TOM FLORIAN STERKENBURG

Sequences with Trivial Initial Segment Complexity

MoL-2011-01, *received*: February 2011

ILLC Scientific Publications
Series editor: Benedikt Löwe

Master of Logic Thesis (MoL) Series, ISSN: 1387-1951

SEQUENCES WITH TRIVIAL INITIAL SEGMENT COMPLEXITY

MSc Thesis (*Afstudeerscriptie*)

written by

Tom Florian Sterkenburg
(born April 18th, 1986 in Purmerend, The Netherlands)

under the supervision of **Dr. George Barmpalias**, and submitted to the Board of Examiners in partial fulfillment of the requirements for the degree of

MSc in Logic

at the *Universiteit van Amsterdam.*

INSTITUTE FOR LOGIC, LANGUAGE AND COMPUTATION

Die Gestalt! sagte er, und Naphta sagte hochtrabender Weise: »Der Logos!« Aber der, welcher vom Logos nichts wissen wollte, sagte: »Die Vernunft!« während der Mann des Logos »die Passion« verfocht. Das war konfus. »Das Objekt« sagte der eine, und der andere: »das Ich!« Schließlich war sogar von »Kunst« auf der einen und »Kritik« auf der anderen Seite die Rede und jedenfalls immer wieder von »Natur« und »Geist« und davon, was das Vornehmere sei, vom »aristokratischen Problem«. Aber dabei war keine Ordnung und Klärung, nicht einmal eine zweiheitliche und militante; denn alles ging nicht nur gegeneinander, sondern auch durcheinander, und nicht nur wechselseitig widersprachen sich die Disputanten, sondern sie lagen in Widerspruch auch mit sich selbst.

Thomas Mann, *Der Zauberberg*

Voor Christine

Abstract

The field of algorithmic randomness is concerned with making precise the intuitive notion of the randomness of individual objects, and is grounded on concepts from computability theory. Not only do different formalisations of irregularity, incompressibility and unpredictability lead to the same class of random binary sequences, they also allow us to compare such sequences on their randomness or their power to find regularities in other sets. Much like the Turing-degrees of computational content, we can define degrees of randomness. A triviality notion with respect to such structures is that of the K-trivial sets, the sets all of whose initial segments are trivial in the sense that they are easily compressible.

This thesis provides a general discussion of algorithmic randomness, as well as original results concerning two topics related to sequences with trivial initial segment complexity. First we apply the classical notion of splitting in the c.e. Turing-degrees to the c.e. degrees of randomness given by the LR-, K- and C-reducibilities. But the main topic is a question by Downey, Miller and Yu about the arithmetical complexity of the function that computes the finite number of K-trivial sets via a given constant. Representing these sets as paths of certain trees, we find a solution to this problem by inspecting the general complexity of calculating the number of paths of trees and reducing the complexity of our particular family of K-triviality trees.

Acknowledgements

I want to thank George for first acquainting me with the area of algorithmic randomness in his January project, the project that led me to pursue this subject further; for embarking on this thesis with me and introducing me to true research, especially during the hard work in the summer break, for pushing me the last months, having started on another master's programme, to make haste with the final write-up, which I now, in December, Amsterdam caught in snow again as it was in January, finally found time for; thus concluding this year of my thesis. Thank you for your committed supervision, for your dedication to the subject, and for the many cappuccino's at the Zeedijk.

Thanks to Charlotte for accompanying me the first half year; I enjoyed the discussions we had before you – unfortunately, for me – decided to finish your thesis in time. Thanks also to my other fellow students. Finally, I would like to thank all nonlogician people around me for not trying to stop me from spending way too much time on a subject of which I could neither convincingly explain to you the nature nor the relevance; in spite of the occasional look of horror on opening one of my textbooks (which I fondly remember), I like to believe that you had some inkling of the beauty I found in it.

Tom Sterkenburg
Amsterdam, December 2010

Contents

Introduction

Randomness is a concept that seems, almost by definition, to evade an exact definition. Examples are easy enough to produce: if we toss a fair coin a number of times in a row and assign 0's to tails and 1's to heads, we unroll a sequence like 1011010111010101111... that is irregular, unpredictable and hard to describe – in short, random. Much unlike sequences as 0000000000000000... and 00110011001100..., that follow a clear pattern and certainly do not look very random. But how to make this difference precise?

Classical probability theory has little to say about randomness of single objects. Any binary sequence like the above of length n has the same probability 2^{-n}, whether it looks random or not. In the general case of infinite sequences, all single objects have probability zero. For the present problem, we will have to shift our attention to the objects themselves, and, one way or another, put a definition of randomness in terms of their characteristics, their overall structure – but the thing is that random objects *lack* any structure. Indeed, we can wonder if any attempt to describe randomness in a formal way would not be contradictory. Does randomness not cease to be randomness if it is captured in a strict definition?

The present thesis is on the subject of *algorithmic randomness*, that developed from the aim to provide just such a definition. In its approach to this problem, the discipline relies on notions of algorithmic effectiveness from *computability theory*[1]. This field, in turn, is directed at formalising what it means for a problem or procedure (function, set, relation) to be *computable* in principle. Its core assumption is the Church-Turing thesis that everything that we would intuitively consider an *algorithm* can be modeled by means of *Turing machines*, some kind of idealised computers.

The main idea of algorithmic randomness is to impose a bound of computability or *effectiveness* on characterisations of randomness. Rather than the problematic demand that a random sequence should have no regularities *at all*, we propose, roughly, that a sequence should be called random if there is no *effective* way, no algorithm, to detect any regularities in it. This general idea has led to different formalisations, based on different features of randomness, calling a sequence random if it passes all effective tests for particular properties, if it cannot be effectively compressed, or if there is no successful effective gambling strategy to predict its bits.

In this thesis, we concentrate on two specific topics in algorithmic randomness, both related to sequences with trivial initial segment complexity. First we apply a particular technique of classical computability theory to structures of computably enumerable *levels* of randomness. Second and most importantly, we look into an

[1]Also commonly known as *recursion theory*. There is some debate about the proper name; see for example [Soa96].

open problem of Downey, Miller and Yu ([DH10, Section 10.1.4], [Nie09, Problem 5.2.16]) about the complexity of a function related to a type of very nonrandom sequences. We present a solution of this question. This work is also the subject of the paper [BS10] of George Barmpalias and me.

Overview. Out of the necessity to pick a starting point somewhere, it is assumed throughout this thesis that the reader is familiar with the fundamentals of computability theory. We will, however, devote our first chapter to a discussion of the background and basic intuitions of algorithmic randomness, covering the main concepts and results from the literature. After a number of preliminary observations, we go into a more detailed examination of the two main formalisations that we will work with in this thesis, those of Martin-Löf randomness and Kolmogorov complexity. We consider the test concept of Martin-Löf, the notion of universal and oracle test, both the plain and the prefix-free variant of Kolmogorov complexity and the corresponding (universal) machines, the Kraft-Chaitin Theorem for building such machines, and Schnorr's Theorem that connects both approaches. Next we look at the levels or degrees of randomness induced by the specific randomness reducibilities C, K, LR and LK, and the corresponding lowness notions of K-triviality, low for K and low for random; we see that these are equivalent.

The next two chapters present original research in two distinct but related directions. Chapter 2 is about splitting constructions in c.e. degrees of randomness, derived from the classical Sacks Splitting Theorem of splitting c.e. Turing-degrees. This classical procedure is given a detailed introduction, before we move to the c.e. LR-degrees, and demonstrate how we can adapt the splitting construction of Sacks to first split a c.e. set in two c.e. sets of strictly lower LR-degree and subsequently in two c.e. sets of incomparable LR-degree. With a single construction we then show how to split c.e. K-degrees in two c.e. K-degrees that are both of incomparable K-degree and strictly K-below the original degree. The same construction passes over to splitting in the c.e. C-degrees. We briefly consider questions of density in these degree structures that arise in this context.

In Chapter 3, we first give the statement and context of the question of Downey, Miller and Yu about the arithmetical complexity of the function that computes the number of K-trivial sets via given constant. The problem is rephrased as counting the number of paths of certain trees, so we introduce concepts such as families of trees and K-triviality trees that we use in our approach. We then direct our attention to basic properties of trees with finitely many paths and the complexity of counting the number of paths in such trees, and later make the generalisation to families of trees. Following this, we embed these complexity results in statements about the jump hierarchy. The next step is to take up the strategy of reducing the complexity of families of trees, starting with Σ_1^0 and Δ_2^0 trees in general, and concluding with the K-triviality trees that we are really interested in. All that remains at that point is finding out how to calculate the low$_2$-ness indices of the reduced K-triviality trees, after which we combine the previous work in establishing a definite complexity of the function concerned, thus answering the open question. Finally, with the same methodology, we also answer the related question about the low for K sets.

CHAPTER 1

Algorithmic randomness

To lay the ground for the rest of the thesis, we give a condensed overview of the area of algorithmic randomness and its connection to computability theory. We discuss the underlying intuition and the necessary technicalities of two of the main approaches to the formalisation of the intuitive concept of randomness of individual sequences, the statistical tests of Martin-Löf and the descriptive complexity of Kolmogorov. This leads us to exact definitions of degrees of randomness, and the associated lowness notions of very nonrandom sets.

More detailed information on both the technical and the philosophical aspects of the field can be found in [Nie09], [DH10], [LV08], [vL87].

1. Randomness and computability theory

The goal that originated the field of algorithmic randomness is to capture the informal notion of randomness. Confining ourselves to objects that are infinite binary sequences (i.e., elements in Cantor space, 2^ω), we seek for a way to formally distinguish intuitively random from nonrandom sequences, so sequences like 01001101100... (that could be obtained by repeated toin cossing) from sequences like 0101010101... (following a simple pattern). We can think of random sequences as the outcome of a trial that produces infinitely many bits, where each bit is independent and has an equal chance of being either a 0 or a 1. However, we want to find a way of judging randomness by looking at the single complete object itself and disregarding how it was produced.

There are several properties that are associated with the informal notion of randomness. An intuitively random sequence is one that is irregular, that lacks any patterns and structure. It should be hard to describe, and impossible to compress significantly. We expect such a sequence to be induced by chance; its development should not be predictable. At the same time, we think of a random sequence as typical in the sense that it satisfies the statistical properties given by probability theory. Another way of putting this is saying that such an object should not have any atypical, exceptional properties – if on a given sequence the number of 1's is not equal in the limit to the number of 0's, this is a rare circumstance and we would not consider the sequence to be random.

Actually, in a first modern approach to random sequences, Von Mises [vM19] tried to base probability theory on random sequences, whose existence he took to be empirical fact.[1] He required such sequences, that he called *Kollektivs*, to satisfy the conditions that the ratio of 1's tends to a limit (embracing the *frequency interpretation* of probability) and that any infinite subsequence taken out by some

[1]An approach very different from that of Kolmogorov, who departed from an abstract axiom system the application of which to real phenomena is justified *a posteriori*.

admissable selection rule shows the same frequency limit (formalising the impossiblity of predicting the occurence of 1's). Later, Church ([Chu40]) proposed to take as these selection rules the computable functions. The attempt failed because the definition still included atypical (hence nonrandom) sequences[2], but we see here the first connection of computability theory to randomness.

More recent proposals were directed at exploiting particular properties such as typicalness (Martin-Löf tests), incompressibility (Kolmogorov complexity) and unpredictability (martingales), and proved to be more fruitful. Their success is supported by the fact that they all isolate the same class of random sequences. The approach of martingales, making precise the infeasibility of a successful gambling strategy on a succession of trials that constitute a random sequence, will receive no further attention in this thesis; but the other two will play an important role and are discussed in more detail in the next two sections.

Both the formalisations of Martin-Löf and of Kolmogorov are grounded on computability theoretic definitions. Moreover, they permit the meaningful subdivision of sets into different degrees of randomness, and there are tight links between the characterisations of information content of sets from the perspective of computational complexity and from that of the level of randomness.[3] The next chapter provides a good example of the application of computability theoretic techniques to hierarchies of randomness.

2. Statistical tests: Martin-Löf randomness

Taking his cue from the provision that random sequences should not possess any atypical features, the idea of Martin-Löf [ML66] was to devise a statistical test for all of these exceptional properties, and call a sequence random if it passes them all. For any such atypical property, the family of sets that satisfies it is a *null class*, a class $\mathcal{A}$ such that $\mu(\mathcal{A}) = 0$ with μ the uniform measure on Cantor space.[4] Having thus identified undesirable properties with null classes, we would like a random sequence to stay clear of them.

A test as conceived by Martin-Löf checks if a given set really does evade the null class for one particular property. Here we can use the fact that $\mathcal{A}$ is a null class precisely if there exists a sequence $(G_m)_{m\in\mathbb{N}}$ of open sets such that $\lim_m \mu(G_m) = 0$ and $\mathcal{A} \subseteq \bigcap_m G_m$, so we have a sequence of classes of decreasing measure, converging to the null class that represents the rare property that is tested. The classes, *critical regions*, are of increasing *confidence level*, and if the set that is being tested happens to be contained in one such critical region, it is rejected with the corresponding confidence. Only if the set is rejected on all confidence levels, we decide that it exhibits the particular property – it cannot be random. If, on the other hand, the set passes all these tests, so is not in any of the null classes they test for, it has no exceptional property and we take it to be random.

But we cannot ask a random sequence to avoid *all* null classes: then no sequence A is random as the singleton $\{A\}$ is null as well. Analogously, the probability of satisfying all properties of probability one is zero. So we have to choose which

[2]In particular, sequences that did not satisfy the *law of the iterated logarithm* ([Vil39]).

[3]Naturally, we can view infinite binary sequences as (the characteristic functions of) sets of natural numbers. We use the notions of infinite sequence and set interchangeably.

[4]The canonical topology is based on basic open sets (or *cylinders*) $[\sigma]$ for $\sigma \in 2^{<\omega}$, where $[\sigma]$ is the class of all infinite extensions of σ. The measure of $[\sigma]$ is $\mu([\sigma]) = 2^{-|\sigma|}$.

statistical anomalies, and hence their corresponding null classes, we want to take into account. Martin-Löf focused with his test concept on *effective* null classes.

To be more precise, he required the sequence of open classes to be uniformly computably enumerable, and the convergence effective. An open class $\mathcal{A}$ is *effectively* open if the basic open cylinders are given by a computably enumerable (c.e.) set of strings, so $\mathcal{A} = [W]$ for some c.e. W. Then, given the standard enumeration $(W_e)_{e \in \mathbb{N}}$ of c.e. sets, a uniformly c.e. sequence $(G_m)_{m \in \mathbb{N}}$ of open sets is accompanied by a computable f such that $G_m = [W_{f(m)}]$ for all m. The convergence is effective if $\mu(G_m) < 2^{-m}$ for each m. The full definition is as follows.

Definition 2.1. A *Martin-Löf test* is a uniformly c.e. sequence $(G_m)_{m \in \mathbb{N}}$ of open sets such that for all $m \in \mathbb{N}$, $\mu(G_m) \leq 2^{-m}$. A given set A fails the test if $A \in \bigcap_m G_m$, otherwise it *passes* the test. If A passes each such test, it is *Martin-Löf random.*

Let MLR be the class of all Martin-Löf (ML) random sets. These sets are sometimes called *1-random*. Since there are only countably many tests, the class of the sets that are intercepted by one of them is still null, which means that the class MLR has measure one. Thus most sets by far are ML-random.

Note that Martin-Löf's formalisation of randomness is a negative one: if any reasonable attempt to pin the set down fails, we just say it must be random. The choice of what is reasonable here is rather pragmatic. Deciding on a certain class of tests reflects a choice of a particular class of properties that we think random objects should possess, and necessarily leaves out others. Computability theory might not be the right intuition to isolate this class; and even so, we still have freedom in choosing the level of effectiveness. For example, the weaker notion of *Schnorr randomness* is based on tests that can only use open sets with computable measure, and for the stronger notion of *2-randomness* the tests are permitted to use the halting set $\emptyset'$ as an oracle. We will not go into these alternative notions of randomness in this thesis. It appears the important statistical properties are satisfied by the definition of Martin-Löf, and it interacts nicely with computability theoretic concepts and formalisations of randomness we will introduce below.

To avoid the hassle of having to work with infinitely many different tests, we can fix a listing $(G^e_k)_{e,k \in \mathbb{N}}$ of all Martin-Löf (ML) tests (indexed by e), where G^e_k is uniformly c.e. in e, k. Let $U_b = \cup_{e \in \mathbb{N}} G^e_{b+e+1}$. Then $(U_b)_{b \in \mathbb{N}}$ is uniformly c.e. and the convergence is effective[5], so it is an ML-test. What is more, if $A \in \cap_m G^e_m$ for a particular test $(G^e_m)_{m \in \mathbb{N}}$, so $A \in G^e_m$ for every m, also $A \in G_b$ for every b, hence $A \in \cap_b U_b$. That means that every A that is not ML-random, and thus intercepted by some test, must be contained in $\cap_b U_b$ as well. Accordingly, a set is not ML-random precisely if it is not in this one particular test, making $(U_b)_b$ a *universal* Martin-Löf test.

Fact 2.2. *There exists a universal Martin-Löf test $(U_b)_b$, meaning that $A \in \mathsf{MLR}$ if and only if $A \notin \cap_b U_b$.*

As the open sets of a test are given by c.e. sets of strings, we can also look at an ML-test as a sequence of machines that output strings. Identifying machine M with its range, a set of strings, an ML-test is a uniform sequence $(M_d)_d$ of machines

[5]We have $\mu(U_b) \leq \sum_e 2^{-(b+e+1)} = 2^{-b-1} \sum_e 2^{-e} = 2^{-b-1} \cdot 2 = 2^{-b}$.

outputting finite binary strings such that $\mu(M_d) < 2^{-d}$ and $M_{d+1} \subseteq M_d$. Then we can also use oracle machines, leading to relativised versions of ML-randomness.

Definition 2.3. An oracle Martin-Löf test $(M_e)_e$ is a uniform sequence of oracle machines outputting finite binary strings, such that for every d and oracle A, $\mu(M_d^A) < 2^{-d}$ and $M_{d+1}^A \subseteq M_d^A$. Set B is *A-random*, $B \in \mathsf{MLR}^A$, if for every oracle ML-test $(M_e)_e$, it holds that $B \notin \cap_e U_e^A$.

Naturally, there also exists a universal oracle ML-test.

3. Descriptive complexity: Kolmogorov complexity

The intuition behind the approach[6] that bears Kolmogorov's name is that a string that is not random must exhibit regularities that allow for a shorter *description* of the string. A way of visualising this is that the description or *code* of the string acts as the input for some program, that can exploit the regularities to retrieve the complete string. The much shorter code can thus be said to capture all information content of the nonrandom string. The patterns in the bits of the string make it *compressible.*

Notice that the foregoing only makes sense for *finite* strings – an infinite random string probably has no finite descriptions at all. But we can say that an infinite sequence is random if all its initial segments are incompressible.

3.1. The plain descriptive complexity. In making this concept precise, we take machines (partial computable functions) M that map strings to strings. Then an M-description for finite string $\sigma \in 2^{<\omega}$ is a code $\tau \in 2^{<\omega}$ such that $M(\tau) = \sigma$. Each machine can be viewed as representing a particular decompression algorithm or *rule*, and can only accept short descriptions for the strings that have a pattern that follow this rule. The length of the shortest M-description of a string is what we call the *Kolmogorov complexity* (relative to M) of the string.

Definition 3.1. The *(Kolmogorov) C_M-complexity* of string $\sigma \in 2^{<\omega}$ relative to machine M is

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

But as before we can construct from the effective list $(M_e)_{e\in\mathbb{N}}$ of all machines an *optimal* or *universal machine* U that simulates all machines and thus combines all rules. Simply define U to feed τ to machine M_e if it receives the universal description $0^{e-1}\tau$ itself. Then $U(0^{e-1}\tau) = \sigma$ precisely if $M_e(\tau) = \sigma$, so the shortest universal description of any σ only differs a constant e from the shortest M_e-description of the same string. This gives us a reason to claim that we have a truly objective notion of the shortest description of a string. We denote the length of the shortest universal description of string σ by $C(\sigma)$.

Now we can say that string σ is *b-incompressible* if its universal description gives a compression of less than b bits, so $C(\sigma) > |\sigma| - b$. But the counterintuitive situation arises that there can be no infinite string all of whose initial segments are b-incompressible for the same constant b, thwarting our plan to define the randomness of sets by the incompressibility of their initial segments. However patternless an infinite sequence may be, there are always *complexity dips* where an initial segment suddenly has a short description. The problem is caused by the fact that the

[6]As developed in [Sol64], [Kol65], [ZL70], [Lev73, Lev76], [Cha75].

information content is not only in the bits of the description (as we intended), but also in the *length* of the description, permitting the machine to "cheat" by using this information as well.

3.2. The prefix-free descriptive complexity. Somehow, we have to get rid of the additional information that is given by the length of a string. We would succeed if we could bring this information back in the bits of the string, by forcing the string to encode its own length. To that end, we restrict our machines by taking away the blank symbol; so the reading head will have to infer from the string itself when it has read the last bit. Such machines are precisely the ones with a *prefix-free* domain, that is, a domain in which no string is the prefix of another. Again we can construct a universal prefix-free machine that simulates all other so-called *prefix-free machines* – let $U(0^e1\tau) \simeq M_e(\tau)$ with $(M_e)_e$ the list of all machines that is modified to make them all prefix-free.[7]

Definition 3.2. The *(prefix-free Kolmogorov) K-complexity* of string $\sigma \in 2^{<\omega}$ is

$$K(\sigma) = \min\{|\tau| : U(\tau) = \sigma\},$$

with U an agreed-upon universal prefix-free machine.

Along with a few other advantages, the prefix-free variant solves the problem of complexity dips. There exist sets all of whose initial segments have high K-complexity – and they happen to be precisely the sets that are random according to Martin-Löf's definition.

Fact 3.3 (Schnorr's Theorem [Sch73]). *Set A is Martin-Löf random if and only if there exists b such that $\forall n\ K(A \upharpoonright n) > n - b$.*

Since the shortest universal description of any string is not more than an index constant e longer than the shortest M_e-description of the same string, $K(\sigma) \leq K_{M_e}(\sigma) + e$, we can construct our own machines to show upper bounds on the K-complexity of strings. For instance, consider the machine N that on input τ looks for a decomposition in two strings ρ and σ such that the second has length n, so $|\sigma| = n$, and the first is a universal description for n, so $U(\rho) = n$. If it finds such a decomposition, it outputs the second part σ. Now N is prefix-free because U is, and if ρ is a shortest universal description for $n = |\sigma|$, then $\rho\sigma$ is an N-description for σ; hence $N(\rho\sigma) = \sigma$. It follows that $K(\sigma) \leq K(|\sigma|) + |\sigma| + d$ for any string σ, where d is the index of N.[8]

This upper bound can be further improved by looking at the machine M with $M(0^{|\sigma|}1\sigma) = \sigma$, which is prefix-free because σ and τ have to be equal if $0^{|\sigma|}1\sigma \preccurlyeq 0^{|\tau|}1\tau$. It implies that $K(\sigma) \leq 2|\sigma| + b$ for some b and all σ, so in particular $K(|\sigma|) \leq 2\log|\sigma| + b$. Substituting this in the inequality of the previous paragraph, we obtain the following bound.

[7]For this modification, only accept the computation $M_e(\tau)[s] = \sigma$ at stage s if it is not the case that $M_e(\rho)$ is defined at an earlier stage for some ρ such that $\rho \succcurlyeq \tau$ or $\rho \preccurlyeq \tau$. Also note that the additional 1 in $0^e1\tau$ makes sure that the domain of U is prefix-free.

[8]Whenever we talk about the Kolmogorov complexity of a natural number, we mean the complexity of the string that respresents it. An efficient representation would be by an ordering of all strings first by length and then lexicographically. Then the length of the representing string of n is approximately $\log n$.

Fact 3.4. *For some constant c,*

$$\forall\sigma \in 2^{<\omega}\ (K(\sigma) \leq 2\log|\sigma| + |\sigma| + c).$$

The next important result will provide us with a convenient way of constructing prefix-free machines. First we note that the measure of the open set generated by the domain of a prefix-free machine is bounded. We call this measure the *weight* of the machine.[9]

Definition 3.5. The *weight* of prefix-free machine M is

$$\sum_{\sigma\in\mathrm{dom}(M)} 2^{-|\sigma|}.$$

Since prefix-free M has bounded weight, clearly also $\sum_\sigma 2^{-K_M(\sigma)} \leq 1$. The observation that any prefix-free machine has bounded weight can be turned around. If we have an infinite sequence of natural numbers n_i such that the sum of all 2^{-n_i} is bounded, we can infer the existence of a corresponding prefix-free sequence of strings with exactly these lengths. A useful way to look at this is in terms of *request* pairs of lengths and strings, where we want the string to get a description of the associated length. The following theorem, also known as the *Machine Existence Theorem*, effectivises this observation into a useful tool for constructing prefix-free machines.

Fact 3.6 (Kraft-Chaitin Theorem [Kra49],[Lev73],[Cha75]). *For a c.e. set R of requests $\langle n,\sigma\rangle \subseteq \mathbb{N}\times 2^{<\omega}$ such that*

$$\sum_{\langle n,\sigma\rangle\in R} 2^{-n} \leq 1,$$

we can effectively obtain a prefix-free machine M such that for all lengths n and strings σ

$$\langle n,\sigma\rangle \in R \iff \exists\tau\ (|\tau| = n\ \&\ M(\tau) = \sigma).$$

Moreover, an index d of M can be effectively obtained from the index of the request set R.

As an example, the set of requests $\langle 2\log n + 2, n\rangle$ for all nonzero $n \in \mathbb{N}$ has weight $\sum_n 2^{-2\log n - 2} = 1/2\sum_n 1/n^2 \leq 1$, and even leaves room for a description of a few bits for $n = 0$, so there must be a machine that recognises a description of length $2\log n + 2$ for each n.

Fact 3.7. *For some constant c,*

$$\forall n \in \mathbb{N}\ (K(n) \leq 2\log n + c).$$

In the rest of the thesis, the Kraft-Chaitin Theorem is silently invoked whenever we build prefix-free machines by enumerating descriptions and have to watch that we do not add too much weight. In addition, by the Recursion Theorem we can assume we know the index of the c.e. request set and hence of the machine in advance, which is of course useful because it is the constant by which the length of the shortest universal descriptions may differ.

[9]The weight of M can be interpreted as its *halting probability*. In an experiment where we start machine M and toss a coin to determine the next input bit whenever M asks for it, this is the probability that M halts, that is, the probability that the string we feed it is in its domain. Chaitin's famous random real Ω is given by the halting probability of an agreed-upon universal machine.

4. Degrees of randomness

Much like Turing or m-reduciblity hands us a way of comparing sets to their computational content, our formalisations of randomness point to reduciblities that make it meaningful to consider one set to be more random than another, or more powerful in detecting patterns.

4.1. Reducibilities.

If all initial segments of one set have a descriptive complexity that is not significantly greater than the descriptive complexity of those of another set, we are in a good position to say that the second set is at least as random as the first. This is formalised in the notions of K- and C-reducibility.

Definition 4.1 ([DHL04]). Set A is K-reducible to set B, and we write $A \leq_K B$, if there exists a constant b such that

$$\forall n \in \mathbb{N}\ (K(A \upharpoonright n) \leq K(B \upharpoonright n) + b).$$

Likewise, $A \leq_C B$ if

$$\forall n \in \mathbb{N}\ (C(A \upharpoonright n) \leq C(B \upharpoonright n) + b).$$

We can also compare sets according to their strength as an oracle in making possible short descriptions. Then one set is at least as powerful as another if it allows for descriptions that are not signicantly longer – it is as least as useful in compressing strings.

Definition 4.2 ([Nie05]). Set A is LK-reducible to set B, and we write $A \leq_{LK} B$, if there exists a constant b such that

$$\forall \sigma \in 2^{<\omega}\ (K^B(\sigma) \leq K^A(\sigma) + b).$$

Invoking the definition of Martin-Löf randomness, we can compare the strength of oracles on their ability to "derandomise" whole sets, decreasing the class of sets that are still random.

Definition 4.3 ([Nie05]). Set A is LR-reducible to set B, and we write $A \leq_{LR} B$, if

$$\mathsf{MLR}^B \subseteq \mathsf{MLR}^A.$$

Actually, the LR- and LK-reduciblities are equivalent: $A \leq_{LK} B$ precisely if $A \leq_{LR} B$. They are *weak reducibilities* in the sense that they are implied by Turing reducibility. This does not hold for the K-reducibility: we may have $A \leq_T B$ but $A \not\leq_K B$.

Two sets are equivalent with respect to a particular reducibility if they reduce to each other. The class of all sets can be partitioned in *degrees* of sets that are equivalent in this sense. So sets A and B are K-*equivalent*, and we write $A \equiv_K B$, if both $A \leq_K B$ and $B \leq_K A$; and a K-degree is a class of sets that is closed under K-equivalence. Likewise for the C-, LK- and LR-reducibilities. Thus we have what we could call levels of randomness. There is always a lowest level or degree, the one that contains the computable sets.

4.2. Lowness notions.

The least degrees induced by the reducibilities we discussed are classes of sets that are very nonrandom, or as weak as can be in compressing and derandomising.

We start with the sets in the lowest K- and C degrees, the sets whose initial segments have a Kolmogorov complexity not more (up to a constant) than that of the string representing their lengths. Equivalently, their initial segments are not more complex than those of any computable set.

Definition 4.4 ([Sol75]). Set A is *K-trivial* if there exists a constant b such that

$$\forall n \in \mathbb{N}\ (K(A \upharpoonright n) \leq K(n) + b).$$

A is *C-trivial* if there is b such that $\forall n \in \mathbb{N}\ (C(A \upharpoonright n) \leq C(n) + b)$.

Both definitions obviously encompass all computable sets. It appears that the class of C-trivials coincides with the computable sets (Meyer, unpublished). This is not the case for the class $\mathcal{K}$ of K-trivial sets: there are incomputable K-trivials [Sol75].

The lowest degree in the LK-degrees consists of the sets that are worst at compressing.

Definition 4.5 (Muchnik, unpublished). Set A is *low for K* if there exists a constant b such that

$$\forall \sigma \in 2^{<\omega}\ (K(\sigma) \leq K^A(\sigma) + b)).$$

Finally, the lowest degree in the LR-degrees is characterised as follows.

Definition 4.6 ([Zam90], [KT99]). Set A is *low for ML-randomness* if

$$\mathsf{MLR}^A = \mathsf{MLR}.$$

We denote the classes of low for K's and low for randoms by $\mathcal{M}$ and $\mathcal{L}$, respectively.

The combined efforts of several authors have uncovered that the main lowness properties we have just seen are in fact equivalent. The classes $\mathcal{K}$, $\mathcal{M}$ and $\mathcal{L}$ consist of the very same sets.

Fact 4.7 ([Nie05], [HNS07]). *Set A is K-trivial if and only if it is low for K if and only if it is low for random.*

It points at a rather striking link between the properties of being computationally weak and being far from random, i.e., having trivial initial segment complexity.

CHAPTER 2

Splitting in degrees of randomness

The Sacks Splitting Theorem is a standard result of computability theory that states that any noncomputable c.e. set can be split into two disjoint sets that do not compute another given set. The general method of Sacks restraints introduced in the construction of this splitting can be used to prove similar results in different degree structures. In this chapter, we will apply it to split nontrivial c.e. sets within degrees of randomness, induced by the reducibilities that we saw in the introduction.

We start by giving a detailed account of the classical Sacks Splitting Theorem. All further splitting constructions in this chapter will draw from that presentation, allowing us to focus on the details that differ.

The c.e. degree structures we then look at are those of the *LR*-, *K*- and *C*-degrees, respectively. For all these structures with their corresponding interpretations of complexity, we show how to split a nontrivial set into two sets of strictly lower complexity, as well as into two sets that have incomparable complexity. With the exception of the case of the *LR*-degrees, these two requirements are in fact equivalent.

The splitting theorems answer the question of *downward density* in the corresponding c.e. degree structures. We briefly discuss related problems of density in these structures.

1. The classical splitting theorem

The construction in the proof of the *Sacks Splitting Theorem* [Sac63] divides a given c.e. but noncomputable set A into two disjoint sets that both are not in the Turing-cone[1] above a certain set B. We will simplify this to the case $A = B$, so we will look at a construction that divides a given nontrivial c.e. set A into two disjoint sets that both contain strictly less information than A.

1.1. Splitting into sets of strictly lower degree.

The following presentation is based on [Soa10],[Coo04].

Theorem 1.1. *For c.e. $A >_T \emptyset$, there exist c.e. A_0 and A_1 such that $A_0 \sqcup A_1 = A$ and $A_0, A_1 <_T A$.*

PROOF. We are given a computable enumeration $\{A[s]\}_{s\in\mathbb{N}}$ of A such that (without loss of generality) one new element is enumerated at each stage. This element we have to put in either A_0 or A_1, in such a way that we satisfy the *requirement*

$$R_{\langle e,i\rangle} : A \neq \Phi_e^{A_i}$$

for all e and for $i = 0, 1$, making sure that A cannot reduce to A_0 or A_1.

[1]The (Turing-)cone above a set X is the class of all sets that compute X.

At each stage s and for each index e we have a *length of agreement* between $\Phi_e^{A_i}$ and A, given by

$$l(e,i)[s] = \mu x(A[s] \restriction x \neq \Phi_e^{A_i}[s] \restriction x).$$

A requirement $R_{\langle e,i\rangle}$ is only endangered if the length of agreement

$$l(e,i) = \lim_s l(e,i)[s]$$

goes to infinity, but we actually try to protect the agreement by putting a *restraint* on the maximal use of A_i in computations within the length of agreement:

$$r(e,i)[s] = \mu x(\Phi_e^{A_i \restriction x}[s] \restriction l(e,i)[s] \downarrow).$$

The trick is that reaching an infinite length of agreement monitored by these *Sacks restraints*, which is the only way the requirement can be evaded, will give us a sure way of computing A, contrary to assumption.

CONSTRUCTION. Start with $A_i[0] = \emptyset$ for $i = 0, 1$. At stage $s+1$, we have a new $x \in A[s+1] - A[s]$. If x is not below any restraint $r(e,i)[s]$, put x into $A_0[s+1]$. Otherwise, take the least $\langle e,i\rangle$ such that $x \leq r(e,i)[s]$. This is the restraint with the highest *priority*. We will respect this restraint, and enumerate x not in A_i but in the other set, A_{1-i}. Now it might be the case that x is below restraints $r(e', 1-i)[s]$ for higher $\langle e', 1-i\rangle > \langle e,i\rangle$. Then we say the requirement $R_{\langle e',1-i\rangle}$ is *injured* by this x.

VERIFICATION. We will show by induction that each requirement is satisfied. For this we need to show at the same time that a requirement is injured only finitely often and that the restraints will settle. So to show this for $\langle e,i\rangle$, suppose that for all $\langle e',i'\rangle < \langle e,i\rangle$, $R_{\langle e',i'\rangle}$ is injured only finitely often, $A \neq \Phi_{e'}^{A_{i'}}$, and $r(e',i') = \lim_s r(e',i')[s]$ exists and is finite.

We start by proving that $R_{\langle e,i\rangle}$ is injured finitely many times. Wait for a stage t such that all restraints $r(e',i')$ have settled, and choose an r greater than any of them. Then wait for stage $v > t$ where the initial segment of A up to r has settled: $A[v] \restriction r = A \restriction r$. Now no x below any $r(e,i)[w]$ with $w \geq v$ can enter A_i, because in that case some higher priority $R_{\langle e',1-i\rangle}$ would have forced x to injure $R_{\langle e,i\rangle}$. But then x would be below the upper bound r on all higher priority restraints, and could not appear into A in the first place because by this stage $A \restriction r$ has settled. Thus $R_{\langle e,i\rangle}$ cannot be injured from stage v on.

Now for the main part, the verification of requirement $R_{\langle e,i\rangle}$. Take any number n and suppose instead that $A = \Phi_e^{A_i}$. Then the length of agreement $l(e,i) = \infty$, so there must be some stage s such that not only $R_{\langle e,i\rangle}$ is respected from then on but also $l(e,i)[s] > n$. As the restraints will not be injured, the length of agreement cannot decrease unless at some stage $s' > s$ an element $x < l(e,i)[s']$ is enumerated into A, meaning $A[s'+1] \restriction x \neq A[s] \restriction x = \Phi_e^{A_i}[s] \restriction x$. But then the restraints will make sure this disagreement is preserved forever, and we end up with $A \neq \Phi_e^{A_i}$, contradicting our hypothesis. Thus the length of agreement cannot decrease from s on, and to see whether $n \in A$ it suffices to check if $n \in \Phi_e^{A_i}[s]$, a computable operation. As we took A to be incomputable, we conclude that $A \neq \Phi_e^{A_i}$.

Note that the restraints not only try to protect agreements in order to provide a way of computing A in case an agreement will not stop growing, but, just as importantly, also preserve a disagreement caused by a sufficiently small new element in A.

To make the induction work, we still have to show that the restraint $r(e,i)$ will settle. By our result that $A \neq \Phi_e^{A_i}$, there is a least x such that $A(x) \neq \Phi_e^{A_i}(x)$. So from some stage on, the initial segments of A and $\Phi_e^{A_i}$ up to but not including x have settled and are the same. Now wait for a stage s such that also $A(x)[s] = A(x)$, and, by the first result, $R_{\langle e,i\rangle}$ will not be injured anymore. If $\Phi_e^{A_i}(x)[t]$ is divergent at every stage $t \geq s$ from then on, the length of agreement will stay at x. Then the computations for all inputs up to x will be preserved since the corresponding restraint cannot be breached anymore, so $r(e,i)[s]$ will stay put and $r(e,i) = r(e,i)[s]$. Otherwise, if $\Phi_e^{A_i}(x)[t]\downarrow$ at a later stage $t \geq s$, we know that this computation will also be preserved because the length of agreement then includes x. Since $\Phi_e^{A_i}(x)[t] = \Phi_e^{A_i}(x) \neq A(x)$, the length of agreement has stabilised and the restraint $r(e,i)[t] = r(e,i)$ has settled in this case as well. □

1.2. Splitting into sets of incomparable degree. The fact that c.e. A is not Turing-reducible to one of its disjoint c.e. halves A_0 and A_1 is equivalent to the fact that A_0 and A_1 are Turing-incomparable, i.e., that we have both $A_0 \not\leq_T A_1$ and $A_1 \not\leq_T A_0$ (denoted as '$A_0|_T A_1$'). For this reason, the splitting parts cannot be computable either, and must be of a degree strictly between the computable degree and that of A.

Proposition 1.2 (Folklore). *For splitting $A = A_0 \sqcup A_1$ of c.e. A into disjoint c.e. A_0 and A_1,*

$$A_0|_T A_1 \iff A_0, A_1 <_T A.$$

Proof. We make use of the fact that A is of the same Turing-degree as the join $A_0 \oplus A_1$. To see this, note that $A_0, A_1 \leq_T A$ because to establish whether $x \in A_0$ (say) we check if $x \in A$, and if so, we can enumerate c.e. A_0 and A_1 until x appears in one of them. Hence $A_0 \oplus A_1 \leq_T A$. It is immediate that $A = A_0 \sqcup A_1 \leq_T A_0 \oplus A_1$, and the equivalence follows.

($\Longrightarrow$) If A would reduce to (say) A_0 we would have $A_0 \oplus A_1 \leq_T A_0$, hence by $A_1 \leq A_0 \oplus A_1$ it follows that $A_1 \leq A_0$. So they are comparable. Likewise for $A \leq_T A_0$.

($\Longleftarrow$) If A_0 and A_1 are comparable, say $A_0 \leq_T A_1$, then $A_1 \equiv_T A_0 \oplus A_1 \equiv_T A$ (the join being the supremum), so in particular, $A \leq_T A_1$. Likewise for $A_1 \leq_T A_0$. □

Even though in the Turing case these two facts are equivalent, we lose this property if we later move to splitting in the c.e. LR-degrees. Therefore we also look at a construction that is explicitly directed at splitting A into two incomparable sets. We define the lengths of agreement as

$$l(e,i)[s] = \mu x(A_{1-i}[s] \upharpoonright x \neq \Phi_e^{A_i}[s] \upharpoonright x)$$

and the restraints $r(e,i)[s]$ just as before, and try to satisfy for all e and $i = 0,1$ the requirement

$$R_{\langle e,i\rangle} : A_{1-i} \neq \Phi_e^{A_i}.$$

Construction. Again starting with $A_0[0] = A_1[0] = \emptyset$, at any new stage $s+1$ we look for the least $\langle e,i\rangle$ such that the new $x \in A[s+1] - A[s]$ is not larger than the length of agreement $l(e,i)[s]$ or the restraint $r(e,i)[s]$. If none is found, just put x into A_0. If we do find such a least $\langle e,i\rangle$, enumerate x into A_{1-i} instead. We now look at both the restraint and the length of agreement, and say that $R_{\langle e',i'\rangle}$ is injured if some x below $l(e',i')[s]$ or $r(e',i')[s]$ is put into $A_{i'}$.

VERIFICATION. Inductively assume that for all $\langle e',i'\rangle < \langle e,i\rangle$, $R_{\langle e',i'\rangle}$ is injured only finitely often, $A \neq \Phi_e^{A_i}$, and $r(e',i')$ exists and is finite.

The requirement $R_{\langle e,i\rangle}$ is injured only finitely often by a similar argument as before, but now also applied to the length of agreement. So we take an r beyond all higher priority restraints and lengths of agreements, wait for A up to r to settle, and argue that any x injuring $R_{\langle e,i\rangle}$ after this stage must do so by directions of a higher priority requirement, meaning that x is below r and would change the part of A we assumed already settled.

To verify requirement $R_{\langle e,i\rangle}$, assume that $A_{1-i} = \Phi_e^{A_i}$, so $l(e,i) = \infty$. Given any n, wait for the first stage s after which the requirement is never injured and the length of agreement exceeds n. As in this case also elements within the length of agreement of $\Phi_e^{A_i}$ and A_{1-i} are disallowed, no element up to n can be put into A_i. But neither can any $x \leq n$ be put into A_{1-i}, because as before that would create a disagreement that is preserved and results in $A_{1-i} \neq \Phi_e^{A_i}$. Hence, as we must put such elements in one of A_0 or A_1, nothing up to n will be enumerated into A from this stage on. Thus $n \in A$ precisely if $n \in A_0[s] \cup A_1[s]$, making A computable. This is a contradiction, so it must be the case that $A_{1-i} \neq \Phi_e^{A_i}$.

The argument that the length of agreement and restraint will stabilise on a finite value is identical to the one in the previous construction. □

2. Splitting in the c.e. LR-degrees

Now that we have examined splitting in the Turing degrees, we will look at ways of splitting a nontrivial set in the LR-degrees. Recall from the introduction that we have $A \leq_{LR} B$ if $\mathsf{MLR}^B \subseteq \mathsf{MLR}^A$, and the fact that it is implied by Turing-reducibility. As members of the 'nontrivial' degree we now take sets that are not low for random, so the nontrivial sets are those strictly LR-above the computable sets.

The definition of LR-reducibility does not allow for a straightforward translation of the devices we used in the splitting in the c.e. Turing degrees. If we want to prevent that $A \leq_{LR} B$, how can we effectively define a length of agreement that measures how close we are to $\mathsf{MLR}^B \subseteq \mathsf{MLR}^A$? Therefore we use a more convenient characterisation of this reducibility. The following equivalence was first established in [KH07], and given a more direct presentation and proof in [BLS08a]. Here we look at ML-test members as c.e. sets of strings, as explained in the introduction.

Fact 2.1 ([KH07]). *For all sets A and B, the following are equivalent:*
(1) $A \leq_{LR} B$;
(2) for some member U^A of a universal ML-test relative to A, there is a V^B c.e. in B such that $\mu(V^B) < 1$ and $U^A \subseteq V^B$.

Intuitively, the length of agreement with respect to $A \leq_{LR} B$ and a particular V^B would identify (under the assumption of a fixed listing of elements) up to what point elements in U^A are in V^B, and the restraint is put on the use of B by V in calculating these elements.

In the remainder of this section, we first review the result in [BLS08a] of splitting a nontrivial c.e. set into two c.e. sets of strictly lower LR-degree. As this does not imply that the thus obtained sets are also of incomparable LR-degree, we will introduce a different construction to achieve that. Finally, we have a brief look on how these splittings relate to the questions of density.

2.1. Splitting into sets of strictly lower c.e. LR-degree. In [BLS08a], an adapted splitting argument is presented that divides a nontrivial c.e. set into two c.e. sets of strictly less information in the LR-sense, that is, that are worse at derandomising than the original set. The construction is identical to the classical one, except for the definition of the restraints and lengths of agreements. The latter are protected to make sure that any broken requirement will show the original set to be nontrivial, but this time not in the sense of being computable but in the sense of being low for random.

Theorem 2.2 ([BLS08a, Theorem 5.5]). *For c.e. $A >_{LR} \emptyset$, there exist c.e. A_0 and A_1 such that $A_0 \sqcup A_1 = A$ and $A_0, A_1 <_{LR} A$.*

Proof. This time, we want to distribute elements enumerated into A over A_0 and A_1 in such a way that we satisfy

$$R_{\langle e,i\rangle} : U^A \not\subseteq V_e^{A_i}$$

for all e and for $i = 0, 1$. Here U is a universal oracle Martin-Löf test, and $(V_e, q_e)_{e\in\mathbb{N}}$ an effective enumeration of tuples (V, q) with V a c.e. operator and q a dyadic rational[2] such that $\mu(V^X) < 1 - q$ for all sets X. If we can fulfill every such requirement, the equivalence of Fact 2.1 tells us that A is not LR-reducible to A_0 or A_1.

Our new length of agreement has to express up to what point elements in U^A are in $V_e^{A_i}$. To define it, we use a computable enumeration (σ_i) of finite strings appearing in $U^A[t]$ at some stage, where each σ is enumerated once in this list for each time it appears in some $U^A[t]$ with new use. A string σ_s that is enumerated as stage t and is still in $U^A[t']$ with the same use at a later stage t', is said to have *remained* in U^A. Then $l(e,i)[s]$ gives the maximum n such that the nth member of the sequence is enumerated by stage s and all σ_i for $i \leq n$ are in $V_e^{A_i}[s]$ or have not remained in U^A.

The restraint on the use of A_i is given by

$$r(e,i)[s] = \mu x\ \forall i \leq l(e,i)[s]\ (\sigma_i \in U^A[s] \ \Rightarrow\ \sigma_i \in V_e^{A_i\restriction x}[s]).$$

Note that in this definition of length of agreement, anything new appearing in U^A will be enumerated anew and will thus be greater than (have an index greater than) the current length of agreement. In other words, new elements in U^A cannot breach the length of agreement. The restraints' only purpose is in protecting the agreement, and we do not need it to keep a sudden disagreement as before.

Construction. We do exactly the same as in the classical splitting construction: for new $x \in A[s+1] - A[s]$ we look for the highest priority restraint $r(e,i)[s]$ above x and put x in A_{1-i}. If none is found, x goes into A_0.

Verification. The inductive verification of the facts that the requirements are injured only finitely often and that the restraints will settle is essentially the same as in the classical case. For the first, suppose all $R_{\langle e',i'\rangle}$ with $\langle e',i'\rangle < \langle e,i\rangle$ will

[2]A *dyadic rational* can be written in the form $z2^{-n}$ for some integer z and natural number n. So they are the real numbers with a finite binary expansion.

be respected from some stage on, and choose an r greater than all the corresponding restraints. After the stage where $A \upharpoonright r$ has settled, no x below the current restraint for $\langle e, i\rangle$ can enter A_i or it would have been enforced by a higher priority $R_{\langle e', i'\rangle}$ and $x \leq r$ changes $A \upharpoonright r$ after all. For the second, under the assumption that $U^A \not\subseteq V_e^{A_i}$ let σ_n be the first string in our enumeration that remains in U^A with the same use and that is not in $V_e^{A_i}$. Beyond the stage where each σ_i for $i < n$ has not remained in U^A or the part of A_i that is used by V in calculating it has settled, and the requirement $R_{\langle e,i\rangle}$ is no longer injured, σ_n cannot appear in $V_e^{A_i}[s]$ anymore. For if it does, the length of agreement $l(e,i)[s]$ is put to n, and the restraint protects the calculation of σ_n, making us end up with $\sigma_n \in V_e^{A_i}$ after all. And if it does not, clearly the length of agreement and also the restraint will move no more.

To verify $R_{\langle e,i\rangle}$, assume to the contrary that $U^A \subseteq V_e^{A_i}$. Then $l(e,i)$ goes to infinity and we can find a stage s such that the restraints $r_{s'}(e,i)$ will not be injured for stages s' after s. Then we can enumerate a set of strings E by putting finite string σ_i into it at stage s' if $i \leq l(e,i)[s']$ and $\sigma_i \in U^A[s']$. Since V^{A_i} is protected by the restraints on A_i, by its definition the length of agreement cannot decrease (it does not matter if anything leaves U^A, nor if anything is added). So if U^A is contained in $V_e^{A_i}$ then everything that appears in U^A will be enumerated into E and the restraints guarantee that everything in E will also be in $V_e^{A_i}$. Thus $U^A \subseteq E \subseteq V_e^{A_i}$ and $\mu(E) \leq \mu(V_e^{A_i}) < 1 - q$, so as E is c.e., we have $A \leq_{LR} \emptyset$. But this contradicts our assumption that A is not low for random. $\square$

2.2. Splitting into sets of incomparable c.e. *LR*-degree. In the *LR*-degrees, the join operator '$\oplus$' does not determine a supremum [Nie05]. So the proof of Proposition 1.2 breaks down in this degree structure, and we cannot claim that a c.e. set that is not low for random can be split into two *LR*-incomparable sets just by the result of the previous section. That is why we present here a construction that achieves this explicitly.

Theorem 2.3. *For c.e. $A >_{LR} \emptyset$, there exist c.e. A_0 and A_1 such that $A_0 \sqcup A_1 = A$ and $A_0 |_{LR} A_1$.*

Proof. The lengths of agreement and restraints are defined like in the previous theorem, and the requirements we must satisfy are given by

$$R_{\langle e,i\rangle} : U^{A_{1-i}} \not\subseteq V_e^{A_i}.$$

Construction. We can again keep to the strategy in the corresponding Turing case. So for new x at stage $s+1$, we try to find the highest priority $R_{\langle e,i\rangle}$ such that x is not above the length of agreement $l(e,i)[s]$ or the restraint $r(e,i)[s]$, and put x in A_{1-i}.

Verification. We look at the verification of $R_{\langle e,i\rangle}$, the rest of the argument being the same as before.

If $l(e,i) = \infty$ then there is a stage s where the requirement will be respected and the length of agreement exceeds a given n. The length of agreement is preserved as before, and now nothing below it can be put into A_i. In other words, every new element up to the length of agreement that appears in A must be put into A_{i-1}. So n is in A precisely if it is in $A_i[s]$ or A_{1-i}. That makes A computable in A_{1-i}.

Now after reaching s we build a c.e. set of finite strings E exactly as we did in the previous construction, enumerating σ_i at stage s' if $i \leq l(e,i)[s']$ and $\sigma_i \in U^{A_{1-i}}[s']$. Again $U^{A_{1-i}} \subseteq E \subseteq V_e^{A_i}$ and $\mu(E) < 1-q$, so $A_{1-i} \leq_{LR} \emptyset$ and A_{1-i} is low for random. But we have just demonstrated that A is A_{1-i}-computable, so A cannot have more information than A_{1-i} already has. That implies that A is low for random as well, and we have reached our contradiction. □

This is actually a stronger version of Theorem 2.2, because if the splitting parts are LR-incomparable they must also be strictly LR-below the original set. After all, $A_0, A_1 \leq_{LR} A$ and if also $A_0 \geq_{LR} A$ then we would end up with $A_1 \leq_{LR} A_0$.

2.3. Density in the c.e. LR-degrees. We call a degree structure *downward dense* if for any given nonminimal degree, there is always a third degree to be found strictly between that degree and the lowest degree. The construction of the previous section, providing a method of obtaining a nontrivial splitting part that is strictly below the original set, shows that the c.e. LR-degrees do have this property.

Corollary 2.4 (from Theorem 2.3)**.** *The c.e. LR-degrees are downward dense.*

Proof. Given any nonminimal LR-degree, each member A can be split with the construction of Theorem 2.3 into LR-incomparable c.e. A_0 and A_1. None of these two sets can be LR-reducible to $\emptyset$, for if $A_0 \leq_{LR} \emptyset$ then certainly $A_0 \leq_{LR} A_1$, contrary to $A_0|_{LR}A_1$. And both are LR-below A, as they are computable in A and Turing-reducibility implies LR-reducibility; but they cannot be of the same LR-degree as A, since (as noted before) $A_1 \equiv_{LR} A$ again implies $A_0 \leq_{LR} A_1$. So A_0 and A_1 are of degrees that are strictly between the lowest degree and the degree we started with. □

The next question that presents itself is that of general density. Given any two degrees, one strictly above the other, is there a degree strictly between them? The Sacks Density Theorem [Sac64] gives an affirmative answer for the c.e. Turing degrees. For the c.e. LR-degrees only a weaker result is known.

Fact 2.5 ([BLS08b])**.** *For c.e. sets B and C with $B \leq_T C$ and $B <_{LR} C$, there is c.e. A such that $B \leq_T A \leq_T C$ and $B <_{LR} A <_{LR} C$.*

So this theorem requires the first degree to be computable in the second, which is is no way enforced by the fact that the first LR-reduces to the second. The general problem is still unsolved and in fact one of the open questions in [MN06].

Question 2.6. Are the c.e. LR-degrees dense?

Then there is the question of *upward density*. Depending on whether the degree structure has a *maximal* degree that has no degrees above it, it either asks if there is always a degree strictly between any given degree and this maximal degree, or whether for every degree there is always a degree higher than it. Every c.e. Turing-degree is below the complete degree $\mathbf{0}'$, so the c.e. Turing-degree structure has a *maximum* element. From the general density then follows that the Turing-degrees are upward dense.

This property passes over to the c.e. LR-degrees. Consider the LR-degree containing the complete c.e. sets (there is one single degree containing all these sets, as $A \equiv_T$ implies $A \equiv_{LR} B)$[3]. Any c.e. set is computable in a complete set, hence LR-reducible to a set in this degree. Thus it is a maximum degree, and there can be no degree strictly above it.

Fact 2.7. *There is a maximum c.e. LR-degree.*

The weaker density result above then suffices to show that there is always a c.e. LR-degree strictly between any given c.e. LR-degree and the maximum c.e. LR-degree.

Corollary 2.8 (From Theorem 2.5). *The c.e. LR-degrees are upward dense.*

PROOF. Take any c.e. LR-degree $\mathbf{a}$ that is strictly below the maximum c.e. LR-degree. Pick an arbitrary c.e. $A \in \mathbf{a}$. This set is certainly computable in $\emptyset'$, member of the maximum degree. Then by Theorem 2.5 we can find a set B such that $A <_{LR} B$ yet $B <_{LR} \emptyset'$. We conclude that there is a c.e. LR-degree strictly between $\mathbf{a}$ and the maximum degree. □

3. Splitting in the c.e. K- and C-degrees

Finally we look at the structures of the K- and the C-degrees, that classify sets according to their (prefix-free) initial segment complexity. We start with the prefix-free case, where we show how to split c.e. sets that surpass the lowest degree of sequences with trivial prefix-free initial segment complexity, the K-trivials. Recall that this is really the same class we saw in the previous section, that of the low for randoms. Essentially the same construction provides for splitting in the c.e. C-degrees. Here the trivial sets are precisely the computable ones.

Contrary to the LR-case, for the K- and C-splitting we do not need to introduce separate constructions for splitting into sets of incomparable degree and sets of strictly lower degree.

Lemma 3.1. *For splitting $A = A_0 \sqcup A_1$ of c.e. A into disjoint c.e. A_0 and A_1,*

$$A_0|_K A_1 \Longrightarrow A_0, A_1 <_K A,$$

and

$$A_0|_C A_1 \Longrightarrow A_0, A_1 <_C A.$$

PROOF. It follows from the observation that A_0 and A_1 are identity bounded Turing-reducible to A that they are K-reducible to A. Indeed, to determine if x is in (say) A_0, we can check if $x \in A$. If so, we know it is in one of the disjoint parts A_0 and A_1; and we can computably enumerate both of them until x appears in one. That means that $A_i \upharpoonright n$ can be described using $A \upharpoonright n$, so the initial segment complexity of the former is no more than that of the latter, up to a constant. Hence $A_i \leq_K A$. And if A_0 and A_1 are K-incomparable then certainly not $A \leq_K A_i$ because that would imply $A_0 \leq_K A_1$ (and vice versa).

It follows in the same way from $A_0, A_1 \leq_{ibT} A$ that $A_0, A_1 \leq_C A$. So if $A_0|_C A_1$ then $A \not\leq_C A_0, A_1$ just as in the prefix-free case. □

[3]This degree contains even more sets, as there exists a c.e. incomplete A with $A \equiv_{LR} \emptyset'$ [Nie05].

3.1. Splitting in the c.e. K-degrees. In the construction of Theorem 2.2, the splitting into two strictly LR-lower sets, we can directly infer that the splitting parts must also be strictly Turing-below the original set. For if it were the case that $A_i \equiv_T A$, then (LR-reducibility being implied by Turing reducibility) we would end up with $A_i \equiv_{LR} A$ after all. But K-reducibility does not follow so easily, so we are not allowed to make this simple step in the present case. Instead, we execute the familiar Turing-splitting construction in parallel with the new splitting procedure that makes sure the new sets are of incomparable initial segment complexity.

Theorem 3.2. *For c.e. $A >_K \emptyset$, there exist c.e. A_0 and A_1 such that $A_0 \sqcup A_1 = A$, and*

- $A_0, A_1 <_T A$;
- $A_0 |_K A_1$;
- $A_0, A_1 <_K A$.

Proof. In the course of enumerating the elements of A into either A_0 or A_1, we now have two requirements to satisfy. For all $e \in \mathbb{N}$ (interpreted as functional index and as constant for K-triviality, respectively) and $i = 0, 1$, we want to enforce

- $R_{\langle 2e,i\rangle} : A \neq \Phi_e^{A_i}$, and
- $R_{\langle 2e+1,i\rangle} : \exists n \; (K(A_i \upharpoonright n) > K(A_{1-i} \upharpoonright n) + e)$.

Thus we make sure that A is not computable in A_0 or A_1 and that A_i does not K-reduce to A_{1-i}. The last in conjunction with Lemma 3.1 immediately gives us $A_0, A_1 <_K A$ as well.

Both types of requirements have their own type of *length of agreement* and *restraint*. At each stage s and for each e the length of agreement is given by

- $l(2e,i)[s] = \mu x(A[s] \upharpoonright x \neq \Phi_e^{A_i}[s] \upharpoonright x)$;
- $l(2e+1,i)[s] = \mu n(K(A_{1-i} \upharpoonright n)[s] > K(A_i \upharpoonright n)[s] + e)$,

on which we put the restraint

- $r(2e,i)[s] = \mu x(\Phi_e^{A_i \upharpoonright x}[s] \upharpoonright l(2e,i)[s] \downarrow)$;
- $r(2e+1,i)[s] = \max_{t \leq s} l(2e+1,i)[t]$.

Note that the second restraint is defined in such a way that it cannot decrease. The verification will show that reaching an infinite length of agreement monitored by the restraints forces us to acknowledge A as K-trivial, contrary to assumption.

Construction. Starting with $A_0[0] = A_1[0] = \emptyset$, if $x \in A[s+1] - A[s]$ is not below any restraint $r(e,i)[s]$, put x into $A_0[s+1]$. Otherwise, take the least $\langle e,i\rangle$ such that $x \leq r(e,i)[s]$, the restraint with the highest *priority*. We want to protect A_i and put x in A_{1-i}. As usual, if x is below restraints $r(e',1-i)[s]$ for higher $\langle e',1-i\rangle > \langle e,i\rangle$ we say that the requirement $R_{\langle e',1-i\rangle}$ is *injured* by this x.

Verification. We again seek to show by induction that each requirement is satisfied and at the same time that a requirement is injured only finitely often and that the restraints will settle. So suppose that for all $\langle e',i'\rangle < \langle e,i\rangle$, $R_{\langle e',i'\rangle}$ is injured only finitely often, $R_{\langle e',i'\rangle}$ is satisfied and $r(e',i') = \lim_s r(e',i')[s]$ exists and is finite.

We start by proving that $R_{\langle e,i\rangle}$ is injured finitely many times, in the exact same way as before. When all higher priority restraints have settled, choose an r above all of them and wait for $A \upharpoonright r$ to settle; then $R_{\langle e,i\rangle}$ cannot be injured because that could only be by a new element below r.

Now for the main part, the verification of requirement $R_{\langle e,i\rangle}$. Take any number n and suppose instead that the requirement is violated: $A = \Phi_e^{A_i}$ if e is even or $\forall n\ (K(A_{1-i} \upharpoonright n) \leq K(A_i \upharpoonright n)+e)$ if e is odd. In both cases, the length of agreement $l(e,i)$ goes to infinity, so there must be some stage s from where $R_{\langle e,i\rangle}$ is always respected such that also $l(e,i)[s] > n$.

If e is even, the argument is just as before. If some new x below the current length of agreement is enumerated in A, the disagreement $A[s](x) \neq \Phi_e^{A_i}(x)[s]$ is preserved and we would have $A \neq \Phi_e^{A_i}$. So that cannot happen, and the length of agreement will not decrease; thus to see if $n \in A$ we only have to check if $n \in \Phi_e^{A_i}[s]$. As we took A to be incomputable (otherwise it would be K-trivial), we conclude that $A \neq \Phi_e^{A_i}$.

In the second case, if $l(e,i)[s] > n$ then the restraint $r(e,i)[s]$ must be above n as well. The restraint cannot decrease and will not be injured, so nothing up to n can be put into A_i, which means $A_i[s] \upharpoonright n = A_i \upharpoonright n$. We can do the same for any n we like, so this proves A_i computable and hence K-trivial. But then by our assumption A_{1-i}, having an initial segment complexity never more than e bits above that of A_i, must be K-trivial as well. The descriptions for the initial segments of A_{1-i} can serve as descriptions for the initial segments of the original A (all the additional information it needs is the computable same initial segment of A_i), so A itself is K-trivial. This is a contradiction, which leads us to conclude that in fact the requirement must be satisfied.

The induction is completed with showing that the restraint $r(e,i)$ will settle. By our previous result there is a least n such that, depending on whether e is even or odd, $A(n) \neq \Phi_e^{A_i}(n)$ or $K(A_{1-i} \upharpoonright n) > K(A_i \upharpoonright n) + e$. The argument in the first case is identical to the one in the proof of Theorem 1.1. In the second case, beyond the stage s where A_0 and A_1 up to $n-1$ have settled and the shortest description of both initial segments is found, the length of agreement must stay at n too. Thus the restraint must settle as well. □

3.2. Splitting in the c.e. C-degrees. The construction of Theorem 3.2 above does not use any special properties of prefix-free Kolmogorov complexity that the plain complexity lacks, so a virtually identical proof shows the same result for C-reducibility.

Proposition 3.3. *For c.e. $A >_C \emptyset$, there exist c.e. A_0 and A_1 such that $A_0 \sqcup A_1 = A$, and*

- $A_0, A_1 <_T A$;
- $A_0 |_C A_1$;
- $A_0, A_1 <_C A$.

Proof. The same requirements $R_{\langle 2e+1,i\rangle}$ with 'C' substituted for 'K' make sure that $A_i \not\leq_C A_{1-i}$, and with Lemma 3.1 also $A_0, A_1 \leq_C A$. Plugging in the Turing-splitting construction gives $A_0, A_1 <_T A$.

The definition of the lengths of agreement and restraints for the C-splitting is also no different, apart from replacing 'C' for 'K'. If $R_{\langle 2e+1,i\rangle}$ is evaded then $\forall n\ (C(A_{1-i} \upharpoonright n) \leq C(A_i \upharpoonright n) + e)$, and the fact that we can determine any initial segment by waiting for the length of agreement to grow sufficiently long shows A_i to be C-trivial, hence also A_{1-i}. Therefore $A \leq_C \emptyset$, contrary to assumption. The rest of the inductive argument needs no adaptations. □

3.3. Density in the c.e. K- and C-degrees. The splitting parts in the construction of Theorem 3.2 are certainly not K-trivial, or they would not be K-incomparable. This proves that the c.e. K-degrees are downward dense. The same applies for the c.e. C-degrees.

Corollary 3.4 (From Theorem 3.2 and Proposition 3.3). *The c.e. K-degrees and the c.e. C-degrees are downward dense.*

Similar to the situation with the LR-degrees, it is as yet unknown whether for any given c.e. set and a second c.e. set strictly K-above (C-above) it, there always exists a third c.e. set strictly K-inbetween (C-inbetween) the two.

Question 3.5. Are the c.e. K-degrees dense? The c.e. C-degrees?

The K- and C-degrees are less connected to the Turing-degrees than the LR-degree were. In particular, regarding the question of upward density, we cannot as easily show that that the structure has a maximal degree. Indeed, it is unknown if they exist.

Question 3.6. Is there a maximal c.e. K-degree? A maximal c.e. C-degree?

We do know that all c.e. sets have very low initial segment complexity, which might prove to be relevant for this question. Intuitively, every initial segment of a c.e. set can be fully described by just two numbers: the length n and the number m of ones in the segment. Then we only have to enumerate the set until we have m elements below n. In the case of the plain descriptive complexity, we can encode both numbers in strings of length $\log n$ and concatenate them in one description – they can be recovered separately because they have the same length. For the prefix-free case, an upper bound follows from the subadditivity property[4] and the bound on the complexity of numbers given by Fact 3.7 of Chapter 1.

Fact 3.7 (Barzdins' Lemma [Bar68]). *For any c.e. set A, there exists a constant b such that*

$$\forall n \in \mathbb{N}\ (C(A \upharpoonright n) \leq 2\log n + b),$$

and a d such that

$$\forall n \in \mathbb{N}\ (K(A \upharpoonright n) \leq 4\log n + d).$$

We can even claim that all c.e. sets are very close to K-trivial. The following definition makes this precise.

Definition 3.8. Set A is *infinitely often K-trivial* if there is a constant b with

$$\exists^{\infty} n \in \mathbb{N}\ (K(A \upharpoonright n) \leq K(n) + b).$$

Any c.e. set holds this property, because it has such short descriptions for the initial segments given by its *true enumerations*. We say that an emumeration of n in c.e. A at stage s is true if it finishes the initial segment, in the sense that no elements below n will be enumerated after s and hence $A[s] \upharpoonright n = A \upharpoonright n$. Every c.e. set must have infinitely many of such enumerations. Then we can define a prefix-free M that for every output 0^n from the universal machine waits for a stage s such that n is enumerated in A and then outputs $A[s] \upharpoonright n$. This way, the descriptions of

[4]For all σ and τ it holds that $K(\langle\sigma,\tau\rangle) \leq K(\sigma) + K(\tau) + b$ for some b. Here $\langle\sigma,\tau\rangle$ is a standard way of encoding two strings into one. The subadditivity property is another feature the plain C-complexity lacks.

the infinitely many true enumerations will be descriptions for true initial segments of A – so infinitely many initial segments have a description not longer, up to an index constant, than that of their length.

Fact 3.9 ([BV10, Proposition 2.2]). *Every c.e. set is infinitely often K-trivial.*

We will meet the infinitely often K-trivials again in the next chapter.

CHAPTER 3

The number of K-trivial sets

The vast majority of sets is captured by our formal characterisation of randomness. That leaves little room for the highly nonrandom sets, as collected in the class of K-trivial sets. Putting all K-trivial sets in a cumulative hierarchy of classes $\mathcal{K}_b$ of sets that are K-trivial via constant b, we have a strict finite bound on the cardinality of each of the levels of this hierarchy.

So one could think of a function that for any given constant returns the finite cardinality of the corresponding level, that is, the number of K-trivials via this constant. It has been an open problem where exactly in the arithmetical hierarchy such a function would reside. Finding a solution to this problem is the topic of the current chapter.

We depart from a representation of K-trivial sets as paths of certain trees, reducing our problem to the computation of the number of paths of members of families of trees. A substantial part of this chapter is thus devoted to an investigation of the complexity of calculating the number of paths of such trees. At the same time, we look for ways of reducing the complexity of the representing trees themselves. The combination of these two approaches will lead us to an answer to our question.

Finally, we also raise and solve the related problem about the number of sets that are low for K.

The main lines of the present chapter are summarised in [BS10].

1. The problem

We state the main problem and its context, and sketch the approach we take to solve it.

1.1. The number of K-trivial sets. The K-trivial sets are quite rare. For any particular constant, there is only a limited number of sets that can be compressed within this constant. After all, there are only so many strings of any length that can receive from the universal machine a description that is not more than a constant b bits longer. That is to say, and here we apply the notion of K-triviality to individual strings as well, there are only so many strings of any length that are K-trivial via b.

Definition 1.1. A string $\sigma \in 2^{<\omega}$ is *K-trivial* via b if $K(\sigma) \leq K(|\sigma|) + b$.

Indeed, we can find a specific constant c such that there are less than 2^{b+c} strings of any particular length K-trivial via given b.

Fact 1.2 ([Cha76]). *There is a constant c such that for all constants b, and all lengths n,*

$$\#\{\sigma \mid K(\sigma) \leq K(n) + b \ \& \ |\sigma| = n\} < 2^{b+c}.$$

As a convenient way of representing the members of $\mathcal{K}$, the K-trivial *sets*, we will make heavy use of the following trees of K-trivial strings.

Definition 1.3. For constant $b \in \mathbb{N}$, we call

$$T_b^{\mathcal{K}} = \{\sigma \mid \forall n < |\sigma| \ (K(\sigma \upharpoonright n) \leq K(n) + b)\}$$

the *K-triviality tree* via b.

Recall the definition of a *tree* as a set of strings that is closed under initial segments. In the following, we call an infinite binary sequence $X \in 2^\omega$ a *path* of tree T if all its initial segments $X \upharpoonright n$ are in T. The set of paths of T is denoted $[T]$.

Let $\mathcal{K}_b$ be a shorthand for the class of K-trivial sets via b. We can view these classes as forming a cumulative hierarchy of the all the members of $\mathcal{K}$, with the sets of $\mathcal{K}_b$ at the b-th level. Now it is not hard to see that the paths of K-triviality tree $T_b^{\mathcal{K}}$ are precisely the sets X such that $K(X \upharpoonright n) \leq K(n) + b$ for all $n \in \mathbb{N}$, that is, the sets that are K-trivial via b. In short, $[T_b^{\mathcal{K}}] = \mathcal{K}_b$. Furthermore, Fact 1.2 also serves as an upper bound on the number of strings on level n of tree $T_b^{\mathcal{K}}$ (the number of strings of length n in $T_b^{\mathcal{K}}$), since every string on the n-th level of $T_b^{\mathcal{K}}$ is a string of length n that is K-trivial via b. In particular, we have a bound on the number of paths of the tree – that is, on the number of sets on the b-th level of the cumulative hierarchy, the number of K-trivials via b.

Fact 1.4 ([Zam90]). *There is a constant c such that for all constants b, the number of sets that are K-trivial via b is less than 2^{b+c}.*

So the cardinality of $\mathcal{K}_b$ is below 2^{b+c}. The exact number naturally depends on the particular universal machine that we choose. A more interesting question is how we can uniformly compute this number. That is, given a constant b, what do we need to find the number of sets that are K-trivial via b? Note that this may still depend on the universal machine. The aim of this chapter is to answer the question:

> What is the complexity of uniformly calculating the number of sets that are K-trivial via given constant b? Does it depend on the universal machine?

Using the trees $T_b^{\mathcal{K}}$, we can make this a bit more precise. We first introduce the concept of a family of trees.

1.2. Families of trees. The $T_b^{\mathcal{K}}$ have arithmetical complexity Δ_2^0, as this is the complexity of the membership question. After all, σ is in $T_b^{\mathcal{K}}$ precisely if we have $K(\sigma \upharpoonright n) \leq K(n) + b$ for its finitely many initial segments $\sigma \upharpoonright n$, and the K-complexity of a string is computable in $\emptyset'$. Hence, with Post's Theorem, determining membership is in Δ_2^0.

We call the sequence $(T_b^{\mathcal{K}})_{b \in \mathbb{N}}$ of all K-triviality trees a Δ_2^0 *family of trees*, meaning that it is a uniformly Δ_2^0 sequence of sets. Each member $T_b^{\mathcal{K}}$ is Δ_2^0 in its constant b. In the same way, a computable family $(T_e)_{e \in \mathbb{N}}$ has members $T_e = \{n \mid P(e,n)\}$ with P a computable property. And a c.e. family of trees is a uniformly c.e. sequence of trees.

Definition 1.5. Class $(T_e)_{e \in \mathbb{N}}$ is a Δ_m^0 *family of trees* if each $T_e = \{n \mid P(e,n)\}$ for a Δ_m^0 property P. Likewise for Σ_m^0 and Π_m^0 families of trees.

If all trees in a family only have a finite number of paths, we can imagine a function that on an index of a tree in this family returns its number of paths.

Definition 1.6. For family $\mathcal{F} = (T_i)_{i \in \mathbb{N}}$ of trees with finitely many paths, define function $G_{\mathcal{F}}$ by

$$G_{\mathcal{F}}(i) = \#[T_i].$$

The G-function of the family $(T_b^{\mathcal{K}})_{b \in \mathbb{N}}$ of all K-triviality trees we denote by $G_{\mathcal{K}}$. So $G_{\mathcal{K}}(b)$ returns the number of trees that are K-trivial via constant b – $G_{\mathcal{K}}(b) = \#[T_b^{\mathcal{K}}] = \#\mathcal{K}_b$.

1.3. The question and its solution. We thus ask for the arithmetical complexity of this function $G_{\mathcal{K}}$. This is a question that was originally put forward by Downey, Miller and Yu. They showed that it is certainly *not* in Δ_2^0, indicating that it is a considerably complex function. Nevertheless, it is not hard to establish that it is in Δ_4^0. We will demonstrate these bounds along the way.

Question 1.7 ([DH10, Section 10.1.4], [Nie09, Problem 5.2.16])**.** What is the complexity of the function $G_{\mathcal{K}}$? In particular, is it in Δ_3^0?

We will approach the problem from two main angles. First, we investigate what we can say about the complexity of calculating the number of paths for general families of trees. Second, we try to find ways to reduce the complexity of families of trees, with the aim of finding a less complex representation of our K-triviality trees.

In the following two sections, we discuss some basic properties of trees with finitely many paths and what is needed to establish the exact number of paths. The fourth section generalises the results of the preceding section to families of trees. Subsequently, we relate the complexity of calculating the number of paths to the jump hierarchy, in particular to high$_2$ and low$_2$ degrees.

In the sixth and seventh section, we make serious work of reducing the complexity of families of trees, resulting in a transformation of our family of K-triviality trees in a c.e. family of trees that are themselves K-trivial. Following this, we see how we can find the low$_2$-ness indices of these trees, to connect with the earlier results related to the jump hierarchy. In the concluding section, the main results are brought together in the solution of our problem.

2. The paths of trees

In this section, we make a number of general observations about the complexity of single trees, to set the stage for the results we develop in the rest of the chapter.

2.1. The complexity of paths. We begin by investigating the complexity of the paths of the most general kind of trees we are interested in, those with only finitely many paths. The main property of paths in such trees is that they are all *isolated*.

Definition 2.1. Path X of tree T is *isolated* if there exists an $n \in \mathbb{N}$ such that the only path extending initial segment $X \upharpoonright n$ is X itself.

Clearly, if we climb from the root any path in a tree with exactly n paths, we will see at most $n - 1$ times a branch diverting from our path that can grow into a path itself. So from some point on we have only one infinite extension ahead of us, which means our path is isolated.

Fact 2.2. *All paths in a tree with finitely many paths are isolated.*

This property gives us a way to compute all paths directly from the tree itself.

Proposition 2.3. *All paths in a computable tree with finitely many paths are computable.*

PROOF. Suppose we want to compute path X of computable tree T with finitely many paths. We know X to be isolated, so let n be such that the only infinite extension of $X \restriction n$ is X itself. Then we can compute X solely from the finite information $X \restriction n$.

Naturally, we know $X \restriction m$ with $m \leq n$ from $X \restriction n$. To determine $X \restriction m$ for $m > n$, compute all extensions σ_i of $X \restriction n$ of length m in the computable tree. One of these extensions must be the $X \restriction m$ we are looking for. So we compute all extensions of these extensions, level by level. The true initial segment of X is the only one that is an initial segment of a path, so all other extensions $\sigma_i \neq X \restriction m$ will no longer have extensions at some level in the tree. We will notice when we have computed all extensions at that level, and σ is the only one left that has extensions at that level. Then we can conclude $\sigma = X \restriction m$. □

The general statement is just a relativisation of the computable case.

Corollary 2.4. *All paths in a tree T with finitely many paths are computable in T.*

PROOF. To compute path X of tree T with finitely many paths, take $X \restriction n$ as an initial segment that only has X as infinite extension. If we assume that we have full knowledge of T, we may determine any larger segment $X \restriction m$ just as we did in the proof of Proposition 2.3. □

Via this basic result, the complexity of the trees $T_b^{\mathcal{K}}$ also bounds the complexity of the K-trivial sets.

Corollary 2.5. *All K-trivial sets are Δ_2^0.*

PROOF. Let A be K-trivial via a constant b. Then A is a path of the tree $T_b^{\mathcal{K}}$. As this tree has finitely many paths, it will be able to compute A. And as $T_b^{\mathcal{K}}$ itself is Δ_2^0, so is A. □

It is not hard to see how to use Corollary 2.4 to obtain trees that not only compute all of their finitely many paths, but are in fact of the same Turing-degree as the join of all the paths.

Corollary 2.6. *For any tree with finitely many paths X_i for $i < n$, there exists a tree with exactly the same paths that has the same degree as $\oplus_{i<n} X_i$.*

PROOF. Let given tree T have n paths X_i ($i < n$). From their least upper bound in the Turing degrees, $\oplus_{i<n} X_i$, we can, by definition, compute all X_i, so it is straightforward to build a tree T' with just those paths using only $\oplus_{i<n} X_i$. And Corollary 2.4 says that all X_i are then computable in T', giving that certainly $\oplus_{i<n} X_i \leq_T T'$. Taken together, we have $\oplus_{i<n} X_i \equiv_T T'$ for T' with the same paths as our original T. □

2.2. Reducing the complexity of trees. A strategy that will prove to be fruitful in solving our main problem is transforming trees of a certain arithmetical complexity to trees of a lesser complexity, while leaving the paths of these trees invariant. As a first step in that direction, in this section we look at what we can do with computable, computably enumerable, and Π^0_1 trees.

It is well-known that any Π^0_1 tree can be reduced to a computable tree without affecting its paths.

Proposition 2.7 (Folklore). *For every Π^0_1 tree, there is a computable tree with the same paths.*

PROOF. The complement of a Π^0_1 set of strings is a c.e. set of strings, so for any given Π^0_1 tree T we can enumerate the elements outside it. Call this c.e. complement R.

We define T' as the set of all strings σ such that at stage $s = |\sigma|$, the computable approximation R_s does not (yet) contain σ nor any of its initial segments. So

$$T' = \{\sigma \mid \forall \tau \preccurlyeq \sigma\ (\tau \notin R_{|\sigma|})\}.$$

Obviously, this tree is computable.

Now any σ in T will never be enumerated in the complement R. Neither will any of its initial segments, T being a tree. So they will certainly not be in R_s with $s = |\sigma|$, which means that σ falls within the definition of T'. Thus all strings in T will be in T', and in particular, all paths of T will also be paths of T'.

To see that no more paths are in T', take arbitrary $X \notin [T]$. Then for some sufficiently large n we must have that $X \upharpoonright n \notin T$, so $X \upharpoonright n$ is enumerated in complement R at some stage s. But if $X \upharpoonright n \in R_t$ for all later stages $t \geq s$, the definition of T' excludes any extensions of $X \upharpoonright n$ from being in it. Thus $X \notin [T']$ as well. That shows that computable T' has precisely the same paths as our given Π^0_1 tree T. □

It is not possible to show the same for c.e. trees. We can even give a counterexample with solely computable paths.

Theorem 2.8. *There exists a c.e. tree (with only computable paths) such that no computable tree has the same paths.*

PROOF. We are going to construct our c.e. tree T by diagonalising over an effective list $(\Phi_e)_{e\in\mathbb{N}}$ of all partial computable functions of strings to $\{0, 1\}$. Of course, we are only really interested in the members that are total functions Φ such that $\tau \prec \sigma$ and $\Phi(\sigma)$ imply $\Phi(\tau)$, that is, the characteristic functions of computable trees. An effective list of only these functions, however, cannot exist (by standard diagonalisation); the important thing is that $(\Phi_e)_{e\in\mathbb{N}}$ contains them all.

We build T in stages. First of all, we will develop the leftmost branch of strings of 0's, adding the string 0^s at each stage s. Then for each Φ_e we will work from the node 0^e1, trying to make different paths in that cone[1]. This way the strategies for the different functions obviously cannot interfere with each other.

CONSTRUCTION. We describe the strategy for function Φ_e. First wait until stage $s_0 = e + 1$ is reached. Then at each later stage $s > s_0$, compute $\Phi_e(\sigma)[s]$ for all extensions $\sigma \succ 0^e1$ of length up to s. Repeat this procedure until at some stage

[1]The *cone* of a string is the set of all its finite extensions.

t, function Φ_e returns 0 in at most t execution steps for all extensions $\sigma \succ 0^e1$ of the same length $l \leq t$. At that point, we add the string 0^e10^t and all its prefixes to T. In all following stages u, we continue developing this branch by adding strings 0^e10^u.

Verification. The construction performs a lot of work for functions Φ_e that do not correspond to trees at all, but we do not care about that. If Φ_e indeed gives a tree T_e, we can distinguish two cases. Either there is a path in the cone above 0^e1 in T_e, or there no such path. In the first case, the construction keeps looking in vain for a level l in T_e where there is no extension of 0^e1, so $\Phi(e)\downarrow= 0$ for all $\sigma \succ 0^e1$ with $|\sigma| = l$. Then nothing above σ will ever be added, and T_e has a path that T does not. In the second case, a level without extension of 0^e1 is found, and an infinite path 0^e10^ω is developed. Of course, this path is not in T_e. Thus $[T] \neq [T_e]$.

Performing the same strategy for every e then makes sure that no computable tree has the same infinite paths. All paths in T will be of the form 0^ω (the left-most path) or 01^e0^ω for some $e \in \mathbb{N}$, and these, having finite information, are all computable. □

Combining this result with Proposition 2.7 gives the somewhat stronger:

Corollary 2.9. *There exists a c.e. tree (with only computable paths) such that no Π^0_1 tree has the same paths.*

Proof. From the fact that for every Π^0_1 tree there is a computable tree with the same paths, we can conclude that the c.e. tree from Theorem 2.8 has different paths from any Π^0_1 tree as well. □

Notice that the previous construction yielded a tree with infinitely many paths (there will be infinitely many partial computable Φ_e that give a tree that contain no paths in the cone of 0^e1). It must be so, because for any finite number of computable sets we can easily construct a computable tree with just those sets as paths.

Alternatively, we can construct such a c.e. tree with noncomputable paths. In that case the number of paths can be bounded. Indeed, a tree with just one noncomputable path will not be computable (Proposition 2.3). Hence any noncomputable path that can be the unique path of a c.e. tree will do.

Proposition 2.10. *For every Δ^0_2 set we can construct a c.e. tree with just this path.*

Proof. Given $Z \in \Delta^0_2$, we have an approximation $Z = \lim_s Z_s$ for a computable sequence $(Z_s)_{s\in\mathbb{N}}$. Now we can construct c.e. T by enumerating $Z_s \upharpoonright s$ and its initial segments at every stage s. Then the only path that will emerge is $Z = \lim_s Z_s$. □

So we can just pick any noncomputable member of Δ^0_2 to be the single path in our c.e. tree.

Corollary 2.11. *There exists a c.e. tree with only one path such that no computable tree has precisely the same paths.*

PROOF. Take any noncomputable Δ_2^0 set. By the result above there exists a c.e. tree T with this set as only path. Furthermore, no computable tree can have this set as its only path, because that would make the set computable. So no computable tree has the same paths as T. □

2.3. C.e. trees of bounded width. We can further demand that our trees not only have a finite number of paths, but also contain no more than a certain finite number of strings at any level. Such trees are said to have *bounded width.*

Definition 2.12. The *width* of a tree is the lowest upper bound on the cardinality of its levels. A tree has *bounded width* if its width is finite.

This restriction has its impact on the complexity of the trees. For example, for these kind of c.e. trees we can at all times find computable trees with the same paths.

Proposition 2.13. *For any c.e. tree of bounded width, there exists a computable tree (of bounded width) with precisely the same paths.*

PROOF. For any c.e. tree T of bounded width, we know there is a maximal n such that at infinitely many levels there are n strings, and thus a level k above which no level contains more than n strings.

From this n and k, we can construct a computable subtree T' as follows. Enumerate elements of T until we have found n strings of the same length above k. As there are infinitely many levels of T with n elements, this must happen eventually. Then we add these strings and all their initial segments to T'. We go on looking for n strings of the same length at ever higher levels, repeatedly adding these and their initial segments to T'.

Now subtree T' has the same paths as T. For any path X in $[T]$, clearly each level in T contains an initial segment of X. So as soon as we have found n strings at some level, one of these strings must be the initial segment of the path. Then any time n strings of any particular level and its initial segments are added to T', certainly all initial segments of X up to n are added as well. As this is done for ever higher levels, all of X will be in T'.

The constructed tree T' is computable because to determine whether σ is in T', we just have to wait until the construction adds n strings of a higher level to T'. As noted before, this is guaranteed to happen. Since these n strings are all strings of that level in T, hence in T', we can conclude that σ is in T' if and only if it is an initial segment of one of these strings, that is, contained in T' after that stage. □

The paths of a c.e. tree of bounded width themselves must be very simple as well, which follows directly from applying Proposition 2.3 to the previous result.

Corollary 2.14. *All paths in a c.e. tree of bounded width are computable.*

PROOF. We saw before that all paths in a computable tree with finitely many paths are computable. Since for any c.e. tree of bounded width (hence, naturally, with a finite number of paths) there exists a computable tree of bounded width with exactly the same paths, the fact that the paths of the latter are all computable implies that the paths of the former are computable as well. □

Nevertheless, c.e. trees of bounded width are not so simple as to be necessarily computable themselves, as the following straightforward construction shows.

Proposition 2.15. *There exists a noncomputable c.e. tree of bounded width.*

PROOF. Let $(\Phi_e)_{e\in\mathbb{N}}$ be an enumeration of all partial computable functions from finite strings to $\{0,1\}$, as in Proposition 2.8. We will build our c.e. T in such a way that it disagrees on string 0^e1 for all Φ_e that correspond to a tree.

At each stage s, we compute $\Phi_e(0^e1)[s]$, if it was still divergent at stage $s-1$, for all $e < s$. As soon as a $\Phi_e(0^e1)[s]$ converges, we will put 0^e1 into T if $\Phi_e(0^e1)$ returns 0, and do nothing if it returns 1.

This way, for any Φ_e that is the characteristic function of a tree T_e, we have that $0^e1 \in T$ if $\Phi_e(0^e1) = 0$ and $0^e1 \notin T$ if $\Phi_e(0^e1) = 1$. So $0^e1 \in T$ if and only if $0^e1 \notin T_e$.

Additionally, the width of the resulting tree T is never more than 2. At each level s only 0^s and possibly $0^{s-1}1$ are contained in T. □

3. The number of paths

The next step is looking at the complexity of calculating the exact number of paths in a tree with finitely many of them.

3.1. Computable trees. First we note that the question whether a tree has paths at all is very easily expressible. Tree T does if it has strings of any length:

$$\forall n \exists \sigma\ (|\sigma| = n \ \&\ \sigma \in T).$$

So for computable trees this is a Π^0_1-question, as the existential quantifier is bounded (there are only finitely many strings of any particular length, so only finitely many strings to be searched through). That makes it decidable with the help of oracle $\emptyset'$.

Fact 3.1. *The question whether a computable tree has paths is uniformly decidable by $\emptyset'$.*

The exact number of paths is slightly more difficult to determine.

Proposition 3.2. *The number of paths of a computable tree with finitely many paths is uniformly computable in $\emptyset''$.*

PROOF. The following algorithm will compute the number of paths of a given computable tree T with finitely many paths.

We first ask if T has any paths at all, as above. If not, the number is 0. Otherwise, starting at 1, we ascend through the natural numbers, testing for each n whether T has at least n paths. As soon as we get a negative answer for some $n+1$, we will know that the tree has n paths. Since T has only finitely many paths, this search must terminate with such an answer at some point.

The tree has at least n paths if there exists a group of n strings $\sigma_0, \ldots, \sigma_{n-1}$ of the same length, such that at every higher level in the tree there are extensions τ_i for each of these σ_i:

$$\exists \sigma_0, \ldots, \sigma_{n-1} \in T\ (\forall i, j < n\ (|\sigma_i| = |\sigma_j|)\ \&\ \forall l > |\sigma_0|\ P(l, \sigma_0, \ldots, \sigma_{n-1}))),$$

with

$$P(l, \sigma_0, \ldots, \sigma_{n-1}) \equiv \exists \tau_0, \ldots, \tau_{n-1} \in T\ (\forall i < n\ (|\tau_i| = l\ \&\ \sigma_i \prec \tau_i)$$

expressing that all σ_0 to σ_{n-1} have an extension at level l.

This is a Σ^0_2-question, hence in Δ^0_3, hence decidable in $\emptyset''$. Thus the whole algorithm, calculating the number of paths of T, can be executed in $\emptyset''$. □

This procedure can be easily relativised.

Proposition 3.3. *The number of paths of an A-computable tree with finitely many paths is uniformly computable in A''.*

Proof. To determine the finite number of paths of A-computable tree T, we can just trace the algorithm of Proposition 3.2 above. Asking if T has paths at all is in Π^A_1, and the query about the group of strings that has extensions at each level is now in Σ^A_2. Thus we can perform it using A''. $\square$

We can improve on this upper bound within a restricted class of computable trees, those with no infinite anti-chains.

Definition 3.4. An *anti-chain* on tree T is a set of pairwise incomparable strings in T. A *split* on T is a pair of strings $\sigma 0$, $\sigma 1$ (for any $\sigma \in 2^{<\omega}$) in T.

Having a bound on the length of the anti-chains is equivalent to having only a finite number of splits. For if tree T has an infinite number of splits, we can build an infinite anti-chain by repeatedly looking for a split (that grows in two different subtrees, where at least one must have infinitely many splits as well), and continuing this procedure in the subtree with infinitely many splits after adding a string in the other subtree to the anti-chain. Conversely, if T contains an infinite anti-chain, we can reason that every two elements of this anti-chain imply a split below them in the tree (as they are incomparable, they cannot be on the same branch), giving an infinite number of splits in total.

Fact 3.5. *A tree has finitely many splits if and only if it does not contain infinite anti-chains.*

For a computable tree with no infinite anti-chains, already the halting set suffices to calculate the number of paths.

Theorem 3.6. *The number of paths of a computable tree without infinite anti-chains is uniformly computable in $\emptyset'$.*

Proof. A tree T without infinite anti-chains has a bounded number of splits, so there is a level above all splits. We look for this level k by asking whether

$$\forall n \geq k\ ((|\sigma| = n\ \&\ \sigma \in T) \Rightarrow \neg(\sigma 0 \in T\ \&\ \sigma 1 \in T)),$$

a Π^0_1-question solvable by $\emptyset'$.

Then the number of infinitely extendible strings of length k is exactly the number of infinite paths of T. So these are the strings σ of length k such that

$$\forall n > k\ \exists \tau (|\tau| = n\ \&\ \sigma \prec \tau),$$

which we can again test using $\emptyset'$. $\square$

3.2. Computably enumerable trees. Next we look at computably enumerable trees. Determining whether a c.e. tree has any paths at all still amounts to checking if it has a string of any length, but because we now need the halting set to settle membership of the tree, this question is in Π^0_2.

Fact 3.7. *The question whether a c.e. tree has paths is uniformly decidable by $\emptyset''$.*

The same goes for determining the exact number of paths of a tree with finitely many paths.

Proposition 3.8. *The number of paths of a c.e. tree with finitely many paths is uniformly computable in $\emptyset'''$.*

Proof. Any c.e. tree is of a degree below $\mathbf{0}'$. With Proposition 3.3 we have that the degree of the problem of determining its number of paths is below $\mathbf{0}'''$, so computable in $\emptyset'''$. □

We can improve on this upper bound if we confine ourselves again to the more restricted class of c.e. trees that have bounded width.

Theorem 3.9. *The number of paths of a c.e. tree of bounded width is uniformly computable in $\emptyset''$.*

Proof. Given the program that enumerates c.e. tree T with finitely many paths and bounded width, we first want to find the maximum number n such that infinitely many levels in T contain n many strings. As our tree is of bounded width, such a maximum must exist.

So for this number n there are infinitely many levels k with n distinct strings, that is, for all levels l there is a level $k > l$ such that there are distinct $\sigma_0, \dots, \sigma_{n-1}$ of length k in T, and indeed any string of length k in T is one of these σ_i. As membership in c.e. T is Σ^0_1, this constitutes a Π^0_2-question. Furthermore, n is the largest number for which this holds: there is a level l such that at all later levels $k > l$ there are at most n distinct strings in T; more precisely, for all stages s there are less than $n+1$ distinct strings of length k in $T[s]$. This is clearly again a Π^0_2-question, so the oracle $\emptyset''$ will provide us with this n.

Let k be a level above which there will be no more than n distinct strings at the same level. Now we can define a c.e. sequence $\{(\sigma^l_0, \dots, \sigma^l_{n-1})\}_{l\in\mathbb{N}}$ of n-tuples of strings of the same length on T – simply put every enumerated string of length $k+i$ in the same temporary tuple, and only insert the tuple in the sequence if it is filled with n elements. Note that we can insert the tuples in the sequence in such a way that they are ordered by the length of their elements. The important property of this sequence is that any infinite path through our tree T clearly has to intersect all of the tuples.

We start by checking if the number of infinite paths might be n. This is the case if there is some level l such that for all greater h all strings in $(\sigma^l_0, \dots, \sigma^l_{n-1})$ have an extension in $(\sigma^h_0, \dots, \sigma^h_{n-1})$. To be even more exact, if there is some stage s where a tuple $(\sigma^l_0, \dots, \sigma^l_{n-1})$ of level l has been enumerated, such that for all stages t any enumerated $(\sigma^h_0, \dots, \sigma^h_{n-1})$ of higher level h contains extensions of all strings in the former tuple. If $\emptyset''$ answers this Σ^0_2-question in the positive, the number of paths equals n and we are done.

Otherwise, for each level there must exist a greater level such that not all strings in the n-tuple of the former level have an extension in the n-tuple of the latter level. By taking out all n-tuples $(\sigma^{l+1}_0, \dots, \sigma^{l+1}_{n-1})$ that do have an extension for all strings in the directly preceding n-tuple $(\sigma^l_0, \dots, \sigma^l_{n-1})$, we obtain a subsequence $\{(\sigma^h_0, \dots, \sigma^h_{n-1})\}_{h\in\mathbb{N}}$ of the original sequence where for each h there is a σ in $(\sigma^h_0, \dots, \sigma^h_{n-1})$ without an extension in $(\sigma^{h+1}_0, \dots, \sigma^{h+1}_{n-1})$. If we now remove these τ in all n-tuples of the sequence, we get an infinite sequence of $(n-1)$-tuples. As none of the τ we removed could be extended, each infinite path trough T still has to intersect all of the $(n-1)$-tuples in our new sequence.

Now we can check if the number of infinite paths is $n-1$, as we did above for n. If there is indeed some level l such that for all greater h all strings in $(\sigma_0^l, \ldots, \sigma_{n-2}^l)$ have an extension in $(\sigma_0^h, \ldots, \sigma_{n-2}^h)$, we conclude that the number is $n-1$. If not, we know there cannot be $n-1$ paths. Then we continue as before, defining a subsequence of $(n-2)$-tuples.

This way we define sequences of ever smaller tuples, until we hit upon the right number. On reaching this number, our check must yield a positive result, and we know we have reached the true number of infinite paths. $\square$

Actually, we could have gotten to the same result via another route.

Proposition 3.10. *For any c.e. tree of bounded width, in $\emptyset''$ we can uniformly construct a computable tree (with bounded width) that has the same paths.*

PROOF. In Proposition 2.13, we saw how we could construct a computable tree with the same paths as a given c.e. tree T with bounded width. This was not a uniform construction, because it depends on the n and k that stand for the maximal number such that there are infinitely many levels with that many strings, and a level such that no higher level contains more than that number of strings, respectively.

But these n and k can be found from the index of T by means of two-quantifier questions. To be a little bit more precise, n is the first number such that for all levels there is a greater level that has n strings, and from some level on there are no levels with $n+1$ strings. And k is the first level such that at all greater levels there are no more than n strings. Inspection of the complexity of these searches reveals that they can be done in oracle $\emptyset''$. That means that the whole construction can be performed uniformly in the index of the c.e. tree, using $\emptyset''$. $\square$

So with the help of $\emptyset''$ a computable subtree T' of given c.e. T with bounded width can be constructed, and, by Proposition 3.2, with the same oracle we can calculate the number of paths of this computable T' as well. As T has the same number of paths as T', this is an alternative demonstration that the number of paths of c.e. tree with bounded width is computable in $\emptyset''$.

4. Families of trees

Now that we have seen what we need to determine the existence or indeed the number of paths in a given tree for various classes of computable and c.e. trees, we shift our attention to families of such trees.

Analogous to the G-function, we will associate with each family of trees with finitely many paths a function that determines whether a given tree has paths at all. For such a family of trees $\mathcal{F} = (T_i)_{i\in\mathbb{N}}$, let

$$H_{\mathcal{F}}(i) = \begin{cases} 1 & \text{if } \#[T_i] > 0 \\ 0 & \text{otherwise} \end{cases}.$$

4.1. Calculating the number of paths.

Since the algorithms we gave in the previous section were all uniform, the complexity results about calculating the existence and number of paths of single trees immediately transfer to families of trees.

Corollary 4.1 (of Fact 3.1 and Proposition 3.2). *For computable family $\mathcal{F}$ of trees with finitely many paths, $H_{\mathcal{F}} \leq_T \emptyset'$ and $G_{\mathcal{F}} \leq_T \emptyset''$.*

Proof. All members T_i of the uniformly computable sequence $\mathcal{F}$ are computable from a single algorithm that takes indices i. But then we can adapt this algorithm to ask the $\emptyset'$-decidable question about the computable tree given by i. This is an algorithm for $H_{\mathcal{F}}$, hence this function is computable in $\emptyset'$. In the same way we can execute the uniform procedure of Proposition 3.2 on each given index, giving $G_{\mathcal{F}} \leq_T \emptyset''$. □

The relativisation of the foregoing is straightforward.

Corollary 4.2. *For A-computable family $\mathcal{F}$ of trees with finitely many paths, $H_{\mathcal{F}} \leq_T A'$ and $G_{\mathcal{F}} \leq_T A''$.*

Proof. To determine the finite number of paths of A-computable tree T, we can just trace the algorithm of Proposition 3.2 above. Asking if T has paths at all is in Π^A_1, and the query about the group of strings that has extensions at each level is now in Σ^A_2. Thus we can perform it using A''. □

With the fact that the family $(T^{\mathcal{K}}_b)_{b\in\mathbb{N}}$ of K-triviality trees is Δ^0_2, hence $\emptyset'$-computable, we obtain the first upper bound on the complexity of the $G_{\mathcal{K}}$ function.

Corollary 4.3. *The function $G_{\mathcal{K}}$ is in Δ^0_4.*

Proof. From Corollary 4.2 we get $G_{\mathcal{K}} \leq_T \emptyset'''$. □

Now that we have upper bounds on the complexity of the H- and G-functions, we may wonder if they are strict. In the next two sections, we show for computable and c.e. families of trees that indeed they cannot be improved in general.

4.2. Computable families of trees. We have just seen that $\emptyset'$ can uniformly establish the existence of paths in any member of a computable family of trees. As could be expected, such H-function is not computable in general. We can make this explicit by constructing a computable family of trees for which the halting set $\emptyset'$ is directly coded into the information whether a given tree in the family has any path. The problem of determining the existence of a path for a tree in this family then truly has degree $\mathbf{0}'$.

Theorem 4.4. *There exists a computable family $\mathcal{F}$ of trees (with only finitely many paths) such that the degree of $H_{\mathcal{F}}$ is $\mathbf{0}'$.*

Proof. With Corollary 4.1 we know that the H-function for any computable family of trees must be below $\emptyset'$. So our goal is to construct a computable family $\mathcal{F}$ of trees such that $H_{\mathcal{F}} \geq_T \emptyset'$.

We use the standard enumeration $(\phi_e)_{e\in\mathbb{N}}$ of all partial computable functions. For each index e, we want to construct a tree T_e that has no paths if $\phi_e(e)$ converges (hence $e \in \emptyset'$), and one path if it does not ($e \notin \emptyset'$). The idea is that we only develop one branch, the leftmost one, and keep doing that as long as the computation of $\phi_e(e)$ continues.

If the computation $\phi_e(e)[s]$ does not yet converge, we add the string 0^s to the tree. The result is that the leftmost branch of only 0's is extended with one, to length s. But if $\phi_e(e)[s]$ does converge, we will stop the construction of this tree immediately.

Now if $\phi_e(e)$ is in fact divergent, the computation of $\phi_e(e)$ will never settle. So we will keep on extending our branch, and it will grow into an infinite path. If, on the other hand, $\phi_e(e)$ converges, at some stage the computation will halt. At that point we will stop expanding the branch, and our tree will have no paths. Thus $H_{\mathcal{F}}(e) = 1$ precisely if $\phi_e(e)\downarrow$, and $H_{\mathcal{F}}(e) = 0$ precisely if $\phi_e(e)\uparrow$.

Finally, the tree is computable in its index e. To see if a string 0^t of length t is in it, we just have to construct the tree up to the same height as this length, that is, compute $\phi_e(e)$ up to stage t. Only if $\phi_e(e)[t]$ does not converge, will the string be in the tree. □

Every tree in the family of the above construction only consists of strings of 0's. That means that none of these trees has any splits. So even though this family $\mathcal{F}$ of trees is complex in the sense that $H_{\mathcal{F}}$ is above $\emptyset'$, it is very simple in the sense that its members do not have any nontrivial anti-chains. Indeed, by Theorem 3.6 the number of paths of every tree in this family is already computable in $\emptyset'$, resulting in a function $G_{\mathcal{F}}$ below $\emptyset'$.

So if we want to take the next step by coding $\emptyset''$ in the number of paths of a computable family of trees (by Corollary 4.1, we also had that $\emptyset''$ is an upper bound on its G-function), we already know that the trees in this family must have infinitely many splits. The next Theorem 4.5 yields such a family.

For the construction in that theorem, we need to go into a small technicality. Suppose we want to approximate a computation $\Phi^A(n)$ on some $n \in \mathbb{N}$ that uses a c.e. oracle A. In general, we would like this approximation to have the property that if $\Phi^A(n)$ diverges, also $\Phi^A(n)[s]$ diverges for infinitely many stages s. As the approximation may be unstable, forever jumping between values, this is not necessarily the case. So we have to enforce it, which we do by modifying the approximation in a minor way. This is called the *hat trick*.

The trick comprises making $\Phi^A(n)[s]$ divergent if an x below the current use $\phi^A(n)[s]$ is enumerated in A at this stage s. As a change in value of the approximation must be caused by a change in the oracle, any such change will now give a divergent state. So if the original approximation changed value infinitely often (thus diverged), the new one will diverge infinitely often.

Theorem 4.5. *There exists a computable family $\mathcal{F}$ of trees such that the degree of $G_{\mathcal{F}}$ is $\mathbf{0}''$.*

PROOF. We are going to construct a sequence of computable trees such that the tree with index e has two paths if $\Phi_e^{\emptyset'}(e)$ halts, and one path if it does not. Then the function $G_{\mathcal{F}}$ returning the number of paths from a tree index e directly computes $\emptyset''$. With Proposition 3.2, we thus have $G_{\mathcal{F}} \equiv_T \emptyset''$.

Using the fact that c.e. $\emptyset'$ has a computable approximation, we can compute the approximation $\Phi_e^{\emptyset'}(e)[s]$ of the e-th oracle function with oracle $\emptyset'$ at every stage s. We assume we have applied the hat trick to our approximation, so if $\Phi_e^{\emptyset'}$ diverges, its approximations will do so at infinitely many stages. With the help of the outcomes of these adapted $\Phi_e^{\emptyset'}(e)[s]$, we construct the tree with index e in stages.

The backbone of the tree is the leftmost branch of only 0's, which we build by adding the string 0^s at each stage s. As long as the approximation $\Phi_e^{\emptyset'}(e)[s]$ at these stages does not diverge, we only develop this branch. But if at some stage t we have that $\Phi_e^{\emptyset'}(e)[t]$ converges, we create a new branch, by adding in addition the string

$0^{t-1}1$. Now we keep developing both branches, adding both 0^u and $0^{t-1}10^{u-t-1}$ at each new stage u, but only if $\Phi_e^{\emptyset'}(e)[u]$ does not diverge again at this u. If it does, we simply stop developing the second branch, starting anew if at a later stage the approximation happens to converge again.

Now if $\Phi_e^{\emptyset'}(e)$ eventually converges, from some stage on we will keep on extending the second branch, and our tree has two paths. If, on the other hand, $\Phi_e^{\emptyset'}(e)$ diverges, our hat trick guarantees that the approximation diverges infinitely often. Hence any second branch will always end up being cut off, and will have no chance of growing into a path. So in this case the tree has precisely one path. In the end, $G_{\mathcal{F}}(e) = 2$ precisely if $e \in \emptyset''$ and $G_{\mathcal{F}}(e) = 1$ precisely if $e \notin \emptyset''$.

Furthermore, the tree is computable in the index. To see if a string of a certain length is in it, we only have to construct the tree up to the same height as this length, which we can do in a computable way. □

The same argument can be located in any degree.

Proposition 4.6. *For any degree* $\mathbf{a}$, *there exists an* $\mathbf{a}$*-computable family* $\mathcal{F}$ *of trees such that the degree of* $G_{\mathcal{F}}$ *is* $\mathbf{a}''$.

Proof. If we allow ourselves to use A in performing a construction as in Theorem 4.5 above, we can code A'' in a family of trees by computing the approximations $\Phi_e^{A'}(e)[s]$ at each stage s. We may of course do this because A' is c.e. in A. Then we construct trees for each index e precisely as in the original procedure, enforcing that the e-th tree has one path if $\Phi_e^{A'}(e)$ diverges (so $e \notin A''$) and two paths if it converges ($e \in A''$). Adding Corollary 4.2, the resulting family has a G-function that is of the same degree as $A'' \in \mathbf{a}''$. □

4.3. Computably enumerable families of trees. Turning to c.e. families of trees with finitely many paths, we first have by Corollary 4.2 that $\emptyset''$ decides if any given member has paths at all. Again we can make this upper bound on the H-function strict.

Theorem 4.7. *There exists a c.e. family* $\mathcal{F}$ *of trees such that the degree of* $H_{\mathcal{F}}$ *is* $\mathbf{0}''$.

Proof. In this case, we want the e-th tree T_e in our family $\mathcal{F}$ to have no paths if $\Phi_e^{\emptyset'}(e)$ converges (hence $e \in \emptyset''$), and one path if $\Phi_e^{\emptyset'}(e)$ diverges (hence $e \notin \emptyset''$). Thus its H-function computes $\emptyset''$, and indeed $H_{\mathcal{F}} \equiv_T \emptyset''$. We again develop only one branch, the leftmost one, and keep doing that as long as approximations to the computation keep changing.

We build the tree with index e in stages, computing $\Phi_e^{\emptyset'}(e)[s]$ at each stage s. If this computation yields the same value as the one in the previous stage, we do nothing. But if the result of this computation is divergent, or different from the result in the previous stage, we add the string 0^u with u the height of the tree so far plus one. The result in the second case is that we extend the leftmost branch by one.

Now if $\Phi_e^{\emptyset'}(e)$ is in fact divergent, the values of $\Phi_e^{\emptyset'}(e)[s]$ as s increases will never settle. So we will keep on extending our branch, and it will grow into an infinite path. If, on the other hand, $\Phi_e^{\emptyset'}(e)$ converges, at some stage it will settle, and $\Phi_e^{\emptyset'}(e)[t]$ will always converge to the same value for every t greater than some s. Then from this stage s on, we will never find the opportunity to expand the branch again, and our tree will have no paths. Thus $H_{\mathcal{F}}(e) = 1$ precisely if $\Phi_e^{\emptyset'}(e)\downarrow$, and $H_{\mathcal{F}}(e) = 0$ precisely if $\Phi_e^{\emptyset'}(e)\uparrow$.

Finally, the tree is computably enumerable in its index e. The previous is a clearly computable procedure for enumerating strings in the tree with index e. □

Similar to the computable case, the previously constructed family only contains trees without splits. Via a straightforward relativisation of Theorem 3.6, the number of paths of any given tree in the family is computable in $\emptyset''$. So again we have this discrepancy between a H-function that is as simple as possible and a G-function that is as complex as can be.

Finally, the upper bound of $\emptyset'''$ on the G-function of c.e. families of trees is tight as well. The coding of this oracle in such a family does need some more work now, because we cannot approximate $\emptyset'''$ as easily as we could $\emptyset''$. The problem is that $\emptyset''$ as an oracle has no simple c.e. approximation, as $\emptyset'$ had.

We do have that $\emptyset''$ is c.e. in $\emptyset'$, so there exists a c.e. operator W such that $W^{\emptyset'} = \emptyset''$. Let us define computable $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ as follows.

$$f(e,s) = \begin{cases} s & \text{if } \Phi_e^{W^{\emptyset'}}(e)[s]\uparrow \\ \langle \sigma, \tau\rangle & \text{if } \Phi_e^{W^{\emptyset'}}(e)[s]\downarrow \end{cases},$$

where σ is the use of $\emptyset'[s]$ in the calculation of the members of $\tau \prec W^{\emptyset'}[s]$, the latter being the use in the convergent computation of $\Phi_e^{W^{\emptyset'}}(e)[s]$. In our construction of the sequence of trees we can now use the following equivalence.

Lemma 4.8. *For computable f defined as above,*

$$e \in \emptyset''' \iff \liminf_s f(e,s) \text{ exists.}$$

Proof. In the following, we assume that the hat trick has been applied already. We first show that for each $k \in \mathbb{N}$, there are infinitely many stages s such that the initial segment $W^{\emptyset'}[s] \upharpoonright k$ equals $\emptyset'' \upharpoonright k$.

Each element n that is in $\emptyset'' \upharpoonright k$, will from some stage s on also always be in the approximations to $W^{\emptyset'} \upharpoonright k$. This is because it will be enumerated by a computation of $W^{\emptyset'}$ with a certain use of $\emptyset'$, and this part of $\emptyset'$ will have settled at some point. It is only about elements m that are not in $\emptyset'' \upharpoonright k$ that the approximations to $W^{\emptyset'}$ might never be conclusive about: one might always find a later stage u such that $W^{\emptyset'}(m)[u] = 1$, because W may compute this with ever greater use.

However, if we wait for the stage where the part of $\emptyset'$ has settled that is used for the computations of every element that is truly in $\emptyset'' \upharpoonright k$, we have that at every later stage v where a true enumeration into $\emptyset'$ takes place, indeed $W^{\emptyset'}[s] \upharpoonright k = \emptyset'' \upharpoonright k$ holds. Say only the initial segment $\emptyset' \upharpoonright m$ is used by any computation of $W^{\emptyset'}$ that enumerates an element in $\emptyset'' \upharpoonright k$, and this segment has settled by stage s. Now consider the true enumeration of $n > m$ in $\emptyset'$ at a later stage v. A computation of $W^{\emptyset'}[v]$ that enumerates an element $p \notin \emptyset''$ below k has a use either up to n or greater than n. But in the first case, the use is a true initial segment of $\emptyset'$, which would mean that in fact $p \in \emptyset''$. And in the second case any such computation would have been made divergent by the hat trick (after all, an n within the use is enumerated into the oracle). Hence no element $p \notin \emptyset''$ below k could be enumerated by $W^{\emptyset'}[v]$. And since every $n \in \emptyset'' \restriction k$ is indeed enumerated at this stage, we have $W^{\emptyset'}[s] \restriction k = \emptyset'' \restriction k$. As there are infinitely many true enumerations, there are infinitely many s such that the initial segment $W^{\emptyset'}[s] \restriction k$ equals $\emptyset'' \restriction k$.

We are now ready to prove the lemma.

($\Longrightarrow$) For the left-to-right direction, if $e \in \emptyset'''$ then $\Phi_e^{\emptyset''}(e) \downarrow$ with a certain use k. By the result above, the part $W^{\emptyset'}[s] \restriction k$ that is used of the oracle will be correct infinitely many times. So at infinitely many stages the same convergent computation will be performed. That means $f(e, s)$ must return the same part σ of the oracle infinitely many times,

($\Longleftarrow$) Conversely, if the limit infimum of f does exist, some pair $\langle \sigma, \tau \rangle$ will keep coming back as outcome of $f(e, s)$ for ever greater s. That means that again and again the initial segment τ of approximations to oracle $W^{\emptyset'}$ is used in approximations to computations of the e-th oracle machine on input e, and that in calculating the elements in this segment, the same initial segment σ of approximations to $\emptyset'$ is used. But every initial segment of c.e. $\emptyset'$ will settle at some point, so this recurring σ must in fact be a correct initial segment. So all computations by $W^{\emptyset'}$ with use σ are valid as well, meaning that the 1's in τ must be correct. □

Theorem 4.9. *There exists a c.e. family $\mathcal{F}$ of trees such that the degree of $G_{\mathcal{F}}$ is $\mathbf{0}'''$.*

Proof. We want to code $\emptyset'''$ by constructing for index e a c.e. tree T_e with two paths if $\Phi_e^{\emptyset''}(e)$ halts, and one path if it does not. We make use of the equivalence between the convergence of $\Phi_e^{W^{\emptyset'}}(e)$ and the existence of $\lim_s \inf f(e, s)$.

We develop the leftmost path of 0's as before, adding 0^s at each stage s. Next we compute the outcome $f(e, s)$. We make a distinction between *active* and *inactive* outcomes, initially setting all possible outcomes to inactive. If $f(e, s) = n$ and n is currently inactive, we make this n active and we add the string $0^{s-1}1$ to the tree, creating a branch on the leftmost path. If n was active already, we continue expanding the branch that we created when it was last turned active. So we add the string $0^{t-1}10^{s-t}$ and all of its initial segments, with t the least stage such that on all stages between t and s the outcome n is active. Finally, we make all greater $m > n$ inactive.

The crux of the construction is that whenever an outcome becomes inactive, the branch built for it so far will never be continued. So the only outcome that will give rise to an infinite path is one that is infinitely often visited and will never be made inactive from some stage on. But that is precisely the lowest outcome that is infinitely often given, i.e., $\lim_s \inf f(e, s)$. So if $\lim_s \inf f(e, s)$ indeed exists, the tree will have two paths, the leftmost one and the branch belonging to outcome $\lim_s \inf f(e, s)$. If it does not, only the leftmost path will be developed into infinity. Thus the tree has two paths if $e \in \emptyset'''$, and one path if $e \notin \emptyset'''$. □

The previous construction must give trees with unbounded width, because the upper bound of $\emptyset''$ established in Theorem 3.9 makes it impossible to code $\emptyset'''$ in the G-function of a sequence of c.e. trees with bounded width. But if we allow our trees to be just a little more complex, we can construct such a sequence of trees of bounded width. Namely, there exists a computable sequence of *difference of c.e.* trees of bounded width such that the function computing the number of paths of a given tree, computes $\emptyset'''$. In the enumeration of difference of c.e. (d.c.e.) sets, also called 2-c.e. sets, any element may be removed after it is added, but after that it cannot be enumerated again.[2] The construction in the following proof is nearly identical to the one we have just seen, but it results in trees of bounded width.

Theorem 4.10. *There exists a d.c.e. family $\mathcal{F}$ of trees with bounded width such that the degree of $G_{\mathcal{F}}$ is $\mathbf{0}'''$.*

PROOF. We will again code $\emptyset'''$ by constructing trees T_e that have two paths if $\Phi_e^{\emptyset''}(e)\downarrow$, and one path if $\Phi_e^{\emptyset''}(e)\uparrow$. This time we allow ourselves to remove an already added string once, resulting in d.c.e. trees.

As before, we calculate $f(e,s)=n$ at each stage s, making n active and creating a new branch deviating from the leftmost path of 0's (i.e., adding $0^{s-1}1$) if n was at that time inactive. And if n was active already, we extend the branch we created for it earlier. But in the current construction, we do not just make all $m>n$ inactive, we also remove all strings in branches created at levels above the starting point of the branch for n. To be precise, if t is the least stage such that n is active on all stages from t to s, we add the string $0^{t-1}10^{s-t}$ and its initial segments, but we remove all strings 0^i1 and their extensions for $i>t$.

Now we still have that if $\lim_s \inf f(e,s)$ exists, there is one smallest outcome that is infinitely often visited, hence infinitely often visited and always active from some stage s on. Then this branch will grow into a path, yielding a tree with two paths (including the leftmost one). Otherwise, if $\lim_s \inf f(e,s)$ does not exist, only the leftmost path remains. Furthermore, there can never be more than two strings at the same level of the tree. Creating a new branch will just add the second string to the highest level; and the moment a branch is extended for an already active n (adding a string to each level between the current level and the level the branch started), all branches in-between are removed. So at any of these levels, only two strings remain, the one in the leftmost path and the one in the branch of n.

Thus the procedure yields for each e a tree with maximum width of two, that has two paths if $e\in\emptyset'''$, and one path if $e\notin\emptyset'''$. Finally, in the construction of T_e every branch can only be created at the highest level s at stage s, meaning that no branch that is removed can be reconstructed. Hence no single string can be removed more than once from T_e, making it a proper d.c.e. tree. □

5. The jump hierarchy

The moral of the previous section is that the complexity of the G-function of a family of trees is generally two jumps higher than the complexity of the family itself. The sharpest expressions of this observation are Corollary 4.2 and Proposition 4.6.

Recall that a set A is high$_2$ if its double jump can compute $\emptyset'''$, so $\emptyset'''\leq_T A''$. A set B is low$_2$ if its double jump is computable by $\emptyset''$, so $B''\leq_T\emptyset''$. The previous observation leads in a very natural way to a characterisation of these two classes.

5.1. High$_2$ and low$_2$ degrees.

Theorem 5.1. *Set A is high$_2$ if and only if there is an A-computable family $\mathcal{F}$ of trees with finitely many paths such that $\emptyset'''\leq_T G_{\mathcal{F}}$.*

PROOF. ($\Longrightarrow$) Assume that we have high$_2$ A, so $\emptyset'''\leq A''$. The function $G_{\mathcal{F}}$ of the A-computable family $\mathcal{F}$ we obtain via Proposition 4.6 computes A''. Hence,

[2]See [Ers68] for a generalisation of this, the Ershov hierarchy of n-c.e. sets.

with our assumption, $\emptyset''' \leq_T G_{\mathcal{F}}$. It follows that there is an A-computable family of trees $\mathcal{F}$ with finitely many paths and $G_{\mathcal{F}}$ that computes $\emptyset'''$.

($\Longleftarrow$) Proposition 3.3 yields that A'' suffices to determine the number of paths of a tree in a family that is computed by A. Then if we have such a family $\mathcal{F}$ of trees where in addition $G_{\mathcal{F}}$ computes $\emptyset'''$ (and we assume this $\mathcal{F}$ exists), we know that $G_{\mathcal{F}}$ is computed by A''. From $\emptyset''' \leq_T G_{\mathcal{F}}$ and $G_{\mathcal{F}} \leq_T A''$ it is immediate that A is indeed high$_2$. □

Given any A-computable family of trees, we can explicitly code A in it. Then we obtain a family of trees that is computed by A and also computes A, so a family that is in fact of the same degree as A. Of course, if the previous family computed $\emptyset'''$, we can make sure the adapted one still does. Conversely, any family of trees that is of the same degree as A naturally computes A. This means that replacing "A-computable" by "of the degree of A" makes no difference to the validity of the statement above. The same holds for the following, translating the statement in terms of degrees.

Corollary 5.2. *A degree* $\mathbf{a} \leq \mathbf{0}'$ *is high*$_2$ *if and only if it computes a family* $\mathcal{F}$ *of trees such that the degree of* $G_{\mathcal{F}}$ *is* $\mathbf{0}'''$.

PROOF. Virtually immediate from Theorem 5.1. The fact that we take $\mathbf{a}$ to be below $\mathbf{0}'$ implies that this particular $G_{\mathcal{F}}$ must in fact have *degree* $\mathbf{0}'''$. For if A is of degree below $\mathbf{0}'$, then the G-function of any family of trees computable from A must also be computable in $\emptyset'''$. This again follows from Proposition 3.3. □

For the degrees that are low$_2$ we also have a nice characterisation in terms of the G-function of families of trees.

Theorem 5.3. *A set A is low*$_2$ *if and only if every A-computable family $\mathcal{F}$ of trees with finitely many paths has* $G_{\mathcal{F}} \leq_T \emptyset''$.

PROOF. ($\Longrightarrow$) Suppose that A is low$_2$, so $A'' \leq_T \emptyset''$. We turn once more to Proposition 3.3, which gives us that for any A-computable family of trees $\mathcal{F}$, the function $G_{\mathcal{F}}$ reduces to A''. Then again transitivity of $\leq_T$ yields $G_{\mathcal{F}} \leq_T \emptyset''$.

($\Longleftarrow$) Here we perform the construction of Theorem 4.6 again on given A, obtaining an A-computable family $\mathcal{F}$ of trees with $G_{\mathcal{F}}$ of the same degree as A''. Then, assuming that every A-computable family of trees has a G-function that is computable in $\emptyset''$, we have that A is low$_2$ from $A'' \leq_T G_{\mathcal{F}}$ and $G_{\mathcal{F}} \leq_T \emptyset''$. □

Corollary 5.4. *A degree* $\mathbf{a}$ *is low*$_2$ *if and only if every family $\mathcal{F}$ of trees with finitely many paths that it computes has $G_{\mathcal{F}}$ computable in* $\mathbf{0}''$.

PROOF. This follows directly from Theorem 5.3. □

The next observation returns to single low$_2$ trees, and is essentially a corollary of Proposition 3.3. It will be put to good use at the end of the chapter in answering the main question.

Proposition 5.5. *Given a tree with finitely many paths and its low*$_2$*-ness index, the number of paths it has is uniformly computable in* $\emptyset''$.

PROOF. The low$_2$-ness index e of tree T gives us the oracle procedure Φ_e that with the help of $\emptyset''$ yields T''. The relativised procedure of Proposition 3.3 describes the algorithm that computes the number of paths of T'', invoking Φ_e with $\emptyset''$ to answer questions about T''. So we can uniformly $\emptyset''$-compute T's number of paths from its low$_2$-ness index. $\square$

5.2. High$_2$ and low$_2$ c.e. degrees. We can state the same characterisations for degrees and families of trees that are c.e. Obviously, in case of a c.e. degree we will still know that it is high$_2$ as soon as we have found a (c.e.) family of trees with G-function of degree $\mathbf{0}'''$ that is computed by it. But for the other direction, producing such a family of trees for a given c.e. degree, we have to make our hands dirty in constructing the family in a computable way. The following construction will help us with that.

Theorem 5.6. *For all c.e. sets A, any A''-computable function is computable in the G-function of some c.e. A-computable family of trees with finitely many paths.*

Proof. Let function g be computable in A''. This is the same as saying that g is a Δ^0_3 function relative to A. We exploit the fact that there exists an approximation procedure for Δ^0_3 functions ([SS90, p. 207]). Namely, for every Δ^0_3 function there must be a total computable function f with two arguments, that approaches our Δ^0_3 as follows. For any $n \in \mathbb{N}$, if we look at the value of the limit infimum (as s goes to infinity) of $f(n,s)$ as a pair of natural numbers (using the inverse of the standard pairing function), and take the first coordinate, then we get the output of the Δ^0_3 function on input n. In our case, that means that for $g \in \Delta^{0,A}_3$, there must be some total A-computable function Φ^A of two arguments such that for all $n \in \mathbb{N}$

$$g(n) = (\liminf_s \Phi^A(n,s))_0$$

(where "$()_0$" denotes the first inverse of the standard pairing function).

Our aim is to enumerate for each $n \in \mathbb{N}$ a tree T_n that has a number of paths equal to $\liminf_s \Phi^A(n,s) + 1$. That way the G-function of the resulting family of trees clearly codes function g. Moreover, since we use A-computable approximations to g in building them, all trees will be computable in A.

Construction. We describe the construction of T_n for fixed $n \in \mathbb{N}$. First of all, the leftmost branch is developed by inserting 0^s at stage s. Then, at the same stage s, we put in T_n all *active* strings 0^u10^j1 and their prefixes.

The active strings, at stage s, are those strings 0^u10^j1 for u, j smaller than s that satisfy

- $\Phi^A(n,u-1)[s] \leq j$;
- $\Phi^A(n,u)[s] > j$;
- $\Phi^A(n,v)[s] \uparrow$ or $\Phi^A(n,v)[s] > j$ for all $u < v < s$.

Verification of paths. Let $\liminf_s \Phi^A(n,s) = m$. We start by showing that tree T_n has the right number of paths, so $\#[T_n] = m+1$.

Suppose that $j < m$ and u is the least number such that $\Phi^A(n,u)$ has grown beyond j, so $\Phi^A(n,v) > j$ for all $v \geq u$. Then the string 0^u10^j1 will be active infinitely often, because from some stage on $\Phi^A(n,u-1) \leq j$ and $\Phi^A(n,u) > j$ have settled (the enumeration of c.e. oracle A has finished within their use), and there will always be new stages s where $\Phi^A(n,v)[s] \uparrow$ or $\Phi^A(n,v)[s] > j$ for all $u < v < s$. To see the latter, suppose that $\Phi^A(n,v)[s] < j$ for some v strictly between u and s. This must be wrong, so apparently the oracle A still has to change within its use; and if a *true enumeration* into A finishes this initial segment for good in a later stage, under the assumption that we have applied the hat trick again, every computation for a v' that uses at least the same part of the oracle will diverge, and computations using smaller segments must be correct, giving outcomes larger than j. Hence, for all these 0^u10^j1 for $j < m$ a path will be developed, and there are at least $m+1$ paths (including the leftmost one) in T_n.

And these are all the paths of T_n, because all other strings 0^u10^j1 will only be active finitely many times. For if $j \geq m$, then there will always be a $v \geq u$ such that the computation $\Phi^A(n,v)$ eventually gives an outcome below j (the limit infimum being $m \leq j$), forever blocking the third condition after some large enough stage. And if $j < m$ but u is different from the least number such that $\Phi^A(n,v) > j$ for all v above it, either $\Phi^A(n,u-1)[s] > j$ (if u is below this number) or $\Phi^A(n,u)[s] \leq j$ (if u is above this number) for sufficient large s. Thus T_n has exactly $m+1$ paths.

Verification of effectiveness in A. It remains to show that our c.e. family of trees is computable by A. Suppose we want to know if string σ is in some T_n. If σ is not a sequence of only 0's (in which case it is part of the leftmost branch, and trivially in T_e), it has to have prefix 0^u1 for some $u \in \mathbb{N}$. With A we can compute $\Phi^A(n,u)$, and find the first stage s_0 where the computation converges with output y, and the segment that is used of A has settled (so $\Phi^A(n,u) = y$ is correct).

If $0^u10^j \prec \sigma$ for some $j \in \mathbb{N}$ that is at least as great as output y, it suffices to look at $T_n[s_0]$ to determine if σ is in T_n. After all, $\sigma \succ 0^u10^j$ with $j \geq y$ can only be added if a string 0^u10^k1 with $k \geq j$ becomes active, and this can never happen after stage s_0 because $\Phi^A(n,u)[s] = y \leq j \leq k$ for $s \geq s_0$ prevents the second condition from being fulfilled.

Otherwise, σ has prefix 0^u10^j1 with $j < y$. Extensions of 0^u10^j1 are only added if this string is active, and are all of the form $0^u10^j10^s$ for stages s. So for any σ with prefix 0^u10^j1 in T_n, we must have $\sigma = 0^u10^j10^k$ for some $k \in \mathbb{N}$. Now we search for a stage s larger than k such that either 0^u10^j1 becomes active at s, or one of $\Phi^A(n,u-1)[s]$ or $\Phi^A(n,w)[s]$ for some $w > u$ converges with correct use and gives outcome $> j$ or $\leq j$, respectively. We must find such a stage, because if 0^u10^j1 never becomes active, total Φ^A must converge to values that violate the conditions for becoming active permanently. We already know the second condition to be satisfied eventually, as $\Phi^A(n,u) = y > j$. Therefore, either $\Phi^A(n,u-1) > j$ (breaking the first condition) or $\Phi^A(n,w) \leq j$ for $w > u$ (the third condition).

In the first case, if 0^u10^j1 becomes active, our $\sigma = 0^u10^j10^k$ is enumerated in T_n. In the other case, we can be sure that 0^u10^j1 will not become active in the future, and if σ is not in $T_e[s]$ at this stage s, it will never be in T_n. Thus we have established whether $\sigma \in T_n$ or not, demonstrating that any T_n in our c.e. family is computable in A. □

With the previous work, the characterisations of high$_2$ and low$_2$ c.e. degrees follow automatically.

Corollary 5.7. *A c.e. degree* $\mathbf{a} \leq \mathbf{0}'$ *is high*$_2$ *if and only if it computes a c.e. family* $\mathcal{F}$ *of trees such that the degree of* $G_{\mathcal{F}}$ *is* $\mathbf{0}'''$.

Proof. ($\Longrightarrow$) Given a c.e. set A that is high$_2$, we want to construct in A a c.e. family of trees with a G-function that codes $\emptyset'''$. The fact that A'' is above $\emptyset'''$ just means that we can compute $\emptyset'''$ with some function that is computable in A''. Applying Theorem 5.6 above, we obtain a c.e. A-computable family $\mathcal{F}$ of trees with finitely many paths with $G_{\mathcal{F}}$ that computes $\emptyset'''$. By Proposition 3.8, function $G_{\mathcal{F}}$ in fact has degree $\mathbf{0}'''$.

($\Longleftarrow$) This direction is an instance of the one in the equivalence of Corollary 5.2. If c.e. degree $\mathbf{a}$ computes a c.e. family of trees $\mathcal{F}$ with $G_{\mathcal{F}} \equiv_T \emptyset'''$, then it follows that it is high$_2$. □

Corollary 5.8. *A c.e. degree $\mathbf{a} \leq \mathbf{0}'$ is low$_2$ if and only if every c.e. family $\mathcal{C}$ of trees with finitely many paths that it computes has $G_{\mathcal{C}}$ computable in $\mathbf{0}''$.*

PROOF. ($\Longrightarrow$) If a c.e. degree $\mathbf{a}$ is low$_2$, then by Corollary 5.4 all families of trees with finitely many paths that it computes, including the c.e. families, have a G-function computable in $\mathbf{0}''$.

($\Longleftarrow$) Let f be the characteristic function of A'' for c.e. A of degree $\mathbf{a}$. Certainly $f \leq_T A''$, and by Theorem 5.6 we derive the existence of a c.e. A-computable family $\mathcal{F}$ of trees with finitely paths such that $f \leq_T G_{\mathcal{F}}$. Under the assumption that the G-function of every c.e. A-computable family of trees with finitely many paths is computable in $\mathbf{0}''$, we may conclude from $A'' \equiv_T f$ and $f \leq_T G_{\mathcal{F}}$ that $\mathbf{a}$ is low$_2$. □

6. From Δ^0_2 trees to c.e. trees

Having seen the general complexity of the G-function of families of computable and c.e. trees and the relation with the jump hierarchy, we now pick up the topic of reducing the complexity of trees. In section 2.2 we considered the possibilities of reducing c.e. and Π^0_1 trees to trees of a lower arithmetical complexity but with the same paths; in this section, we investigate what we can do with Δ^0_2 trees.

Theorem 6.1. *For every Δ^0_2 tree we can uniformly construct a c.e. tree that has the same paths.*

PROOF. Given a Δ^0_2 tree T, we may assume that its computable approximation $(T[s])_{s\in\mathbb{N}}$ is such that if some σ is in $T[s]$, then so are all initial segments $\tau \prec \sigma$. We can adapt an approximation that does not satisfy this by only accepting $\sigma \in T[s]$ if also $\tau \in T[s]$ for all $\tau \prec \sigma$, and changing it to $\sigma \notin T[s]$ otherwise. If T is indeed a tree, this clearly makes no difference in the limit.

Now, assuming such an approximation, we construct c.e. tree T' by enumerating σ as soon as we find $\sigma \in T[s]$ at some stage $s \geq |\sigma|$.

To see that all paths of T will be paths of T', it suffices to notice that T is a subset of T'. Any string that is truly in T will remain in the approximations from some point on, and so is inserted in T' on reaching a state beyond the length of the string.

Conversely, suppose that X is not a path of T. That just means there is some initial segment $\rho \prec X$ that is not in T. On this, too, the approximations will always agree after some stage s_0. But then level s_0 in T' cannot contain an extension of ρ. For if it did, this extension would have to be added after stage s_0. And that is impossible with our approximation, because if the extension is in $T[s]$ then so is ρ itself, which we excluded for $s \geq s_0$. Thus $X \upharpoonright s_0$ is not in T', and $X \notin [T']$. That concludes the proof that c.e. tree T' has the same paths as the given Δ^0_2 tree T. □

Even though we can always find a c.e. tree with the same paths for a given Δ^0_2 tree, we do not have much control over the Turing degree of this c.e. tree. For some Δ^0_2 trees of low computational complexity, we can only find such c.e. trees of maximal complexity. We will show this as a corollary of the more general Theorem 6.2 below.

Theorem 6.2. *In every degree* $\mathbf{a}$ *there exists a tree (with only computable paths) such that every tree with exactly the same paths computes* $\mathbf{a}$.

PROOF. Let $\mathbf{a}$ be some degree, and f a function of this degree with range $\{0, 1\}$. We construct tree T by the outcomes of $f(e)$ for every $e \in \mathbb{N}$. Whenever $f(e) = 0$ we insert all strings $0^{2e}10^n$ (for all $n \in \mathbb{N}$) in T. Otherwise, if $f(e) = 1$, put $0^{2e+1}10^n$ for al n in T.

Our tree T is directly constructed from f, and we can compute any $f(e)$ from T by looking which of $0^{2e}1$ and 0^{2e+1} is in. So $T \equiv_T f$, and T is of the same degree $\mathbf{a}$. Additionally, all its paths, of the form 0^i10^ω, are computable.

Now suppose that T' is some tree that happens to have exactly the same paths as T. We can compute any $f(e)$ from T' as follows. Knowing that either $0^{2e}10^\omega$ or $0^{2e+1}10^\omega$ is a path of T', and that there are no other paths extending $0^{2e}1$ or $0^{2e+1}1$, we just search for a level where one of $0^{2e}1$ and $0^{2e+1}1$ has no extensions anymore. In the first case, the path of T' and T must be $0^{2e+1}10^\omega$, so by the construction of T we can be sure that $f(e) = 1$. In the second case, we know that $f(e) = 0$. That makes f computable in T', so any given tree with the same paths as $T \in \mathbf{a}$ indeed computes degree $\mathbf{a}$. □

We need some standard concepts to derive our corollary from this result. First, we call a function *diagonally noncomputable* if for each $e \in \mathbb{N}$, it gives a different output on e than $\Phi_e(e)$ does (with $(\Phi_e)_{e\in\mathbb{N}}$ the standard enumeration of p.c. functions). Such a function is far from computable because with every partial computable Φ_e it already disagrees on the very value e.

Definition 6.3. A total function $f : \mathbb{N} \to \mathbb{N}$ is *diagonally noncomputable* (d.n.c.) if $f(e) \neq \Phi_e(e)$ for all $e \in \mathbb{N}$.

We call a degree d.n.c. if it contains a d.n.c. function. Such degrees that are also c.e. must be maximally complex by *Arslanov's Completeness Criterion* [Ars81], that states that a c.e. set is complete if and only if it computes a diagonally noncomputable function.

Furthermore, we want to posit a d.n.c. degree that is nevertheless low. This we get by the *Low Basis Theorem* [JS72] that every nonempty Π^0_1 class has a low member. The class of d.n.c. functions with range $\{0, 1\}$ (represented as infinite binary strings),

$$\{f \in 2^\omega \mid \forall e, s\ \phi_e(e)[s] \downarrow\neq f(e)\},$$

is certainly Π^0_1.

Corollary 6.4. *There exists a* Δ^0_2 *tree of low degree (with only computable paths), such that all c.e. trees that have the same paths are of degree* $\mathbf{0}'$.

PROOF. By the discussion above, the Low Basis Theorem asserts the existence of some low d.n.c. degree $\mathbf{a}$. By Theorem 6.2 above there is a tree T of this degree $\mathbf{a}$ (with only computable paths) such that all trees with the same paths compute $\mathbf{a}$; and this is the low (hence Δ^0_2) tree we are looking for. For suppose there is some c.e. tree T' with the same paths as T. Then c.e. T' computes d.n.c. degree $\mathbf{a}$. Applying Arslanov's Completeness Criterion, we conclude that T' must be of degree $\mathbf{0}'$. □

If we forget about computable paths, we can also find such a tree of bounded width.

Corollary 6.5. *There exists a Δ^0_2 tree of low degree with width 1, such that all c.e. trees that have the same path are of degree* $\mathbf{0}'$.

PROOF. Take some low set X of d.n.c. degree (which must exist by the Low Basis Theorem), and let tree T only consist of this path X. Any other tree T' with only this unique path clearly computes (the d.n.c. degree of) X. If, moreover, T' is c.e., Arslanov's Completeness Criterion implies that T' is complete, so in degree $\mathbf{0}'$. □

At the same time, we can find for every Δ^0_2 tree a c.e. tree with the same paths that does not have the power to wtt-compute any d.n.c. function. The construction of such a c.e. tree is an extensive augmentation of the construction in Theorem 6.1, where we did not have to worry about any additional conditions in constructing a c.e. tree from a given Δ^0_2 tree.

The new construction requires a diagonalisation argument over an effective list of all weak truth table reductions. More precisely, this is an enumeration $(\Psi_e; \psi_e)_{e\in\mathbb{N}}$ of functionals associated with partial computable functions that bound their use. The enumeration is effective because we can effectively build a list of all possible pairs of functionals and p.c. functions, and this construction procedure we can modify to make sure that the p.c. functions converge if the associated functionals do, and that their use remains below that prescribed by the functions. Further, since our new tree will act as an oracle, we assume the functionals take as oracles sets of *strings*. Then the use of an oracle in a particular computation is the length of the longest string that was involved in a query.

Theorem 6.6. *For every Δ^0_2 tree, there exists a c.e. tree with the same paths that wtt-computes no diagonally noncomputable function.*

PROOF. The core of our construction of a c.e. tree T' with the same paths as a given Δ^0_2 tree T is still the procedure of Theorem 6.1, but now we also want to satisfy the requirements

$$R_e : \Psi_e^{T'} \text{ is not diagonally noncomputable}$$

for all $e \in \mathbb{N}$, to ascertain that this c.e. T' cannot wtt-compute any d.n.c. function. Here the Ψ_e are taken from the effective enumeration $(\Psi_e; \psi_e)_{e\in\mathbb{N}}$ of all weak truth table reductions that we discussed above.

Fulfilling a requirement R_e means that we have $\Psi_e^{T'}(n) \downarrow= \phi_n(n)$ for some n. In order to achieve this for all requirements, we construct a p.c. function g from values given by the $\Psi_e^{T'}$, and for which we know in advance that it will reflect the computations of the diagonal function. This latter seemingly paradoxical property we obtain from the Recursion Theorem, that gives us an index k such that function $g = \phi_k$ before we even start building it. Now we can define $q(e, n)$ to return the index of a function that for any input value just computes $\phi_e(n)$. So $\phi_e(n) \simeq \phi_{q(e,n)}(m)$ for any $m \in \mathbb{N}$, and in particular $\phi_k(n) \simeq \phi_{q(k,n)}(q(k, n))$. Taking as a shorthand $p(n) = q(k, n)$ we thus have that $g(n) \simeq \phi_{p(n)}(p(n))$ for all inputs $n \in \mathbb{N}$.

Now we want to find for each Ψ_e an input value n on which it converges with oracle T', and define $g(n)$ to return the same output value. Since we do not know exactly when these computations settle, we will have to make repeated attempts. For the strategy of R_e, we will try values of the form $\langle e, n\rangle$ with $n \in \mathbb{N}$, i.e., values in $\mathbb{N}^{[e]}$. At the beginning of each stage s of the construction, we define $n_{e,s}$ to be the least element $n \in \mathbb{N}^{[e]}$ such that $g(n)$ is not yet defined or equals $\Psi_e^{T'}(p(n))[s]$.

We say that requirement R_e *requires attention* at stage s if

- for all $i \leq n_{e,s}$, computation $\Psi_e^{T'}(p(i))[s]$ converges, and
- for all $i \leq n_{e,s}$, if $g(i)$ is defined at the beginning of s then it is different from $\Psi_e^{T'}(p(i))[s]$.

If R_e requires attention, we can be sure that $g(n_{e,s})$ is not yet defined. For otherwise we would have that $g(n_{e,s}) \neq \Psi_e^{T'}(p(i))[s]$. But by the definition of $n_{e,s}$, either $g(n_{e,s})$ is still undefined or $g(n_{e,s}) = \Psi_e^{T'}(p(i))[s]$.

Meanwhile, we have to put strings of our original T in the tree T' we construct to ensure it has the same paths, as we did in the construction of Theorem 6.1. But we do want to have more control over the changes in oracle T', lest the oracle computations keep changing and we will never succeed in fulfilling our requirements. To that end, we employ *movable markers* l_e that point to certain levels of T'. Define marker $l_{e,s}$ at stage s to be the least number strictly above all markers $l_{i,s}$ for $i < e$, and strictly above all defined uses $\psi_e(p(i))[s]$ for $i \leq n_{e,s}$. We construct T' in such a way that changes strictly below level l_e (so within the use $\psi_e(p(n_e))$) are always accompanied by changes below level l_{e-1}. That way, it suffices that the levels up to l_{e-1} have settled to know that the use $\psi_e(p(n_e))$ has stabilised.

Construction. At stage $s + 1$, perform the following.

(1) Look for the least string σ of the precise length $l_k[s] < s$ for some k, such that σ is in $T[s]$ but not in $T'[s]$. When found, enumerate it and all its extensions of length strictly below $l_{k+1}[s]$ into T'.
(2) Let e be the least up to s such that R_e requires attention – this is the requirement with the highest priority. Define $g(n_{e,s}) = \Psi_e^{T'}(p(n_{e,s}))[s]$.

We write $T' \upharpoonright n$ for the set of strings in T' of length less than n.

Verification of requirements. We start with showing that all requirements R_e will be satisfied by our tree T'. Fix an arbitrary $e \in \mathbb{N}$. By induction, assume that for all $i < e$ we have that $l_{i,s}$ and $n_{i,s}$ have limits l_i and n_i, and that R_i will be satisfied. Let s_0 be the first stage where markers $l_{e-1,s}$ have settled at l_{e-1} and $T' \upharpoonright l_{e-1} + 1$ remains constant for all $s \geq s_0$.

If after this stage s_0 our requirement R_e never needs attention, there will be at each $s > s_0$ some value i such that $\Psi_e^{T'}(p(i))[s] \uparrow$ or $\Psi_e^{T'}(p(i))[s] = g(i)$. A function that is not total or equals $\phi_{p(i)}(p(i))$ on $p(i)$ is for a fact not diagonally noncomputable, so in that case we do not have to do any work. We can also be sure that n_e settles after s_0, because if the requirement will not be able to act and define g on any value in $\mathbb{N}^{[e]}$ any longer, there is a smallest $n \in \mathbb{N}^{[e]}$ such that $g(n)$ will always remain undefined. The $n_{e,s}$ can then never grow beyond this value, and change only finitely many times beneath it, as the part of oracle T' that is used by Ψ_e in calculating from $p(n_{e,s})$ for these $n_{e,s}$ has settled at some point.

If R_e does require attention at some later stage s, it will be the requirement of the highest priority. So the construction acts for R_e and defines $g(n_{e,s})$ to be $\Psi_e^{T'}(p(n_{e,s}))[s]$. The latter has to converge, as that was a condition for R_e requiring attention. This equality will be preserved in all later stages, unless oracle T' changes within the use $\psi_e(p(n_{e,s}))[t]$ at a stage $t > s$.

But this cannot happen after stage s_0. For a change in $T' \restriction \psi_e(p(n_{e,s}))[t]$ implies a change in $T' \restriction l_{e-1}+1$, in contradiction with our choice of s_0. To see this implication, observe that the first step of the construction enforces that if an element of length below $l_{e,t}$ is enumerated into T', this is accompanied by the enumeration of an element below length $l_{e-1,t}$. For if this element is not below $l_{e-1,t}$ itself, it is enumerated because it is the extension of a new element of precisely length $l_{e-1,t}$. Hence if $T' \restriction l_{e,t}$ changes, then so will $T' \restriction l_{e-1,t}+1$. And by its definition $l_{e,t}$ is strictly greater than the use $\psi_e^{T'}(p(n_{e,s}))[t]$, so if $T' \restriction \psi_e^{T'}(p(n_{e,s}))[t]$ changes, so does $T' \restriction l_{e,t}$. By stage s_0, marker l_{e-1} and initial segment $T' \restriction l_{e-1}+1$ have settled, so it follows that $T' \restriction \psi_e^{T'}(p(n_{e,s}))[t]$ must have settled as well.

That proves that $g(n_{e,s})$ equals $\Psi_e^{T'}(p(n_{e,s}))$, that is,

$$\Psi_e^{T'}(p(n_{e,s})) = \phi_{p(n_{e,s})}(p(n_{e,s})).$$

Thus $\Psi_e^{T'}$ is certainly not diagonally noncomputable, fulfilling requirement R_e. The later $n_{e,t}$ can never exceed this $n_{e,s}$, so we can again take for granted that n_e settles. To finish the induction, we notice that the limit l_e, the least strictly above all l_i for $i < e$ and all $\psi_e(p(n_e))$ for $i \leq n_e$, will then be reached as well.

Verification of paths. It remains to prove that T' has the same paths as T. The easy direction is that $[T] \subseteq [T']$, owing to the fact that $\sigma \in T$ will be enumerated in T' as soon as all smaller strings in T are enumerated in T'. Hence $T \subseteq T'$.

For the converse, let $X \notin [T]$. So there is some initial segment $\sigma \prec X$ that is not in T, and the approximations will not dispute that after some stage. Nor will any extensions of σ be in later approximations of T, by the enumeration we chose. Now take an even larger stage s_0 such that also all levels up to $|\sigma|$ in T' have settled. Our claim is that no extension of σ of length s_0 can appear in T'. Then $X \restriction s_0$ is not in T', so that would suffice to demonstrate that $X \notin [T']$.

Suppose that some τ of length s_0 is enumerated in T', say at stage s_1+1. Then it is an extension in the cone of some least ρ that is enumerated at the same stage. By the construction, this ρ is enumerated because some extension $\tau' \succcurlyeq \rho$ was found in $T[s_1]$. This τ' is certainly no extension of σ, as these will not appear in approximations of T this far in the construction. Hence prefix ρ is not an extension of σ either, $\sigma \not\preccurlyeq \rho$. Now the length of ρ is larger than that of σ (because T' up to level $|\sigma|$ had settled by this stage), so the fact that $\rho \preccurlyeq \tau$ is not an extension of σ implies that τ is not an extension of σ. Thus any string of length s_0 enumerated in T' is not an extension of σ. Hence we have shown that $X \notin [T']$. That concludes the proof that $[T'] \subseteq [T]$, so $[T'] = [T]$ □

Another version of Arslanov's Completeness Criterion says that a c.e. set is *weak truth table*-complete if and only if it *weak truth table*-computes a d.n.c. function. This leads to the following corollary.

Corollary 6.7. *For any Δ_2^0 tree, there exists a c.e. tree with the same paths that is not wtt-complete.*

Proof. Immediate from Theorem 6.6, with Arlanov's Completeness Criterion that a c.e. set is wtt-incomplete precisely if it does not wtt-compute a d.n.c. function. □

In summary, there exist low Δ^0_2 trees with only computable paths such that any c.e. tree with the same paths must be complete; but even for these Δ^0_2 trees there must still exist c.e. trees with the same paths that are not wtt-complete.

Additionally, we can rephrase the previous result in terms of Kolmogorov complexity. Recall from the previous chapter that an infinitely often K-trivial set has infinitely many K-trivial initial segments, and that every c.e. set is infinitely often K-trivial. Furthermore, we call a set *complex* if a lower bound on the plain Kolmogorov complexity of its initial segments is given by an unbounded non-decreasing computable function.

Definition 6.8. A set A is *complex* if there is a computable order f such that $C(A \upharpoonright n) \geq f(n)$ for all $n \in \mathbb{N}$.

Now in [KHMS06] it was shown that for c.e. sets, being complex is equivalent to being able to compute a d.n.c. function.[3]

Corollary 6.9. *For any Δ^0_2 tree, there exists an infinitely often K-trivial tree with the same paths. Moreover, this tree is not complex.*

Proof. Immediate from Theorem 6.6, with the givens that any c.e. set is infinitely often K-trivial and that it is complex precisely if it does not wtt-compute a d.n.c. function. □

Note that in order to bring a tree, a set of strings, in accordance with the notion of K-triviality, we have to code it as an infinite binary sequence. The standard way is to assign strings a natural number by ordering them first by length and then lexicographically.

7. From K-triviality trees to c.e. K-trivial trees

In this section, we will finally apply our strategy of complexity reduction to the trees we are really interested in, the K-triviality trees $T^{\mathcal{K}}_b$. Being Δ^0_2 trees, we know from the previous section that there exist c.e. trees with the same K-trivial paths. At the end we even saw that we can find such c.e. trees that are infinitely often K-trivial themselves. We will show in a moment that we can in fact reduce these particular Δ^0_2 trees to c.e. trees that are fully K-trivial.

We start with an observation about trees that are computable in a c.e. set.

Proposition 7.1. *For given tree computable in a c.e. set, there exists a computably enumerable tree with the same paths that is computable in the same c.e. set as well.*

Proof. Let T be a tree that is computable in a c.e. set A. Since T is Δ^0_2 as well, we can construct a c.e. tree T' with the same paths precisely as in Theorem 6.1. So under the assumption that we have an approximation to T that gives a tree at any time, we put σ in T' if σ is in $T[s]$ at a stage $s \geq |\sigma|$.

[3]Incidentally, a result in the discussion of infinitely often K-trivials in [BV10] is that a set is infinitely often K-trivial if it is not complex.

This time we we also have the c.e. A that computes T, so $T = \Phi^A$ for some functional Φ. But then we can determine in A for every string σ when the approximation $T(\sigma)[s]$ will stabilise, and remain the same for all s after some s_0. $\square$

This will be useful because of the known fact that every K-trivial set (hence, tree) is computable in some computably enumerable K-trivial set.

Fact 7.2 ([Nie05]). *Every K-trivial set is computable in a c.e. K-trivial set.*

Further, the class of K-trivials is closed under the join operator.

Fact 7.3 ([DHNS03]). *If $X, Y \in \mathcal{K}$, then also $X \oplus Y \in \mathcal{K}$.*

Then we can use the method of Proposition 7.1 of obtaining a c.e. tree to show that for every K-triviality tree there exists some c.e. K-trivial tree with the same paths.

Proposition 7.4. *For all constants b, there is c.e. K-trivial tree that has the same paths as $T_b^{\mathcal{K}}$.*

PROOF. The tree $T_b^{\mathcal{K}}$ only has a finite number n of paths, that are all K-trivial. Let X_i for $i < n$ denote these sets, and let X be the join $\oplus_{i<n} X_i$. Proposition 2.6 gives us a tree R with the same paths as $T_b^{\mathcal{K}}$ and of the same degree as X.

Note that X is K-trivial as well, since the K-trivials are closed under join. We also remarked above that every K-trivial set is computable in some c.e. K-trivial set. Take A to be such a c.e. and K-trivial set that computes X. Since tree R has the same degree as X, c.e. A will also compute R. Now Proposition 7.1 asserts that for tree R computable in c.e. A there exists some *c.e.* tree with the same paths as R that is also computable in A. Call this c.e. tree R'. Being computable in K-trivial A, this new tree R' has to be K-trivial as well. Thus R' is a c.e. K-trivial tree that has the same paths as R, hence as $T_b^{\mathcal{K}}$. $\square$

However, if we are willing to put more work in it, we can improve this result drastically. For it is possible to effectively transform the complete family of K-triviality trees to a family of c.e. K-trivial trees with the same paths.

For this construction, we define the approximation to the K-triviality tree $T_b^{\mathcal{K}}$ at stage s as

$$T_b^{\mathcal{K}}[s] = \{\sigma \mid \forall \tau \preccurlyeq \sigma \ (K(\tau)[s] \leq K(|\tau|)[s] + b)\}.$$

So the approximations are trees and all their strings appear to be K-trivial at the corresponding stage.

We also have to make more explicit our way of coding sets of strings into sets of numbers, that was needed to make sense of K-trivial trees. We define f to be a computable 1-1 function from the set of finite strings $2^{<\omega}$ onto the natural numbers $\mathbb{N}$, ordering all strings first by length and then lexicographically. A set of finite strings can thus be identified with an infinite binary sequence, where the n-th digit signifies whether the string corresponding to n (via the inverse of f) is in or not.

Theorem 7.5. *There is a uniformly c.e. sequence $(T_b)_{b \in \mathbb{N}}$ of trees such that for all $b \in \mathbb{N}$, member T_b has the same paths as $T_b^{\mathcal{K}}$ and is K-trivial via $2b + c$, for some constant c.*

PROOF. With the help of the approximations to the K-triviality trees we aim to computably enumerate for all b a K-trivial tree T_b sharing the same paths with $T_b^{\mathcal{K}}$. The K-triviality of T_b is enforced by building a prefix-free machine M_b that has sufficiently short descriptions. In the end, we will take all these M_b together in the machine $M = \cup_b M_b$, that accounts for the constant c.

The main complication we have to overcome in the construction is the fact that we have to add short descriptions in M_b for the initial segments of (the binary code of) T_b. We must be careful that this does not become too costly, as there is only so much weight we can put in M_b with our descriptions. Our strategy is to exploit the fact that changes in the initial segment of T_b are caused by new strings appearing in the approximations to $T_b^{\mathcal{K}}$, strings that must look K-trivial at that stage themselves. So these strings in $T_b^{\mathcal{K}}[s]$ have short descriptions in the universal machine U, and this gives us some room to put short descriptions in M_b. Indeed, we are sure to be safe if we never put more weight in our machine than there appears in the universal machine.

We define the function g to help us pick at each stage the changed initial segments whose descriptions we want to count against U-descriptions.

$$g(n,s) = \mu k \; \Big(\sum_{k \leq i \leq s} 2^{-K(i)[s]} < 2^{-n} \Big)$$

It is not hard to see from the definition that g is uniformly computable and non-decreasing in n. Moreover, it will have a limit $g(n) = \lim_s g(n,s)$.

CONSTRUCTION. At stage $s+1$, perform the following for every $b < s$.

(1) Search for the least $k < s$ such that that the M_b-description of T_b up to k is too large, $K_{M_b}(T_b \restriction k)[s] > K(k)[s] + 2b$. If such k exists, give a sufficiently short description of this initial segment by enumerating into M_b a description for $M_b \restriction k$ of length $K(k)[s] + 2b$.

(2) Look for the least string $\sigma \notin T_b[s]$ such that both $f(\sigma)$ and $g(f(\sigma), s)$ are smaller than s, and that has an extension $\tau \succ \sigma$ in $T_b^{\mathcal{K}}[s]$ with $g(f(\sigma), s) < f(\tau) < s$. On finding such σ, we enumerate τ and all its initial segments $\succcurlyeq \sigma$ into T_b. (Note that all prefixes $\prec \sigma$ are already in, or σ would not be the least string satisfying the above.)

    This enumeration has changed initial segments $T_b \restriction n$ for $n \geq f(\sigma)$, so for these segments up to length s we enumerate new descriptions of length $K(n)[s] + 2b$ in M_b, making sure that $K_{M_b}(T_b[s+1] \restriction n) \leq K(n)[s] + 2b$ for all $n < s$.

VERIFICATION OF TREES. With the above construction we enumerate the trees in a computable and uniform way. To verify that the construction yields c.e. T_b with all the properties we require, fix a $b \in \mathbb{N}$. First we look at the paths of T_b.

All paths of T_b must be paths of $T_b^{\mathcal{K}}$ because, just as in the construction in Theorem 6.1, the moment the approximation to $T_b^{\mathcal{K}}$ will always exclude a certain initial segment, no extensions of it will ever be put in T_b.

all strings in T_b are in $T_b^{\mathcal{K}}$. This is easy to see from the second step in the construction, keeping in mind that if a string is in $T_b^{\mathcal{K}}[s]$ then so are all its prefixes.

Conversely, take X to be a path of $T_b^{\mathcal{K}}$. We want to show that any initial segment $X \restriction n$ is enumerated in T_b at some stage. So take $\sigma = X \restriction n$ for any $n \in \mathbb{N}$.

Pick a stage beyond n where the initial segment of T_n below $f(\sigma)$ has settled (so membership in c.e. T_b has been decided for every string smaller than σ, in the sense of the ordering by f) and g of $f(\sigma)$ has reached its limit $g(f(\sigma)) = \lim_s g(f(\sigma), s)$. After this stage, we wait until we find an extension τ of σ in $T_b^{\mathcal{K}}$ that is large enough to satisfy $f(\tau) > g(f(\sigma))$. As σ is infinitely extendable in $T_b^{\mathcal{K}}$ and f is increasing on the length of its inputs, we are sure to find such a stage and such a string. Finally, we wait for stage s_0 such that also $s_0 > f(\tau)$. If σ has not been enumerated yet, it will be now. For at this stage we have that both $f(\sigma) < f(\tau) < s_0$ and $g(f(\sigma), s_0) = g(f(\sigma)) < f(\tau) < s_0$ for $\tau \succ \sigma$ in $T_b^{\mathcal{K}}$, and σ is the least such string not yet in T_b. This triggers the second part of the construction to enumerate τ and all its prefixes, including σ, into T_b. Thus $X \restriction n \in T_b$, for all n. That concludes the proof that all paths of $T_b^{\mathcal{K}}$ are in T_b, and indeed that they have exactly the same paths.

It remains to show that T_b is K-trivial. If we can enumerate descriptions as the construction prescribes, we do have that $K_M(T_b \restriction n) \leq K(n)+2b$ for all $n \in \mathbb{N}$. The first step provides for a first description of each initial segment of T_b, and replenishes it if the length $K(n)$ drops too much. At each stage this is done for at most one initial segment, but as $K(n)$ settles for each n at some point, new descriptions have to be given for any segment only finitely often. Any changes in the initial segments themselves are taken care of by the second step, where new descriptions are issued for the $T_b \restriction n$ that are affected by the enumeration of new elements. Thus if we take $M = \cup_b M_b$, we have $K_M(T_b \restriction n) \leq K(n) + 2b$ for all n. But the difficulty lies in verifying that we really have the space to follow the construction in giving all these short descriptions, i.e., that the weight of all descriptions for machine M is bounded.

Verification of bounded weight. First we consider the weight of the descriptions that are produced in step 1 of the construction. There, a new M_b-description of an initial segment $T_b \restriction n$ is given precisely if a description from the universal machine of n shows up that is too short. Then this new M_b-description is $2b$ bits longer than the description from the universal machine. That means that the total weight w_b of all M_b-descriptions issued in step 1 of the construction is bounded by the weight of all U-descriptions, the domain of U, divided by 2^{2b}. But the weight of the domain of U is not more than 1, so the weight of all M_b descriptions issued in step 1 is bounded by 2^{-2b}. Thus $w_b < 2^{-2b} \leq 2^{-b}$.

As for the weight that is spent in the second step, we observe that the larger part can be counted against the weight of unique U-descriptions. Suppose that at stage s we have picked string σ with extension τ. If we view T_b as a binary sequence, the enumeration of new elements corresponds to changes of the digits on positions between and including $f(\sigma)$ and $f(\tau)$. Now consider the initial segments $T_b \restriction i$ for i in the interval $I = [f(\sigma), g(f(\sigma), s)]$. As we demanded that $g(f(\sigma), s) < |\tau| < f(\tau)$, all $T_b \restriction i$ for these i will need new descriptions; and the length of the description issued for $T_b \restriction i$ is $K(i)[s] + 2b$. So their total weight is

$$w'_{b,s} := \sum_{i \in I} 2^{-K(i)[s]-2b}.$$

Let us now look at the prefixes $\tau \restriction i$ for the i in the same interval. This makes sense, as all these i are below $g(f(\sigma), s) < |\tau|$. We have that τ is in $T_b^{\mathcal{K}}[s]$, along with all its prefixes. Then this approximation to $T_b^{\mathcal{K}}$, holding all strings with prefixes that appear K-trivial at this stage, guarantees that $K(\rho)[s] \leq K(|\rho|)[s] + b$ for all these prefixes $\rho \preccurlyeq \tau$. In particular, $K(\tau \restriction i)[s] \leq K(i)[s] + b$ for all our i. Then their total weight is at least

$$\sum_{i \in I} 2^{-K(i)[s]-b} = 2^b \cdot \sum_{i \in I} 2^{-K(i)[s]-2b},$$

which is 2^b times greater than the total weight $w'_{b,s}$ of the descriptions of $T_b \restriction i$ for $i \in I$ we added.

If we can show that during the whole procedure we use every string in $T_b^{\mathcal{K}}$ at most once for this trick, we may claim to have counted these descriptions against unique U-descriptions. Indeed, the $\tau \restriction i$ for $i \in I$ are all strings that are inserted in T_b at stage s, because by $f(\sigma) > |\sigma|$ they are all extensions of σ. None of them was in T_b before (because σ was not), nor will any be enumerated again. So the sets of U-descriptions that are used in different stages are disjoint, and each such set has 2^b more weight than the M_e-descriptions we counted it against. That means that the sum of the $w'_{b,s}$ for all stages s is certainly less than 2^b times the weight of the domain of the universal machine. In short,

$$w'_b := \sum_s w'_{b,s} < 2^{-b} \cdot \mathsf{wgt}(U) < 2^{-b}.$$

Then we are still left with the weight of the new descriptions for $T_b \restriction j$ with $j > g(f(\sigma), s)$. This is where the definition of g comes in. After all, the total weight of all shortest U-descriptions at s of j above $g(f(\sigma), s)$ is bounded by $2^{-f(\sigma)}$. The segments $T_b \restriction j$ receive descriptions that are $2b$ bits longer than the corresponding U-descriptions of the j, so the total weight $w''_{e,s}$ of these M_e-descriptions is bounded by $2^{-f(\sigma)-2b}$. To derive the upper bound on the sum of these weights over all stages, we use that each σ is enumerated in T_b at most once and that f is 1-1 onto $\mathbb{N}$. We get

$$\begin{aligned} w''_b := \sum_s w''_{b,s} &< \sum_s 2^{-f(\sigma_s)-2b} \\ &\leq \sum_{\rho \in 2^{<\omega}} 2^{-f(\rho)-2b} \\ &\leq \sum_{i \in \mathbb{N}} 2^{-i-2b} = 2^{-2b} \sum_{i \in \mathbb{N}} 2^{-i} = 2 \cdot 2^{-2b} \leq 2^{-b+1}. \end{aligned}$$

Bringing everything together, we have that the final total weight of M_b is

$$\mathsf{wgt}(M_b) = w_b + w'_b + w''_b < 2^{-b} + 2^{-b} + 2^{-b+1} < 2^{-b+2}.$$

Hence the total weight of $M = \cup_b M_b$, below the sum of 2^{-b+2} over all $b \in \mathbb{N}$, is bounded by a finite number. That proves that our M really is a prefix-free machine. □

Looking closely at the previous proof, we see that its construction can in fact show something more general.

Corollary 7.6. *There exists constant c such that given Δ^0_2 tree T (by Δ^0_2 index e) and constant b via which all its paths are K-trivial, we can uniformly produce a c.e. tree T' with the same paths as T that is K-trivial via $2b + e + c$.*

PROOF. In applying the construction of Theorem 7.5 to the list of K-triviality trees, the facts that we essentially used were that all the trees are Δ^0_2 and that the paths are K-trivial via a known constant. So with some minor adjustments, we should be able to apply the construction to all combinations of Δ^0_2 indices e and constants b. In the cases that this is a meaningful combination, so $\Phi_e^{\emptyset'}$ gives a tree that has paths K-trivial via b, this construction must uniformly give a c.e. tree with the same paths that is K-trivial via a specific constant.

In the original construction we enumerated for the new tree for index b descriptions that were $2b$ bits longer than the universal descriptions of the lengths; now we enumerate for the new tree for index-constant combination $\langle e, b\rangle$ descriptions that are $2b + e$ bits longer than the lengths. Then if $\langle e, b\rangle$ indeed yields a valid Δ^0_2 index e for a tree that is K-trivial via b the weight of machine $M_{\langle e,b\rangle}$ is bounded by 2^{-b-e+2}. By simply prohibiting the machine for $\langle e, b\rangle$ to receive more than 2^{-b-e+2} weight, we ensure that we do not spend too much if the combination $\langle e, b\rangle$ is not valid. The final machine $M = \cup_{e,b} M_{\langle e,b\rangle}$ has a weight bounded by

$$\sum_{b,e} 2^{-b-e+2} = \sum_{b} (2^{-b+2} \cdot \sum_{e} 2^{-e}) = \sum_{b} 2^{-b+3} < \infty,$$

so the construction is feasible. If we let c be the coding constant of M, the complete procedure yields for Δ^0_2 tree with index e and constant b via which it is K-trivial a c.e. tree with the same paths that is K-trivial via $2b + e + c$. □

8. The lowness indices of K-trivial sets

We now know that we can transform our sequence of K-triviality trees in a c.e. sequence of K-trivial trees. These trees are particularly nice to work with because of the fact that all K-trivials are low.[4]

Fact 8.1. *Every K-trivial set is low.*

For this fact means that all K-trivials are also low$_2$, and we have seen in Proposition 5.5 that the number of paths of a tree with given low$_2$-ness index is already computable in $\emptyset''$.

But then we still have to find a way of obtaining the actual lowness indices. We first show that the information of $\emptyset'$ will not do in finding the lowness-index of a K-trivial set, as a consequence of the following theorem. This theorem is an extension of [Nie06, Theorem 5.1], where it is proven that we can construct a c.e. set that is not computable in a given low c.e. set and its lowness index. Actually, the possibility of this extension is noted in [Nie06, Corollary 5.7], but not given a proof.

In our construction, as well as in the original, a K-trivial set is built via the elegant method of *cost functions*.[5] Here we define the cost function

$$c(x, s) = \sum_{x<i\leq s} 2^{-K_s(i)}$$

[4]Directly proven in [Nie09, Section 5.4] as an introduction to the *decanter method* that is ultimately used to show that every K-trivial is low for random.

[5]Introduced in [KT99] to construct an incomputable c.e. low for random set, and subsequently used in [DHNS03] to build an incomputable c.e. K-trivial set. Both provide a solution to Post's Problem whether there exists an incomputable but noncomplete c.e. set.

to represent the cost of the potential enumeration of x into the set A we are building. We try to make sure that the sum

$$S = \sum_{x,s} \{c(x,s) \mid x = \mu y(y \in A[s] - A[s-1])\} \tag{1}$$

of the costs of the *least* elements enumerated at each stage is bounded (if multiple elements are enumerated at the same time, clearly their total cost is covered by the cost function of the least one). If we succeed, we say that the enumeration *obeys* the cost function. By [Nie09, Theorem 5.3.10], this is sufficient to ensure that our set is K-trivial.

Theorem 8.2. *For an effective sequence $\{(B_i, J_i)\}_{i\in\mathbb{N}}$ of pairs of low c.e. sets and approximations to their lowness indices, we can effectively produce a K-trivial set that is not computable in any of the sets B_i.*

PROOF. In order to be incomputable in any of the c.e. B_i of the effective sequence, we want the K-trivial set A we construct to satisfy the requirements

$$R_{\langle e,i\rangle} : A \neq \Phi_e^{B_i}$$

for all $e, i \in \mathbb{N}$.

The obvious way to meet $R_{\langle e,i\rangle}$ is to enumerate an $x \in \mathbb{N}^{[\langle e,i\rangle]}$ in A if we find that $\Phi_e^{B_i}(x) = 0$; but as B_i may change, we can never be sure we arrived at the definite computation. However, we know that B_i is low, and this provides us a handle for guessing whether the computation is correct.

We use auxiliary sets of oracles, in which we enumerate the use of the corresponding B_i in the oracle computation if we find an x that could serve in the diagonalisation. We would like to be able to ask if this use is a correct initial segment of B_i, so we can be sure the computation will not change. What we do is asking whether there is an initial segment of B_i that will be enumerated in the auxiliary set. This question is c.e. in B_i, which means, B_i being low, that it is Δ^0_2. So there is a computable approximation to this question, which we can uniformly obtain from the correct lowness index of B_i.

But we are only given a computable approximation $J_i = (j_{i,s})_{s\in\mathbb{N}}$ to lowness index j_i of B_i. To keep track, we let $m_{i,s}$ be the total number of changes in the approximation to the lowness indices of B_i by stage s. For each requirement $R_{\langle e,i\rangle}$, we will start enumerating a new auxiliary set $D_{e,i,m_{i,s}}$ if the lowness index changes.

Now the Δ^0_2 question that we ask to look ahead if we have the correct use is as follows:

"is use σ an initial segment of $\Phi^{\emptyset'}_{j(m)}$, if the lowness index $j(m)$ after m number of changes is the right one, so indeed $\Phi^{\emptyset'}_{j(m)} = B_i'$?"

We assume it is approximated by computable g, so

$$\lim_s g(e,i,m,s) = \begin{cases} 1 & \text{if } \Phi^{\emptyset'}_{j(m)} = B_i' \Rightarrow \exists \sigma \prec \Phi^{\emptyset'}_{j(m)}\ (\sigma \in D_{e,i,m}) \\ 0 & \text{otherwise} \end{cases}$$

CONSTRUCTION. At stage $s > 0$, perform the following for each $\langle e,i\rangle < s$. Look for the least $x \in \mathbb{N}^{[\langle e,i\rangle]}$ such that

$$\Phi_e^{B_i}(x)[s] = 0 \ \&\ c(x,s) \leq 2^{-(\langle e,i\rangle - n)},$$

with $n = \#(A[s-1] \cap \mathbb{N}^{[\langle e,i\rangle]})$ the number of elements enumerated in A for this requirement so far. So this is an element that seems not to be in $\Phi_e^{B_i}$ and would not add too much weight to A. First put use $B[s] \upharpoonright u$ with $u = \phi_e^{B_i}(x)[s]$ in $D_{e,i,m_{i,s}}$. Now if this use of B is correct, it looks like the $\Sigma_1^{B_i}$ question has a positive answer, and $\lim_s g(e,i,m_{i,s},s) = 1$. We try to certify this by looking for the first stage $t \geq s$ such that $g(e,i,m_{i,t},t) = 1$, or there is either a change in the used part of B, so $B[t] \upharpoonright u \neq B[s] \upharpoonright u$, or in the approximation to the lowness index, $j_{i,t} \neq j_{i,s}$. In the latter cases, we see it has failed; but in the first case, there is some evidence we are on the right track, so in that case we insert x in A.

VERIFICATION. To verify $R_{\langle e,i\rangle}$, assume towards a contradiction that $A = \Phi_e^{B_i}$. Let m_i be the number of times the approximation to the lowness index of B_i has changed before it settles at the right index j_i. We can distinguish two cases: either $\lim_s g(e,i,m_i,s) = 1$ or $\lim_s g(e,i,m_i,s) = 0$.

In the first case, there will be a $\sigma = B_i[s] \upharpoonright u$ enumerated in D_{e,i,m_i} that is in fact a correct initial segment of B_i, as a result of finding an x such that $\Phi_e^{B_i}(x)[s] = 0$. Since $u = \phi_e^{B_i}(x)[s]$ is correct, we can be sure that $x \notin \Phi_e^{B_i}$. Moreover, in waiting for confirmation, we will never see this segment of B_i change after s; and neither will the approximation to the lowness index change because it has already changed m_i times. But since $\lim_s g(e,i,s) = 1$, at some later t we will definitely see $g(e,i,t) = 1$. At that point, x is enumerated in A, and we have $A(x) \neq \Phi_e^{B_i}(x)$ after all.

Otherwise, if $\lim_s g(e,i,m_i,s) = 0$, we must have that $g(e,i,m_i,s) = 0$ for all s beyond some s_0. We can take s_0 such that j_i has settled as well. There is no way that any $x \in \mathbb{N}^{[\langle e,i\rangle]}$ can be enumerated in A at a later $t \geq s_0$, because we are not allowed to do so unless $g(e,i,m_i,t) = 1$. Say there were n numbers enumerated for the requirement in A before this stage. With our assumption that $A = \Phi_e^{B_i}$, we are sure to find $x \in \mathbb{N}^{[\langle e,i\rangle]}$ and $s \geq s_0$ such that $\Phi_e^{B_i}(x)[s] = \Phi_e^{B_i}(x) = 0$ with correct use and also $c(x,s) \leq 2^{-(\langle e,i\rangle - n)}$. Then the construction will enumerate the correct segment $\sigma \prec B_i$, which means that $\lim_s g(e,i,m_i,s) = 1$, contrary to assumption.

For the verification of the K-triviality of A, we only have to show that the enumeration obeys the cost function. But every time we enumerate an x for a requirement $R_{\langle e,i\rangle} : A \neq \Phi_e^{B_i}$, and n elements have been enumerated for this requirement before, we have made sure that the cost $c(x,n)$ is not more than $2^{-(\langle e,i\rangle + n)}$. Then S as defined in 1 is certainly not more than $\sum_{\langle e,i\rangle} 2^{-(\langle e,i\rangle)}(\sum_n 2^{-n}) = \sum_{\langle e,i\rangle} 2^{-(\langle e,i\rangle+1)} = 4$, hence bounded. Thus the cost function is obeyed, and A will be K-trivial. □

It is a result from [DHNS03] that there is an effective listing of the c.e. K-trivial sets with constants (not necessarily the least ones).

Fact 8.3. *There is an effective sequence of pairs of Δ_2^0 indices and constants such that for each c.e. K-trivial set there is an index and the set is K-trivial via the associated constant.*

That shows that we cannot find the lowness index via the K-triviality constant.

Corollary 8.4. *The lowness index of a given set that is K-trivial via given constant is not uniformly computable in $\emptyset'$.*

PROOF. Assume for a contradiction that we know how to uniformly compute in $\emptyset'$ the lowness index from any given K-trivial set and its associated constant. Then we can transform the effective sequence of all c.e. K-trivial sets with constants into an effective sequence of all c.e. K-trivial sets with Δ^0_2 approximations to their lowness indices. But by Theorem 8.2 above, that implies the existence of a K-trivial set that is not computable in any of the sets in that sequence. As every K-trivial set must be computable in some c.e. K-trivial set, this is impossible. $\square$

Still, we can manage it with the help of $\emptyset''$.

Proposition 8.5. *The lowness index of a given set that is K-trivial via given constant is uniformly computable in $\emptyset''$.*

PROOF. Let A be some K-trivial set, and let $\Phi^{\emptyset'}_e$ give its characteristic function. We want to define a $\mathbf{0}''$-computable function f that gives a lowness index $f(e)$ for A (so $\Phi^{\emptyset'}_{f(e)} = A'$) from the given index e, using $\emptyset''$.

We will make use of the fact that the sets that are low for K are uniformly low [Nie09, Proposition 5.1.2]. That means that for every such set X, which satisfies $\forall n\ (K(n) \leq K^X(n) + b)$ for some constant b, we can compute the lowness index directly from the associated constant b. Recall that the notions of K-triviality and lowness for K are equivalent, so for our K-trivial X there must also be a constant via which it is low for K. It suffices to uniformly compute this constant (in $\emptyset''$), to show that the required f can indeed be defined.

To find it for our given X, we just try every possible $b \in \mathbb{N}$ in ascending order, each time asking whether $\forall n\ (K(n) \leq K^X(n) + b)$ holds. As $K(n)$ and $K^X(n)$ are both computable in $\emptyset'$, this is a question that is solved by $\emptyset''$. When it is true for b but false for $b-1$, we will know we have found our b.

Thus on given e such that $\Phi^{\emptyset'}_e = X$ for K-trivial X, function f first uses $\emptyset''$ to compute the constant b for which X is low for K, and then executes the function that uniformly computes a lowness index from this constant. $\square$

As explained in the beginning of the section, it is the low$_2$-ness indices that we are really interested in. But it is a standard computability theoretic fact that if we have the index of a reduction from A to B we can obtain one of the reduction from A' to B'.

Corollary 8.6. *The low$_2$-ness index of a given set that is K-trivial via given constant is uniformly computable in $\emptyset''$.*

PROOF. By Theorem 8.5 we $\emptyset''$-uniformly obtain index i such that $\Phi^{\emptyset'}_i = A'$ for given K-trivial A and constant. Consider the $\emptyset'$-executable program that for given e looks for initial segment $\tau \prec \emptyset'$, stage s and length n that give $\sigma = \Phi^{\tau}_i[s] \upharpoonright n$ (an initial segment of A') such that $\Phi^{\sigma}_e(e)[s]\downarrow$. This program will terminate if and only if $e \in A''$, and $\emptyset''$ can decide if it terminates. The index of the program that decides this is a low$_2$-ness index of A. $\square$

9. The complexity of $G_{\mathcal{K}}$

We can finally bring the results of the previous sections together to determine whether our $G_{\mathcal{K}}$ is in fact Δ^0_3. Recall from Corollary 4.3 that it is certainly Δ^0_4. For completeness' sake, we first also give the proof that it is *not* in Δ^0_2.

In the first section of this chapter, we already noted that $G_{\mathcal{K}}(b) = O(2^b)$. This can be made even stronger.

Fact 9.1 ([DH10, Theorem 10.1.11]). $\sum_{b\in\mathbb{N}} G_{\mathcal{K}}(b)/2^b < \infty$.

It follows that
$$\lim_b \frac{G_{\mathcal{K}}(b)}{2^b} = 0.$$
This fact is exploited in the proof that $G_{\mathcal{K}} \notin \Delta^0_2$.

Proposition 9.2 ([DH10, Theorem 10.1.13]). *The function $G_{\mathcal{K}}$ is not Δ^0_2.*

Proof. We assume that $G_{\mathcal{K}}$ does have an effective approximation $(G_s)_{s\in\mathbb{N}}$, and derive a contradiction from that. This we do by employing this approximation in the construction of a prefix-free machine that renders more sets K-trivial via a certain constant than $G_{\mathcal{K}}$ actually asserts.

As mentioned before, by the Recursion Theorem we may assume we know in advance the index d of the machine we construct. Now take r to be a constant via which the computable set $\emptyset$ is K-trivial. Let b_s be the least number above r such that
$$\frac{G_s(b_s)}{2^{b_s}} < 2^{-d}.$$
Then b_s might not be defined for all s – but for our purposes it is enough that it is for sufficiently large s, and this is guaranteed by the result that $G_{\mathcal{K}}(b)/2^b$ approaches zero as b increases.

Construction. At stage $s+1$, see if the approximation to b has changed, so $b_s \neq b_{s-1}$.

- If it does, we start a new attempt to build too many K-trivial sets. Define the sets A_σ as $0^s 1 \sigma 0^\omega$ for all σ of length $b_s - d$.
- If b does not change at this stage, so $b_s = b_{s-1}$, let s_0 be the last stage where b did change. We try keeping the original sets A_σ K-trivial via b_{s_0}. Note that by the choice of b_s above the constant r via which $\emptyset$ is K-trivial, we do not have to worry about the initial segments 0^n for $n \leq s_0$. Therefore we only enumerate for the n with $s_0 < n \leq s$ new descriptions of $A_\sigma \upharpoonright n$ of length $K(n)[s] + b_s - d$ into M (if indeed still $K_M(A_\sigma \upharpoonright n)[s] > K(n)[s] + b_{s_0} - d$).

Verification. The weight of M is bounded by that of the universal machine, because for every $n \in \mathbb{N}$, only the initial segments $A_\sigma \upharpoonright n$ for the σ of length $b_s - d$ with $s = n$ get new descriptions. For if this stage has not been reached yet, no initial segments of this length will be considered; on reaching this stage, as long as the approximation $b_{s_o} = b_s$ stays the same new descriptions of the current $A_\sigma \upharpoonright n$ may be given; and if later on the approximation b_t of b has changed from b_s, only new descriptions for initial segments of length greater than $t > s$ are enumerated. The initial segments up to n of the $2^{b_s - d}$ different A_σ all receive a description of length $K(n)[s] + b_s - d$, so in the end there is no more weight added to M than the descriptions of n add to the universal machine.

Now at some point b_s settles (so it is the least such that $G_{\mathcal{K}}(b_s)/2^{b_s} < 2^{-d}$), and we will no longer have to redefine the A_σ. We can be sure that $K(A_\sigma \upharpoonright n) \leq K(n) + b_s$ for $n \leq s$ because $A_\sigma \upharpoonright n = 0^n$ and $K(0^n) \leq K(n) + r \leq K(n) + b_s$.

For $n > s$, the construction ascertains that $K_M(A_\sigma \upharpoonright n)[s] = K(n)[s] + b_s - d$, so $K(A_\sigma \upharpoonright n)[s] = K(n)[s] + b_s$. Thus $\forall n\ (K(A_\sigma \upharpoonright n) \leq K(n) + b_s)$ for all our A_σ, and we have $2^{b_s - d}$ sets that are K-trivial via b_s. But that is in conflict with the value $G_{\mathcal{K}}(b_s)$ gave previously, since $2^{b_s-d}/2^{b_s} = 2^{-d}$.

We conclude that there can be no computable approximation to function $G_{\mathcal{K}}$, so it is not a Δ^0_2 function. □

We have arrived at the answer to our question.

Theorem 9.3. *The function $G_{\mathcal{K}}$ is Δ^0_3.*

Proof. Our function $G_{\mathcal{K}}$ behaves exactly the same as the G-function corresponding to the uniformly c.e. family of K-trivial trees of Theorem 7.5. This function is computable in $\emptyset''$, as follows. Given $b \in \mathbb{N}$, we can determine a $\Phi_e^{\emptyset'}$ that computes the tree T_b and by Corollary 8.6 we can then uniformly compute its low$_2$-ness index. Finally, by Proposition 5.5, we uniformly compute the number of paths.

Thus $G_{\mathcal{K}}$ is computable in $\emptyset''$ as well. That makes it Δ^0_3. □

Now that we have established the arithmetical complexity of $G_{\mathcal{K}}$, we may still wonder how strong it is as an oracle. Is the halting set, or even the double jump, encoded in the information about the number of K-trivials via each constant? This is a question that could still depend on the particular universal machine we choose.

Question 9.4. Is $\emptyset'$ or even $\emptyset''$ computable in $G_{\mathcal{K}}$? Does this depend on the choice of the underlying universal prefix-free machine?

10. The number of low for K sets

Set A is low for K via constant b if A as an oracle will not help to compress any string more than b bits. So the class $\mathcal{M}_b$ consists of the sets A such that for all strings τ we have $K(\tau) \leq K^A(\tau) + b$. Then the paths of the Δ^0_2 tree

$$T_b^{\mathcal{M}} = \{\sigma \mid \forall s \in \mathbb{N}\ \forall l > |\sigma|\ \forall \tau \in 2^{<|\sigma|}\ \exists \rho \in 2^l\ (\rho \succ \sigma \ \&\ K(\tau) \leq K^\rho(\tau)[s] + b)\}$$

are precisely the sets in $\mathcal{M}_b$. Admittedly, this is not as easy to see as it was for the K-triviality trees.

Proposition 10.1. *Set A is low for K via b if and only if A is a path of $T_b^{\mathcal{M}}$. Moreover, $T_b^{\mathcal{M}}$ is a Δ^0_2 tree.*

Proof. Take any initial segment $A \upharpoonright n$ of given A that is low for K via constant b. Then for every stage s and level $l > n$, for extension $\rho = A \upharpoonright l$ we have $K(\tau) \leq K^\rho(\tau)[s] + b$ from the facts that $K(\tau) \leq K^A(\tau) + b$ and $K^A(\tau) \leq K^\rho(\tau) \leq K^\rho(\tau)[s]$. So all initial segments of A are in the tree, giving $A \in [T_b^{\mathcal{M}}]$.

Conversely, if A is *not* low for K via b, so $K(\tau) > K^A(\tau) + b$ for at least one τ, there is some part $\sigma \prec A$ that is used in giving such a short description of τ. So, for all extensions $\rho \succ \sigma$, we have $K(\tau) > K^\rho(\tau)[s] + b$ for large enough s. But then initial segment σ cannot be in the tree $T_b^{\mathcal{M}}$, and A is certainly not a path.

To show that the trees are Δ^0_2, we explain how $\emptyset'$ can decide the membership question. Given string σ, it first computes the values of $K(\tau)$ for all shorter strings τ. Then it is a Π^0_1 question whether for all stages s and higher levels l there is an extension $\rho \succ \sigma$ such that $K^\rho(\tau)[s] + b$ is above all the previously computed values, so that is solvable by $\emptyset'$ as well. □

Now the problem of this section is the complexity of the function $G_{\mathcal{M}}$ with

$$G_{\mathcal{M}}(b) = [T_b^{\mathcal{M}}].$$

Remember that in the proof of Proposition 8.5 we had to calculate the low for K constant from a c.e. set's K-triviality constant, and that we needed oracle $\emptyset''$ to do it. Unfortunately, it cannot be done in an effective way.

Fact 10.2 ([DHNS03]). *We cannot effectively obtain a constant d via which c.e. set A is low for K from A and a constant b via which A is K-trivial.*

This means that the result of our previous chapter that the $G_{\mathcal{K}}$ function is in Δ^0_3 is not directly transferable to the current problem.

Still, it is an easy matter to obtain a K-triviality index of a low for K set. Take a prefix-free oracle machine M that on oracle X and output 0^n of the universal machine returns the initial segment $X \upharpoonright n$. Thus a universal machine description for n is turned in an M^X-description for $X \upharpoonright n$. If d is the coding constant of this machine, clearly $K^X(X \upharpoonright n) \leq K(n) + d$. So if A is low for K with constant b, we have that $K(A \upharpoonright n) \leq K^A(A \upharpoonright n) + b \leq K(n) + b + d$ for each $n \in \mathbb{N}$. Hence A is K-trivial via $b + d$.

Fact 10.3 ([Nie09, Proposition 5.2.3]). *There exists a constant d such that for all sets A, if A is low for K via constant b, then it is K-trivial via constant $b + d$.*

That means that always $G_{\mathcal{M}}(e) \leq G_{\mathcal{K}}(e + d)$, and we can directly infer the following fact from its counterpart Fact 9.1 for the $G_{\mathcal{K}}$ function.

Fact 10.4. $\sum_{b \in \mathbb{N}} G_{\mathcal{M}}(b)/2^b < \infty$.

This fact allows us to show that $G_{\mathcal{M}}$ is not Δ^0_2 in a way that has a resemblance to how we proved Proposition 9.2 that $G_{\mathcal{K}} \notin \Delta^0_2$.

Theorem 10.5. *The function $G_{\mathcal{M}}$ is not Δ^0_2.*

Proof. From the assumption that we do have a computable approximation $(G_s)_{s \in \mathbb{M}}$ of $G_{\mathcal{M}}$, we construct a prefix-free oracle machine M that makes more sets low for K via a certain constant than originally given by $G_{\mathcal{M}}$. The Recursion Theorem gives us a constant d of this machine in advance. By Fact 10.4 above, at every stage s we let b_s be the least such that

$$\frac{G_s(b_s + d)}{2^{b_s + d}} < 2^{-d}$$

Construction. At stage $s = 0$, choose an anti-chain of 2^{b_0} pairwise incomparable strings $\sigma_{0,i}$ $(i < 2^{b_0})$. At any later stage $s + 1$, check if $b_s = b_{s-1}$.

- If that is still the case, let $\sigma_{s,i} = \sigma_{s-1,i}$ for all $i < 2^{b_s}$. We want to give a new M-description for every string τ of length strictly smaller than s. The M-description for τ must be smaller than that of the universal machine with any oracle $\sigma_{s,i}0^\omega$, up to constant b_s. So

$$K_M(\tau)[s] \leq \min_{i < 2^{b_{s,i}}} K^{\sigma_i 0^\omega}(\tau)[s] + b_s$$

for all τ with $|\tau| < s$.

- If not, $b_s \neq b_{s-1}$, pick one $\sigma_{s-1,j}$ such that the weight of M so far is less than the weight of the universal machine relativised to $\sigma_{s-1,j}0^\omega$ for this j, so
$$\mathsf{wgt}(M[s-1]) < \mathsf{wgt}(U^{\sigma_{s-1,j}0^\omega}[s-1]).$$
Subsequently, take 2^{b_s} pairwise incomparable extensions of this $\sigma_{s-1,j}$, and let them be the new $\sigma_{i,s}$ for $i < 2^{b_s}$, abandoning the previous anti-chain.

VERIFICATION. There are two parts to the verification. First, we have to prove that we can perform the above construction at all; second, we have to show that it achieves what we want.

To prove that the construction can be done, we have to demonstrate that we can give enough sufficiently small descriptions in case $b_s = b_{s-1}$, and that we can find an appropriate j if $b_s \neq b_{s-1}$. This is achieved by proving with simultaneous induction that for all s

$$\mathsf{wgt}(M[s]) \leq 2^{-b_s} \sum_{i<2^{b_s}} \mathsf{wgt}(U^{\sigma_{s,i}0^\omega})[s] \tag{2}$$

and that, if $b_s \neq b_{s-1}$,

$$\exists j < 2^{b_{s-1}}(\mathsf{wgt}(M[s-1]) \leq \mathsf{wgt}(U^{\sigma_{s-1,j}0^s}[s-1])). \tag{3}$$

In the base case, $s = 0$, (2) and (3) hold trivially because M is still empty and there is no earlier stage where b changed. For the induction step, assume that they hold at stage $s-1$.

Suppose that we have that $b_s \neq b_{s-1}$ for this stage s. It is not possible that (3) fails, because in that case $\mathsf{wgt}(M[s-1]) > \mathsf{wgt}(U^{\sigma_{s-1,j}0^s}[s-1])$ for all $j < 2^{b_{s-1}}$. Then certainly

$$\mathsf{wgt}(M[s-1]) > 2^{-b_{s-1}} \sum_{i<2^{b_{s-1}}} \mathsf{wgt}(U^{\sigma_{s-1,i}0^\omega})[s-1],$$

contrary to the induction hypothesis that (2) holds at $s-1$.

Moreover, $\mathsf{wgt}(M[s]) = \mathsf{wgt}(M[s-1])$ because there are no descriptions enumerated in M at this stage. The new $\sigma_{s,i}$ for $i < 2^{b_s}$ are all extensions of a string $\sigma_{s-1,j}$ of the previous anti-chain that did satisfy (3). Hence

$$\mathsf{wgt}(U^{\sigma_{s,i}0^\omega}[s]) \geq \mathsf{wgt}(U^{\sigma_{s-1,j}0^\omega}[s-1]) \geq \mathsf{wgt}(M[s-1]) = \mathsf{wgt}(M[s])$$

for all $i < 2^{b_s}$, satisfying (2) for s.

If, on the other hand, we have $b_s = b_{s-1}$ for our s, condition (3) is trivially met. To verify that (2) holds as well, observe that any increase in the weight of M is due to a new description of a length m appearing in one of $U^{\sigma_{s,i}0^\omega}[s]$ for $i < 2^{b_s}$, causing an increase of 2^{-m} in $\sum_{i<2^{b_s}} \mathsf{wgt}(U^{\sigma_{s,i}0^\omega}[s])$. The increase in the weight of M for this description is 2^{-m-b_s}. Thus, by the induction hypothesis that (2) holds for $s-1$, the weight of M is still below 2^{-b_s} times the total weight of all $U^{\sigma_{s,i}0^\omega}$. It follows that (2) will also hold for s. That completes the induction, and the proof of (2) and (3). Thus the construction is valid.

It is left to show that the construction indeed gives too many low for K sets. As $b = \lim_s b_s$ must settle at some point, there are only a finite number of stages s where $b_s \neq b_{s-1}$. At the last such stage, the final anti-chain $(\sigma_i)_{i<2^b}$ is formed. Then the construction guarantees that $K_M(\tau) \leq \min_{i<2^{b_s}} K^{\sigma_i 0^\omega}(\tau) + b$ for all strings τ. Hence, with d the coding constant of M, we have $K(\tau) \leq K^{\sigma_i 0^\omega}(\tau) + b + d$ for all τ and all of our σ_i. But then the 2^b sets $\sigma_i 0^\omega$ are all low for K via constant $b + d$, and we must have $G_{\mathcal{M}}(b + d) \geq 2^b$. But this contradicts the fact that we took the b_s such that for $b = \lim_s b_s$ we have $G_{\mathcal{M}}(b + d) < 2^{b+d}2^{-d} = 2^b$.

Thus a computable approximation of $G_{\mathcal{M}}$ must lead to an absurdity, and we conclude that $G_{\mathcal{M}} \notin \Delta^0_2$. □

Fact 10.3 will also lead the way to a positive solution of the complexity of $G_{\mathcal{M}}$.

Theorem 10.6. *The function $G_{\mathcal{M}}$ is Δ^0_3.*

PROOF. Since we know of a constant d such that every set that is low for K via b is K-trivial via $b + d$, we have that $[T^{\mathcal{M}}_b] \subseteq \mathcal{K}_{b+d}$. But then we can apply Corollary 7.6 from the previous chapter to convert our Δ^0_2 family of trees $(T^{\mathcal{M}}_b)_{b\in\mathbb{N}}$ to a c.e. family of trees $(T_b)_{b\in\mathbb{N}}$ that are K-trivial via uniformly obtained constants.

Now the proof proceeds as Theorem 9.3 about $G_{\mathcal{K}}$. With the help of oracle $\emptyset''$, by Corollary 8.6 we can uniformly compute the low$_2$-ness index of tree T_b, and Proposition 5.5 then says that we can compute its number of paths. Thus we can compute $G_{\mathcal{M}}(b)$ on any b in $\emptyset''$, so indeed $G_{\mathcal{M}} \in \Delta^0_3$. □

Conclusion

In this thesis, we concentrated on two separate topics in the field of algorithmic randomness in general and about sequences with trivial initial segment complexity in particular. By the overview in the introductory first chapter and the range of methods and concepts we encountered in the two following chapters, I hope to have given the reader an appreciation of both the scope and depth of the general subject.

As a poignant example of the intertwining of algorithmic randomness with computability theory, in the second chapter we transferred the technique of splitting in c.e. Turing-degrees to the c.e. degrees induced by reducibilities with respect to randomness – degrees that contain sequences with quite trivial initial segment complexity because c.e. sets are not very random. After introducing the classical construction of the Sacks Splitting Theorem for splitting a noncomputable c.e. set into two c.e. sets strictly Turing-below the original one, we showed how adapted versions can be used to split c.e. sets that are not low for random into c.e. sets of strictly lower LR-degree and of incomparable LR-degree, respectively. We concluded with presenting a construction that likewise splits c.e. sets of the nontrivial C- or K-degrees into sets of both strictly lower and incomparable C- or K-degree.

The main part of the thesis, however, consists of the third chapter, that is devoted to a question of Downey, Miller and Yu about the arithmetical complexity of the function that computes the finite number of sets that are K-trivial via given constant. Put more concretely, if we have K-triviality trees $T_b^{\mathcal{K}}$ whose paths coincide with the sets that are K-trivial via b, this function computes the number of paths of any given member of the family $(T_b^{\mathcal{K}})_{b\in\mathbb{N}}$ of trees. Thus, as one part of our strategy, we set out to examine the general complexity of calculating the number of paths of members of certain families of trees, and found that this calculation can generally be done in an oracle that lies two jumps higher than the degree of the family of trees itself. Significantly, if a single tree is low$_2$ we can compute its number of paths in $\emptyset''$.

The other part of our strategy aimed at reducing the complexity of the Δ_2^0 family of K-triviality trees without touching the paths of its members. Discussing such reductions of particular and general types of Π_1^0, Σ_1^0 and Δ_2^0 trees, we finally arrived at a c.e. family of trees with the same paths, such that the trees are K-trivial themselves. Here we connected again with the other approach, noting that all K-trivial sets are low$_2$ and hence we could compute the number of paths of these trees in $\emptyset''$ from their low$_2$-ness indices (that are $\emptyset''$-computable as well). We concluded that the arithmetical complexity of our function is Δ_3^0, thereby settling the question.

Further questions

There are a number of yet unanswered questions we touched upon throughout the thesis.

Density in the c.e. randomness degrees. The work on splitting in the c.e. randomness degrees raised the question about the density of those degrees. For example, we saw that the c.e. LR-degrees are downward dense and upward dense, but the matter of general density is still undecided. For the c.e. C-and K-degrees, the splitting theorems implied their downward density, but it is unknown whether they are dense in general and whether they have maximal degrees. The latter is related to the question whether for given c.e. set there is a second c.e. set with only 0's in the odd positions yet that is still K-above the first one. The c.e. sets are not very complex in the K-sense, lying very close to each other; and having spent some time on them, this is what seems to make these questions difficult to solve.

The trees of K-trivial sets and low for K sets. We already remarked that it is an interesting open question whether we can compute $\emptyset'$ or even $\emptyset''$ with the help of function $G_{\mathcal{K}}$. Likewise for $G_{\mathcal{M}}$. One could further ask if the Δ^0_2 K-triviality trees are in fact complete (they are not m-complete).

In [Nie09, Section 5.2], Nies also considers the function G_{fin} that calculates the number of *finite* (that is, having only 0's from some point on) K-trivial sequences via given constant. This function is also in Δ^0_3. For it is a consequence of our work (as explained in [BB10, Section 3]) that we can in fact obtain in $\emptyset''$ a list of indices of the K-trivial paths of the transformed K-triviality trees in our c.e. family. Then we only have to count how many of them are finite, and it is decidable in $\emptyset''$ whether a given set is finite. It looks more difficult to settle the same question for the function G_{com} related to the computable K-trivials.

It is likely that the function $G_{\mathcal{C}}$ for the C-trivial sets is Δ^0_3 via the same methodology.

The number of low for random sets. We have defined hierarchies for the K-trivial and the low for K sets, and established the arithmetical complexity of the function that returns the number of members of each given level of these hierarchies. What about the remaining lowness notion that we treated in this thesis, the class $\mathcal{L}$ of low for random sets A with $\mathsf{MLR}^A = \mathsf{MLR}$? This definition does not immediately suggest a parametrisation as in the cases of the K-trivials and the low for K's.

One way to address this problem, though it remains somewhat artificial, would be to use the equivalent definition that there exists a member U^A of a universal Martin-Löf test relative to A with a c.e. set W of strings such that $\mu(W) < 1$ and $U^A \subseteq W$, gotten by Fact 2.1 of Chapter 2 since the low for randoms are precisely the sets $A \leq_{LR} \emptyset$. Then we could fix an oracle universal ML-test $(U_b)_{b\in\mathbb{N}}$, and let $\mathcal{L}_e$ contain sets A such that $U^A \subseteq W_e$ for a particular member U of the test and this specific W_e of a computable enumeration $(W_e)_{e\in\mathbb{N}}$ of all c.e. sets of strings with measure below one.

In [BLS08a], a particular universal ML-test $(U_b)_{b\in\mathbb{N}}$ is constructed with the property that for every other oracle ML-test $(V_b)_{b\in\mathbb{N}}$ we can uniformly compute $k \in \mathbb{N}$ such that $V_{b+k} \subseteq U_b$. For any member U of this test, there are for each c.e. W with $\mu(W) < 1$ only finitely many A such that $U^A \subseteq W_e$, as shown in [BLS08a, Theorem 7.1]. An extension of the construction in this theorem even yields the specific bound 2^{c+e} on the number of A with $U^A \subseteq W_e$, for some constant c. If we then define $G_{\mathcal{L}}(e)$ to return $\#\{A \mid U^A \subseteq W_e\}$, we have $G_{\mathcal{L}}(e) \leq 2^{c+e}$.

One could likewise ask for the complexity of $G_{\mathcal{L}}$.

The set of K-trivial strings. In the previous chapter, we extended the definition of K-triviality to finite strings in order to define the trees $T_b^{\mathcal{K}}$ as the sets of strings with only K-trivial initial segments. By concentrating on the strings themselves and disregarding the initial segments, we obtain the sets

$$\mathcal{K}_b^{<\omega} = \{\sigma \mid K(\sigma) \leq K(|\sigma|) + b\}$$

of finite strings K-trivial via b. In the same way, $\mathcal{C}_b^{<\omega} = \{\sigma \mid C(\sigma) \leq C(|\sigma|) + b\}$ is the set of strings that are C-trivial via b. This we can put in contrast with what is usually taken to be the collection of nonrandom strings, the set

$$\overline{R_C} = \{\sigma \mid C(\sigma) < |\sigma|\}.$$

By an intricate argument, Kummer has shown that $\overline{R_C}$ is truth table complete [Kum96]. The answer to the question for its prefix-free counterpart $\overline{R_K}$ depends on the choice of the underlying prefix-free machine [MP02].

With respect to the parallel question whether the sets $\mathcal{K}_b^{<\omega}$ and $\mathcal{C}_b^{<\omega}$ are truth table complete (also related to the question of the completeness of the K-triviality trees), we can claim they are wtt-complete (even uniformly in the constant b) but not m-complete. Due to considerations of space and time, the proofs are omitted from this thesis; they may be included in a future paper. The question about tt-completeness is still open.

Samenvatting

Het vakgebied van algoritmische *randomness* komt voort uit de vraag hoe we op een formele wijze *random* (toevallige, willekeurige) objecten kunnen herkennen, waarbij we ons voor het gemak beperken tot (oneindige) reeksen van nullen en enen. Intuïtief zien we dat een regelmatige binaire reeks als 0101010101010101... zeker niet toevallig is, in tegenstelling tot een reeks als 1011010111101010111... die we bijvoorbeeld ontwikkelen door een munt op te gooien. De eerste heeft een voorspelbaar verloop, bezit duidelijke patronen en is eenvoudig te beschrijven, eigenschappen die de tweede allemaal mist. Maar het definiëren van "toevallig" als het ontbreken van elk patroon voert onvermijdelijk tot een tegenspraak; sterker, we kunnen ons afvragen of het vangen van het fenomeen toeval of willekeur in een exacte definitie niet op zich al contradictoir is.

Een oplossing wordt gevonden in de discipline van de *berekenbaarheidstheorie* (of *recursietheorie*), welke gericht was op het formaliseren van de notie van *algoritme*, van wat überhaupt berekenbaar is. Als we de Church-Turing These accepteren, die beweert dat alles wat intuïtief berekenbaar is gegeven wordt door de vermogens van geïdealiseerde computers die we *Turing-machines* noemen, dan hebben we een exacte karakterisering van berekenbaarheid of *effectiviteit*. Dit geeft ons een praktische grens aan de soort patronen die we puur toeval willen kunnen ontzeggen: een toevallig object is dan, grofweg, een object waarbij we niet in staat zijn op effectieve wijze patronen te herkennen. Verschillende formaliseringen die op dit idee voortborduren hanteren bijvoorbeeld effectieve testen op patronen, effectieve comprimeertechnieken, en effectieve gokstrategiën om vast te stellen dat een object structuur heeft en dus niet toevallig is.

In deze scriptie geven we eerst een beknopt overzicht van de belangrijkste concepten, intuïties en resultaten van het vakgebied. We beschouwen in enig detail de twee belangrijkste formaliseringen voor deze scriptie, Martin-Löf's testconcept en Kolmogorov-complexiteit, en zien dat deze dezelfde klasse van toevallige reeksen bepalen. Vervolgens besteden we aandacht aan reduceerbaarheden die resulteren in niveaus of *graden* van toeval. Concreet kunnen we met de C- en K-, en de LK- en LR-reduceerbaarheden oneindige reeksen (ook te interpreteren als verzamelingen) vergelijken aan de hand van respectievelijk hun comprimeerbaarheid (hun mate van toeval) en hun kracht als orakel om patronen in andere verzamelingen te vinden (hun kracht om te "derandomiseren"). Deze reduceerbaarheden geven ook aanleiding tot laagste graden of noties van trivialiteit; en deze kenschetsen de verzamelingen die de meest triviale initiële-segmentcomplexiteit hebben en dus volstrekt niet toevallig zijn (de K-*triviale* verzamelingen), en de verzamelingen die waardeloos zijn als orakel in het vinden van regelmatigheden (*laag voor* K).

Vervolgens presenteren we origineel onderzoek binnen twee specifieke onderwerpen in algoritmische *randomness*. Beide hebben te maken met reeksen met initiële segmenten van erg lage complexiteit – dus reeksen die ver van toevallig zijn.

Ten eerste voeren we een techniek in de klassieke Turing-graden van berekenbaarheid over naar onze gradenstructuren van toeval. Dit is de constructie gegeven door de Splitsstelling van Sacks, die een gegeven berekenbaar opsombare (b.o.) maar onberekenbare verzameling splitst in twee b.o. onberekenbare verzamelingen die van een strikt lagere Turing-graad zijn en ook elkaar niet kunnen berekenen. Ondanks het feit dat b.o. verzamelingen ver van toevallig zijn, kunnen we met licht aangepaste constructies b.o. verzamelingen die niet van de laagst mogelijke toevalsgraad zijn splitsen in b.o. verzamelingen van strikt lagere maar nog steeds niet volstrekt triviale toevalsgraden (die bovendien onvergelijkbaar zijn). Dit geldt zowel voor de LR- als de C- en K-gradenstructuren. Deze resultaten bevestigen de *neerwaartse dichtheid* van deze b.o. gradenstructuren (tussen elke b.o. graad en de laagste graad zit strikt een derde b.o. graad), en leiden tot de vraag naar de algemene en opwaartse dichtheid van deze b.o. gradenstructuren, waar we kort bij stil staan.

Maar het hoofdonderwerp van de scriptie is een open probleem van Downey, Miller en Yu. Voor elke constante zijn er maar een zeer beperkt aantal verzamelingen K-triviaal via deze specifieke constante, en zij vroegen naar de aritmetische complexiteit van de functie die voor elke gegeven constante het precieze aantal zulke verzamelingen retourneert. Een geschikte manier om de K-triviale verzamelingen via constante b te representeren is als de oneindige paden van een bepaalde binaire boom, die we de *K-trivialiteitsboom* via b zullen noemen. Dan geeft de functie in kwestie voor invoer b dus het aantal paden van de K-trivialiteitsboom via b.

Als een eerste aanzet tot de oplossing van ons probleem, bestuderen we voor algemene klassen van bomen met maar eindig veel paden de complexiteit van het berekenen van het exacte aantal paden. De conclusie van dit onderzoek is dat voor een familie van zulke bomen van zekere Turing-graad, een orakel dat in de tweede sprong van deze graad ligt krachtig genoeg is om ons in staat te stellen van elke boom in de familie het aantal paden uit te rekenen. In het bijzonder geldt dat we het aantal paden van een boom die *laag*$_2$ is al kunnen bepalen met behulp van de tweede sprong $\emptyset''$.

Tegelijkertijd zoeken we naar manieren om de complexiteit van families van bomen te verlagen zonder aan de paden van de bomen te komen. Hierbij komen we uiteindelijk tot het resultaat dat we onze familie van K-trivialiteitsbomen kunnen reduceren naar een berekenbaar opsombare familie van bomen met dezelfde paden, bomen die bovendien *zelf* K-triviaal zijn. Een relevante eigenschap van K-triviale verzamelingen is hier dat deze laag$_2$ zijn, en dat de bijbehorende indices uniform berekenbaar zijn met orakel $\emptyset''$. Nu volgt met de uitkomst hierboven dat we onze functie dus kunnen berekenen middels $\emptyset''$, waarmee we de aritmetische complexiteit op Δ^0_3 bepaald hebben. Dit beantwoordt de open vraag. Tot slot komen we via een vergelijkbare strategie tot een antwoord op de analoge vraag voor de functie die het exacte aantal verzamelingen berekent die laag voor K via een gegeven constante zijn.

Bibliography

[Ars81] M. M. Arslanov. Some generalizations of a fixed-point theorem. *Izv. Vyssh. Uchebn. Zaved. Mat.*, 228(5):9–16, 1981.

[Bar68] Ja. M. Barzdins. Complexity of programs which recognize whether natural numbers not exceeding n belong to a recursively enumerable set. *Dokl. Akad. Nauk SSSR*, 182:1249–1252, 1968.

[BB10] George Barmpalias and Martijn Baartse. On the gap between trivial and nontrivial initial segment prefix-free complexity. Preprint., 2010.

[BLS08a] George Barmpalias, Andrew E. M. Lewis, and Mariya Soskova. Randomness, lowness and degrees. *J. Symbolic Logic*, 73(2):559–577, 2008.

[BLS08b] George Barmpalias, Andrew E. M. Lewis, and Frank Stephan. Π^0_1 classes, LR degrees and Turing degrees. *Ann. Pure Appl. Logic*, 156(1):21–38, 2008.

[BS10] George Barmpalias and Tom Sterkenburg. On the number of infinite sequences with trivial initial segment complexity. Preprint., 2010.

[BV10] George Barmpalias and Charlotte Vlek. Kolmogorov complexity of initial segments of sequences and arithmetical definability. Preprint., 2010.

[Cha75] Gregory J. Chaitin. A theory of program size formally identical to information theory. *J. Assoc. Comput. Mach.*, 22:329–340, 1975.

[Cha76] Gregory J. Chaitin. Information-theoretic characterizations of recursive infinite strings. *Theoret. Comput. Sci.*, 2(1):45–48, 1976.

[Chu40] Alonzo Church. On the concept of a random sequence. *Bull. Amer. Math. Soc.*, 46:130–135, 1940.

[Coo04] S. Barry Cooper. *Computability theory.* Chapman & Hall/CRC, Boca Raton, FL, 2004.

[DH10] Rod Downey and Denis Hirshfeldt. *Algorithmic Randomness and Complexity.* Springer-Verlag, to appear, 2010.

[DHL04] Rod G. Downey, Denis R. Hirschfeldt, and Geoff LaForte. Randomness and reducibility. *J. Comput. System Sci.*, 68(1):96–114, 2004.

[DHNS03] 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, Singapore, 2003. Singapore Univ. Press.

[Ers68] Yuri L. Ershov. A certain hierarchy of sets. *Algebra i Logika*, 7(1):47–74, 1968.

[HNS07] Denis R. Hirschfeldt, André Nies, and Frank Stephan. Using random sets as oracles. *J. Lond. Math. Soc. (2)*, 75(3):610–622, 2007.

[JS72] Carl G. Jockusch, Jr. and Robert I. Soare. Π^0_1 classes and degrees of theories. *Trans. Amer. Math. Soc.*, 173:33–56, 1972.

[KH07] Bjørn Kjos-Hanssen. Low for random reals and positive-measure domination. *Proc. Amer. Math. Soc.*, 135(11):3703–3709 (electronic), 2007.

[KHMS06] Bjørn Kjos-Hanssen, Wolfgang Merkle, and Frank Stephan. Kolmogorov complexity and the recursion theorem. In *STACS 2006*, volume 3884 of *Lecture Notes in Comput. Sci.*, pages 149–161. Springer, Berlin, 2006.

[Kol65] A. N. Kolmogorov. Three approaches to the definition of the concept "quantity of information". *Problemy Peredači Informacii*, 1(vyp. 1):3–11, 1965.

[Kra49] L.G. Kraft. *A device for quantizing, grouping and coding amplitude and modulated pulses.* MSc. Thesis, MIT, 1949.

[KT99] Antonín Kučera and Sebastiaan A. Terwijn. Lowness for the class of random sets. *J. Symbolic Logic*, 64(4):1396–1402, 1999.

[Kum96] Martin Kummer. On the complexity of random strings (extended abstract). In *STACS 96 (Grenoble, 1996)*, volume 1046 of *Lecture Notes in Comput. Sci.*, pages 25–36. Springer, Berlin, 1996.

[Lev73] L. A. Levin. The concept of a random sequence. *Dokl. Akad. Nauk SSSR*, 212:548–550, 1973.

[Lev76] L. A. Levin. The various measures of the complexity of finite objects (an axiomatic description). *Dokl. Akad. Nauk SSSR*, 227(4):804–807, 1976.

[LV08] Ming Li and Paul Vitányi. *An introduction to Kolmogorov complexity and its applications*. Texts in Computer Science. Springer, New York, third edition, 2008.

[ML66] Per Martin-Löf. The definition of random sequences. *Information and Control*, 9:602–619, 1966.

[MN06] Joseph S. Miller and André Nies. Randomness and computability: open questions. *Bull. Symbolic Logic*, 12(3):390–410, 2006.

[MP02] Andrej A. Muchnik and Semen Ye. Positselsky. Kolmogorov entropy in the context of computability theory. *Theoret. Comput. Sci.*, 271(1-2):15–35, 2002. Kolmogorov complexity.

[Nie05] André Nies. Lowness properties and randomness. *Adv. Math.*, 197(1):274–305, 2005.

[Nie06] André Nies. Reals which compute little. In *Logic Colloquium '02*, volume 27 of *Lect. Notes Log.*, pages 261–275. Assoc. Symbol. Logic, La Jolla, CA, 2006.

[Nie09] André Nies. *Computability and randomness*, volume 51 of *Oxford Logic Guides*. Oxford University Press, Oxford, 2009.

[Sac63] Gerald E. Sacks. On the degrees less than $0'$. *Ann. of Math. (2)*, 77:211–231, 1963.

[Sac64] Gerald E. Sacks. The recursively enumerable degrees are dense. *Ann. of Math. (2)*, 80:300–312, 1964.

[Sch73] C.P. Schnorr. Process complexity and effective random tests. *J. Comput. System Sci.*, 7:376–388, 1973. Fourth Annual ACM Symposium on the Theory of Computing (Denver, Colo., 1972).

[Soa96] Robert I. Soare. Computability and recursion. *Bull. Symbolic Logic*, 2(3):284–321, 1996.

[Soa10] Robert I. Soare. *Computability Theory and Applications:The Art of Classical Computability*. Springer-Verlag, 2010. To appear.

[Sol64] R. J. Solomonoff. A formal theory of inductive inference. I. *Information and Control*, 7:1–22, 1964.

[Sol75] R. Solovay. Handwritten manuscript related to Chaitin's work. IBM Thomas J. Watson Research Center, Yorktown Heights, NY, 215 pages, 1975.

[SS90] Richard A. Shore and Theodore A. Slaman. Working below a low_2 recursively enumerably degree. *Arch. Math. Logic*, 29(3):201–211, 1990.

[Vil39] J. Ville. *Étude critique de la notion de collectif*, volume 3 of *Monographies des Probabilités*. Gauthier-Villars, Paris, 1939.

[vL87] M. van Lambalgen. *Random sequences*. PhD Dissertation, Universiteit van Amsterdam, The Netherlands, 1987.

[vM19] R. von Mises. Grundlagen der Wahrscheinlichkeitsrechnung. *Mathematische Zeitschrift*, 5:52–99, 1919.

[Zam90] D. Zambella. On sequences with simple initial segments. Technical report ML 1990-05, ILLC, Universiteit van Amsterdam, 1990.

[ZL70] A. K. Zvonkin and L. A. Levin. The complexity of finite objects and the basing of the concepts of information and randomness on the theory of algorithms. *Uspehi Mat. Nauk*, 25(6(156)):85–127, 1970.

Index