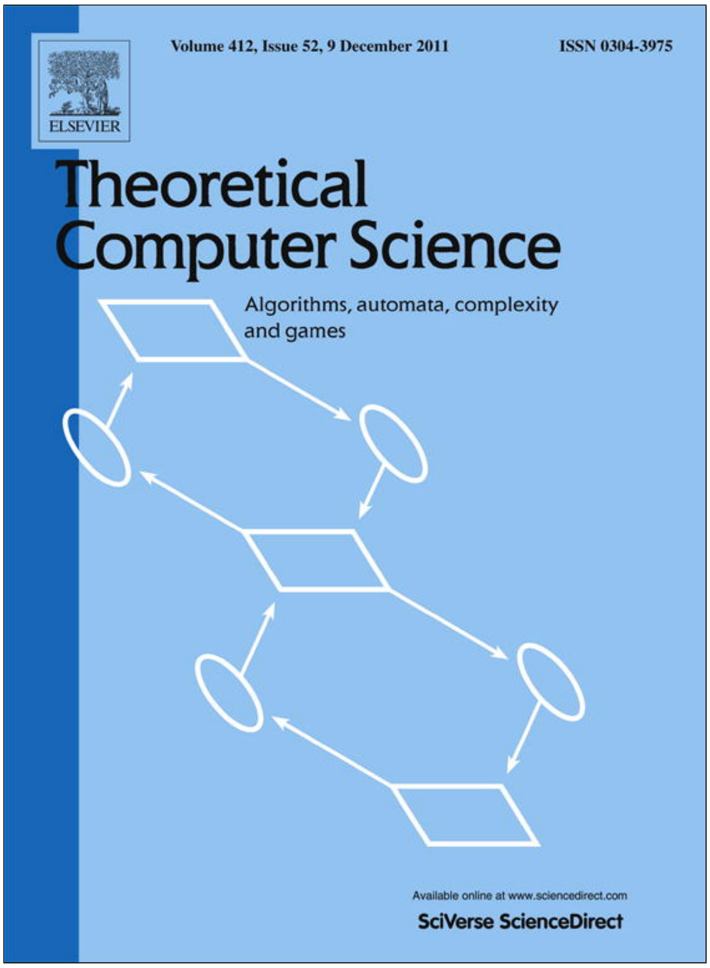
Volume 412, Issue 52, 9 December 2011
ISSN 0304-3975
ELSEVIER
Theoretical Computer Science
Algorithms, automata, complexity and games
Available online at www.sciencedirect.com
SciVerse ScienceDirect

Theoretical Computer Science 412 (2011) 7133–7146

Contents lists available at SciVerse ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

On the number of infinite sequences with trivial initial segment complexity

George Barmpalias [a,*], T.F. Sterkenburg [b]

[a] State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100190, P.O. Box 8718, China

[b] Institute for Logic, Language and Computation, Universiteit van Amsterdam, P.O. Box 94242, 1090 GE Amsterdam, The Netherlands

ARTICLE INFO

Article history:
Received 25 March 2011
Received in revised form 16 September 2011
Accepted 20 September 2011
Communicated by F. Cucker

Keywords:
Kolmogorov complexity
K-trivial sets
Arithmetical complexity
Trees

ABSTRACT

The sequences which have trivial prefix-free initial segment complexity are known as K-trivial sets, and form a cumulative hierarchy of length ω. We show that the problem of finding the number of K-trivial sets in the various levels of the hierarchy is Δ^0_3. This answers a question of Downey/Miller/Yu (see Downey (2010) [7, Section 10.1.4]) which also appears in Nies (2009) [17, Problem 5.2.16].

We also show the same for the hierarchy of the low for K sequences, which are the ones that (when used as oracles) do not give a shorter initial segment complexity compared to the computable oracles. In both cases the classification Δ^0_3 is sharp.

1. Introduction

Kolmogorov complexity is a standard tool for measuring the initial segment complexity of an infinite binary sequence. Identifying subsets of $\mathbb{N}$ with their characteristic sequence, we say that the initial segment (Kolmogorov) complexity of $A \subseteq \mathbb{N}$ is *trivial* if it is bounded by the Kolmogorov complexity of a computable set (modulo a constant). This concept was introduced by Chaitin (see [5,19]). For the case of plain Kolmogorov complexity Chaitin [5] showed that any set with trivial initial segment complexity must be computable. In the case of the prefix-free complexity K, Solovay [19] showed that there are non-computable sets with trivial initial segment complexity. In the last decade these so-called K-trivial sets, the sets A with $\forall n\, K(A\restriction_n) \leq K(n) + c$ for some $c \in \mathbb{N}$, have been the subject of intense research in algorithmic randomness. They are known to form a very interesting Σ^0_3 class $\mathcal{K}$ which is an ideal in the Turing degrees, see [17, Chapter 5].

The members of $\mathcal{K}$ are stratified in a cumulative hierarchy where level e consists of the sets A such that $\forall n\, K(A\restriction_n) \leq K(n) + e$. In this case we say that A is K-trivial with constant e, or even that e is a K-triviality constant for A. Chaitin [5] also showed that for each e there are finitely many K-trivial sets with constant e. A question of Downey/Miller/Yu (see [7, Section 10.1.4] and [17, Problem 5.2.16]) asked about the complexity of the following problem.

$$\text{Given } e \in \mathbb{N}, \text{ find the number of } K\text{-trivial sets with constant } e. \tag{1.1}$$

Let $\mathcal{K}_e$ denote the class of K-trivial sets with constant e. The following question refers to the complexity of the function $e \to |\mathcal{K}_e|$.

Question 1.1 (*Section 10.1.4 in [7] and Problem 5.2.16 in [17]*)**.** What is the arithmetical complexity of (1.1)? In particular, is it Δ^0_3?

* Corresponding author.
E-mail addresses: barmpalias@gmail.com (G. Barmpalias), tsterken@science.uva.nl (T.F. Sterkenburg).
URL: http://www.barmpalias.net (G. Barmpalias).

0304-3975/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2011.09.020

In this paper we give a positive answer to the above question, thus showing that the function $e \to |\mathcal{K}_e|$ lies exactly at level Δ^0_3 of the arithmetical hierarchy. In particular, although the function $e \to |\mathcal{K}_e|$ depends on the choice of the underlying universal machine, its arithmetical complexity does not. Moreover, the solution of this problem gives a general methodology for answering the same question for related Σ^0_3 classes, like the low for K sets. A set A is low for K if it does not compress strings more efficiently than a computable oracle. More precisely, if the prefix-free complexity K^A relative to A is not smaller than their unrelativized prefix-free complexity K, modulo a constant. In symbols, if $K(\tau) \leq K^A(\tau) + c$ for some constant c and all strings τ. This Σ^0_3 class can also be seen as the union of a cumulative hierarchy whose eth level consists of the sets A with $K(\tau) \leq K^A(\tau) + e$ for all strings σ. As in the case of $\mathcal{K}$ we say that A is low for K with constant e if it lies in the eth level of this hierarchy.

Hirschfeldt and Nies showed in [16] that the class of low for K sets coincides with $\mathcal{K}$. However this coincidence is not effective, in the sense that there is no algorithm which outputs a level in the low for K hierarchy where a set lives, given a level of it in the K-triviality hierarchy. Hence determining the complexity of the functions giving the cardinality of the levels of the two hierarchies constitutes two separate problems.

The positive answer to Question 1.1 can be used in order to refine the work of Csima and Montalbán in [6]. In this paper the authors construct an unbounded nondecreasing function f such that for all reals X and all constants c

if $\forall n\ (K(X\restriction_n) \leq K(n) + f(n) + c)$ then X is K-trivial.

An analysis of the construction shows that f is Δ^0_4. The complexity of f can be reduced to Δ^0_3 using the answer to Question 1.1 as it is shown in [2]. Moreover this is optimal in the sense that if f is Δ^0_2 then it does not have the above property. This was shown in [3] and in [2] it was extended, showing that if f is Δ^0_2 unbounded and nondecreasing then there is a large and rich collection of reals X which satisfy $\forall n\ (K(X\restriction_n) \leq K(n) + f(n) + c)$ for some constant c.

The purpose of constructing f in [6] was the construction of minimal pairs with respect to $\leq_K$, where $A \leq_K B$ if $\exists c\ \forall n\ (K(A\restriction_n) \leq K(B\restriction_n) + c)$. According to the above discussion the complexity of minimal pairs is reduced to Δ^0_3. However different methods in [3] give a simpler construction of a Σ^0_2 set that forms a minimal pair with every Σ^0_1 with respect to $\leq_K$.

1.1. Overview

We start by discussing the problem of finding the number of infinite paths through a given tree. Throughout this paper we are only interested in trees that have finitely many infinite paths. An analysis of this general problem is given in Section 2 and provides the framework for answering Question 1.1. This is because the class $\mathcal{K}_e$ is canonically given as the infinite paths through a certain tree (by [5]). In Section 2.1, given a sequence $\mathcal{C} = (T_e)$ of trees we relate the complexity of the function $G_{\mathcal{C}} : e \to |[T_e]|$ (where $[T]$ denotes the class of infinite paths through T) to the complexity of $\mathcal{C}$. The main observation here is that in general, the degree of $G_{\mathcal{C}}$ is the double jump of the degree of $\mathcal{C}$ (see below for more precise formulations of this statement). Based on this, in Section 2.2 we characterize the jump classes high$_2$ and low$_2$ in terms of trees. The latter characterization will be used in the solution of Question 1.1 in Section 3. In Section 2.3 we study formal versions of the following question.

$$\text{Given a tree } T \text{ is there a `simpler' tree } Q \text{ with the same infinite paths?} \tag{1.2}$$

Here 'simple' is formalized via some standard complexity measures like definability hierarchies, degrees of unsolvability and Kolmogorov complexity. For example, we show that if T is Δ^0_2 then there is an infinitely often K-trivial tree Q with the same paths as T. To answer Question 1.1 in Section 3 we will obtain a K-trivial tree Q with the same paths as T, under the stronger assumption that all paths of T are K-trivial. In particular, we show that the trees associated with the K-trivial sets can be effectively replaced with much simpler trees with the same infinite paths. We also observe that using $\emptyset''$ and a K-triviality constant of a set, we can obtain a lowness index of it. We conclude that Question 1.1 admits a positive answer. Finally, in Section 4 we use our methodology to answer the analogue of Question 1.1 for the related class of the low for K sets.

1.2. Basic concepts

Throughout this paper the notion of a parametrized family of sets and its arithmetical complexity plays a central role.

Definition 1.1. A class $\mathcal{C} \subseteq 2^\omega$ is a Δ^0_1 family of sets if it can be written in the form $\{C_e \mid e \in \mathbb{N}\}$ where $C_e = \{n \mid \psi(e, n)\}$ and ψ is a Δ^0_1 property (i.e. a property that can be expressed in arithmetic with equivalent Σ^0_1 and Π^0_1 formulas). Similar definitions apply for each level of the arithmetical hierarchy.[1] The degree of a parameterized family $(C_e)_{e\in\omega}$ of subsets of $\mathbb{N}$ is the degree of $\oplus_{e\in\omega} C_e$.

[1] We use the word 'family' to refer to these countable parametrized subsets of 2^ω, as opposed to the *arithmetical classes*, like the Σ^0_1 or Π^0_2 classes, which could well be uncountable. The difference is that in the case of Definition 1.1 the arithmetical formula underlying the definition of the class has only first order variables, while in the general case second order variables can be used.

We often write $\mathcal{C} \equiv_T X$ to denote that the family $\mathcal{C}$ (with an underlying parameterization) has the same degree as X. Notice that a Δ^0_1 family is nothing more than a uniformly computable (i.e. computable from a single algorithm, on different indices) sequence of sets. Similarly, a Σ^0_1 family is nothing more than a uniformly c.e. sequence of sets. According to Definition 1.1, each arithmetical family has an underlying parameterization. We often identify the various arithmetical families with the sequences of sets that correspond to their parametrizations.

We recall the definition of a tree in the Cantor space.

Definition 1.2. A tree is a downward closed (with respect to the prefix relation) set of binary strings. The set of infinite paths through a tree T is denoted by $[T]$. Level n of a tree consists of the strings of length n that belong to the tree. The *width* of a tree is the supremum of the cardinality of its levels. A tree is said to have *bounded width* if its width is finite. A split on a tree T is a pair of incomparable (i.e. not equal and neither one is an extension of the other) strings in T. An antichain in T is a set of pairwise incomparable strings in T.

The following is a basic fact about the notions of Definition 1.2.

A tree in the Cantor space has finitely many splits iff there is a constant bound on the size of its antichains, iff it does not contain an infinite antichain. (1.3)

In the following we will be considering trees of different complexities. For example, a Σ^0_1 (or computably enumerable) tree is one that is Σ^0_1, as a set of strings. Notice that these are exactly the partial computable trees. Each parametrized family of trees $\mathcal{C} = \{T_e\}$ is naturally associated with the function $G_{\mathcal{C}}(e) = |[T_e]|$ which gives the number of infinite paths through a tree in the family, given its index. We will study how various complexities of classes $\mathcal{C}$ of trees relate to the complexity of the corresponding $G_{\mathcal{C}}$ function.

Apart from different levels of the arithmetical hierarchy, we will also use trees on different (finite) levels of the Ershov hierarchy of n-c.e. sets, see [9]. This class, also denoted as Σ^{-1}_n, contains the sets that have a computable approximation according to which the membership status of each number changes at most n times. For example, 1-c.e. sets are c.e. sets. A uniform family of n-c.e. sets is one that can be presented as $\{\lim_s \varphi(e, s) \mid e \in \mathbb{N}\}$, where φ is a computable function and $|\{s \mid \varphi(e, s) \neq \varphi(e, s+1)\}| \leq n$ for each $e \in \mathbb{N}$. We will also refer to such a family as discretely Σ^{-1}_n.

In a number of proofs in this paper we use a standard tool from computability theory which is often called the *hat-trick*. It is merely a slight modification of the enumeration of the Turing functionals (Φ_e) that depends on the enumeration (B_s) of some c.e. set B. If $\hat{\Phi}_e$ denotes the modified functional, we let $\hat{\Phi}^B_e(n)[s] \downarrow = k$ if $\Phi^B(n)[s] \downarrow = k$ with use u and u is not larger than any number entering B at stage s. In particular, if $\hat{\Phi}^B(n)[s] \downarrow$ with use u and $B_{s+1} \restriction_u \neq B_s \restriction_u$ then $\hat{\Phi}^B(n)[s+1] \uparrow$. Clearly Φ^B_e is total if and only if $\hat{\Phi}^B_e$ is, in which case they are equal. However we have the extra effect that if $\hat{\Phi}^B_e(n) \uparrow$ then $\hat{\Phi}^B_e(n)[s] \uparrow$ for infinitely many stages s. Whenever we use the hat-trick we may omit the hat notation.

Finally, recall that if a set of strings does not contain any pair of comparable strings (i.e. one is an extension of the other) then it is called *prefix-free*. A prefix-free machine is one whose domain (i.e. the set of strings on which it halts) is a prefix-free set. The prefix-free complexity K of a string τ under a fixed universal prefix-free machine U is the length of the shortest string σ such that $U(\sigma) = \tau$. In the following we fix U to be the canonical universal prefix-free machine, i.e. $U(0^e 1\sigma) = M_e(\sigma)$ for all strings σ, where M_e is the e-th prefix-free machine in an effective enumeration of all prefix-free machines. Also we let $\mathtt{wgt}(M)$ be the 'weight' of a prefix-free machine M, namely the number $\sum_{M(\sigma)\downarrow} 2^{-|\sigma|}$. For more background on Kolmogorov complexity we refer to [7,17,15].

2. The number of paths through a tree

2.1. Arithmetical complexity of the number of paths through a tree

By compactness, the oracle $\emptyset'$ can determine if a given computable tree has at least one infinite path. Similarly, $\emptyset''$ can determine if a given c.e. tree has at least one infinite path. Moreover the following is very easy to show.

Proposition 2.1. *There exists a Δ^0_1 family of trees (with no infinite antichains) such that the degree of the problem of which trees in the family have infinite paths is $\emptyset'$. Also there exists a Σ^0_1 family of trees (with no infinite antichains) such that the degree of the problem of which trees in the family have infinite paths is $\emptyset''$.*

The following observations show that $\emptyset'$ is not sufficiently strong so as to determine whether a given computable tree has at least two infinite paths. In fact, if an oracle A can perform this task, it has to also compute $\emptyset''$. More precisely, this task is an algorithm which uses A as an oracle and if the input is a program which defines a computable tree T, the output is $|[T]|$. Notice that such an algorithm may be partial on inputs that are (codes of) programs that define a computable tree.

Proposition 2.2. *The oracle $\emptyset''$ can compute the number of infinite paths through a given computable tree with finitely many paths.*

Proof. Given a program e that computes a tree T, we can use $\emptyset''$ to find the largest $m \in \mathbb{N}$ such that

There exist m strings of the same length ℓ which are extendible at all levels $\geq \ell$ of the tree.

This is a Σ^0_2 question. If the tree has finitely many paths, this m must be found. Clearly it is the number of paths through the tree. □

The following proof uses the *hat-trick*, as described in Section 1.2.

Theorem 2.3. *There is a Δ^0_1 family $\mathcal{C}$ of trees of bounded width (at most 2), such that $G_{\mathcal{C}} \equiv_T \emptyset''$.*

Proof. We define a uniform sequence of computable trees T_e as follows. Given $e \in \mathbb{N}$ we check the state of $\Phi^{\emptyset'}_e(e)[s]$ (i.e. whether it converges or not) at the various stages s, and determine the T_e membership or not of the strings of length s. We use the *hat trick* to ensure that if $\Phi^{\emptyset'}_e(e) \uparrow$ then $\Phi^{\emptyset'}_e(e)[s] \uparrow$ for infinitely many stages s. If $\Phi^{\emptyset'}_e(e)[s] \uparrow$ then the only string of length s that is in the tree is 0^s. If $\Phi^{\emptyset'}_e(e)[s] \downarrow$ and $t \leq s$ is the least stage such that $\Phi^{\emptyset'}_e(e)[i] \downarrow$ for all $i \in [t, s]$ then the strings of length s that are in the tree are 0^s and $0^t * 1 * 0^{s-t-1}$. Clearly $e \in \emptyset''$ iff $|[T_e]| = 2$. The fact that $G_{\mathcal{C}} \leq_T \emptyset''$ follows from Proposition 2.2. □

By relativizing Theorem 2.3 to $\emptyset^{(j)}$, $j \in \mathbb{N}$ we get the following.

Corollary 2.4. *Let $n \in \mathbb{N}$. There is a Δ^0_n family $\mathcal{C}$ of trees of width at most 2, such that $G_{\mathcal{C}} \equiv_T \emptyset^{(n+1)}$.*

Since $\Sigma^0_1 \subseteq \Delta^0_2$, a relativization of Proposition 2.2 gives the following fact about Σ^0_1 families of trees (i.e. partial computable families).

Proposition 2.5. *The oracle $\emptyset'''$ can compute the number of infinite paths through a given Σ^0_1 tree with finitely many paths.*

If $(A[s])$ is an enumeration of a set A, we say that a number n that enters A at stage s is a *true enumeration* if $A[s] \upharpoonright_n = A \upharpoonright_n$. Let $\langle \sigma, \tau \rangle$ denote the number that codes the pair of binary strings σ, τ.

Theorem 2.6. *There is a Σ^0_1 family $\mathcal{C}$ of trees with at most 2 infinite paths such that $G_{\mathcal{C}} \equiv_T \emptyset'''$.*

Proof. Since $\emptyset''$ is c.e. in $\emptyset'$, let W be a c.e. operator such that $W^{\emptyset'} = \emptyset''$. By $W^{\emptyset'}[s]$ we mean the set consisting of the numbers in W with use $\emptyset'[s]$. By the *hat trick* (as described in Section 1.2, and the fact that there are infinitely many true enumerations into $\emptyset'$) we can assume that for all t there exists a stage $s > t$ such that $W^{\emptyset'}[s] \upharpoonright_t = \emptyset'' \upharpoonright_t$. Consider the computable function $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ such that $f(e, s)$ is s if $\Phi^{W^{\emptyset'}}_e(e)[s] \uparrow$ and $\langle \sigma, \tau \rangle$ if $\Phi^{W^{\emptyset'}}_e(e)[s] \downarrow$ with use $\tau \subseteq W^{\emptyset'}[s]$ such that the use of the oracle $\emptyset'[s]$ in the computations that give membership to the numbers in $\tau \subset W^{\emptyset'}[s]$ is $\sigma \subset \emptyset'[s]$. By the *hat trick* applied to the functionals Φ_e we have the following for each $e \in \mathbb{N}$.

$$e \in \emptyset''' \iff \liminf_s f(e, s) \text{ is finite.} \tag{2.1}$$

We enumerate the tree T_e as follows. By convention, when a string is enumerated into T_e all initial segments of it are also enumerated without explicitly mentioning it. All strings 0^n, $n \in \mathbb{N}$ are in T_e. At stage 0 all numbers are *inactive*. At stage s let $f(e, s) = n$ ask if n is *inactive*. If it is, enumerate into T_e the string $0^s * 1$, say that n is *active*. If not, let $t < s$ be the least stage such that n has been active during the stages in $[t, s]$. Enumerate into T_e the string $0^t * 10^{s-t}$. Also say that all $m > n$ are *inactive*.

Now it is clear that for each $e \in \mathbb{N}$ the tree T_e has either 1 or 2 infinite paths and by (2.1),

$$e \in \emptyset''' \iff |[T_e]| = 2.$$

Hence for the Σ^0_1 family $\mathcal{C}$ of trees T_e, $e \in \mathbb{N}$ we have $\emptyset''' \leq_T G_{\mathcal{C}}$. The fact that $G_{\mathcal{C}} \leq_T \emptyset'''$ follows from Proposition 2.5. □

The width of a tree T can be seen as a function $w_T(n)$ which gives the number of nodes at level n of the tree. We say that a function f bounds the width of a tree T if it dominates w_T. It is not hard to extend the argument of Theorem 2.6 in order to show the following. *Let f be a computable order. There is a Σ^0_1 family $\mathcal{C}$ of trees of width bounded by f and at most 2 infinite paths such that $G_{\mathcal{C}} \equiv_T \emptyset'''$.*

We note that some of the trees constructed in the proof of Theorem 2.6 do not have bounded width. The following observation contrasts Theorem 2.6 and Proposition 2.5, and shows that this is unavoidable.

Theorem 2.7. *The oracle $\emptyset''$ can compute the number of infinite paths through a given Σ^0_1 tree with bounded width.*

Proof. Let e be a program which enumerates a tree T. Using $\emptyset''$ we can find the largest number n such that there are infinitely many stages $s, t \in \mathbb{N}$ where there are n distinct strings on level t of $T[s]$. Moreover, let k be a number such that there is no level of T above k which has $n + 1$ distinct strings. Then there exists a c.e. sequence $\{(\sigma^j_0, \dots, \sigma^j_{n-1})\}_j$ of n-tuples of strings on T such that $|\sigma^j_u| = |\sigma^j_v|$ for $u, v < n$ and $k < |\sigma^j_0| < |\sigma^i_0|$ for $j < i$. The downward closure of these strings forms a computable sub-tree of T with bounded width. Moreover every path of T is a path of the new tree by the maximality of $\{(\sigma^j_0, \dots, \sigma^j_{n-1})\}_j$. By Proposition 2.2 using $\emptyset''$ we can find the number of paths through the new tree. This is also the number of paths through T. □

Table 1
The degree of the problem of computing the number of infinite paths of a given tree of certain complexity and with various properties.

	Δ^0_1	Σ^0_1	Σ^{-1}_2	Δ^0_2	Δ^0_n	Σ^0_n	Σ^{-1}_{n+1}
No infinite antichains	$\emptyset'$	$\emptyset''$	$\emptyset''$	$\emptyset''$	$\emptyset^{(n)}$	$\emptyset^{(n+1)}$	$\emptyset^{(2)}$
Bounded width	$\emptyset''$	$\emptyset''$	$\emptyset'''$	$\emptyset'''$	$\emptyset^{(n+1)}$	$\emptyset^{(n+1)}$	$\emptyset^{(3)}$
Finitely many infinite paths	$\emptyset''$	$\emptyset'''$	$\emptyset'''$	$\emptyset'''$	$\emptyset^{(n+1)}$	$\emptyset^{(n+2)}$	$\emptyset^{(3)}$

By Theorem 2.3 the bound $\emptyset''$ provided in Theorem 2.7 is optimal. The proof of Theorem 2.7 produced an oracle program which used $\emptyset''$ to calculate the number of paths through the given trees. It is instructive to write a program which works without an oracle, and such that $\emptyset''$ can extract from it the true number of paths through the given tree. More precisely, to define a computable function $f : \mathbb{N}\times\mathbb{N} \to \mathbb{N}$ such that if e is a description of a Σ^0_1 tree T of bounded width then $\liminf_s f(e, s)$ is finite and equals the number of infinite paths of T. We leave this as an exercise for the reader.

Theorem 2.8. *There is a Σ^{-1}_2 family $\mathcal{C}$ of trees of bounded width (at most 2), such that $G_{\mathcal{C}} \equiv_T \emptyset'''$.*

Proof. This is a modification of the proof of Theorem 2.6. The additional feature that the trees have bounded width is obtained by removing some strings that have already been enumerated in the tree T_e. However such a removal has to be done at most once for each string in order for our trees to be d.c.e. (i.e. in Σ^{-1}_2).

All strings 0^n, $n \in \mathbb{N}$ are in T_e (and are never removed). At stage 0 all numbers are *inactive*. At stage s let $f(e, s) = n$ ask if n is *inactive*. If it is, enumerate into T_e the string $0^s * 1$, say that n is *active*. If not, let $t < s$ be the least stage such that n has been active during the stages in $[t, s]$. Clearly, at stage t the number n became active. Remove from T_e all strings $0^i * 1$ (and their extensions) for $i > t$. Also, enumerate into T_e the string $0^t * 10^{s-t}$ and say that all $m > n$ are *inactive*.

It is not hard to see that each string is removed from T_e at most once. Therefore the family of the trees T_e is Σ^{-1}_2. Moreover, due to these removals at each stage there are at most two strings at each level of T_e. Hence T_e has bounded width (at most 2). Now it is clear that for each $e \in \mathbb{N}$ the tree T_e has either 1 or 2 infinite paths and by (2.1),

$$e \in \emptyset''' \iff |[T_e]| = 2.$$

Hence for the Σ^{-1}_2 family $\mathcal{C}$ of trees T_e, $e \in \mathbb{N}$ we have $\emptyset''' \le_T G_{\mathcal{C}}$. The fact that $G_{\mathcal{C}} \le_T \emptyset'''$ follows from the relativization of Proposition 2.2 to $\emptyset'$ and the fact that each Σ^{-1}_2 family of sets is also a Δ^0_2 family. □

Theorem 2.9. *The oracle $\emptyset'$ can compute the number of infinite paths through a given computable tree with no infinite antichains.*

Proof. Let T be the given computable tree. By (1.3) we can use $\emptyset'$ in order to find some $k \in \mathbb{N}$ such that there is no split in T at any level $\ge k$. Then the number of extendible strings of length k is the number of infinite paths through T. □

By Proposition 2.1 the bound $\emptyset'$ in Theorem 2.9 is optimal. Similarly, the oracle $\emptyset''$ can compute the number of infinite paths through a given c.e. tree with no infinite antichains. Again, this bound is optimal, by Proposition 2.1. The case for Σ^{-1}_2 trees with no infinite antichains is similar to the case for Σ^0_1 trees with the same properties, giving $\emptyset''$ as the optimal bound. The results in this section, along with their straightforward relativizations to all levels of arithmetical complexity and all levels of the Ershov hierarchy are displayed in Table 1.

Finally we note that, although in this section we focused on the Turing degrees of the functions $G_{\mathcal{C}}$ for various families $\mathcal{C}$, a bit more can be said about the complexity of these functions. For example, when $\mathcal{C}$ is a Δ^0_1 family, the function $G_{\mathcal{C}}$ can be $\emptyset'$-computably approximated from below. Moreover every such function can be represented as the $G_{\mathcal{C}}$ function for some Δ^0_1 family $\mathcal{C}$. Similar refinements hold for other levels of the arithmetical complexity.

2.2. Paths through trees and the jump hierarchy

By relativizing Proposition 2.2 we have that given $A \in 2^\omega$ and any uniformly A-computable family $\mathcal{C}$ of trees (i.e. a $\Delta^0_1(A)$ family of trees) with finitely many infinite paths, $G_{\mathcal{C}} \le_T A''$. Moreover, by relativizing Theorem 2.3 we have that there is a uniformly A-computable family $\mathcal{C}$ of trees with finitely many infinite paths such that $A'' \le_T G_{\mathcal{C}}$. Hence we have the following immediate corollary.

Corollary 2.10 (*High$_2$ and Low$_2$ in Terms of Trees*)**.** *A degree $\mathbf{a} \le \mathbf{0}'$ is high$_2$ iff there is an $\mathbf{a}$-computable family $\mathcal{C}$ of trees (with finitely many paths) such that the degree of $G_{\mathcal{C}}$ is $\mathbf{0}'''$. Moreover it is low$_2$ iff for any $\mathbf{a}$-computable family $\mathcal{C}$ of trees (with finitely many paths) the degree of $G_{\mathcal{C}}$ is $\le \mathbf{0}''$.*

Notice that the same argument shows the following.

Given a tree T with finitely many paths and a low$_2$-ness index of it (i.e. some index of a reduction of T'' to $\emptyset''$) we can effectively produce an index of an $\emptyset''$-computation of the number of infinite paths of T. (2.2)

This fact will be used in Section 3.2 for answering Question 1.1. An analogous result holds for c.e. degrees.

Theorem 2.11 (*Computably Enumerable High$_2$ and Low$_2$ in Terms of Trees*). *A c.e. degree* **a** *is high$_2$ iff there is an* **a***-computable Σ^0_1 family $\mathcal{C}$ of trees (with finitely many paths) such that the degree of $G_{\mathcal{C}}$ is $\mathbf{0}'''$. Moreover* **a** *is low$_2$ iff for any* **a***-computable Σ^0_1 family $\mathcal{C}$ of trees (with finitely many paths) $G_{\mathcal{C}}$ is computable from $\mathbf{0}''$.*

Proof. The 'if' direction of the statement about high$_2$ and the 'only if' statement about low$_2$ follow from Corollary 2.10. For the other two statements it suffices to show the following:

$$\text{If } A \text{ is c.e. and } f \leq_T A'' \text{ there exists an } A\text{-computable } \Sigma^0_1 \text{ family } \mathcal{C} \text{ of trees (with finitely many paths) such that } f \leq_T G_{\mathcal{C}}. \tag{2.3}$$

Indeed, for the 'only if' direction of the statement about high$_2$ we have that if A is high$_2$ then $\emptyset'''$ is computable by a $\Delta^0_3(A)$ function f. But such a function f can be written as a computable image of the limit infimum of the values of an A-computable function Ψ^A of two variables (e.g. see [20, p. 207]). In particular, $\emptyset'''(n) = (\liminf_s \Psi^A(n, s))_0$ for all $n \in \mathbb{N}$, where $x \to (x)_0$ denotes the first inverse of the standard pairing function (giving the first coordinate of the pair). Therefore (2.3) shows how to obtain the required family $\mathcal{C}$ of trees for the 'only if' direction of the statement about high$_2$ in Theorem 2.11. On the other hand, the 'if' direction for the statement about low$_2$ follows directly from (2.3) by letting $f = A''$.

By the representation of a $\Delta^0_3(A)$ function that we mentioned above (as the computable image of the limits infimum of the values of an A-computable function) to prove (2.3) it suffices to show the following.

$$\text{Given an } A\text{-computable function } \Psi^A \text{ of two variables such that } \liminf_s \Psi^A(n, s) \text{ is finite for all } n \in \mathbb{N}, \text{ there exists a uniformly } A\text{-computable family } \mathcal{C} \text{ of trees } T_n, n \in \mathbb{N} \text{ such that } |[T_n]| = \liminf_s \Psi^A(n, s) + 1 \text{ for each } n \in \mathbb{N}. \tag{2.4}$$

Fix such a function Ψ^A (given as a Turing functional Ψ with oracle A) and $n \in \mathbb{N}$.

Construction of of T_n. We say that $0^i * 1 * 0^j * 1$ is *active* at stage s if $i, j < s$ and

- $\Psi^A(n, i)[s] > j$
- if $i < z < s$ then either $\Psi^A(n, z)[s] \uparrow$ or $\Psi^A(n, z)[s] > j$
- $\Psi^A(n, i - 1)[s] \leq j$.

At stage s and for each $i, j < s$ such that $0^i * 1 * 0^j * 1$ is active enumerate into T_n the string $0^i * 1 * 0^j * 1 * 0^s$ and all of its prefixes. Moreover put 0^s into T_n.

Verification. First we verify that $|[T_n]| = \liminf_s \Psi^A(n, s) + 1$. Let $\liminf_s \Psi^A(n, s) = m$. If $j < m$ and i is the least number such that $\Psi^A(n, k) > j$ for all $k \geq i$ then (by the hat trick, described in Section 1.2, and the use of true stages) the string $0^i * 1 * 0^j * 1$ will be active infinitely often. Therefore $|[T_n]| \geq m + 1$. If $0^k * 1 * 0^t * 1$ is any other string, according to the definition of *active* (and since Ψ^A is total) it will only be active finitely often. Hence $|[T_n]| = m + 1$.

It remains to show that the sequence $(T_n)_{n\in\mathbb{N}}$ is (uniformly) computable in A. Given a string σ, we first check if $\sigma \subset 0^\omega$ (in which case $\sigma \in T_n$). If not, $0^i * 1 \subseteq \sigma$ for some $i \in \mathbb{N}$. Using A compute $\Psi^A(n, i - 1)$, $\Psi^A(n, i)$ and find the first stage s_i where they converge (with correct A use). If none of the strings $0^i * 1 * 0^j * 1$ with $\Psi^A(n, i - 1) \leq j < \Psi^A(n, i)$ are prefixes of σ, then σ is in T_n iff it is in $T_n[s_i]$. Otherwise, suppose that $0^i * 1 * 0^j * 1 \subseteq \sigma$. If σ is not of the form $0^i * 1 * 0^j * 1 * 0^k$ for some $k \in \mathbb{N}$, then it is not in T_n. If $\sigma = 0^i * 1 * 0^j * 1 * 0^k$ (for some $k \in \mathbb{N}$) search for a stage $s > k$ such that either $\Psi^A(n, t) < j$ for some $t > i$ (and the computation converges with correct A use) or $0^i * 1 * 0^j * 1$ is active at s. Such a stage exists, by the definition of *active* strings, the fact that $\Psi^A(n, i - 1) \leq j < \Psi^A(n, i)$ and the totality of Ψ^A. If $0^i * 1 * 0^j * 1$ is active at s, clearly σ is in T_n. Otherwise σ is in T_n iff it has been enumerated by stage s. This concludes the proof of the reduction $T_n \leq_T A$ (uniformly in n) and the proof of (2.4). □

We note that via standard coding procedures one can replace '**a**-computable family' in the statements of Corollary 2.10 and Theorem 2.11 with 'family of degree **a**'.

2.3. Representing closed sets by trees of certain arithmetical complexity

In this section we discuss the general question (1.2) from the introduction. In other words, we are interested in replacing a given tree T with a simpler tree Q such that $[T] = [Q]$, i.e. the two trees have the same infinite paths. The complexity of Q will ultimately depend on that of T. In the following we make this relation precise by considering various measures of complexity. We note that the main idea behind the solution of Question 1.1 is exactly this: to replace certain trees that generate the K-trivial sets with simpler ones (see Section 3). The following fact is the simplest statement of this kind.

Theorem 2.12 (*Folklore*). *Let $n > 0$. If T is a Π^0_n tree, there is a Δ^0_n tree Q such that $[T] = [Q]$.*

The case $n = 1$ of Theorem 2.12 is a well known fact that allows the representation of Π^0_1 classes with computable trees. Namely, the fact that for each Π^0_1 tree there is a computable tree with the same infinite paths. The other cases follow by straightforward relativization, given that Π^0_n is $\Pi^0_1(\emptyset^{(n-1)})$ and Δ^0_n is $\Delta^0_1(\emptyset^{(n-1)})$.

Theorem 2.12 fails for Σ^0_n trees.

Theorem 2.13. *Let $n > 0$. There is a Σ^0_n tree T which only has computable paths and $[T] \neq [Q]$ for all Π^0_n trees Q.*

Proof. By Theorem 2.12 it suffices to prove this for Q restricted to Δ^0_n trees. We prove the case $n = 1$. The other cases follow by relativization, since Δ^0_n is $\Delta^0_1(\emptyset^{(n-1)})$ and Σ^0_n is $\Sigma^0_1(\emptyset^{(n-1)})$.

Consider an effective list (Φ_e) of all partial computable functions from strings to $\{0, 1\}$. We define a c.e. tree T with only computable infinite paths, such that for each $e \in \mathbb{N}$ either Φ_e is partial or the set of strings that it computes is not a tree, or the set of paths through the tree it computes is different from the set of paths through T. First, we put all 0^n, $n \in \mathbb{N}$ in T. For each $e \in \mathbb{N}$ we do the following. Wait until a stage s such that for some $\ell < s$, $\ell > e + 1$ and all extensions τ of 0^e1 of length ℓ we have $\Phi_e(\tau)[s] = 0$. In that case put all 0^e10^n, $n \in \mathbb{N}$ in T.

Now suppose that Φ_e is total and computes a downward closed set of strings Q_e. If there is an infinite path through Q_e extending 0^e1, clearly there is no such path through T. Otherwise $0^e10^\omega \in [T]$. In any case, $[T] \neq [Q]$. □

The following is another basic example of reducing the arithmetical complexity of a tree while leaving the class of infinite paths invariant.

Theorem 2.14. *Let $n > 1$. If T is a Δ^0_n tree, there exists a Σ^0_{n-1} tree Q such that $[T] = [Q]$.*

Proof. We give a proof for the special case $n = 2$. The other cases follow by straightforward relativization of this argument, since Δ^0_n is $\Delta^0_2(\emptyset^{(n-2)})$ and Σ^0_{n-1} is $\Sigma^0_1(\emptyset^{(n-2)})$.

Let T be a Δ^0_2 tree (as a downward closed set of strings) and let $T[s]$ be a computable approximation to it. Since T is a tree, we may assume that for each $s \in \mathbb{N}$, if $\sigma \in T[s]$ then all $\tau \subset \sigma$ are in $T[s]$. We define a Σ^0_1 tree Q as follows. For each string σ,

$$\sigma \in Q \iff \exists s \geq |\sigma|, \quad \sigma \in T[s].$$

Clearly $T \subseteq Q$, so $[T] \subseteq [Q]$. For the converse, suppose that $X \notin [T]$. Then there exists some $\rho \subset X$ such that $\rho \notin T$. Let s_0 be a stage such that $\rho \notin T[s]$ for all $s \geq s_0$. Then level s_0 of Q does not contain any extension of ρ, since this can only happen when $\rho \in T[s]$ for some $s \geq s_0$. Hence X is not in $[Q]$. □

The following result shows that we do not have much control over the Turing degree of the Σ^0_{n-1} tree of Theorem 2.14.

Lemma 2.15. *Let **a** be any degree. There exists a tree T of degree **a** which contains only computable paths and such that for every tree Q, if $[T] = [Q]$ then Q computes **a**.*

Proof. Let f be a 0–1 function of degree **a**. For each $e \in \mathbb{N}$, if $f(e) = 0$ put all $0^{2e}10^n$, $n \in \mathbb{N}$ in T. If $f(e) = 1$ put all $0^{2e+1}10^n$, $n \in \mathbb{N}$ in T. Clearly $T \equiv_T f$. Suppose that Q is a tree with $[T] = [Q]$. To compute $f(e)$ from Q search for a level ℓ of Q such that either there are no extensions of $0^{2e}1$ at level ℓ or there are no extensions of $0^{2e+1}1$ at level ℓ. By the assumptions, one of the two cases must occur. In the first case $f(e) = 1$ and in the second case $f(e) = 0$. □

Lemma 2.15 exhibits a notion of 'highness' of trees T, in the sense that every tree with the same paths must be at least as complicated as T. Corollary 2.16 shows that a tree may possess a 'highness' property of this type while being computationally low in other respects. The proof uses the notion of a diagonally non-computable function, i.e. a function f such that each $f(e)$ is different from the value of the eth machine on e, if the latter is defined. The diagonally non-computable functions that only take values 0,1 form a Π^0_1 class in the Cantor space. Moreover Arslanov's completeness criterion from [1] says that a c.e. degree is complete if and only if it computes a diagonally non-computable function. We refer to [7, Section 2.22] for background on this topic. Moreover the proof uses the low basis theorem from [10]. See [17, Section 1.8] for a thorough discussion of the latter.

Corollary 2.16. *There exists a Δ^0_2 tree T of superlow degree (with only computable paths) such that all Σ^0_1 trees Q with $[T] = [Q]$ have degree $\mathbf{0}'$.*

Proof. This follows from Lemma 2.15. Indeed, if we apply the low basis theorem to the Π^0_1 class of diagonally non-computable functions which only take values 0,1 we obtain a superlow diagonally non-computable degree **a**. An application of Arslanov's completeness criterion concludes the proof. □

Corollary 2.16 also holds for trees of bounded width.

Proposition 2.17. *There exists a Δ^0_2 tree T of superlow degree of width 1, such that all Σ^0_1 trees Q with $[T] = [Q]$ have degree $\mathbf{0}'$.*

Proof. Let T consist of a single superlow set X of PA degree. If Q is a tree with unique infinite path X, then $X \leq_T Q$. By Arslanov's completeness criterion, if Q is also c.e. then $\emptyset' \leq_T Q$. □

Recall that if a tree T has finitely many infinite paths X, then each such X is isolated and computable from T. Therefore

$$\text{If } T \text{ is a tree with finitely many paths } X_i,\ i < n \text{ then there exists a tree } Q \text{ such that } [T] = [Q] \text{ and } Q \equiv_T \oplus_{i<n} X_i. \quad (2.5)$$

The following is a version of Theorem 2.14 for $n = 2$, where the given tree is computable in some c.e. set C. In that case the constructed tree Q can be chosen to be computable in C.

Proposition 2.18. *If T is a tree, C is a c.e. set and $T \leq_T C$ then there exists a c.e. tree Q such that $[T] = [Q]$ and $Q \leq_T C$.*

Proof. The construction of Q is identical to the construction in the proof of Theorem 2.14, only that we use the computable approximation $T[s]$ to T that corresponds to the reduction $T \leq_T C$ and the computable enumeration of C. Since C is c.e. and $T \leq_T C$, the oracle C can compute for each string σ a stage s_σ such that $T(\sigma)[s]$ remains constant for all $s \geq s_\sigma$. Therefore, for each string σ the oracle C can compute a stage t_σ such that either $\sigma \in Q[t_\sigma]$ or $\sigma \notin Q$. Hence $Q \leq_T C$. $\square$

Theorem 2.19 shows that, despite Corollary 2.16, we do have some control over the weak truth table degrees of the Σ^0_1 tree representation of the class of paths through a Δ^0_2 tree. For a number of characterizations of the sets which wtt-bound a diagonally non-computable function (in terms of Kolmogorov complexity) we refer to [11,12]. In this section we will use the notion of a *complex set*. According to [11,12] we say that a set A is *complex* if there is an unbounded non-decreasing computable function f such that $K(A \upharpoonright_n) \geq f(n)$ for all $n \in \mathbb{N}$. This is equivalent to the existence of a computable function g such that $K(A \upharpoonright_{g(n)}) \geq n$ for all $n \in \mathbb{N}$. In the same paper it was shown that a set is complex iff it wtt-computes a diagonally non-computable function.

Theorem 2.19. *Given any Δ^0_2 tree T there is a Σ^0_1 tree Q such that $[T] = [Q]$ and there is no diagonally non-computable function $f \leq_{wtt} Q$. In particular, Q is weak truth table incomplete.*

Proof. We combine the argument of Theorem 2.14 with diagonalization. Let $(\Psi_e; \psi_e)$ be a list of all partial weak truth table reductions (where ψ_e is a partial computable function giving a strict upper bound on the use of the oracle in procedure Ψ_e). We may assume that ψ_e are increasing and if $n < m$, $\psi_e(m)[s] \downarrow$ then $\psi_e(n)[s] \downarrow$. By convention, if $\psi_e(n)[s] \downarrow$ then $e, n, \psi_e(n) < s$. If $\Psi_e^X(n) \downarrow$ for some oracle X then $\psi_e(n)[s] \downarrow$ and the use of X in the computation $\Psi_e^X(n)$ is bounded by $\psi_e(n)[s]$. Also, we may choose the machines Ψ_e to take sets of strings as oracles. In this case, the use of the oracle in a computation is the length of the longest string which was involved in a query during the computation.

Let $T[s]$ be a computable approximation to T such that for each $s \in \mathbb{N}$, if $\sigma \in T[s]$ then all $\tau \subset \sigma$ are in $T[s]$. In order to deal with diagonally non-computable functions, we need a partial control over the diagonal function $\varphi_e(e)$ (where (φ_e) is a universal enumeration of all partial computable functions). We construct a partial computable function g and by the recursion theorem we may use a computable function p such that $g(n) \simeq \varphi_{p(n)}(p(n))$ for each $n \in \mathbb{N}$.

We construct a c.e. tree Q and denote by $Q \upharpoonright_n$ the set of strings in Q of length less than n. The following requirements must be met.

R_e : Ψ_e^Q is not diagonally non-computable.

Satisfying one requirement. The *standard* enumeration of Q consists of enumerating σ into Q whenever there is a stage $s > |\sigma|$ such that $\sigma \in T[s]$. While running the *standard* enumeration of Q, we also wait until $\psi_e(p(0)) \downarrow$. If and when this happens, we enumerate into Q all strings of length $\psi_e(p(0))$. If at a later stage $\Psi_e^Q(p(0)) \downarrow$, we define $g(0) = \Psi_e^Q(p(0))$.

Since the extra enumeration into Q happens only once (and the *standard* enumeration of Q continues throughout the construction) the argument given in the proof of Theorem 2.14 shows that Q is a tree and $[T] = [Q]$. Also, if Ψ_e^Q is total, clearly $g(0) = \varphi_{p(0)}(p(0)) = \Psi_e^Q(p(0))$. Hence Ψ_e^Q is not diagonally non-computable and R_e is satisfied.

Satisfying all requirements. The global construction is a finite injury argument. Strategy R_e is allowed to define g on $\mathbb{N}^{[e]} = \{\langle e, n\rangle \mid n \in \mathbb{N}\}$. It also has current witness $n_e[s]$ at stage s, on which it is about to define g. We may define $n_e[s]$ in advance, as the least $n \in \mathbb{N}^{[e]}$ such that either $\Psi_e^Q(p(n))[s] \downarrow = g(n)[s-1]$ or $g(n)[s-1] \uparrow$. As usual, the suffix '$[s]$' indicates the value of a parameter at the end of stage s. We say that R_e *requires attention at stage* s if

- $\psi_e(p(i))[s] \downarrow$ and $\Psi_e^Q(p(i))[s] \downarrow$ for all $i \leq n_e[s]$
- $g(i)[s-1] \downarrow \Rightarrow g(i) \neq \Psi_e^Q(p(i))[s]$, for all $i \leq n_e[s]$.

Notice that if R_e requires attention at stage s then $g(n_e[s])$ is undefined at the beginning of stage s.

To ensure that $[Q] = [T]$ we use movable markers ℓ_e which correspond to levels of Q where its enumeration is 'controlled'. Let $\ell_{-1}[s] = 0$ for each $s \in \mathbb{N}$ and let $\ell_e[s]$ be the least number which is greater than all $\ell_i[s]$, $i < e$ and greater than all $\psi_e(p(i))[s]$, $i \leq n_e[s]$ that are defined.

Fig. 1 illustrates the relation of these parameters. The shaded cones above level ℓ_{e-1} indicate that changes strictly below level ℓ_e (in particular, levels $\leq \psi_e(p(n_e))$) are always accompanied by changes at levels $\leq \ell_{e-1}$. This crucial principle is met in Step (a) of the construction, where it is also explained in italics. As a result, a diagonalization for R_e when the shaded cone below ℓ_{e-1} has settled is a successful one. We order the strings first by length and then lexicographically.

Construction at stage $s+1$.

(a) *Enumeration of Q:* If some string of length $\ell_k[s] < s$ for some k is in $T[s]$, enumerate it and all of its extensions of length $< \ell_{k+1}[s]$ into Q.

> *Under the enumeration of Q, if $\Psi_e^Q(p(n_e[s]))[s] = g(n_e[s])$ the only reason why this computation may change is that $Q \upharpoonright_{\ell_{e-1}[s]+1}$ changes. Inductively, marker ℓ_{e-1} will reach a limit and after finitely many definitions of g in $\mathbb{N}^{[e]}$ we will have $\Psi_e^Q(p(n_e)) = g(n_e)$ permanently, for a final witness n_e.*

(b) *Action of strategies:* Find the least $e \leq s$ such that R_e requires attention and define $g(n_e[s]) = \Psi_e^Q(p(n_e[s]))[s]$.

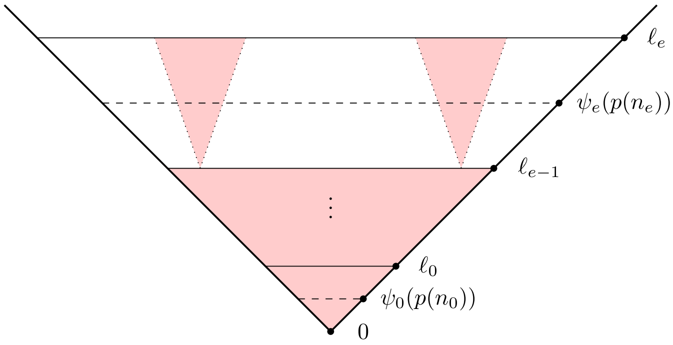

Fig. 1. The tree Q with the parameters at the various levels.

Verification. First we show by induction that R_e is satisfied and ℓ_e, n_e reach a limit, for all $e \in \mathbb{N}$. Suppose that this holds for all $i < e$ and let s_0 be the least stage such that ℓ_{e-1} and $Q \upharpoonright_{\ell_{e-1}+1}$ remain constant for all $s \geq s_0$. If at stages $\geq s_0$ strategy R_e does not require attention, requirement R_e is satisfied. Otherwise it will define $g(n_e[s]) = \Psi_e^Q(p(n_e))[s]$ at some stage $s \geq s_0$. This equality is going to be preserved in later stages, since $Q \upharpoonright_{\psi_e(p(n_e[s]))}$ does not change unless $Q \upharpoonright_{\ell_{e-1}+1}$ changes. Therefore R_e is satisfied. Moreover n_e can only move to smaller values after s_0. Since Ψ_e has computable use this can only happen finitely often. So n_e reaches a limit. This also means that ℓ_e will reach a limit, which concludes the induction step.

Second, we show that $[T] = [Q]$. Since the markers ℓ_e, $e \in \mathbb{N}$ are in increasing order and they all reach a limit, Step (a) of the construction implies that $[T] \subseteq [Q]$.

For the converse suppose that $X \notin [T]$. Then there exists some $\sigma \subset X$ and a stage s_1 such that $T(\sigma)[s] = 0$ for all $s \geq s_1$. We may also choose s_1 large enough so that all levels $\leq |\sigma|$ in Q have settled. By the enumeration we chose for T, this means that at stages $s \geq s_1$ all extensions of σ are outside $T[s]$. We claim that no extension of σ appears at level s_1 of Q. Indeed, suppose that τ of length s_1 is enumerated in Q at some stage $s_2 + 1$. Let $\rho \subseteq \tau$ be the minimal string that was enumerated in Q at the same stage. Then $\rho \in T[s_2]$ so $\sigma \not\subseteq \rho$ by the choice of s_1. But $|\sigma| < |\rho|$ by the choice of s_1. Therefore $\sigma \not\subseteq \tau$. Since no extension of σ appears at level s_1 of Q we have $X \notin [Q]$. $\square$

In [3, Section 2] the class of *infinitely often (i.o.) K-trivial* sets was studied. These are the sets A such that for some $e \in \mathbb{N}$ there are infinitely many n such that $K(A \upharpoonright_n) \leq K(n) + e$. It was shown that *if a set is not complex, it is i.o. K-trivial.* Identifying sets of binary strings with subsets of $\mathbb{N}$ (according to the standard ordering of strings, first by length and then lexicographically) we have the following.

Corollary 2.20. *Given any Δ_2^0 tree T there is an infinitely often K-trivial c.e. tree Q which has the same infinite paths as T.*

In Section 3 we will show how we can obtain a c.e. K-trivial tree Q, under a stronger assumption about T.

We say that a function g is diagonally non-computable relative to A if $g(e) \not\simeq \Phi_e^A(e)$ for all $e \in \mathbb{N}$ (where (Φ_e) is a universal enumeration of all Turing functionals). The class of these functions is also denoted by DNC[A]. We note that the proof of Theorem 2.19 relativizes to all levels of the arithmetical hierarchy, giving the following: If $n > 1$ and T is a Δ_n^0 tree, there is a Σ_{n-1}^0 tree Q such that $[T] = [Q]$ and there is no function $f \in \mathrm{DNC}[\emptyset^{(n-2)}]$ such that $f \leq_{\mathrm{wtt}} Q$.

Finally the following variation of Theorem 2.19 holds.

Theorem 2.21. *Given any Δ_2^0 tree T and a c.e. non-computable set A there is a Σ_1^0 tree Q such that $[T] = [Q]$ and $A \not\leq_{wtt} Q$.*

We omit the proof since it is very similar to the proof of Theorem 2.19. We note that Theorem 2.19 can be proved using a less direct and more economical argument which is based on Kolmogorov complexity and uses the results of [11,12] that were discussed above. The reason we followed the more direct approach is that it extends to a proof of Theorem 2.21. In the following we sketch this alternative proof.

Since T is Δ_2^0 it is the pointwise limit of a sequence of finite trees (F_n). Let f be a left-c.e. (i.e. it can be computably approximated from below) increasing function which grows faster than any computable one. Let Q be the tree defined by

$$\tau \in Q \iff \exists n\, [\tau \in F_n \wedge f(n) \geq |\tau|]. \tag{2.6}$$

It is not hard to see that Q is a c.e. tree and that $[Q] = [T]$. In order to talk about the Kolmogorov complexity of Q, let us identify strings with numbers (in the standard way, numbering them first by length and then lexicographically). In this way, binary strings represent finite sets of strings (in the same way that they represent finite sets of numbers). Similarly, an infinite tree can be represented by a certain infinite binary code. Let $Q \upharpoonright_n$ denote the (code of the) subtree of Q consisting of its first n levels (i.e. strings of length $< n$).

By (2.6) it follows that for each $n \in \mathbb{N}$, if $\ell \in [f(n), f(n+1))$ then (identifying trees with their codes) $Q \upharpoonright_\ell$ is obtained from $Q \upharpoonright_{f(n)}$ by adding all extensions of its leaves of length $<\ell$. Now using the fact that f grows faster than any computable function, it is not hard to show that there is no computable function g such that $K(Q \upharpoonright_{g(n)}) \geq n$ for all n. In other words, Q is not complex. Hence by [11,12] it does not compute a diagonally non-computable function.

3. The number of K-trivial sets

In this section we apply the general analysis and the ideas from Section 2 in order to give a positive answer to Question 1.1 about the K-trivial sets that was discussed in the introduction. Throughout the section the trees T_e are fixed as follows.

Definition 3.1 (K-*triviality Trees*)**.** Let T_e be the set of strings σ with the property $K(\sigma\restriction_i) \le K(i) + e$ for all $i < |\sigma|$.

Clearly each T_e is a tree and by the coding theorem (from [18,14,4,21]) there is a constant c such that the number of strings at each level of T_e is bounded by 2^{e+c}. In particular the trees T_e have bounded width, which implies the they also have finite number of paths. Notice that $\mathcal{K}_e = [T_e]$. Hence the range of the function $e \to |\mathcal{K}_e|$ is a subset of $\mathbb{N}$.

3.1. Replacing trees with K-trivial trees

Identifying trees in the Cantor space with the infinite binary code describing its characteristic function, we can talk about K-triviality of trees. Recall that K-triviality is a degree-theoretic property. Hence the K-triviality of a tree in the Cantor space does not depend on the way we code it into a binary sequence.

Proposition 3.2. *For each $e \in \mathbb{N}$ there exists a c.e. K-trivial tree Q_e such that $[T_e] = [Q_e]$.*

Proof. The tree T_e has only finitely many infinite paths X_i, $i < k$ and all of them are K-trivial. Since the K-trivial sets are closed under join, $X := \oplus_{i<k} X_i$ is K-trivial. Also, every K-trivial set is computable from a c.e. K-trivial set. Let C be a c.e. K-trivial set such that $X \le_T C$. By (2.5) there is a tree $S_e \le_T C$ such that $[S_e] = [T_e]$. By Proposition 2.18 there is a c.e. K-trivial tree Q_e such that $[S_e] = [Q_e]$. Hence $[T_e] = [Q_e]$. □

As observed above, whether a tree is K-trivial is independent of the way that it is coded into an infinite binary sequence. However when we talk about K-triviality of a tree *via a specific constant*, the way the tree is coded into an infinite binary sequence does matter. In other words, different codings may give rise to different K-triviality constants. For this reason we fix the canonical way of coding, which corresponds to the natural enumeration of binary strings, first by length and then lexicographically. For a uniform version of Proposition 3.2 we need a direct construction.

Theorem 3.3. *There is a constant c and a uniformly c.e. sequence (Q_e) of trees such that Q_e is K-trivial with constant $2e + c$ and $[T_e] = [Q_e]$, for all $e \in \mathbb{N}$.*

Proof. Let $\sigma \to f(\sigma)$ be the standard 1–1 (computable) function from $2^{<\omega}$ onto $\mathbb{N}$ that enumerates the binary strings first by length and then lexicographically. This function also defines a total ordering amongst the strings. Using f, any infinite binary sequence can be seen as a subset of $2^{<\omega}$ and vice-versa. In the following we identify trees with their binary codes (via f). Let $g(n)[s]$ be the least number $k > n$ such that $\sum_{i>k} 2^{-K(i)[s]} < 2^{-n}$. Clearly $g(\cdot)[\cdot]$ is computable, non-decreasing in n and $\lim_s g(n)[s] := g(n)$ exists for each $n \in \mathbb{N}$. Let $T_e[s]$ be a computable approximation to T_e (uniformly in e) such that for each $s \in \mathbb{N}$, if $\sigma \in T_e[s]$ then all $\tau \subset \sigma$ are in $T_e[s]$. To demonstrate the K-triviality of the constructed sets, we build a prefix-free machine $M = \cup_e M_e$. Let U be the underlying universal machine. We order the strings first by length and then lexicographically.

Intuition. At each stage we will be enumerating short M_e-descriptions of the code of Q_e that ensure that it is K-trivial. Each enumeration into Q_e carries a certain cost, namely the weight of the new descriptions needed for the segments of its code that change. On the other hand such enumerations are triggered by new strings appearing in T_e with short descriptions. The construction makes sure that an enumeration into Q_e is only done when we can juxtapose the cost of it with the weight of the new U-descriptions that have been produced in order for certain strings to appear in T_e. This accounting of the M_e-cost of changes in Q_e against the U-cost of new strings appearing in T_e is highlighted in (3.1) and illustrated in Fig. 2. The top line in Fig. 2 refers to lengths of the code of Q_e and the bottom line refers to lengths of strings that are described by U. The bold segment in the top line covers the 'main' segments of the code of Q_e that will need new descriptions, should we choose to enumerate σ in Q_e. The dotted segment in the top line covers the segments of the code of Q_e for which the weight of possible new descriptions at stage s is negligible. The bold segment of the bottom line depicts the segments of an extension τ of σ which have received short descriptions by U. The construction will only enumerate σ into Q_e if there is an extension τ as shown in the figure. In this case it will also enumerate all initial segments of τ. Notice that the construction of Q_e and M_e does not interact with the constructions of Q_i, M_i for $i \neq e$.

Construction. At stage $s + 1$ do the following for each $e < s$:

(a) Look for the least $k < s$ such that $K_{M_e}(Q_e\restriction_k)[s] > K(k)[s] + 2e$ and (if it exists) enumerate an M_e-description of $Q_e\restriction_k$ of length $K(k)[s] + 2e$.
(b) Look for the least string σ such that $f(\sigma), g(f(\sigma))[s] < s, \sigma \notin Q_e[s]$ and there is an extension $\tau \in T_e[s]$ of σ with $|\tau| < s$ such that $g(f(\sigma))[s] < |\tau|$. If such string σ exists, we pick the least one and enumerate τ and all of its initial segments into Q_e. We also make sure that

$$K_{M_e}(Q_e[s+1]\restriction_n) \le K(n)[s] + 2e$$

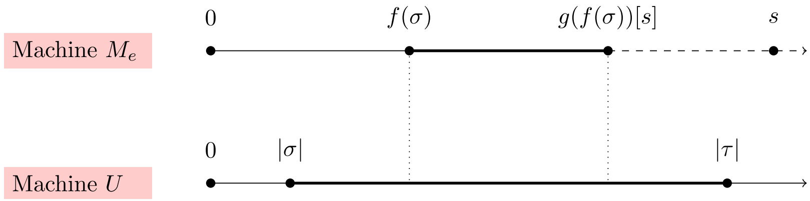

Fig. 2. The relation between the weights of M_e and U as this is described in (3.1).

for all $n < s$ by enumerating new M_e-descriptions for the $n < s$ such that $Q_e[s+1] \upharpoonright_n \neq Q_e[s] \upharpoonright_n$.

$$\text{\textit{All n for which we enumerate new descriptions at Step (b) are} } > f(\sigma)\text{. \textit{Since} } g(f(\sigma))[s] < |\tau| \text{ \textit{we can count the} } M_e\text{-\textit{weight} } \sum_{i\in[f(\sigma),g(f(\sigma))[s]]} 2^{-K(i)[s]-2e} \text{ \textit{against the} } U\text{-\textit{weight of the current descriptions of} } \tau \upharpoonright_i \text{ \textit{for} } i \in [f(\sigma), g(f(\sigma))[s]]. \tag{3.1}$$

Verification. Clearly the constructed sets Q_e are uniformly c.e. trees. We first show that $[T_e] = [Q_e]$ for all $e \in \mathbb{N}$. Fix $e \in \mathbb{N}$. By the construction, a string σ can only be enumerated into Q_e at a stage $s > |\sigma|$ such that $\sigma \in T_e[s]$. Hence the properties of the approximations $T_e[s]$ imply that $[Q_e] \subseteq [T_e]$ for each $e \in \mathbb{N}$. To show the converse let $X \in [T_e]$, $n \in \mathbb{N}$. It suffices to show that at some stage $\sigma := X \upharpoonright_n$ is enumerated in Q_e. Let stage $s_0 > n$ be large enough such that the membership in Q_e of all smaller strings than σ has been settled, $g(f(\sigma))[\cdot]$ has reached a limit $g(f(\sigma))$ and there is some extension $\tau \in T_e[s_0]$ of σ such that $|\tau| < s_0$ and $g(f(\sigma)) < |\tau|$. Since there is an infinite extension of σ in T_e such a stage s_0 will be reached. If σ has not been enumerated in Q_e before s_0, according to the construction it will be at this stage. This completes the proof that $[Q_e] = [T_e]$ for each $e \in \mathbb{N}$.

The instructions for M_e, $e \in \mathbb{N}$ that were produced in the construction ensure that $K_M(Q_e \upharpoonright_n) \leq K(n) + 2e$ for all $n, e \in \mathbb{N}$. Indeed, for each $n \in \mathbb{N}$ Step (a) provides the initial description $Q_e \upharpoonright_n$ and replenishes it in case $K(n)$ changes. If $Q_e \upharpoonright_n$ changes at some stage s (due to an enumeration of some strings) Step (b) makes sure that a new description is produced for the new value of $Q_e \upharpoonright_n$. It remains to show that the weight of the descriptions produced by M is bounded. Indeed, in that case M is a prefix-free machine and $K_M(Q_e \upharpoonright_n) \leq K(n) + 2e$ for all $n, e \in \mathbb{N}$. Therefore if c is an index of it we have $K(\sigma) \leq K_M(\sigma) + c$ for all strings σ and so, $K(Q_e \upharpoonright_n) \leq K(n) + 2e + c$ for all $n, e \in \mathbb{N}$. In particular Q_e is K-trivial with constant $2e + c$.

The weight of the M_e descriptions that are produced in Step (a) is bounded by 2^{-2e} times the weight of the domain of the universal machine. Indeed, for each new U-description of some $n \in \mathbb{N}$ Step (a) enumerates a description of the current $Q_e \upharpoonright_n$ which is longer by $2e$ bits. Since the weight of the domain of U is less than 1, the following holds.

$$\text{The weight of the } M_e \text{ descriptions issued in Step (a) is } < 2^{-2e}. \tag{3.2}$$

The bulk of the weight of the descriptions that are produced in Step (b) can be counted against the weight of the U-descriptions, as we indicate in (3.1). Indeed, by the definition of T_e the weight of the new M_e-descriptions that are enumerated at Step (b) for segments of Q_e up to length $g(f(\sigma))[s]$ is 2^e times smaller than the U-weight of the current descriptions of $\tau \upharpoonright_i$ for $i \in [f(\sigma), g(f(\sigma))[s]]$. Let us denote this additional M_e-weight at stage s by $w_e[s]$. The descriptions in the U-weight have not been counted in previous stages because σ (and all of its extensions) are not in $Q_e[s]$. Also the descriptions in the U-weight will not be counted in later stages because $\tau \upharpoonright_i \in Q_e[s+1]$ for $i \in [f(\sigma), g(f(\sigma))[s]]$. Hence each increase $w_e[s]$ in the weight of M_e at Step (b) corresponds to a finite set of U-descriptions of weight $2^e \cdot w_e[s]$, in a way that increases in different stages correspond to disjoint sets of U-descriptions. Hence $\sum_s w_e[s] < 2^{-e} \cdot \mathtt{wgt}(U) < 2^{-e}$. By the definition of g and since $g(f(\sigma))[s] < |\tau|$ the remaining descriptions issued at Step (b) of stage s (that were not counted in $w_e[s]$) have weight at most $2^{-f(\sigma)-e}$. Therefore the total weight we add to M_e at Step (b) of stage s is $\leq w_e[s] + 2^{-f(\sigma)-e}$. But each string is enumerated into Q_e at most once. Since f is 1–1, by (3.2) and the previous discussion we have

$$\mathtt{wgt}(M_e) \leq 2^{-2e} + \sum_s w_e[s] + \sum_{\rho\in 2^{<\omega}} 2^{-f(\rho)-e} < 2^{-e+4}.$$

So $\mathtt{wgt}(M) = \sum_e \mathtt{wgt}(M_e) < \infty$. This shows that M is a prefix-free machine and concludes the proof. □

Notice that the trees c.e. Q_e in Theorem 3.3 cannot have bounded width. Indeed, for each c.e. tree of bounded width there exists a computable tree with the same (finitely many) paths. Therefore if Q_e was c.e. all of its paths would be computable; so $[T_e] \neq [Q_e]$ for sufficiently large e, since there are K-trivial non-computable paths.

The proof of Theorem 3.3 shows something more than the statement. Indeed, the only facts about the trees T_e that was used in this proof is that they are Δ^0_2 and $[T_e] \subseteq \mathcal{K}_e$. Hence the same proof shows the following. Recall that a Δ^0_2 index of a set X is a number n such that $X = \Phi^{\emptyset'}_n$. A Σ^0_1 index of X is a number n such that X is the n th c.e. set, under the standard universal enumeration of all c.e. sets.

Corollary 3.4. *There exists a partial computable function p and a constant c such that for each n, e the following holds. If n is a Δ^0_2 index of a tree T and $[T] \subseteq \mathcal{K}_e$ then $p(n, e)$ is defined and equal to a c.e. index of a tree Q such that $[T] = [Q]$ and (the code of) Q is in $\mathcal{K}_{c+2e+n}$.*

Corollary 3.4 can be stated more informally as follows.

$$\text{Given a } \Delta^0_2 \text{ tree } T \text{ and } e \in \mathbb{N} \text{ such that } [T] \subseteq \mathcal{K}_e \text{ we can uniformly produce a } \Sigma^0_1 \text{ tree } Q \text{ and some } m \in \mathbb{N} \text{ such that } [T] = [Q] \text{ and (the code of) } Q \text{ is in } \mathcal{K}_m. \tag{3.3}$$

In Section 4 we will use (3.3) in order to answer the analogue of Question 1.1 for the class of low for K sets.

3.2. Answer to Question 1.1

Recall that a set X is K-trivial with constant e if $K(X\restriction_n) \le K(n) + e$ for all $n \in \mathbb{N}$. In this case e is a K-triviality constant for X. There is an effective list of the K-trivial sets (in terms of Δ^0_2 indices of them) along with K-triviality indices of them. This was shown in [8]. Also, in [16] it was shown that all K-trivial sets are low. But how easy is it to extract the 'lowness' of a K-trivial set? A lowness index of a set X is an index of a reduction of X' to $\emptyset'$. By [16] (also see [17, Proposition 5.5.5]) there is no effective procedure which takes a K-trivial (along with a constant for its K-triviality) and returns a lowness index of it. In fact, this can be extended as follows.

Proposition 3.5. *There is no $\emptyset'$-effective procedure which takes $e \in \mathbb{N}$ and a Δ^0_2 index of a K-trivial set X with constant e and returns a lowness index of X.*

We sketch the proof of Proposition 3.5. It is based on the following extension of [17, Theorem 5.3.22].

$$\text{Given a low c.e. set } X \text{ and a computable approximation to a lowness index of } X \text{ we can effectively produce a } K\text{-trivial set } A \text{ such that } A \not\le_T X. \tag{3.4}$$

The proof of (3.4) is an extension of the argument that was used in [17, Theorem 5.3.22], where changes in the computable approximations initialize the strategies and lower the 'cost quotas'. Assuming that Proposition 3.5 does not hold, one could use the effective list of K-trivial sets from [8] and get an effective list of all the K-trivial sets along with computable functions that converge to a lowness index of them. Then using a minor modification of (3.4), one would be able to construct a K-trivial set which is not computable by any of the sets in the effective list. This gives the desired contradiction.

It turns out however that $\emptyset''$ has sufficient power to recover a lowness index of a K-trivial set, given its K-triviality constant.

Proposition 3.6. *There is a partial $\mathbf{0}''$-computable function f such that for each $e \in \mathbb{N}$, if $\Phi^{\emptyset'}_e := X_e$ is total and K-trivial then $f(e)$ is a lowness index of it (i.e. $X'_e = \Phi^{\emptyset'}_{f(e)}$).*

Proof. Given $e, b \in \mathbb{N}$ such that $\Phi^{\emptyset'}_e := X_e$ is total, the question whether X_e is low for K with constant b (i.e. $\forall n,\ K(n) \le K^{X_e}(n) + b$) is decidable in $\mathbf{0}''$. Indeed, this follows from the fact that $K(n)$ and $K^{X_e}(n)$ are computable in $\mathbf{0}'$. Hirschfeldt and Nies showed in [16] that every K-trivial is low for K (also see [17, Theorem 5.4.1]). Hence given $e \in \mathbb{N}$ such that $\Phi^{\emptyset'}_e := X_e$ is total and K-trivial, we can $\mathbf{0}''$-effectively search and find a constant b such that X_e is low for K with constant b. Let h be a partial $\mathbf{0}''$-computable function which computes this b (given input e as above).

By (essentially) [13] (also see [17, Proposition 5.1.2]) the low for K sets are uniformly generalized low. Hence, given e, b such that $\Phi^{\emptyset'}_e := X_e$ is total and $\forall n\, K(n) \le K^{X_e}(n) + b$) we can compute a lowness index of X_e. Let g be a partial computable function that gives this computation. Then the partial $\mathbf{0}''$-computable function $g(f(e))$ takes any $e \in \mathbb{N}$ and if $\Phi^{\emptyset'}_e := X_e$ is total and K-trivial it returns a lowness index for X_e. □

Corollary 3.7. *The function $G_{\mathcal{K}}$ is Δ^0_3.*

Proof. Notice that from a lowness index of a set X we can effectively compute a low$_2$-ness index of it (i.e. a number e such that $\Phi^{\emptyset''}_e = X''$). Therefore the corollary follows by a combination of (2.2), Theorem 3.3 and Proposition 3.6. □

Downey/Miller/Yu also showed (see [7, Theorem 10.1.12] or [17, Proposition 5.2.13, Exercise 5.2.15]) that the function $G_{\mathcal{K}}$ is not Δ^0_2. Hence Corollary 3.7 gives a sharp classification of the arithmetical complexity of $G_{\mathcal{K}}$. However it seems harder to code specific information in $G_{\mathcal{K}}$. We close this section with the following question.

Question 3.1. Does $G_{\mathcal{K}}$ compute $\emptyset''$ or $\emptyset'$? Does this depend on the choice of the underlying universal prefix-free machine?

4. The number of low for K sets

Question 1.1 can be asked about related Σ^0_3 classes, like the low for K sets. Recall that a set X is low for K if the prefix-free complexity function is dominated by its X-relativized version modulo a constant. The latter constant is said to be a low for K constant of X. Although by [16] the low for K sets coincide with the K-trivial sets, they are parametrized in different ways that are not effectively equivalent. To wit, as it is explained in [8] (and subsequently in [16]) one cannot effectively obtain a low for K constant of a set, given a K-triviality constant of it. Hence the analogue of Question 1.1 for low for K sets cannot be answered simply by using Corollary 3.7. In this section we provide a positive answer.

The class of low for K sets can be expressed as an effective union of the Π^0_2 classes

$$\mathcal{L}_e = \{X \mid \forall \tau \in 2^{<\omega},\ K(\tau) \le K^X(\tau) + e\}$$

for $e \in \mathbb{N}$. A set in $\mathcal{L}_e$ is said to be *low for K with constant e*. Let us denote the class of low for K sets by $\mathcal{L}$. We are interested in the arithmetical complexity of the function $G_{\mathcal{L}}(e) := |\mathcal{L}_e|$.

The next step is to express each class $\mathcal{L}_e$ as the set of infinite paths through a certain Δ^0_2 tree. Let

$$L_e = \{\sigma \mid \forall \ell > |\sigma|\ \forall s\ \forall \tau \in 2^{<|\sigma|}\ \exists \rho \in 2^{\ell}\ [\sigma \subset \rho \wedge K(\tau) \le K^{\rho}(\tau)[s] + e]\}.$$

Each set L_e is clearly a tree, and since the function $\tau \to K(\tau)$ is Δ^0_2, so is L_e. Moreover, $\mathcal{L}_e = [L_e]$ for each $e \in \mathbb{N}$.

Theorem 4.1. *The function $G_{\mathcal{L}}$ is Δ^0_3.*

Proof. By [16] (also see [17, Proposition 5.2.3]) there is a constant d such that if a set is low for K with constant e then it is K-trivial with constant $e + d$. Therefore $[L_e] \subseteq \mathcal{K}_{e+d}$ for all $e \in \mathbb{N}$. By Corollary 3.4 (also see (3.3)) there is a computable function q and a uniformly Σ^0_1 sequence (Q_e) of trees such that $[Q_e] = [L_e]$ and $Q_e \in \mathcal{K}_{q(e)}$ for all $e \in \mathbb{N}$. Given $e \in \mathbb{N}$ and $\emptyset''$ we describe how to compute $G_{\mathcal{L}}(e)$. By Proposition 3.6 we can use $\emptyset''$ to obtain a lowness index of Q_e. Then by (2.2) we can compute the number of infinite paths through Q_e. By the choice of Q_e this is equal to $G_{\mathcal{L}}(e)$. $\square$

As we have already pointed out, by [16] (also see [17, Proposition 5.2.3]) there is a constant d such that if a set is low for K with constant e then it is K-trivial with constant $e + d$. Hence $G_{\mathcal{L}}(e) \le G_{\mathcal{K}}(e + d)$ for all $e \in \mathbb{N}$. On the other hand a result of Downey/Miller/Yu (see [7, Section 10.1.4] and [17, Theorem 5.2.12]) says that $\sum_e G_{\mathcal{K}}(e)/2^e < \infty$. Therefore $\sum_e G_{\mathcal{L}}(e)/2^e < \infty$.

Downey/Miller/Yu also showed (see [7, Section 10.1.4] and [17, Proposition 5.2.13, Exercise 5.2.15]) that there is no Δ^0_2 function $h : \mathbb{N} \to \mathbb{Q}$ such that $\lim_s h(s) = 0$ and $G_{\mathcal{K}}(e)/2^e < h(e)$ for all $e \in \mathbb{N}$. This was used in order to show that $G_{\mathcal{K}}$ is not Δ^0_2. We do the same for $G_{\mathcal{L}}$ in order to show that the complexity bound given in Theorem 4.1 is optimal.

Theorem 4.2. *There is no Δ^0_2 function $h : \mathbb{N} \to \mathbb{Q}$ such that $\lim_s h(s) = 0$ and $G_{\mathcal{L}}(e)/2^e < h(e)$ for all $e \in \mathbb{N}$. Therefore $G_{\mathcal{L}}$ is not Δ^0_2.*

Proof. Let $h : \mathbb{N} \to \mathbb{Q}$ be a Δ^0_2 function such that $\lim_s h(s) = 0$. We will construct a number t such that $G_{\mathcal{L}}(t)/2^t \ge h(t)$. Let U be the underlying universal oracle prefix-free machine.

Suppose, for the sake of an example, that h was computable. Then we would proceed as follows. We construct a prefix-free machine M and by the recursion theorem we may use an index d of it in its definition. We compute the least e such that $h(e + d) < 2^{-d}$, fix pairwise incomparable strings σ_i, $i < 2^e$ of the same length and define M such that $K_M(\tau) \le \min_{i<2^e} K^{\sigma_i 0^\omega}(\tau) + e$ for all strings τ. Clearly

$$\mathtt{wgt}(M) < 2^{-e} \sum_{i<2^e} \mathtt{wgt}(U^{\sigma_i 0^\omega}) < 1$$

so M is a prefix-free machine. Moreover $K(\tau) \le K_M(\tau) + d$ for all strings τ. Hence $K(\tau) \le K^{\sigma_i 0^\omega}(\tau) + e + d$ for all strings τ. This shows that all $\sigma_i 0^\omega$, $i < 2^e$ are in $\mathcal{L}_{e+d}$ and $G_{\mathcal{L}}(e + d) \ge 2^e$. So $h(e + d) < G_{\mathcal{L}}(e + d)/2^{e+d}$.

If the given function h is merely Δ^0_2, we only have a computable approximation such that $h(n)[s] \to h(n)$ as $s \to \infty$. We proceed with a finite injury version of the previous argument. We construct a prefix-free machine M and by the recursion theorem we may use an index d of it in its definition. Define $g(s)$ to be the least t such that $h(t + d)[s] < 2^{-d}$. Since $\lim_n h(n) = 0$ we have that $\lim_s g(s)$ exists.

Construction. Let $s_0 = 0$. We choose pairwise incomparable strings σ_i, $i < 2^{g(s_0)}$ of the same length and at stages $s > s_0$ with $g(t) = g(s_0)$ for each $t \le s$, we define M such that $K_M(\tau) \le \min_{i<2^{g(s_0)}} K^{\sigma_i 0^\omega}(\tau) + g(s_0)$ for all strings τ of length $< s$. Upon the first stage s_1 such that $g(s_1) \ne g(s_1 - 1)$ we choose $j < 2^{g(s_0)}$ such that

$$\mathtt{wgt}(M[s_1 - 1]) < \mathtt{wgt}(U^{\sigma_j 0^{s_1}}[s_1 - 1]) \quad \text{(such } j \text{ exists, see verification)}$$

and pick new strings $\sigma_i[s_1]$, $i < 2^{g(s_1)}$ which are pairwise incomparable and are extensions of $\sigma_j[s_1 - 1]$. We continue defining M as before but based on the new parameters, and for the stages $s > s_1$ such that $g(t) = g(s_1)$ for each $t \le s$. Continue as above for $s_0, s_1, \ldots$.

Verification. By simultaneous induction we show that

$$\mathtt{wgt}(M[s]) \le 2^{-g(s)} \sum_{i<2^{g(s)}} \mathtt{wgt}(U^{\sigma_i 0^\omega}[s]) \tag{4.1}$$

$$\exists j < 2^{g(s-1)}[\mathtt{wgt}(M[s-1]) \le \mathtt{wgt}(U^{\sigma_j 0^s}[s-1])] \quad \text{if } s = s_k, k > 0 \tag{4.2}$$

for each $s \in \mathbb{N}$. Notice that in (4.2) the suffix $[s-1]$ on $U^{\sigma_j 0^s}$ also applies to σ_j since the antichains σ_i, $i < 2^{g(s-1)}$ may change values in the course of the construction. At stage 0 both (4.1) and (4.2) hold trivially. Let $s > 0$ and suppose that they hold at stage $s-1$. If $s = s_k$ for some $k \in \mathbb{N}$, condition (4.2) holds because its negation contradicts (4.1) at stage $s-1$, which holds by the induction hypothesis. On the other hand, the construction does not enumerate M-descriptions at this stage so $\mathtt{wgt}(M[s]) = \mathtt{wgt}(M[s-1])$. Also, the new antichain σ_i, $i < 2^{g(s)}$ is defined so that all strings extend a string $\sigma_j[s-1]$ of the previous antichain with the property (4.2). Hence $\mathtt{wgt}(U^{\sigma_i 0^\omega}[s]) \geq \mathtt{wgt}(M[s])$ for all $i < 2^{g(s)}$ and (4.1) holds.

Now suppose that s is not one of the special stages s_k, so (4.2) holds trivially. For (4.1) we can argue exactly as in the case where h was computable. Any increase in $\mathtt{wgt}(M)$ would be due to a new description σ enumerated in one of $U^{\sigma_i 0^\omega}[s]$, $i < 2^{g(s)}$. But in that case the increase in $\mathtt{wgt}(M)$ would be $2^{-|\sigma|-g(s)}$. Hence by the induction hypothesis (4.1) holds at stage s. This concludes the induction and the proof of (4.1) and (4.2).

By (4.1) we have $\mathtt{wgt}(M) < 1$, so M is a prefix-free machine with index d. Since $\lim_s g(s)$ exists there are only finitely many special stages s_k. Let $\sigma_j, j < 2^e$ be the final antichain that is chosen, at the modulus of convergence s_* of g. Also let $g(s_*) = e$, so that $h(e+d) < 2^{-e}$. Since (4.2) holds at special stages, the construction will never get stuck and $K_M(\tau) \leq \min_{i<2^e} K^{\sigma_i 0^\omega}(\tau) + e$ for all strings τ. Since $K(\tau) \leq K_M(\tau) + d$ we have $K(\tau) \leq K^{\sigma_i 0^\omega}(\tau) + e + d$ for all strings τ. This shows that all $\sigma_i 0^\omega, i < 2^e$ are in $\mathcal{L}_{e+d}$ and $G_{\mathcal{L}}(e+d) \geq 2^e$. By the definition of e and g we have $h(e+d) < 2^{-d} \leq G_{\mathcal{L}}(e+d)/2^{e+d}$. $\square$

It is likely that the methodology provided in this paper will be useful for the solution of other similar problems in Kolmogorov complexity. For example it may be interesting to study versions of Question 1.1 with respect to plain Kolmogorov complexity.

Acknowledgement

Barmpalias was supported by a research fund for international young scientists No. 611501-10168 and an *International Young Scientist Fellowship* number 2010-Y2GB03 from the Chinese Academy of Sciences. Partial support was also obtained by the *Grand project: Network Algorithms and Digital Information* of the Institute of Software, Chinese Academy of Sciences.

References

[1] Marat M. Arslanov, On some generalizations of the fixed-point theorem, Soviet Mathematics 25 (1981) 1–10.
[2] Martijn Baartse, George Barmpalias, On the gap between trivial and nontrivial initial segment prefix-free complexity. Submitted, 2010.
[3] George Barmpalias, C.S. Vlek, Kolmogorov complexity of initial segments of sequences and arithmetical definability, Theoretical Computer Science 412 (41) (2011) :5656–5667.
[4] Gregory J. Chaitin, A theory of program size formally identical to information theory, Journal of the Association for Computing Machinery 22 (1975) 329–340.
[5] G. Chaitin, Information-theoretical characterizations of recursive infinite strings, Theoretical Computer Science 2 (1976) 45–48.
[6] Barbara F. Csima, Antonio Montalbán, A minimal pair of K-degrees, Proceedings of the American Mathematical Society 134 (5) (2006) 1499–1502. electronic.
[7] Rod Downey, Denis Hirshfeldt, Algorithmic Randomness and Complexity, Springer, 2010.
[8] Rod G. Downey, Denis R. Hirschfeldt, André Nies, Frank Stephan, Trivial reals, in: Proceedings of the 7th and 8th Asian Logic Conferences, Singapore Univ. Press, 2003, pp. 103–131.
[9] Yuri L. Ershov, A certain hierarchy of sets, Algebra i Logika 7 (1) (1968) 47–74.
[10] Carl G. Jockusch Jr., Robert I. Soare, Π^0_1 classes and degrees of theories, Transactions of the American Mathematical Society 173 (1972) 33–56.
[11] Bjørn Kjos-Hanssen, Wolfgang Merkle, Frank Stephan, Kolmogorov complexity and the recursion theorem, in: STACS, 2006, pp. 149–161.
[12] Bjørn Kjos-Hanssen, Wolfgang Merkle, Frank Stephan, Kolmogorov complexity and the recursion theorem, Transactions of the American Mathematical Society 363 (2011).
[13] Antonín Kučera, Sebastiaan A. Terwijn, Lowness for the class of random sets, Journal of Symbolic Logic 64 (4) (1999) 1396–1402.
[14] L.A. Levin, Laws of information conservation (nongrowth) and aspects of the foundation of probability theory, Problems of Information Transmission 10 (1974) 206–210.
[15] M. Li, P. Vitányi, An Introduction to Kolmogorov Complexity and its Applications, second edition, in: Graduate Texts in Computer Science, Springer-Verlag, New York, 1997.
[16] André Nies, Lowness properties and randomness, Advances in Mathematics 197 (1) (2005) 274–305.
[17] André Nies, Computability and Randomness, Oxford University Press, 2009.
[18] R.J. Solomonoff, A formal theory of inductive inference. I, Information and Control 7 (1964) 1–22.
[19] R. Solovay, Handwritten manuscript related to Chaitin's work. IBM Thomas J. Watson Research Center, Yorktown Heights, NY, 215 p., 1975.
[20] Richard Shore, Theodore Slaman, Working below a low_2 recursively enumerable degree, Archive for Mathematical Logic 29 (1990) 201–211.
[21] D. Zambella, On sequences with simple initial segments. ILLC technical report ML 1990-05, Univ. Amsterdam, 1990.