $\Omega_{\mathbf{M}'}[\{\sigma\}] = \sum_{\rho\in\mathbf{M}'^{-1}(\sigma)} 2^{-|\rho|}$ is a finite sum of rational numbers and hence is rational. The same is true for finite sets $X \subseteq 2^{<\omega}$. □

Applying the above Proposition to a universal machine $\mathbf{U}$, we get the following straightforward corollary, also a consequence of Proposition 3.5.1.

Corollary 3.5.3. *There is a universal Turing machine* $\mathbf{U}$ *such that for every finite set* $X \subseteq 2^{<\omega}$ *the real* $\Omega_{\mathbf{U}}[X]$ *is rational, hence not random.*

The following proposition shows the existence of an infinite co-c.e. set $X \subseteq \mathbb{N}$ such that for any two elements $x, y \in X$ with $x < y$, the program-size complexity of y and beyond is guaranteed to be much larger than the one of the strings x and the elements smaller than x. The basic idea of the construction is to check in every stage for every current elements x, y with $x < y$ whether the strings beyond y are much more complicated than x in the way specified below and to enumerate y into the complement of X whenever it turns out that this is not the case. Note that the construction of X does not make any requirements on $\mathbf{U}$, so it works for every universal machine.

We asked if there is an analog of Theorem 3.4.3 for co-c.e. sets. This question appeared later in [58, Question 8.10]. Of course we know that it is not true for finite sets, but we might analyze the randomness $\Omega_{\mathbf{U}}[X]$ for infinite co-c.e. sets. Using the next Proposition 3.5.4, we obtain a partial negative answer to this question for a specific universal machine. But this is not an answer of this question since Question 8.10 considers only machines which are universal by adjunction. Such machines are more difficult to handle.

Proposition 3.5.4. *There is an infinite co-c.e. set* $X \subseteq \mathbb{N}$ *such that for no* $x, y \in X$ *with* $x < y$ *there are* v, w *such that* $y \leq w$, $K(w) \leq v$ *and* $K(v) \leq x$.

Proof. One constructs the complement Y of X by stages:

Stage 0: Let $Y_0 = \emptyset$.

Stage $s+1$: A number $y \in \{0, 1, \ldots, s+1\} \setminus Y_s$ is enumerated into Y_{s+1} iff there are $x, v, w \leq s$ with

$$x < y \wedge x \notin Y_s \wedge y \leq w \wedge K_s(w) \leq v \wedge K_s(v) \leq x. \tag{3.7}$$

It is easy to see that the so constructed enumeration is computable and thus X is a co-c.e. set. Furthermore, if $x, y \in X$ and $x < y$ there cannot be any v, w such that $y \leq w$, $K(w) \leq v$ and $K(v) \leq x$ since there is a state s where $K_s(v) = K(v)$ and $K_s(w) = K(w)$ and so $K_s(v) \leq x$ and $K_s(w) \leq v$.

It remains to show that X is infinite. So assume by way of contradiction that X is finite and let $a_0 = \max X$. Then the following maxima and minima exist:

$$\begin{aligned} a_1 &= \max\{u \colon K(u) \leq a_0\}; \\ a_2 &= \max\{u \colon K(u) \leq a_1\}; \\ t &= \min\{s \colon \{0, 1, \ldots, a_2\} \subseteq X \cup Y_s\}; \\ y &= \min\{z \colon z \notin X \cup Y_t\}. \end{aligned}$$

By assumption y is enumerated into Y at some stage $s \geq t$ and so there are $x, v, w \leq t$ witnessing this fact in the sense that conditions (3.7) hold. Then $K_s(v) \geq K(v)$ and $K_s(w) \geq K(w)$.

Observe that $x \notin Y_s$ and then $x \notin Y_t$; so if $x \notin X$ then $y \leq x$ and this is not the case. Hence $x \in X$. Now, if $K(v) > a_0$ then $K(v)$ would be greater than all $z \in X$, in particular $K(v) > x$ and this contradicts (3.7). So $K(v) \leq a_0$ and then $v \leq a_1$, and from (3.7) we conclude $K(w) \leq v \leq a_1$. By definition of a_2 and (3.7), $y \leq w \leq a_2$. Since $\{0, 1, \dots, a_2\} \subseteq X \cup Y_t$, we have $y \in X \cup Y_t$ but by definition $y \notin X \cup Y_t$. From this contradiction we conclude that X is infinite. □

Theorem 3.5.5. *There is a universal machine* $\mathbf{U}$ *and a co-c.e. set* X *such that* $\Omega_{\mathbf{U}}[X]$ *is neither left-c.e. nor random.*

Proof. Let $\mathbf{U}$ as in Proposition 3.5.1 and X as in Proposition 3.5.4. Let $K = K_{\mathbf{U}}$. We prove that $\Omega_{\mathbf{U}}[X]$ is as required.

Proposition 3.5.6. $\Omega_{\mathbf{U}}[X]$ *is not random.*

Proof. Note that for $x, y \in X$ with $x < y$, we have $x < K(K(y))$. Indeed, assume $K(K(y)) \leq x$. Take $v = K(y)$ and $w = y$. By the definition of X this cannot be possible. So, for large enough $x, y \in X$ with $x < y$, $K(x) < x < K(y)$. This follows from the fact that for all $\sigma \in 2^{<\omega}$ $K(\sigma) < 2|\sigma| + \mathcal{O}(1)$ and then $K(n) < n + \mathcal{O}(1)$ for almost all $n \in \mathbb{N}$. So

$$\Omega_{\mathbf{U}}[X] = \sum_{x \in X} 2^{1-K(x)}$$

satisfies that all ones (except finitely many) in its binary representation correspond to some term $2^{1-K(x)}$ and that between two ones there is at least one zero, namely, the one corresponding to 2^{1-x}. Thus one knows that after every 1 in the binary representation comes a 0. Therefore $\Omega_{\mathbf{U}}[X]$ is not normal in the scale of 2 and hence not random. □

Proposition 3.5.7. $\Omega_{\mathbf{U}}[X]$ *is not left-c.e.*

Proof. Assume by way of contradiction that $\Omega_{\mathbf{U}}[X]$ is left-c.e. via an approximation $b_0, b_1, \dots$ and let $X = \{x_1, x_2, \dots\}$ such that $x_i < x_{i+1}$. For large enough i, let

$$a = \sum_{y \in X \cap \{0,1,\dots,x_i\}} 2^{1-K(y)}.$$

Observe that a is a rational number and can be represented with the first $K(x_i) + 1$ bits because, as explained in Proposition 3.5.6, for large enough j, $K(x_j) < K(x_{j+1})$. Then the last bit 1 in the binary expansion of a is at position $K(x_i)$.

Given x_i and a, one can compute numbers s, v such that s is the first number with $b_s > a$ and v the least number with $b_s > a + 2^{2-v}$. Note that $\Omega_{\mathbf{U}}[X] \leq a + 2^{2-K(x_{i+1})}$ and thus $2^{2-K(x_{i+1})} > 2^{2-v_x}$. It follows that

$$K(x_{i+1}) < v. \tag{3.8}$$

Since we can represent a with $2(K(x_i) + 1)$ many bits and we can represent x_i with $K(x_i)$ many bits, we obtain

$$K(v) \leq 3K(x_i) + \mathcal{O}(1) \leq x_i \tag{3.9}$$

for large enough i. Take $x = x_i$, $y = w = x_{i+1}$ refer to the definition of X, from Proposition 3.5.4. From (3.8) and (3.9) we obtain that x_{i+1} will eventually be enumerated into the complement of X. This is a contradiction and then $\Omega_{\mathbf{U}}[X]$ cannot be left-c.e. □

This completes the proof of the whole result. □

There might be an alternative approach to prove this result. If one succeeds to construct $\mathbf{U}, X$ such that $\Omega_{\mathbf{U}}[X]$ is neither left-c.e. nor right-c.e., then $\Omega_{\mathbf{U}}[X]$ is not random: as $\Omega_{\mathbf{U}}[X]$ is the difference of the two left-c.e. reals $\Omega_{\mathbf{U}}$ and $\Omega_{\mathbf{U}}[2^{<\omega} \setminus X]$, this follows from a result of Rettinger and Zheng [65].

We now show that the Conjecture also fails for some Δ^0_2 sets regardless of the underlying universal machine.

As Recall that by the Coding Theorem 1.4.6 we have

$$(\exists c_1)(\forall \sigma)\, 2^{-K_{\mathbf{U}}(\sigma)} \leq \Omega_{\mathbf{U}}[\{\sigma\}] \leq 2^{-K_{\mathbf{U}}(\sigma)+c_1}. \tag{3.10}$$

(observe that, according to our definitions, $P_{\mathbf{U}}(\sigma) = \Omega_{\mathbf{U}}[\{\sigma\}]$). Recall also that Chaitin [23] proved

$$(\exists c_2)(\forall \sigma)\, K_{\mathbf{U}}(\sigma) < |\sigma| + K_{\mathbf{U}}(|\sigma|) + c_2 \tag{3.11}$$

and

$$(\exists c_3)\, \|\{\sigma \in 2^m \colon K_{\mathbf{U}}(\sigma) < m + K_{\mathbf{U}}(m) - k\}\| \leq 2^{m-k+c_3}. \tag{3.12}$$

The next lemma can be found in an unpublished work of Solovay [73, IV-20]. We include the proof because Solovay's notes are not universally available.

Lemma 3.5.8. *If* $\mathbf{U}$ *is universal then* $(\exists c_4)(\forall n)(\exists m \leq n)\,[n \leq m + K_{\mathbf{U}}(m) \leq n + c_4]$.

Proof. Choose $c_4 \in \mathbb{N}$ such that $c_4 > K_{\mathbf{U}}(0)$ and $K_{\mathbf{U}}(m+1) \leq K_{\mathbf{U}}(m) + c_4 - 1$, for all $m \in \mathbb{N}$. Given $n \in \mathbb{N}$, let $m \in \mathbb{N}$ be the least number satisfying $n \leq m + K_{\mathbf{U}}(m)$, which clearly holds for some $m \leq n$. We claim that $m + K_{\mathbf{U}}(m) < n + c_4$. This holds because $0 + K_{\mathbf{U}}(0) < c_4 \leq n + c_4$ and, since $m - 1 + K_{\mathbf{U}}(m-1) < n$, then $m + K_{\mathbf{U}}(m) \leq m - 1 + K_{\mathbf{U}}(m-1) + c_4 < n + c_4$. □

Putting these two lemmas together, we get the following result.

Lemma 3.5.9. *If* $\mathbf{U}$ *is universal then* $(\exists d)(\forall n)(\exists \sigma)\,[2^{-n-d} \leq \Omega_{\mathbf{U}}[\{\sigma\}] \leq 2^{-n+d}]$. *In fact, for some constant* d' *there are at least* $2^n/(d'\,n^2)$ *strings* $\sigma \in 2^{<\omega}$ *satisfying the inequalities.*

Proof. Let c_1, c_2, c_3, c_4 be constants as in equations 3.10, 3.11, 3.12 and Lemma 3.5.8. Then $\|\{\sigma \in 2^m \colon K_{\mathbf{U}}(\sigma) < m + K_{\mathbf{U}}(m) - (c_3+1)\}\| \leq 2^{m-1}$, for all $m \in \mathbb{N}$. For $n+c_3+1$, there is an $m \leq n+c_3+1$ such that $n+c_3+1 \leq m+K_{\mathbf{U}}(m) \leq n+c_3+1+c_4$. In particular, all strings σ in 2^m satisfy $K_{\mathbf{U}}(\sigma) \leq m+K_{\mathbf{U}}(m)+c_2 \leq n+c_2+c_3+c_4+1$.

Now, there are at least 2^{m-1} strings $\sigma \in 2^m$ such that $K_{\mathbf{U}}(\sigma) \geq m + K_{\mathbf{U}}(m) - (c_3+1)$ hence such that $K_{\mathbf{U}}(\sigma) \geq n + c_3 + 1 - (c_3+1) = n$. For such strings, we then have $n \leq K_{\mathbf{U}}(\sigma) \leq n + c_2 + c_3 + c_4 + 1$. Therefore, for $d = \max(c_1, c_2 + c_3 + c_4 + 1)$, there are at least 2^{m-1} strings σ such that $2^{-n-d} \leq \Omega_{\mathbf{U}}[\{\sigma\}] \leq 2^{-n+d}$. Finally, note that

$$m - 1 \geq n + c_3 - K_{\mathbf{U}}(m) \geq n - 2\log(m) - \mathcal{O}(1).$$

Therefore, at least $\mathcal{O}(1)2^n/n^2$ strings $\sigma \in 2^{<\omega}$ satisfying

$$2^{-n-d} \leq \Omega_{\mathbf{U}}[\{\sigma\}] \leq 2^{-n+d}.$$

This completes the proof. □

With this lemma we can prove that Conjecture 3.1.1 fails for Δ^0_2 sets.

Theorem 3.5.10. *For every universal* $\mathbf{U}$ *there is a* Δ^0_2 *set* $X \subseteq 2^{<\omega}$ *such that* $\Omega_{\mathbf{U}}[X]$ *is not random.*

Proof. Let $d, d' \in \mathbb{N}$ be the constants from Lemma 3.5.9 and let k be such that $i < 2^i/(d'\,i^2)$ for $i \geq k$. Letting $c = k + d$, Lemma 3.5.9 ensures the existence of a sequence $(\sigma_i)_{i\in\mathbb{N}}$ of distinct strings such that $2^{-i-c-1} < \Omega_{\mathbf{U}}[\{\sigma_i\}] \leq 2^{-i+c}$, for all $i \in \mathbb{N}$. Note that $\emptyset'$ can compute such a sequence (and even compute the set of strings in the sequence). Indeed, denoting by $\mathbf{U}_s$ the computable approximation of $\mathbf{U}$ obtained within s computation steps, $\Omega_{\mathbf{U}}[\{\tau\}]_s = \sum_{\mathbf{U}_s(\rho)=\tau} 2^{-|\rho|}$ is non-decreasing in s and tends to $\Omega_{\mathbf{U}}[\{\tau\}]$ when $s \to \infty$. Thus, for any rational r, $\Omega_{\mathbf{U}}[\{\tau\}] > r$ iff $(\exists s)\, \Omega_{\mathbf{U}}[\{\tau\}]_s > r$. Hence it is decidable in $\emptyset'$ whether $\Omega_{\mathbf{U}}[\{\tau\}] > r$ or not. Notice that $\emptyset'$ can take all strings of $(\sigma_i)_{i\in\mathbb{N}}$ distinct because there are $2^n/(d'n^2)$ many such strings.

We build a Δ^0_2 set X in stages $\{X_s\}_{s\in\mathbb{N}}$. At stage $s+1$ we determine if σ_s is in X in order to ensure that the block of bits of $\Omega_{\mathbf{U}}[X]$ from $s-c$ to $s+c+1$ is not all zeros.

Stage 0. Let $X_0 = \emptyset$.

Stage $s+1$. If $s < c$ then $X_{s+1} = X_s$. Else, using $\emptyset'$, decide if the $2c+2$ bits of $\Omega_{\mathbf{U}}[X_s]$ from $s-c$ to $s+c+1$ are all zero. If these bits are all zero, let $X_{s+1} = X_s \cup \{\sigma_s\}$. Otherwise, let $X_{s+1} = X_s$. Consider the first case. Because $\Omega_{\mathbf{U}}[\{\sigma_s\}] > 2^{-s-c-1}$ there exists $j \leq s+c+1$ such that the j-th bit of $\Omega_{\mathbf{U}}[\{\sigma_s\}]$ is 1. On the other hand, because $\Omega_{\mathbf{U}}[\{\sigma_s\}] \leq 2^{-s+c}$, we have $\Omega_{\mathbf{U}}[\{\sigma_s\}] \upharpoonright s-c-1 = 0^{s-c-1}$. Then there is $s-c \leq j \leq s+c+1$ such that the j-th bit of $\Omega_{\mathbf{U}}[\{\sigma_s\}]$ is 1. Notice that if bit $s-c$ is 1 then all the bits of positions greater than $s-c$ are 0. Hence, $\Omega_{\mathbf{U}}[X_{s+1}] \upharpoonright s-c-1 = \Omega_{\mathbf{U}}[X_s] \upharpoonright s-c-1$. Therefore, the work of earlier stages has been preserved and also $\Omega_{\mathbf{U}}[X_{s+1}]$ is not all zeros on the block of bits from $s-c$ to $s+c+1$.

It follows inductively that, for every s, the block of bits of $\Omega_{\mathbf{U}}[X]$ from $s-c$ to $s+c+1$ is not all zeros. Therefore, $\Omega_{\mathbf{U}}[X]$ is not normal in the scale of 2 and hence not random.

Notice that this construction works independently of the universal machine chosen $\mathbf{U}$ and the binary representation of $\Omega_{\mathbf{U}}[X_s]$ in case such real is a dyadic rational. □

The above result can be dramatically improved: Theorem 3.7.2 shows that there are Δ^0_2 sets X such that $\Omega_{\mathbf{U}}[X]$ is a *rational* number. Another improvement shows that hardness is not enough to get randomness.

Theorem 3.5.11. *For every universal* $\mathbf{U}$ *and any* $A \subseteq \mathbb{N}$*, there is a set* $Z \equiv_T A'$ *such that* $\Omega_{\mathbf{U}}[Z]$ *is not random.*

Proof. First observe that there is a constant b such that

$$2^{-b}\Omega_{\mathbf{U}}[\{0\sigma\}] \leq \Omega_{\mathbf{U}}[\{\sigma\}] \leq 2^{b}\Omega_{\mathbf{U}}[\{0\sigma\}].$$

As in the proof of Theorem 3.5.10, let $(\sigma_i)_{i\in\mathbb{N}}$ be the sequence of distinct strings such that

$$2^{-i-c-1} < \Omega_{\mathbf{U}}[\{\sigma_i\}] \leq 2^{-i+c}$$

for an appropriate constant c and such that all σ_i start with a 0. Let $Y = \{e+1\colon e \in A'\}$ be a set which codifies A' with all strings starting with 1. So no σ_i belongs to Y.

The construction of X is similar to the one in the proof of Theorem 3.5.10, but now it is relative to A': at stage $s+1$ we may or may not put the string σ_s depending whether the block of bits of $\Omega_{\mathbf{U}}[Y \cup X_s]$ from $s-c$ to $s+c+1$ are all zeroes. Observe that $\Omega_{\mathbf{U}}[Y \cup X_s]$ is A'-computable because

$$\Omega_{\mathbf{U}}[Y \cup X_s] = \Omega_{\mathbf{U}}[Y] + \Omega_{\mathbf{U}}[X_s] \tag{3.13}$$

is a left-c.e. real relative to A. Define $Z = X \cup Y$ and note that by construction $Z \leq_T A'$ and since A' is codified inside Z using strings starting with 1, $Z \geq_T A'$. □

Corollary 3.5.12. *If* $n \geq 1$ *there is a set* $X \equiv_T \emptyset^{(n)}$ *such that* $\Omega_{\mathbf{U}}[X]$ *is not random.*

Since not random implies not n-random and $X \equiv_T \emptyset^{(n)}$ is Σ^0_n-hard, the second part of the Conjecture 3.1.1 is also false.

The next theorem implies that if $n \geq 2$ and X is *any* Σ^0_n or Π^0_n then $\Omega_{\mathbf{U}}[X]$ is not n-random, showing again that the second part of the Conjecture is false.

Theorem 3.5.13. *Let* $A \subset \mathbb{N}$ *be such that* $\emptyset' \leq_T A$*. If* $\mathbf{U}$ *is any universal machine and* $X \subseteq 2^{<\omega}$ *is* $\Sigma^{0,A}_1$ *or* $\Pi^{0,A}_1$ *then* $\Omega_{\mathbf{U}}[X]$ *is not random in* A*.*

Proof. The case X is finite is trivial since then $\Omega_{\mathbf{U}}[X]$ is Δ^0_2 hence computable in $\emptyset'$.

Case X *is infinite* $\Sigma^{0,A}_1$. Fix $m \in \mathbb{N}$. With oracle $\emptyset'$, we can (uniformly in m) find a finite subset $D \subset 2^{<\omega}$ such that $\Omega_{\mathbf{U}}[D] > \Omega_{\mathbf{U}} - 2^{-m-1}$ and compute (relatively to A) a rational $\varepsilon > 0$ such that $\varepsilon < \min\{\Omega_{\mathbf{U}}[\{\sigma\}]\colon \sigma \in D\}$. Then

$$\begin{aligned}
\Omega_{\mathbf{U}}[D] &\leq \sum_{\Omega_{\mathbf{U}}[\{\sigma\}]>\varepsilon} \Omega_{\mathbf{U}}[\{\sigma\}] \\
&= \Omega_{\mathbf{U}} - \sum_{\Omega_{\mathbf{U}}[\{\sigma\}]\leq\varepsilon} \Omega_{\mathbf{U}}[\{\sigma\}].
\end{aligned}$$

and so

$$\sum_{\Omega_{\mathbf{U}}[\{\sigma\}]\leq\varepsilon} \Omega_{\mathbf{U}}[\{\sigma\}] \;<\; 2^{-m-1}. \tag{3.14}$$

Let $(\sigma_s)_{s\in\mathbb{N}}$ be an injective A-computable enumeration of X and set $X_s = \{\sigma_t \colon t < s\}$. We build an Martin-Löf test relative to A, $(\mathcal{T}_m)_{m\in\mathbb{N}}$ for $\Omega_{\mathbf{U}}[X]$. The idea is to define a $\Sigma_1^{0,A}$ class $\mathcal{T}_m$ by laying down successive intervals covering $\Omega_{\mathbf{U}}[X_s]$. Set $\mathcal{T}_m = \bigcup_{s\in\mathbb{N}} \mathcal{I}_{m,s}$ where $\mathcal{I}_{m,s} = (\Omega_{\mathbf{U}}[X_s], \Omega_{\mathbf{U}}[X_s] + \delta)$ and choose $\delta \in \mathbb{Q}^+$ such that $\delta < \varepsilon$ and $\delta(1/\varepsilon + 1) < 2^{-m-1}$. For s big enough, $\Omega_{\mathbf{U}}[X_s] < \Omega_{\mathbf{U}}[X] < \Omega_{\mathbf{U}}[X_s] + \delta$, so that $\Omega_{\mathbf{U}}[X] \in \mathcal{I}_{m,s}$. Thus, $\Omega_{\mathbf{U}}[X] \in \mathcal{T}_m$.

Since $\Omega_{\mathbf{U}}[X_{s+1}] = \Omega_{\mathbf{U}}[X_s] + \Omega_{\mathbf{U}}[\{\sigma_s\}]$, we have $\Omega_{\mathbf{U}}[\{\sigma_s\}] \geq \delta$ if and only if $\mathcal{I}_{m,s}$ and $\mathcal{I}_{m,s+1}$ are disjoint. Assume there are k strings of X with probability greater than or equal to ε, i.e.

$$k = \|\{\sigma \colon \sigma \in X \wedge \Omega_{\mathbf{U}}[\{\sigma\}] > \varepsilon\}\|.$$

Then, the set $\mathcal{T}_m$ consists of at least $k+1$ disjoint intervals and

$$\begin{aligned} \mu\,(\mathcal{T}_m) &\leq \delta(k+1) + \sum_{\sigma\in X\wedge\Omega_{\mathbf{U}}[\sigma]\leq\varepsilon} \Omega_{\mathbf{U}}[\sigma] \\ &< \delta(k+1) + 2^{-m-1} \\ &< 2^{-m-1} + 2^{-m-1} = 2^{-m}. \end{aligned}$$

The above second inequality follows from (3.14) and the last inequality follows from the fact that

$$\begin{aligned} k &\leq \|\{\sigma \colon \Omega_{\mathbf{U}}[\{\sigma\}] > \varepsilon\}\| \\ &\leq \Omega_{\mathbf{U}}/\varepsilon \leq 1/\varepsilon \end{aligned}$$

and the definition of δ.

Thus, we have constructed a Martin-Löf test relative to A $(\mathcal{T}_m)_{m\in\mathbb{N}}$ such that $\Omega_{\mathbf{U}}[X] \in \bigcap_{m\in\mathbb{N}} \mathcal{T}_m$, proving that $\Omega_{\mathbf{U}}[X]$ is not random relative to A.

Case X is infinite $\Pi_1^{0,A}$. Since $\Omega_{\mathbf{U}}[X] = \Omega_{\mathbf{U}} - \Omega_{\mathbf{U}}[2^{<\omega} \setminus X]$, use the above case and the fact that $\Omega_{\mathbf{U}}$ is A-computable. □

3.6 Positive results about the Conjecture

In this section we give positive instances of Conjecture 3.1.1.

Theorem 3.6.1. *Let* $\mathbf{U}$ *be universal. If* $n \geq 2$ *and* X *is* Σ_n^0*-complete then* $\Omega_{\mathbf{U}}[X]$ *is random.*

Proof. We will build a Σ_1^0 relative to $\emptyset^{(n-1)}$ set $S \subseteq 2^{<\omega}$; by the Recursion Theorem we may assume that we know the computable injection $f \colon 2^{<\omega} \to 2^{<\omega}$ by which $S \leq_1 X$. Take a $\emptyset^{(n-1)}$-computable enumeration of X; we may assume that it agrees with the stages of our $\emptyset^{(n-1)}$-enumeration of S. Let $x_s = \Omega_{\mathbf{U}}[X_{s-1}]$.

We will also define a prefix-free machine $\mathbf{M}$; again by the Recursion Theorem we can assume that we know a constant c such that

$$(\forall\sigma)\, K_{\mathbf{U}}(\sigma) \leq K_{\mathbf{M}}(\sigma) + c.$$

Define $\mathbf{M}(\rho)$ as follows: if $\mathbf{U}(\rho)\downarrow$, then let $\mathbf{M}(\rho) = f(\rho)$. Clearly $\mathbf{M}$ is prefix-free.

Next we must define S. Put ρ into S at stage s if:

1. $\mathbf{U}_s(\rho)\downarrow= q$ (finite strings can be treated as dyadic rationals).

2. $|x_s - q| \leq 2^{-|\rho|-c-1}$ (x_s is $\emptyset^{(n-1)}$-computable).

Now assume, by way of contradiction, that $K_{\mathbf{U}}(\Omega_{\mathbf{U}}[X]\upharpoonright m) < m-c-1$ for some $m \in \mathbb{N}$. So, there is a $\rho \in 2^{<\omega}$ such that $\mathbf{U}(\rho) = \Omega_{\mathbf{U}}[X]\upharpoonright m$ and $|\rho| < m-c-1$. Let $q = \Omega_{\mathbf{U}}[X]\upharpoonright m$. For all sufficiently large s, $|x_s - q| \leq 2^{-m} < 2^{-|\rho|-c-1}$. Let s be the least stage such that both 1 and 2 are satisfied. At this stage, ρ goes into S and $f(\rho)$ goes into X. Therefore, $x_{s+1} - x_s = 2^{-K_{\mathbf{U}}(f(\rho))}$. But $\mathbf{M}(\rho) = f(\rho)$ because $\mathbf{U}(\rho)\downarrow$, so $K_{\mathbf{U}}(f(\rho)) \leq K_{\mathbf{M}}(f(\rho)) + c \leq |\rho| + c$. Thus, $x_{s+1} - x_s \geq 2^{-|\rho|-c}$. Together with 2 this implies that

$$\begin{aligned}\Omega_{\mathbf{U}}[X] - \Omega_{\mathbf{U}}[X]\upharpoonright m &\geq x_{s+1} - q\\ &\geq 2^{-|\rho|-c} - 2^{-|\rho|-c-1}\\ &= 2^{-|\rho|-c-1} > 2^{-m}.\end{aligned}$$

This is a contradiction, so $(\forall m)\, K_{\mathbf{U}}(\Omega_{\mathbf{U}}[X]\upharpoonright m) \geq m-c-1$. Therefore, $\Omega_{\mathbf{U}}[X]$ is random.

Here is the full argument for the paradoxical fact that we can use f, the reduction function from S to X, though we are actually *constructing* S. We also justify the assumption that the enumeration of X coincides with the enumeration of S.

Suppose $(\tilde{X})_{s\in\mathbb{N}}$ is the given $\emptyset^{(n-1)}$-computable enumeration of X and let g be a computable injection such that $x \in \emptyset^{(n)}$ iff $g(x) \in X$. Let $p : \mathbb{N}\times\mathbb{N} \to \mathbb{N}$ be a computable function such that for all $e \in \mathbb{N}$, $x \in \mathbb{N}$ and $Z \subseteq \mathbb{N}$, $\varphi_e^Z(x) = J^Z(p(e,x))$ (recall the definition of *reduction function* from 1.3).

We define, for each i, a $\emptyset^{(n-1)}$-enumeration for Y_i and S_i. Start with $Y_{i,0} = \emptyset$ and $S_{i,0} = \emptyset$.

Stage $2s+1$: if some ρ enters into $\tilde{X}_s$ and there is no σ such that $g(p(i,\sigma)) = \rho$ (this can be decided using $\emptyset^{(n-1)}$) then define

$$Y_{i,s+1} = Y_{i,s} \cup \{\rho\}. \tag{3.15}$$

stage $2s+2$: if there is a ρ of length at most s such that

1. $\mathbf{U}_s(\rho)\downarrow= q$

2. $|x_s - q| \leq 2^{-|\rho|-c-1}$ (we redefine $x_s = \Omega_{\mathbf{U}}[Y_{s-1}]$ instead of $x_s = \Omega_{\mathbf{U}}[\tilde{X}_{s-1}]$)

then define

$$S_{i,s+1} = S_{i,s} \cup \{\rho\} \qquad \text{and} \qquad Y_{i,s+1} = Y_{i,s} \cup \{g(p(i,\rho))\}. \tag{3.16}$$

Let $Y_i = \bigcup Y_{i,s}$ and $S_i = \bigcup S_{i,s}$ and let $\psi : 2^{<\omega} \times 2^{<\omega} \to 2^{<\omega}$ be a $\emptyset^{(n-1)}$-computable function such that on input i, x it does the following: if ever x enters into S_i then $\psi(i,x)$ halts, else it gets undefined.

Observe that for a fixed i, $S_i =$ dom $\lambda y.\psi(i,y)$. We only put elements into Y in case (3.16) or (3.15). Hence $\rho \in Y_i$ iff either $\rho \in g(p(i,S_i))$ or both $\rho \in X$ and $\rho \notin \mathrm{ran}(\lambda y.g(p(i,y))$.

Additionally, if σ enters into S_i at stage s then $g(p(i,\sigma))$ enters into Y_i at the same stage s.

Now, by the *s-m-n* Theorem, there is a computable h such that for all x and i, $\psi(i,x) = \varphi_{h(i)}(x)$ and by the Recursion Theorem there is an e such that $\psi(e,x) = \varphi_{h(e)}(x) = \varphi_e(x)$ for all x.

Thus

$$x \in \text{dom } \varphi_e^{\emptyset^{(n-1)}} = S_e \quad \Leftrightarrow \quad p(e,x) \in \emptyset^{(n)} \quad \Leftrightarrow \quad g(p(e,x)) \in X. \tag{3.17}$$

Hence, $Y_e = X$. Indeed, if $\gamma \in Y_e$ then either $\gamma \in X$ or $\gamma \in p(i, g(S_e))$. In the last case, there is $\rho \in S_e$ such that $\gamma = g(p(i,\rho))$. By equivalence 3.17, $\gamma \in X$. If $\gamma \in X$ and $\rho \notin \mathrm{ran}(\lambda y.g(p(i,y))$ then $\gamma \in Y_e$; if $\gamma \in X$ and there is ρ such that $\gamma = g(p(i,\rho))$ then this ρ is unique and $\rho \in S_e$, so $\gamma \in g(p(e,S_e))$ and hence $\gamma \in Y_e$

Both g and $\lambda y.p(e,y)$ are total computable injections (recall the definition of *reduction function* from section 1.3), injectivity follows from the *s-m-n* Theorem), hence $\lambda y.p(e,g(y))$ also is and then $S_e \leq_1 X$ via $\lambda y.g(p(e,y))$. Therefore, $f = \lambda y.p(e,g(y))$ is the function we are using in the proof, and S_e is the set defined there. Also $(Y_{e,s})_{s\in\mathbb{N}}$ is the assumed enumeration of X. □

The case of Π^0_1-complete sets X is obtained with a similar argument.

Theorem 3.6.2. *Let* $\mathbf{U}$ *be universal. If* $n \geq 2$ *and* X *is* Π^0_n*-complete then* $\Omega_{\mathbf{U}}[X]$ *is random.*

Proof. We follow the strategy used for the Σ^0_n-complete case of Theorem 3.6.1. Again define $x_s = \Omega_{\mathbf{U}}[X_{s-1}]$ and define a $\emptyset^{(n-1)}$-computable enumeration of $2^{<\omega} \setminus S$ based on the given $\emptyset^{(n-1)}$-computable enumeration of $2^{<\omega} \setminus X$. Conditions 1 and 2 from the proof of Theorem 3.6.1 for putting some σ into $2^{<\omega} \setminus S$ are the same. All the argument goes through, but since this time $x_s \to \Omega_{\mathbf{U}}[X]$ from above, we have $x_s - x_{s+1} \geq 2^{-|\sigma|-c}$ and then

$$\begin{array}{rcl} \Omega_{\mathbf{U}}[X] - \Omega_{\mathbf{U}}[X] \upharpoonright m & \leq & x_{s+1} - q \\ & \leq & x_s - q + x_{s+1} - x_s \\ & < & 2^{-|\sigma|-c-1} - 2^{|\sigma|-c} \\ & = & -2^{-|\sigma|-c-1} < 0. \end{array}$$

This is a contradiction because $\Omega_{\mathbf{U}}[X] \geq \Omega_{\mathbf{U}}[X] \upharpoonright m$. □

Observe that all the proofs of Theorems 3.6.1 and 3.6.2 can be relativized to any oracle A. In this case, we obtain the following stronger result that shows that although the conjecture was not completely true, it does have a lot of positive instances.

Corollary 3.6.3. *Let* $\mathbf{U}$ *be universal. If* $X \subseteq 2^{<\omega}$ *is* $\Sigma_1^{0,A}$*-complete or* $\Pi_1^{0,A}$*-complete for some* $A \subset \mathbb{N}$ *such that* $\emptyset' \leq_T A$ *then* $\Omega_{\mathbf{U}}[X]$ *is random.*

Proof. For X $\Sigma_1^{0,A}$-complete, follow the proof of Theorem 3.6.1. Instead of constructing $S \in \Sigma_1^{0,\emptyset^{(n-1)}}$, we construct $S \in \Sigma_1^{0,A}$. We use the computable injection $f \colon 2^{<\omega} \to 2^{<\omega}$ by which $S \leq_1 A'$. Since $A \geq_T \emptyset'$, we can still compute x_s and decide whether condition 2 holds.

For X $\Pi_1^{0,A}$-complete, follow the proof of Theorem 3.6.2. □

3.7 The set $\{\Omega_{\mathbf{U}}[X] \colon X \subseteq 2^{<\omega}\}$

The following Lemma states that for a sequence of real numbers which goes to zero but not very fast, it is always possible to find a subserie of the sequence which tends to any real small enough.

Lemma 3.7.1. *Let* $(a_i)_{i\in\mathbb{N}}$ *be a sequence of strictly positive real numbers satisfying*

1. $\lim_{i\to+\infty} a_i = 0$;
2. $a_i \leq \sum_{j>i} a_j$ *for all* i.

Let $\alpha = \sum_{i\in\mathbb{N}} a_i$ *(which may be* $+\infty$*). Then* $\left\{\sum_{i\in I} a_i \colon I \subseteq \mathbb{N}\right\} = [0,\alpha]$*. Furthermore, for every* $r \in [0,\alpha]$ *there exists* $I(r) \subseteq \mathbb{N}$ *such that* $\sum_{i\in I(r)} a_i = r$ *and which is computable (non-uniformly) from* r *and* $(a_i)_{i\in\mathbb{N}}$*.*

Proof. Take $r \in [0,\alpha]$. We define a monotone increasing sequence $(I_t(r))_{t\in\mathbb{N}}$ of finite subsets of $\mathbb{N}$ by the following induction:

$$I_0(r) = \emptyset \quad , \quad I_{t+1}(r) = \begin{cases} I_t(r) \cup \{t\} & \text{if } a_t + \sum_{i\in I_t(r)} a_i \leq r; \\ I_t(r) & \text{otherwise.} \end{cases}$$

Let $I(r) = \bigcup_{t\in\mathbb{N}} I_t(r)$. Since inequality $\sum_{i\in I_t(r)} a_i \leq r$ is true for all t, we get $\sum_{i\in I(r)} a_i \leq r$. We show that $r = \sum_{i\in I(r)} a_i$.

Case $r = \alpha$. Then $I(r) = \mathbb{N}$ and the equality is trivial.

Case $r < \alpha$ *and there are infinitely many* t*s such that* $I_{t+1}(r) = I_t(r)$. For such ts we have

$$\sum_{i\in I_t(r)} a_i \leq r < a_t + \sum_{i\in I_t(r)} a_i.$$

Taking limits over such ts and using condition 1 on the sequence $(a_i)_{i\in\mathbb{N}}$, we get equality $\sum_{i\in I(r)} a_i = r$.

Case $r < \alpha$ and there are finitely many ts such that $I_{t+1}(r) = I_t(r)$. We show that this case does not occur. Since $r < \alpha$ we have $I(r) \neq \mathbb{N}$ so that there is at least one t such that $I_{t+1}(r) = I_t(r)$. Let u be the largest such t. Then, $\sum_{i \in I_u(r)} a_i \leq r < a_u + \sum_{i \in I_u(r)} a_i$ and, for all $v > u$, $I_{v+1} = I_v \cup \{v\}$. Therefore, $I(r) = I_u(r) \cup \{i \colon i > u\}$. Since condition 2 of the hypothesis ensures $a_u \leq \sum_{i>u} a_i$, we get

$$r < \sum_{i>u} a_i + \sum_{i \in I_u(r)} a_i = \sum_{i \in I(r)} a_i,$$

which contradicts inequality $\sum_{i \in I(r)} a_i \leq r$.

The last assertion of the Lemma about the relative computability of $I(r)$ is trivial if $I(r)$ is finite. Since the a_ts are strictly positive, if $I(r)$ is infinite then $r \neq a_t + \sum_{i \in I_t(r)} a_i$ for all t. Thus, enumerating the digits of r and $a_t + \sum_{i \in I_t(r)} a_i$, we get at some finite time either $r < a_t + \sum_{i \in I_t(r)} a_i$ or $r > a_t + \sum_{i \in I_t(r)} a_i$, which proves that the test in the definition of $I_{t+1}(r)$ can be done computably in r and $(a_i)_{i \in \mathbb{N}}$. □

Point 2 of the following theorem gives an alternative proof of Theorem 3.5.10 above.

Theorem 3.7.2. *Let $\mathbf{U}$ be universal.*

1. *The set $\{\Omega_{\mathbf{U}}[X] \colon X \subseteq 2^{<\omega}\}$ is the union of finitely many pairwise disjoint closed intervals with positive lengths, i.e.*

$$\{\Omega_{\mathbf{U}}[X] \colon X \subseteq 2^{<\omega}\} = [a_1, b_1] \cup [a_2, b_2] \cup \cdots \cup [a_n, b_n]$$

   where $0 = a_1 < b_1 < \cdots < a_n < b_n = \Omega_{\mathbf{U}}$.

2. *Every real $s \in \{\Omega_{\mathbf{U}}[X] \colon X \subseteq 2^{<\omega}\}$ is of the form $\Omega_{\mathbf{U}}[Y]$ for some Y which is computable in $s \oplus \emptyset'$. In particular, there exists some Δ^0_2 set $X \subseteq 2^{<\omega}$ such that $\Omega_{\mathbf{U}}[X]$ is rational, hence not random.*

Proof. For item 1, let us first see that we can get $\alpha > 0$ such that

$$\{\Omega_{\mathbf{U}}[X] \colon X \subseteq 2^{<\omega}\} \supseteq [0, \alpha].$$

Let $d, d' \in \mathbb{N}$ be the constants of Lemma 3.5.9 and let k be such that $2^{2d+1}(i+1) \leq 2^i/(d'i^2)$ for $i \geq k$. Using this inequality and Lemma 3.5.9, one can inductively define a sequence of *pairwise disjoint* sets of strings $(S_i)_{i \geq k}$ such that $\|S_i\| = 2^{2d+1}$ and $2^{-i-d-1} < \Omega_{\mathbf{U}}[\{\sigma\}] \leq 2^{-i+d}$ for every $\sigma \in S_i$. Notice that, as in Theorem 3.5.10, the sequence $(S_i)_{i \geq k}$ is computable in $\emptyset'$.

We define an enumeration ψ of $S = \bigcup_{i \in \mathbb{N}} S_i$: for $i, m \in \mathbb{N}$ and $m < 2^{2d+1}$, let $\psi(2^{2d+1}i + m)$ be the m-th element of S_{k+i}.

Set $a_i = \Omega_{\mathbf{U}}[\{\psi(i)\}]$, it is clearly positive and $\lim_{i \to +\infty} a_i = 0$. Observe that for any $m \in [0, 2^{2d+1})$, $2^{-(k+j)-d-1} < a_{2^{2d+1}j+m} \leq 2^{-(k+j)+d}$ and it is computable in $\emptyset'$. Then, for any such m we have

$$\begin{aligned}
\sum_{j>2^{2d+1}q+m} a_j &\geq \sum_{j>q} \sum_{s<2^{2d+1}} a_{2^{2d+1}j+s} \\
&> \sum_{j>q} 2^{2d+1} 2^{-(k+j)-d-1} \\
&= 2^{-(k+q)+d} \geq a_{2^{2d+1}q+m}.
\end{aligned}$$

Thus, the conditions of Lemma 3.7.1 are satisfied: $\{\Omega_{\mathbf{U}}[Y] \colon Y \subseteq S\} = [0, \alpha]$ where $\sum_{i \in \mathbb{N}} a_i = \alpha > 0$.

Now,

$$\begin{aligned}
\{\Omega_{\mathbf{U}}[X] \colon X \subseteq 2^{<\omega}\} &= \{\Omega_{\mathbf{U}}[Y] + \Omega_{\mathbf{U}}[Z] \colon Y \subseteq S,\ Z \cap S = \emptyset\} \\
&= [0, \alpha] + \{\Omega_{\mathbf{U}}[Z] \colon Z \cap S = \emptyset\} \\
&= \bigcup_{r \in \mathcal{R}} [r, r + \alpha]
\end{aligned}$$

where $\mathcal{R} = \{\Omega_{\mathbf{U}}[Z] \colon Z \cap S = \emptyset\}$ and $0 \in \mathcal{R}$.

Let $\mathcal{R}_i = \mathcal{R} \cap [i\alpha, (i+1)\alpha)$. Observe that if $r, r' \in \mathcal{R}_i$ then $[r, r+\alpha]$ and $[r', r'+\alpha]$ have non-empty intersection. Hence the union $\bigcup_{r \in \mathcal{R}_i} [r, r + \alpha]$ is an interval $\mathcal{J}_i$ (a priori not necessarily closed). Since $\mathcal{R}_i = \emptyset$ for $i\alpha > 1$, we see that $\mathcal{R} = \mathcal{R}_1 \cup \cdots \cup \mathcal{R}_k$ where $k \leq \lceil 1/\alpha \rceil$. Thus,

$$\{\Omega_{\mathbf{U}}[X] \colon X \subseteq 2^{<\omega}\} = \mathcal{J}_1 \cup \cdots \cup \mathcal{J}_k.$$

Grouping successive intervals $\mathcal{J}_i$s having non-empty intersection, we get the representation

$$\{\Omega_{\mathbf{U}}[X] \colon X \subseteq 2^{<\omega}\} = \mathcal{I}_1 \cup \cdots \cup \mathcal{I}_n$$

where the $\mathcal{I}_i$s are pairwise disjoint intervals in $[0, 1]$.

Since the map $X \mapsto \Omega_{\mathbf{U}}[X]$ is continuous from the compact space $\mathcal{P}(2^{<\omega})$ (with the Cantor topology) to $[0, 1]$, its range $\{\Omega_{\mathbf{U}}[X] \colon X \subseteq 2^{<\omega}\}$ is compact. In particular, the intervals $\mathcal{I}_i$s may be taken closed. This proves item 1 of the Theorem.

For item 2, first, observe that if $I \subseteq \mathbb{N}$ is computable in $\emptyset'$ then so is $\{\psi(n) \colon n \in I\}$. Given $\sigma \in 2^{<\omega}$, using $\emptyset'$, one can check whether $2^{-j} < \Omega_{\mathbf{U}}[\{\sigma\}]$. Hence one can compute i and m such that σ is the m-th element of S_{k+i}, i.e. such that $\sigma = \psi(2^{2d+1}i + m)$. Then $\sigma \in \{\psi(n) \colon n \in I\}$ if and only if $2^{2d+1}i + m \in I$

Case $s \in [0, \alpha]$. Lemma 3.7.1 ensures that there is a set $I(s) \subseteq \mathbb{N}$, computable from $s \oplus \emptyset'$, such that $\sum_{i \in I(s)} a_i = s$. Let $X = \{\psi(n) \colon n \in I(s)\}$. Then X is computable from $s \oplus \emptyset'$ and $\Omega_{\mathbf{U}}[X] = s$.

Case $s \in [r, r+\alpha)$ for some $r \in \mathcal{R}$. Let $s = \Omega_{\mathbf{U}}[Z] + \beta$ where $r = \Omega_{\mathbf{U}}[Z]$ and $Z \cap S = \emptyset$ and $\beta < \alpha$. Let Z' be a finite subset of Z such that $\Omega_{\mathbf{U}}[Z \setminus Z'] < \alpha - \beta$. Then the real $\Omega_{\mathbf{U}}[Z']$ is computable in $\emptyset'$ and $\Omega_{\mathbf{U}}[Z \setminus Z'] + \beta = s - \Omega_{\mathbf{U}}[Z']$ is computable in $s \oplus \emptyset'$. Since $\Omega_{\mathbf{U}}[Z \setminus Z'] + \beta < \alpha$, Lemma 3.7.1 yields $X \subseteq S$ which is computable in $s \oplus \emptyset'$ such that $\Omega_{\mathbf{U}}[Z \setminus Z'] + \beta = \Omega_{\mathbf{U}}[X]$. Since Z' is finite, we see that $X \cup Z'$ is computable in $s \oplus \emptyset'$. Finally, $s = \Omega_{\mathbf{U}}[X \cup Z']$.

Case $s \in [a_j, b_j)$ with $1 \leq j \leq n$. Observe that $\bigcup_{r \in \mathcal{R}_i}[r, r+\alpha)$ is equal to $\mathcal{J}_i$ with the right endpoint removed. Suppose $\mathcal{I}_j = \mathcal{J}_i \cup \cdots \cup \mathcal{J}_{i+m}$. Then

$$[a_j, b_j) = \bigcup_{i \leq p \leq i+m} \; \bigcup_{r \in \mathcal{R}_p} [r, r+\alpha).$$

Thus, $s \in [r, r+\alpha)$ for some $r \in \mathcal{R}$ and the previous case applies.

Case $s = b_j$ with $1 \leq j \leq n$. Let $b_j = \Omega_{\mathbf{U}}[X]$. If $\sigma \notin X$ then $\Omega_{\mathbf{U}}[X \cup \{\sigma\}] > b_j$ hence $\Omega_{\mathbf{U}}[X \cup \{\sigma\}] \geq a_{j+1}$. In particular, $\Omega_{\mathbf{U}}[\{\sigma\}] \geq a_{j+1} - b_j$, which proves that the complement of X contains at most $\lceil \frac{1}{a_{j+1}-b_j} \rceil$ elements. Thus, X is cofinite, hence computable. □

In relation with Theorem 3.7.2, we consider the following question: how much disconnected is $\{\Omega_{\mathbf{U}}[X] : X \subseteq 2^{<\omega}\}$?

Proposition 3.7.3. *For each $n \geq 1$ there exists a universal machine $\mathbf{U}$ such that the set $\{\Omega_{\mathbf{U}}[X] \colon X \subseteq 2^{<\omega}\}$ is not the union of less than n intervals.*

Proof. Let $\mathbf{V}$ be universal and define the universal machine $\mathbf{U}_0$ as follows:

$$\begin{aligned} \mathbf{U}_0(0) &= \lambda; \\ \mathbf{U}_0(10\rho) &= \mathbf{V}(\rho). \end{aligned}$$

Then $\Omega_{\mathbf{U}_0}[\{\lambda\}] = 1/2 + \Omega_{\mathbf{V}}[\{\lambda\}]/4$ and for any X such that $\lambda \notin X$ we have $\Omega_{\mathbf{U}}[X] = \Omega_{\mathbf{V}}[X]/4$. Hence,

$$\begin{aligned} \{\Omega_{\mathbf{U}_0}[X] \colon X \subseteq 2^{<\omega}\} &= \{\Omega_{\mathbf{U}_0}[X] \colon \lambda \notin X\} \cup \{\Omega_{\mathbf{U}_0}[X] \colon \lambda \in X\} \\ &= \{\Omega_{\mathbf{V}}[X]/4 \colon \lambda \notin X\} \;\cup\; \{1/2 + \Omega_{\mathbf{V}}[X]/4 \colon \lambda \in X\} \\ &\subseteq 002^{\omega} \cup 102^{\omega}. \end{aligned}$$

So, $\{\Omega_{\mathbf{U}_0}[X] \colon X \subseteq 2^{<\omega}\}$ has at least two disjoint components because neither $\{\Omega_{\mathbf{V}}[X]/4 \colon \lambda \notin X\}$ nor $\{1/2 + \Omega_{\mathbf{V}}[X]/4 : \lambda \in X\}$ are empty.

Using the same idea, we can define an optimal machine $\mathbf{U}_n$ such that

$$\{\Omega_{\mathbf{U}_n}[X] \colon X \subseteq 2^{<\omega}\} \subseteq \bigcup_{|\sigma|=n+1} \sigma 02^{\omega}$$

and for each σ of length $n+1$,

$$\{\Omega_{\mathbf{U}_n}[X] \colon X \subseteq 2^{<\omega}\} \cap \sigma 02^{\omega} \neq \emptyset.$$

Hence for any n, the optimal machine $\mathbf{U}_n$ will have to be such that $\{\Omega_{\mathbf{U}_n}[X] \colon X \subseteq 2^{<\omega}\}$ contains at least 2^{n+1} disjoint intervals.

Here are the details of the inductive construction of $\mathbf{U}_n$. For any $n > 0$, define the optimal machine $\mathbf{U}_n$ in the following way:

$$\begin{aligned}
\mathbf{U}_n(0^i1) &= 0^{i-1} \quad \text{for } 1 \leq i \leq n; \\
\mathbf{U}_n(0^n0) &= 0^n; \\
\mathbf{U}_n(1\rho) &= \mathbf{U}_{n-1}(\rho).
\end{aligned}$$

For any $X \subseteq 2^{<\omega}$ and $i \in \mathbb{N}$, let us define $X_i = 1$ if $0^i \in X$ and $X_i = 0$ otherwise. We will prove by induction on n that

$$\emptyset \neq \{\Omega_{\mathbf{U}_n}[X] \colon X_0 \dots X_n = \sigma\} \subseteq \sigma 0 2^\omega.$$

It clearly holds for $n = 0$. Fix any σ of length n. By inductive hypothesis we know that

$$\mathcal{A} = \{\Omega_{\mathbf{U}_{n-1}}[X] \colon X_0 \dots X_{n-1} = \sigma\} \subseteq \sigma 0 2^\omega.$$

Thus, for $j \in \{0, 1\}$

$$\begin{aligned}
\{\Omega_{\mathbf{U}_n}[X] \colon X_0 \dots X_{n-1} X_n = \sigma j\} &= \mathcal{A}/2 + \left(\sum_{i=0}^{n-1} \sigma(i) 2^{-(i+2)} \right) + j 2^{-(n+1)} \\
&\subseteq 0\sigma 0 2^\omega + .0\sigma + .0^n j \\
&= \sigma 0 0 2^\omega + .0^n j = \sigma j 0 2^\omega.
\end{aligned}$$

For convenience, in the last two lines, we have used addition of sets of reals using binary notation. The inequality of the second line follows from the inductive hypothesis. Since

$$\{\Omega_{\mathbf{U}_n}[X] \colon X \subseteq 2^{<\omega}\} = \bigcup_{|\sigma|=n+1} \{\Omega_{\mathbf{U}_n}[X] \colon X_0 \dots X_n = \sigma\}$$

we conclude

$$\{\Omega_{U_n}[X] \colon X \subseteq 2^{<\omega}\} \subseteq \bigcup_{|\sigma|=n+1} \sigma 0 2^\omega$$

and also that each $\{\Omega_{\mathbf{U}_n}[X] \colon X_0 \dots X_n = \sigma\}$ is non-empty, as we wanted. □

From Theorem 3.7.2 we know that for every universal machine $\mathbf{U}$ and for each sufficiently small but positive computable real number R there is a $X \in \Delta^0_2$ such that $\Omega_{\mathbf{U}}[X] = R$. If one can choose the universal machine freely then one can even get that the corresponding X is a co-c.e. set. To prove this we first need the following lemma:

Lemma 3.7.4. *There is a universal machine $\mathbf{U}$ such that for almost all n there is an $x \in \mathbb{N}$ such that $K_x(x) = K(x) = n$, where $K = K_{\mathbf{U}}$ and x is the largest number with $K(x) \leq n$.*

Proof. Let $\mathbf{V}$ be any universal machine and $\mathbf{V}_s$ be a computable approximation such that if $\mathbf{V}_s(\rho)\downarrow=\sigma$ then $\mathrm{str}^{-1}(\sigma)\leq 2^{s+1}$ and $|\rho|\leq s$. Observe that if $\sigma=x$ is a natural number then $\mathrm{str}^{-1}(x)\geq 2x$ and so if $\mathbf{V}_s(\rho)\downarrow=x$ then $x\leq 2^s$. This last property will be used in the following construction.

We define $\mathbf{U}$ in the following way: for every ρ such that $\mathbf{V}_s(\rho)\downarrow=\sigma$, define

$$\begin{aligned}\mathbf{U}(\rho 00) &= \sigma \quad \text{and} && (3.18)\\ \mathbf{U}(\rho 1^k 0) &= 2^s\cdot 3^k \quad \text{for every } k>0 && (3.19)\end{aligned}$$

by the stage s of $\mathbf{U}$. Observe that $\mathbf{U}$ is universal since $\mathbf{V}$ is.

For short, we write K instead of $K_{\mathbf{U}}$. Let n large enough and let x be the largest number with $K(x)\leq n$. There is a program γ such that $\mathbf{U}(\gamma)=x$ and $|\gamma|=K(x)$. This γ was defined at some stage s of $\mathbf{U}$ following either definition (3.18) or (3.19).

Suppose γ was defined by case (3.18). Then $\gamma=\rho 00$ for some ρ such that $\mathbf{V}(\rho)=x$. By the assumption on $\mathbf{V}$, $x\leq 2^s$. Observe that $\mathbf{U}(\rho 10)=2^s\cdot 3$, $|\rho 10|=|\rho 00|\leq n$ and $2^s\cdot 3>2^s\geq x$. So $2^s\cdot 3>x$ has the property that $K(2^s\cdot 3)\leq n$ and hence x was not the largest, a contradiction.

Hence γ was defined by case (3.19) and $\gamma=\rho 1^k 0$ for some $k>0$ and $x=2^s\cdot 3^k$. If $|\gamma|<n$ then $|\rho 1^{k+1}0|\leq n$ and $K(2^s\cdot 3^{k+1})\leq n$. But this is impossible because $2^s\cdot 3^{k+1}>x$. So $|\gamma|=K(x)=n$.

To show that $K_x(x)=K(x)$, we first need to establish K_s the computable approximation of K from above. First define

$$\tilde{K_s}(\sigma)=\{|\rho|\colon |\rho|\leq s\wedge \mathbf{U}_s(\rho)=\sigma\}$$

as usual, and then define

$$K_s(\sigma)=\min\{2+\tilde{K_s}(\sigma)\}\cup\{|\rho|+k+1\colon |\rho|\leq s\wedge \mathbf{V}_s(\rho)\downarrow\} \tag{3.20}$$

if σ is of the form $2^s\cdot 3^k$ and

$$K_s(\sigma)=\min\{2+\tilde{K_s}(\sigma)\}$$

if σ is not of the form $2^s\cdot 3^k$. It is not difficult to see that for all $\sigma\in 2^{<\omega}$, $K_t(\sigma)\to K(\sigma)$ from above when $t\to\infty$.

Recall from the previous paragraphs that γ, a shortest $\mathbf{U}$-description of $x=2^s\cdot 3^k$ was defined at stage s of $\mathbf{V}$ and that $\gamma=\rho 0^k 1$. Then $|\gamma|=|\rho|+k+1$ for some ρ such that $\mathbf{V}_s(\rho)\downarrow$, and in particular $|\rho|\leq s$. By (3.20) this means that $K_s(x)$ has already reached $|\gamma|=K(x)$. Since $s\leq x=2^s\cdot 3^k$, this proves that

$$K_s(x)=K_x(x)=K(x)=n$$

and this completes the proof. □

Theorem 3.7.5. *There is a universal machine* $\mathbf{U}$ *and an integer* m *such that for every* K*-trivial real* R *between* 0 *and* 2^{-m} *there is a co-c.e. set* X *with* $R=\Omega_{\mathbf{U}}[X]$.

Proof. Let $\mathbf{W}$ be the constructed $\mathbf{U}$ from Theorem 3.7.4. If we let the starting machine $\mathbf{V}$ of the proof of Theorem 3.5.1 be $\mathbf{W}$, we end up with a universal machine $\mathbf{U}$ such that $\Omega_{\mathbf{U}}[\{\sigma\}] = 2^{1-K(\sigma)}$ for all $\sigma \in 2^{<\omega}$ and $\mathbf{U}$ has the properties of Lemma 3.7.4, that is, there is some $m \geq 2$ such that for all $n \geq m$, we have

$$K_{x_n}(x_n) = K(x_n) = n,$$

where $K = K_{\mathbf{U}}$ and

$$x_n = \max\{z \colon K(z) \leq n\}.$$

Let R be a K-trivial real with $0 < R < 2^{-m}$. The aim is now to build a co-c.e. set X such that

$$R = \sum_{r \in R} 2^{-1-r} = \sum_{x \in X} 2^{1-K(x)} = \Omega_{\mathbf{U}}[X]$$

where this goal is by choosing

$$X \subseteq \{x_m, x_{m+1}, \ldots\} \tag{3.21}$$

such that

$$x_n \in X \Leftrightarrow n - 2 \in R. \tag{3.22}$$

The further construction makes use of the fact that there is a c.e. K-trivial set $Q \geq_T R$ [62, Theorem 7.4]. This fact guarantees that R has a computable approximation $R_0, R_1, \ldots$ such that the function $c_R(n)$ that computes the step in which the approximation of the length n prefix stabilizes, defined as

$$c_R(n) = \max\{s \colon s = 1 \vee (\exists m < n)\, R_{s-1}(m) \neq R(m)\}$$

can be computed relative to Q. For all n let

$$y_n = \max\{z \leq c_R(n-2) \colon z = 0 \vee K_z(z) = n\}.$$

Note that the sequence $y_0, y_1, \ldots$ can be computed relative to Q and then

$$K^Q(y_n) \leq K(n) + \mathcal{O}(1). \tag{3.23}$$

Since Q is K-trivial, by Nies' Theorem [62], we know that K^Q differs from K only by a constant, and so

$$n = K(x_n) \leq K^Q(x_n) + \mathcal{O}(1) \tag{3.24}$$

From (3.23) and (3.24) we conclude that for almost all n, $K^Q(y_n) < K^Q(x_n)$. Also, since $K_{y_n}(y_n) = n$ and x_n is the largest such, we have $y_n < x_n$. If $x_n < c_R(n-2)$ then $y_n \geq x_n$ and this is a contradiction. Then $x_n \geq c_R(n-2)$ and therefore

$$R_{x_n}(n-2) = R(n-2). \tag{3.25}$$

Without loss of generality one can assume this property for all $n \geq m$ since a finite modification of the approximation $R_0, R_1, \ldots$ would enforce it. After ensuring this property, one defines the co-c.e. set

$$X = \{x \colon K_x(x) \geq m \wedge R_x(K_x(x) - 2) = 1 \wedge (\forall y > x)(\forall t)\, K_t(y) > K_x(x)]\}.$$

Now (3.21) and the connection (3.22) between X and R is verified:

Lemma 3.7.6. $X \subseteq \{x_m, x_{m+1}, \ldots\}$.

Proof. Consider any $x \in X$ and let $n = K(x)$. Then the condition

$$(\forall y > x)(\forall t)\, K_t(y) > K_x(x)$$

enforces that $K(y) > n$ for all $y > x$ and thus $x = x_n$. Also, by definition of X, $n \geq m$. □

Lemma 3.7.7. *For all n, $x_n \in X$ if and only if $n - 2 \in R$.*

Proof. On the one hand, since $K_{x_n}(x_n) = K(x_n) = n$, one has by (3.25) that $R_{x_n}(n-2) = R(n-2) = 1$, so $n - 2 \in R$.

On the other hand, for $n \geq m$, if $n - 2 \in R$ then there is some x_n such that $K_{x_n}(x_n) = n$ and so $x_n \in X$. Since by choice of R no number below m is in R, the equivalence $x_n \in X \Leftrightarrow n - 2 \in R$ holds for all n. □

Finally we have the following equalities:

$$\begin{aligned}
\Omega_{\mathbf{U}}[X] &= \sum_{x \in X} R_x(K_x(x) - 2)2^{1-K(x)} \\
&= \sum_{x \in X} R(K(x) - 2)2^{1-K(x)} \\
&= \sum_{n=m}^{\infty} R(n-2)2^{1-n} = R.
\end{aligned}$$

This completes the proof. □

We now prove that at least $\Omega_{\mathbf{U}}[X]$ can be made n-random for some Δ^0_{n+1} sets. As a corollary of Theorem 3.7.2, we get the following result which is in contrast with Theorems 3.5.10, 3.5.11 and 3.5.13.

Corollary 3.7.8. *Let $\mathbf{U}$ be any universal machine.*

1. *For any $A \subseteq \mathbb{N}$ there is a set $X \leq_T A'$ such that $\Omega_{\mathbf{U}}[X]$ is random in A.*
2. *For every $n \geq 2$ there is a Δ^0_{n+1} set X such that $\Omega_{\mathbf{U}}[X]$ is n-random. For $n = 1$, there is a computable such X.*

Proof. For item 1, let b_1 be as in Point 1 of Theorem 3.7.2, let Ω^A be the Chaitin real $\Omega_{\mathbf{U}^A}[2^{<\omega}]$ associated to the universal machine $\mathbf{U}^A$ with oracle A and let $k \in \mathbb{N}$ be such that $\Omega^A 2^{-k} < b_1$. Then Ω^A and $\Omega^A 2^{-k}$ are A'-computable and random in A. Theorem 3.7.2 ensures that there exists some set X which is computable in $\Omega^A 2^{-k} \oplus \emptyset' \leq_T A'$ such that $\Omega^A 2^{-k} = \Omega_{\mathbf{U}}[X]$.

For item 2: If $n = 1$, set $X = 2^{<\omega}$ and apply Chaitin's result that Ω is random. If $n \geq 2$, apply Point 1 with $A = \emptyset^{(n-1)}$. □

Also as a consequence of Theorem 3.7.2, we can see that, for any given universal machine $\mathbf{U}$, every small enough left-c.e. random real r is $\Omega_{\mathbf{U}}[X]$ for some $X \subseteq 2^{<\omega}$ which is Δ^0_2. We now show that if we fix r left-c.e. and random, and we fix a Σ^0_1 set X, we can pick an appropriate universal machine $\mathbf{U}$ for which $\Omega_{\mathbf{U}}[X] = r$.

To prove this, we need some well-known facts. In [21] Calude et al. showed that for any left-c.e. real a there exists a prefix-free set $R \subseteq 2^{<\omega}$ such that $a = \mu(R2^\omega)$.

Let us recall the definition of Solovay's domination between left-c.e. reals: let a and b be left-c.e. reals. We say that a *dominates* b, and write $b \leq_S a$ iff there is a constant c and a partial computable function $f\colon \mathbb{Q} \to \mathbb{Q}$ such that for each rational $q < a$, $f(q)$ is defined and $f(q) < b$ and $b - f(q) \leq c(a - q)$.

In [33], Downey et al. proved that if a and b are left-c.e. reals such that $b \leq_S a$, then there is a left-c.e. real d and constant c such that $ca = b + d$.

Using these results, we can prove the following:

Theorem 3.7.9. *Let $X \subseteq 2^{<\omega}$ be Σ^0_1, $X \neq \emptyset$, and let $r \in (0,1)$ be left-c.e. random. There is a universal machine $\mathbf{U}$ such that $r = \Omega_{\mathbf{U}}[X]$.*

Proof. Let $\mathbf{V}$ be any universal machine. By Chaitin's Theorem, $\Omega_{\mathbf{V}}[X]$ is a left-c.e. random real and following [48] we know that $r \equiv_S \Omega_{\mathbf{V}}[X]$ (this just means that $r \leq_S \Omega_{\mathbf{V}}[X]$ and $\Omega_{\mathbf{V}}[X] \leq_S r$). Hence, from the above mentioned result of [33], there is a constant $\tilde{c}$ and a left-c.e. real d such that

$$2^{\tilde{c}} r = \Omega_{\mathbf{V}}[X] + \tilde{d}. \tag{3.26}$$

Let $c \geq \tilde{c}$ large enough such that $2^{-c}\Omega_{\mathbf{V}} < r$ and $2^{-c}\Omega_{\mathbf{V}} < 1 - r$. By equation (3.26) we have $2^c r = \Omega_{\mathbf{V}}[X] + d$ where $d = \tilde{d} + r(2^c - 2^{\tilde{c}})$ is a left-c.e. random real. Hence $r - 2^{-c}\Omega_{\mathbf{V}}[X]$ is a left-c.e. random real in $(0,1)$. From [21] there is a c.e. prefix-free set R such that $r - 2^{-c}\Omega_{\mathbf{V}}[X] = \mu(R2^\omega)$. We define the Kraft-Chaitin set for $\mathbf{U}$ with the axioms

$$W = \{\langle |\rho|, \sigma\rangle \colon \rho \in R\} \cup \{\langle |\rho| + c, \mathbf{V}(\rho)\rangle \colon \mathbf{V}(\rho)\downarrow\},$$

where σ is any string of X. Observe that

$$\begin{aligned} \mathrm{wt}(W) &= \mu(R2^\omega) + 2^{-c}\Omega_{\mathbf{V}} \\ &= r - 2^{-c}\Omega_{\mathbf{V}}[X] + 2^{-c}\Omega_{\mathbf{V}} < 1. \end{aligned}$$

Since for any ρ, if $\mathbf{V}(\rho)\downarrow$ then $\mathbf{V}(\rho) = \mathbf{U}(\gamma)$, for some γ with $|\gamma| = |\rho| + c$, we conclude that $\mathbf{U}$ is universal. By construction, we have

$$\Omega_{\mathbf{U}}[X] = \mu(R2^\omega) + 2^{-c}\Omega_{\mathbf{V}}[X] = r.$$

and this completes the proof. □

3.8 Conjecture for infinite computations

In chapter 4 we will introduce the possibly infinite computations for monotone machines in full detail. In this section we analyze the validity of the conjecture 3.1.1 for this scheme of infinite computations.

In a *monotone machine* the output grows with respect to the prefix ordering in $2^{<\omega}$ as the computational time increases because the output tape only moves to the left. See [51, 67, 14] the complete architecture of a monotone machine and more details of this model of computation. In section 4.2 we work with this model.

Definition 3.8.1. Let $\mathbf{M}$ be a monotone machine. For $Z \in 2^\omega$, $\mathbf{M}_t(Z)$ is the *current* output of $\mathbf{M}$ on input Z by stage t. For $\rho \in 2^{<\omega}$, $\mathbf{M}_t(\rho)$ is the *current* output of $\mathbf{M}$ by stage t if $\mathbf{M}$ has not read beyond the end of ρ; otherwise $\mathbf{M}_t(\rho) \uparrow$.

One can then define the behavior of monotone machines for infinite computations, in the following way:

Definition 3.8.2. Let $\mathbf{M}$ be a monotone machine. The input/output behavior of $\mathbf{M}$ for *possibly infinite computations* is the map $\mathbf{M}^\infty : 2^\omega \to 2^{\leq\omega}$ given by $\mathbf{M}^\infty(Z) = \lim_{t\to\infty} \mathbf{M}_t(Z)$

Notice that we now consider not only finite strings as outputs but also infinite sequences. The output space is $2^{\leq\omega} = 2^{<\omega} \cup 2^\omega$.

Considering possibly non-halting computations, one can associate to a universal monotone Turing machine $\mathbf{U}$ (see [14] for more details) a total map $\mathbf{U}^\infty \colon 2^\omega \to 2^{\leq\omega}$, and for $\mathcal{X} \subseteq 2^{\leq\omega}$, define

$$\Omega^\infty_{\mathbf{U}}[\mathcal{X}] = \mu\left((\mathbf{U}^\infty)^{-1}(\mathcal{X})\right),$$

i.e. $\Omega^\infty_{\mathbf{U}}[\mathcal{X}]$ is the probability that the infinite computation of $\mathbf{U}$ gives an output in $\mathcal{X}$.

An analog of Conjecture 3.1.1 can be stated for infinite computations on universal monotone machines.

Conjecture 3.8.3. *For any proper subset $\mathcal{X}$ of $2^{\leq\omega}$, the probability $\Omega^\infty_{\mathbf{U}}[\mathcal{X}]$ that an arbitrary infinite input to a universal monotone machine performing infinite computations gives an output in $\mathcal{X}$ is random. Moreover, if X is Σ^0_n-hard (for an appropriate notion of hardness) then this probability is n-random.*

Relatively to monotone Turing machines which are universal by adjunction, this conjecture has been proved in [14, 12] for many $\mathcal{X} \subseteq 2^{\leq\omega}$, considering the effective levels of the Borel hierarchy on $2^{\leq\omega}$ with a spectral topology (for which the basic open sets are of the form $\sigma 2^{\leq\omega}$, for $\sigma \in 2^{<\omega}$).

Theorem 3.8.4 ([14, 12])**.** *Let $\mathcal{X} \subseteq 2^{\leq\omega}$ be Σ^0_n(spectral) and hard for the class $\Sigma^0_n(2^\omega)$ with respect to effective Wadge reductions, for any $n \geq 1$. Then, $\Omega^\infty_{\mathbf{U}}[\mathcal{X}]$ is n-random.*

We now prove that the conjecture fails in about the same way as Conjecture 3.1.1. The key fact is that for any computable injective prefix-free codification of strings $f \colon 2^{<\omega} \to 2^{<\omega}$ (for example $f(\sigma) = 0^{|\sigma|}1\sigma$) the reals $\Omega_{\mathbf{U}}[\{\sigma\}]$ and $\Omega^\infty_{\mathbf{U}}[\{f(\sigma)\}]$ are very similar:

Theorem 3.8.5. *Let* $\mathbf{U}$ *be a monotone universal by adjunction machine. Let* $f\colon 2^{<\omega} \to 2^{<\omega}$ *be an injective computable function with computable and prefix-free range. Then there exists* k *such that for every* $\sigma \in 2^{<\omega}$*,*

$$2^{-k}\Omega_{\mathbf{U}}[\{\sigma\}] \leq \Omega^{\infty}_{\mathbf{U}}[\{f(\sigma)\}] \leq 2^{k}\Omega_{\mathbf{U}}[\{\sigma\}].$$

Proof. Consider the following prefix-free machine $\mathbf{M}$ which on input ρ simulates $\mathbf{U}(\rho)$ step by step until the current output is of the form $f(\sigma)$. If this ever happens then $\mathbf{M}(\rho) = \sigma$, else it becomes undefined. If $\mathbf{U}^{\infty}(Z) = f(\sigma)$ then there is n such that the constructed $\mathbf{M}$ guarantees that $\mathbf{M}(Z \upharpoonright n) = \sigma$. Since $\mathbf{U}$ is universal by adjunction, there is γ such that for all ρ, $\mathbf{U}(\gamma\rho) = \mathbf{M}(\rho)$. Therefore,

$$\begin{aligned}\Omega^{\infty}_{\mathbf{U}}[\{f(\sigma)\}] &\leq \mu\left(\{Z \in 2^{\omega}\colon (\exists n)\, \mathbf{M}(Z \upharpoonright n) = \sigma\}\right)\\ &= \sum_{\mathbf{U}(\gamma\rho)=\sigma} 2^{-|\rho|}\\ &\leq 2^{|\gamma|}\,\Omega_{\mathbf{U}}[\{\sigma\}].\end{aligned}$$

For the other inequality, let $\mathbf{N}\colon 2^{<\omega} \to 2^{<\omega}$ be the prefix-free machine such that $\mathbf{N}(\rho) = f(\mathbf{U}(\rho))$ and let ν be such that $\mathbf{U}(\nu\rho) = \mathbf{N}(\rho) = f(\mathbf{U}(\rho))$. We have $\mathbf{U}(\rho) = \sigma$ if and only if $\mathbf{U}(\nu\rho) = f(\sigma)$. Then

$$\begin{aligned}\Omega_{\mathbf{U}}[\{f(\sigma)\}] &\geq \sum_{\mathbf{U}(\nu\rho)=f(\sigma)} 2^{-|\nu\rho|}\\ &= 2^{-|\nu|} \sum_{\mathbf{U}(\rho)=\sigma} 2^{-|\rho|}\\ &= 2^{-|\nu|}\,\Omega_{\mathbf{U}}[\{\sigma\}].\end{aligned}$$

To conclude, observe that for any τ, $\Omega^{\infty}_{\mathbf{U}}[\{\tau\}] \geq \Omega_{\mathbf{U}}[\{\tau\}]$ and take $k = \max(|\gamma|, |\nu|)$. □

Using the above result, Lemma 3.5.9 can be easily transferred to infinite computations.

Lemma 3.8.6. *Let* $\mathbf{U}$ *be a monotone universal by adjunction machine. Then* $(\exists d)(\forall n)(\exists \sigma)\, 2^{-n-d} \leq \Omega^{\infty}_{\mathbf{U}}[\{\sigma\}] \leq 2^{-n+d}$*. In fact, for some constant* d'*, there are at least* $2^n/(d'\, n^2)$ *strings* $\sigma \in 2^{<\omega}$ *satisfying the inequalities.*

From Lemma 3.8.6, the proofs of Theorems 3.5.10 and 3.5.11 adapt easily to $\Omega^{\infty}_{\mathbf{U}}$, giving counterexamples which are included in the subset $2^{<\omega}$ of $2^{\leq\omega}$.

Theorem 3.8.7. *For every universal by adjunction* $\mathbf{U}$ *there is a* Δ^0_3 *set* $X \subseteq 2^{<\omega}$ *such that* $\Omega^{\infty}_{U}[X]$ *is not random.*

Proof. Follow the proof of Theorem 3.5.10. Recall the definition of $\mathbf{U}^{\infty}_t(\rho)$ from Definition 3.8.1. Observe that for a finite output $\sigma \in 2^{<\omega}$, if $\mathbf{U}^{\infty}(Z) = \sigma$ then there is least $\rho \prec Z$ such that $\mathbf{U}^{\infty}(\rho Y) = \sigma$ for any $Y \in 2^{\omega}$. This ρ is precisely the prefix of Z read by $\mathbf{U}^{\infty}$ at some stage t when the last bit of σ has been written in the output. This ρ can be characterized by

- $(\exists t)\, \mathbf{U}_t^\infty(\rho) = \sigma$: at some point $\mathbf{U}^\infty(\rho)$ produces σ;
- $(\forall t)\, \mathbf{U}_t^\infty(\rho) \preceq \sigma$: $\mathbf{U}^\infty(\rho)$ does not extend σ;
- $(\exists t)\, \mathbf{U}_t^\infty(\rho \upharpoonright (|\rho| - 1)) \uparrow$: $\mathbf{U}^\infty(\rho)$ needs to read the whole ρ.

All these three conditions are $\emptyset'$-computable, and hence $\Omega_{\mathbf{U}}^\infty[\{\sigma\}]$ is left-c.e. relative to $\emptyset'$ via some $\emptyset'$-computable approximation $\Omega_{\mathbf{U}}^\infty[\{\sigma\}]_s \to \Omega_{\mathbf{U}}^\infty[\{\sigma\}]$. Since for any rational r, condition $\Omega_{\mathbf{U}}^\infty[\{\sigma\}] < r$ holds if and only if $(\exists s)\, \Omega_{\mathbf{U}}^\infty[\{\sigma\}]_s$, $\emptyset''$ can compute any bit of $\Omega_{\mathbf{U}}^\infty[\{\sigma\}]$, and any bit of $\Omega_{\mathbf{U}}^\infty[X]$ for finite $X \subseteq 2^{<\omega}$. This gives a shift to Δ_3^0 instead of Δ_2^0. □

Theorem 3.8.8. *For every universal by adjunction* $\mathbf{U}$ *and any* $A \subseteq \mathbb{N}$*, there is a set* $X \subseteq 2^{<\omega}$ *such that* $A' \leq_T X \leq_T A' \oplus \emptyset''$ *and such that* $\Omega_{\mathbf{U}}^\infty[X]$ *is not random.*

Proof. Follow the proof of Theorem 3.5.11, replacing $\Omega_{\mathbf{U}}[X]$ by $\Omega_{\mathbf{U}}^\infty[X]$. Observe that the real number in equation (3.13) is no longer left-c.e. relative to A because we also need $\emptyset''$ to calculate it (see the explanation in the proof of Theorem 3.8.7). Hence this number in equation (3.13) is $A' \oplus \emptyset''$-computable. □

4. INFINITE COMPUTATIONS

In this chapter we define a program-size complexity function K^∞ as a variant of the prefix-free program-size complexity, based on Turing monotone machines performing possibly unending computations. We consider definitions of randomness and triviality for sequences in 2^ω relative to the K^∞ complexity. We prove that the classes of random sequences and K^∞-random sequences coincide, and that the K^∞-trivial sequences are exactly the computable ones. We also study some properties of K^∞ and compare it with other complexity functions. In particular, K^∞ is different from K^A, the prefix-free complexity of monotone machines with oracle A. Furthermore, we consider some properties of the program-size complexity for infinite computations for plain machines.

This chapter comprises joint work [10] with Verónica Becher, Silvana Picchi and André Nies, and joint work [8] with Verónica Becher.

4.1 Introduction

We consider monotone Turing machines (a one-way read-only input tape and a one-way write-only output tape) performing possibly infinite computations, and we define a program-size complexity function $K^\infty \colon 2^{<\omega} \to \mathbb{N}$ as a variant of the classical prefix-free program-size complexity: given a universal monotone machine $\mathbf{U}$, for any string $\sigma \in 2^{<\omega}$, $K^\infty(\sigma)$ is the length of a shortest string $\rho \in 2^{<\omega}$ read by $\mathbf{U}$, which produces σ via a possibly infinite computation (either a halting or a non-halting computation), having read exactly ρ from the input.

The classical prefix-free complexity K (recall Definition 1.4.2) is an upper bound of the function K^∞ (up to an additive constant), since the definition of K^∞ does not require that the machine $\mathbf{U}$ halts. We prove that K^∞ differs from K in that it has no monotone decreasing computable approximation and it is not subadditive.

The complexity K^∞ is closely related with the monotone complexity Km, independently introduced by Levin [53, 51] and Schnorr [67] (see [68] and [54] for historical details and differences between various monotone complexities). Levin defines $Km(\sigma)$ as the length of the shortest halting program that provided with n $(0 \leq n \leq |\sigma|)$, outputs $\sigma \restriction n$. Equivalently $Km(\sigma)$ can be defined as the least number of bits read by a monotone machine $\mathbf{U}$ which via a possibly infinite computation produces any finite or infinite extension of σ.

Km is a lower bound of K^∞ (up to an additive constant) since the definition of K^∞ imposes that the machine $\mathbf{U}$ reads exactly the input ρ and produces exactly the output σ. Every computable $A \in 2^\omega$ is the output of some monotone machine with no input, so there is some c such that $(\forall n)\, Km(A \restriction n) \leq c$. Moreover, there exists

n_0 such that $(\forall n, m \geq n_0)\, Km(A \upharpoonright n) = Km(A \upharpoonright m)$. We show this is not the case with K^∞, since for every infinite $B = \{b_1, b_2, \ldots\} \subseteq 2^{<\omega}$, $\lim_{n\to\infty} K^\infty(b_n) = \infty$. This is also a property of the classical prefix-free complexity K, and we consider it as a decisive property that distinguishes K^∞ from Km.

The formal definitions for an infinite computation are stated in section 4.2 and in section 4.3 we formally define the complexity K^∞.

The prefix-free complexity of a universal machine with oracle $\emptyset'$, the function $K^{\emptyset'}$, is also a lower bound of K^∞ (up to an additive constant). In section 4.4 we analyze some properties of the K^∞-complexity of strings and compare it with K and $K^{\emptyset'}$ showing that these three complexities are different. In addition we show that for every oracle A, K^∞ differs from K^A, the prefix-free complexity of a universal machine with oracle A (recall Definition 1.4.7).

In sections 4.5 and 4.6 we turn from strings to infinite sequences and we study the K^∞ complexity along the initial segments of infinite sequences in 2^ω. We consider definitions of randomness and triviality based on the K^∞ complexity. A sequence is K^∞-random if its initial segments have maximal K^∞ complexity. Since Km gives a lower bound of K^∞ and Km-randomness coincides with randomness as defined in section 1.5 [51], the classes of random, K^∞-random and Km-random coincide. We argue for a definition of K^∞-trivial sequences as those whose initial segments have minimal K^∞ complexity. While every computable $A \in 2^\omega$ is both K-trivial and K^∞-trivial, we show that the class of K^∞-trivial sequences is strictly included in the class of K-trivial sequences. Moreover, we prove that the computable sequences coincide with those which are K^∞-trivial.

Finally, in section 4.7 we investigate some properties of the graph of the complexity function C^∞, the plain version of K^∞, and we compare them with those of C and $C^{\emptyset'}$.

4.2 Infinite computations on monotone machines

A *monotone* machine, defined by Levin [51], is a Turing machine with a one-way read-only input tape, some work tapes, and a one-way write-only output tape. The input tape contains a first dummy cell (representing the empty input) and then a one-way infinite sequence of 0s and 1s, and initially the input head scans the leftmost dummy cell. The output tape is written one symbol, 0 or 1 at a time. Notice that the output grows with respect to the prefix ordering in $2^{<\omega}$ as the computational time increases because the output tape only moves to the left (or, equivalently, the output head only moves to the right).

A *possibly infinite computation* is either a halting or a non-halting computation. If the machine halts, the output of the computation is the finite string written on the output tape. Else, the output is either a finite string or an infinite sequence written on the output tape as a result of a never ending process. This leads us to consider $2^{\leq\omega} = 2^{<\omega} \cup 2^\omega$ as the output space.

In this chapter we restrict ourselves to possibly infinite computations on monotone machines which read just finitely many symbols from the input tape. However, one might consider the more general scenario, where the monotone machine could read either finite or infinite many bits from the input tape [13, 14].

For convenience, in this chapter we will use a definition of step computation as in Definition 3.8.1. This definition is slightly different from the one described in section 1.4:

Definition 4.2.1. Let $\mathbf{M}$ be a monotone machine. $\mathbf{M}_t(\rho)$ is the *current* output of $\mathbf{M}$ on input $\rho \in 2^{<\omega}$ by stage t if it has not read beyond the end of ρ. Otherwise, $\mathbf{M}_t(\rho)$ is undefined.

Notice that $\mathbf{M}_t(\rho)$ does not require that the computation on input ρ halts.

Remark 4.2.2. Observe that:

1. If $\mathbf{M}_t(\rho)\uparrow$ then $\mathbf{M}_u(\gamma)\uparrow$ for all $\gamma \preceq \rho$ and $u \geq t$.
2. If $\mathbf{M}_t(\rho)\downarrow$ then $\mathbf{M}_u(\gamma)\downarrow$ for any $\gamma \succeq \rho$ and $u \leq t$. Also, if at stage t, $\mathbf{M}$ reaches a halting state without having read beyond the end of ρ, then $\mathbf{M}_u(\rho)\downarrow = \mathbf{M}_t(\rho)$ for all $u \geq t$.
3. Since $\mathbf{M}$ is monotone, $\mathbf{M}_t(\rho) \preceq \mathbf{M}_{t+1}(\rho)$, in case $\mathbf{M}_{t+1}(\rho)\downarrow$.
4. $\mathbf{M}_t(\rho)$ has computable domain, in the sense that the function which takes a ρ and t, and decides whether $\mathbf{M}_t(\rho)\downarrow$ or $\mathbf{M}_t(\rho)\uparrow$ is computable.

The following definition states the precise meaning of the infinite behavior for the computations we study in this chapter.

Definition 4.2.3 (Infinite computation)**.** Let $\mathbf{M}$ be a monotone machine.

1. The *input/output behavior of* $\mathbf{M}$ *for halting computations* is the partial computable map $\mathbf{M} : 2^{<\omega} \to 2^{<\omega}$ given by the usual computation of $\mathbf{M}$, i.e., $\mathbf{M}(\rho)\downarrow$ iff $\mathbf{M}$ enters into a halting state on input ρ without reading beyond ρ. If $\mathbf{M}(\rho)\downarrow$ then $\mathbf{M}(\rho) = \mathbf{M}_t(\rho)$ for some stage t at which $\mathbf{M}$ entered a halting state.
2. The *input/output behavior of* $\mathbf{M}$ *for possibly infinite computations* is the map $\mathbf{M}^\infty : 2^{<\omega} \to 2^{\leq\omega}$ given by $\mathbf{M}^\infty(\rho) = \lim_{t\to\infty} \mathbf{M}_t(\rho)$.

Proposition 4.2.4.

1. dom($\mathbf{M}$) *is closed under extensions and its syntactical complexity is* Σ^0_1.

2. dom($\mathbf{M}^\infty$) *is closed under extensions and its syntactical complexity is* Π^0_1.

3. $\mathbf{M}^\infty$ *extends* $\mathbf{M}$.

Proof. 1. is trivial.

2. $\mathbf{M}^\infty(\rho)\downarrow$ iff for all t $\mathbf{M}$ on input ρ does not read $\rho0$ and does not read $\rho1$ at stage t. Clearly, dom($\mathbf{M}^\infty$) is closed under extensions since if $\mathbf{M}^\infty(\rho)\downarrow$ then $\mathbf{M}^\infty(\gamma)\downarrow = \mathbf{M}^\infty(\rho)$ for every $\gamma \succeq \rho$.

3. Since in the input/output behavior of $\mathbf{M}$ for possibly infinite computations, the machine $\mathbf{M}$ is not required to halt, $\mathbf{M}^\infty$ extends $\mathbf{M}$.

□

Remark 4.2.5. An alternative definition of the functions $\mathbf{M}$ and $\mathbf{M}^\infty$, would be to consider them with prefix-free domains (instead of closed under extensions):

1. $\mathbf{M}(\rho)\downarrow$ iff at some stage t $\mathbf{M}$ enters a halting state having read exactly ρ. If $\mathbf{M}(\rho)\downarrow$ then its value is $\mathbf{M}_t(\rho)$ for such stage t.
2. $\mathbf{M}^\infty(\rho)\downarrow$ iff there exists t at which $\mathbf{M}$ has read exactly ρ and for every t' $\mathbf{M}$ does not read $\rho 0$ nor $\rho 1$. If $\mathbf{M}^\infty(\rho)\downarrow$ then its value is $\lim_{t\to\infty} \mathbf{M}_t(\rho)$.

As we mentioned in section 1.4, there is an effective enumeration $(\mathbf{T}_e)_{e\in\mathbb{N}}$ of all monotone machines. We fix any monotone universal machine $\mathbf{U}$ as in Definition 1.4.3.

By Shoenfield's Limit Lemma every $\mathbf{M}^\infty\colon 2^{<\omega} \to 2^{<\omega}$ is computable in $\emptyset'$. However, possibly infinite computations on *monotone* machines cannot compute all $\emptyset'$-computable functions. For instance, the characteristic function of the Halting Problem cannot be computed in the limit by a monotone machine. In contrast, the Busy Beaver function in unary notation $bb\colon \mathbb{N} \to 1^*$:

$$bb(n) \quad = \quad \begin{array}{l}\text{the maximum number of 1's produced by any Turing machine}\\ \text{with } n \text{ states which halts with no input}\end{array}$$

is just $\emptyset'$-computable and $bb(n)$ is the output of a non-halting computation which on input n, simulates every Turing machine with n states and for each one that halts updates, if necessary, the output with more 1s.

4.3 Program size complexities on monotone machines

Let $\mathbf{M}$ be a monotone machine, and $\mathbf{M}$, $\mathbf{M}^\infty$ the respective maps for the input/output behavior of $\mathbf{M}$ for halting computations and possibly infinite computations as in Definition 4.2.3. We define the program-size complexity for infinite computations associated to $\mathbf{M}$ in the same way as in Definition 1.4.2, but considering the mapping $\mathbf{M}^\infty$:

Definition 4.3.1 (Prefix-free program-size complexity for infinite computations)**.** $K^\infty_{\mathbf{M}}\colon 2^{\leq\omega} \to \mathbb{N}$, *the program-size complexity for infinite computations based on machine* $\mathbf{M}$, is defined as:

$$K^\infty_{\mathbf{M}}(\sigma) = \begin{cases} \min\{|\rho|\colon \mathbf{M}^\infty(\rho) = \sigma\} & \text{if } \sigma \text{ is in the range of } \mathbf{M}^\infty;\\ \infty & \text{otherwise.}\end{cases}$$

For $\mathbf{U}$ we drop subindexes and we simply write K and K^∞. The Invariance Theorem 1.4.4 holds for K^∞: for all monotone machine $\mathbf{M}$ there exists a constant c such that

$$(\forall \sigma \in 2^{\leq\omega})\, K^\infty(\sigma) \leq K^\infty_{\mathbf{M}}(\sigma) + c.$$

The complexity function K^∞ was first introduced in [6] without a detailed study of its properties. Notice that if we take monotone machines $\mathbf{M}$ according to Remark 4.2.5 instead of Definition 4.2.3, we obtain *the same* complexity functions $K_{\mathbf{M}}$ and $K^\infty_{\mathbf{M}}$.

In this work we only consider the K^∞ complexity of finite strings, that is, we restrict our attention to $K^\infty \colon 2^{<\omega} \to \mathbb{N}$. We will compare K^∞ with these other complexity functions:

- $K^A \colon 2^{<\omega} \to \mathbb{N}$ is the program-size complexity function for $\mathbf{U}^A$, a monotone universal machine with oracle A (recall Definition 1.4.7). We pay special attention to $A = \emptyset'$.

- $Km \colon 2^{\leq\omega} \to \mathbb{N}$, where

  $$Km_{\mathbf{M}}(\sigma) = \min\{|\rho| \colon \mathbf{M}^\infty(\rho) \succeq \sigma\}$$

  is the *monotone complexity function* defined by Levin [53, 51] for a monotone machine $\mathbf{M}$ and, as usual, for $\mathbf{U}$ we simply write Km.

Schnorr [67] worked with monotone machines which are *discrete*, that is, mapping strings to strings. If $\tilde{U}$ is a universal discrete machine, the *discrete monotone program-size complexity complexity* or *process complexity* (as Schnorr called it) is defined as $Km_D(\sigma) = \{|\rho| \colon \tilde{\mathbf{U}}(\rho) = \sigma\}$. Levin [53, 51] (see also [40]) showed that a real A is random if and only if there is a constant c such that for all n $Km(A \upharpoonright n) > n - c$. Schnorr used his process complexity to characterize the same class of random reals. Chaitin's characterization of computable reals by the plain C complexity [24], works for the discrete monotone complexity: A is computable if and only if there is a constant c such that for all n, $Km_D(A \upharpoonright n) \leq Km_D(1^n) + c$. The proof of this can be found in [32].

In [6], Becher et al. proved that:

Proposition 4.3.2. *For all strings $\sigma \in 2^{<\omega}$, $K^{\emptyset'}(\sigma) \leq K^\infty(\sigma) + \mathcal{O}(1)$ and $K^\infty(\sigma) \leq K(\sigma)$.*

Proof. $K^\infty(\sigma) \leq K(\sigma)$ is trivial because for halting computations $\mathbf{U}(\rho) = \mathbf{U}^\infty(\rho)$. That is, according to Definition 4.2.3 if there is a stage t where $\mathbf{U}_t(\rho)\downarrow$ then $\mathbf{U}$ has read all the input ρ and will not write or read any other symbol, so $\mathbf{U}^\infty(\rho) = \mathbf{U}(\rho)$. Therefore if ρ is a minimal $\mathbf{U}$- description of σ then it is a $\mathbf{U}^\infty$-description of σ (possibly not minimal).

To see $K^{\emptyset'}(\sigma) \leq K^\infty(\sigma) + \mathcal{O}(1)$, observe that any infinite computation can be simulated with the help of the $\emptyset'$ oracle. Suppose the machine $\mathbf{M}$ with oracle $\emptyset'$ and input ρ executes $\mathbf{U}_t(\rho)$ for $t = 0, t = 1, \ldots$ until it finds a t such that $(\forall s \geq t)\, \mathbf{U}_s(\rho) = \mathbf{U}_t(\rho)$ (this question can be answered by $\emptyset'$). Once this happens, it writes $\mathbf{U}_t(\rho)$ in the output and terminates. Suppose now that ρ is a minimal $\mathbf{U}^\infty$-description of σ, that is $\mathbf{U}^\infty(\rho) = \sigma$ and $|\rho| = K^\infty(\sigma)$. Then $\mathbf{M}(\rho) = \mathbf{U}^\infty(\rho)$ and hence $K^{\emptyset'}(\sigma) \leq K^\infty(\sigma) + \mathcal{O}(1)$. □

4.4 Properties of K^∞

It can be seen that K^∞ shares a lot of properties with K. The following properties of K^∞ are in the spirit of those of K.

Proposition 4.4.1. *For all strings σ and τ*

1. $K(\sigma) \leq K^\infty(\sigma) + K(|\sigma|) + \mathcal{O}(1)$.
2. *For any n,* $\|\{\sigma \in 2^{<\omega} \colon K^\infty(\sigma) \leq n\}\| < 2^{n+1}$.
3. $K^\infty(\tau\sigma) \leq K^\infty(\sigma) + K(\tau) + \mathcal{O}(1)$.
4. $K^\infty(\sigma) \leq K^\infty(\sigma\tau) + K(|\tau|) + \mathcal{O}(1)$.
5. $K^\infty(\sigma) \leq K^\infty(\sigma\tau) + K^\infty(|\sigma|) + \mathcal{O}(1)$.

Proof. 1. Let $\rho, \gamma \in 2^{<\omega}$ such that $\mathbf{U}^\infty(\rho) = \sigma$ and $\mathbf{U}(\gamma) = |\sigma|$. Then there is a machine that first simulates $\mathbf{U}(\gamma)$ to obtain $|\sigma|$, then it starts a simulation of $\mathbf{U}^\infty(\rho)$ writing its output on the output tape, until it has written $|\sigma|$ symbols, and then it halts.

2. It follows immediately from the fact that there are at most $2^{n+1} - 1$ strings of length less than or equal to n.

3. Let $\rho, \gamma \in 2^{<\omega}$ such that $\mathbf{U}^\infty(\rho) = \sigma$ and $\mathbf{U}(\gamma) = \tau$. Then there is a machine that first simulates $\mathbf{U}(\gamma)$ until it halts and prints τ on the output tape. Then, it starts a simulation of $\mathbf{U}^\infty(\rho)$ writing its output on the output tape. This process will finally fill the output tape with the string $\tau\sigma$.

4. Let $\rho, \gamma \in 2^{<\omega}$ such that $\mathbf{U}^\infty(\rho) = \sigma\tau$ and $\mathbf{U}(\gamma) = |\tau|$. Then there is a machine that first simulates $\mathbf{U}(\gamma)$ until it halts to obtain $|\tau|$. Then it starts a simulation of $\mathbf{U}^\infty(\rho)$ such that at each stage s of the simulation it writes the symbols needed to leave $\mathbf{U}_s(\rho) \upharpoonright (|\mathbf{U}_s(\rho)| - |\tau|)$ on the output tape, that is, it does not write the last $|\tau|$ bits at any stage.

5. Consider the following monotone machine $\mathbf{M}$:

```
1. set $t = 1$, $\rho = \lambda$ and $\gamma = \lambda$
2. repeat
   a. if $\mathbf{U}_t(\rho)$ asks for reading then append to $\rho$ the next bit in the input of $\mathbf{M}$
   b. if $\mathbf{U}_t(\gamma)$ asks for reading then append to $\gamma$ the next bit in the input of $\mathbf{M}$
   c. extend the actual output to $\mathbf{U}_t(\gamma) \upharpoonright (\mathbf{U}_t(\rho))$
   d. set $t = t + 1$
```

If $\tilde{\rho}$ and $\tilde{\gamma}$ are shortest programs such that $\mathbf{U}^\infty(\tilde{\rho}) = |\sigma|$ and $\mathbf{U}^\infty(\tilde{\gamma}) = \sigma\tau$ respectively, then we can interleave $\tilde{\rho}$ and $\tilde{\gamma}$ in a way such that at each stage t, $\rho \preceq \tilde{\rho}$ and $\gamma \preceq \tilde{\gamma}$ (notice that eventually $\rho = \tilde{\rho}$ and $\gamma = \tilde{\gamma}$). Thus, this machine $\mathbf{M}$ computes σ and it never reads more than $K^\infty(\sigma\tau) + K^\infty(|\sigma|)$ bits.

This completes the proof. □

As we explained in section 1.4, K is computably approximable from above. The following results show that this is not the case with K^∞.

Proposition 4.4.2. *There is no effective decreasing approximation of K^∞.*

Proof. Suppose there is a computable function $h\colon 2^{<\omega} \times \mathbb{N} \to \mathbb{N}$ such that for every string σ, $\lim_{t\to\infty} h(\sigma, t) = K^\infty(\sigma)$ and $(\forall t)\, h(\sigma, t) \geq h(\sigma, t+1)$. We write $h_t(\sigma)$ for $h(\sigma, t)$. Consider the monotone machine $\mathbf{M}$ with coding constant $d \geq 0$ given by the Recursion Theorem, which on input ρ does the following:

```
1. set $t = 1$ and print 0
2. repeat forever
    a. let $n$ be the number of bits read by $\mathbf{U}_t(\rho)$
    b. print $\sigma$ for each string $\sigma$ not yet printed, $|\sigma| \leq t$ and $h_t(\sigma) \leq n + d$
    c. set $t = t + 1$
```

Let ρ be a program such that $\mathbf{U}^\infty(\rho) = \chi$ and $|\rho| = K^\infty(\chi)$. Notice that, as $t \to \infty$, the number of bits read by $\mathbf{U}_t(\rho)$ goes to $|\rho| = K^\infty(\chi)$. Let t_0 be such that for all $t \geq t_0$, $\mathbf{U}_t(\rho)$ reads no more from the input. Since there are only finitely many strings σ such that $K^\infty(\sigma) \leq K^\infty(\chi) + d$, there is a $t_1 \geq t_0$ such that for all $t \geq t_1$ and for all those strings σ, $h_t(\sigma) = K^\infty(\sigma)$. Hence, every string σ with $K^\infty(\sigma) \leq K^\infty(\chi) + d$ will be printed (in some order).

Let $\tilde{\chi} = \mathbf{M}^\infty(\rho)$. On one hand, we have $K^\infty(\tilde{\chi}) \leq |\rho| + d = K^\infty(\chi) + d$. On the other hand, by the construction of $\mathbf{M}$, $\tilde{\chi}$ cannot be the output of a program of length at most $K^\infty(\chi) + d$ (because $\tilde{\chi}$ is different from each string σ such that $K^\infty(\sigma) \leq K^\infty(\chi) + d$). So it must be that $K^\infty(\tilde{\chi}) > K^\infty(\chi) + d$, a contradiction. □

The following Lemma states a critical property that distinguishes K^∞ from K. It implies that K^∞ is not sub-additive, i.e., there is no constant c such that for every pair of strings σ and τ, $K^\infty(\sigma\tau) \leq K^\infty(\sigma) + K^\infty(\tau) + c$. It also implies that K^∞ is not invariant under computable permutations $2^{<\omega} \to 2^{<\omega}$.

Lemma 4.4.3. *For every total computable function f there is a natural k such that*

$$K^\infty(0^k 1) > f(K^\infty(0^k)).$$

Proof. Let f be any computable function and let $\mathbf{M}$ be the following monotone machine with coding constant d given by the Recursion Theorem:

```
1. set $t = 1$
```

```
2. do forever
    a. for each $\rho$ such that $|\rho| \leq \max\{f(i)\colon 0 \leq i \leq d\}$
    b. if $\mathbf{U}_t(\rho) = 0^j1$ then print enough 0s to leave at least $0^{j+1}$ on the output tape
    c. set $t = t + 1$
```

Let $n = \max\{f(i)\colon 0 \leq i \leq d\}$. We claim there is a k such that $\mathbf{M}^\infty(\lambda) = 0^k$. Since there are only finitely many programs of length less than or equal to n which output a string of the form 0^j1 for some j, then there is some stage t at which $\mathbf{M}$ has written 0^k, with k greater than all such js. From that stage t on, $\mathbf{M}^\infty$ does not print anything else. Therefore, there is no program ρ with $|\rho| \leq n$ such that $\mathbf{U}^\infty(\rho) = 0^k1$.

If $\mathbf{M}^\infty(\lambda) = 0^k$ then $K^\infty(0^k) \leq d$. So, $f(K^\infty(0^k)) \leq n$. Also, for this k, there is no program ρ of length at most n such that $\mathbf{U}^\infty(\rho) = 0^k$ and thus $K^\infty(0^k1) > n$. Hence, $f(K^\infty(0^k)) \leq n < K^\infty(0^k1)$. □

Note that $K(0^k) = K(0^k1) = K^\infty(0^k1)$ up to additive constants, so the above lemma gives an example of a string where K^∞ is much smaller that K. Here *much smaller* means that even when we apply an arbitrary computable function f to $K^\infty(0^k)$, this is still below $K(0^k)$.

Proposition 4.4.4. K^∞ *is not sub-additive*

Proof. Suppose there is a constant c such that for every two strings σ and τ, $K^\infty(\sigma\tau) \leq K^\infty(\sigma)+K^\infty(\tau)+c$. Take the computable injection $f(n) = n+K^\infty(1)+c$. By Lemma 4.4.3 there is k such that $K^\infty(0^k1) > K^\infty(0^k) + K^\infty(1) + c$. This contradicts the assumption for $\sigma = 0^k$ and $\tau = 1$. □

The complexity function K is invariant under computable injections, in the sense that for any computable one-one function $g\colon 2^{<\omega} \to 2^{<\omega}$ there is a constant c such that $(\forall\sigma \in 2^{<\omega})\ |K(g(\sigma)) - K(\sigma)| \leq c$. It is immediate from Lemma 4.4.3 that this is not the case with K^∞.

It is known that the complexity function K is smooth in the length and lexicographic order over $2^{<\omega}$ in the sense that there is a constant c such that

$$|K(\mathrm{str}(n)) - K(\mathrm{str}(n+1))| \leq c.$$

However, this is not the case for K^∞.

Proposition 4.4.5. K^∞ *is not smooth in the length and lexicographical order over* $2^{<\omega}$.

Proof. Notice that there is a constant c such that $(\forall n > 1)\, K^\infty(0^n1) \leq K^\infty(0^{n-1}1)+c$, because if $\mathbf{U}^\infty(\rho) = 0^{n-1}1$ then there is a machine that first writes a 0 on the output tape and then simulates $\mathbf{U}^\infty(\rho)$. By Lemma 4.4.3, for each c there is an n such that $K^\infty(0^n1) > K^\infty(0^n) + c$. Joining the two inequalities, we obtain $(\forall c)(\exists n)\, K^\infty(0^{n-1}1) > K^\infty(0^n) + c - d$. Since $\mathrm{str}^{-1}(0^{n-1}1) = \mathrm{str}^{-1}(0^n) + 1$, K^∞ is not smooth. □

However, K^∞ is smooth in a less restrictive way:

Proposition 4.4.6. $(\forall n)$ $|K^\infty(\mathrm{str}(n)) - K^\infty(\mathrm{str}(n+1))| \leq K(|\mathrm{str}(n)|) + \mathcal{O}(1)$.

Proof. Consider the following monotone machine $\mathbf{M}$ with input $\rho\gamma$:

```
1. obtain $m = \mathbf{U}(\rho)$
2. simulate $\sigma = \mathbf{U}^\infty(\gamma)$ till it outputs $m$ bits
3. write $\mathrm{str}(\mathrm{str}^{-1}(\sigma) + 1)$
```

Let $\rho, \gamma \in 2^{<\omega}$ such that $\mathbf{U}(\rho) = |\mathrm{str}(n)|$ and $\mathbf{U}^\infty(\gamma) = \mathrm{str}(n)$. Then, $\mathbf{M}^\infty(\rho\gamma) = \mathrm{str}(n+1)$ and

$$K^\infty(\mathrm{str}(n+1)) \leq K^\infty(\mathrm{str}(n)) + K(|\mathrm{str}(n)|) + \mathcal{O}(1).$$

Similarly, if $\mathbf{M}$, instead of writing $\mathrm{str}(\mathrm{str}^{-1}(\sigma) + 1)$, writes $\mathrm{str}(\mathrm{str}^{-1}(\sigma) - 1)$, we conclude

$$K^\infty(\mathrm{str}(n)) \leq K^\infty(\mathrm{str}(n+1)) + K(|\mathrm{str}(n+1)|) + \mathcal{O}(1).$$

Since $|K(|\mathrm{str}(n)|) - K(|\mathrm{str}(n+1)|)| \leq \mathcal{O}(1)$, it follows that

$$|K^\infty(\mathrm{str}(n)) - K^\infty(\mathrm{str}(n+1))| \leq K(|\mathrm{str}(n)|) + \mathcal{O}(1).$$

This completes the proof. □

Proposition 4.3.2 states that K^∞ is between K and $K^{\emptyset'}$. The following result shows that K^∞ is really strictly in between them, in the sense that it can be separated from those upper and lower bounds.

Proposition 4.4.7. *For every c there is $\sigma \in 2^{<\omega}$ such that*

$$K^{\emptyset'}(\sigma) + c < K^\infty(\sigma) < K(\sigma) - c.$$

Proof. Let $\tau_n = \min\{\sigma \in 2^n \colon K(\sigma) \geq n\}$ and consider a machine $\mathbf{M}$ which on input i does the following:

```
1. set $j = 0$
2. repeat
    a. write $j$ in binary
    b. search for a program $\rho$, $|\rho| \leq 3i$, such that $\mathbf{U}(\rho) = j$. If found, go to the next step.
    c. set $j = j + 1$
```

If $\mathbf{M}^\infty(i)$ outputs the string $01\dots k_i$, then $K(k_i) > 3i$ and for all j, $0 \le j < k_i$ we have $K(j) \le 3i$. We define $\sigma_i = \tau_i 01 \dots k_i$. Let us see that both $K^\infty(\sigma_i) - K^{\emptyset'}(\sigma_i)$ and $K(\sigma_i) - K^\infty(\sigma_i)$ grow arbitrarily when increasing i.

On one hand, we can construct a machine which on input i and ρ executes $\mathbf{U}^\infty(\rho)$ till it outputs i bits and then halts. Now suppose we instantiate ρ in the shortest $\mathbf{U}^\infty$-program for σ_i. Since the first i bits of σ_i corresponds to τ_i and $K(i) \le 2|i| + \mathcal{O}(1)$, we have $i \le K(\tau_i) \le K^\infty(\sigma_i) + 2|i| + \mathcal{O}(1)$. But with the help of the $\emptyset'$-oracle we can compute σ_i from i, so $K^{\emptyset'}(\sigma_i) \le 2|i| + \mathcal{O}(1)$. Thus we have $K^\infty(\sigma_i) - K^{\emptyset'}(\sigma_i) \ge i - 4|i| - \mathcal{O}(1)$.

On the other hand, given i and σ_i, we can effectively compute k_i. Hence, we have $3i < K(k_i) \le K(\sigma_i) + 2|i| + \mathcal{O}(1)$ for all i. Also, given τ_i, we can compute σ_i in the limit using the idea of machine $\mathbf{M}$, and hence $K^\infty(\sigma_i) \le 2|\tau_i| + \mathcal{O}(1) = 2i + \mathcal{O}(1)$. Then, for all i $K(\sigma_i) - K^\infty(\sigma_i) > i - 2|i| - \mathcal{O}(1)$. □

Not only K^∞ is different from $K^{\emptyset'}$ but it differs from K^A (the prefix-free complexity of a universal monotone machine with oracle A), for every A.

Theorem 4.4.8. *There is no oracle A such that $\left|K^\infty - K^A\right| \le c$ for some fixed constant c.*

Proof. Immediate from Lemma 4.4.3 and from the standard result that for all A, K^A is sub-additive, so in particular there is a c such that for every k, $K^A(0^k 1) \le K^A(0^k) + c$. □

The advantage of K^∞ over K can be seen along the initial segments of every computable sequence: if $A \in 2^\omega$ is computable then there are infinitely many ns such that $K(A \upharpoonright n) - K^\infty(A \upharpoonright n) > c$, for an arbitrary c.

Proposition 4.4.9. *Let $A \in 2^\omega$ be a computable sequence. Then*

1. $\limsup_{n\to\infty} K(A \upharpoonright n) - K^\infty(A \upharpoonright n) = \infty$
2. $\limsup_{n\to\infty} K^\infty(A \upharpoonright n) - Km(A \upharpoonright n) = \infty$

Proof.

For 1, consider the following monotone machine $\mathbf{M}$ with input ρ:

```
1. obtain $n = \mathbf{U}(\rho)$
2. write $A \upharpoonright (\mathrm{str}^{-1}(0^n) - 1)$
3. for every string $\sigma$ of length $n$ in lexicographic order
    a. write $A(\mathrm{str}^{-1}(\sigma))$
    b. search for a program $\gamma$ such that $|\gamma| < n$ and $\mathbf{U}(\gamma) = \sigma$. If never found, get undefined.
```

Recall that $A(n)$ denotes the n-th bit of A (starting with 0). If $\mathbf{U}(p) = n$, then $\mathbf{M}^\infty(p)$ outputs $A \upharpoonright k_n$ for some k_n such that $2^n - 1 \leq k_n < 2^{n+1} - 1$, since for all n there is a string of length n with K-complexity greater than or equal to n. Let us fix n. On one hand, $K^\infty(A \upharpoonright k_n) \leq K(n) + \mathcal{O}(1)$. On the other, $K(A \upharpoonright k_n) \geq n - \mathcal{O}(1)$, because we can compute the first string in the lexicographic order with K-complexity greater than or equal to n from a program for $A \upharpoonright k_n$. (Indeed, from $A \upharpoonright k_n$, we can compute k_n, and, from it, n by taking logarithms. Knowing n we can drop the first 2^n bits of $A \upharpoonright k_n$ and the obtained string will have complexity greater than or equal to n.) Hence, for each n, $K(A \upharpoonright k_n) - K^\infty(A \upharpoonright k_n) \geq n - K(n) - \mathcal{O}(1)$.

Item 2 is trivial because for each computable sequence A there is a constant c such that $Km(A \upharpoonright n) \leq c$ and $\lim_{n\to\infty} K^\infty(B \upharpoonright n) = \infty$ for every $B \in 2^\omega$. □

4.5 K^∞-triviality

As we mentioned in section 1.4, there is a standard convention to use K with arguments in $\mathbb{N}$. That is, for any $n \in \mathbb{N}$, $K(n)$ is written instead of $K(f(n))$ where f is some particular representation of the natural numbers in $2^{<\omega}$. This convention makes sense because K is invariant (up to a constant) for any computable representation of the natural numbers. That is, one can codify the natural n with its binary representation, σ_n, or with 0^n, and one knows that $K(0^n) = K(\sigma_n)$ up to an additive constant.

Recall from Definition 1.6.1 that $A \in 2^\omega$ is *K-trivial* if and only if there is a constant c such that for all n, $K(A \upharpoonright n) \leq K(n)+c$. The idea is that K-trivial sequences are exactly those whose initial segments have minimal K-complexity. Considering the above convention, A is K-trivial if and only if

$$(\exists c)(\forall n)\, K(A \upharpoonright n) \leq K(0^n) + c.$$

In general K^∞ is not invariant for computable representations of $\mathbb{N}$. We propose the following definition that ensures that computable sequences are K^∞-trivial.

Definition 4.5.1 (K^∞-triviality)**.** $A \in 2^\omega$ is *K^∞-trivial* if and only if

$$(\exists c)(\forall n)\, K^\infty(A \upharpoonright n) \leq K^\infty(0^n) + c.$$

Our choice of the right hand side of the above definition is supported by the following proposition.

Proposition 4.5.2. *Let $f\colon \mathbb{N} \to 2^{<\omega}$ be computable and strictly increasing with respect to the length and lexicographical order over $2^{<\omega}$. Then*

$$(\forall n)\, K^\infty(0^n) \leq K^\infty(f(n)) + \mathcal{O}(1).$$

Proof. Notice that, since f is strictly increasing, f has computable range. We construct the following monotone machine $\mathbf{M}$ with input ρ:

1. set $t = 0$

```
2. repeat
    a. if $\mathbf{U}_t(\rho)\downarrow$ is in the range of $f$ then let $n = f^{-1}(\mathbf{U}_t(\rho))$
    b. print the needed 0s to leave $0^n$ on the output tape
    c. $t = t + 1$
```

Since f is increasing in the length and lexicographic order over $2^{<\omega}$, if ρ is a program for $\mathbf{U}$ such that $\mathbf{U}^\infty(\rho) = f(n)$, then $\mathbf{M}^\infty(\rho) = 0^n$. □

Chaitin observed that every computable $A \in 2^\omega$ is K-trivial [24] and that K-trivial sequences are Δ^0_2. However, K-triviality does not characterize the class Δ^0_1 of computable sequences: Solovay [73] constructed a Δ^0_2 sequence which is K-trivial but not computable (see also [34] for the construction of a strongly left-c.e. real with the same properties). Our next result implies that K^∞-trivial sequences are Δ^0_2, as it happens with K-triviality. Theorem 4.5.6 characterizes Δ^0_1 as the class of K^∞-trivial sequences.

Theorem 4.5.3. *Suppose that A is a sequence such that, for some $b \in \mathbb{N}$,*

$$(\forall n)\, K^\infty(A \upharpoonright n) \leq K(n) + b.$$

Then A is K-trivial.

Proof. The idea is to define a Δ^0_2 tree T such that $A \in [T]$, and a Kraft-Chaitin set W showing that each path of T is K-trivial (recall Definition 1.4.5 of a Kraft-Chaitin set). For $\sigma \in 2^{<\omega}$ and $t \in \mathbb{N}$, let

$$\begin{aligned} K^\infty_t(\sigma) &= \min\{|\rho| \colon \mathbf{U}_t(\rho) = \sigma\} \\ K_t(\sigma) &= \min\{|\rho| \colon \mathbf{U}_t(\rho) = \sigma \text{ and } \mathbf{U}(\rho) \text{ halts in at most } t \text{ steps}\} \end{aligned}$$

be effective approximations of K^∞ and K respectively. Notice that for all $\sigma \in 2^{<\omega}$, $\lim_{t\to\infty} K^\infty_t(\sigma) = K^\infty(\sigma)$ and $\lim_{t\to\infty} K_t(\sigma) = K(\sigma)$.

Given s, let

$$T_s = \{\sigma \colon |\sigma| < s \ \wedge\ (\forall m \leq |\sigma|)\, K^\infty_s(\sigma \upharpoonright m) \leq K_s(m) + b\}\,.$$

Then $(T_s)_{s\in\mathbb{N}}$ is an effective approximation of a Δ^0_2 tree T, and $[T]$ is the class of sequences A satisfying

$$(\forall n)\, K^\infty(A \upharpoonright n) \leq K(n) + b.$$

Let $r = K_s(|\sigma|)$. We define a Kraft-Chaitin set W as follows: if $\sigma \in T_s$ and either

1. there is $u < s$ greatest such that $\sigma \in T_u$ and $r < K_u(|\sigma|)$, or

2. $\sigma \notin T_u$ for all $u < s$,

then put an axiom $\langle r+b+1, \sigma\rangle$ into W. That is, in case 2 we put an axiom $\langle K_s(|\sigma|)+b+1, \sigma\rangle$ when σ enters T_s for the first time and in case 1 we put the axiom each time the approximation to $K(|\sigma|)$ goes down.

Once we show that W is indeed a Kraft-Chaitin set, we are done: by Chaitin's result, there is d such that $\langle k, \sigma\rangle \in W$ implies $K(\sigma) \leq k+d$. Thus, if $A \in [T]$, then

$$K(\sigma) \leq K(|\sigma|)+b+d+1$$

for each initial segment σ of A. This is true because for any prefix σ of A, there is some stage s such that the approximation of $K(\sigma)$ and $K(\tau)$ for every $\tau \prec \sigma$ is stable.

To show that W is a Kraft-Chaitin set, define the string $D_s(\sigma)$ as the more recent $\mathbf{U}$-description found of σ by stage s. That is, when we put an axiom $\langle r+b+1, \sigma\rangle$ into W at stage s,

- let $D_s(\sigma)$ be a shortest ρ such that $\mathbf{U}_s(\rho) = \sigma$ (recall from Definition 4.2.1 that it is not required that $\mathbf{U}$ halts at stage s);

- if $\tau \prec \sigma$, we have not defined $D_s(\tau)$ yet and $D_{s-1}(\tau)$ is defined as a prefix of ρ, then let $D_s(\tau)$ be a shortest γ such that $\mathbf{U}_s(\gamma) = \tau$.

In all other cases, if $D_{s-1}(\tau)$ is defined then we let $D_s(\tau) = D_{s-1}(\tau)$.

Lemma 4.5.4. *For each s, all the strings $D_s(\tau)$ are pairwise incompatible (i.e., they form a prefix-free set).*

Proof. For contradiction, suppose that $\gamma \prec \rho$, where $\gamma = D_s(\tau)$ was defined at stage $u \leq s$, and $\rho = D_s(\sigma)$ was defined at stage $t \leq s$. Thus, $\tau = \mathbf{U}_u(\gamma)$ and $\sigma = \mathbf{U}_t(\rho)$. By the definition of monotone machines and the minimality of ρ, $u < t$ and $\tau \prec \sigma$. But then, at stage t we would have redefined $D_u(\tau)$, a contradiction. □

If we put an axiom $\langle r+b+1, \sigma\rangle$ into W at stage t, then for all $s \geq t$, $D_s(\sigma)$ is defined and has length at most $K_t(|\sigma|)+b$ (by the definition of the tree T_s). Define $\widetilde{W_s}$ as the set of axioms $\langle k_\sigma, \sigma\rangle$ in W_s where k_σ is minimal for each σ and let us call $t_\sigma \leq s$ the stage at which $\langle k_\sigma, \sigma\rangle$ enters into W_s. Thus

$$\begin{aligned}
\mathrm{wt}\left(\widetilde{W_s}\right) &= \sum_{\langle k_\sigma, \sigma\rangle \in \widetilde{W_s}} 2^{-k_\sigma} \\
&\leq \sum_{\langle k_\sigma, \sigma\rangle \in \widetilde{W_s}} 2^{-K_{t_\sigma}(|\sigma|)-b-1} \\
&\leq \sum_{\langle k_\sigma, \sigma\rangle \in \widetilde{W_s}} 2^{-|D_s(\sigma)|-1} \leq \frac{1}{2}.
\end{aligned}$$

The last inequality follows from Lemma 4.5.4. Since all axioms weigh at most twice as much as the minimal ones,

$$\mathrm{wt}\left(W_s\right) \leq 2\mathrm{wt}\left(\widetilde{W_s}\right) \leq 1$$

and W_s is a Kraft-Chaitin set for each s. Hence W is a Kraft-Chaitin set and this completes the proof. □

Corollary 4.5.5. *If $A \in 2^\omega$ is K^∞-trivial then A is K-trivial, hence in Δ^0_2.*

Proof. If A is K^∞-trivial then $K^\infty(A \restriction n) \leq K^\infty(0^n) + \mathcal{O}(1)$ and $K^\infty(0^n) \leq K(0^n) + \mathcal{O}(1)$ so we can apply Theorem 4.5.3 to conclude that A is K-trivial, and by [23] it is in Δ^0_2. □

Theorem 4.5.6. *Let $A \in 2^\omega$. A is K^∞-trivial iff A is computable.*

Proof. From right to left, it is easy to see that if A is a computable sequence then A is K^∞-trivial: we can think of a machine $\mathbf{M}$ with input σ such that $\mathbf{M}^\infty(\sigma) = A \restriction n$ when $\mathbf{U}^\infty(\sigma) = 0^n$.

For the converse, let A be K^∞-trivial via some constant b. By Corollary 4.5.5 A is Δ^0_2, hence, there is a computable approximation $(A_s)_{s\in\mathbb{N}}$ such that $\lim_{s\to\infty} A_s = A$.

Recall from the proof of Theorem 4.5.3 that

$$K^\infty_t(\sigma) = \min\{|\rho| \colon \mathbf{U}_t(\rho) = \sigma\}.$$

Consider the following program with coding constant c given by the Recursion Theorem:

```
1. set $k = 1$, $s_0 = 0$ and print 0
2. while $\exists s_k > s_{k-1}$ such that $K^\infty_{s_k}(A_{s_k} \restriction k) \leq c + b$ do
    a. print 0
    b. $k = k + 1$
```

Let us see that the above program prints out infinitely many 0's. By contradiction, suppose it writes 0^k for some k. Then, on one hand, $K^\infty(0^k) \leq c$, and on the other,

$$(\forall s > s_{k-1})\, K^\infty_s(A_s \restriction k) > c + b.$$

Also, $K^\infty_s(A_s \restriction k) = K^\infty(A \restriction k)$ for s large enough. Hence,

$$K^\infty(A \restriction k) > K^\infty(0^k) + b,$$

which contradicts the fact that A is K^∞-trivial via b.

Hence, the program enters infinitely many times into the loop and so for each k, there is some $\gamma \in 2^{<\omega}$ with $|\gamma| \leq c + b$ such that $\mathbf{U}_{s_k}(\gamma) = A_{s_k} \restriction k$. Since there are only $2^{c+b+1} - 1$ strings of length at most $c + b$, there must be at least one γ such that, for *infinitely many* k, $\mathbf{U}_{s_k}(\gamma) = A_{s_k} \restriction k$. Let us call I the set of all these ks. We show that such a γ necessarily computes A. Suppose not. Then, there is a t such that for all $s \geq t$, $\mathbf{U}_s(\gamma)$ is not an initial segment of A. Thus, noticing that $(s_k)_{k\in\mathbb{N}}$ is increasing and I is infinite, there are infinitely many $k \in I$ such that $s_k \geq t$ and $\mathbf{U}_{s_k}(\gamma) = A_{s_k} \restriction k \neq A \restriction k$. This contradicts that $A_{s_k} \restriction k \to A$ when $k \to \infty$. □

Corollary 4.5.7. *The class of K^∞-trivial sequences is strictly included in the class of K-trivial sequences.*

Proof. By Corollary 4.5.5, any K^∞-trivial sequence is also K-trivial. Solovay [73] built an K-trivial sequence in Δ^0_2 which is not computable. By Theorem 4.5.6 this sequence cannot be K^∞-trivial. □

4.6 K^∞-randomness

In this section, we define the notion of K^∞-randomness, and we prove that it coincides with the notion of randomness.

Recall from section 1.5 that $A \in 2^\omega$ is random if and only if

$$(\exists c)(\forall n)\, K(A \upharpoonright n) > n - c.$$

Levin [51] defines $A \in 2^\omega$ as *Km-random* when

$$(\exists c)(\forall n)\, Km(A \upharpoonright n) > n - c.$$

We now define K^∞-randomness in the natural way:

Definition 4.6.1 (K^∞-randomness)**.** $A \in 2^\omega$ is *K^∞-random* if and only if

$$(\exists c)(\forall n)\, K^\infty(A \upharpoonright n) > n - c.$$

Since $K^\infty \leq K$ up to an additive constant, it is clear that K^∞-randomness implies randomness. Using Levin's result [53, 51] that *Km*-randomness coincides with randomness, and the fact that *Km* gives a lower bound of K^∞, it follows immediately that the classes of K-randomness, K^∞-randomness and *Km*-random sequences coincide. For the sake of completeness we give here an alternative proof.

Proposition 4.6.2 (with D. Hirschfeldt)**.** *There is a b_0 such that for all $b \geq b_0$ and σ, if $Km(\sigma) \leq |\sigma| - b$, then there is $\tau \preceq \sigma$ such that $K(\tau) \leq |\tau| - b/2$.*

Proof. Consider the following machine $\mathbf{M}$ with coding constant c. On input $\gamma\rho$, first it simulates $\mathbf{U}(\gamma)$ until it halts. Let us call b the output of this simulation. Then it simulates $\mathbf{U}^\infty(\rho)$ till it outputs a string τ of length $b + l$ where l is the length of the prefix of ρ read by $\mathbf{U}^\infty$. Then it writes this string τ on the output and stops.

Let b_0 be the first number such that $2|b_0| + c \leq b_0/2$ and take $b \geq b_0$. Suppose $Km(\sigma) \leq |\sigma| - b$. Let ρ be a shortest program such that $\mathbf{U}^\infty(\rho) \succeq \sigma$ and let γ be a shortest program such that $\mathbf{U}(\gamma) = b$. This means that $|\rho| = Km(\sigma)$ and $|\gamma| = K(b)$. On input $\gamma\rho$, the machine $\mathbf{M}$ will compute b and then it will start simulating $\mathbf{U}^\infty(\rho)$. Since $|\sigma| \geq Km(\sigma) + b = |\rho| + b$, the machine will eventually read l bits from ρ in a way that the simulation of $\mathbf{U}^\infty(\rho \upharpoonright l) = \tau$ and $|\tau| = l + b$. When this happens, the machine $\mathbf{M}$ writes τ and stops. Then for $\rho' = \rho \upharpoonright l$, we have $\mathbf{M}(\gamma\rho')\downarrow = \tau$ and $|\tau| = |\rho'| + b$. Hence

$$\begin{aligned} K(\tau) &\leq |\gamma| + |\rho'| + c \\ &\leq K(b) + |\tau| - b + c \\ &\leq 2|b| - b + |\tau| + c \\ &\leq |\tau| - b/2. \end{aligned}$$

This completes the proof. □

Corollary 4.6.3. *$A \in 2^\omega$ is random iff A is Km-random iff A is K^∞-random.*

Proof. Since $Km \le K$ up to an additive constant, it is clear that if a sequence is Km-random then it is random. For the opposite, suppose A is random but not Km-random. Let b_0 be as in Proposition 4.6.2 and let $2c \ge b_0$ be such that $(\forall n)\, K(A \upharpoonright n) > n - c$. Since A is not Km-random,

$$(\forall d)(\exists n)\, Km(A \upharpoonright n) \le n - d.$$

In particular for $d = 2c$ there is an n such that $Km(A \upharpoonright n) \le n - 2c$. On one hand, by Proposition 4.6.2, there is a $\tau \preceq A \upharpoonright n$ such that $K(\tau) \le |\tau| - c$. On the other, since τ is a prefix of A and A is random, we have $K(\tau) > |\tau| - c$. This is a contradiction. □

Since $Km \le K^\infty \le K$ except for additive constants, the above equivalence implies A is random iff A is K^∞-random.

4.7 Oscillations of C^∞

We now drop the condition of monotone machines having a prefix-free (or closed by extensions) domain. We consider just monotone machines, where the end of the input is now delimited with a special symbol. As before, the output tape is one-way and write-only. For any such machine $\mathbf{M}$, we define the *input/output behavior of* $\mathbf{M}$ *for possibly infinite computations* in the same way as in Definition 4.2.3, that is as the output of the computation of $\mathbf{M}$, without halting states when the time goes to infinity. We define $C^\infty_{\mathbf{M}}$, the program-size complexity based on $\mathbf{M}$, in the same way as in Definition 4.3.1.

It is easy to see that an analog to Proposition 4.3.2 holds for C, C^∞ and $C^{\emptyset'}$ instead of K, K^∞ and $K^{\emptyset'}$:

Proposition 4.7.1. *For all strings $\sigma \in 2^{<\omega}$, $C^{\emptyset'}(\sigma) \le C^\infty(\sigma) + \mathcal{O}(1)$ and $C^\infty(\sigma) \le K(\sigma)$.*

First, we see that there are strings that separate the three complexity functions C, $C^{\emptyset'}$ and C^∞ arbitrarily:

Theorem 4.7.2. *For every c there is a string $\sigma \in 2^{<\omega}$ such that*

$$C^{\emptyset'}(\sigma) + c < C^\infty(\sigma) < C(\sigma) - c.$$

Proof. We know that for every n there is a string τ of length n such that $C(\tau) \ge n$. Let τ_n be the first string of length n in the lexicographic order satisfying this inequality, i.e.,

$$\tau_n = \min\{\tau \colon \tau \in 2^n \wedge C(\tau) \ge n\}.$$

Consider a machine $\mathbf{M}$ which on input i does the following:

```
1. $k = 0$
2. repeat
   a. write $k$ (in binary notation)
   b. search for a program $\rho$, $|\rho| \leq 2i$, such that $\mathbf{U}(\rho) = k$. If found, go to the next step.
   c. set $k = k + 1$
```

Now, machine $\mathbf{M}$ on input i outputs (in the limit) $0123\ldots k_i$ where $C(k_i) > 2i$ and $(\forall z \in \{0, \ldots, k_i - 1\})\, C(z) \leq 2i$. For each i, we define $\sigma_i = \tau_i 0123 \ldots k_i$. Here 0, 1, 2, 3 are, of course, written in binary.

Let us fix k and see that there is an i_1 such that $(\forall i \geq i_1)\, C^\infty(\sigma_i) - C^{\emptyset'}(\sigma_i) > k$. On the one hand, we can compute τ_i from i and a minimal program ρ such that $\mathbf{U}^\infty(\rho) = \sigma_i$ by simulating $\mathbf{U}(\rho)$ until it outputs i bits. Then

$$i \leq C(\tau_i) \leq C^\infty(\sigma_i) + 2|i| + \mathcal{O}(1). \tag{4.1}$$

On the other hand, with the help of the $\emptyset'$ oracle, we can compute σ_i from i. Hence

$$C^{\emptyset'}(\sigma_i) \leq |i| + \mathcal{O}(1). \tag{4.2}$$

From (4.1) and (4.2) we have $C^\infty(\sigma_i) - C^{\emptyset'}(\sigma_i) + \mathcal{O}(1) \geq i - 3|i|$ and then, there is an i_1 such that for all $i \geq i_1$, $C^\infty(\sigma_i) - C^{\emptyset'}(\sigma_i) > k$.

Let us see now that there is an i_2 such that $(\forall i \geq i_2)\, C(\sigma_i) - C^\infty(\sigma_i) > k$. Given i and a shortest program ρ such that $\mathbf{U}(\rho) = \sigma_i$ we construct a classical halting machine that computes k_i:

```
1. obtain $i$
2. compute $\sigma = \mathbf{U}(\rho)$
3. set $\tau = \sigma \restriction i$ and set $k = 0$
4. repeat
   a. set $\tau = \tau k$
   b. if $\tau = \sigma$ then write $k$ and halt
   c. set $k = k + 1$
```

Hence, for all i

$$2i < C(k_i) \leq C(\sigma_i) + 2|i| + \mathcal{O}(1). \tag{4.3}$$

Using the machine $\mathbf{M}$ described above we can construct a machine which, via an infinite computation, computes σ_i from a minimal program ρ such that $\mathbf{U}(\rho) = \tau_i$. Then, for every i

$$C^\infty(\sigma_i) \leq C(\tau_i) + \mathcal{O}(1) \leq i + \mathcal{O}(1). \tag{4.4}$$

From (4.3) and (4.4) we get $C(\sigma_i) - C^{\infty}(\sigma_i) + \mathcal{O}(1) > i - 2|i|$ so the difference between $C(\sigma_i)$ and $C^{\infty}(\sigma_i)$ can grow arbitrarily as we increase i. Let i_2 be such that for all $i \geq i_2$, $C(\sigma_i) - C^{\infty}(\sigma_i) > k$.

Taking $i_0 = \max\{i_1, i_2\}$, we obtain

$$C^{\emptyset'}(\sigma_i) + k < C^{\infty}(\sigma_i) < C(\sigma_i) - k$$

for all $i \geq i_0$. □

The above result shows that the three complexity functions separate themselves by an arbitrary difference. This is not always the case, since C, $C^{\emptyset'}$ and C^{∞} also get close infinitely many times.

Theorem 4.7.3. *There is a constant c such that for every n:*

$$(\exists \sigma \in 2^n)\,[\ \left|C^{\emptyset'}(\sigma) - C^{\infty}(\sigma)\right| \leq c \wedge |C^{\infty}(\sigma) - C(\sigma)| \leq c\ ].$$

Proof. Let σ_n be of length n such that $C^{\emptyset'}(\sigma_n) \geq n$. From Proposition 4.7.1 and the fact that for all σ, $C\sigma \leq |\sigma| + \mathcal{O}(1)$ (see section 1.4), there exist constants c_1 and c_2 such that

$$n \leq C^{\emptyset'}(\sigma_n) \leq C^{\infty}(\sigma_n) + c_1 \leq C(\sigma_n) + c_1 \leq n + c_1 + c_2.$$

Take $c = c_1 + c_2$. □

In the following we will use $\overline{\sigma}$ for denoting $\sigma(0)0\sigma(1)0\ldots\sigma(|\sigma|-2)0\sigma(|\sigma|-1)1$, that is the string σ interleaving 0 except in the last position, where we put a 1.

For infinitely many strings, C and C^{∞} get close but they separate from $C^{\emptyset'}$ as much as we want.

Theorem 4.7.4. *There is a constant c such that for all m*

$$(\exists \sigma \in 2^{<\omega})\,[\ C(\sigma) - C^{\emptyset'}(\sigma) > m \wedge |C^{\infty}(\sigma) - C(\sigma)| < c\ ].$$

Proof. We know that $\|\{\sigma \in 2^{n+2|n|} \colon C(\sigma) < n\}\| < 2^n$ and then

$$\|\{\sigma \in 2^{n+2|n|} \colon C(\sigma) \geq n\}\| > 2^{n+2|n|} - 2^n.$$

Let

$$S_n = \{\overline{n}\sigma \colon \sigma \in 2^n\}.$$

Clearly, $\|S_n\| = 2^n$. Assume by way of contradiction that there is an n such that

$$S_n \cap \{\sigma \in 2^{n+2|n|} \colon C(\sigma) \geq n\} = \emptyset.$$

Then

$$2^{n+2|n|} \geq \|S_n\| + \|\{\sigma \in 2^{n+2|n|} \colon C(\sigma) \geq n\}\| > 2^{n+2|n|}$$

which is impossible. For every n, let us define σ_n

$$\sigma_n = \min\{\sigma \in S_n \colon C(\sigma) \geq n\}. \tag{4.5}$$

Given a minimal program ρ such that $\mathbf{U}^\infty(\rho) = \sigma_n$, we can compute σ_n in an effective way. The idea is to take advantage of the structure of σ_n to know when $\mathbf{U}^\infty$ stops writing in its output tape: we simulate $\mathbf{U}^\infty(\rho)$ until we detect $\overline{n}$ and we continue the simulation of $\mathbf{U}^\infty$ until we see it writes exactly n more bits. Then for each n, $C(\sigma_n) \leq C^\infty(\sigma_n) + \mathcal{O}(1)$ and from Proposition 4.7.1 we have that for all n the difference $|C(\sigma_n) - C^\infty(\sigma_n)|$ is bounded by a constant.

Using the oracle $\emptyset'$, we can compute σ_n from n. Hence $C^{\emptyset'}(\sigma_n) \leq |n| + \mathcal{O}(1)$. From (4.5) we conclude $C(\sigma_n) - C^{\emptyset'}(\sigma_n) + \mathcal{O}(1) \geq n - |n|$. Thus, the difference between $C(\sigma_n)$ and $C^{\emptyset'}(\sigma_n)$ can be made arbitrarily large. □

Infinitely many times C^∞ and $C^{\emptyset'}$ get close but they separate from C arbitrarily.

Theorem 4.7.5. *There is a constant c such that for each m*

$$(\exists \sigma \in 2^{<\omega})\,[\; C(\sigma) - C^\infty(\sigma) > m \wedge \left| C^\infty(\sigma) - C^{\emptyset'}(\sigma) \right| < c \;].$$

Proof. As in the proof of Theorem 4.7.2, slightly modify the machine $\mathbf{M}$ such that on input i, it first writes $\bar{i}$ and then it continues writing k until it finds a k_i such that

$$C(k_i) > 2i \wedge (\forall z \in \{0, \ldots, k_i - 1\})\, C(z) \leq 2i.$$

Thus, given $\mathrm{str}(n)$, we can compute n and then σ_n in the limit, that is $\mathbf{M}(\mathrm{str}(n)) = \sigma_n = \overline{n}0011011\ldots k_n$. Hence for every n

$$C^\infty(\sigma_n) \leq |\mathrm{str}(n)| + \mathcal{O}(1). \tag{4.6}$$

Given a $\emptyset'$ oracle minimal program for σ_n, we can compute $\mathrm{str}(n)$ in an oracle machine. Then for every n

$$C^{\emptyset'}(\mathrm{str}(n)) \leq C^{\emptyset'}(\sigma_n) + \mathcal{O}(1). \tag{4.7}$$

We define $\chi_n = \min\{\sigma \in 2^n \colon C^{\emptyset'}(\sigma) \geq n\}$ and $\tau_n = \sigma_{\mathrm{str}^{-1}(\chi_n)}$. From (4.7) we know

$$n \leq C^{\emptyset'}(\chi_n) \leq C^{\emptyset'}(\tau_n) + \mathcal{O}(1) \tag{4.8}$$

and from (4.6) we have

$$C^\infty(\tau_n) \leq |\chi_n| + \mathcal{O}(1). \tag{4.9}$$

From (4.8) and (4.9) we obtain $C^\infty(\tau_n) - C^{\emptyset'}(\tau_n) \leq \mathcal{O}(1)$ and by Proposition 4.7.1 we conclude that for all n, $\left|C^\infty(\tau_n) - C^{\emptyset'}(\tau_n)\right| \leq \mathcal{O}(1)$. In the same way as we did in Theorem 4.7.2, we construct an effective machine that outputs k_n from a shortest program ρ such that $\mathbf{U}(\rho) = \sigma_n$, but in this case the machine gets n from the input itself (we do not need to pass it as a distinct parameter). Hence for all n,

$$2n < C(k_n) \leq C(\sigma_n) + \mathcal{O}(1)$$

and in particular for $n = \mathrm{str}^{-1}(\chi_n)$ we have $2\mathrm{str}^{-1}(\chi_n) < C(\tau_n) + \mathcal{O}(1)$. Since for each string σ, $|\sigma| \leq \mathrm{str}^{-1}(\sigma)$ we have $2|\chi_n| < C(\tau_n) + \mathcal{O}(1)$. From (4.9) and recalling that $|\chi_n| = n$, we have $C(\tau_n) - C^\infty(\tau_n) + \mathcal{O}(1) > n$. Thus, the difference between $C(\tau_n)$ and $C^\infty(\tau_n)$ grows as n increases. □

The proof of Lemma 4.4.3 applies straightforwardly to C instead of K. This already gives us that C is not sub-additive, nor smooth in the lexicographic order over $2^{<\omega}$ (to see this, follow the proofs of Propositions 4.4.4 and 4.4.5). It can be proved an analog of 4.4.6 for C^∞, but with a factor of 2.

Proposition 4.7.6. *For all n*

$$|C^\infty(\mathrm{str}(n)) - C^\infty(\mathrm{str}(n+1))| \leq 2C(|\mathrm{str}(n)|) + \mathcal{O}(1).$$

Proof. Consider the following monotone machine $\mathbf{M}$ with input $\overline{\rho}\gamma$:

1. obtain $y = \mathbf{U}(\rho)$
2. simulate $z = \mathbf{U}^\infty(\gamma)$ till it outputs y bits
3. write $\mathrm{str}(\mathrm{str}^{-1}(z) + 1)$

Let $\rho, \gamma \in 2^{<\omega}$ be such that $\mathbf{U}(\rho) = |\mathrm{str}(n)|$ and $\mathbf{U}^\infty(\gamma) = \mathrm{str}(n)$. Then, $\mathbf{M}^\infty(\overline{\rho}\gamma) = \mathrm{str}(n+1)$ and $C^\infty(\mathrm{str}(n+1)) \leq C^\infty(\mathrm{str}(n)) + 2C(|\mathrm{str}(n)|) + \mathcal{O}(1)$.

Similarly, if $\mathbf{M}$ above instead of writing $\mathrm{str}(\mathrm{str}^{-1}(z)+1)$, it writes $\mathrm{str}(\mathrm{str}^{-1}(z)-1)$, we conclude

$$C^\infty(\mathrm{str}(n)) \leq C^\infty(\mathrm{str}(n+1)) + 2C(|\mathrm{str}(n+1)|) + \mathcal{O}(1).$$

Since, $|C(\mathrm{str}(n)) - C(\mathrm{str}(n+1))| \leq \mathcal{O}(1)$, we have

$$|C^\infty(\mathrm{str}(n)) - C^\infty(\mathrm{str}(n+1))| \leq 2C(|\mathrm{str}(n)|) + \mathcal{O}(1).$$

This completes the proof. □

As in the proof of Proposition 4.4.9, it can be shown that the advantage of C^∞ over C can be seen along the initial segments of every computable sequence: if $A \in 2^\omega$ is computable then there are infinitely many ns such that $C(A \restriction n) - C^\infty(A \restriction n) > c$, for an arbitrary c.

5. LOWNESS PROPERTIES AND APPROXIMATIONS OF THE JUMP

In this chapter, we study and compare two combinatorial lowness notions: *strong jump-traceability* and *well-approximability of the jump*, by strengthening the notion of jump-traceability and super-lowness for sets of natural numbers. A computable non-decreasing unbounded function h is called an order function. Informally, a set A is strongly jump-traceable if for each order function h, for each input e one may effectively enumerate a set T_e of possible values for the jump $J^A(e)$, and the number of values enumerated is at most $h(e)$. A' is well-approximable if it can be effectively approximated with less than $h(x)$ changes at input x, for each order function h. We prove that there is a strongly jump-traceable set which is not computable, and that if A' is well-approximable then A is strongly jump-traceable. For c.e. sets, the converse holds as well. We characterize jump-traceability and the corresponding strong variant in terms of program-size complexity, and we investigate other properties of these lowness notions.

This chapter comprises the joint work [38] with André Nies and Frank Stephan.

5.1 *Introduction*

A *lowness property* of a set A says that A is computationally weak when used as an oracle, and hence A is close to being computable. In this chapter we study and compare some "combinatorial" lowness properties in the direction of characterizing K-trivial sets.

A set is K-trivial when it is highly compressible in terms of program-size complexity (see definition 1.6.1). In [62], Nies proved that a set is K-trivial if and only if A is low for random (i.e. each random set is already random relative to A).

Terwijn and Zambella [74] defined a set A to be *recursively traceable* if there is a computable bound p such that for every $f \leq_T A$, there is a computable r such that for all x, $\|D_{r(x)}\| \leq p(x)$, and $(D_{r(x)})_{x \in \mathbb{N}}$ is a set of possible values of f: for all x, we have $f(x) \in D_{r(x)}$. They showed that this combinatorial notion characterizes the sets that are low for Schnorr tests.

This property was modified in [60] to *jump-traceability.* A set A is jump-traceable if its jump at argument e, written $J^A(e) = \varphi_e^A(e)$, has few possible values.

Definition 5.1.1 (Trace)**.** A uniformly c.e. family $T = \{T_0, T_1, \ldots\}$ of sets of natural numbers is a *trace* if there is a computable function h such that $(\forall n)\, \|T_n\| \leq h(n)$. We say that h is a *bound* for T. The set A is *jump-traceable* if there is a trace T

such that

$$(\forall e)\,[J^A(e)\downarrow \Rightarrow\ J^A(e) \in T_e].$$

We say that A is jump-traceable *via a function* h if, additionally, T has bound h.

Another notion studied in [60] is *super-lowness*, first introduced in [15, 59].

Definition 5.1.2 (ω-c.e. and super-low)**.** A set A is *ω-c.e.* iff there exists a computable function b such that $A(x) = \lim_{s\to\infty} g(x,s)$ for a computable $\{0,1\}$-valued g such that $g(x,s)$ changes at most $b(x)$ times, i.e.

$$\|\{s\colon g(x,s) \neq g(x,s+1)\}\| \leq b(x).$$

In this case, we say that A is ω-c.e. *via the function* g and *bound* b. A is *super-low* iff A' is ω-c.e.

Recall that a set A is *low* if $A' \leq_T \emptyset'$. The above definition of A being super-low is equivalent to $A' \leq_{tt} \emptyset'$. Hence super-lowness, clearly implies lowness.

Both jump-traceable and super-low sets are closed downward under Turing reducibility and imply being generalized low (i.e. $A' \leq A \oplus \emptyset'$). In [60] it was proved that these two lowness notions coincide within the c.e. sets but that none of them implies the other within the ω-c.e. sets.

In this chapter, we define the notions of *strong jump-traceability* and *well-approximability of the jump*, strengthening the notions of super-lowness and ω-c.e. respectively. A special emphasis is given to the case where the jump of A is ω-c.e. In the strong variant of these notions consider *all* order functions as the bound instead of just *some* computable bound. Here, an *order* function is a slowly growing but unbounded computable function. Our first two results are:

- There is a non-computable strongly jump-traceable set;
- If A' is well-approximable then A is strongly jump-traceable. The converse also holds, if A is c.e.

Our approach is used to study interesting lowness properties related to plain and prefix-free program-size complexity. We investigate the properties of sets A such that the program-size complexity relative to A is only a bit smaller than the unrelativized one. We prove some characterizations of jump-traceability and its strong variant in terms of prefix-free and plain program-size complexity, respectively:

- A is jump-traceable if and only if there is a computable p, growing faster than linearly such that $K(\sigma)$ is bounded by $p(K^A(\sigma)+c_0)+c_1$, for some constants c_0 and c_1;
- A is strongly jump-traceable if and only if $C(\sigma) - C^A(\sigma)$ is bounded by $h(C^A(\sigma))$, for every order function h and almost all σ.

Recall that A is low for K iff $K(\sigma) \leq K^A(\sigma) + \mathcal{O}(1)$ for each $\sigma \in 2^{<\omega}$. Nies [62] showed that this property is equivalent to being K-trivial. In particular, non-computable low for K sets exist. The corresponding property involving C is only satisfied by the computable sets (because it implies being C-trivial and by [24] this is the same as computable). The characterization of strong jump-traceability is via a property that states that C^A is very close to C, while not implying computability.

A notion equivalent to jump-traceability was introduced in [62] (refer to Proposition 5.2.14 for a proof):

Definition 5.1.3 (**U**-traceable)**.** A is **U**-*traceable* if there is a trace T such that

$$(\forall \sigma)\,[\mathbf{U}^A(\sigma)\downarrow \;\Rightarrow\; \mathbf{U}^A(\sigma) \in T_{|\sigma|}].$$

Using this notion of **U**-traceability, in [62] is was shown that K-triviality implies jump-traceability and super-lowness, but it is unknown whether K-triviality implies strong jump-traceability. The reverse direction is also open.

5.2 *Strong jump-traceability*

In this section, we introduce a stronger version of jump-traceability and we prove that there is a promptly simple (hence simple and then non-computable) strongly jump-traceable set. We also prove that there is no maximal order function as bound for jump-traceability.

Definition 5.2.1 (Order function)**.** A computable function $h\colon \mathbb{N} \to \mathbb{N}^+$ is an *order function* if h is non-decreasing and unbounded.

Notice that any reduction function is an order function.

Definition 5.2.2 (Strong jump-traceability)**.** A set A is *strongly jump-traceable* iff for each order function h, A is jump-traceable via h.

Clearly, strong jump-traceability implies jump-traceability and it is not difficult to see that strong jump-traceability is closed downward under Turing reducibility.

Proposition 5.2.3. *The set $\{A\colon A$ is strongly jump-traceable$\}$ is closed downward under Turing reducibility.*

Proof. Suppose A is strongly jump-traceable and $B \leq_T A$. We prove that B is jump-traceable via the given order function h. Let ψ be the function computable by an oracle machine such that $\psi^A(x) = J^B(x)$ for all x and let α be the reduction function such that $J^A(\alpha(x)) = \psi^A(x)$. We know that A is jump-traceable via a trace $(T_i)_{i\in\mathbb{N}}$ with bound $\tilde{h}$, where

$$\tilde{h}(z) = h(\min\{y\colon y \in \mathbb{N} \wedge \alpha(y+1) \geq z\}).$$

Observe that, since α is an order function, $\tilde{h}$ also is. Clearly,

$$J^B(e)\downarrow \;\Rightarrow\; J^B(e) \in T_{\alpha(e)}.$$

Now, $\tilde{h}(\alpha(e)) = h(y)$ for some y such that $\alpha(y) < \alpha(e)$ or $y = 0$. Then $y \leq e$ and $\tilde{h}(\alpha(e)) = h(y) \leq h(e)$. Hence $(T_{\alpha(i)})_{i\in\mathbb{N}}$ is a trace for the jump of B with bound h. □

Clearly each computable set A is strongly jump-traceable, because we can trace the jump by

$$T_e = \begin{cases} \{J^A(e)\} & \text{if } J^A(e) \downarrow; \\ \emptyset & \text{otherwise.} \end{cases}$$

In Theorem 5.2.5 below we show the existence of a non-computable strongly jump-traceable set. We need the following result, proven in [54, Theorem 2.3.1]:

Lemma 5.2.4. *The function*

$$m(x) = \min\{C(y) \colon y \geq x\}$$

is unbounded, non-decreasing and for every order function f there is an x_0 such that $m(x) < f(x)$ for all $x \geq x_0$. Also, $m(x) = \lim_{s \to \infty} m_s(x)$, where

$$m_s(x) = m(s, x) = \min\{C_s(y) \colon s \geq y \geq x \vee y = 0\}$$

is computable and $m_s(x) \geq m_{s+1}(x)$, for all x and s.

Recall from section 1.4 that here $\lambda x, s.C_s(x)$ is the standard computable approximation from above of $C(x)$ (that is $\lambda s.C_s(x) \to C(x)$ when $s \to \infty$ and $C_s(x) \geq C_{s+1}(x)$).

A c.e. set A is *promptly simple* [71] if A is co-infinite and there is a computable function p and an effective approximation $(A_s)_{s \in \mathbb{N}}$ of A such that, for each e,

$$\|W_e\| = \infty \;\Rightarrow\; (\exists s)(\exists x)\,[x \in W_{e,s+1} \setminus W_{e,s} \;\wedge\; x \in A_{p(s)}].$$

Theorem 5.2.5. *There exists a promptly simple strongly jump-traceable set.*

Proof. We construct a promptly simple set A in stages satisfying the following positive requirements

$$P_e \;:\; \|W_e\| = \infty \;\Rightarrow\; (\exists s)(\exists x)\,[x \in W_{e,s+1} \setminus W_{e,s} \;\wedge\; x \in A_{s+1}].$$

These requirements will ensure that A is promptly simple. Each time we enumerate an element into A in order to satisfy P_e, we may destroy $J^A(k)$, and then our trace for the jump of A will grow. Hence, we must enumerate elements into A in a controlled way, and sometimes we should restrain from putting elements into A. Formally, we also want to satisfy the following negative requirements:

$$N_k \;:\; [(\exists^\infty s)\, J^A(k)[s] \downarrow] \;\Rightarrow\; J^A(k) \downarrow .$$

(recall the definition of $J^A(k)[s]$ from section 1.3). Since for any order function h there has to be a trace for J^A bounded by h, we will work with the function m defined in Lemma 5.2.4, which grows slower than any order function. The rule will be that during the construction, P_e may destroy $J^A(k)$ at stage s only if $e < m_s(k)$. (Observe that the restriction on P_e imposed rule may strengthen as s grows, because we may have $m_s(k) > m_{s+1}(k)$.) In this way, we will guarantee that the size of our trace for $J^A(e)$ will be bounded by $m(e)$, which will suffice because $m \leq h$ from some point on. As we will see, the exact choice of the trace for J^A with bound h depends on h, and is made in a non-uniform way.

Construction of A. Let m_s be the non-decreasing, unbounded function defined in Lemma 5.2.4.

Stage 0: set $A_0 = \emptyset$ and declare P_e unsatisfied for all e.

Stage $s+1$: choose the least $e \leq s$ such that

- P_e yet not satisfied;
- There exists x such that $x \in W_{e,s+1} \setminus W_{e,s}$, $x > 2e$ and for all k such that $m_s(k) \leq e$, if $J^A(k)[s]$ is defined then x is greater than the use of $J^A(k)[s]$.

If such e exists, put least such x for e into A_{s+1}. We say that P_e *receives attention* at stage $s+1$, and declare P_e satisfied. Otherwise, $A_{s+1} = A_s$. Finally, define $A = \bigcup_s A_s$. We say that x *injures* N_e at stage $s+1$ if x enters A_{s+1}, $J^A(e)[s]$ is defined and x is below its use.

We next verify that the described construction is as desired.

Lemma 5.2.6. *For all k, N_k is injured at most finitely often.*

Proof. Clearly, P_e receives attention at most once. So every positive requirement influences the enumeration of A at most once. Since N_k can be injured at stage s only if $e < m_s(k)$ then N_k is injured at most $m_0(k)$ times. □

Lemma 5.2.7. *For all k, N_k is met.*

Proof. By Lemma 5.2.6, choose s such that N_k is not injured after stage s. If $J^A(k)[t]\downarrow$ for some $t \geq s$, then this computation is stable from stage t on. □

Lemma 5.2.8. *For all e, P_e is met.*

Proof. Fix e such that W_e is infinite and let us see that P_e is met. Let s be such that

$$(\forall k)\,[m(k) \leq e \;\Rightarrow\; m_s(k) = m(k)]$$

and $s' > s$ such that no P_i receives attention after stage s' for any $i < e$. Then, by the construction, no computation N_k with $m(k) \leq e$ can be injured after stage s'. So there is $t > s'$ such that for all k where $m_t(k) \leq e$, if $J^A(k)$ converges then the computation is stable from stage t on. Choose $t' \geq t$ such that there is $x \in W_{e,t'+1} \setminus W_{e,t'}$, $x > 2e$ and x is greater than the use of all converging $J^A(k)$ for all k where $m_{t'}(k) \leq e$. Now either P_e was already satisfied or P_e receives attention at stage $t'+1$. In either case P_e is met. □

Lemma 5.2.9. *A is strongly jump-traceable.*

Proof. Fix an order function h. We will prove that there exists a trace T for J^A as in Definition 5.1.1. Let h be any order function. By Lemma 5.2.4, there exists k_0 such that for all $k \geq k_0$, $m(k) \leq h(k)$. Define the computable function

$$f(k) = \begin{cases} \min\{s\colon m_s(k) \leq h(k)\} & \text{if } k \geq k_0; \\ 0 & \text{otherwise.} \end{cases}$$

For $k \geq k_0$ and $s \geq f(k)$, $m_s(k)$ will be below $h(k)$, so $J^A(k)$ may change because P_e receives attention, for $e < m_s(k) \leq h(k)$. Since each P_e receives attention at most once, $J^A(k)$ can change at most $h(k)$ times after stage $f(k)$. So

$$T_k = \begin{cases} \{J^A(k)[s]\colon J^A(k)[s]\downarrow \ \wedge\ s \geq f(k)\} & \text{if } k \geq k_0; \\ \{J^A(k)\} & \text{if } J^A(k)\downarrow \ \wedge\ k < k_0; \\ \emptyset & \text{otherwise.} \end{cases}$$

is as required. □

This completes the proof of the whole result. □

Next we study the size of the trace bound for jump-traceability. Given an order function h, it is always possible to find a jump-traceable set A for which h is too small to be a bound for any trace for the jump of A.

We use the following auxiliary Lemma:

Lemma 5.2.10. *Let h be an order. There exists a computable enumeration of all the traces with bound h.*

Proof. We know that there is a computable enumeration $\tilde{T}(0), \tilde{T}(1), \dots$ of all the traces. Define the i-th trace with bound h, denoted

$$T(i) = \{T(i)_0, T(i)_1, \dots\},$$

as in Definition 5.1.1, in stages: $T(i)_e[0] = \emptyset$ and

$$T(i)_e[s+1] = \begin{cases} \tilde{T}(i)_e[s] & \text{if } \|\tilde{T}(i)_e[s]\| \leq h(e); \\ T(i)_e[s] & \text{otherwise.} \end{cases}$$

where $\tilde{T}(i)_e[s]$ is the approximation of $\tilde{T}(i)_e$ by stage s. □

Theorem 5.2.11. *For any order function h there is a c.e. set A and an order function $\tilde{h}$ such that A is jump-traceable via $\tilde{h}$ but not via h.*

Proof. We will define an auxiliary function computable by n oracle machine ψ and we use α, the reduction function for ψ (i.e. $\psi^X(e) = J^X(\alpha(e))$ for all X and e), in advance by the Recursion Theorem (see [71]). At the same time, we will define a c.e. set A and a trace $\tilde{T}$ for J^A. Finally, we will verify that there is an order function $\tilde{h}$ for $\tilde{T}$ with the desired property.

Following Lemma 5.2.10, let $T(0), T(1), \dots$ be an enumeration of all the traces with bound h, so that

$$T(e) = \{T(e)_0, T(e)_1, \dots\},$$

the e-th such trace, is as in Definition 5.1.1. Positive requirement P_e tries to show that J^A is not traceable via the trace $T(e)$ with bound h, that is,

$$P_e \quad : \quad (\exists x)\, \psi^A(x) \notin T(e)_{\alpha(x)}$$

and negative requirement N_e tries to stabilize the jump when it becomes defined, i.e.

$$N_e \quad : \quad [(\exists^\infty s)\, J^A(e)[s]\downarrow] \;\Rightarrow\; J^A(e)\downarrow.$$

The strategy for a single procedure P_e consists of an initial action and a possible later action.

Initial action at stage $s+1$:

- Choose a new candidate $x_e = \langle e, n\rangle$, where n is the number of times that P_e has been initialized. Define $\psi^A(x_e)[s+1] = 0$ with large use (that is, with use greater than all the seen computations).

Action at stage $s+1$:

- Let $x_e = \langle e, n\rangle$ be the current candidate. Put y into A_{s+1}, where y is the use of the defined $\psi^A(x_e)[s]$. Notice that the construction will guarantee that this action will not affect $J^A(i)[s]$ for $i < e$ because of the choice of y;
- Define $\psi^A(x_e)[s+1] = \psi^A(x_e)[s] + 1$ with use $y' > y$ and greater than the use of all defined computations of $J^A(i)[s+1]$ for all $i < e$.

We say that P_e *requires attention* at stage $s+1$ if $\psi^A(x_e)[s] \in T(e)_{\alpha(x_e)}[s]$ and we say that N_e *requires attention* at stage $s+1$ if $J^A(e)[s]$ becomes defined for the first time.

Construction of A. We define $\tilde{T} = \{\tilde{T}_0, \tilde{T}_1, \dots\}$ by stages. The s-th stage of $\tilde{T}_i$ will be denoted by $\tilde{T}_i[s]$. We start with $A_0 = \emptyset$ and $\tilde{T}_i[0] = \emptyset$ for all i. At stage $s+1$ we consider the procedures N_j for $j \le s$ and P_j for $j < s$. We also initialize the new P_s. We look at the least procedure requiring attention in the order

$$P_0, N_0, \dots, P_s, N_s.$$

If there is no one, do nothing. Otherwise, suppose P_e is the first one. We let P_e take action at $s+1$, changing A below the use of $\psi^A(x_e)[s]$ and redefining $\psi^A(x_e)[s+1]$ without affecting N_i for $i < e$. We keep the other computations of P_j with the new definition of A, for $j \neq i$ and large use. If N_e is the least procedure requiring attention, there is y such that $J^A(e)[s]\downarrow = y$. We put y into $\tilde{T}_e[s+1]$ and initialize P_j for $e < j \le s$. In this case, we say that N_e *acts*. Notice that it is necessary to initialize P_j for $e < j \leq s$ because the use of the defined $J^A(e)[s]$ might by greater than the use of $\psi^A(x_j)[s]$ and it is required that the computation of $\psi^A(x_j)[s]$ has always an use greater than the computation of $J^A(e)[s]$.

We say that N_e is *injured* at stage $s+1$ if we put y into A_{s+1} and y is less or equal than the use of $J^A(e)[s]$.

Lemma 5.2.12. *N_e is injured finitely often and there is a computable bound $c_N(e)$ for such number of injuries. Furthermore, $\tilde{h}(e) = c_N(e)$ is an order function and it constitutes a bound for the trace $(\tilde{T}_i)_{i\in\mathbb{N}}$.*

Proof. We define $c_P(k)$ as a bound for the number of initializations of P_r, for $r \leq k$; and define $c_N(k)$ as a bound for the number of injuries to N_r, for $r \leq k$. Since P_0 is initialized just once and makes at most $h(\langle 0,0\rangle)$ changes in A, $c_P(0) = 1$ and $c_N(0) = h(\langle 0,0\rangle)$. The number of times that P_{k+1} is initialized is bounded by the number of times that N_r acts, for $r \leq k$, so

$$c_P(k+1) = c_P(k) + c_N(k).$$

Each time N_r is injured, for $r \leq k$ then N_{k+1} may also be injured; additionally, N_{k+1} may be injured each time P_{k+1} changes A. The latter occurs at most $h(\langle k+1, i\rangle)$ for the i-th initialization of P_{k+1}. Hence

$$c_N(k+1) = 2c_N(k) + \sum_{i \leq c_P(k+1)} h(\langle k+1, i\rangle).$$

Once N_e is not injured anymore, if $J^A(e)\downarrow$ then $J^A(e) \in \tilde{T}_e$. Since the number of changes of $J^A(e)$ is at most the number of injuries to N_e, we define the function $\tilde{h}(e) = c_N(e)$ which is clearly an order function and it constitutes a bound for the trace $(\tilde{T}_i)_{i\in\mathbb{N}}$. □

Lemma 5.2.13. *For all e, P_e is met.*

Proof. Take s such that all $J^A(i)$ are stable for $i < e$. Suppose x_e is the actual candidate of P_e. Since P_e is not going to be initialized again, x_e is the last candidate it picks. Each time $\psi^A(x_e)[t] \in T(e)_{\alpha(x_e)}[t]$ for $t > s$, P_e acts and changes the definition of $\psi^A(x_e)$ to escape from $T(e)_{\alpha(x_e)}$. Since $\|T(e)_{\alpha(x_e)}\| \leq h(\alpha(x_e))$, there is $s' > s$ such that $T(e)_{\alpha(x_e)}[s'] = T(e)_{\alpha(x_e)}$. By construction, $\psi^A(x_e)[s'+1] \notin T(e)_{\alpha(x_e)}$ and $\psi^A(x_e)[s'+1]$ is stable. □

This completes the proof of the whole result. □

It is open if there is a *minimal* bound for jump-traceability. That is, given an order function h, is there a set A and an order function $\tilde{h}$ such that A is jump-traceable via h but not via $\tilde{h}$. If this fails for some order function h, then strong jump-traceability is the same as jump traceability for that single order function.

To finish this section, we present the proof of the fact that, because of the universality of the jump, the definition of **U**-traceability actually coincides with the definition of jump-traceability.

Proposition 5.2.14. *A is $\mathbf{U}$-traceable if and only if A is jump-traceable.*

Proof. Recall Definition 5.1.3 of $\mathbf{U}$-traceability. Suppose A is $\mathbf{U}$-traceable via a trace T with bound h and let α be a reduction function such that $\mathbf{U}^A(\alpha(x)) = J^A(x)$. Then if $J^A(x)\downarrow$ then $J^A(x) \in T_{|\alpha(x)|}$. So by taking $\tilde{T}_i = T_{|\alpha(i)|}$ we have that A is jump-traceable via a trace $\tilde{T}$ with bound $h(|\alpha(i)|)$.

For the other direction, if A is jump-traceable via T with bound h, then there is a reduction function α such that $J^A(\alpha(x)) = \mathrm{str}^{-1}(\mathbf{U}^A(\mathrm{str}(x)))$. Hence if $\mathbf{U}^A(\sigma)\downarrow$, and $x = \mathrm{str}^{-1}(\sigma)$ then $\mathbf{U}^A(\mathrm{str}(x)) = \mathbf{U}^A(\sigma)$ and $\mathrm{str}^{-1}(\mathbf{U}^A(\sigma)) \in T_{\alpha(x)}$. Hence

$$\mathbf{U}^A(\sigma) \in \mathrm{str}\left(T_{\alpha(\mathrm{str}^{-1}(\sigma))}\right) \subseteq \bigcup_{\tau\colon |\tau|=|\sigma|} \mathrm{str}\left(T_{\alpha(\mathrm{str}^{-1}(\tau))}\right)$$

and A is $\mathbf{U}$-traceable via

$$\tilde{T}_i = \bigcup_{\tau\colon |\tau|=i} \mathrm{str}\left(T_{\alpha(\mathrm{str}^{-1}(\tau))}\right)$$

with bound

$$\sum_{\tau\colon |\tau|=i} h(\alpha(\mathrm{str}^{-1}(\tau))).$$

This completes the proof. □

5.3 Well-approximability of the jump

We strengthen the notion of super-lowness and study the relationship to strongly jump-traceability.

Definition 5.3.1 (Well-approximability)**.** A set D is *well-approximable* if and only if for each order function b, D is ω-c.e. via b.

Clearly, if A' is well-approximable, then A is super-low. It is not difficult to see that well-approximability of the jump is closed downward under Turing reducibility.

Proposition 5.3.2. *The set $\{A\colon A' \text{ is well approximable}\}$ is closed downward under Turing reducibility.*

Proof. Suppose A is such that A' is well-approximable, and let $B \leq_T A$. We prove that B' is well-approximable via the given order function b. Define ψ and α as in Proposition 5.2.3. We know that there is a computable $\{0,1\}$-valued g such that $A'(x) = \lim_{s\to\infty} g(x,s)$ and $g(x,s)$ changes at most $\tilde{b}(x)$ times, where

$$\tilde{b}(z) = b(\min\{y\colon y \in \mathbb{N} \wedge \alpha(y+1) \geq z\}).$$

Then

$$\lim_{s\to\infty} g(\alpha(x), s) = A'(\alpha(x)) = B'(x)$$

and $g(\alpha(x), s)$ changes at most $\tilde{b}(\alpha(x))$ times. As in Proposition 5.2.3, $\tilde{b}(\alpha(x)) \leq b(x)$. □

We next prove that if A is c.e. then A is strongly jump-traceable if and only if A' is well-approximable. We first need the following lemmas.

Lemma 5.3.3. *Let f and $\hat{f}$ be order functions such that $f(x) \leq \hat{f}(x)$ for almost all x.*

1. *If A is jump-traceable via f then A is jump-traceable via $\hat{f}$;*
2. *If A is well-approximable via f then A is well-approximable via $\hat{f}$.*

Proof. Assume

$$(\exists x_0)(\forall x)\,[x \geq x_0 \ \Rightarrow\ f(x) \leq \hat{f}(x)].$$

For (1), suppose T is a trace for J^A with bound f. We can define the trace $\hat{T}$:

$$\hat{T}_x = \begin{cases} T_x & \text{if } x \geq x_0; \\ \{J^A(x)\} & \text{otherwise.} \end{cases}$$

Hence, if $x \geq x_0$ then $\|\hat{T}_x\| = \|T_x\| \leq f(x) \leq \hat{f}(x)$, and if $x < x_0$ then $1 = \|\hat{T}_x\| \leq \hat{f}(x)$.

For (2), suppose A is well-approximable via the $\{0,1\}$-valued $g(x,s)$ which changes at most $f(x)$ times. Define

$$\hat{g}(x,s) = \begin{cases} g(x,s) & \text{if } x \geq x_0; \\ A(x) & \text{otherwise.} \end{cases}$$

If $x \geq x_0$ then $\hat{g}(x,s)$ changes at most $f(x) \leq \hat{f}(x)$ times, and if $x < x_0$ then $\hat{g}$ does not change at all. □

Lemma 5.3.4. *There exists a computable f such that for all c.e. A:*

1. *If A is jump-traceable via an order function h then A is super-low via the order function $b(x) = 2h(f(x)) + 2$;*
2. *If A is super-low via an order function b then A is jump-traceable via the order function $h(x) = \lfloor \frac{1}{2} b(f(x)) \rfloor$.*

Proof. We follow the proof of [60, Theorem 4.1], together with Lemma 5.3.3.

1⇒2. Suppose A is jump-traceable via h. By [60] A is super-low via a $\{0,1\}$-valued computable g such that $g(x,s)$ changes at most $2h(\alpha(x)) + 2$ times. Here, α is a reduction function (hence primitive recursive) which depends on A. The diagonal f of the Ackermann-function satisfies $f(x) \geq \alpha(x)$ for almost all x [63, Volume 2, Theorem VIII.8.10]. Since h is an order function, $2(h \circ f) + 2$ also is, and $2h(f(x)) + 2 \geq 2h(\alpha(x)) + 2$ for almost all x. By Lemma 5.3.3, A is super-low via $b(x) = 2h(f(x)) + 2$.

2⇒1. Suppose A is super-low via an order function b and the $\{0,1\}$-valued function g. Again following [60], there is a trace for J^A via $\lfloor \frac{1}{2}(b \circ f) \rfloor$, for a primitive recursive α which depends on g. As we did in the previous implication, $\lfloor \frac{1}{2} b(f(x)) \rfloor \geq \lfloor \frac{1}{2} b(\alpha(x)) \rfloor$ for almost all x. Thus A is jump-traceable via $h(x) = \lfloor \frac{1}{2} b(f(x)) \rfloor$. □

Theorem 5.3.5. *Let A be a c.e. set. Then the following are equivalent:*

1. *A is strongly jump-traceable;*
2. *A' is well-approximable.*

Proof. 1⇒2. Given an order function b, let us prove that A is super-low via b. By part 1 of Lemma 5.3.4, it suffices to define an order function h such that $2h(f(x))+2 \leq b(x)$ for almost all x. If $b(x) \geq 4$ then define $h(f(x)) = \lfloor \frac{b(x)-2}{2} \rfloor$ and if $b(x) < 4$, define $h(f(x)) = 1$. Since f can be taken strictly monotone, the above definition is correct and we can complete it to make h an order function.

2⇒1. Given an order function h, we will prove that A is jump-traceable via h. By part 2 of Lemma 5.3.4, it suffices to define an order function b such that $\lfloor \frac{1}{2} b(f(x)) \rfloor \leq h(x)$ for almost all x. The argument is similar to the previous case. □

Later, in Corollary 5.4.4, we will improve this result and we will see that, in fact, the implication 2⇒1 holds for any A.

We finish this section by proving that the prefixes $D \upharpoonright n$ of a well-approximable set D have low program-size complexity, of order logarithmic in n. Hence D is not Martin-Löf random and furthermore, the effective Hausdorff dimension is 0. The latter is just equivalent of saying that there is no $c > 0$ such that cn is a linear lower bound for the prefix-free program-size complexity of $D \upharpoonright n$ for almost all n.

Theorem 5.3.6. *If D is well-approximable then for almost all n, $K(D \upharpoonright n) \leq 4|n|$.*

Proof. Suppose $D(n) = \lim_{s\to\infty} g(n, s)$, where g is computable and changes at most n times. Given n, there is a unique s and some $m < n$ such that $g(m, s) \neq g(m+1, s)$ but $g(q, t) = g(q, t+1)$ for all $t > s$ and $q < n$. That is, s is the time when g converges on below n and m is the place where the last change takes place. The stage s can be computed from m and the number k of stages with $g(m, t+1) \neq g(m, t)$. So one can compute $D \upharpoonright n$ from m, n, k. Since $k, m \leq n$, one can, for almost all n, code m, n, k in a prefix-free way in $4|n|$ many bits. This is done by using a prefix of the form 1^q0 followed by $2q$ bits representing n, $2q$ bits representing m and $2q$ bits representing k as binary numbers; here q is just the smallest number such that $2q$ bits are enough. Since $k, m \leq n$ and since $2q \leq |n| + c$ for some constant c and since the additionally necessary coding needed to transform the above representation into a program for $\mathbf{U}$ is bounded by a constant, we have that there is a constant d such that

$$(\forall n)\, K(D \upharpoonright n) \leq 3|n| + |n|/2 + d$$

and then the relation $K(D \upharpoonright n) \leq 4|n|$ holds for almost all n. In fact, using binary notation to store q instead of 1^q0, it would even give

$$K(D \upharpoonright n) \leq 3|n| + 2\log(|n|)$$

for almost all n. □

We can prove a similar result for strongly jump-traceable sets:

Theorem 5.3.7. *If A is strongly jump-traceable then for almost all n, $K(A \upharpoonright n) \leq 3|n|$.*

Proof. Let α be a reduction function such that $J^A(\alpha(n)) = A \upharpoonright n$ and let T be a trace for J^A with bound h such that $h(\alpha(n)) \leq n$. We know that $A \upharpoonright n \in T_{\alpha(n)}$ and $\|T_{\alpha(n)}\| \leq n$. Suppose $A \upharpoonright n$ is the m-th element enumerated into $T_{\alpha(n)}$. Then we can describe $A \upharpoonright n$ with m and n and so

$$K(A \upharpoonright n) \leq |n| + 2\log|n| + |m| + 2\log|n| + c$$

for some constant c. Since $m \leq n$ we finally have $K(A \upharpoonright n) \leq 2|n| + 4\log|n| + c$, which implies $K(A \upharpoonright n) \leq 3|n|$ for almost all n. □

Corollary 5.3.8. *If A is strongly jump-traceable or A' is well-approximable then $K(A \upharpoonright n) \leq 4|n|$ for almost all n and hence A is not random.*

Proof. It follows immediately from Theorem 5.3.7 and Theorem 5.3.6 together with the fact that $A \leq_1 A'$ and Proposition 5.3.2. □

5.4 Traceability and plain program-size complexity

We give a characterization of strong jump-traceability in terms of plain program-size complexity and we show that if A' is well-approximable then A is strongly jump-traceable for any set A. This is a stronger result than the implication 2⇒1 of Theorem 5.3.5.

A *binary machine* is a partial computable function $\tilde{\mathbf{M}}\colon 2^{<\omega} \times 2^{<\omega} \mapsto 2^{<\omega}$. Let $\tilde{\mathbf{U}}$ be a binary universal function given as

$$\tilde{\mathbf{U}}(0^d 1\sigma, \tau) = \tilde{\mathbf{M}}_d(\sigma, \tau),$$

where $(\tilde{\mathbf{M}}_d)_{d\in\mathbb{N}}$ is an enumeration of all partial computable functions of two arguments. We define the plain conditional program-size complexity $C(\sigma|\tau)$ as the length of the shortest description of σ using $\tilde{\mathbf{U}}$ with string τ as the second argument, that is,

$$C(\sigma|\tau) = \min\{|\rho|\colon \tilde{\mathbf{U}}(\rho, \tau) = \sigma\}.$$

Theorem 5.4.1. *If A' is well-approximable then for every order function h and almost all σ, $C(\sigma) \leq C^A(\sigma) + h(C^A(\sigma))$.*

Proof. The idea of the proof is the following. Let h be any order function. Suppose ρ_σ is a minimal A-program for σ. It is known (see for example [54]) that there is a c such that

$$C(\sigma) \leq |\rho_\sigma| + 2C(\sigma|\rho_\sigma) + c.$$

Since $|\rho_\sigma| = C^A(\sigma)$, we only need to show that

$$2C(\sigma|\rho_\sigma) + c \leq h(|\rho_\sigma|)$$

for almost all σ to obtain the desired upper bound on $C(\sigma)$. Given ρ_σ and the value of $C(\sigma|\rho_\sigma)$, we can find a program γ_σ of length $C(\sigma|\rho_\sigma)$ which describes σ with the help of ρ_σ, that is $\tilde{\mathbf{U}}(\gamma_\sigma, \rho_\sigma) = \sigma$. It can be shown that there is a computable $\{0,1\}$-valued approximation of the bits of γ_σ which changes few times (in the proof, this is done with the help of the function ψ, computable with an oracle machine). Hence, σ can be described by ρ_σ and γ_σ, while γ_σ can be represented with $C(\sigma|\rho_\sigma)$ and the number of changes of the mentioned $\{0,1\}$-valued approximation. Therefore, in total we can represent σ with ρ_σ, $C(\sigma|\rho_\sigma)$ and the appropriate number of changes. This will show $C(\sigma|\rho_\sigma) \leq 2\log h(|\rho_\sigma|) + \mathcal{O}(1)$, which is sufficient to get the desired upper bound on $2C(\sigma|\rho_\sigma) + c$.

Here are the details. Let $\psi^A(m,n,\rho)$ be a function computable by an oracle machine which does the following:

1. Compute $\sigma = \mathbf{U}^A(\rho)$. If $\mathbf{U}^A(\rho)\uparrow$ then $\psi^A(m,n,\rho)\uparrow$;
2. Find the first program γ such that $|\gamma| = n$ and $\tilde{\mathbf{U}}(\gamma,\rho) = \sigma$. If there is no such γ then $\psi^A(m,n,\rho)\uparrow$;
3. In case $m \notin [1,n]$ then $\psi^A(m,n,\rho)\uparrow$. Otherwise, if the m-th bit of γ is 1 then $\psi^A(m,n,\rho)\downarrow$, else $\psi^A(m,n,\rho)\uparrow$.

Let α be a reduction function such that $J^A(\alpha(m,n,\rho)) = \psi^A(m,n,\rho)$. Choose an order b such that $b(\alpha(n,n,\rho)) \leq nh(|\rho|)$ for all n, ρ. We can approximate $A'(x)$ with a $\{0,1\}$-valued computable function which changes at most $b(x)$ times.

Let ρ_σ be a minimal A-program for σ, that is, $\mathbf{U}^A(\rho_\sigma) = \sigma$ and $|\rho_\sigma| = C^A(\sigma)$. Let $n_\sigma = C(\sigma|\rho_\sigma)$. Then by construction, $\psi^A(m, n_\sigma, \rho_\sigma)\downarrow$ iff the m-th bit of γ_σ is 1, where γ_σ is the first program such that $|\gamma_\sigma| = n_\sigma$ and $\tilde{\mathbf{U}}(\gamma_\sigma, \rho_\sigma) = \sigma$.

Since A' is ω-c.e. via b,

$$\gamma_\sigma = A'(\alpha(1, n_\sigma, \rho_\sigma)) \ldots A'(\alpha(n_\sigma, n_\sigma, \rho_\sigma))$$

changes at most

$$\begin{aligned} n_\sigma \max\{b(\alpha(m, n_\sigma, \rho_\sigma)) \colon 1 \leq m \leq n_\sigma\} &\leq n_\sigma b(\alpha(n_\sigma, n_\sigma, \rho_\sigma)) \\ &\leq n_\sigma^2 h(|\rho_\sigma|) \end{aligned}$$

many times. Since $\tilde{\mathbf{U}}(\gamma_\sigma, \rho_\sigma) = \sigma$ and we can describe γ_σ with n_σ, ρ_σ and the number of changes of $A'(\alpha(1, n_\sigma, \rho_\sigma)) \ldots A'(\alpha(n_\sigma, n_\sigma, \rho_\sigma))$, we have

$$\begin{aligned} n_\sigma = C(\sigma \mid \rho_\sigma) &\leq 2\log n_\sigma + \log(n_\sigma^2 h(|\rho_\sigma|)) + \mathcal{O}(1) \\ &\leq 4\log n_\sigma + \log h(|\rho_\sigma|) + \mathcal{O}(1). \end{aligned} \tag{5.1}$$

To finish, let us prove that for almost all σ, $n_\sigma \leq 2\log h(|\rho_\sigma|) + \mathcal{O}(1)$. Since $C(\sigma) \leq |\rho_\sigma| + 2n_\sigma + \mathcal{O}(1)$, this upper bound of n_σ will imply that

$$\begin{aligned} C(\sigma) &\leq |\rho_\sigma| + 4\log h(|\rho_\sigma|) + \mathcal{O}(1) \\ &\leq |\rho_\sigma| + h(|\rho_\sigma|) \\ &= C^A(\sigma) + h(C^A(\sigma)) \end{aligned}$$

for almost all σ, as we wanted. Hence, let us see that $n_\sigma \leq 2\log h(|\rho_\sigma|) + \mathcal{O}(1)$ for almost all σ. There is a constant n_0 such that for all $n \geq n_0$, $8\log n \leq n$. Since h is non-decreasing and unbounded, we know that for almost all σ, ρ_σ satisfies $\log h(|\rho_\sigma|) \geq n_0$. Suppose σ has this property. Then either $n_\sigma \leq \log h(|\rho_\sigma|)$ or $4\log n_\sigma \leq n_\sigma/2$. In the second case $n_\sigma - 4\log n_\sigma \geq n_\sigma/2$ and by (5.1), $n_\sigma/2 \leq \log h(|\rho_\sigma|) + \mathcal{O}(1)$. So, in both cases, we have $n_\sigma \leq 2\log h(|\rho_\sigma|) + \mathcal{O}(1)$ and this completes the proof. □

Lemma 5.4.2. *For all $\sigma \in 2^{<\omega}$ and $d \in \mathbb{N}$,*

$$\|\{\tau \colon C(\sigma,\tau) \leq C(\sigma) + d\}\| \leq \mathcal{O}(d^4 2^d).$$

Proof. Chaitin [23] proved that

$$(\forall d, n \in \mathbb{N})\, \|\{\sigma \colon |\sigma| = n \ \wedge\ C(\sigma) \leq C(n) + d\}\| \leq \mathcal{O}(2^d).$$

We can represent σ with $0^{\mathrm{str}^{-1}(\sigma)}1$. Let c be such that $(\forall\sigma)\, C(\sigma) \leq \mathrm{str}^{-1}(\sigma) + c$. Consider the partial computable function $f(\sigma,\tau,d)\colon 2^{<\omega} \times 2^{<\omega} \times \mathbb{N} \to 2^{<\omega}$ which enumerates all strings χ such that $C(\sigma,\chi) \leq \mathrm{str}^{-1}(\sigma) + d + c$ until it finds $\chi = \tau$. If χ was the i-th string to appear in the enumeration, then $f(\sigma,\tau,d)$ is the number i written in binary with initial zeroes such that $|f(\sigma,\tau,d)| = \mathrm{str}^{-1}(\sigma) + d + c + 1$. Notice that it is always possible to write $f(\sigma,\tau,d)$ in this way because there are at most $2^{\mathrm{str}^{-1}(\sigma)+d+c+1}$ such strings χ. If no such χ exists, then $f(\sigma,\tau,d)\uparrow$.

Let σ and d be given. Consider τ such that $C(\sigma,\tau) \leq C(\sigma) + d$. Since $C(\sigma,\tau) \leq \mathrm{str}^{-1}(\sigma) + d + c$ then $f(\sigma,\tau,d)\downarrow$ and we can represent $f(\sigma,\tau,d)$ with σ, τ and d, so

$$\begin{aligned} C(f(\sigma,\tau,d)) &\leq C(\sigma,\tau) + 2\log d + \mathcal{O}(1) \\ &\leq C(\sigma) + d + 2\log d + \mathcal{O}(1). \end{aligned} \tag{5.2}$$

We can compute the string σ from the numbers $\mathrm{str}^{-1}(\sigma) + d + c + 1$ and d (recall that c is a fixed constant) and then from 5.2 we conclude that there is a constant c' such that

$$C(f(\sigma,\tau,d)) \leq C(\mathrm{str}^{-1}(\sigma) + d + c + 1) + d + 4\log d + c'.$$

Let $n = \mathrm{str}^{-1}(\sigma) + d + c + 1$ and $d' = d + 4\log d + c'$. For fixed σ and d, the mapping $\tau \mapsto f(\sigma,\tau,d)$ is injective and thus

$$\begin{aligned} \|\{\tau \colon C(\sigma,\tau) \leq C(\sigma) + d\}\| &\leq \|\{\sigma \colon |\sigma| = n \ \wedge\ C(\sigma) \leq C(n) + d'\}\| \\ &\leq \mathcal{O}(2^{d'}) = \mathcal{O}(d^4 2^d). \end{aligned}$$

This completes the proof. □

Theorem 5.4.3. *The following are equivalent:*

1. *A is strongly jump-traceable;*

2. *For every order function h and almost every σ,*

$$C(\sigma) \leq C^A(\sigma) + h(C^A(\sigma)).$$

Proof. 1⇒2. Let h_0 be a given order function. It is sufficient to show that $C(\sigma) \leq C^A(\sigma) + h(C^A(\sigma)) + \mathcal{O}(1)$ for almost all σ, where $h = \lfloor h_0/2 \rfloor$. Let α be a reduction function such that $J^A(\alpha(x)) = \mathbf{U}^A(\mathrm{str}(x))$. Let T be a trace for J^A with bound g such that $g(\alpha(x)) \leq h(|\mathrm{str}(x)|)$. Let $m \in \mathbb{N}$ be such that $\mathbf{U}^A(\mathrm{str}(m)) = \tau$ and $|\mathrm{str}(m)| = C^A(\tau)$. Since $\tau \in T_{\alpha(m)}$, we can code τ with m and a number not greater than $g(\alpha(m))$ representing the time in which τ is enumerated into $T_{\alpha(m)}$, using at most

$$|\mathrm{str}(m)| + g(\alpha(m)) \leq C^A(\tau) + h(C^A(\tau))$$

many bits. Then $(\forall \sigma)\, C(\sigma) \leq C^A(\sigma) + h(C^A(\sigma)) + \mathcal{O}(1)$.

2⇒1. Since there are at most $2^n - 1$ programs of length less than n,

$$(\forall n)(\exists \sigma)\, [|\sigma| = n \ \wedge \ n \leq C(\sigma)].$$

Let c be a constant such that

$$(\forall \sigma)\, [J^A(|\sigma|)\downarrow \Rightarrow \ C^A(\sigma, J^A(|\sigma|)) \leq |\sigma| + c].$$

This last inequality holds because, given σ, we can compute $J^A(|\sigma|)$ relative to A.

Let h be any order function and let us prove that A is jump-traceable via h. Define the order function g such that for almost all e, $3^{g(e+c)} \leq h(e)$. By hypothesis, for almost all σ, if $J^A(|\sigma|)\downarrow$ then

$$\begin{aligned} C(\sigma, J^A(|\sigma|)) &\leq C^A(\sigma, J^A(|\sigma|)) + g(C^A(\sigma, J^A(|\sigma|))) \\ &\leq |\sigma| + g(|\sigma| + c) + c. \end{aligned} \tag{5.3}$$

Define the trace

$$T_e = \{\tau \colon (\forall \sigma)\, [|\sigma| = e \ \Rightarrow \ C(\sigma, \tau) \leq e + g(e + c) + c]\}.$$

It is clear that for almost all e, if $J^A(e)\downarrow$ then $J^A(e) \in T_e$, because given σ such that $|\sigma| = e$, we have by (5.3) that $C(\sigma, J^A(e)) \leq e + g(e+c) + c$. To verify that for almost all e, $\|T_e\| \leq h(e)$, suppose $\tau \in T_e$. Take σ, $|\sigma| = e$ and $C(\sigma) \geq e$. Then

$$\begin{aligned} C(\sigma, \tau) &\leq e + g(e + c) + c \\ &\leq C(\sigma) + g(e + c) + c. \end{aligned}$$

By Lemma 5.4.2, for almost all e there are at most $3^{g(e+c)} \leq h(e)$ such τs in T_e. □

In [60], it was proven that there is a super-low which is not jump-traceable (namely, a super-low random set). In contrast, from Theorem 5.4.1 and Theorem 5.4.3 we can conclude that the strong version of super-lowness implies strong jump-traceability.

Corollary 5.4.4. *If A' is well-approximable then A is strongly jump-traceable.*

5.5 Variations on K-triviality

Throughout this section, let $p\colon \mathbb{N} \to \mathbb{N}$ be strictly increasing such that $\lim_n p(n) - n = \infty$. We call p an *estimation function* if, in addition, $p(n) = \lim_s p_s(n)$ where $p_{s+1}(n) \leq p_s(n)$, and the function $\lambda s, n.p_s(n)$ is computable. As we said before, Nies [62] proved that A is K-trivial if and only if A is low for K. In this section we weaken the notion of K-triviality and lowness for K:

Definition 5.5.1 (Weakly p-low and p-low).

1. A set A is *weakly p-low* iff

$$(\forall n)\, K(A \upharpoonright n) \leq p(K(n) + c_0) + c_1$$

   for some constants c_0 and c_1. Let $\mathcal{K}[p]$ denote the class of such sets.

2. A set A is *p-low* iff

$$(\forall \sigma)\, K(\sigma) \leq p(K^A(\sigma) + c_0) + c_1$$

   for some constants c_0 and c_1. Let $\mathcal{M}[p]$ denote the class of such sets.

We say that $A \leq_K B$ if and only if $(\forall n)\, K(A \upharpoonright n) \leq K(B \upharpoonright n) + \mathcal{O}(1)$.

Proposition 5.5.2.

1. *If $A \in \mathcal{M}[p]$ and $B \leq_T A$, then $B \in \mathcal{M}[p]$.*
2. *If $A \in \mathcal{K}[p]$ and $B \leq_K A$ or $B \leq_{wtt} A$, then $B \in \mathcal{K}[p]$.*
3. *Suppose p is an estimation function. Then no random set is in $\mathcal{K}[p]$.*
4. *If $A, B \in \mathcal{K}[p]$ and A, B are c.e., then $A \oplus B \in \mathcal{K}[p]$.*
5. *$\mathcal{M}[p] \subseteq \mathcal{K}[p]$.*

Proof.

1. Since $B \leq_T A$, there exists a constant c_2 such that for each string σ, $K^A(\sigma) \leq K^B(\sigma) + c_2$. Then

$$\begin{aligned} K(\sigma) &\leq p(K^A(\sigma) + c_0) + c_1 \\ &\leq p(K^B(\sigma) + c_0 + c_2) + c_1. \end{aligned}$$

2. This is trivial for $\leq_K$. Now suppose $B = \Gamma^A$ for a weak truth-table reduction Γ with computable bound f. Without loss of generality, we may assume f strictly increasing. Given $A \upharpoonright f(n)$ we can compute n and $B \upharpoonright n$, and then there is a constant c_2 such that for all n,

$$\begin{aligned} K(B \upharpoonright n) &\leq K(A \upharpoonright f(n)) + c_2 \\ &\leq p(K(f(n)) + c_0) + c_1 + c_2. \end{aligned}$$

   Since f is computable, we have $K(f(n)) \leq K(n) + \mathcal{O}(1)$, and hence $B \in \mathcal{K}[p]$.

3. Assume $(\forall n)\, K(A \upharpoonright n) > n - c$ and $A \in \mathcal{K}[p]$ via constants c_0 and c_1. Define the strictly increasing computable function $\tilde{p}(0) = p_0(0)$ and $\tilde{p}(k+1) = p_0(j)$, where $j = \min\{i \colon i > k \ \wedge\ p_0(i) > \tilde{p}(k)\}$. Since $\tilde{p} \geq p$, $A \in \mathcal{K}[\tilde{p}]$. Define the Kraft-Chaitin set
$$\{\langle i, n_i \rangle \colon i \in \mathbb{N}^+ \ \wedge\ n_i = \tilde{p}(i + d + c_0) + c_1 + c\}$$
for $\mathbf{M}_d$ with d given in advance by the Recursion Theorem. Then $K(n_i) \leq i + d$ and hence $\tilde{p}(K(n_i) + c_0) \leq \tilde{p}(i + d + c_0)$. Finally,
$$\begin{aligned} K(A \upharpoonright n_i) &\leq \tilde{p}(K(n_i) + c_0) + c_1 \\ &\leq \tilde{p}(i + d + c_0) + c_1 = n_i - c, \end{aligned}$$
and this is a contradiction.

4. Ignoring constants, for each n,
$$\begin{aligned} K(A \oplus B \upharpoonright n) &\leq K(A \oplus B \upharpoonright 2n) \\ &\leq \max\{K(A \upharpoonright n), K(B \upharpoonright n)\} \\ &\leq p(K(n)). \end{aligned}$$
In the second inequality we used [35, Theorem 6.4].

5. Again ignoring constants, for all n,
$$\begin{aligned} K(A \upharpoonright n) &\leq p(K^A(A \upharpoonright n)) \\ &\leq p(K^A(n)) \\ &\leq p(K(n)). \end{aligned}$$

This completes the proof. □

The following proposition shows a connection between jump-traceability and p-lowness. In Theorem 5.4.3 we proved a similar result, relating strong jump-traceability and plain program-size complexity.

Proposition 5.5.3. *1. Suppose p is a computable function. There is a constant c such that if $A \in \mathcal{M}[p]$ via constants c_0 and c_1 then A is jump-traceable via $h(x) = 2^{p(2\log x + c_0 + c) + c_1 + 1}$;*

2. There is a reduction function α such that if A is jump-traceable via h then $A \in \mathcal{M}[p]$ for $p(z) = 3z + 2\log h(\alpha(2^{z+1}))$.

Proof. For 1, we know that there is a constant c such that $K^A(J^A(\sigma)) \leq 2|\sigma| + c$ because we can compute $J^A(\sigma)$ from σ and the oracle A. Define the trace
$$T_x = \{\mathbf{U}(\rho) \colon |\rho| \leq p(2\log x + c_0 + c) + c_1\}.$$
Clearly $\|T_x\| \leq 2^{p(2\log x + c_0 + c) + c_1 + 1}$. Let $y = J^A(x)$. By hypothesis $K(y) \leq p(K^A(y) + c_0) + c_1$ and then $K(y) \leq p(2\log x + c + c_0) + c_1$. Hence $y \in T_x$.

For 2, let α be a reduction function such that

$$J^A(\alpha(x)) = \mathrm{str}^{-1}(\mathbf{U}^A(\mathrm{str}(x))).$$

Let T be a trace for J^A with bound h and let us define the trace

$$\tilde{T}_n = \bigcup_{x:|\mathrm{str}(x)|=n} T_{\alpha(x)}.$$

Notice that

$$\begin{aligned} \|\tilde{T}_n\| &\leq \sum_{x:|\mathrm{str}(x)|=n} h(\alpha(x)) \\ &\leq 2^n h(\alpha(2^{n+1})), \end{aligned}$$

since α is increasing. Let $m \in \mathbb{N}$ be such that $\mathbf{U}^A(\mathrm{str}(m)) = \sigma$ and $|\mathrm{str}(m)| = K^A(\sigma)$. Since $\mathrm{str}^{-1}(\sigma) \in T_{\alpha(m)}$, we know that $\mathrm{str}^{-1}(\sigma) \in \tilde{T}_{|\mathrm{str}(m)|}$, hence we describe σ by saying "$\mathrm{str}^{-1}(\sigma)$ is the i-th element enumerated into $\tilde{T}_{|\mathrm{str}(m)|}$". If we code $|\mathrm{str}(m)|$ in unary and we code i with

$$\begin{aligned} 2\log i &\leq 2\log(2^{|\mathrm{str}(m)|} h(\alpha(2^{|\mathrm{str}(m)|+1}))) \\ &\leq 2|\mathrm{str}(m)| + 2\log h(\alpha(2^{|\mathrm{str}(m)|+1})) \end{aligned}$$

many bits, we have

$$K(\sigma) \leq p(K^A(\sigma)) + \mathcal{O}(1),$$

for

$$p(z) = 3z + 2\log h(\alpha(2^{z+1})).$$

This completes the proof. □

Corollary 5.5.4. *A is jump-traceable iff there exists a computable function p* (*of the type considered in this section*) *such that* $A \in \mathcal{M}[p]$.

Proposition 5.5.5. *Let p be an estimation function. Then there is a Turing complete c.e. set A which is weakly p-low and also satisfies the corresponding property for C, i.e. there are constants* c_K, c_C *such that* $K(A \upharpoonright x) \leq p(K(x)) + c_K$ *and* $C(A \upharpoonright x) \leq p(C(x)) + c_C$ *for all x.*

Proof. Construction of A. For defining an enumeration of A, fix a one-one enumeration $b_0, b_1, \ldots$ of the Halting Problem and approximations C_s and K_s to C and K respectively.

Stage 0: set $A_0 = \emptyset$.

Stage $s+1$: let a_m be the m-th non-element of A_s in ascending order. Now the set A_{s+1} is computed as follows.

- Let n be the minimum of all m such that one of the following conditions holds:

  1. $a_m > s$;

2. $b_s \leq m$;
3. $p_s(K_s(k)) - K_s(k) \leq m$ for some k with $a_m \leq k \leq s$;
4. $p_s(C_s(k)) - C_s(k) \leq m$ for some k with $a_m \leq k \leq s$.

- Let $A_{s+1} = A_s \cup \{x\colon a_n \leq x \leq s\}$.

The so constructed set A satisfies the following properties:

- A is co-infinite and c.e.;
- A is Turing complete;
- $K(A \upharpoonright x) \leq p(K(x)) + c_K$ for some constant c_K and all x;
- $C(A \upharpoonright x) \leq p(C(x)) + c_C$ for some constant c_C and all x.

Let us prove separately each of the statements below.

Lemma 5.5.6. *A is co-infinite and c.e.*

Proof. The first property states the obvious fact that A is c.e. by the construction. The other fact that A is co-infinite needs some more thought. Assume by way of contradiction that $\|\mathbb{N} \setminus A\| = m$ for some finite number m. Let $a_0, a_1, \ldots, a_{m-1}$ denote the non-elements of A in ascending order and assume that s is so large that the following conditions hold:

(i) if $b_t \leq m$ then $t < s$;

(ii) for all $x \in A \setminus A_s$ there is no $k \geq x$ and no $e \geq \min\{C(k), K(k)\}$ such that $p(e) - e \leq m$;

(iii) if $x \leq a_{m-1} + 1$ then $x \in A \Leftrightarrow x \in A_s$.

Because of condition (iii), one can see that the parameters $a_0, a_1, \ldots, a_{m-1}$ chosen in the definition at stage $s+1$ coincide with the m least non-elements of A and are just not enumerated. Furthermore, a_m is also defined as the next non-element of A_s, that is, a_m is the least non-element of A_s greater than a_{m-1}. Since by construction $s \notin A_s$, we have $a_m \leq s$.

Now one can see that a_m is not enumerated into A_{s+1} because the n selected at stage $s+1$ of the construction is larger than m: for all $m' < m$, $n \neq m'$ because otherwise $a_0, a_1, \ldots, a_{m-1}$ would not remain outside A. Furthermore one can see that $n \neq m$ analyzing which of the search-conditions 1, 2, 3 or 4 in the definition of A at stage $s+1$ apply for n.

- If condition 1 holds then $a_m > s$ and this contradicts the fact mentioned above;
- If condition 2 holds then $b_s \leq m$ but this cannot happen because of condition (i) in the selection of s;

- If condition 3 holds then $p_s(K_s(k)) - K_s(k) \leq m$ for some k with $a_m \leq k \leq s$. For $e = K_s(k)$ this means that $m \geq p_s(e) - e \geq p(e) - e$, as p_s approximates p from above. Now this is not possible because of condition (ii);
- The same happens if condition (4) holds.

So $a_m \notin A_{s+1}$ and one can show by induction that $a_m \notin A_t$ for all $t > s$, this contradicts the assumption that $\|\mathbb{N} \setminus A\| = m$. Therefore, A is co-infinite. □

Lemma 5.5.7. *A is Turing complete.*

Proof. It follows from the construction. If $a_0, a_1, \ldots$ are the non-elements of A in ascending order, then $b_s \leq m$ implies $s \leq a_m$. Indeed, if $b_s \leq m$ then m satisfies the search-condition 2 and thus $\{a_n, \ldots, s\} \subseteq A_{s+1}$ for some $n \leq m$. If $s > a_m$ then a_m would be put into A_{s+1} and this contradicts the fact that a_m is a non-element of A.

Thus m is in the Halting Problem if and only if $m \in \{b_0, b_1, \ldots, b_{a_m}\}$ and so the Halting Problem is Turing reducible to A. □

Lemma 5.5.8. *Let n be the number of non-elements of A below x. Then $K(x)+n \leq p(K(x))$.*

Proof. Let $a_0, \ldots, a_{n-1}$ be the non-elements of A below x in ascending order. Suppose there exists infinitely many s such that $K_s(x)+n > p_s(K_s(x))$. Since $x > a_{n-1}$, for s large enough the search-condition 3 would become true and then a_{n-1} would be enumerated into A. Hence, for all sufficiently large s, $K_s(x) + n \leq p_s(K_s(x))$. Therefore $K(x) + n \leq p(K(x))$. □

Lemma 5.5.9. *$K(A \upharpoonright x) \leq p(K(x)) + c_K$ for some constant c_K and all x.*

Proof. Given x and the shortest description ρ for x with respect to a fixed prefix-free universal machine and, as in Lemma 5.5.8, let n be the number of non-elements of A below x. Then one can construct a prefix-free machine which from input $1^n0\rho$ first evaluates the universal machine on ρ to get the value x and then searches for a stage s such that A_s contains all but n elements below x. Having this x and s, the machine outputs $A_s \upharpoonright x$. If ρ and n are chosen correctly, then the output is correct. Thus one has that $K(A \upharpoonright x)$ is at most $K(x) + n + c_K$ where the constant c_K comes from translating the given prefix-free coding of $K(A \upharpoonright x)$ of length $K(x) + n + 1$ for some machine into inputs for the universal machine. By Lemma 5.5.8, we conclude that A is weakly p-low, as we wanted. □

Lemma 5.5.10. *$C(A \upharpoonright x) \leq p(C(x)) + c_C$ for some constant c_C and all x.*

Proof. This can be proven analogously; here the constructed machine is not prefix-free and ρ is the shortest input producing x with respect to some fixed universal plain machine, nevertheless ρ and n can of course still be recovered from $1^n0\rho$. The rest of the proof follows the previous item but is working with C in place of K. □

This completes the proof of the whole result. □

As a corollary, we obtain that the inclusion from Proposition 5.5.2 item 5 is strict.

Corollary 5.5.11. *For all estimation functions p, $\mathcal{M}[p] \subsetneq \mathcal{K}[p]$.*

Proof. Let p be any estimation function and let $A \in \mathcal{K}[p]$ be the set constructed in Proposition 5.5.5. Since A is Turing complete, $\Omega \leq_T A$. Suppose $A \in \mathcal{M}[p]$. Then by item 1 of Proposition 5.5.2, we have that $\Omega \in \mathcal{M}[p]$ and by item 5 of Proposition 5.5.2, $\Omega \in \mathcal{K}[p]$. This contradicts item 3 of Proposition 5.5.2. □

Proposition 5.5.12. *For every estimation function p there is a whole Turing degree outside Δ^0_2 contained in $\mathcal{K}[p]$.*

Proof. For any estimation function p one can consider the estimation function q given as $q(n) = n + (p(n) - n)/4$. Observe that $\lim_n q(n) - n = \infty$ and q is approximated from above because p is an estimation function. Then one can construct a c.e. set A as in Proposition 5.5.5 which is in $\mathcal{K}[q]$.

The set A is not computable. Thus, due to Yates's version of the Friedberg-Muchnik Splitting Theorem [63, Theorem IX.2.4 and Exercise IX.2.5], one can construct a partial computable $\{0, 1\}$-valued function ψ with domain A such that $\psi^{-1}(0), \psi^{-1}(1)$ form a recursively inseparable pair, that is, ψ does not have a total extension. Actually, given a one-one enumeration $a_0, a_1, \ldots$ of A, this function ψ can be inductively defined on this domain by taking $\psi(a_s)$ in $\{0, 1\}$ such that $\psi(a_s)$ differs from $\varphi_{e,s}(a_s)$ for the least e where either $e = s$ or $\varphi_{e,s}(a_s)$ is defined and $\psi(a_t) = \varphi_{e,s}(a_t)$ for all $t < s$ with $a_t \in \mathrm{dom}(\varphi_{e,s})$.

Let B be any total extension of ψ. Given any x, by Lemma 5.5.8 the number n of places below x where ψ is undefined satisfies $n < q(K(x)) - K(x)$. Let $x_1, x_2, \ldots, x_n$ be these places and let ρ be the shortest input such that the prefix-free universal machine computes x. Then one can code $B \upharpoonright x$ by $1^n 0 B(x_1) B(x_2) \ldots B(x_n) \rho$ and thus concludes that

$$\begin{aligned} K(B \upharpoonright x) &\leq 2n + K(x) + 1 \\ &\leq 2q(K(x)) - K(x) \\ &= K(x) + (p(K(x)) - K(x))/2 \\ &\leq p(K(x)) \end{aligned}$$

As one can take B to have hyperimmune-free Turing degree [63, Theorem V.5.34] and as $\mathcal{K}[p]$ is closed under *wtt*-reducibility, one has that a whole Turing degree outside Δ^0_2 is contained in $\mathcal{K}[p]$. □

Note that the above result also holds with C in place of K, the proof is exactly the same. So given an estimation function p, one can construct a hyperimmune-free Turing degree only consisting of sets E satisfying $C(E \upharpoonright x) \leq p(E(x))$ for all x up to an additive constant. Unfortunately, it is not guaranteed that this degree is also strongly jump-traceable, it is even a bit unlikely, as only the use of total E-computable functions but not of the jump is computably bounded in the case of a set E of hyperimmune-free degree.

6. OPEN QUESTIONS AND FUTURE RESEARCH

We close this thesis with a list of the main questions that remain open in each chapter.

In chapter 2 we showed two algorithms for constructing absolutely normal reals. However these algorithms are highly exponential and do not lead to feasible computer implementations. To determine just the first digit we would need an enormous amount of time. One important question that remained unsolved is the existence of polynomial time bounded algorithms for computing absolutely normal numbers. It is known that there are such algorithms for reals that are normal *to a given scale*, but nothing is known about being normal to *every scale*. It seems that the techniques used in the proof of theorems 2.4.9 and 2.6.2 are not suitable for polynomial time bounded constructions because the idea of such algorithms is to somehow detect reals that are candidates to be normal numbers to more and more scales by looking at the fractional expansion of larger and larger prefixes. This brute force search induces a natural exponential solution. In this way, we can prove the absolute normality only for numbers that are constructed on purpose to be absolutely normal. In order to get a fast algorithm, we probably need a new technique to prove absolute normality in a more general framework, where the digits are not so tightly dependent on the definition of absolute normality.

Another intriguing problem that remained unanswered in chapter 2 is Turing's unproved Lemma 2.7.2. We were unable to prove it, and we were also unable to find a counter example because the numbers one must deal with are very large, even for a computer program. In case this statement is true, the proof technique should be rather different from our strategy for proving Lemma 2.7.7.

In chapter 3 we presented Conjecture 3.1.1 on the randomness of halting probabilities and we identified exhaustively the cases in which this conjecture is true and false. A rather simple question (that later appeared in [58, Question 8.10]) remained unsolved: is $\Omega_{\mathbf{U}}[X]$ random when X is co-c.e.? We know that the answer is *no* for finite sets, but we might analyze what happens when X is infinite. Theorem 3.5.5 gives a partial negative answer to this question when we can fix a suitable universal machine. We do not know what happens when $\mathbf{U}$ is given (for example, it would be nice to know the answer for universal by adjunction machines). Is there always an infinite co-c.e. set X for which $\Omega_{\mathbf{U}}[X]$ is not random, regardless the underlying $\mathbf{U}$? We believe that an avenue to give a positive answer may be to use a result of Rettinger and Zheng [65] saying that any random which is the difference of two left-c.e. reals must be either left-c.e. or right-c.e. Indeed, if X is co-c.e., $\Omega_{\mathbf{U}}[X]$ is the difference of two left-c.e. reals (namely, $\Omega_{\mathbf{U}}$ and $\Omega_{\mathbf{U}}[\mathbb{N} \setminus X]$). The idea is for any given universal machine $\mathbf{U}$, to construct a co-c.e. set X for which $\Omega_{\mathbf{U}}[X]$ is neither left-c.e. nor right-c.e.

Theorem 3.5.13 shows that for any $n \geq 2$, if X is Σ^0_n then $\Omega_{\mathbf{U}}[X]$ cannot be n-random. But Theorem 3.6.1 says that if such X is Σ^0_n-complete, then $\Omega_{\mathbf{U}}[X]$ is random. This leaves open the possibility that there is a shift in the second part of Conjecture 3.1.1. It could be that n-randomness was too much, and we have to decrease to m-randomness. In other words, is it true that for $n \geq 2$, if X is Σ^0_n-complete then $\Omega_{\mathbf{U}}[X]$ is m-random, for some $m \in \{2, \ldots, n-1\}$?

In chapter 4 we talked about a new behavior of Turing machines and we defined K^∞, the associated prefix-free program-size complexity for these new unending computations. In Theorem 4.5.6 we characterized the computable sequences as those being K^∞-trivial. Recall that sequences can be also seen as sets or reals. We might think about other unending machine behaviors with the aim to characterize the c.e. sets or the left-c.e. reals as those being $\tilde{K}$-trivial, where $\tilde{K}$ is the respective induced prefix-free program-size complexity by each behavior. For example, one may consider a monotone machine (as described in section 4.2) with some relaxing conditions regarding the action with respect to the output tape, such as *the output tape can be overwritten but it is only permitted to put a* 1 *over a* 0 or such as *the output tape may be overwritten but the partial output is monotone with respect to the order of the reals, as time increases.* As future research we will study if these (or a variation of these) behaviors induce a definition of $\tilde{K}$-triviality which captures the notion of c.e. set or left-c.e. real respectively.

In chapter 5 we introduced the notions of strong jump-traceability and well-approximability with the aim to capture the class of K-trivial reals. Proposition 5.2.3, Theorem 5.2.5, Theorem 5.3.7 and Corollary 5.5.4 give some clues that strong jump-traceability might work for our purpose. In the same way, Proposition 5.3.2 and Theorem 5.3.6 give some evidence that well-approximability can also be suitable. Furthermore, Theorem 5.3.5 and Corollary 5.4.4 show some connection between our two studied combinatorial notions each other. Theorem 5.4.1 shows an interesting and peculiar relationship between strong jump-traceability and the plain C complexity, while Corollary 5.5.4 shows some connection between jump-traceability and a variant of the low for K sets. We were not able to characterize the class of K-trivial reals but the results we obtained so far do not falsify this idea. Another open question that we left in this chapter is if there is a *minimal* bound for jump-traceability. In other words, is there a very slow growing order function h such that being jump-traceable via h is equivalent to being strongly jump-traceable?

BIBLIOGRAPHY

[1] Klaus Ambos-Spies. Algorithmic randomness revisited. In B. McGuinness, editor, *Language, Logic and Formalization of Knowledge. Coimbra Lecture and Proceedings of a Symposium held in Siena in September 1997*, pages 33–52. Bibliotheca, 1998.

[2] Klaus Ambos-Spies, Elvira Mayordomo, Yongge Wang, and Xizhong Zheng. Resource bounded balanced genericity stochasticity and weak randomness. In *Proceedings of the 13th Symposium on Theoretical Aspects of Computer Science*, volume 1046, pages 63–74. Springer Lecture Notes in Computer Science, 1996.

[3] David H. Bailey and Richard E. Crandall. On the random character of fundamental constant expansions. *Experimental Mathematics*, 10(2):175–190, 2001.

[4] David H. Bailey and Richard E. Crandall. Random generators and normal numbers. *Experimental Mathematics*, 11(4):527–546, 2004.

[5] Verónica Becher and Gregory J. Chaitin. Another example of higher order randomness. *Fundamenta Informaticae*, 51(4):325–338, 2002.

[6] Verónica Becher, Sergio Daicz, and Gregory J. Chaitin. A highly random number. In *Combinatorics, Computability and Logic: Proceedings of the Third Discrete Mathematics and Theoretical Computer Science Conference (DMTCS'01)*, pages 55–68. Springer-Verlag London, 2001.

[7] Verónica Becher and Santiago Figueira. An example of a computable absolutely normal number. *Theoretical Computer Science*, 270:947–958, 2001.

[8] Verónica Becher and Santiago Figueira. Kolmogorov complexity for possibly infinite computations. *Journal of Logic, Language and Information*, 14(2):133–148, 2005.

[9] Verónica Becher, Santiago Figueira, Serge Grigorieff, and Joseph S. Miller. Randomness and halting probabilities. *The Journal of Symbolic Logic*, 2006. To appear.

[10] Verónica Becher, Santiago Figueira, André Nies, and Silvana Picchi. Program size complexity for possibly infinite computations. *Notre Dame Journal of Formal Logic*, 46(1):51–64, 2005.

[11] Verónica Becher, Santiago Figueira, and Rafael Picchi. Turing's unpublished algorithm for normal numbers. Submitted, 2006.

[12] Verónica Becher and Serge Grigorieff. Random reals and possibly infinite computations. Part II: From index sets to higher order randomness. Manuscript, in preparation.

[13] Verónica Becher and Serge Grigorieff. Recursion and topology on $2^{\leq\omega}$ for possibly infinite computations. *Theoretical Computer Science*, 322(1):85–136, 2004.

[14] Verónica Becher and Serge Grigorieff. Random reals and possibly infinite computations Part I: Randomness in $\emptyset'$. *The Journal of Symbolic Logic*, 70(3):891–913, 2005.

[15] Mark Bickford and Charlie F. Mills. Lowness properties of r.e. sets. Manuscript, UW Madison, 1982.

[16] Émile Borel. Les probabilités dénombrables et leurs applications arithmétiques. *Rendiconti del Circolo Matematico di Palermo*, 27:247–271, 1909.

[17] Jonathan Borwein and David H. Bailey. *Mathematics by Experiment: Plausible Reasoning in the 21st Century.* A K Peters Ltd., Natick, MA.

[18] Cristian Calude. Borel normality and algorithmic randomness. In G.Rozenberg and A.Salomaa, editors, *Developments in Language Theory*, pages 113–129, Singapore, 1994. World Scientific.

[19] Cristian Calude. *Information and Randomness, an Algorithmic Perspective.* Springer-Verlag, Berlin, 1994.

[20] Cristian Calude and Richard J. Coles. Program size complexity of initial segments and domination relation reducibility. In J.Karhümaki, H.Mauer, G.Paŭn, and G.Rozenberg, editors, *Jewels are Forever*, pages 225–237. Springer-Verlag, 1999.

[21] Cristian Calude, Peter Hertling, Bakhadyr Khoussainov, and Yongge Wang. Recursively enumerable reals and Chaitin Ω number. In *Symposium on Theoretical Aspects of Computer Science*, volume 1373 of *Lecture Notes in Computer Science*, pages 596–606, 1998.

[22] Gregory J. Chaitin. On the length of programs for computing finite binary sequences: statistical considerations. *Journal of the ACM*, 16(1):145–159, 1969.

[23] Gregory J. Chaitin. A theory of program size formally identical to information theory. *Journal of the ACM*, 22:329–340, 1975.

[24] Gregory J. Chaitin. Information-theoretical characterizations of recursive infinite strings. *Theoretical Computer Science*, 2:45–48, 1976.

[25] Gregory J. Chaitin. *Algorithmic information theory.* Cambridge University Press, New York, NY, USA, 1987.

[26] Gregory J. Chaitin. Incompleteness theorems for random reals. *Advances in Applied Mathematics*, 8:119–146, 1987.

[27] David G. Champernowne. The construction of decimals in the scale of ten. *Journal of the London Mathematical Society*, 8:254–260, 1933.

[28] Alonzo Church. On the concept of a random sequence. *Bulletin of the American Mathematical Society*, 46:130–135, 1940.

[29] A.H. Copeland and P. Erdös. Note on normal numbers. *Bulletin of the American Mathematical Society*, 52:857–860, 1946.

[30] Nigel Cutland. *Computability, an introduction to recursive function theory.* Cambridge University Press, 1980.

[31] Rod Downey, André Nies Denis R. Hirschfeldt, and Sebastiaan Terwijn. Calibrating randomness. *Bulletin of Symbolic Logic*, 2006. To appear.

[32] Rod G. Downey and Denis R. Hirschfeldt. Algorithmic randomness and complexity. Inpreparation. Accesible form `http://www.mcs.vuw.ac.nz/~downey/randomness.pdf`.

[33] Rod G. Downey, Denis R. Hirschfeldt, and André Nies. Randomness, computability, and density. *SIAM Journal on Computing*, 31(4):1169–1183, 2002.

[34] Rod G. Downey, Denis R. Hirschfeldt, André Nies, and Frank Stephan. Trivial reals. *Electronic Notes in Theoretical Computer Science*, 66(1), 2002. Final version in [35].

[35] Rod G. Downey, Denis R. Hirschfeldt, André Nies, and Frank Stephan. Trivial reals. In *Proceedings of the 7th and 8th Asian Logic Conferences*, pages 103–131. World Scientific, River Edge, NJ, 2003.

[36] Bruno Durand and Nikolai K. Vereshchagin. Kolmogorov-loveland stochasticity for finite strings. *Information Processing Letters*, 91(6):263–269, 2004.

[37] Santiago Figueira, Joseph S. Miller, and André Nies. Indifferent sets. Unpublished, 2006.

[38] Santiago Figueira, André Nies, and Frank Stephan. Lowness properties and approximations of the jump. In *12th Workshop on Logic, Language, Information and Computation*, volume 143 of *Electronic Notes in Computer Science*, pages 45–57, 2005.

[39] Santiago Figueira, Frank Stephan, and Guohua Wu. Randomness and universal machines. In *CCA 2005, Second International Conference on Computability and Complexity in Analysis*, pages 103–116, Fernuniversität Hagen, Informatik Berichte, July 2005.

[40] Péter Gács. On the symmetry of algorithmic information. *Soviet Mathematics Dolkady*, 15:1477–1480, 1974.

[41] G.H. Hardy and E.M. Wright. *An Introduction to the Theory of Numbers.* Oxford University press, 1979.

[42] Glyn Harman. *Metric Number Theory*, volume 18 of *London Mathematical Society Monographs.* Oxford Universaity Press, 1998.

[43] Jr. Hartley Rogers. *Theory of recursive functions and effective computability.* MIT Press, Cambridge, MA, USA, 1987.

[44] Stephen Cole Kleene. General recursive functions of natural numbers. *Mathematische Annalen*, 112:727–742, 1936.

[45] Stephen Cole Kleene. On notations for ordinal numbers. *The Journal of Symbolic Logic*, 3:150–155, 1938.

[46] Andrei N. Kolmogorov. Three approaches to the quantitative definition of information. *Problems of Information Transmission*, 1:1–17, 1965.

[47] L. G. Kraft. A device for quantizing, grouping, and coding amplitude modulated pulses. M.Sc. Thesis, Massachusetts Institute of Technology, 1949.

[48] Antonin Kučera and Theodore A. Slaman. Randomness and recursive enumerability. *SIAM Journal on Computing*, 31(1):199–211, 2001.

[49] L. Kuipers and H. Niederreiter. *Uniform distribution of sequences.* Wiley Interscience, New York, 1974.

[50] Leonid A. Levin. Some theorems on the algorithmic approach to probability theory and information theory. Dissertation in Mathematics, Moscow, 1971.

[51] Leonid A. Levin. On the concept of a random sequence. *Doklady Akad. Nauk SSSR*, 14(5):1413–1416, 1973.

[52] Leonid A. Levin. Laws of information conservation (non-growth) and aspects of the foundations of probability theory. *Problems of Information Transmission*, 10(3):206–210, 1974.

[53] Leonid A. Levin and A. K. Zvonkin. The complexity of finite objects and the development of the concepts of information and randomness by means of the theory of algorithms. *Russian Mathematical Surveys*, 25:83–124, 1970.

[54] Ming Li and Paul Vitányi. *An introduction to Kolmogorov complexity and its applications.* Springer, 2nd edition, 1997.

[55] Greg Martin. Absolutely abnormal numbers. *The American Mathematical Monthly*, 2000.

[56] Per Martin-Löf. The definition of random sequences. *Information and Control*, 9:602–619, 1966.

[57] Wolfgang Merkle. The complexity of stochastic sequences. In *Conference on Computational Complexity 2003*, pages 230–235. IEEE Computer Society Press, 2003. To appear in *Journal of Computer and System Sciences*.

[58] Joseph S. Miller and André Nies. Randomness and computability: open questions. In *Annual Meeting of the Association of Symbolic Logic in Stanford 2005*, 2005. `http://www.cs.auckland.ac.nz/~nies/papers/questions.pdf`.

[59] Jeanleah Mohrherr. A refinement of low n and high n for the r.e. degrees. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 32(1):5–12, 1986.

[60] André Nies. Reals which compute little. In *Proceedings of Logic Colloquium*, 2002. To appear.

[61] André Nies. Personal communication, 2004.

[62] André Nies. Lowness properties and randomness. *Advances in Mathematics*, 197:274–305, 2005.

[63] Piergiorgio Odifreddi. *Classical recursion theory*, volume 1. North-Holland, Amsterdam, 1999.

[64] John C. Oxtoby. *Measure and Category.* Springer, New York, 2nd edition, 1980.

[65] Robert Rettinger and Xizhong Zheng. Solovay reducibility on d-c.e. real numbers. In *Computing and Combinatorics, Eleventh Annual International Conference, COCOON 2005, Kunming, China, August 16-29, 2005*, volume 3595 of *Lecture Notes in Computer Science*, pages 359–368, 2005.

[66] Claus-Peter Schnorr. Zufälligkeit und Wahrscheinlichkeit. *Lecture Notes in Mathematics*, 218, 1971.

[67] Claus-Peter Schnorr. Process complexity and effective random tests. *Journal of Computer Systems Science*, 7:376–388, 1973.

[68] Alexander Shen' and Vladimir A. Uspensky. Relations between varieties of Kolmogorov complexities. *Mathematical Systems Theory*, 29:271–292, 1996.

[69] Waclaw Sierpinski. Démonstration élémentaire du théorème de m. borel sur les nombres absolument normaux et détermination effective d'un tel nombre. *Bulletin de la Société Mathématique de France*, 45:127–132, 1917.

[70] Waclaw Sierpinski. *Elementary Theory of Numbers.* Warszawa, 1964.

[71] Robert I. Soare. *Recursively enumerable sets and degrees.* Springer, Heidelberg, 1987.

[72] Ray J. Solomonoff. A formal theory of inductive inference, Part I and Part II. *Information and Control*, 7:1–22 and 224–254, 1964.

[73] Robert Solovay. Draft of a paper (or series of papers) on Chaitin's work done for the most part during the period Sept. to Dec. 1974. Unpublished manuscript, IBM Thomas J. Watson Research Center, Yorktown Heights, New York. 215 pp., May 1975.

[74] Sebastiaan Terwijn and Domenico Zambella. Algorithmic randomness and lowness. *The Journal of Symbolic Logic*, 66:1199–1205, 2001.

[75] Alan M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society, Series 2*, 42:230–265, 1936.

[76] Alan M. Turing. A note on normal numbers. In J.L. Britton, editor, *Collected Works of A.M. Turing: Pure Mathematics*, pages 117–119. North Holland, Amsterdam, 1992.

[77] Michael van Lambalgen. *Random sequences.* PhD thesis, University of Amserdam, 1987.

[78] Richard von Mises. Grundlagen der wahrscheinlichkeitsrechnung. *Mathematische Zeitschrift*, 5:52–99, 1919.

[79] Abraham Wald. Die widerspruchsfreiheit des kollectivbegriffes der wahrsheinlichkeitsrechnung. *Ergebnisse eines mathematischen Kolloquiums*, pages 38–72, 1937.

[80] Klaus Weihrauch. *Computability.* EATCS Monographs on Theoretical Computer Science 9. Springer, Heidelberg, 1987.

[81] Domenico Zambella. On sequences with simple initial segments. ILLC technical report ML 1990-05, University of Amsterdam, 1990.

INDEX OF NOTATION

INDEX