Definability in the degrees of randomness

MSc Thesis (*Afstudeerscriptie*)

written by

Charlotte Vlek
(born July 16th, 1985 in Groningen)

under the supervision of **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.*

Date of the public defense:	**Members of the Thesis Committee:**
September 7th, 2010	Frank Veltman (chair)
	George Barmpalias
	Harry Buhrman
	Sebastiaan Terwijn

Institute for Logic, Language and Computation

Abstract

A set or sequence is random when the prefix-free Kolmogorov complexity of its initial segments is relatively high: equal to the length of the segment (up to a constant). Using Kolmogorov complexity of initial segments, we can not only define when a set is random, but we can also compare which of two sets is more random. We say that a set A is K-below (or K-reduces to) a set B if $K(A\upharpoonright_n) \leq^+ K(B\upharpoonright_n)$ for all n. This reducibility gives the structure of the K-degrees. The sets in the lowest degree are called K-trivial sets.

This thesis studies arithmetical definability in the K-degrees. The main result we present, is the construction of a non-K-trivial Δ^0_2 set that does not bound any non-K-trivial set in a given Δ^0_2 family of sets. This implies that there is a non-K-trivial Δ^0_2 set that does not bound any non-K-trivial c.e. set. Furthermore, this result shows a structural difference between the K-degrees and the LK-degrees.

Similar to the above result, we also show that for all $n > 1$ there is a non-K-trivial Σ^0_n set that does not bound any non-K-trivial Δ^0_n set. We present the construction for the particular case of $n = 2$, and we show that this specific Σ^0_2 set forms a minimal pair in the K-degrees with any non-K-trivial c.e. set. This improves on the lowest complexity known so far for minimal pairs in the K-degrees.

Finally, we investigate the possibility of constructing a minimal pair in the K-degrees via gap functions for K-triviality. We show that no unbounded non-decreasing Δ^0_2 gap function can exist, thus showing that this method is not suitable for constructing a Δ^0_2 minimal pair in the K-degrees.

Acknowledgements

This thesis would not have been here without the great help and support from the people around me. I would like to thank George for his supervision. His great enthusiasm for the subject and the challenges he has set for me have helped a great deal to make this thesis into what it is now. Furthermore I want to thank Tom for all his feedback and the discussions we had about randomness, K and *De Toverberg*.

While writing this thesis and during this entire masters programme, the support and care from the people around me has meant a lot to me. I hope the those concerned know how important they have been to me. In particular I would like to mention a few who have been directly responsible for motivating me from time to time to keep studying logic and other abstract nonsense. Thanks to Christian for squash, Pål for listening to all infusian herbal albums with me and Alex for Mexican food in Diemen.

Contents

Introduction

A monkey that randomly writes on a typewriter for an infinite amount of time will eventually type the entire text of Shakespeare's Hamlet. This is called the *Infinite Monkey Theorem*, and it is a well known illustration of the concept of infinity. The odds that a monkey writes Hamlet in a finite time are of course extremely small, but in infinite time it will at some point surely happen.

The Infinite Monkey Theorem is not only a good example of infinity, it also illustrates randomness. If the monkey were to type the same letter over and over, or follow some other type of pattern (that does not contain Shakespeare's Hamlet itself), it would definitely not type up Hamlet since it keeps repeating the same thing. In other words: the monkey in the Infinite Monkey Theorem really needs to type random letters for the example to work.

We can think of randomness of infinite sequences in precisely this way. An infinite sequence is random if it does not follow any type of pattern. Then random sequences coincide with those that could contain the entire text of Hamlet, *and* Euclid's *Elements*, *and* Tolstoi's *Anna Karenina*. Random sequences are those that code a lot of information in them.

The field of *Algorithmic Randomness* studies infinite sequences. One can use that to study sets over natural numbers using the infinite sequences of zeros and ones representing them. An important distinction is that we do not study the randomness of *events* like coin tosses or rolling dice, but we are rather interested in the randomness of an infinite sequence that is given to us.

One way to define a random sequence is the one described above: a sequence is random if it does not follow any patterns. This is called 1-randomness, and is based on the concept of *Kolmogorov complexity*. Both will be defined in Chapter 1. In that chapter we also define a way to talk about *how* random different sets or sequences are compared to each other. This divides up the class of all sets into different *degrees of randomness*.

In this thesis we are interested in these degrees of randomness, and in particular how hard it is to compute certain sets in these degrees. This is what the title of this thesis refers to: we study arithmetical definability in the degrees of randomness.

Context and related work

As mentioned above the focus of this thesis lies in the area where Algorithmic Randomness and Computability Theory meet. For background on both fields we refer to either [DH10] or [Nie08]. More on Kolmogorov complexity can be found in those, as well as in [LV93].

We already briefly mentioned the fact that we can say that one set is *less random* than another. In that case we say that the less random set K-*reduces* to the other set (see Chapter 1). Therefore the question of arithmetical definability in the randomness degrees is a specific case of definability related to reducibilities.

When it comes to definability related to Turing reducibility, quite a number of results are known. For example, when a set Turing reduces to a Δ^0_2 set, it must be Δ^0_2 itself. The question whether such results hold in the case of other reducibilities is an interesting subject. The hope is that possibly some results about Turing reducibility can be transferred to other reducibilities quite easily.

However, in contrast to Turing reducibility, K-reducibility is a so-called *weak reducibility*. The term *weak* refers to the fact that there is no underlying map that shows how one set is related to the other. Another weak reducibility is for example LK-reducibility[1]. For this reducibility it was proved in [BLS08] that a set reducing to a Δ^0_2 set is not necessarily Δ^0_2, unlike in the Turing degrees. This is because certain sets can have uncountable lower cones, and in particular Δ^0_2 sets have uncountably many sets LK-reducing to them (unless the Δ^0_2 set LK-reduces to $\emptyset$).

The concept of K-reducibility has only been introduced quite recently, in the 2004 paper by Downey, Hirschfeldt and LaForte [DHL04]. In their paper they treat several different kinds of reducibilities, and in particular weak reducibilities. Since then, the K-degrees have been a subject of interest, but not much is known about them so far. Miller and Yu studied the K-degrees of random sets in [MY08] and [MY10].

In investigating arithmetical definability in the K-degrees, this thesis was first motivated by the following open question asked by Downey and Hirschfeldt [DH10, 554]:

> **Question**: Is there a pair of Δ^0_2 sets that forms a minimal pair in the K-degrees?

This asks for two sets that are themselves not in the lowest degree (the very non-random sets, called K-*trivial* sets), but everything that K-reduces to both is in the lowest degree (the sets form a minimal pair). Closely related to this question is the paper by Barmpalias that proves the impossibility of a Δ^0_2 minimal pair in the LK-degrees [Bar10].

In the K-degrees it is already known that a minimal pair exists, due to a construction from Csima and Montalbán [CM06]. They did not care much

[1] $A \leq_{LK} B$ iff $K^B(\sigma) \leq^+ K^A(\sigma)$ for all strings σ.

about the arithmetical complexity of the minimal pair, but it turned out to be Δ^0_4. Merkle and Stephan improved this by constructing a minimal pair of two Σ^0_2 sets [MS07], which is the lowest complexity known for such a pair. In this thesis we improve this result by constructing a minimal pair of a Σ^0_2 set with any Σ^0_1 set of non-zero K-degree.

Results in this thesis

The main result in this thesis is the existence of a non-K-trivial Δ^0_2 set that does not bound any non-K-trivial set in a given Δ^0_2 family of sets. Since the computably enumerable (c.e.) sets form a Δ^0_2 family of sets, it follows that there is a non-K-trivial Δ^0_2 set that does not bound any non-K-trivial c.e. set. This is surprising since c.e. sets have quite low initial segment complexity, even when they are non-K-trivial.

Furthermore, the existence of such a Δ^0_2 set shows a structural difference between the LK-degrees and the K-degrees, since in [Bar10] the opposite was shown for the LK-degrees: every nontrivial Δ^0_2 set bounds a nontrivial c.e. set. Extending that result led to the insight that every pair of nontrivial Δ^0_2 sets bounds one nontrivial c.e. set, showing that no Δ^0_2 minimal pair can exist in the LK-degrees. With the aforementioned result in this thesis we see that this argument does not carry over to the K-degrees.

Similar to the previously mentioned result, we also show that for any n, there is a non-K-trivial Σ^0_n set that does not bound any non-K-trivial Δ^0_n set. In particular, we do the construction for $n = 2$, which relativizes to higher complexity classes. We also show that the constructed Σ^0_2 set forms a minimal pair in the K-degrees with any non-K-trivial c.e. set, improving the result by Merkle and Stephan, as mentioned above.

Finally we also prove a result concerning gap functions for K-triviality: we prove that there cannot be an unbounded non-decreasing Δ^0_2 gap function. This is interesting because the first construction of a minimal pair in the K-degrees was done using a gap function [CM06]. An unbounded non-decreasing gap function always induces a minimal pair. By showing that no unbounded non-decreasing Δ^0_2 gap function can exist, we show that this method of constructing minimal pairs cannot be used to construct a Δ^0_2 minimal pair in the K-degrees.

Outline of the thesis

The remainder of this thesis is organized in the following way: we start with a chapter on some basic notions and results about randomness, (prefix-free) Kolmogorov complexity and randomness degrees. This chapter is meant for the reader to gain some familiarity with the subject: we define the basics and discuss some elementary results.

The second chapter introduces a new definition: that of infinitely often K-trivial sets. These sets are essential in some of the proofs in the following chapters, for example in constructing the minimal pair consisting of a Σ^0_2 set with any non-K-trivial c.e. set. In Chapter 2 we investigate the infinitely often K-trivial sets, showing some results on how they behave and where they occur.

The third chapter is about definability in the K-degrees and presents two main results, mentioned above: the construction of a non-K-trivial Σ^0_2 set that does not bound any non-K-trivial Δ^0_2 set, and a non-K-trivial Δ^0_2 set that does not bound any non-K-trivial set in a given Δ^0_2 family of sets.

The final chapter is about gap functions. It investigates how they can be constructed, how they induce a minimal pair, and what they could look like under different restrictions. The main result there is the impossibility of an unbounded non-decreasing Δ^0_2 gap function, showing that this method cannot be used for constructing a Δ^0_2 minimal pair.

1

Randomness and degrees of randomness

The main goal of this chapter is to briefly discuss some intuitions on randomness, Kolmogorov complexity and randomness degrees. We refer to [Nie08], [DH10] or [LV93] for more on this. Here we present the most important definitions that we need throughout the thesis, as well as some basic results that will come back in the following chapters.

As mentioned in the introduction we are interested in studying randomness of sets over natural numbers ($A \subseteq \mathbb{N}$), viewing them as infinite sequences[1]. One way to capture randomness is the one explained in the introduction, where a sequence is random when it follows no patterns. This is called 1-randomness, and we define it formally in the next section.

1-randomness deals with randomness from a perspective of computability theory. There are two other main approaches to randomness: it can also be seen from the perspective of unpredictability, where an infinite sequence is random if it is not possible to predict the next digit of the sequence, even when all previous digits are known. Or we can view randomness in a measure theoretic perspective, where a random infinite sequence should not satisfy any *effectively rare* properties. This is captured in *Martin-Löf* randomness (see [DH10, 233] for more on this).

It was shown that all three approaches (computability, unpredictability and measure theory) can be used to define the same notion of randomness. In other words, sets or sequences that are random in one sense, will also be random in the other. In particular, there is a theorem by Schnorr that shows that Martin-Löf randomness and 1-randomness are equivalent (see [DH10, 234]).

The advantage of using descriptive complexity (and 1-randomness), is that it gives us the opportunity to compare the degree of randomness of sets. We can compare the complexity of initial segments of the same length, and thus

[1] Each set A has a characteristic function $\chi_A : \mathbb{N} \to \mathbb{N}$ where $\chi_A(n)$ is 1 if $n \in A$ and 0 otherwise. We study the infinite sequence $f_A(0)f_A(1)f_A(2)...$, so a 1 at place n represents that $n \in A$.

get an idea of which set is more random than the other. We will study this in Section 1.3.

1.1 Descriptive complexity and 1-randomness

In this section we formalize the idea that an infinite sequence is random when its initial segments have no short descriptions. We begin by defining what constitutes a description of a finite string. A description τ of a string σ should be such that we can computably find the string σ when we are given τ. Also, a description should only describe one string, but it can be that one string has multiple descriptions. So we define a string τ to be a description of a string σ when there is a *machine* (which is essentially just a (partial) computable function) that produces σ from input τ.

One can think of each machine as representing a certain pattern that a string might follow [vL87, 124]. For example, we could have a machine that represents the pattern "zeros and ones interchanging starting with 1". It will take a natural number as input (let us say 5 or, in binary, 101), and output a string of length that number with zeros and ones interchanging starting with 1 (10101). In the end we would like to take into account all possible machines (or all patterns that a string might follow), and find the shortest description in any of them.

Formally a machine is defined as a partial computable function $M : 2^{<\omega} \to 2^{<\omega}$. Essentially such a machine is just a partial computable function on $\mathbb{N}$, since we can identify each string with a natural number [Nie08, 75]. For a machine M we say that τ is an *M-description* of σ if $M(\tau) = \sigma$. We define the *plain Kolmogorov complexity* of a string σ with respect to a machine M as the length of the shortest M-description of σ [Nie08, 76]:

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

As mentioned before we are not only interested in finding the descriptive complexity of a string in one machine, but we would like to take into account *all* machines. We can do this using a *universal* machine: this is a machine N such that for each machine M, every M-desciption of a string x is reflected in N with only an added *coding constant* c_M:

$$\forall \sigma, x\, [M(\sigma) = x \to \exists \tau (N(\tau) = x \;\&\; |\tau| \leq |\sigma| + c_M)].$$

The canonical universal plain machine lists all machines M_e. Note that such an effective list exists because the machines correspond to partial computable functions on $\mathbb{N}$. Then it codes which machine to simulate by adding $e-1$ zeros and a 1 in front of each description. The coding constant of any machine is therefore just its index. For any input τ the canonical universal machine reads how many zeros precede the first 1, say $d-1$. Then if the remaining string is τ it outputs $M_d(\tau)$ [Nie08, 76].

Now we can define the *plain Kolmogorov complexity* $C(\sigma)$, which will take into account all possible descriptions in all machines, see the definition below as can be found in [Nie08]. Note that using a universal machine for this adds a constant to the descriptions in a certain machine, but for defining randomness this constant is not of any influence. Also the choice of universal machine may result in different coding constants, but since each universal machine also simulates the others, this is once again up to a constant that has no influence. In general we assume that we are dealing with the canonical universal machine as described above.

Definition 1.1 (Plain Kolmogorov complexity). *The plain complexity* $C(\sigma)$ *of a string* σ *is*

$$C(\sigma) = \min\{|\tau| : N(\tau) = \sigma\}$$

where N *is a universal machine.*

But this definition of descriptive complexity is not optimal, although it is still of interest to us. The machines describing the strings can 'cheat' by reading off information from the length of the describing string. An example of an explicit use of this length is the proof of Theorem 2.1.4 in [DH10, 114]. This is a problem, because in the Turing machines on which the definition of plain complexity is based (via partial computable functions), there is no way to mark the end of a description; the tape that goes in has no symbol for marking the end. Using the length for reducing the complexity should therefore not be possible.

Another problem with plain complexity is the following: suppose we have two strings x and y with shortest descriptions σ_x in M_x (so $M_x(\sigma_x) = x$) and σ_y in M_y (so $M_y(\sigma_y) = y$). Now we take the string that is given by concatenating x and y to $x * y$. By the way that C is defined, there is nothing we can say about the description of this new string. We would like to have a machine M_{x*y} that simulates M_x and M_y in such a way that $M_{x*y}(\sigma_x * \sigma_y) = M_x(\sigma_x) * M_y(\sigma_y)$. This would give C the property $C(x * y) \leq^+ C(x) + C(y)$, called *subadditivity*. However, we cannot just concatenate descriptions like this, because there is no way to tell where the first ends and the second starts.

To solve these matters, we define *prefix-free complexity* K. Instead of looking at all possible descriptions, it only considers machines that have a prefix-free domain, which mean that when a string is in the domain, no proper extension of it is in the domain ($\sigma \prec \tau$, then $\sigma = \tau$). This way the machine only halts when it has reached the end of a string in the domain, solving both problems mentioned above. We define prefix-free complexity in the same way as we defined the plain complexity C, except we only take machines into account that have a prefix-free domain. A universal prefix-free machine U exists [Nie08, 84], and we define

Definition 1.2 (Prefix-free Kolmogorov complexity). *The prefix-free complexity $K(x)$ of a string x is*

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

where U is a universal prefix-free machine.

Now that we formalized the idea of describing a string, we can say what it means for a string not to have a short description. Intuitively, we would say that a string is hard to describe if the only description is the string itself. Formally, we say that it *incompressible* if there is no prefix-free description shorter than the string itself [DH10, 133]:

$$K(\sigma) \geq |\sigma|.$$

We say that an infinite sequence is random if all its initial segments are hard to compress. However, if we require all initial segments to be incompressible, no sequence will be random. So instead, we base our definition of randomness on d-incompressibility [DH10, 133]:

Definition 1.3 (d-incompressibility). *A string σ is d-incompressible if*

$$K(x) > |x| - d.$$

We can fix some d and call every d-incompressible string weakly K-random. Given an infinite sequence, if all its initial segments are d-incompressible with the same constant d, we say that it is 1-random [DH10, 133].

Definition 1.4 (1-randomness). *An infinite sequence X is 1-random if there is a constant d such that every initial segment $X \upharpoonright_n$ is d-incompressible:*

$$\forall n \; K(X \upharpoonright_n) > n - d.$$

1.2 Behaviour of K

To study randomness and degrees of randomness we study prefix-free Kolmogorov complexity K and its properties. In this section we briefly discuss the behaviour of K as a function: upper bounds for K and computability of K. But first we note that there are incompressible strings of every length. This is because there are less descriptions than strings of a certain length, and each description only describes one string. We say that K satisfies the *counting condition* [Nie08, 78].

Claim (K satisfies the counting condition). For each k, strictly less than 2^k strings have a description strictly shorter than k.

Proof. A prefix-free set of strings that all have length strictly less than k, can have at most 2^{k-1} elements. So at most 2^{k-1} strings could get a description strictly shorter than k. □

Since not all 2^k strings of length k can get a shorter description, some of them must be incompressible.

Next we look at the behaviour of K as a function on natural numbers. This is very relevant for this thesis, since we often need to know the prefix-free complexity of the length (a natural number) of a string. Plotting K on natural numbers results in a graph as in Figure 1.1. It is unbounded, and also the lower bound $m(x) := \{K(y) : y \geq x\}$ goes to infinity.

This graph shows that $K(n)$ sometimes drops radically. This is because in general we can describe a number with its binary expansion. But sometimes we can describe it even shorter, for example because it is the product of two other numbers. Think for example of $n = 100$. This can be described in a very short way via a machine that takes powers of 10, and it only needs the description of 2 to output $10^2 = 100$.

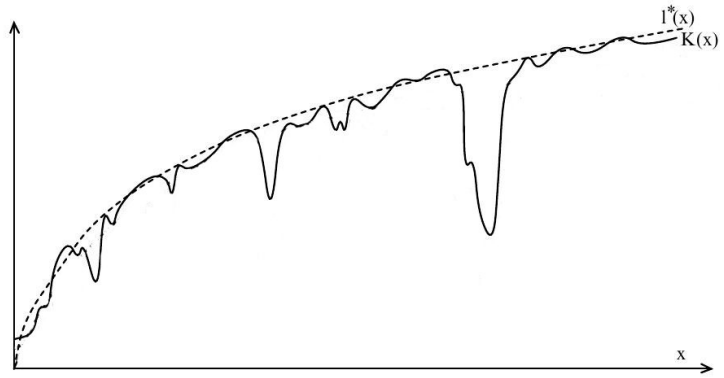


Fig. 1.1: Behaviour of K on natural numbers, adapted from [LV93, 217]

We can find an upper bound for $K(\sigma)$, depending on σ. If we build a machine M that describes each σ in a smart way, we know that all of these descriptions will be reflected in the universal machine. So the prefix-free complexity of a string must be less than the prefix-free complexity in M plus a coding constant.

Before building the required machine we first build one for the plain complexity C, to get an upper bound there. This machine can just be the identity: let $M(\sigma) := \sigma$ for all σ. Then we get that $C(\sigma) \leq^+ |\sigma|$. In fact the identity machine will have coding constant 1 in the canonical machine [Nie08, 77], so we get for all σ

$$C(\sigma) \leq |\sigma| + 1.$$

However, the identity machine is not prefix-free. For obtaining an upper bound for K, we can make the following machine: $M(0^{|\sigma|} * 1 * \sigma) = \sigma$ for all σ. This machine is prefix-free since $0^{|\sigma|} * 1 * \sigma \preceq 0^{|\tau|} * 1 * \tau$ first implies that $|\sigma| = |\tau|$ and then $\sigma = \tau$ [Nie08, 85]. This machine is reflected in the universal machine via some coding constant, so we get

$$K(\sigma) \leq^+ 2|\sigma|.$$

In fact we could make this bound a bit tighter by not coding the length of σ with $0^{|\sigma|}$, but using the shortest description of $|\sigma|$. With the same argument like before [Nie08, 85] this gives a prefix-free machine and we get

$$K(\sigma) \leq^+ K(|\sigma|) + |\sigma| \leq^+ 2\log(|\sigma|) + |\sigma|.$$

Observe that in this argument we have built our own machines to push down the prefix-free complexity K. Since $K(\sigma)$ is the minimum of all descriptions in the universal machine, any description that we force via some machine will be reflected in K via a coding constant. All we need to do is make sure that we build a prefix-free machine. The following theorem from [DH10, 126] is very useful to do this, and we use it a lot throughout this thesis:

Theorem 1.5 (Kraft-Chaitin theorem, Levin [Lev71], Schnorr [Sch73], Chaitin [Cha76]). *Let $(d_i, \tau_i)_{i\in\omega}$ be a computable sequence of pairs with $d_i \in \mathbb{N}$ and $t_i \in 2^{<\omega}$ such that $\sum_i 2^{-d_i} \leq 1$. Then there is a prefix-free machine M and strings σ_i of length d_i such that for all i*

$$M(\sigma_i) = \tau_i$$

and $dom(M) = \{\sigma_i : i \in \omega\}$. An index for M can be obtained effectively from an index for the computable sequence.

We call this computable sequence $(d_i, \tau_i)_{i\in\omega}$ with property $\sum_i 2^{-d_i} \leq 1$ a *bounded request set*. The proof of this theorem can be found in [DH10, 126]. Whenever we want to build some machine that gives short descriptions to certain strings, we make a computable sequence like in the theorem, where the first part of the pair is the length we want our machine to use to describe the second part of the pair: the string that needs a description. When this is in fact a bounded request set, we can be sure that there is a machine that gives the required descriptions.

We conclude this section with a note on the computability of K. Finding the value $K(\sigma)$ for a string σ is not computable, since we never know whether a shorter string might also describe σ. But there is a computable approximation $K[s]$ to K, making K a Δ^0_2 function. The approximation is defined as follows (see [DH10, 125]) for all σ:

$$K(\sigma)[s] = \min\{|\tau| : U(\tau)[s] \downarrow = \sigma\}.$$

In the limit this will be equal to $K(\sigma)$ since we then just take the minimum over all descriptions in U again. Furthermore by definition this approximates K from above. We use the approximation in various occasions in this thesis.

1.3 Degrees of randomness and K-triviality

After defining and investigating the prefix-free complexity K, we now turn to one of its most interesting features. As mentioned before, defining randomness via the descriptive complexity of strings gives us the opportunity to compare the degrees of randomness of different sets. We do this by comparing the complexity of their initial segments, and we say that a set A is K-below B (or K-reducible to B) if each initial segment of A has lower complexity than the initial segment of B of the same length, up to a constant. This concept of K-reducibility was only introduced quite recently by Downey, Hirschfeldt and LaForte in their 2004 paper [DHL04]. Formally, the definition is the following:

Definition 1.6 (K-reducibility). *A is K-below B ($A \leq_K B$) iff for all n,*

$$K(A \upharpoonright_n) \leq^+ K(B \upharpoonright_n).$$

This gives us a structure of different levels of randomness, the K-degrees. Sets in the lowest degrees are the K-trivial sets. We define them to be the sets of which all initial segments are as easy to describe as their length, up to a constant:

Definition 1.7 (K-trivial set). *A is K-trivial if for all n,*

$$K(A \upharpoonright_n) \leq^+ K(n).$$

Saying that a segment is as easy to describe as its length is the same as saying that it is as easy to describe as a string of zeros of that length. Therefore a set being K-trivial coincides with a set being K-below $\emptyset$:

Note 1.8. A is K-trivial iff $A \leq_K \emptyset$.

The above definition of K-triviality requires all initial segments to have a short description. The theorem below, taken from [DH10, 503], shows that it is sufficient if for all n in some infinite computable set S, $K(A \upharpoonright_n) \leq^+ K(n)$. Also then A is K-trivial. Moreover we can show something stronger: S merely needs to be computably enumerable (c.e.). The latter is not mentioned anywhere in the literature, to our knowledge. We show it as a corollary to the following theorem that can be found in [DH10].

Theorem 1.9 (Downey, Hirschfeldt [DH10]). *If $K(A \upharpoonright_n) \leq^+ K(n)$ holds for all $n \in S$ for some infinite computable set S, then A is K-trivial.*

Proof. Since S is computable, there is a total computable function $h(n)$ with range S that lists all elements in their natural order. Let M be the machine that on input σ applies the universal machine U and then the function h. This way, a description σ of n in U (so $U(\sigma) = n$) is a description of $h(n)$ in M (since $M(\sigma) = h(U(\sigma)) = h(n)$). So $K(h(n)) \leq^+ K(n)$.

Observe that given $h(n)$ we can retrieve n (look at how many numbers precede $h(n)$ in the set S). If we have the initial segment $A \upharpoonright_{h(n)}$, we can then find n from its length, and since $n \leq h(n)$ always holds, we can find the shorter segment $A \upharpoonright_n$. So for all n,

$$K(A \upharpoonright_n) \leq^+ K(A \upharpoonright_{h(n)}).$$

Since $h(n) \in S$ for all n we have by assumption that $K(A \upharpoonright_{h(n)}) \leq^+ K(h(n))$ for all n, so with $K(h(n)) \leq^+ K(n)$ and the previous it follows that for all n

$$K(A \upharpoonright_n) \leq^+ K(n).$$

□

To get the result for c.e. sets, we refer to Lemma 1.7.4 from [LV93, 33]. This says that every infinite c.e. set has an infinite computable subset.

Corollary 1.10. *If $K(A \upharpoonright_n) \leq^+ K(n)$ holds for all $n \in S$ for some infinite c.e. set S, then A is K-trivial.*

Proof. Take some infinite c.e. set S. By Lemma 1.7.4 from [LV93, 33] there is an infinite computable subset $S' \subseteq S$. For this infinite computable set we also have for all $n \in S'$ that $K(A \upharpoonright_n) \leq^+ K(n)$. By Theorem 1.9 it follows that A is K-trivial. This concludes the proof. □

Intuitively the results in this section can be explained by realizing that the initial segment of length n can be found from the initial segment of length $n+1$. Therefore, once we know just where the set has low complexity (because the set of n for which this holds is computable or even c.e.), we can use this to our advantage to get short descriptions of the preceding initial segments.

1.3.1 Computability of K-trivial sets

One may wonder how complex a K-trivial set can be. Each computable set is K-trivial, since we can just use the program that computes it to describe the initial segments. K-triviality does not imply computability: below we will construct a K-trivial set that is not computable. However, K-trivial sets must always be Δ^0_2, as we will also show below.

But first we briefly look at a similar notion for C. Chaitin proved that sets A such that $C(A \upharpoonright_n) \leq^+ C(n)$ for all n are computable. This is a theorem Chaitin first published in his 1976 article [Cha76], it can be found in [DH10, 120] in the following form:

Theorem 1.11 (Chaitin [Cha76]). *A set A is computable if and only if $C(A\upharpoonright_n) \leq^+ C(n)$. Furthermore, for each k there is are only $\mathcal{O}(2^d)$ many A such that $C(A\upharpoonright_n) \leq C(n) + d$.*

We will omit the proof here, it can be found in [DH10]. As mentioned above such a theorem does not hold for K, showing a difference between definability in the C- and K-degrees. In the proof of the following theorem we construct a K-trivial set that is not computable. This proof is originally due to Solovay in some unpublished notes on Chaitin's work. Zambella [Zam90] adapted it into the proof as presented below.

Theorem 1.12 (Zambella [Zam90], after Solovay (unpubl. notes)). *There is a noncomputable K-trivial c.e. set.*

Proof. To make the set noncomputable, we make sure it is a *simple set*. Recall by definition a simple set is c.e., and its complement is infinite and contains no infinite c.e. set. Therefore, if we make an infinite simple set we can be sure that it is noncomputable.

Furthermore we will make sure we get a K-trivial set A by simultaneously building a request set L for describing initial segments of A. We base these descriptions on those of n in the universal machine, making sure that we only put numbers into A that will not add too much weight to the request set.

To make a simple set we satisfy the following requirements for all e, where $(W_e)_{e\in\omega}$ is an effective list of all c.e. sets:

$$R_e: \ W_e \text{ is infinite} \Rightarrow \exists x(x > 2e \wedge x \in W_e \cap A).$$

This makes sure that each infinite c.e. set cannot be in the complement of e entirely.

We build A as follows: at stage s we search for the least e such that $W_{e,s}\cap A$ is empty (we say the requirement R_e requires attention), we search for a number $n > 2e$ such that it is in the e-th c.e. set at this stage ($n \in W_{e,s}$) and it does not add more than $2^{-(e+2)}$ weight to describe initial segments of A again with the descriptions available at this stage ($\sum_{n\leq l\leq s} 2^{-K(l)[s]} < 2^{-(e+2)}$). We put this number into A.

Note that if W_e is indeed infinite, we will eventually be able to find such a number and put it into A, since the sum $\sum_{n\leq l\leq s} 2^{-K(l)[s]}$ will become small enough after some n. This way whenever W_e is infinite, we can be sure that $\exists x(x > 2e \wedge x \in W_e \cap A)$, hence the requirements will be satisfied. Also we know that A will be infinite, since we keep putting more numbers into it. To summarize, A will be the following set:

$$A := \{n : \exists e \exists s(W_{e,s}\cap A[s] = \emptyset \wedge n > 2e \wedge n \in W_{e,s} \wedge \sum_{n\leq l\leq s} 2^{-K(l)[s]} < 2^{-(e+2)})\}.$$

We let the request set L be as follows: whenever at some stage there is some shorter description σ for a number n (so $U(\sigma) = n$), shorter than any we

have previously seen, we enumerate $\langle |\sigma|+1, A[s]\upharpoonright_n\rangle$ into L. Also when $A\upharpoonright_n$ has changed since the last stage (by putting some number in), we add the same request to L. By describing the segments with length $|\sigma|+1$, we keep the weight of describing them once below half the weight of the universal machine. Some more weight is added because some new descriptions might be required whenever A changes again. But because of the restraint we put on the numbers we enumerate into A (namely $\sum_{n\leq l\leq s} 2^{-K(l)[s]} < 2^{-(e+2)}$), we know that this is bounded by $\sum_e 2^{-(e+2)}$. This is because once a number is put into W_e, it stays in there. When we then make sure that the intersection with A is nonempty, it will never be empty after that stage. So we only need to put at most one number into A for each e. Therefore the weight of L is bounded by

$$\sum_{\mathrm{dom}(U)} 2^{|\sigma|+1} + \sum_e 2^{-(e+2)} \leq \frac{1}{2} + \frac{1}{2} = 1.$$

We apply the Kraft-Chaitin theorem to get a machine M that describes $A\upharpoonright_n$ with a string of length $K(n)+1$. So we certainly have for all n

$$K(A\upharpoonright_n) \leq^+ K(n)$$

and A is K-trivial. Note that A is c.e. by construction and noncomputable because it is an infinite simple set. □

The above theorem implies that a K-trivial set is not necessarily computable. However, we do have a bound on the arithmetical complexity: every K-trivial set is Δ^0_2, as follows from the theorem below. This theorem uses the Coding Theorem, as can be found in [Nie08, 91]. For this we define the probability that a prefix-free machine M outputs a string x:

$$P_M(x) = \sum_{M(\sigma)=x} 2^{-|\sigma|}.$$

The Coding Theorem says the following:

Theorem 1.13 (Coding Theorem, Solomonoff [Sol64a] [Sol64b], Levin [Lev71] [Lev73], Chaitin [Cha75]). *From a prefix-free machine M, we may effectively obtain a constant c such that for all x, $2^c 2^{-K(x)} > P_M(x)$.*

We omit the proof here (it can be found in [Nie08]) and we go straight to the theorem proving that each K-trivial set is Δ^0_2. In this proof, for each e we look at the set of sets that are K-trivial via this constant:

$$\mathrm{KT}(e) := \{X : \forall n\ K(X\upharpoonright_n) \leq K(n) + e\}.$$

For each e this set is finite. This is the first item in the theorem below, and was originally presented as a theorem in Chaitins 1976 paper [Cha76]. Using this, for each e we can list all sets in $\mathrm{KT}(e)$ in a $\emptyset'$-computable tree. That tree

then only has finitely many infinite paths (part 1 of the theorem), which shows that all of them will be $\emptyset'$-computable (part 2 of the theorem). This argument is made more formal in the proof below, as can be found in [Nie08, 178].

Theorem 1.14 (Chaitin [Cha76]).

(1) There is a constant $c \in \mathbb{N}$ such that for each b, at most 2^{c+b} sets are K-trivial with constant b.
(2) Each K-trivial set is Δ^0_2.

Proof. For item (1), let c be a constant such that

$$\forall d \in \mathbb{N}\, \forall n\, \#\{x : |x| = n \,\&\, K(x) \leq n + K(n) - d\} < 2^c 2^{n-d}.$$

This exists because of the coding theorem: let M be the prefix-free machine that for input σ outputs the length of what σ describes in the universal machine ($M(\sigma) = |U(\sigma)|$). By the coding theorem there is a constant c such that for all n,

$$2^c 2^{-K(n)} > P_M(n).$$

For arbitrary d and n, take any x such that $|x| = n$ and $K(x) \leq n + K(n) - d$. The shortest description of x in the universal machine will be a description of n in M, so it adds at least $2^{-n-K(n)+d}$ to $P_M(n)$. Suppose $2^c 2^{n-d}$ would *not* bound the number of such x, then there would be at least 2^{n+c-d} and each of them can add some to $P_M(n)$. So

$$P_M(n) \geq 2^{n+c-d} 2^{-n-K(n)+d} = 2^c 2^{-K(n)}.$$

But by construction, $P_M(n) < 2^c 2^{-K(n)}$, so this is a contradiction. Therefore this c is the required constant.

Now observe that if we let $d = n - b$ for some arbitrary b, we get that for this constant c,

$$\forall b \in \mathbb{N}\, \forall n\, \#\{x : |x| = n \,\&\, K(x) \leq K(n) + b\} < 2^c 2^b.$$

This implies that there can be at most 2^{c+b} K-trivial sets via constant b.

For item (2), take some arbitrary b. Define the tree consisting of strings of complexity less than their length plus this constant b:

$$T_b = \{s : \forall n \leq |s| [K(s \upharpoonright_n) \leq K(n) + b]\}.$$

The infinite paths of this tree will be precisely the sets that are K-trivial via constant b. By item (1) at each level of the tree there are at most 2^{c+b} nodes, so there are finitely many infinite paths. Furthermore the tree is $\emptyset'$-computable, since finding K is. Each infinite path is therefore $\emptyset'$-computable (see Appendix A). So each K-trivial set is Δ^0_2, which concludes the proof. □

We have seen that K-trivial sets are not necessarily computable, but they are always Δ^0_2. Some more properties of K-trivial sets are known. For example, a set is K-trivial if and only if in is low for K (or equivalently low for ML-randomness [Nie08, 170]). And K-trivial sets are superlow, weak truth-table incomplete and Turing incomplete [Nie08, 177]. These results all build on a method called the *decanter method*, which gives a way to construct non-K-trivial sets. We treat this method in detail in Appendix C, and we use it in Section 4.4.

To summarize, this chapter has given us the foundations for the rest of this thesis by defining (prefix-free) Kolmogorov complexity, 1-randomness and K-reducibility. We have discussed some of the properties of K and K-trivial sets, and some theorems that we need in the following chapters.

2

Infinitely often K-trivial sets

In the previous chapter we have taken some time to study K-trivial sets and their behaviour. In this chapter we introduce a new concept: that of infinitely often K-trivial sets. As the name shows they are similar to K-trivial sets, but we only require infinitely many initial segments to be of low complexity, instead of all.

The notion of infinitely often K-trivial sets turns out to be very useful for studying arithmetical definability in the K-degrees. It will come back more than once in the proofs in the next chapter. Using infinitely often K-trivial sets, we have been able to construct a minimal pair of lower complexity than what was known so far (see the next chapter for this).

In the current chapter we formally define these infinitely often K-trivial sets. We study their behaviour, showing that each c.e. set is infinitely often K-trivial, which is a crucial feature for constructing the minimal pair in the next chapter. Furthermore we will see that infinitely often K-trivial sets behave nicely in the sense that their lower cones are countable, and every set in the lower cone is computable from it and the halting problem.

We also show that there are uncountably many infinitely often K-trivial sets, and we construct a Π^0_1 class of infinitely often K-trivial sets that are non-K-trivial, which allows us to apply basis theorems. Finally we show that infinitely often K-trivial sets can be found in various places: there is one in each truth-table degree (this was implicit in [MS07], but is now proved using different methods) and every 1-generic set is infinitely often K-trivial. All definitions and proofs in this chapter can be found in our paper [BV10] as well.

2.1 Infinitely often K-trivial sets

Based on the definition of K-trivial sets, we introduce the notion of infinitely often K-trivial sets. The idea to define a more general notion this way is inspired on Millers work, who introduced weakly low for K-sets based on low

for K sets [Mil09]. This turned out to be very useful for proving some new results.

A set is low for K if having it as an oracle does not help in compressing strings: $\forall\sigma\ K^A(\sigma) =^+ K(\sigma)$. Nies proved this to be equivalent to K-triviality [Nie05]. Miller defined a set to be weakly low for K if the compression is no better for infinitely many strings: $\exists^\infty\sigma\ K^A(\sigma) =^+ K(\sigma)$. This concept can for example be used to characterize lowness for Ω.

In a similar fashion we introduce infinitely often K-trivial sets. We require infinitely many initial segments σ to have low complexity $K(\sigma) \leq^+ K(|\sigma|)$ as oppose to *all* initial segments.

Definition 2.1 (Infinitely often K-trivial set). *A set A is K-trivial on a set $M \subseteq \mathbb{N}$ with constant c if for all $n \in M$, $K(A\upharpoonright_n) \leq K(n) + c$. If M is infinite, we say that A is infinitely often K-trivial.*

With Theorem 1.10 it follows that when a set is K-trivial on an infinite c.e. set M, it is K-trivial. Another fundamental result is that every c.e. set is infinitely often K-trivial. This is because when enumerating the elements of a c.e. set, there will be infinitely many stages in which some number n is enumerated and after that stage the initial segment up to n will not change. We call these the *true stages*, and we prove this claim in the following lemma.

Lemma 2.2. *Given an infinite c.e. set X and an approximation $X[s]$, there are infinitely many stages s such that a number n is enumerated into X at this stage, and $X[s]\upharpoonright_n = X\upharpoonright_n$.*

Proof. Without loss of generality, assume that at each stage precisely one number is enumerated into X. Intuitively, the claim holds because when at stage s_1 a number n is enumerated into X, the only way to change this initial segment is to enumerate a number $< n$ into X. That can happen only finitely often. We prove that infinitely often we enumerate a number n into X, and the initial segment up to $n-1$ was already finished.

Suppose at most finitely often an n was enumerated and the initial segment up to n is finished. Then after some stage the initial segments are never done: there is some stage s_0 such that for all n that are enumerated into X at stages $s > s_0$, there is a stage $s' > s$ such that $X[s']\upharpoonright_n \neq X[s]\upharpoonright_n$. Take a stage $s_1 > s_0$, where n_1 is enumerated into X. As said, the only way to make sure that $X[s_1]\upharpoonright_{n_1} \neq X\upharpoonright_{n_1}$, is to enumerate a number $n_2 < n_1$ into X at stage $s_2 > s_1$. But by assumption, also this initial segment $X[s_2]\upharpoonright_{n_2}$ will change again at a later stage. So there is an $n_3 < n_2$ enumerated into X at stage s_3 such that $X[s_3]\upharpoonright_{n_2} \neq X[s_2]\upharpoonright_{n_2}$. We can continue this argument to get an infinite descending chain $n_1 > n_2 > n_3...$, which is a contradiction. This proves the lemma. □

We use this lemma to prove that every c.e. set is infinitely often K-trivial. We do this by giving initial segments up to n short descriptions at the very

stage that n is enumerated into the set. Infinitely often it will be the case that that initial segment is final, so infinitely often have we been able to give short descriptions for the final set X. This idea was used to prove a similar claim for the plain Kolmogorov complexity in [HKM09]. It seems that a number of researchers are aware of the result in the theorem below, but we have not found any explicit reference in the literature.

Theorem 2.3. *Every c.e. set is infinitely often K-trivial.*

Proof. Take an arbitrary c.e. set X. If the set is finite it is computable so it is infinitely often K-trivial. Let X be infinite. Without loss of generality, assume that at each stage precisely one new number n is enumerated into X. We build a machine M as follows: at stage s, when n is enumerated into X at this stage, M describes $X[s] \upharpoonright_n$ with all descriptions of n in the universal machine. This machine exists since we encounter each n only once and the weight of all descriptions of natural numbers in the universal machine is bounded.

By Lemma 2.2, there are infinitely many stages s such that a number n is enumerated into X and $X[s] \upharpoonright_n = X \upharpoonright_n$. At each such stage we have enumerated a short description of the initial segment $X[s] \upharpoonright_n = X \upharpoonright_n$ into M, so we get $K_M(X \upharpoonright_n) \leq K(n)$ for infinitely many n and

$$K(X \upharpoonright_n) \leq^+ K(n)$$

for infinitely many n. This proves the theorem. □

Theorem 2.3 and the next proposition will be crucial in the construction of a minimal pair of a Σ^0_2 set with any non-K-trivial c.e. set in the next chapter. In the proposition below we show that when a set Y is infinitely often K-trivial, every set in the lower cone is computable in Y plus the halting problem.

Proposition 2.4. *Suppose Y is infinitely often K-trivial. Then every set X in $\{X : X \leq_K Y\}$ is computable in $Y \oplus \emptyset'$.*

Proof. Let Y be K-trivial on the infinite set M, via constant c_0. Let X be a set such that $X \leq_K Y$ via constant c_1. Let $c = c_0 + c_1$. Observe that the set M on which Y is K-trivial is computable from Y and the halting problem, since K is computable from the halting problem. That means from $Y \oplus \emptyset'$ we can compute a set of strings σ that are of length some n in M and for each σ, $K(\sigma) \leq K(|\sigma|) + c$.

First we define $F_c(n) := \{\sigma : |\sigma| = n \wedge K(\sigma) \leq K(|\sigma|) + c\}$. With the Coding Theorem (see Chapter 1) we get that there is a constant b such that

$$|F_c(n)| \leq 2^{b+c}$$

for all n. If we take the downward closure of the union of these sets for all n, we get a tree where the width is bounded by this 2^{b+c} so all infinite paths are isolated. In particular we can define L_c to be the downward closure of

$\bigcup_{n\in M} F_c(n)$ (now we only take the union over $n \in M$). This tree is computable from M, hence from $Y \oplus \emptyset'$. Once again it is of bounded width, so all paths are isolated. That means that they are all computable from $Y \oplus \emptyset'$ as well. Finally, note that X is an infinite path through this tree L_c. So X is computable from $Y \oplus \emptyset'$. □

Note that the proof of the above proposition could have been mended into a proof that showed the following:

Remark 2.5. Suppose Y is K-trivial on an infinite set M. Then any set X in $\{X : X \leq_K Y\}$ is computable in $M \oplus \emptyset'$.

Furthermore Proposition 2.4 allows us to derive a nice result for the specific case of c.e. sets. Since they are always infinitely often K-trivial, any set X that is K-below a c.e. set Y is computable in $Y \oplus \emptyset'$. Therefore any set below a c.e. set must be Δ^0_2. This is an important observation that will come back in the next chapter. But first we take some time to study infinitely often K-trivial sets.

2.2 Uncountably many infinitely often K-trivial sets

Infinitely often K-trivial sets are rather common. There are uncountably many, and there are uncountably many that are non-K-trivial. In this section we construct the perfect trees that prove this. The next section will treat the question of *where* all these infinitely often K-trivial sets occur.

To construct uncountably many infinitely often K-trivial sets, we build a perfect tree such that all infinite paths are infinitely often K-trivial. A perfect tree is a tree T such that for every $\sigma \in T$, there are at least two proper extensions $\tau_1, \tau_2 \in T$ that are incomparable (see Appendix A). Since a perfect tree has uncountably many infinite paths, a perfect tree with infinitely often K-trivial sets as infinite paths will prove that there are uncountably many infinitely often K-trivial sets.

To construct the required tree it is convenient to use the following lemma:

Lemma 2.6. *If V is an infinite c.e. set of strings such that for all $n \in \mathbb{N}$ there is at most one string of length n in V, then for all $\sigma \in V$ we have $K(\sigma) \leq^+ K(|\sigma|)$.*

Proof. Take some set V with properties as in the lemma. Since V is c.e. we can go through all its elements. We build a machine M that for each $\sigma \in V$ takes all descriptions of $|\sigma|$ in the universal machine as descriptions of σ. Since we have only one string of each length $n \in \mathbb{N}$, we can be sure that the weight of all descriptions is bounded by the weight of this universal machine. Therefore, this machine M exists, and we get $K_M(\sigma) \leq K(|\sigma|)$, and $K(\sigma) \leq^+ K(|\sigma|)$. □

Using this lemma we construct a perfect tree with infinitely often K-trivial sets as infinite paths. The following theorem states that we can find such a tree to be computable. The same proof can be found much shorter in our paper [BV10].

Theorem 2.7. *There is a computable perfect tree such that the infinite paths through this tree are infinitely often K-trivial sets.*

Proof. Intuitively, the idea is the following: we build a perfect tree T that consists of strings that all have low complexity ($\leq^+ K(n)$ for a string of length n). An infinite path in this tree will go through infinitely many nodes, so it will be infinitely often K-trivial by construction. In order to make all nodes have low complexity, we make sure that we put at most one string of length n into T for each $n \in \mathbb{N}$. That way we get a c.e. set of strings as in Lemma 2.6, which gives us the required low complexity.

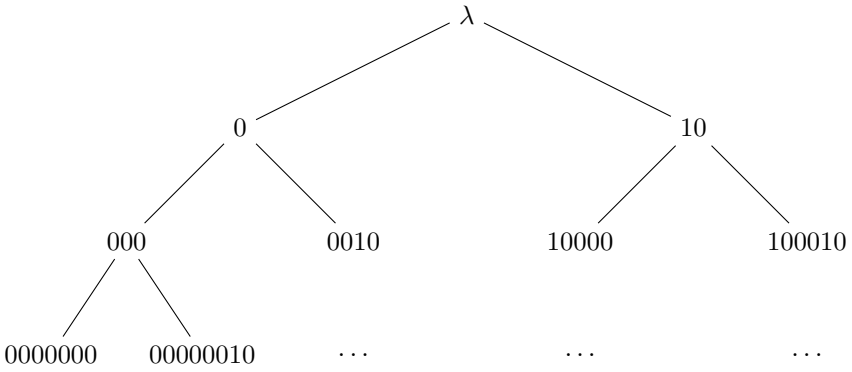
To construct the required tree, we view it as a partial function from strings to strings $T : 2^{<\omega} \to 2^{<\omega}$ (see Appendix A) such that

$$\sigma \prec \tau \Rightarrow T(\sigma) \prec T(\tau)$$
$$\sigma | \tau \Rightarrow T(\sigma) | T(\tau).$$

We build the tree T as follows: pick the root of T to be the empty string ($T(\lambda) = \lambda$). Then inductively define $T(\tau * 0)$ and $T(\tau * 1)$ assuming that $T(\tau)$ was already defined:

- let $T(\tau * 0) = \sigma$ where σ is the first string (in the length-lexicographical ordering) such that it extends $T(\tau)$ and it has a length longer than any string in the tree so far.
- let $T(\tau * 1) = \sigma$ where σ is the first string such that it extends $T(\tau)$ and is incomparable to $T(\tau * 0)$. Also make sure $|T(\tau * 1)| > |T(\tau * 0)|$.

Note that this algorithm will give us the following first few nodes in T:



Since we can always find strings σ to map $T(b)$ to, this tree is perfect. Since the range of T is a c.e. set with at most one string of each length, for each b, $K(T(b)) \leq^+ K(|T(b)|)$ by Lemma 2.6.

Take some infinite path P in the tree. For infinitely many n, $P \upharpoonright_n = T(b)$ for some b. By construction $K(T(b)) \leq^+ K(|T(b)|)$. So it follows that for all these infinitely many n, $K(P \upharpoonright_n) \leq^+ K(n)$. So each infinite path in this tree is infinitely often K-trivial. Also T was made using a computable algorithm, so T is computable. □

From this theorem we immediately get the following result:

Corollary 2.8. *There are uncountably many infinitely often K-trivial sets.*

Note that since we have constructed the tree to be computable, we can use it to show some more properties of infinitely often K-trivial sets. For example, every set A is computable from some infinitely often K-trivial B, so $A \leq_T B$. In fact, they will be Turing-equivalent, and even truth-table equivalent. We will get back to this in Section 2.3.

First we adapt the construction the the proof above into a construction of a tree that does not have any K-trivial sets as infinite paths. We do this by adding strings of high complexity on even levels. However, in the construction we will need to take strings out in a later stage when they turn out to be of low complexity after all. This makes the tree only Π^0_1 instead of computable. From this we can get a computable tree with the same infinite paths by Lemma A.6 in Appendix A, and thus get a Π^0_1 class of infinitely often K-trivial but non-K-trivial sets. Then we can apply basis theorems to get results such as there being an infinitely often K-trivial but non-K-trivial set that is low. The following theorem and its proof can be found in a shorter version in our paper [BV10].

Theorem 2.9. *There is a computable perfect tree whose infinite paths are infinitely often K-trivial but non-K-trivial sets.*

Proof. We construct a Π^0_1 perfect tree T such that strings σ at level e satisfy the following requirement:

$$R_e : \begin{cases} K(\sigma) \leq^+ K(|\sigma|) & \text{if } e \text{ is odd} \\ K(\sigma) > K(|\sigma|) + e & \text{if } e \text{ is even.} \end{cases}$$

Again we view T as a partial function mapping strings to strings. We make the tree Π^0_1 and then apply Lemma A.6, to obtain the required computable perfect tree.

Similar to the previous construction we will make sure on odd levels to only put one string for each length n into the tree, so that by Lemma 2.6 they are all of low complexity. On the even stages we then need to make sure to find strings that are of high complexity ($> K(n) + e$). We can do this by searching for a length l such that we are guaranteed that the number of strings extending the given node with length l is higher than the number of short descriptions. We can do this computably (see lemma below). Then we put *all* extensions of this length l into the tree, and at the next level for each

one of them we define two extensions that are of low complexity. When at a later stage some string at an even level turns out to have a short description (which we find through the computable approximation of K), we take it, and all its successors, out of the tree. This is why we need a Π^0_1 construction, as oppose to a computable construction.

Lemma 2.10. *There is a computable function $f(k,e) : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ such that given a string σ of length k, for an arbitrary e there is at least one string τ such that $\sigma \prec \tau$, $|\tau| = k + f(k,e)$ and $K(\tau) > 2\log(|\tau|) + c + e$ where c is a constant*[1] *such that for all n, $K(n) \leq 2\log(n) + c$.*

Proof (of lemma). First observe that there exists a computable function[2] $g(e)$ such that for all e, $g(e) > 2\log(g(e)) + c + e$. We adapt this function to some $f(k,e)$ such that $f(k,e) > 2\log(k + f(k,e)) + c + e$.

Given a computable function with this property, some σ and an arbitrary e. Take all τ such that $\sigma \prec \tau$, $|\tau| = |\sigma| + f(|\sigma|, e)$ and suppose for all of these τ, $K(\tau) \leq 2\log(|\tau|) + c + e$. Observe that if we extend σ with $f(|\sigma|, e)$ bits, there are $2^{f(|\sigma|,e)}$ ways to do so, so this is the number of different τ's. We assumed them all to have a short description: $K(\tau) \leq 2\log(|\tau|) + c + e$. Each adds a weight of at least $2^{-2\log(|\tau|)-c-e}$ to the machine, which gives a total weight of

$$2^{f(|\sigma|,e)} \cdot 2^{-2\log(|\tau|)-c-e} = 2^{f(|\sigma|,e)-2\log(|\sigma|+f(|\sigma|,e))-c-e}$$

which is by definition of f larger than 1, giving a contradiction. So there is at least some τ of the required complexity. □

Now we know enough to describe an algorithm to make the required tree T. We start with the empty string as the root node ($T(\lambda) = \lambda$), and set $n = 0$. Then define $T(\tau*0)$ and $T(\tau*1)$ assuming that $T(\tau)$ was defined. At each step we increase n with 1 to keep track of which requirement we want to satisfy:

- if n is even, define it like before: let $T(\tau * 0) = \sigma$ where σ is the first string (in the length-lexicographical ordering) such that it extends $T(\tau)$ and it has a length longer than any string in the tree so far. Let $T(\tau * 1) = \sigma$ where σ is the first string such that it extends $T(\tau)$ and is incomparable to $T(\tau * 0)$. Also make sure $|T(\tau * 1)| > |T(\tau * 0)|$. Put $n = n + 1$.
- if n is odd, let $l = f(|\tau|, n+1)$ and for all $\upsilon \in 2^{\leq l}$ define $T(\tau*\upsilon) = T(\tau)*\upsilon$. Put $n = n + 1$, and define a set $H_n = \{T(\tau) * \upsilon | \upsilon \in 2^l\}$ (later on we will need this to check whether strings have complexity high enough).

[1] Note that this exists because $2\log(n)$ is an upper bound for $K(n)$ (we can make a machine that gives each n a description of length $2\log(n)$, because the weight of this machine will be bounded). We define $\log(n)$ as the largest number k such that $2^k \leq n$.

[2] For example, $g(e) = 2^e$ will after some large enough e satisfy the equation. For the finitely many e that do not satisfy it, we can have the value of g be some large enough constant.

- for all odd $k \leq n$, check for all strings $\zeta \in H_k$ whether $K(\zeta)[s] \leq 2\log(|\zeta|) + c + k + 1$. If this is the case, remove this string and all its extensions from the tree and from H_k.

Note that by Lemma 2.10 above, we can be sure that not all strings will be removed. Therefore this will be a perfect tree (since for even n it always splits). Furthermore by construction it is a Π^0_1 tree, and all infinite paths are infinitely often K-trivial sets, that are non-K-trivial. By applying Lemma A.6, we get a perfect computable tree with the same infinite paths. □

From the theorem above we get that there is a Π^0_1 class of infinitely often K-trivial sets such that they are all non-K-trivial. Using basis theorems we get a number of results from this. For example, there is an infinitely often K-trivial but non-K-trivial set that is low (with Theorem 1.19.8 in [DH10, 77]), and one that is superlow (Corollary 1.19.9 in [DH10, 78]).

Furthermore, there are non-K-trivial (so noncomputable) infinitely often K-trivial sets that are computably dominated (with Theorem 1.8.42 from [Nie08, 59]). This is interesting considering that when a computably dominated set is K-trivial (and hence Δ^0_2), it is also computable [Nie08, 163].

2.3 Occurrence of infinitely often K-trivial sets

In the previous section we saw how common infinitely often K-trivial sets really are: there are uncountably many. One might wonder where these can be encountered, which is the subject of this section. We note that each truth-table degree contains an infinitely often K-trivial set. This is implicitly in [MS07], but our proof is obtained using different methods. We also find that each weakly 1-generic set is infinitely often K-trivial. We start with a result that follows from the proof of Theorem 2.7 above: every set is computable from an infinitely often K-trivial set.

Proposition 2.11. *Every set A is computable from some infinitely often K-trivial set B.*

Proof. Take an arbitrary set A. The way we code A into something that is infinitely often K-trivial is the following: take the tree as constructed in the proof of Theorem 2.7. Start at the rootnode. Whenever a digit of A is 0, take the leftmost immediate successor, when it is 1 take the rightmost. Let B be the resulting path through the tree. We can retrieve A from this as follows: to find $A(n+1)$ given $A(n)$, we find the path in the tree that corresponds to the first n digits like described above. Then we find out whether B extends the leftmost successor of this, in which case $A(n+1)$ is 0, or the rightmost in which case $A(n+1)$ is 1. Since T is total, precisely one of these two cases will hold.

Observe that B is infinitely often K-trivial since it is a path in the tree. And A is computable from B:

$$A \leq_T B.$$

This concludes the proof. □

If we look closely at the proof of the above proposition, we can see that the reduction is not just a Turing-reduction, but a truth-table reduction. Furthermore it is an equivalence, so we get the following theorem (as mentioned above this result is also in [MS07] and in our paper [BV10]):

Theorem 2.12. *Every set A is truth-table equivalent to an infinitely often K-trivial set B.*

Proof. Take an arbitrary set A. We define B as in the proof of the proposition above, which means that B is computed by the following Turing functional with oracle A:

$$\Phi_e^A(n) = T(A \upharpoonright_n).$$

Since this Turing functional is in fact total with any oracle, this is a truth-table reduction (by [Nie08, 14]). Also, we can define another Turing functional to retrieve A from B. To find the $n+1$st digit of A we check if $T(A \upharpoonright_n 0) \prec B$ or $T(A \upharpoonright_n 1) \prec B$. Formally, we define (starting with $\Psi^B(0) = \lambda$ the empty string):

$$\Psi^B(n+1) \simeq \begin{cases} 0 & \text{if } \exists k \leq \sum_{i\geq 1}^{n+1} 2^i [\Phi^A(\Psi^B(n) * 0) = B \upharpoonright_k] \\ 1 & \text{if } \exists k \leq \sum_{i\geq 1}^{n+1} 2^i [\Phi^A(\Psi^B(n) * 1) = B \upharpoonright_k]. \end{cases}$$

Note that the way we constructed T bounds our search in the initial segment of B. This Turing functional is once again total for any B. So again, this is a truth-table reduction: $A \leq_{tt} B$. So we get

$$A \equiv_{tt} B.$$

□

This theorem shows that any truth-table degree contains an infinitely often K-trivial set. Furthermore, truth-table reduction implies weak truth-table and Turing reduction, so we have similar results there. In particular, every Turing degree contains an infinitely often K-trivial set.

Another result concerning the occurence of infinitely often K-trivial sets is that all weakly 1-generic sets are infinitely often K-trivial. The definition of a weakly 1-generic set relies on that of a dense set of strings. Both can be found in [DH10, 105].

Definition 2.13 (Dense set of strings). *A set $S \subseteq 2^{<\omega}$ is dense if every string in S has a proper extension in the set: for every $\sigma \in S$ there is some $\tau \in S$ such that $\sigma \prec \tau$.*

Definition 2.14 (Weakly n-generic set). *A set X is weakly n-generic if for every dense Σ^0_n set S there is some initial segment of X that is in S ($X\upharpoonright_n \in S$ for some n).*

We can prove that every weakly 1-generic is infinitely often K-trivial using a dense Σ^0_1 set such that all strings in it are of low complexity ($\leq^+ K(n)$ for a string of length n). Infinitely many initial segments of a weakly 1-generic set X will intersect it, since a cofinite subset of a dense set is still dense[3]. It follows immediately that X is infinitely often K-trivial. This theorem is also in our paper [BV10].

Theorem 2.15. *Every weakly 1-generic set is infinitely often K-trivial.*

Proof. We construct a dense Σ^0_1 set S. Using Lemma 2.6 again, we make sure that for each n, we put only one string of length n into S, thus making sure that they are all of low complexity.

We construct the sets inductively as follows:

- Base case: put a string of length 1 into S
- Step $i+1$: put a string of length $i+1$ into S such that it extends all strings currently in S.

By construction we go through each length m only once, so S is as in Lemma 2.6. Therefore for each $\sigma \in S$, $K(\sigma) \leq^+ K(|\sigma|)$. S is also dense by construction, and since the above construction is a computable enumeration the set is Σ^0_1.

Now take some weakly 1-generic set X. It follows that there are infinitely many n such that $X\upharpoonright_n \in S$. For each of these, we have $K(X\upharpoonright_n) \leq^+ K(n)$. In other words, X is infinitely often K-trivial. □

Another result (that is also in our paper [BV10]) is that whenever a set does not compute any diagonally noncomputable function (DNC), then it must be infinitely often K-trivial. The definition of a diagonally noncomputable function is the following:

Definition 2.16 (DNC function). *A function f is diagonally noncomputable (DNC) if $f(e) \neq \phi_e(e)$ for all e such that $\phi_e(e)\downarrow$.*

Theorem 2.17. *If a set A does not compute a DNC function then it is infinitely often K-trivial.*

Proof. Take some A that does not compute a DNC function. Let f be the function that does the following[4]:

[3] Suppose X is 1-generic. Take some dense Σ^0_1 set S. Then there is an n such that $X\upharpoonright_n \in S$. But $S\backslash\{X\upharpoonright_n\}$ is still a dense Σ^0_1 set, so there must be another intersection. Continuing this argument we get that there must be infinitely many intersections.

[4] $\langle\sigma\rangle$ is some coding of strings into numbers

$$f : n \mapsto \langle A \upharpoonright_n \rangle.$$

This function f is computable from A, so it cannot be a DNC function. That means that there are infinitely many[5] e such that $\phi_e(e) \downarrow$ and $\phi_e(e) = f(e) = \langle A \upharpoonright_e \rangle$. We now build an infinite c.e. set V as follows: for all e, wait for $\phi_e(e)$ to converge. If the output codes a string of length e, enumerate it into V.

Since f is not DNC, we will have infinitely many e such that $\phi_e(e)$ codes a string of length e, so V is infinite. By construction there is at most one string of each length in V, so with Lemma 2.6 each of them have a complexity less then $K(n)$ if n is their length. Infinitely many of them are equal to an initial segment of A, so A is infinitely often K-trivial. □

From this theorem we get the following corollary, that is again in our paper [BV10]:

Corollary 2.18. *Each set that is computed by a 1-generic set is infinitely often K-trivial.*

Proof. This follows from a result by Demuth and Kučera, namely that when a set is 1-generic, it cannot compute any DNC function [DK87]. With Theorem 2.17 it then follows that it is infinitely often K-trivial. □

This corollary implies that there are sets that have a lower cone (for Turing reducibility) that consists of only infinitely often K-trivial sets. In general, infinitely often K-triviality is not downward closed under Turing reducibility. Think for example of the halting problem $\emptyset'$, that even has random sets in its lower cone, while it is infinitely often K-trivial itself (since it is c.e.).

To summarize, we have seen a number of results concerning infinitely often K-trivial sets. There are uncountably many of them, and even uncountably many are non-K-trivial. Infinitely often K-trivial sets can be found in each truth-table degree, each weakly 1-generic set is infinitely often K-trivial and a set computed by a 1-generic is infinitely often K-trivial.

However, the most interesting result is yet to come: we can construct a minimal pair in the K-degrees consisting of a Σ^0_2 set with any non-K-trivial c.e. set. There the fact that every c.e. set is infinitely often K-trivial is crucial. The construction of this minimal pair will be presented in the next chapter.

[5] If there were only finitely many, we could make a new function that takes a different value on these finitely many points. This will still be computable from A, so it still needs to be DNC and we can find another e such that $\phi_e(e) \downarrow$ and $\phi_e(e) = f(e)$.

3

Arithmetical definability in the K-degrees

Returning to our main topic, in this chapter and the next we study arithmetical definability in the K-degrees. This chapter is mainly dedicated to the construction of non-K-trivial sets that do not bound any non-K-trivial sets in a given arithmetical class of sets. Our main result is that given a Δ^0_2 family of sets, there is a non-K-trivial Δ^0_2 set that does not bound any non-K-trivial set in this family. This result implies that there is a non-K-trivial Δ^0_2 set that does not bound any non-K-trivial c.e. set, which is surprising since the initial segment complexity of c.e. sets is quite low.

We also show that for all $n > 1$ there is a non-K-trivial Σ^0_n set that does not bound any non-K-trivial Δ^0_n set. For the particular case of $n = 2$ the constructed Σ^0_2 set forms a minimal pair in the K-degrees with any non-K-trivial c.e. set. This improves on the lowest complexity of minimal pairs in the K-degrees known before (Merkle and Stephan constructed two Σ^0_2 sets [MS07]). All theorems in this chapter are new results, and they can also be found in our paper [BV10].

3.1 Sets that do not bound any non-K-trivial sets in a given class

As said, this chapter is about the construction of non-K-trivial sets that exclude all non-K-trivial sets in a given arithmetical class. We construct two instances of such results: a Σ^0_2 set that does not bound any non-K-trivial Δ^0_2 set (which relativizes to higher complexity classes), and a Δ^0_2 set that does not bound any non-K-trivial set in a given Δ^0_2 family of sets. The two constructions can be found in Section 3.2 and 3.3, respectively. They have some basic ideas in common, which we discuss in this first section.

In both cases we are dealing with an effective list of the sets involved. The first construction is about all Δ^0_2 sets, which are the total functionals in the list of all $\Phi^{\emptyset'}_e$. The second is about a Δ^0_2 family of sets, which means that they

are listed as $\Phi^{\emptyset'}_{f(e)}$ for some computable function $f(e)$ (the formal definition of a Δ^0_2 family of sets is discussed in the beginning of Section 3.3. We assume that for each e, $\Phi^{\emptyset'}_{f(e)}$ is total). To construct a set A that does not bound any non-K-trivial set in the list, we can go through all sets and test whether they are K-trivial. When a set X in the list is non-K-trivial, we make sure that for infinitely large e, there is an initial segment of A that has lower descriptive complexity than the corresponding segment of X plus the constant e. That way, $X \not\leq_K A$. This is expressed in the following requirement for the first case (of Δ^0_2 sets):

$$N_e : [\Phi^{\emptyset'}_e \text{ is total and } \Phi^{\emptyset'}_e >_K \emptyset] \Rightarrow \exists n\, [K(\Phi^{\emptyset'}_e \upharpoonright_n) > K(A \upharpoonright_n) + e].$$

The requirement for the second case (the Δ^0_2 family of sets) is as follows:

$$R_e : [\Phi^{\emptyset'}_{f(e)} >_K \emptyset] \Rightarrow \exists n[K(\Phi^{\emptyset'}_{f(e)} \upharpoonright_n) > K(A \upharpoonright_n) + e)].$$

Note that without loss of generality, we can in both cases assume that each set has infinitely many indices in the list. Therefore if we have some X that satisfies the antecedent of some requirement, we will have infinitely many e such that $\Phi^{\emptyset'}_e = X$ (or $\Phi^{\emptyset'}_{f(e)} = X$ in the second case). If all N_e (resp. R_e) are satisfied, that means that there are infinitely large e for which there is an n such that $K(X \upharpoonright_n) > K(A \upharpoonright_n) + e$, making sure that X is not K-below A.

To get an initial segment $A \upharpoonright_n$ of low descriptive complexity, we can add in all numbers up to n such that $A \upharpoonright_n$ consists of only ones, making it easy to describe. However, we also need A to be non-K-trivial. This corresponds to the following requirement:

$$Q_e : \exists n\, [K(A \upharpoonright_n) > K(n) + e].$$

In case we only wanted to satisfy one specific N_e (or R_e), we can do the following: after fixing the required initial segment of A with low complexity (to have $\Phi^{\emptyset'}_e \not\leq_K A$ or $\Phi^{\emptyset'}_{f(e)} \not\leq_K A$), we extend A with some Martin-Löf random sequence Y. This way, A will be of the form $A = \sigma * Y$ for some finite string σ, and we are guaranteed that A is non-K-trivial.

For satisfying all N_e (or R_e), we might need to make infinitely many segments of A have quite low complexity. Extending it with some infinite random sequence will therefore not work. Instead, we make sure that after each initial segment of low complexity, we add a complicated part to raise the complexity. In the two following sections, we explain for each case how we can implement these ideas.

3.2 A non-K-trivial Σ^0_2 set that does not bound any non-K-trivial Δ^0_2 set

In this section we construct a non-K-trivial Σ^0_2 set that does not bound any non-K-trivial Δ^0_2 set, proving Theorem 3.1. This theorem has an analog in the

Turing degrees, and can also be found in our paper [BV10]. Below we show that the construction will relativize to higher complexity classes as well.

Theorem 3.1. *There is a Σ^0_2 set $A >_K \emptyset$ such that for all Δ^0_2 sets $X >_K \emptyset$, $X \not\leq_K A$.*

Proof. We will enumerate the required A in a $\emptyset'$-computable construction. As explained in the previous section, we need to satisfy the following requirements to make A non-K-trivial (Q_e) and not bound any non-K-trivial Δ^0_2 (N_e), assuming that we have an effective list of all $\Phi_e^{\emptyset'}$.

$$\begin{aligned} Q_e :&\exists n\ [K(A\upharpoonright_n) > K(n) + e] \\ N_e :&[\Phi_e^{\emptyset'} \text{ is total and } \Phi_e^{\emptyset'} >_K \emptyset] \Rightarrow \\ &\exists n\ [K(\Phi_e^{\emptyset'}\upharpoonright_n) > K(A\upharpoonright_n) + e]. \end{aligned}$$

Satisfying only one N_e and all Q_e

For now we assume we only have to satisfy one N_e. As explained in the previous section, we pick some Martin-Löf random set Y, which we will use to give A the required complexity for satisfying Q_e. Then we search for some n such that $\Phi_e^{\emptyset'}\upharpoonright_n$ gets complex, and we make sure that it is not below A with constant e by adding all numbers up to n into A, making $A\upharpoonright_n$ of sufficiently low complexity because it will consist of only ones in a row. Then we continue with the digits of Y.

In particular, observe that there is a machine that describes any string consisting of only ones using only the length of this string. Let c be the coding constant of this machine: $K(1^n) \leq K(n)+c$ for all n. As soon as we find some n such that $K(\Phi_e^{\emptyset'}\upharpoonright_n) > K(n) + c + e$, we enumerated all numbers up to n into A. This makes $A\upharpoonright_n = 1^n$, thus making sure that $K(A\upharpoonright_n) \leq K(n) + c$. Therefore

$$K(A\upharpoonright_n) + e \leq K(n) + c + e < K(\Phi_e^{\emptyset'}\upharpoonright_n).$$

Construction of A. At stage s we define $A\upharpoonright_s [s] = Y\upharpoonright_s$, unless we find an n such that $K(\Phi_e^{\emptyset'}\upharpoonright_n) > K(n) + c + e$, in which case we enumerate all numbers up to n into A. Then we stop searching and continue to define $A[t] = 1^n * Y[t]$ where $*$ denotes concatenation.

Verification Observe that when we never find an n as described above, $\Phi_e^{\emptyset'}$ is K-trivial (via constant $c+e$) and N_e is trivially satisfied. Then $A = Y$ and all Q_e are satisfied as well. If we do find such an n, as shown above we can be sure that $K(\Phi_e^{\emptyset'}\upharpoonright_n) > K(A\upharpoonright_n) + e$, thus meeting N_e. Since A will consist of some finite number of ones and continue with the digits of Y it is still random, so we meet all Q_e as well.

Satisfying all N_e and all Q_e

The above strategy cannot be applied to all N_e right away, because we used the fact that only some finite initial segment of A will be different from Y to get A non-K-trivial. However, if we want to satisfy all N_e this might cause a problem, since we have to put ones into A more than just once.

So we mend our strategy to get the following: when we encounter a complex initial segment of $\Phi_e^{\emptyset'}$ (we will be more specific later) we put numbers into A. After that we extend A with a segment of Y as long as necessary to make A complex again: we need $K(A\upharpoonright_n) > K(n) + e$. Then in the next step, we will have to work with the initial segment of A as defined so far. The constant c as we had it before will therefore no longer work. However, we can have different constants like this depending on the given initial segment. We define the following parameters for this construction:

- $r_e[s]$ The length of the restraint that is put on A by an attempt to satisfy N_e and Q_e. We define $r_e[0] = 0$ for all e and $r_{-1}[s] = 0$ for all s. If $r_e[s+1]$ is not explicitly defined, we let it be equal to $r_e[s]$.
- $c_e[s]$ The constant that A can stay below given its current initial segment. $c_e[s]$ is such that

$$K((A\upharpoonright_{r_{e-1}})[s] * 1^\omega\upharpoonright_n) \leq K(n) + c_e[s].$$

- m_s The length of what is currently defined. $m_s = \max\{s, \max A[s]\}$.

We say that N_e *requires attention* at stage $s+1$ if there exists $n \leq s$ such that

$$\Phi_e^{\emptyset'}\upharpoonright_n\downarrow \text{ and } K(\Phi_e^{\emptyset'}\upharpoonright_n) > K(n) + c_e[s] + e.$$

Construction of A. At stage $s+1$ find the least $e \leq s$ such that N_e requires attention and is not declared satisfied. Enumerate all numbers n with $r_{e-1}[s] < n \leq m_s$. Then add a segment of Y as long as needed to make A complex. We define:

$$A[s+1] = (A\upharpoonright_{r_{e-1}})[s] * 1^{m_s - r_{e-1}[s]} * Y\upharpoonright_k$$

with k the least number such that $K(A[s+1]\upharpoonright_{m_s+k}) > K(m_s + k) + e$. Set $r_e[s+1] = m_s + k$ and declare N_e *satisfied* and all N_i with $i > e$ *not satisfied*. In case no N_e with $e < s$ requires attention, let $A[s+1] = A[s] * Y(s+1)$ ($Y(s+1)$ is the $s+1$st digit of Y).

Verification. We show by induction that each N_e is satisfied, and r_e and c_e reach a limit for all e. Assume this holds for all $k < e$ for some e. Let s_0 be the stage after which for all $k < e$, all N_k are satisfied and r_k and c_k have reached a limit. Either $\Phi_e^{\emptyset'}$ is K-trivial, and N_e is met. It will never require attention and r_e and c_e will remain constant after stage s_0.

Or $\Phi_e^{\emptyset'}$ is not K-trivial. Since all N_k with $k < e$ are satisfied by the induction hypothesis, r_e will stay on the same value, and so will c_e. From

some stage t on we will have $K(\Phi_e^{\emptyset'} \upharpoonright_n) > K(n) + c_e[t] + e$, and N_e requires attention. Since by the induction hypothesis all N_k with $k < e$ are satisfied, at a certain stage $s+1$ in the construction we will get to N_e since it is the least unsatisfied requirement that requires attention. Note that $n \leq s \leq m_s$ so by construction and the choice of $c_e[s]$ we get $K(A[s+1] \upharpoonright_n) \leq K(n) + c_e[s]$. It follows that

$$K(A[s+1] \upharpoonright_n) + e < K(\Phi_e^{\emptyset'} \upharpoonright_n)$$

and indeed N_e is satisfied. Since $A \upharpoonright_{r_e}$ will never change after this stage by setting $r_e[s+1] = m_s + k$, the above equation will hold at any stage after $s+1$. Furthermore, since all N_k with $k < e$ are satisfied by the induction hypothesis, N_e will remain satisfied, and r_e and c_e will remain constant.

Finally each Q_e is satisfied because of the following argument: either there are finitely many N_e that require attention, in which case A will be of the shape $A = \sigma * Y$ for some finite string σ. So we can be sure that A satisfies all Q_e. Or there are infinitely many N_e that require attention, and by the above argument all of them will at some stage receive attention and remain satisfied from then on. When that happens, we also add digits of Y to make sure that $K(A[s+1] \upharpoonright_{m_s+k}) > K(m_s + k) + e$. Since we do this for infinitely many e, Q_e is satisfied for infinitely large e. Therefore all Q_e are satisfied. □

As promised this theorem implies a minimal pair consisting of a Σ^0_2 set with any c.e. set of non-zero K-degree, improving the result from Merkle and Stephan [MS07].

Corollary 3.2. *There is a Σ^0_2 set that forms a minimal pair in the K-degrees with any non-K-trivial c.e. set.*

Proof. Let A be the Σ^0_2 set as in Theorem 3.1, and let B be any non-K-trivial c.e. set. By Theorem 2.3, B is infinitely often K-trivial. Take an arbitrary set X such that $X \leq_K A$ and $X \leq_K B$. From the latter it follows with proposition 2.4 that X is computable from $B \oplus \emptyset'$. So X is Δ^0_2. But by Theorem 3.1 we get that X must be K-trivial. Hence Σ^0_2 forms a minimal pair with the arbitrarily chosen c.e. set. □

Furthermore the proof of Theorem 3.1 used a $\emptyset'$-computable enumeration, making the resulting set $\Sigma^0_1(\emptyset')$ and hence Σ^0_2. This construction relativizes to a construction of a Σ^0_n set that does not bound any non-K-trivial Δ^0_n set for any $n > 1$. Therefore we have the following theorem, with a relativized corollary:

Theorem 3.3. *Let $n > 1$. There is a Σ^0_n set $A >_K \emptyset$ such that for all Δ^0_n sets $X >_K \emptyset$, $X \not\leq_K A$.*

Corollary 3.4. *Let $n > 1$. There is a Σ^0_n set that forms a minimal pair with any infinitely often K-trivial but non-K-trivial Δ^0_n set.*

The results in this section show that for any $n > 1$, we can have one Σ^0_n set that separates the Δ^0_n class into K-trivial and non-K-trivial sets. A natural question is whether we can do something similar for Δ^0_n classes and Σ^0_{n-1} classes. In the next section we show that indeed we can construct a non-K-trivial Δ^0_2 set that does not bound any non-K-trivial Σ^0_1 set, but this time the result will not relativize.

3.3 A non-K-trivial Δ^0_2 set that does not bound any non-K-trivial set in a Δ^0_2 family

In this section we study a similar problem to that in the previous section. Once again we construct a set that does not bound any non-K-trivial set in a given class. We prove a theorem similar to Theorem 3.1: given a Δ^0_2 family of sets there is a non-K-trivial Δ^0_2 that does not bound any non-K-trivial set in this family. This theorem and its proof can also be found in our paper [BV10]. The definition of a Δ^0_2 family is the following:

Definition 3.5 (Δ^0_2 family). *A class $\mathcal{C}$ of subsets of $\mathbb{N}$ is a Δ^0_2 family of sets (or uniformly $\emptyset'$-computable) if it can be written in the form $\{C_e : e \in \mathbb{N}\}$ where $C_e = \{n : \psi(e, n)\}$ and ψ is a Δ^0_2 property (expressible in both Σ^0_2 and Π^0_2 formulas).*

Note that for any C_e in a Δ^0_2 family $\mathcal{C}$, finding whether $n \in C_e$ is uniformly $\emptyset'$-computable from n and e. Therefore, with the s-m-n-theorem, a computable function f exists such that

$$\mathcal{C} = \{\Phi^{\emptyset'}_{f(e)} : e \in \mathbb{N}\}$$

where $\Phi^{\emptyset'}_{f(e)}$ is total for each e. We will use this in the proof below.

The class of all c.e. sets is a Δ^0_2 family of sets. Therefore Theorem 3.6 below has as a corollary that there is a Δ^0_2 set that does not bound any non-K-trivial c.e. set. This is surprising since c.e. sets have quite low initial segment complexity. For the plain complexity C this is quite easy to see: we can describe any initial segment of a c.e. set with the number of ones and the length of the segment[1], resulting in $C(B \upharpoonright_n) \leq^+ 2\log(n)$.

The following theorem is the main result in this chapter:

[1] For a c.e. set B we have a computable approximation $B[s]$ to this set. Given the length of the segment n and the number of ones k occurring in it, we can find $B \upharpoonright_n$ by going through the stages of the approximation. As soon as we find $B[s] \upharpoonright_n$ with k ones, this must be the final segment. So all we need to describe $B \upharpoonright_n$ is a description of n and k. Since $k \leq n$, this means that for the plain complexity, we can describe both numbers with at most $\log(n)$ bits (namely with their binary representation).

Theorem 3.6. *Given any Δ^0_2 family of sets there exists a Δ^0_2 set $A >_K \emptyset$ such that for any $X >_K \emptyset$ in the family, $X \not\leq_K A$.*

This proof builds on the ideas we discussed in Section 3.1. However, we need a slightly stronger method than in the previous section: the infinite injury method. This method is explained and illustrated with an example in Appendix B. A short formal version of the proof of Theorem 3.6 is presented below. A reader familiar with the infinite injury method can skip ahead to that self-contained proof, but to understand it well we will discuss the intuitions and the atomic case first.

A Δ^0_2 family $\mathcal{D} = \{\Phi^{\emptyset'}_{f(e)} : e \in \mathbb{N}\}$ is given, so we can effectively go through all the sets in this family to assure that A is such that for each e, if $\Phi^{\emptyset'}_{f(e)} >_K \emptyset$ then $\Phi^{\emptyset'}_{f(e)} \not\leq_K A$. For simplicity we write $X_e = \Phi^{\emptyset'}_{f(e)}$, and we have approximations

$$X_e[s] \text{ approximates } X_e = \Phi^{\emptyset'}_{f(e)}$$
$$K(X_e \upharpoonright_n)[s] \text{ approximates } K(X_e \upharpoonright_n).$$

As explained in Section 3.1, we have the following requirements to make sure that A is non-K-trivial (Q_e) and that A does not bound any non-K-trivial set in the family (R_e):

$$Q_e : \exists n[K(A \upharpoonright_n) > K(n) + e]$$
$$R_e : [X_e >_K \emptyset] \Rightarrow \exists n[K(X_e \upharpoonright_n) > K(A \upharpoonright_n) + e)].$$

For the sake of the argument below we will take on a different requirement to satisfy instead of R_e. From this new requirement, R_e will follow. We will build an infinite c.e. set V of strings such that for every n, there is at most one string of length n in V. By Lemma 2.6 it follows that for each string $\sigma \in V$, $K(\sigma) \leq^+ K(|\sigma|)$. Then we will meet the following requirement for each e:

$$N_e : [\forall n(K(X_e \upharpoonright_n) \leq K(A \upharpoonright_n) + e)] \Rightarrow \forall \sigma \in V[K(X_e \upharpoonright_{|\sigma|}) \leq^+ K(\sigma)].$$

This says that when a set is below A via constant e, for all $\sigma \in V$ the initial segment $X_e \upharpoonright_{|\sigma|}$ must have complexity less than $K(\sigma)$ up to a constant. By the properties of V it follows that X_e is K-trivial. Therefore this requirement implies that each set below A must be K-trivial, which is the contrapositive of R_e.

The strategy for one N_e

First we discuss how we can satisfy just one specific N_e and all Q_e. Given a specific set X_e in the family, its complexity either needs to stay below that of all strings in V for some infinite V as in Lemma 2.6 (X_e is K-trivial), or we need to make A such that X_e cannot stay below it with constant e. Simultaneously we want to keep A non-K-trivial.

To have A of high complexity, we start from a Martin-Löf random Δ^0_2 set Y. We will make sure that A is either equal to this Y, or it has some finite initial segment, and then continues with the digits of Y. Either way this will guarantee the satisfaction of all Q_e. Unfortunately this strategy can only be used for satisfying one N_e. As soon as we need to combine strategies, we will have to do something different to satisfy Q_e. However, it is still instructive to study this atomic strategy for understanding the strategy for all N_e combined.

When we are given X_e, we first need to find out whether it is K-trivial. This is not computable, but at each stage we have a guess based on the current situation, and we act accordingly. There can be two different outcomes (below we will explain what information we base our guess on):

0 guessing that X_e is K-trivial
1 guessing that X_e is non-K-trivial.

We picture this as a node with two successors (the possible outcomes), and to each outcome we attach a strategy. We may believe one of these outcomes at a certain stage, and another at the next. But after infinitely many stages, we can be sure that one of them is infinitely often visited, and the strategy for the leftmost infinitely often visited outcome will be successful.

To check whether X_e is K-trivial, we will build a set V as in Lemma 2.6. While going through the stages we have a *witness* attached to each possible outcome, which we intend to enumerate into V. At every stage s we test whether $K(X_e \upharpoonright_{|\sigma|}) \leq K(\sigma) + e$ for all $\sigma \in V[s]$. If this is the case, we guess that X_e is K-trivial, and we use the witness to put a longer string into V. This way we can make a more accurate guess at the next stage. Otherwise, there must be some $\sigma \in V[s]$ such that $K(X_e \upharpoonright_{|\sigma|})[s] > K(\sigma)[s] + e$, and we keep this σ as a witness, which we can in the end use to build A.

If we infinitely often guess that X_e is K-trivial, we will keep putting longer strings into V and by Lemma 2.6 we can be sure that X_e is in fact K-trivial. If we guess this only finitely often, we will eventually get stuck on a witness σ such that $K(X_e \upharpoonright_{|\sigma|}) > K(\sigma) + e$, from which we can build A. We will go into this in more detail below.

Formally, the possible outcomes and their witnesses are defined as follows: at stage $s + 1$ the outcome is

0 when $\forall \sigma \in V[s]$, $K(X_e \upharpoonright_{|\sigma|})[s+1] \leq K(\sigma)[s+1] + e$. The *witness* is $Y[s] \upharpoonright_i$ where i is the least number such that $V[s]$ does not contain a string of that length
1 when $\exists \sigma \in V[s]$ such that $K(X_e \upharpoonright_{|\sigma|})[s+1] > K(\sigma)[s+1] + e$. The *witness* is the shortest $\sigma \in V[s]$ such that $K(X_e \upharpoonright_{|\sigma|})[s] > K(\sigma)[s] + e$.

We build V as follows:

Building V. At stage $s + 1$, put the witness of the current outcome into V.

For building A, the idea is to find out what the witnesses are in the limit. First we prove that indeed the witnesses behave nicely:

Lemma 3.7. *For each $k \in \mathbb{N}$ there is a stage s_0 and some string σ of length k such that after stage s_0, all witnesses of N_e are compatible with σ.*

Proof. Either the outcome is infinitely often 1, and only finitely often 0. Then the witness will get stuck on the same σ (we keep picking the least) such that $K(X_e \upharpoonright_{|\sigma|}) > K(\sigma) + e$ (since X_e also converges). Or the outcome is infinitely often 0, and we will keep picking initial segments of Y. Since the approximation of Y converges, we can be sure that at some stage s_0 the initial segment $Y \upharpoonright_k$ has stabilized, and we only pick extensions of this. □

Definition of A. Now we can compute A from $\emptyset'$: at stage k, simultaneously look for

- a stage s_k and a string σ_k such that the witness of N_e after stage s_k is always an extension of σ_k.
- a stage s_0 such that the witness of N_e stays constant.

By Lemma 3.7, one of the above to stages must exist. If s_k is found first, let $A \upharpoonright_k := \sigma_k$ and continue with stage $k+1$. If s_0 is found first, let $A = \sigma_{k-1} * Y$ (where σ_{-1} is the empty string) and stop the procedure.

Verification. A is defined uniformly from the index e and $\emptyset'$. If no stage s_0 is found, that means that the witness never stabilizes on a constant value, and we will keep putting larger numbers into V, making it infinite. So by Lemma 2.6, X_e must be K-trivial and N_e is satisfied. Also by construction, we will have $A = Y$, so A is non-K-trivial and all Q_e are satisfied. If indeed a stage s_0 is found, there must be a least $\sigma \in V$ such that $K(X_e \upharpoonright_{|\sigma|}) > K(\sigma) + e$. The set A is constructed in such a way the σ is then surely an initial segment. So N_e is satisfied. And $A = \sigma * Y$ so A is non-K-trivial and all Q_e are satisfied.

Strategy for all N_e

The strategy as described above will not work for satisfying all requirements. Since we may need to define many initial segments for A to exclude the non-K-trivial X_e, we can no longer argue that A is non-K-trivial because of the way we use Y in the construction (compare this to the two cases in the construction of the Σ^0_2 set in the previous section). But we can keep a similar idea, and we add a secondary witnesses to the nodes to add complexity. If the primary witness is σ, we get the secondary witness τ using a Δ^0_2 function p_e with $p_e(\sigma) = \tau$ such that $\sigma \prec \tau$ and $K(\tau) > K(|\tau|) + e$. This way we can get a non-K-trivial A.

The idea is roughly as follows: we put all strategies together in a priority tree (see the proof of Theorem B.1). Each node has possible outcomes 0,1 similar to the atomic case above: 0 corresponds to the guess that X_e is K-trivial, 1 corresponds to X_e not being K-trivial. Again we base our guess on an infinite c.e. set of strings. We now make one such set V_α for each node α. Each V_α has only one string of length n for all $n \in \mathbb{N}^\alpha$ where $\mathbb{N}^{[\alpha]}$ is a partitioning of $\mathbb{N}$ and each $\mathbb{N}^{[\alpha]}$ is infinite.

At a stage s we can calculate the current path δ_s through the tree. We say that a stage s is an *α-stage* when $\alpha \preceq \delta_s$. Only at α-stages will we enumerate strings into V_α. While going through the stages of the construction we keep primary witnesses like in the atomic construction and secondary witnesses to add complexity. They may change as a result of outcomes of strategies with higher priorities. Below we prove a lemma that on the leftmost infinitely often visited path (the true path) they will eventually all settle. Then we define A to be the union of these *true witnesses*.

We prove another lemma below to show that these true witnesses make A satisfy all requirements. Finally we need an extra lemma to prove that A is Δ^0_2: it turns out that it is not only the union of the true witnesses, but also the limit of all $V_\alpha[s]$. We will go into this in more detail in the formal proof below.

Proof (of Theorem 3.6). Recall that we are given a Δ^0_2 family $\mathcal{D} = \{\Phi^{\emptyset'}_{f(e)} : e \in \mathbb{N}\}$ with $X_e[s]$ approximating $X_e = \Phi^{\emptyset'}_{f(e)}$. We need to satisfy for each e

$$Q_e : \exists n[K(A\upharpoonright_n) > K(n) + e)]$$
$$N_e : [\forall n(K(X_e\upharpoonright_n) \leq K(A\upharpoonright_n) + e)] \Rightarrow \forall \sigma \in V[K(X_e\upharpoonright_{|\sigma|}) \leq^+ K(\sigma)]$$

where V is some c.e. set as in Lemma 2.6. We put all strategies into a priority tree where the outcomes and witnesses of a node α visited at stage $s+1$ are as follows:

0 When for all $\sigma \in V_\alpha[s]$ that extend all current witnesses of each $\beta \prec \alpha$ the following holds:

$$[K(X_{|\alpha|}\upharpoonright_{|\sigma|})[s+1] \leq K(\sigma)[s+1] + |\alpha|].$$

The *primary witness* σ is the union of current witnesses of all $\beta \prec \alpha$. The *secondary witness* τ is $p_{|\alpha|}(\sigma)[s]$.

1 When there is some $\sigma \in V_\alpha[s]$ that extends all current witnesses of each $\beta \prec \alpha$ such that the follwing holds:

$$[K(X_{|\alpha|}\upharpoonright_{|\sigma|})[s+1] > K(\sigma)[s+1] + |\alpha|].$$

The *primary witness* σ is the shortest $\sigma \in V_\alpha[s]$ that extends all current witnesses of each $\beta \prec \alpha$ such that the following holds:

$$[K(X_{|\alpha|}\upharpoonright_{|\sigma|})[s+1] > K(\sigma)[s+1] + |\alpha|].$$

The *secondary witness* τ is $p_{|\alpha|}(\sigma)[s]$.

Here $p_e[s]$ is the approximation to a Δ^0_2 function such that

$$p_e(\sigma) = \tau \text{ such that } \tau \succ \sigma \text{ and } K(\tau) > K(|\tau|) + e.$$

Construction. At stage s, calculate the current path δ_s (we start with the root and for each node we can calculate the current outcome). Pick the least number $n < s$ that is in some $\mathbb{N}^{[\alpha]}$ for $\alpha * 0 \prec \delta_s$ and such that there is no string of that length in the corresponding V_α. Enumerate the least (in the length-lexicographical ordering) string of length n that is compatible with the current witness of δ_s. There might not be such an n; in that case do nothing and go to the next stage.

In the lemma below we will prove that the witnesses of nodes α reach a limit in the β-stages when β is the immediate successor of α and both are on the true path δ. We call this limit the *true witness* of α. We define A based in this:

Definition of A. Define A to be the union of the true witnesses of the true path δ.

Lemma 3.8. *Suppose that $\beta \prec \delta$ and α is the immediate predecessor of β. The witnesses of α reach a limit in the β-stages.*

Proof. Note that the secondary witnesses uniquely follow from the primary witnesses. So it suffices to show that the primary witnesses converge in the β-stages. We prove this by induction on the length of α. Assume that it holds for all α of length $< n$ and let σ be the union of the true witnesses of these nodes in the $\delta \upharpoonright_n$-stages. Let $\alpha = \delta \upharpoonright_n$ and $\beta = \delta \upharpoonright_{n+1}$.

Suppose 0 is the leftmost infinitely visited outcome of α: $\alpha * 0 \prec \delta$. Then by definition the primary witness of α will be the union of the current witnesses of all predecessors. Since by the induction hypothesis they converge in the α-stages to have a union σ, in the limit this σ will be the primary witness for α.

Suppose $\alpha * 1 \prec \delta$. In that case V_α will be finite since we only put things in on stages where α has outcome 0, and since δ is the *leftmost* infinitely often visited path, 0 will be visited only finitely often. Note that that the approximation of X_e converges, so at a large enough stage the primary witness for α will settle on the least string $\sigma \in V_\alpha$ such that $K(X_{|\alpha|} \upharpoonright_{|\sigma|}) > K(\sigma) + |\alpha|$, such that σ extends all witnesses of predecessors of α. □

Before we have defined A as the true witnesses of the nodes on the true path. Since the path can move around through the tree it is only $\emptyset''$-computable. To show that A is in fact Δ^0_2, we show that we can also find it by looking at the sets V_α: after some stage the strings that are enumerated into them will converge to the same A. This follows from the following lemma:

Lemma 3.9. *For every node $\beta \prec \delta$ with true secondary witness σ, there is a stage after which all strings enumerated into any V_α (α is any node on the tree) are extensions of σ.*

Proof. Take some arbitrary $\beta \prec \delta$ with true secondary witness σ. Let s_* be the stage after which this witness has reached its limit in the $\delta_{|\beta|+1}$-stages. In

the following argument all stages are taken to be larger than s_* and the last stage where a node to the left of β was visited.

We consider the different possible cases:

- α is to the left of β. Then there are only finitely many α-stages, so only finitely many things can be enumerated into V_α. After the last stage where something was enumerated into V_α, the claim is trivially satisfied.
- α is an extension of β and $\alpha \prec \delta$. In that case by construction only extensions of σ will be enumerated into V_α.
- α is an initial segment of β, which we split up into the following cases:
  - $\alpha \preceq \beta$ and $\alpha * 1 \prec \delta$, in which case only finitely many things will be enumerated into V_α (because this only happens at stages with outcome 0, and since we are on the leftmost infinitely often visited path, this happens only finitely often). So again the claim is trivially satisfied.
  - $\alpha \preceq \beta$ and $\alpha * 0 \prec \delta$, in which case we can prove this by finite induction. Let $\eta_0 \succ ... \succ \eta_t$ be the strings such that $\eta_i \preceq \beta$ and $\eta_i * 0 \preceq \delta$ for each $i \leq t$. Fix some $i < t$ and assume that it holds for all η_j with $j < i$. We prove it for η_i. Let ρ_j be the union of all the witnesses of the predecessors of η_j, for each $j < i$. And let s_i be a stage such that for each $j < i$
    - after s_i we have $K(X_{|\eta_j|} \upharpoonright_{|\tau|}) \leq K(\tau) + |\eta_j|$ for each string τ in V_{η_j} that is an extension of ρ_j but not of σ.
    - the longest string in V_{η_j} is an extension of σ and after s_i only extensions of σ are enumerated into V_{η_j}

    such a stage exists because of the induction hypothesis. If a string is enumerated into V_{η_i} at stage $s > s_i$, then either δ_s extends the true outcome of β, or $\delta_s \succ \eta_j * 1$ for some $j < i$. In the first case (and the latter does not apply) the enumerated string must be an extension of the true primary witness σ of β by definition. If $\delta_s \succ \eta_j * 1$ for some $j < i$ the enumerated string must the current primary witness of η_j, but by the choice of s_i this will extend σ. So in either case it must be an extension of σ.
- α is to the right of β: $\alpha \succ \eta$ such that $\eta \prec \beta$ and $\eta * 0 \prec \delta$ and $\eta * 1 \prec \alpha$. In this case the length of the witnesses of η go to infinity, because the outcome of η is infinitely often 0. By the previous item, after some stage the strings enumerated into them will all extend σ. Since the strings enumerated into V_α must extend those of V_η, that means they must also extend σ.

In all cases the strings enumerated into V_α are extensions of σ, so this concludes the proof. □

Verification. Since the tree is finitely branching, we can be sure that there is indeed an infinitely often visited path, and we take the true path δ to be the leftmost such. Note that from the construction it follows that at any stage s the witnesses on the initial segments of δ_s are linearly ordered. In the lemma below we show that the strategies at each node α on the true path do what we want them to do:

Lemma 3.10. *If $\alpha * 1 \prec \delta$ then V_α is finite and $K(X_{|\alpha|} \upharpoonright_{|\sigma|}) > K(\sigma) + |\alpha|$ where σ is the true primary witness of α. If $\alpha * 0 \prec \delta$ then V_α contains a string of each length in $\mathbb{N}^{[\alpha]}$ and $X_{|\alpha|}$ is K-trivial.*

Proof. We prove the two cases. If $\alpha * 1 \prec \delta$ then V_α is finite because it is the leftmost infinitely often visited outcome and we only enumerate strings into V_α on outcome 0. Since $\alpha * 1 \prec \delta$, by construction the true primary witness will be the least σ such that $K(X_{|\alpha|} \upharpoonright_{|\sigma|}) > K(\sigma) + |\alpha|$ and σ extends the true witnesses of the predecessors of α.

If $\alpha * 0 \prec \delta$ then since we infinitely often have outcome 0, we enumerate infinitely many strings into V_α. By the way the construction is defined we can be sure that it contains a string of each length in $\mathbb{N}^{[\alpha]}$. The witnesses of the predecessors of α will reach a limit in the α-stages by Lemma 3.8, say that the union of those true witnesses is σ. Then by construction after some stage only extensions τ of sigma will be enumerated into V_α, and since $\alpha * 0 \prec \delta$, it follows that for each such τ, $K(X_{|\alpha|} \upharpoonright_{|\tau|}) \leq K(\tau) + |\alpha|$. With Lemma 2.6 we have that $K(\tau) \leq^+ K(|\tau|)$ for all these τ. It follows that for allmost all $k \in \mathbb{N}^{[\alpha]}$, $K(X_{|\alpha|} \upharpoonright_k) \leq^+ K(k)$ so $X_{|\alpha|}$ is K-trivial. □

Finally we have all we need to prove that A does indeed satisfy all the requirements:

Lemma 3.11. *The set A is Δ^0_2 and satisfies all N_e, Q_e for $e \in \mathbb{N}$.*

Proof. From Lemma 3.9 we see that we can find A by taking the limit of all strings enumerated into all V_α. From the same lemma it follows that an initial segment of A changes only finitely often before it settles on the true witness. This makes A a Δ^0_2 set.

Next take some $e \in \mathbb{N}$. Let α be the unique node of length e on δ, and let σ be the true primary witness of α. We will have $\sigma \prec A$ by definition of A, and by construction also $p_e(\sigma) \prec A$. The function $p_e(\sigma)$ was defined such that $K(p_e(\sigma)) > K(|p_e(\sigma)|) + e$, so we can be sure that Q_e is satisfied.

For N_e, suppose that X_e is not K-trivial. Then by Lemma 3.10 it follows that $\alpha * 0 \not\prec \delta$. So $\alpha * 1 \prec \delta$, and $K(X_e \upharpoonright_{|\sigma|}) > \sigma + e$. By definition of A, $\sigma \prec A$, so also N_e is satisfied. □

This concludes the proof. □

As mentioned before the theorem above can be applied to the class of all c.e. (Σ^0_1) sets, since they form a Δ^0_2 family. We get the following corollary.

Corollary 3.12. *There is a non-K-trivial Δ^0_2 set that does not bound any non-K-trivial Σ^0_1 set.*

The corollary is quite surprising, since c.e. sets have quite low initial segment complexity. But the deeper result is the theorem itself, which shows an elementary difference between the K-degrees and the LK-degrees. For the latter,

it was proved in [Bar10] that for any nontrivial Δ^0_2 set there is a nontrivial c.e. set LK-below it. Extending that proof led to the result that below any pair of nontrivial Δ^0_2 sets there is one nontrivial c.e. set LK-below both. Because of that, no Δ^0_2 minimal pair can exist in the LK-degrees.

We can conclude from Theorem 3.6 and in particular Corollary 3.12 that the proof of the impossibility of a Δ^0_2 minimal pair in the LK-degrees does not carry over to the K-degrees. Furthermore, unlike in the previous section, there is no direct corollary to Theorem 3.6 that shows the existence of a Δ^0_2 minimal pair in the K-degrees. This means that the question whether such a minimal pair can exist, remains open.

To summarize, this chapter investigated arithmetical definability in the K-degrees. We constructed a non-K-trivial Σ^0_2 set that does not bound any non-K-trivial Δ^0_2 set, and we remarked that this construction relativizes to a non-K-trivial Σ^0_n set that does not bound any non-K-trivial Δ^0_n set for all $n > 1$. Furthermore, the constructed Σ^0_2 set forms a minimal pair in the K-degrees with any non-K-trivial c.e. set. The main result in this chapter was the existence of a non-K-trivial Δ^0_2 set that does not bound any non-K-trivial set in a given Δ^0_2 family.

4

Gap functions and minimal pairs

In the previous chapter we discussed non-K-trivial sets that do not bound any non-K-trivial set in a class of sets. We proved that for all $n > 1$ there is a Σ^0_n set that does not bound any non-K-trivial Δ^0_n set, and that there is a Δ^0_2 set that does not bound any non-K-trivial set in a given Δ^0_2 family of sets. The first induced a minimal pair of a Σ^0_2 set with any non-K-trivial c.e. set, but the latter did not result in a Δ^0_2 minimal pair.

In this chapter we investigate a method for constructing minimal pairs in the K-degrees, namely via gap functions for K-triviality. Gap functions were used for the first construction of a minimal pair in the K-degrees by Csima and Montalbán [CM06], as we will present below.

We will see in this chapter that if a gap function is unbounded and non-decreasing, a minimal pair immediately follows from it. We reconstruct Csima and Montalbán's unbounded non-decreasing Δ^0_4 gap function, inducing a Δ^0_4 minimal pair. We also construct some gap functions with different properties. Finally, we prove that there cannot be an unbounded non-decreasing Δ^0_2 gap function, which can also be found in our paper [BV10]. That means that this technique of constructing a minimal pair via a gap function cannot be used to construct a Δ^0_2 minimal pair.

4.1 Gap functions

Before, we defined a set to be K-trivial if its descriptive complexity stays within a constant from the complexity of the lengths of the segments. Now we characterize the K-trivial sets using a function that shows how far the complexity $K(X \upharpoonright_n)$ can rise above that of $K(n)$, and we will see that this is not necessarily a constant function. In fact we will see that we can have an unbounded non-decreasing gap function for K-triviality, which is quite surprising considering the definition of K-triviality. The following definition of a gap function is due to Csima and Montalbán [CM06]:

Definition 4.1 (Gap function for K-triviality). *$f : \mathbb{N} \to \mathbb{N}$ is a gap function for K-triviality if for all sets X,*

$$X \text{ is } K\text{-trivial} \Leftrightarrow \forall n\, K(X \upharpoonright_n) \leq^+ K(n) + f(n).$$

When a set X satisfies the righthandside of this, we say that X *obeys* f. Note that by the definition of K-triviality, any function f satisfies the ($\Rightarrow$)-direction, and in particular the constant 0 function is a gap function. Moreover, note that any bounded function f also satisfies the ($\Leftarrow$)-direction:

Proposition 4.2. *For any bounded function $f : \mathbb{N} \to \mathbb{N}$, it holds for any set X that*

$$X \text{ is } K\text{-trivial} \Leftrightarrow \forall n\, K(X \upharpoonright_n) \leq^+ K(n) + f(n).$$

Proof. ($\Rightarrow$) Follows immediately from the definition of K-triviality.
($\Leftarrow$) Since f is bounded, there is some b such that for all $n \in \mathbb{N}$, $f(n) \leq b$. Now take some arbitrary set X and assume the righthandside:

$$\forall n\, K(X \upharpoonright_n) \leq^+ K(n) + f(n).$$

It follows that

$$\forall n\, K(X \upharpoonright_n) \leq^+ K(n) + b.$$

So X is K-trivial:

$$\forall n\, K(X \upharpoonright_n) \leq^+ K(n).$$

□

Because of the above proposition, unbounded gap functions are the more interesting case. In this chapter we will show that an unbounded non-decreasing gap function induces a minimal pair in the K-degrees, and we will show that such a function does in fact exist. We present the construction as first done by Csima and Montalbán in their 2006 paper [CM06], to show that a minimal pair in the K-degrees exists. Studying the complexity of these sets was something that only started later on. In light of that development we investigate whether Csima and Montalbán's technique can be adapted to construct a minimal pair of lower complexity, which we answer in the negative for Δ^0_2.

4.2 Minimal pairs via gap functions

A minimal pair in the K-degrees consisting of sets A and B must satisfy the following two properties:

(i) A and B are non-K-trivial
(ii) For any X such that $X \leq_K A$ and $X \leq_K B$, X must be K-trivial.

Intuitively it may seem that two sets A and B can only satisfy the second requirement when they are alternately of low complexity. Namely when they satisfy

$$\forall n \ \min\{K(A\upharpoonright_n), K(B\upharpoonright_n)\} \leq^+ K(n).$$

However, this is a very tight restriction on the behaviour of both sets. A gap function for K-triviality gives us a way to circumvent this. Using gap functions the restriction on the two sets A and B can become the following:

$$\forall n \ \min\{K(A\upharpoonright_n), K(B\upharpoonright_n)\} \leq^+ K(n) + f(n).$$

When given an unbounded non-decreasing gap function, this restriction gives us enough space to construct a minimal pair of the same complexity as the gap function itself. We will see this in the proof of Theorem 4.3 below, which was not proved anywhere else but can be extracted from Csima and Montalbán's argument in [CM06]. They also constructed an unbounded non-decreasing Δ^0_4 gap function and derived a Δ^0_4 minimal pair from it using the method below. We will go into their construction of a gap function in the next section.

Theorem 4.3 (Csima, Montalbán [CM06]). *Given an unbounded non-decreasing gap function f, there exists a minimal pair in the K-degrees such that both sets have complexity $f \oplus \emptyset'$.*

Proof. We will define sets A and B such that they satisfy the requirement as mentioned before: for all n, $\min\{K(A\upharpoonright_n), K(B\upharpoonright_n)\} \leq^+ K(n) + f(n)$. We define A and B in stages, making A complex and B of low complexity at odd stages, and the other way around at even stages. We define them inductively as follows:

- $A_0 = B_0 = \lambda$ the empty string.
- At $e{+}1$ if e is even, we find an extension of A_e that adds enough complexity: let
$$\tilde{A}_{e+1} \succ A_e \text{ such that } K(\tilde{A}_{e+1}) > K(|\tilde{A}_{e+1}|) + e.$$
Observe that this is always possible: by the counting condition (see Chapter 1) there is always an extension that raises the complexity sufficiently. Let m_e be the length of the added segment:
$$m_e = |\tilde{A}_{e+1}| - |A_e|.$$
Next we will add enough zeros to the extension $\tilde{A}_{e+1}$ to get to a length where the complexity is below the gap function again. This way A will be of low complexity in the next step, and we can then continue to raise B's complexity. First we find the constant by which $\tilde{A}_{e+1}$ is K-trivial if we extend it with infinitely many zeros: let c_e be such that
$$\forall n \ K((\tilde{A}_{e+1} * 0^\omega)\upharpoonright_n) \leq K(n) + c_e.$$

If we can find a number where the gap function grows beyond this constant, we are safe, since the function is non-decreasing. Since it is unbounded we are guaranteed to always find such a number n_e:

$$f(n_e) > c_e.$$

Now we define

$$A_{e+1} = \tilde{A}_{e+1} * 0^{n_e}$$
$$B_{e+1} = B_e * 0^{m_e+n_e}.$$

By adding this many zeros to B_e we make B_{e+1} as long as A_{e+1}, and by the inductive argument we can be sure that B_{e+1} stays below the gap function.

- At $e+1$ if e is odd, follow the above procedure but with the roles of A and B interchanged.

Observe that by construction, for every e there are n and m such that

$$K(A \upharpoonright_n) > K(n) + e$$
$$K(B \upharpoonright_m) > K(m) + e.$$

So both sets are non-K-trivial. Furthermore, by construction we have

$$\forall n \ \min\{K(A \upharpoonright_n), K(B \upharpoonright_n)\} \leq^+ K(n) + f(n).$$

Note that this construction relies mostly on f, and we need $\emptyset'$ to find the complexity K at a few points in the construction. Therefore the complexity of both sets is $f \oplus \emptyset'$. This concludes the proof. □

With this result, all we have to do to find a minimal pair is construct an unbounded non-decreasing gap function. We do this in the next section. Later on in this chapter we will get back to the relation between minimal pairs and gap functions: we will see then that a minimal pair also canonically induces a gap function. However, this gap function might not be unbounded, so the above theorem cannot be extended to a bi-implication.

4.3 An unbounded non-decreasing gap function

The previous section discussed how an unbounded non-decreasing gap function induces a minimal pair in the K-degrees. However, an even more significant result from Csima and Montalbán's paper [CM06] is the existence of an unbounded non-decreasing gap function:

Theorem 4.4 (Csima and Montalbán [CM06]). *There is an unbounded non-decreasing Δ^0_4 function f such that a set X is K-trivial iff $K(X \upharpoonright_n) \leq^+ K(n) + f(n)$ for all n.*

With Theorem 4.3 we immediately get the following corollary:

Corollary 4.5. *There is a Δ^0_4 minimal pair in the K-degrees.*

Before reconstructing Csima and Montalbán's gap function and thus proving this theorem and its corollary, we note how surprising it is that an unbounded gap function can exist. For when we want to prove that a set X is not K-trivial, we show that for every c there is some initial segment $X \upharpoonright_{n_c}$ such that $K(X \upharpoonright_{n_c}) > K(n_c) + c$. It seems like an unbounded gap function allows us to do precisely this.

In fact, if X were to take up exactly all the space that some arbitrary unbounded non-decreasing f gives (so for all n, $K(X \upharpoonright_n) = K(n) + f(n)$) then it *would* be non-K-trivial. However, the complexity of a certain segment $X \upharpoonright_n$ heavily depends on the previous segments. If f increases very slow, we cannot expect to have enough control over the complexity of each $X \upharpoonright_n$ to make a set that takes precisely this value at any n.

Below we will see that if an unbounded non-decreasing gap function f were computable, it is possible to construct a non-K-trivial set X that obeys f. The same holds for a Δ^0_2 function, which is the main result in this chapter. But first we show that a more complex function can exclude all non-K-trivial sets. We present the construction as can be found in Csima and Montalbán's paper, but we get into it in much more detail than they do.

What we would like to do, is check every set X for K-triviality. If it is not K-trivial, we would make sure that it cannot obey f. However, we have no effective listing of all sets. Below we will see that if there were such an effective list, we could make a gap function of lower complexity. But for this case we need to come up with something smarter. What we do, is to first look at the sets X such that they are K-trivial with constant $e + i$, but not with constant e (so $X \in \mathrm{KT}(e + i) \backslash \mathrm{KT}(e)$, where $\mathrm{KT}(e)$ is the class of sets that are K-trivial via constant e). Observe that for any e, $\mathrm{KT}(e)$ is finite and we can list all these sets (see the proof of Theorem 1.14). Then we let e tend to infinity, and we get the sets that are not K-trivial with any constant.

Proof (of Theorem 4.4). We will make a function f_e for each e. We want a function f_e to rule out that a set is K-trivial with constant e:

$$X \notin \mathrm{KT}(e) \Rightarrow \exists m \; K(X \upharpoonright_m) > K(m) + f_e(m).$$

If we have a set $X \notin \mathrm{KT}(e)$, we get one number m such that $K(X \upharpoonright_m) > K(m) + e$, which we will use to have f_e satisfy the equation above. We will also make sure that for all $e, m > 0$, $f_e(m) \geq e$, and we define f to be

$$f(m) = \min\{f_{2e}(m) - e : e > 0\}.$$

This will do exactly what we want it to do. To see this, take some non-K-trivial set X, then X is not K-trivial with any constant. So for all e, $\exists m \; K(X \upharpoonright_m) >$

$K(m)+f_e(m)$. Now take an arbitrary c. Then $\exists m\ K(X\upharpoonright_m) > K(m)+f_{2c}(m)$, and we can be sure that $f(m) \leq f_{2c}(m) - c$. So it will follow that

$$K(X\upharpoonright_m) > K(m) + f_{2c}(m) - c + c \geq K(m) + f(m) + c.$$

Making f non-decreasing and unbounded

To get an unbounded and non-decreasing function f, we need the functions f_e to satisfy some more properties. Suppose for each e we have the following:

- f_e is non-decreasing
- f_e is unbounded
- $f_e(m) \geq e$ for all m, e.

Then we can get that f is non-decreasing and unbounded: suppose for some m, $f(m+1) < f(m)$. We have

$$f(m) = f_{2e}(m) - e$$
$$f(m+1) = f_{2e'}(m+1) - e'$$

for some $e, e' > 0$. Since $f_{2e}(m) - e$ is the minimum over all e, we have $f_{2e}(m) - e \leq f_{2e'}(m) - e'$. It follows that $f_{2e'}(m+1) - e' < f_{2e}(m) - e \leq f_{2e'}(m) - e'$. So $f_{2e'}(m+1) < f_{2e'}(m)$. This contradicts the claim that $f_{2e'}$ is non-decreasing. So f must be non-decreasing.

And f will be unbounded: suppose that for all m, $f(m) \leq q$ for some bound q. For each function f_{2e} with $e > q$, the value of $f_{2e}(m) - e$ is strictly larger than q for any m, because $f_{2e}(m) - e \geq 2e - e = e > q$. Now for each $e \leq q$, since f_{2e} is unbounded, we can find some m_e such that $f_{2e}(m_e) > 2q$. Since the set of these m_e is finite, we can take the maximum: $k := \max\{m_e : e \leq q\}$. For this k we get by construction (f_{2e} is non-decreasing) that if $e \leq q$, then $f_{2e}(k) - e > 2q - e \geq q$. So together with the previous we have

$$f(k) = \min\{f_{2e}(k) - e : e > 0\} > q$$

which shows that f is unbounded.

Constructing f_e

What remains, is to construct functions f_e such that they satisfy

- if $X \notin \mathrm{KT}(e)$ then $\exists m\ K(X\upharpoonright_m) > K(m) + f_e(m)$
- f_e is non-decreasing
- f_e is unbounded
- $f_e(m) \geq e$ for all m, e.

To make f_e unbounded, we not only look at $\mathrm{KT}(e+i)\backslash\mathrm{KT}(e)$ for increasing i, but also at $\mathrm{KT}(e+i)\backslash\mathrm{KT}(e+j)$ for $j<i$. We use this to increase the value of f_e at the right moment, to make the function unbounded.

To construct f_e non-decreasing, we mark the values of f_e with markers n_i, and keep them in order ($n_0 < n_1 < n_2 ...$). In the end we let

$$f_e(m) = e + i - 1 \text{ the least } i \text{ s.t. } m \leq n_i.$$

Now the strategy is as follows: set $n_0 = 0$, and find n_1 such that for every $Y \in \mathrm{KT}(e+1)\backslash\mathrm{KT}(e)$, there is an $m_Y < n_1$ that witnesses that $Y \notin \mathrm{KT}(e)$ (so $K(Y\upharpoonright_{m_Y}) > K(m_Y)+e$). Note that such an n_1 exists, because $KT(e+1)$ is finite, so we only need n_1 to be larger than finitely many such m_Y.

Similarly, we pick n_2 such that for every $Y \in \mathrm{KT}(e+2)\backslash\mathrm{KT}(e)$ there is an $m_Y < n_2$ such that $K(Y\upharpoonright_{m_Y}) > K(m_Y) + e$. And we require a similar thing for n_3. However, now we can come across sets $Z \in \mathrm{KT}(e+3)\backslash\mathrm{KT}(e)$ such that the least m_Z is in $[n_1, n_2)$. If Z were in $\mathrm{KT}(e+1)$, by definition of n_1 we would have been able to find such an $m_Z < n_1$, but we did not. So we know that $Z \notin \mathrm{KT}(e+1)$. Hence there must be some number l_Z such that $K(Z\upharpoonright_{l_Z}) > K(l_Z) + e + 1$. We put an extra requirement on n_3, namely that we can also find[1] such an $l_Z < n_3$.

Formally, we define the sequence $n_0 < n_1 < n_2 ...$ such that

- for any $Y \in \mathrm{KT}(e+i)\backslash\mathrm{KT}(e)$, there is an $m_Y < n_i$ with $K(Y\upharpoonright_{m_Y}) > K(m_Y) + e$.
- for any Z such that the least m_Z such that $K(Z\upharpoonright_{m_Z}) > K(m_Z) + e$ is in $[n_{i-2}, n_{i-1})$, there is $l_Z < n_i$ such that $K(Z\upharpoonright_{l_Z}) > K(l_Z) + e + i - 2$.

Now we define

$$f_e(k) := e + i - 1 \text{ if } k \in [n_{i-1}, n_i).$$

Since $n_1 > 0$ we have $f_e(m) \geq e$ for all $m > 0$. For every e this function is non-decreasing, since $n_0 < n_1 < n_2 ....$ It is also unbounded, because for every $q \geq e-2$ we can find n_{q-e+2}, and by construction $k = n_{q-e+2}$ gives us $f_e(k) = e + (q-e+2) - 1 = q+1$, so for every attempt of a bound we can find a value of f that is larger.

We have now constructed the f_e for $e > 0$ such that they are unbounded and nondecreasing, and $f_e(m) \geq e$ for all m, e. As promised, we define

$$f(k) := \min\{f_{2e}(k) - e : e > 0\}.$$

[1] Such an n_3 exists: suppose it did not. That means that for any choice of $n_3 > 0$, we have a set Z_{n_3} such that the least m_Z such that $(Z_{n_3}\upharpoonright_{m_Z}) > K(m_Z) + e$ is in $[n_1, n_2)$, but still $K(Z_{n_3}\upharpoonright_{n_3-1}) \leq K(n_3 - 1) + e + 1$. Take all these initial segments $Z_{n_3}\upharpoonright_{n_3-1}$ for each choice of n_3, they form an infinite tree. Since the tree is finitely branching, by König's tree lemma we know there must be an infinite path, Z. For this Z we have for all n that $K(Z\upharpoonright_n) \leq K(n) + e + 1$: Z is in $\mathrm{KT}(e+1)$. This contradicts the choice of n_1 and the claim that $m_Z \in [n_1, n_2)$

As shown before, this f is an unbounded and non-decreasing gap function for K-triviality.

This function is Δ^0_4 because if we want to find $f(m)$ for some m, we need to find the minimum over all $f_e(m)$. Each f_e is Δ^0_3 because for finding the value of $f_e(m)$ we keep increasing i until we find $m \leq n_i$. We will find this value because f_e is unbounded. For finding n_i for some i we go through the sets in $\mathrm{KT}(e+i)\backslash\mathrm{KT}(e+j)$ for $j \leq i$, which are Δ^0_3 since each $\mathrm{KT}(e)$ is Δ^0_2 and finite (see the proof of Theorem 1.14). For any set $X \notin \mathrm{KT}(e)$ we can just go through all initial segments to find m_X such that $K(X\upharpoonright_{m_X}) > K(m_X) + e$, eventually we will find one. This makes f_e a Δ^0_3 function, and f a Δ^0_4 function. □

The above construction became quite complex because we do not have an effective enumeration of all sets. But if a gap function were only to work for c.e. or Δ^0_2 sets, we would have such an enumeration, and we can make a function of lower complexity. By a gap function for c.e. sets we mean the following:

Definition 4.6 (Gap function for c.e. sets). *$f : \mathbb{N} \to \mathbb{N}$ is a gap function for K-triviality of c.e. sets if for all c.e. sets X,*

$$X \text{ is } K\text{-trivial} \Leftrightarrow \forall n\ K(X\upharpoonright_n) \leq^+ K(n) + f(n).$$

An analogous definition holds for other complexity classes, such as Δ^0_2 sets. Both in the case of c.e. sets and Δ^0_2 sets, we have an effective enumeration, and in both cases this allows us to construct a gap function of Σ^0_2 degree[2] for that class. We show this for a gap function for Δ^0_2 sets, which is also in our paper [BV10]. The case for c.e. sets is similar but simpler. Note that both resemble the proof of Theorem 3.1.

Theorem 4.7. *There is an unbounded non-decreasing gap function of Σ^0_2 degree for Δ^0_2 sets.*

Proof. The idea for this function will be to go through the list of all $\Phi_e^{\emptyset'}$ and whenever it converges and is non-K-trivial, make sure it cannot obey the function. In other words, we want to satisfy the following requirement:

$$R_e : [\Phi_e^{\emptyset'}[t] \text{ is total and } \Phi_e^{\emptyset'}[t] >_K \emptyset] \Rightarrow \exists m\ [K(\Phi_e^{\emptyset'}[t]) > K(m) + f(m) + e].$$

Note that this requirement is enough, because we can assume without loss of generality that every set has infinitely many indices in the list (see for more on this the proof of Theorem 3.1). If we only had one requirement R_e to satisfy, we could make the following function:

[2] Note that in this case we do not mean that the graph of the function is Σ^0_2, but that the degree of the function is Σ^0_2. This means that the function f has a $\emptyset'$-computable approximation $f[s]$ such that for all n, $f(n) = \lim_{s\to\infty} f(n)[s]$ and it approximates f from above: for all n, if $s < t$ then $f(n)[s] \geq f(n)[t]$

- Let $f(n)[s] = n$ for all $n \leq s$ until at some stage t we have some number $m > e$ such that $\Phi_e^{\emptyset'}[t] \upharpoonright_m \downarrow$ and $K(\Phi_e^{\emptyset'} \upharpoonright_m) > K(m)+2e$. Then let $f(i)[t] = e$ for all i with $e \leq i \leq m$, and $f(i)[t] = i$ for all i with $m < i \leq t$.

Observe that this function is by construction unbounded and non-decreasing. It satisfies the requirement, because if $\Phi_e^{\emptyset'}$ converges and is non-K-trivial we will find such a number m at a certain stage and we will get $f(m) = e$. So $K(\Phi_e^{\emptyset'} \upharpoonright_m) > K(m) + 2e = K(m) + f(m) + e$.

If we want to satisfy all requirements, we need to do some more work. We make sure that a function does not mess up the previous definition of f by keeping track of $k_e[s]$, the least number that is larger than the *witness* m of any R_i with $i < e$. Then we do the same as above, except now we only look for some $m > k_e[s]$ that witnesses the complexity of $\Phi_e^{\emptyset'}$, and we only change the value of $f(i)$ if $k_e[s] \leq i \leq m$. We say that R_e requires attention at stage t if there is a number m with $k_e[t] < m \leq t$ such that $\Phi_e^{\emptyset'}[t] \upharpoonright_m \downarrow$ and $K(\Phi_e^{\emptyset'} \upharpoonright_m) > K(m) + 2e$. We define f as follows:

- At stage t, find the least e such that R_e requires attention. Then let $f(i)[t] = e$ for all i with $k_e[t] \leq i \leq m$ and $f(i)[t] = i$ for i with $m < i \leq t$. If there is no such e, let $f(n)[t] = n$ for all $k_e[t] \leq n \leq t$.

This way f is monotonically approximable from above, so since we only need $\emptyset'$ to determine the descriptive complexity K at each stage, this function is of Σ_2^0 degree. Note that there is no injury amongst the requirements, and each requirement acts only once, so this function will be unbounded and non-decreasing. All requirements are met, because if they require attention they will receive it at a certain stage. This concludes the proof. □

A similar technique can be applied to the list of c.e. sets, to obtain an unbounded non-decreasing gap function of Σ_2^0 degree for c.e. sets.

Theorem 4.8. *There is an unbounded non-decreasing gap function of Σ_2^0 degree for K-triviality of c.e. sets.*

Proof. We satisfy requirements

$$R_e : W_e >_K \emptyset \Rightarrow \exists m\, [K(W_e \upharpoonright_m) > K(m) + f(m) + e].$$

The construction to do this is analogous to the proof of Theorem 4.7. □

We conclude this section noting that a gap function for c.e. sets can be of even lower complexity if the requirement for f to be non-decreasing is dropped. Then we can just go through the requirements above, and set $f(m) = e$ if $K(W_e \upharpoonright_m) > K(m) + 2e$, and $f(m) = m$ otherwise. This function will be Δ_2^0, and by construction unbounded. However, it is no longer non-decreasing.

4.4 No Δ_2^0 gap function

In the previous section we showed Csima and Montalbán's construction of an unbounded non-decreasing Δ_4^0 gap function, thus obtaining a Δ_4^0 minimal pair. In this section we will prove that no unbounded non-decreasing Δ_2^0 gap function can exist. This shows that no Δ_2^0 minimal pair can be constructed this way, though it might exists nonetheless.

Theorem 4.9. *There is no unbounded non-decreasing Δ_2^0 gap function for K-triviality.*

The proof of this theorem uses the Δ_2^0 approximation of f. Before we get into the proof, we prove the following lemma:

Lemma 4.10. *Given an unbounded non-decreasing Δ_2^0 function f, we can find an unbounded non-decreasing Δ_2^0 function $g \leq f$ with an approximation $g[s]$ such that*

- *Each $g[s]$ is non-decreasing*
- *$g[s]$ approximates g from above (for all x, $\lim_{s\to\infty} g(x)[s] = g(x)$ and for all x and $s_1 < s_2$, $g(x)[s_1] \geq g(x)[s_2]$).*

Proof (of lemma). Take an unbounded non-decreasing Δ_2^0 function f, and some approximation $f[s]$ to it. First we recursively define approximations $h[s]$ to a function h such that each $h[s]$ is non-decreasing:

$$h(0)[s] = f(0)[s]$$
$$h(n+1)[s] = \begin{cases} f(n+1)[s] & \text{if } f(n+1)[s] \geq h(n)[s] \\ h(n)[s] & \text{otherwise} \end{cases}$$

We claim that this approximates the same function: for each n

$$\lim_{s\to\infty} f(n)[s] = \lim_{s\to\infty} h(n)[s].$$

We prove this by induction on n. For $n = 0$, it follows immediately from the construction of h. Now assume that for some n we have $\lim_{s\to\infty} h(n)[s] = \lim_{s\to\infty} f(n)[s] = f(n)$. Now look at $n+1$. Since $\lim_{s\to\infty} f(n+1)[s] = f(n+1)$, there is some stage s_1 such that for all $s \geq s_1$, $f(n+1)[s] = f(n+1)$. Since f is non-decreasing, we have $f(n) \leq f(n+1)$, and by the induction hypothesis there is some stage s_2 such that for all $s \geq s_2$, $h(n)[s] = f(n)$. Then we have for all $s \geq s_1, s_2$ that $h(n)[s] = f(n) \leq f(n+1) = f(n+1)[s]$. So by definition of h it follows that for all $s \geq s_1, s_2$, $h(n+1)[s] = f(n+1)[s]$. So $\lim_{s\to\infty} h(n+1)[s] = f(n+1) = \lim_{s\to\infty} f(n+1)[s]$. This concludes the inductive proof.

We define g using this function h, assuming that for all n, $h(n)$ has been assigned a value by stage n. We define g as follows:

$$g(n)[s] = \min\{h(n)[t] : n \leq t \leq s\}.$$

By construction this is an approximation from above. We prove that each $g[s]$ is non-decreasing. By construction of g for all stages t such that $n \leq t \leq s$, $h(n)[t] \geq g(n)[s]$. But each $h[t]$ is non-decreasing, so for all stages u with $n+1 \leq u \leq s$ it follows that $h(n+1)[t] \geq h(n)[t] \geq g(n)[s]$. So $g(n+1)[s] \geq g(n)[s]$.

Furthermore g must be non-decreasing since each $g[s]$ is. Suppose this is not true: for some n we have $g(n+1) < g(n)$. There are stages s_1 and s_2 such that for all $s \geq s_1$, $g(n)[s] = g(n)$ and for all $s \geq s_2$, $g(n+1)[s] = g(n+1)$. Now take some $s \geq s_1, s_2$. For this s we have $g(n+1)[s] = g(n+1) < g(n) = g(n)[s]$. This is a contradiction.

Finally we prove that g is unbounded. Suppose for a contradiction that for all n, $g(n) \leq c$ for some c. Since f is unbounded, we can find some n_0 such that $f(n_0) > c$. This means that for some s_0, for all $s \geq s_0$, $h(n_0)[s] = f(n_0) > c$. Since each $h[s]$ is non-decreasing, we have for all $n \geq n_0$ and for all $s \geq s_0$, $h(n)[s] \geq f(n_0)$. Now we let $m \geq s_0, n_0$ and it follows by definition of g that $g(m) \geq c$. □

This lemma gives us the opportunity to prove Theorem 4.9. We can do this in two different ways. One is rather short and constructs a Turing-complete c.e. set obeying such a function. The other directly constructs a non-K-trivial set that obeys the function using the approximation to the function. The latter gives a slightly better understanding of what is going on, so we will present this proof first. The proof uses a simple version of the *decanter method*, a common technique for constructing non-K-trivial sets, that is for example also used for proving that no K-trivial set can be Turing complete. A discussion of this method and that specific proof can be found in Appendix C.

Proof (of Theorem 4.9). Suppose that there is an unbounded non-decreasing Δ_2^0 gap function. We will construct a set X that satisfies

$$K(X \upharpoonright_n) \leq^+ K(n) + f(n) \tag{4.1}$$

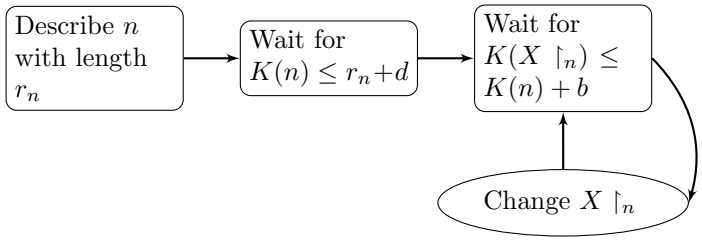
for all n, but is not K-trivial. In order to guarantee non-K-triviality, we need to satisfy requirements R_b for all b:

$$R_b : \exists m[K(X \upharpoonright_m) > K(m) + b].$$

For an arbitrary b we can satisfy the requirement R_b as follows: we build a machine M that describes numbers n. By the recursion theorem we know the index of the machine M in advance, so we know the coding constant d in the universal machine. So when we describe a number n with a string of length r_n in M, we can push down the complexity of $K(n)$.

Like in the proof of Theorem C.1, we assume for a contradiction that X is kept K-trivial by the universal machine. We then go through a process of describing n with short descriptions r_n, waiting for the machines reaction ($K(X \upharpoonright_n) \leq K(n) + b$). Then we change $X \upharpoonright_n$ so the machine needs to use up a new description. Since we force $K(n) \leq r_n + d$, the machine will run out of

short descriptions, which makes X non-K-trivial. This process is depicted in the diagram below.



While constructing X, we keep track of the *cost functions* for describing the initial segments of X. When the segment $X\upharpoonright_n$ changes, all segments up to the current stage s need a new description. The total weight that the universal machine needs to spend on this to keep the set K-trivial is given by the cost:

$$c_t(n,s) = \sum_{n\leq t\leq s} 2^{-(K(t)+b)}.$$

We want to make the total cost for this exceed 1, to get a contradiction and show that X cannot be K-trivial. At the same time, we want to have X obey the gap function. The cost to do this is given by cost c_f, which is always slightly lower because the required new descriptions can be a bit longer. It is given by the following:

$$c_f(n,s) = \sum_{n\leq t\leq s} 2^{-(K(t)+f(t)+c)}.$$

Note that we can rewrite $\sum_{n\leq t\leq s} 2^{-(K(t)+f(t)+c)} = \sum_{n\leq t\leq s} 2^{-K(t)}\cdot 2^{-(f(t)+c)}$ and $\sum_{n\leq t\leq s} 2^{-(K(t)+b)} = \sum_{n\leq t\leq s} 2^{-K(t)}\cdot 2^{-b}$. We can conclude that

$$c_f(n,s) \leq 2^{-(f(n)+c-b)}\cdot c_t(n,s). \tag{4.2}$$

Now we can make sure that X is not K-trivial by picking some n, describing it with a short string, and making at most 2^{b+d} changes in $X\upharpoonright_n$, as in the diagram above. We might not even need to make all these changes: we can stop as soon as the total cost for keeping the set K-trivial exceeds 1. Then the machine describing X will run out of short strings. The key to this argument will be to only do this on initial segments of X where the gap $f(n)$ is large enough (we will be more specific in the argument below). With (4.2) we can keep changing $X\upharpoonright_n$ to make the cost of keeping the set K-trivial exceed 1 (thus making X non-K-trivial), while keeping the cost of satisfying (4.1) below 1, making X obey f.

Suppose that we had a computable gap function f. Then finding the places where the gap is sufficiently large would be easy. Then it is enough to find one appropriate n to describe, and change $X\upharpoonright_n$ many times. We will work

this out below. When f is Δ^0_2, all we can do is pick some n such that the approximation at that stage gives a large enough gap. But the approximation might change, and as soon as the gap is no longer sufficient, we pick a new n and describe it with a longer string, and go through the diagram again.

If f is computable

First, we find a digit of X where the gap is large enough: larger than b (actually, larger than $b - c$ would already be enough). So we need $f(z) > b$. Note that such a z exists because f is unbounded, and we can find it because f is computable. Note that for all $n \geq z$, we have $f(n) > b$, since f is non-decreasing. We are going to describe the number m with a description of length 0 (we put $\langle 0, m\rangle$ into a request set L), and we need to make $2^{(0+b+d)}$ changes in X past the zth digit, so we choose $m = 2^{(b+d)} + z$ and make the required changes in $X \upharpoonright_m$.

Now note that the procedure stops as soon as the total cost for K-triviality (c_t) exceeds 1, while the total costs of satisfying (4.1) satisfies

$$Tc_f = \sum \{c_f(n, s) : n \text{ is put into } X \text{ at stage } s\}$$

and since for all n we put into X, with (4.2):

$$Tc_f \leq 2^{-(f(z)+c-b)} \cdot \sum \{c_t(n, s) : n \text{ is put into } X \text{ at stage } s\}.$$

By construction we have $f(z) > b$ so

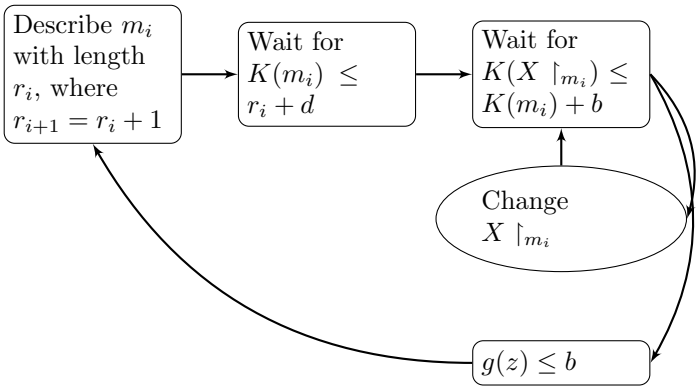
$$Tc_f < 2^{-1} \cdot \sum \{c_t(n, s) : n \text{ is put into } X \text{ at stage } s\}.$$

This will not exceed 1, because the loop stops right after the total cost for c_t does. So $K(X \upharpoonright_n) \leq^+ K(n) + f(n)$, but X is not K-trivial. This is a contradiction. Therefore, f cannot be computable.

If f is Δ^0_2

If f is Δ^0_2, we have an approximation $f[s]$ such that for all x, $f(x) = \lim_{s\to\infty} f(x)[s]$. With Lemma 4.10 we get another function g below f, such that $g[s]$ approximates g from above, and each $g[s]$ is non-decreasing.

We can use this to our advantage when we are constructing the set X. We use a similar approach as for the computable function, but as mentioned, the approximation might change and require us to describe a different number n. This makes our diagram look as below:



Now at a stage s we can find z such that $g(z)[s] > b$. We can do this since $g[s]$ is computable, and unbounded. Note that for all $n > z$ we have $g(n)[s] > b$, because $g[s]$ is non-decreasing. Now we will describe a number m_i with a description of length r_i. Since we might need to describe many m_i now, we will no longer choose $r_0 = 0$, but we will pick $r_0 = 1$ and put $r_{i+1} = r_i + 1$. This guarantees that the weight of the machine for describing m_i's will not be larger than $\sum_i 2^{-r_i} = 2 \cdot 2^{-r_0} = 1$. Since we will now need to make 2^{r_i+b+d} changes in X past the zth digit, we choose $m_i = 2^{r_i+b+d} + z$. Then we can make the required changes in $X \upharpoonright_{m_i}$. However, if at some stage s', $g(z)[s'] \leq b$, then we stop this procedure, and we go to $i+1$ where $r_{i+1} = r_i + 1$. We pick a new z_{i+1} such that $g(z_{i+1})[s'] \leq b$ and $m_{i+1} = 2^{r_{i+1}+b+d} + z_{i+1}$, and start the procedure again for this m_{i+1}.

Since $\lim_{s\to\infty} g[s] = g$, eventually we will be at a stage with a number z_i such that $g(z_i)[s] > b$ long enough to go through the loop untill the cost Tc_t exceeds 1. We let $c_g = \sum_{n\leq t\leq s} 2^{-(K(t)+g(t)+c)}$ and like before we have

$$Tc_g \leq 2^{-(g(z)+c-b)} \cdot \sum\{c_t(n,s) : n \text{ is put into } X \text{ at stage } s\}$$

so since $g(z) > b$

$$Tc_g \leq 2^{-1} \cdot \sum\{c_t(n,s) : n \text{ is put into } X \text{ at stage } s\}$$

and since we made $Tc_t = \sum\{c_t(n,s) : n \text{ is put into } X \text{ at stage } s\}$ just exceed 1, we can be sure that $Tc_g \leq 1$. So X satisfies $K(X \upharpoonright_n) \leq^+ K(n) + g(n)$, and since $g \leq f$ we have $K(X \upharpoonright_n) \leq^+ K(n) + f(n)$. But X is not K-trivial. This is a contradiction. Therefore, there cannot be a Δ^0_2 gap function. □

4.5 Another proof of Theorem 4.9

As promised we can also prove Theorem 4.9 by constructing a Turing complete c.e. set. We prove the following theorem, which is in our paper [BV10] in a slightly different version:

Theorem 4.11. *If f is Δ^0_2, computably approximable from above*[3] *and its limit is infinity ($\lim_{n\to\infty} f(n) = \infty$), then there is a Turing complete c.e. set which obeys f.*

We get the following corollary:

Corollary 4.12. *There cannot be an unbounded non-decreasing Δ^0_2 gap function for K-triviality.*

Proof. Suppose there were such a function f. With Lemma 4.10 we get a function g below f that is approximated from above. It is unbounded and non-decreasing, so it follows that $\lim_{n\to\infty} g(n) = \infty$. Applying the theorem gives a Turing complete c.e. set obeying g. Since g is below f, it also obeys f. However, by Theorem C.2, no K-trivial set can be Turing complete. So there is a set obeying f that is not K-trivial, and therefore f is not a gap function. □

For the proof of Theorem 4.11 we will construct the required Turing complete c.e. set using a construction that is very similar to a proof by Frank Stephan (see [Nie08, 183]).

Proof (of Theorem 4.11). We construct a Turing complete c.e. set A obeying f in stages, by making sure that for each n, the number of zeros in $A\upharpoonright_n$ is $\leq f(n)$. Then we can describe $A\upharpoonright_n$ with its length, and a string of length $f(n)$ describing[4] what places in $A\upharpoonright_n$ are 0. This will give us $K(A\upharpoonright_n) \leq^+ K(n) + f(n)$.

Furthermore, we code numbers from the halting problem into A, to make it Turing complete. First we present the construction of A, and then we will show that the complement of A is infinite, so that we can in fact code it this way.

Start the construction of A with the empty set in the first stage. At each stage we check for the least initial segment $A\upharpoonright_n$ such that $A\upharpoonright_n$ has more than $f(n)$ zeros. We put numbers into A to correct this, as many as necessary starting from the largest. Simultaneously, when m is the least number enumerated into $\emptyset'$ at this stage, enumerate the mth number (in order of magnitude) that is not yet in A (at that stage) into A.

The complement of A will be infinite if for every number c there is some initial segment of A such that c digits are zero. We prove that this is the case by induction on c. For $c = 0$ it trivially holds. Next assume that for some $c-1$

[3] f is computably approximable from above when there are computable functions $f[s]$ such that for all x, $\lim_{s\to\infty} f(x)[s] = f(x)$ and for all x, $s_1 < s_2$ we have $f(x)[s_1] \geq f(x)[s_2]$.

[4] If we have a description with the length n and the number of zeros k, we can do the following: we go through the approximation of A, which is just computable. As soon as we find $A[s]\upharpoonright_n$ with k zeros, it must be that $A[s]\upharpoonright_n = A\upharpoonright_n$ since A is a c.e. set.

we have some m such that there are precisely $c-1$ zeros in $A \upharpoonright m$. Let t be the stage where $A[t] \upharpoonright_m = A \upharpoonright_m$. Take $k \geq m, t$ such that for all $r \geq k$ and all s, $f(r)[s] > c$ (remember that each $f[s]$ is unbounded and non-decreasing, and it is an approximation from above). If c is enumerated into $\emptyset'$ at some stage, some number will be enumerated into A simultaneously, let this be v. If c is not in $\emptyset'$, let $v = 0$.

Now let $n \geq v+1, k+1$ be the least number such that $n-1 \notin A[k+1]$. Then if there are more than $c-1$ zeros in $A \upharpoonright_{n-1}$, we are done. Otherwise, it must be that all numbers strictly between m and $n-1$ are in A, since we assumed that $A \upharpoonright_m$ had $c-1$ zeros. Note that we picked $n \geq k+1$ so by the choice of k it follows that for all s, $f(n-1)[s] > c$. Therefore we will never encounter a stage where the number $n-1$ will be put into A by the first clause of the construction (fixing the number of zeros).

Furthermore, $n-1$ is by assumption the cth digit where A is zero. If it were put into A by the second clause of the construction (coding of $\emptyset'$), that could only happen because c would be enumerated into $\emptyset'$. However, we picked $n \geq v+1$ with v the number that is put into A when c is enumerated into $\emptyset'$. Therefore, $n-1$ will not be put into A by the second clause either.

It follows that $n-1$ can never end up in A. Since we have $c-1$ zeros in $A \upharpoonright_m$ by assumption, there will be c zeros in $A \upharpoonright_n$. This concludes the inductive proof, and shows that we can in fact code the halting problem into A to make A Turing complete. □

In Section 4.2 above we showed that an unbounded non-decreasing gap function induces a minimal pair in the K-degrees (Theorem 4.3). In this section we also proved that there is no unbounded non-decreasing Δ^0_2 gap function. Unfortunately these results together do not imply anything about there not being a Δ^0_2 minimal pair. For that, we would need to know that a minimal pair also induces an unbounded non-decreasing gap function.

A canonical way to define a gap function from a minimal pair A and B is the following:

$$f(n) := \min\{K(A \upharpoonright_n), K(B \upharpoonright_n)\} - K(n).$$

This function f is a gap function: take any set X such that $K(X \upharpoonright_n) \leq K(n) + f(n)$ for all n. It follows that $K(X \upharpoonright_n) \leq \min\{K(A \upharpoonright_n), K(B \upharpoonright_n)\}$, and X must be K-trivial.

Note however that the gap function as defined here is not necessarily unbounded. But one might wonder whether it is possible that the difference between $\min\{K(A \upharpoonright_n), K(B \upharpoonright_n)\}$ and $K(n)$ stays bounded. We pose the following question:

Question 4.13. Can there be non-K-trivial sets A and B such that

$$\min\{K(A \upharpoonright_n), K(B \upharpoonright_n)\} \leq^+ K(n).$$

This seems quite unlikely given results like Theorem 1.9 and Corollary 1.10. These say that as soon as the set S of numbers n such that $K(A\upharpoonright_n) \leq^+ K(n)$ contains an infinite computable set or an infinite c.e. set, then A will be K-trivial. If it were indeed impossible to have both sets non-K-trivial, the function f as defined above would be unbounded. In the conclusion we will suggest this as a topic for further research.

To summarize, we proved that there is no unbounded non-decreasing Δ^0_2 gap function, but this does not prove that no Δ^0_2 minimal pair can exist. It does show that the method of using gap functions for minimal pairs will not work for constructing a Δ^0_2 minimal pair.

Conclusion

This thesis studied arithmetical definability in the K-degrees. The main result we presented, was the construction of a non-K-trivial Δ^0_2 set that does not bound any non-K-trivial set in a given Δ^0_2 family of sets. In particular it follows that there is a non-K-trivial Δ^0_2 set that does not bound any non-K-trivial c.e. set. This is surprising because c.e. sets have quite low initial segment complexity.

Furthermore, that same result shows a structural difference between the LK-degrees and the K-degrees. In the LK-degrees, there is a result by Barmpalias that every nontrivial Δ^0_2 set bounds a nontrivial c.e. set [Bar10]. Extending that proof led to the result that for any pair of nontrivial Δ^0_2 sets, there is one nontrivial c.e. set below both. This shows that a Δ^0_2 minimal pair cannot exist in the LK-degrees [Bar10]. With our current result we see that this proof does not carry over to the K-degrees right away.

Similar to the result mentioned above, we also proved that for all $n > 1$, there is a non-K-trivial Σ^0_n set that does not bound any non-K-trivial Δ^0_n set. We constructed a non-K-trivial Σ^0_2 set that does not bound any non-K-trivial Δ^0_2 set, and we remarked that this construction relativizes to higher complexity classes. With the constructed Σ^0_2 set we also found a minimal pair in the K-degrees of lower complexity than what was known so far. We found a minimal pair consisting of this particular Σ^0_2 set with any c.e. set of non-zero K-degree.

The latter required the introduction of a new definition: that of infinitely often K-trivial sets. We studied their behaviour and we saw that each K-trivial set has a quite well-behaved lower cone: every set in the lower cone of an infinitely often K-trivial set Y is computable in $Y \oplus \emptyset'$. Furthermore we showed that the class of infinitely often K-trivial sets is uncountable, and we constructed a Π^0_1 class of them that does not contain any K-trivial sets.

Finally we studied gap functions for K-triviality. A result by Csima and Montalbán showed that an unbounded non-decreasing gap function f always induces a minimal pair of complexity $f \oplus \emptyset'$ in the K-degrees [CM06]. We studied several possible gap functions, and we proved that no unbounded

non-decreasing Δ_2^0 gap function can exist. This shows that this method is not suitable to construct a Δ_2^0 minimal pair in the K-degrees. This means that the question we mentioned in the introduction, whether there is a Δ_2^0 minimal pair in the K-degrees, remains open.

Suggestions for further research

An obvious suggestion for further research is the question whether there is a Δ_2^0 minimal pair in the K-degrees. In this thesis we have investigated some possible approaches to this already, finding that the method of constructing a minimal pair via a gap function for K-triviality will not work. If one could prove that a minimal pair always induces an unbounded non-decreasing gap function, this would, together with our results, imply that no Δ_2^0 minimal pair can exist. This relates to Question 4.13 we posed in the end of Chapter 4, whether it is possible to have sets A and B both non-K-trivial, and satisfy

$$\forall n \ \min\{K(A\upharpoonright_n), K(B\upharpoonright_n)\} \leq^+ K(n).$$

Another open question is which sets have countable lower cones in the K-degrees. We have seen that infinitely often K-trivial sets have lower cones that behave very nicely, but in general lower cones in weak reducibilities can be uncountable. Finally we have shown that an unbounded gap function for K-triviality can be Δ_4^0, but not Δ_2^0. A natural question is whether there can be an unbounded non-decreasing Δ_3^0 gap function or even one of Σ_2^0 degree.

A

Trees

This appendix contains some basic definitions and results concerning trees. They are used throughout the main text of this thesis. First we look at the definition of a tree, its complexity and the complexity of its infinite paths. After that we will turn to an alternative definition of trees. We show that it amounts to the same. Finally we define perfect trees and prove a lemma about turning a Π^0_1 tree into a computable tree with the same infinite paths, which we use in the proof of Theorem 2.9.

Definition A.1 (Tree). *A tree T is a set of strings that is closed under taking initial segments.*

We say that a tree is computable when it is computable to find whether a string is in the set or not. Similarly a tree can be of some other complexity class depending on how difficult it is to find whether a string is in the set. In many cases we are interested in the infinite paths through a tree:

Definition A.2 (Infinite path in a tree). *An infinite path through a tree T is a set P such that for all n, $P \upharpoonright_n \in T$. We denote the set of infinite paths in T as $[T]$.*

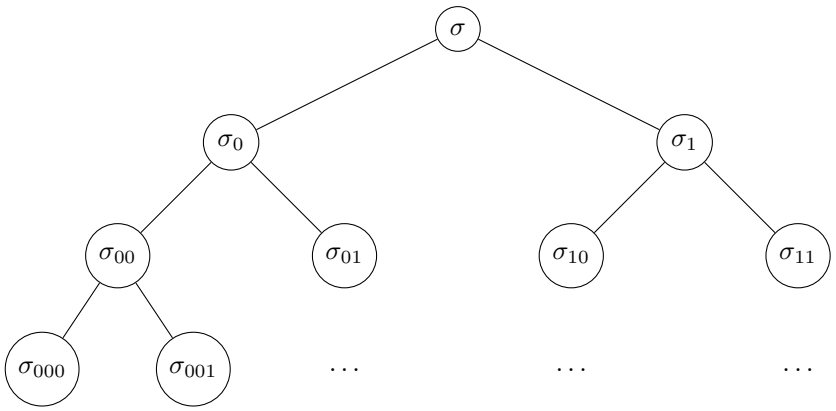
Note that in general when a tree T has complexity $\mathcal{C}$, an infinite path is of higher complexity because we add a quantifier. For example, infinite paths through computable trees are Π^0_1. However, in the case that the tree has only finitely many infinite paths, each of them is of the same complexity as the tree itself. This is because after some length the infinite path is completely determined (since the nodes will not branch anymore). We use this fact quite often, for example in the proof that all K-trivial sets are Δ^0_2 (they are the infinite paths through a Δ^0_2 tree of finite width, see Theorem 1.14).

Sometimes it is more convenient to use a different definition of a tree: we view it as a function.

Definition A.3 (Tree (alternative definition)). *A tree T is a partial function on strings ($2^{<\omega}$) such that*

$$\sigma \prec \tau \Rightarrow T(\sigma) \prec T(\tau)$$
$$\sigma|\tau \Rightarrow T(\sigma)|T(\tau).$$

The interpretation of this definition is the following: every node is coded with a string (λ (the empty string) is the root node, and its leftmost successor is 0 and the rightmost is 1). We map each string to a string such that it extends the string on the predecessor nodes, and it is incomparable to strings that are not predecessors or successors. If a node is indexed with b, we map it to a string σ_b as depicted below:



Note that a string on level n is not necessarily of length n, unlike in the previous definition of a tree. Now we define an infinite path in a tree as follows:

Definition A.4 (Infinite path in a tree). *An infinite path through a tree $T : 2^{<\omega} \to 2^{<\omega}$ is a set P such that for all n there is some $\sigma \in 2^{<\omega}$ of length n such that $T(\sigma) = P \upharpoonright_{|T(\sigma)|}$.*

The complexity of a tree is now defined as the complexity of this function T, and once again the infinite paths are usually of a complexity slightly higher, but when the tree has bounded width, all infinite paths are of the same complexity.

We note that the two Definitions A.1 and A.3 amount to the same when it comes to their infinite paths. Given a tree as a downwards closed set of strings, we can index all the nodes like in the picture above. Then we can map each index to the corresponding string to obtain a function T' as in Definition A.3. This function will have the same complexity and the same infinite paths.

For the other direction, we can take the closure of the range of the function T under initial segments. This way we obtain a set of strings T' of the same complexity as T. We show that the infinite paths of the two are the same. Take some infinite path in T, by definition this is also an infinite path in T'. Take an infinite path P in T', and suppose it is not in T. That means that

for some n, for all $m \geq n$, $P \upharpoonright_m \notin T$. But by construction then $P \upharpoonright_m$ cannot be in T' either, which is a contradiction. So $[T] = [T']$.

Furthermore we define the notion of a *perfect* tree.

Definition A.5 (Perfect tree). *A tree T is perfect if for all $\sigma \in T$ there are at least two proper extensions $\tau_1, \tau_2 \in T$ that are incomparable:*

$$\forall \sigma \in T \, \exists \tau_1, \tau_2 \in T (\tau_1 | \tau_2 \wedge \sigma \prec \tau_1 \wedge \sigma \prec \tau_2).$$

A perfect tree necessarily has uncountably many paths, since branches always split. We use this to prove that certain classes have uncountably many elements, such as the class of infinitely often K-trivial sets (Theorem 2.7).

Finally we include a lemma here that will be useful in constructing computable trees. It says that based on a Π^0_1 tree we can make a computable tree that has the same infinite paths.

Lemma A.6. *Given a Π^0_1 tree T, we can define a computable tree T' such that $[T] = [T']$.*

Proof. Take some Π^0_1 tree T. This means that we have some approximation $T[s]$ such that strings may be taken out after some stage. We can define a tree T' with an approximation $T'[s]$ such that at stage s only strings of precisely length s are put into the tree (and none are taken out), to make T' computable. The idea is to at each stage put all extensions of previously added nodes into T', unless they are an extension of something that was taken out of T. So instead of going back to shorter strings to take them out (like in the Π^0_1 tree), we make sure that the same branches still have dead ends at the current stage. It will be a different tree, but with the same infinite paths. Here is a more formal definition:

- $T'_0 = \{\lambda\}$
- $T'[s+1] = T'[s] \cup \{\sigma : |\sigma| = s+1 \wedge \forall n \leq s+1 (\sigma \upharpoonright_n \in T[s+1])\}$.

Note that T' will be a tree (a set of strings closed under taking initial segments), because when we put a string in, all its initial segments are in $T[s+1]$ and therefore they are in all $T[s']$ for all $s' \leq s$, so they were put into the tree T' at earlier stages. We prove that $[T] = [T']$. Take an infinite path $P \in [T]$. Then for any initial segment $P \upharpoonright_n$ it is in T_s for all s. So by construction for all s, $\forall n \leq s$, $P \upharpoonright_n \in T'[s]$, and it follows that $P \in [T']$. Next take an infinite path $Q \in [T']$, suppose $Q \notin [T]$. That means that for some n and s, $Q \upharpoonright_n \in T[s]$ but for all $s' > s$, $Q \upharpoonright_n \notin T[s']$. Then at stage $s+1$ of the approximation, $Q \upharpoonright_{s+1} \notin T'[s+1]$ and so $Q \upharpoonright_{s+1} \notin T'$. This is a contradiction, so $Q \in [T]$. So $[T] = [T']$. □

B

The infinite injury method

In this appendix we construct a c.e. minimal pair in the Turing degrees. This gives some insight into the construction of a minimal pair in general. Here we mainly use it as an illustration of the infinite injury method. We use this method in the proof of a Δ^0_2 not bounding any non-K-trivial set in a given Δ^0_2 family of sets in Section 3.3. This appendix is mostly based on [DH10, 44], where the following theorem and its proof can be found.

Theorem B.1 (Lachlan [Lac66], Yates [Yat66]). *There exist noncomputable c.e. sets A and B such that every set X computable in both A and B is computable.*

With a result from Merkle and Stephan (see [MS07]) and Theorem 1.11 this gives us a c.e. minimal pair in the C-degrees. Merkle and Stephan showed that for any $Y \subseteq \{2^{2^k} : k \in \mathbb{N}\}$, $X \leq_C Y$ implies $X \leq_T Y$. Therefore, if we take a minimal pair $A, B \subseteq \{2^{2^k} : k \in \mathbb{N}\}$ in the Turing-degrees, any set X that is C-below both will be Turing-below both. This makes that set computable, and with Theorem 1.11 it will satisfy $C(X \upharpoonright_n) \leq^+ C(n)$ for all n. Therefore A and B form a minimal pair in the C-degrees.

Before we get into the proof of Theorem B.1, we we briefly look into why we need a method as strong as the infinite injury method. We take a close look at the requirements we need to meet to have A and B as in the theorem. For each e we need to have (where Φ_e is an effective list of all Turing functionals):

$$R_e : \overline{A} \neq W_e$$
$$Q_e : \overline{B} \neq W_e$$
$$H_{\langle i,j\rangle} : \Phi_i^A = \Phi_j^B \text{ total} \Rightarrow \Phi_i^A \text{ computable.}$$

Requirements R_e and Q_e will make sure that A and B are noncomputable. By giving a c.e. enumeration for A and B that satisfies $H_{\langle i,j\rangle}$, we get the required minimal pair of c.e. sets. For notational convenience, we will adopt the following requirement N_e instead of $H_{\langle i,j\rangle}$:

$$N_e : \Phi_e^A = \Phi_e^B \text{ total} \Rightarrow \Phi_e^A \text{ computable.}$$

This is something we can do based on *Posner's trick* (see [DH10, 48]). Satisfying all N_e requirements guarantees the satisfaction of all $H_{\langle i,j\rangle}$ requirements: suppose that we have all N_e satisfied. Then we cannot have $A = B$, since they would then be computable because all N_e hold. Therefore there must be an n where they differ, without loss of generality assume that $n \in B$ and $n \notin A$. Then let $f \leq_T A, B$, that means that there must be i and j such that $f = \Phi_i^A = \Phi_j^B$. However, there is also a program that, for all oracles X, whenever $n \in X$, $\Phi_e^X = \Phi_j^B$, and whenever $n \notin X$, $\Phi_e^X = \Phi_j^A$. For this particular e we have

$$\Phi_e^A = \Phi_i^A = f = \Phi_j^B = \Phi_e^B.$$

and since all N_e are satisfied by assumption, if $\Phi_i^A = \Phi_j^B$ are total then Φ_e^A is computable. Therefore Φ_i^A is computable, and $H_{\langle i,j\rangle}$ is satisfied.

So we will meet the following requirements:

$$R_e : \overline{A} \neq W_e$$
$$Q_e : \overline{B} \neq W_e$$
$$N_e : \Phi_e^A = \Phi_e^B \text{ total} \Rightarrow \Phi_e^A \text{ computable.}$$

If we have just one strategy for each R_e and Q_e, these may be infinitely injured by the N_e strategies, which poses a problem. In this case alternative strategies for R_e and Q_e will work, but these cannot be used in the finite case. We will explain this in much more detail below. The solution will be to have a binary tree of strategies, where every node represents a certain strategy, and splits into two possibilities representing the two cases (finite and infinite injury). At a finite stage we have a path through the tree, based on the current outcomes of the nodes. The outcomes and therefore also this path might change at later stages. In the end there will be a leftmost infinitely often visited path, which will turn out to have successful strategies.

How to satisfy the requirements

The basic strategy to satisfy a specific R_e is the following: we pick a number x with the intention of putting it into A (x is a *follower*). We wait to see whether x is enumerated into W_e. If this never happens, we have $x \in \overline{A}$ but $x \notin W_e$, so R_e is satisfied. As soon as x is enumerated into W_e, we also enumerate it into A, and once again R_e is satisfied. An analog strategy will work for Q_e. The problem with $N_{e'}$ will be that it might not allow us to put this follower into the set, but we can circumvent this adapting our strategies for the different cases.

For meeting some particular N_e we define the following functions: the length of agreement function $l(e, s)$ shows until what length Φ_e^A and Φ_e^B have the same outcome (and both converge) at stage s. The maximum length of

agreement function $m(e,s)$ is the maximum length of agreement encountered so far, and the use function $u(e,s)$ is the longest length used in A or B to calculate the part up to where both agree. Here are the formal definitions (where ϕ_e is the use of a Turing functional Φ_e):

$$\begin{aligned} l(e,s) &= \max\{n : \forall k < n(\Phi_e^A(k)[s] \downarrow = \Phi_e^B(k)[s] \downarrow)\} \\ m(e,s) &= \max\{l(e,t) : t \leq s\} \\ u(e,s) &= \max\{\phi_e^A(l(e,t)-1)[t], \phi_e^B(l(e,t)-1[t] : t \leq s\} \end{aligned}$$

We say that a stage s is *e-expansionary* if the length of agreement is longer than any agreement encountered before: $l(e,s) > m(e,s-1)$. Then to satisfy N_e we will act on e-expansionary stages. Note that there will be infinitely many e-expansionary stages if in fact $\Phi_e^A = \Phi_e^B$ and they are both total. If they are different, there will only be finitely many, and this distinction is what the different strategies for R_e and Q_e will depend on.

Take some $n < l(e,s)$, that means that $\Phi_e^A(n)[s] \downarrow = \Phi_e^B(n)[s] \downarrow$. All we do now to satisfy N_e, is allow numbers to enter A (as a result of the strategies for R_e and Q_e), but freeze the initial segment of B up to $\phi_e^B(n)[s]$: we put a restraint of length $u(e,s)$ on B. We do this until we encounter the next e-expansionary stage. At the next e-expansionary stage t, we will again have $\Phi_e^A(n)[t] \downarrow = \Phi_e^B(n)[t] \downarrow$, and we also have $\Phi_e^B(n)[t] = \Phi_e^B(n)[s]$ since we have not changed B in the previous step. We lift the restraint on B and this time we freeze A up to $\phi_e^A(n)[t]$ (and we allow numbers into B), until the next e-expansionary stage u. This way we can be sure that $\Phi_e^A(n)[t] = \Phi_e^A(n)[u]$.

We interchange freezing A and B like this. Whenever $\Phi_e^A = \Phi_e^B$ is total, there will be infinitely many e-expansionary stages. By the above argument it follows that $\Phi_e^A(n) = \Phi_e^A(n)[s]$, so we can find the value at n at this finite stage. When we do this for all n, this makes Φ_e^A computable, so N_e is met.

Putting the strategies together

As mentioned above, we have strategies for N_e that put a restraint $u(e,s)$ (of not putting certain numbers in) on either A or B, thus compromising the strategies we have for satisfying $R_{e'}$ and $Q_{e'}$ of weaker priority. In fact the restrains may even go to infinity, since the length of agreement between A and B can go to infinity. That means that when we want to put a follower into A to satisfy $R_{e'}$, we are held back from doing so by the restraint that is part of the strategy for some N_e.

However, note that when this happens (the length of the agreement goes to infinity), we can be sure that there are infinitely many e-expansionary stages. So by construction we know that the restraint on A will at some point be lifted, while putting a restraint on B. At that moment we can act to satisfy $R_{e'}$ by putting a number into A. Note that when there are not infinitely many e-expansionary stages, then it may be that a restraint is never lifted, so we cannot use this strategy. However, we do know that if we look for a number

large enough we can put it into A, thus still satisfying $R_{e'}$. So in this case we will have to cancel the follower of $R_{e'}$ and look for a larger one. This way we have different strategies for these two cases, and analogously we have two strategies for satisfying $Q_{e'}$.

Finally we may have trouble putting the different strategies for N_e together. When N_e has infinitely many e-expansionary stages, our strategy for $R_{e'}$ of weaker priority would be to wait for a moment that the restraint on A is lifted. However, there may be another $N_{e''}$ putting restraints on A precisely when those of N_e are lifted. So again we distinguish between there being infinitely or finitely many e-expansionary stages: when there are finitely many we can do nothing, but when there are infinitely many, we act for $N_{e''}$ of weaker priority only when the restraints are nested. This will be possible, because there are infinitely many e-expansionary stages.

We put all these strategies in a binary tree: at each level e we have the strategies for R_e, Q_e and N_e. All nodes split into two possibilities: 0 corresponds to there being infinitely many e-expansionary stages, and attached to that node we have the corresponding strategies on that level. 1 corresponds to there being only finitely many e-expansionary stages, and that node has the strategies for that case attached to it. At a finite stage we will of course not know in what case we truly are, but we have a guess. So at each stage we have a current path through the tree, a path that may later on still change. After infinitely many stages there will surely be an infinite path that is infinitely often visited, and the leftmost such path will turn out to succesfully build the required A and B.

The formal proof

We can now put together the strategies as described above, and thus prove Theorem B.1.

Proof (of Theorem B.1). We need to satisfy the requirements

$$\begin{aligned} R_e &: \overline{A} \neq W_e \\ Q_e &: \overline{B} \neq W_e \\ N_e &: \Phi_e^A = \Phi_e^B \text{ total} \Rightarrow \Phi_e^A \text{ computable.} \end{aligned}$$

We put all strategies on a tree $T = \{0,1\}^{<\omega}$. We have the following correspondence:

$$\begin{aligned} 0 &: \text{ there are infinitely many } e\text{-expansionary stages} \\ 1 &: \text{ there are finitely many } e\text{-expansionary stages} \end{aligned}$$

For each $\sigma \in T$ we assign strategies N_σ, R_σ, Q_σ for satisfying requirements $N_{|\sigma|}$, $R_{|\sigma|}$, and $Q_{|\sigma|}$, respectively. We define the notion of σ-stage, $m(\sigma, s)$ and an σ-expansionary stage inductively:

Definition B.2 (σ-stage).

- *Every stage is a λ-stage (λ is the root)*
- *If s is a τ-stage. Let $e = |\tau|$. Then*

$$m(\tau, s) = \max\{l(e, t) : t < s \text{ is a } \tau\text{-stage}\}$$

If $l(e, s) > m(\tau, s)$, we call s τ-expansionary, and declare s to be a $\tau * 0$-stage. Otherwise, we declare s a $\tau * 1$-stage.

Finally let δ_s be the path known at stage s: the unique σ of length s such that s is a σ-stage. Now, we say that R_σ (and analogously Q_σ) *requires attention* at a σ-stage $s > |\sigma|$ if $W_{|\sigma|,s} \cap A_s = \emptyset$ and one of the following holds:

- R_σ currently has no follower
- R_σ has a follower $x \in W_{|\sigma|,s}$

Now we construct the sets A and B as follows:

- At stage s, compute δ_s. Initialize all strategies on nodes to the right of δ_s, which means that all R- and Q-strategies forget about their current followers and will have to pick new larger followers, larger than any number seen so far in the construction. This is the influence of the N-requirement on δ_s: it requires no numbers to be put into A and B below the use so far. We do this only to the right of the current path, because that corresponds to the guess that there are finitely many e-expansionary stages.
- Find the strongest R- or Q-requirement whose strategy requires attention at stage s. If this is for example R_e (the case for Q_e is analogous) with corresponding strategy R_σ (by the conditions above s must be a σ-stage), we say that R_σ *acts* at stage s. If R_σ currently has no follower, appoint a new large follower. Otherwise, enumerate the current follower into A. Initialize all strategies attached to nodes properly extending σ.

In the two lemmas below we will verify that the leftmost infinitely often visited path δ satisfies all requirements. The true path δ is the leftmost infinitely often visited path:

$$\sigma \prec \delta \Leftrightarrow \exists^\infty s(\sigma \prec \delta_s) \wedge \exists^\infty s(\delta_s <_{\text{lex}} \sigma)$$

Lemma B.3. *Each R- and Q-requirement is met.*

Proof. Take some arbitrary e and consider R_e (Q_e is analogous). Let $\sigma = \delta \upharpoonright_e$. Assume inductively that strategies attached to proper prefixes of σ only act finitely often. We prove that R_σ is successful. Let s be the least stage such that

- no strategy attached to a proper prefix of σ acts after stage s (we can find this because of the induction hypothesis)
- after stage s, δ_t for $t > s$ never moves to the left of σ.

Since s is picked to be the least such stage, it follows that R_σ must be initialized at stage s. Therefore, as soon as another σ-stage $t > s$ is encountered, a new follower x will be appointed for R_σ. After that there is no way to initialize R_σ again, since none of the strategies with proper prefixes of σ or nodes to the left will be visited. This follower x will not be put into A if it does not enter W_e. But as soon at it does (say, at some stage u), at the next σ-stage $v > u$ (and there will be one since this is the infinitely often visited path), x will be enumerated into A. In any case, R_e will be satisfied. Finally R_σ has only acted twice since stage s. Also strategies R_e are never initialized, so the inductive argument goes through. □

Lemma B.4. *Each N-requirement is met.*

Proof. Pick some arbitrary e. Let $\sigma = \delta \upharpoonright_e$. If $\sigma * 1 \prec \delta$, that means there are only finitely many e-expansionary stages, so $\Phi_e^A \neq \Phi_e^B$ and N_e is met. So suppose that $\sigma * 0 \prec \delta$, and that Φ_e^A is total. Let s be the least stage such that

- no strategy attached to a proper prefix of $\sigma * 0$ acts after stage s
- after stage s, δ_t for $t > s$ never moves to the left of σ.

We show that Φ_e^A is computable: to compute $\Phi_e^A(n)$, find the least $\sigma * 0$ stage $t_0 > s$ such that $l(e, t_0) > n$ (which exists, because there are infinitely many e-expansionary stages). Let $t_1 < t_2 ...$ be the $\sigma * 0$-stages greater than or equal to t_0. They are e-expansionary, so we have $\Phi_e^A(n)[t_i] = \Phi_e^B(n)[t_i]$ for all i. Also for each i we have either $\Phi_e^A(n)[t_{i+1}] = \Phi_e^A(n)[t_i]$ or $\Phi_e^B(n)[t_{i+1}] = \Phi_e^B(n)[t_i]$. To see that this does hold, fix some i. At stage t_i, all strategies attached to nodes on the right of $\sigma * 0$ will be initialized. So any follower for such a strategy must be a number greater than any number seen so far. In particular it must be larger than $\phi_e^A(n)[t_i]$ and $\phi_e^B(n)[t_i]$. By the choice of s no strategies above or to the left of σ can act after stage t_i. So these strategies, that have such large followers, are the only ones that can put numbers into A or B. They can only act at $\sigma * 0$-stages, and only one strategy will be able to act at such a stage. So only one number can enter either $A \upharpoonright_{\phi_e^A(n)[t_i]}$ or $B \upharpoonright_{\phi_e^B(n)[t_i]}$ before stage t_{i+1}. So either $\Phi_e^A(n)[t_{i+1}] = \Phi_e^A(n)[t_i]$ or $\Phi_e^B(n)[t_{i+1}] = \Phi_e^B(n)[t_i]$.

It follows that $\Phi_e^A(n)[t_0] = \Phi_e^A(n)[t_1] = ...$ So $\Phi_e^A(n) = \Phi_e^A(n)[t_0]$ and $\Phi_e^A(n)$ is computable. □

These two lemmas conclude the proof. □

C

The decanter method

A set is K-trivial if and only if it is low for K (or equivalently low for ML-randomness [Nie08, 170]). One implication is not too difficult: every set that is low for K is K-trivial (see Proposition 5.2.3 in [Nie08, 177]). The other direction requires an intricate proof, that can be found in [Nie08]. This appendix is dedicated to the basic method behind that proof, the so-called *decanter method*, due to Downey, Hirschfeldt, Nies and Stephan [DHNS02]. By extending that to the so-called *golden run method*, Nies (together with Hirschfeldt) was able to prove that every K-trivial set is superlow, and that every K-trivial set is low for K [Nie05], which is the claim above.

In this appendix we only treat the decanter method, based on [Nie08, 201]. We illustrate this method with the proofs of the following theorems:

Theorem C.1 (Downey, Hirschfeldt, Nies and Stephan [DHNS02]). *No K-trivial set can be weak truth-table complete.*

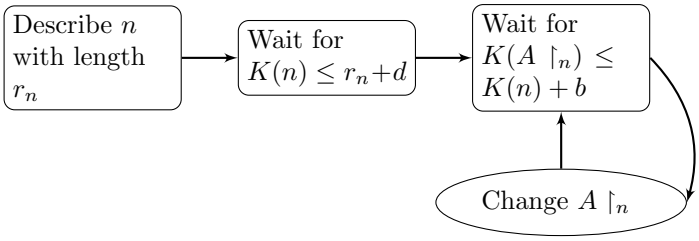
Theorem C.2 (Downey, Hirschfeldt, Nies and Stephan [DHNS02]). *No K-trivial set can be Turing-complete.*

The methods of these proofs are also used in Section 4.4, where we construct a non-K-trivial set below an unbounded non-decreasing Δ^0_2 gap function (see Section 4.4).

The technique relies on a construction to make a set non-K-trivial. In case of proving that a K-trivial set cannot be Turing complete we assume that there is one that is, and derive a contradiction by making this particular set be non-K-trivial.

The method to make A non-K-trivial is roughly the following, as depicted in the diagram below: first we describe n with a string of length r_n (box 1). We can do this using the Kraft-Chaitin theorem, which will give us a machine with coding constant d, that we know in advance using the recursion theorem. Now we wait for a stage s such that the universal machine describes it with the required length: $K(n) \leq r_n + d$ (box 2). Then if A were K-trivial via constant b, it would need a short description: we wait for this $K(A \upharpoonright_n) \leq K(n) + b$

(box 3). After that, we change A in the first n digits (oval). This requires a new description of $A\upharpoonright_n$. However, there are only 2^{r_n+b+d} strings of length $\leq r_n+b+d$ (and we forced $K(n)\leq r_n+d$). So if we make enough changes, this loop will eventually not go through, which shows that A is non-K-trivial.



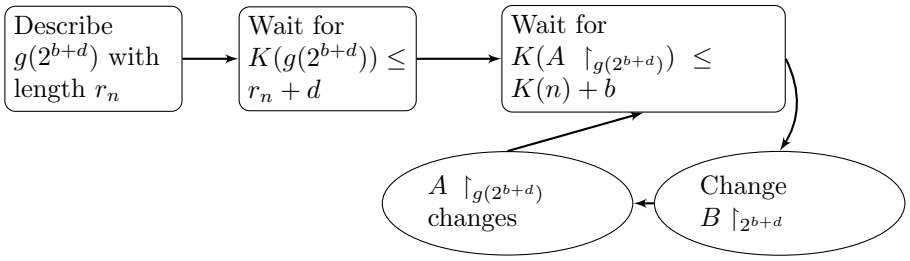
Note that in the case of Turing completeness we are in fact not free to construct a set A as we like. All we have is the assumption that there would be some K-trivial set A that is Turing complete. However, the assumption that it is Turing complete will give us the opportunity to make the required changes in A. To see how this works, it is instructive to first look at a simpler case, namely that no K-trivial set can be weak truth table complete. We will go into both proofs below.

C.1 No K-trivial set is weak truth-table complete

In this section we prove Theorem C.1.

Proof (of Theorem C.1). Take some arbitrary K-trivial set A, assume that it is K-trivial with constant b (for all n, $K(A\upharpoonright_n)\leq K(n)+b$). Since A is K-trivial it must be Δ^0_2 (by Theorem 1.14), so we have a computable approximation $A[s]$ to A. Suppose for a contradiction that A is weak truth-table complete. That is, we have $\emptyset'\leq_{\text{wtt}} A$. In order to show that A is K-trivial, we will construct a machine M_d with the Kraft-Chaitin theorem, to describe numbers n with r_n. The coding constant d for this machine is known in advance by the recursion theorem, so we can use it to force $K(n)\leq r_n+d$. Then if A is K-trivial we must have $K(A\upharpoonright_n)\leq K(n)+b\leq r_n+d+b$ for all n. However, we will reach a contradiction concerning the number of times $A\upharpoonright_n$ needs to be described, using that A is wtt-complete.

Since by assumption A is wtt-complete, we have $\emptyset'\leq_{\text{wtt}} A$. For every c.e. set B, we have $B\leq_m\emptyset'$. So we get a wtt-reduction $B\leq_{\text{wtt}} A$. In other words, $B=\Gamma^A$ for some Turing functional Γ^A, and we have a computable function $g(x)$ such that $\forall x\ \text{use}(\Gamma^A(x))\leq g(x)$. This holds for every c.e. set, so the next step in this proof is to construct a smart c.e. set B and force A to change as we put elements into B.



We will describe a cleverly picked number n with a string of length 0. To have A be K-trivial, that means that the universal machine must describe $A\upharpoonright_n$ with a string of length at most $b+d$. Observe that there are only 2^{b+d} strings of this length, and if we can cause that many changes in $A\upharpoonright_n$, the machine will need at least $2^{b+d}+1$ strings to describe each initial segment $A\upharpoonright_n$. At this point, it is important to note that if we want to cause 2^{b+d} changes via B, we will need to look at the initial segment of A up to the use of 2^{b+d}. In other words, if we cause changes in $B\upharpoonright_{2^{b+d}}$, they will result in changes in $A\upharpoonright_{(\mathrm{use}(\Gamma^A(2^{b+d})))}$.

Now we pick $n = g(2^{b+d})$, since this is an upper bound for the use. We let $L = \{\langle 0, n\rangle\}$, which is a bounded c.e. request set, so there is a machine that describes n with a string of length 0. Now we construct B as follows: wait for a stage t such that $B\upharpoonright_{2^{b+d}} = \Gamma^A\upharpoonright_{2^{b+d}}[t]$. Since $A[t]$ approximates A, its initial segment should have a short description. So we wait for the moment that $K(A\upharpoonright_n)[t] \leq b+d$. Then we put the largest number less than 2^{b+d} that is not yet in B, into B.

Note that by putting this number into B, we make sure that, since $B = \Gamma^A$, it must be that $A\upharpoonright_{(\mathrm{use}(\Gamma^A(2^{b+d})))} \neq A[t]\upharpoonright_{(\mathrm{use}(\Gamma^{A[t]}(2^{b+d})))}$. By our choice of n, this means that $A\upharpoonright_n \neq A[t]\upharpoonright_n$. Now we repeat the same construction: put another number less than 2^{b+d} that is not yet in B, into B.

By following this procedure, we need the universal machine to describe 2^{b+d} many initial segments $A[s]\upharpoonright_n$, before describing the correct $A\upharpoonright_n$. All these descriptions need to be of length 2^{b+d}. So the weight is $(2^{b+d}+1)\cdot 2^{-(b+d)} > 1$. This is a contradiction. □

C.2 No K-trivial set is Turing complete

Proving that no K-trivial set can be Turing complete (Theorem C.2) requires some fine tuning of the above strategy. We no longer have a computable bound on the use function, so we cannot pick a smart number to describe in advance. Instead we need to describe some number and pick a new one as soon as the use changes. This will add to the weight of the machine we build, so we have to do this carefully.

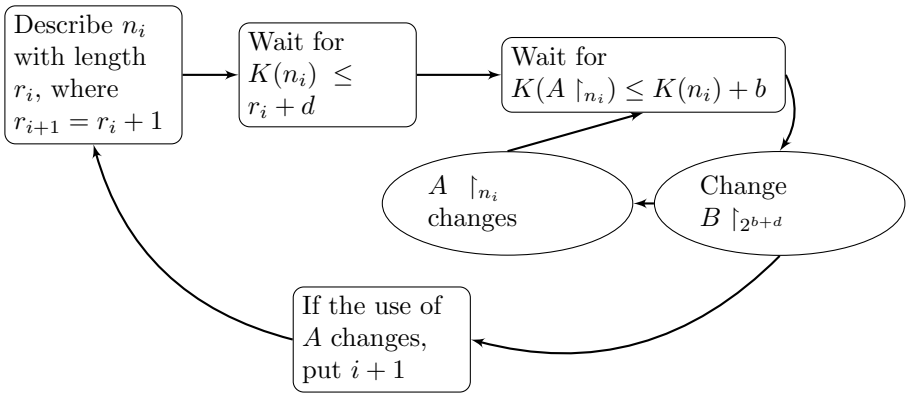
Proof (of Theorem C.2).

The idea

Just like in the previous proof, we take a set A that is K-trivial with constant b. We have an approximation $(A[s])$. We will build a machine M_d (this index is known in advance) to describe numbers n. We suppose towards a contradiction that A is Turing complete: $\emptyset' \leq_T A$. Thus for any c.e. set B we have $B = \Gamma^A$. We will build a clever set B to cause sufficiently many changes in A to get a contradiction with the weight of the machine M_d.

As pointed out earlier, the difference with the previous proof is that we no longer have a computable bound on the use of the Turing functional. That means that if we make changes in B and we require a description of $A \upharpoonright_n$, the changes in B might have no effect on this initial segment because it only uses larger numers from A. There is no clever n we can pick to avoid this scenario. But since we know that $B = \Gamma^A$, the use will eventually stabilize.

Our strategy will be to keep picking a new n whenever the use of A changes, and wait for the use to stabilize. The idea is to no longer describe n with a description of length 0, since this would immediately use up all the weight of the machine M_d. Since we need to describe (many) more n now, this cannot happen. Instead, we first describe n_0 with a string of length r_0 quite large, by enumerating $\langle r_0, n_0 \rangle$ into L. When the above case occurs, and we need to change to a new n_1, we describe this with a string of length $r_1 = r_0 + 1$. In this way, whenever we 'spill' weight at step i, it will only be 2^{r_i} much. We will see later that in the worst case we could only spill at most once for each index i, so the total wasted weight $\sum_i 2^{-r_i} = \sum_i 2^{-(r_0+i)} = 2^{-r_0} \sum_i \cdot 2^{-i} = 2 \cdot 2^{-r_0}$ can be kept below any value by picking r_0 large enough. The diagram now looks as follows:



From now on we will distinguish between numbers n for which the regular loop is ended because the use changes, and the n such that in fact the loop ends because there are no more short descriptions. In the latter case the strategy is successful. In the first case we 'waste' weight because we describe the number itself, but no extra weight is required to describe $A \upharpoonright_n$. We put such n into

a set F_1, called a 1-set because only one description of $A\upharpoonright_n$ is needed. The n where first some different $A[s]\upharpoonright_n \neq A\upharpoonright_n$ requires a description, resulting in two descriptions of the same length, are put into a set F_2, called a 2-set.

First case

For a moment consider the (impossible) case where $b = d = 0$. Then we could go through numbers n long enough to have the total weight[1] of the numbers in F_2 be some value larger than $\frac{1}{2}$ but smaller than 1. This way we keep the weight of the machine describing the natural numbers low enough (less than 1), while the weight of the machine describing the initial segments of A is twice this value (because for each n we needed 2 descriptions), so larger than 1.

The procedure. Formally, the procedure described above is the following, which we call $P_2(p, q)$. Here p is the *goal* (the weight of F_2 that we need to be between $\frac{1}{2}$ and 1), and q is the *garbage quota* (the weight of the numbers n gone to waste). Together, p and q should add up to some value less than 1, and we assume that $q = 2^{-c}$ for some c. Then the procedure is as follows (starting out with empty sets F_0, F_1, F_2):

(1) Choose a large[2] number m.
(2) Wait for $\Gamma^A(m)$ to converge
(3) Let i be the number of times that the procedure has gone through (2). Then pick a large number n_i and let $r_i = c+i$. Put $\langle r_i, n_i\rangle$ into the request set L, and put n_i into a 0-set F_0.
Wait for a stage t such that $K(A\upharpoonright_n)[t] \leq K(n)[t]+b+d$ (for now $b = d = 0$) and take n_i out of F_0 and put it into F_1 (since an initial segment for A of this length is now described once). Note that if M_d is a machine for L, then such a stage t exists. If weight$(F_1 \cup F_2) \leq p$, GOTO (3).
If $\Gamma^A(m)$ changes during this loop the numbers that are already in F_1 will still need two different descriptions, so we can take all of them and put them into F_2. The number n_i in F_0 is now garbage, since we cannot be sure that it results in two different descriptions of $A\upharpoonright_{n_i}$. So put $F_0 = \emptyset$ and GOTO (2).
(4) Put m into B. This causes A to change, so everything that is now in F_1 needs another description: we put all these numbers into F_2 and put $F_1 = \emptyset$.

Verification. To see that this produces a bounded request set L, note that we keep the weight of $F_1 \cup F_2 \leq p$. The weight that is wasted through F_0, is at most r_i once for each i. By our choice of these r_i, we get that the weight of

[1] the weight of a subset F of $\mathbb{N}$ is formally defined as follows: if L is the request set, consisting of pairs $\langle r, n\rangle$, then the weight of F is $\sum_{\{r:\langle r,n\rangle\in L \wedge n\in F\}} 2^{-r}$.

[2] By a large number we mean a number larger than any number seen in the construction so far.

L is at most $p + \sum_{i\geq 1} 2^{-r_i} = p + \sum_{i\geq 1} 2^{-(c+i)} = p + 2^{-c} \cdot \sum_{i\geq 1} 2^{-i} = p + q$, which we took to be less than 1.

Next, notice that Γ^A is total, so we will at some point reach (4). Then we get that the weight of F_2 is p, the goal, which we took to be greater than $\frac{1}{2}$. Since all of of the numbers n_i in F_2 require two descriptions of initial segments of that length, the weight of the machine describing those exceeds 1. This is a contradiction, so A is non-K-trivial.

Second case

The procedure as described above only works for the impossible case where $b = d = 0$. In fact, if we want to get a contradiction with the K-triviality of A where for each n we need descriptions such that $K(A\upharpoonright_n) \leq K(n)+b \leq r+b+d$ (where r is the length of the description of n), then we need to make sets of n such that at least 2^{b+d+1} many descriptions of $A\upharpoonright_n$ are needed, instead of just 2 as before. So instead of making a 2-set, we now need to make a 2^{b+d+1}-set of weight more than $\frac{1}{2}$.

The strategy to do this is to define procedures that make a k-set of weight p_k by calling P_{k-1} a few times. Ultimately, this will go down to the level of P_2, which is as defined above. The problem is that while calling a different procedure, the use of A may change. This cancels the procedure and calls the same procedure again, but with a smaller garbage quota. Like before, this guarantees that the total garbage quota is not exceeded.

If a procedure P_i is not cancelled, it picks a number m to put into B, and calls a procedure with a lower index. Finally this will reach P_2, which actually does the work by describing appropriate numbers n. Note that during this procedure, each new (lower) level picks a new number m to put into B. So if we start on procedure of level 2^{b+d+1}, this many numbers will be put into B, causing that same number of changes, and each of these changes result in different initial segments of A, because only then are the procedures not cancelled, and can they end up at P_2. So then really 2^{b+d+1} different strings are needed to describe initial segments of A, which is a contradiction.

The procedures. Formally, the procedure $P_i(p, q)$ is the following (for $i > 2$, and with $q = 2^{-c}$ for some c):

(1) Choose some large number m
(2) Wait for $\Gamma^A(m)$ to converge
(3) Let j be the number of times the procedure has gone through (2). CALL $P_{i-1}(2^{-j}q, q')$ where $q' = \min(q, 2^{-(w_{i-1}+i+2)})$ where w_{i-1} is the number of P_{i-1}-procedures started so far.
IF weight$(F_{i-2} \cup F_{i-1}) < p$ GOTO (3)
If the $\Gamma^A(m)$ changes during this loop, cancel the run of all subprocedures and GOTO (2). Note that like in the P_2-case, what is in F_{i-1} has to be described again because of this change, so all these numbers can be put into the i-set F_i. The numbers in F_{i-2} go to waste. Declare $F_{i-2}, F_{i-3}, ..., F_0$ empty.

(4) Put m into B. Now we can put F_{i-1} into F_i and put $F_{i-1} = \emptyset$.

Now for proving the theorem we call $P_k(\frac{3}{4}, \frac{1}{8})$ with $k = 2^{b+d+1}$. Now again by the choice of the garbage quotas, which we make smaller each time we request a new procedure, we can be sure that the weight of the request set is less than or equal to 1. And in the end we have a 2^{b+d+1}-set of weight $\frac{3}{4}$, because this was the goal of the procedure. Now the weight of the machine describing initial segments of A is at least

$$\frac{3}{4} \cdot (2^{b+d+1}) \cdot 2^{-(b+d)} = \frac{3}{4} \cdot 2 > 1.$$

This is a contradiction, and therefore A cannot be K-trivial. So no K-trivial can be Turing complete. □

References

Bar10. G. Barmpalias. Elementary differences between the degrees of unsolvability and degrees of compressibility. *Annals of Pure and Applied Logic*, 161:923–934, 2010.

BLS08. G. Barmpalias, A.E.M. Lewis, and M. Soskova. Randomness, lowness and degrees. *Journal of Symbolic Logic*, 73:559 –577, 2008.

BV10. G. Barmpalias and C.S. Vlek. Kolmogorov complexity of initial segments of sequences and arithmetical definability. Not yet published, 2010.

Cha75. G. Chaitin. A theory of program size formally identical to information theory. *Journal of the ACM*, 22:329–340, 1975.

Cha76. G. Chaitin. Information-theoretical characterizations of recursive infinite strings. *Theoretical Computer Science*, 2:45–48, 1976.

CM06. B.F. Csima and A. Montalbán. A minimal pair of K-degrees. *Proceedings of the American Mathematical Society*, 134:1499–1502, 2006.

DH10. R. Downey and D.R. Hirshfeldt. *Algorithmic Randomness and Complexity*. Springer, 2010. Not yet published.

DHL04. R. Downey, D. Hirschfeldt, and G. LaForte. Randomness and reducibility. *Journal of Computer and System Sciences*, 68:96–114, 2004.

DHNS02. R. Downey, D. Hirschfeldt, A. Nies, and F. Stephan. Trivial reals. *Electronic Notes in Theoretical Computer Science*, 66, 2002.

DK87. O. Demuth and A. Kučera. Remarks on 1-genericity, semigenericity, and related concepts. *Commentationes Mathematicae Universitatis Carolinae*, 28:85–94, 1987.

HKM09. R. Holz, T. Kraling, and W. Merkle. Time-bounded Kolmogorov complexity and Solovay functions. In *Mathematical foundations of computer science*, pages 392–402. Springer, 2009.

Lac66. A. Lachlan. Lower bounds for pairs of recursively enumerable degrees. *Proceedings of the London Mathematical Society*, 16:537–569, 1966.

Lev71. L. Levin. *Some Theorems on the Algorithmic Approach to Probability Theory and Information Theory*. PhD thesis, Moscow, 1971.

Lev73. L. Levin. On the notion of a random sequence. *Soviet Mathematics Doklady*, 14:1413–1416, 1973.

LV93. M. Li and P. Vitányi. *An introduction to Kolmogorov complexity and its applications*. Springer, 1993.

Mil09. J.S. Miller. The K-degrees, low for K degrees, and weakly low for K sets. *Notre Dame Journal of Formal Logic*, 50:381–391, 2009.

MS07. W. Merkle and F. Stephan. On C-degrees, H-degrees and T-degrees. In *Twenty-Second Annual IEEE Conference on Computational Complexity (CCC 2007)*, pages 60–69, 2007.

MY08. J.S. Miller and L. Yu. On initial segment complexity and degrees of randomness. *Transactions of the American Mathematical Society*, 360:3193–3210, 2008.

MY10. J.S. Miller and L. Yu. Oscillations in the initial segment complexity of random reals. Not yet published, 2010.

Nie05. A. Nies. Lowness properties and randomness. *Advances in Mathematics*, 197:274–305, 2005.

Nie08. A. Nies. *Computability and randomness*. Clarendon Press, 2008.

Sch73. C. P. Schnorr. Process complexity and effective random tests. *Journal of Computer and System Sciences*, 7:376–388, 1973.

Sol64a. R. Solomonoff. A formal theory of inductive inference I. *Information and Control*, 7:1–22, 1964.

Sol64b. R. Solomonoff. A formal theory of inductive inference II. *Information and Control*, 7:224–254, 1964.

vL87. M. van Lambalgen. *Random Sequences*. PhD thesis, Universiteit van Amsterdam, 1987.

Yat66. C. E. M. Yates. A minimal pair of recursively enumerable degrees. *The Journal of Symbolic Logic*, 31:159–168, 1966.

Zam90. D. Zambella. On sequences with simple initial segments. Technical Report ML-1990-05, ILLC, University of Amsterdam, 1990.

Index