Proceedings of the International Congress of Mathematicians
Hyderabad, India, 2010

Interactions of Computability and Randomness

André Nies*

Abstract

We survey results relating the computability and randomness aspects of sets of natural numbers. Each aspect corresponds to several mathematical properties. Properties originally defined in very different ways are shown to coincide. For instance, lowness for ML-randomness is equivalent to K-triviality. We include some interactions of randomness with computable analysis.

Mathematics Subject Classification (2010). 03D15, 03D32.

Keywords. Algorithmic randomness, lowness property, K-triviality, cost function.

1. Introduction

We will study sets of natural numbers. We refer to them simply as *sets*. Sets can be identified with infinite sequences of bits. Co-infinite sets can also be identified with real numbers in $[0, 1)$ via the binary representation.

We consider two aspects of a set, its *computational complexity* and its *randomness*. The principal observation is that these two aspects interact closely with one another.

The traditional interaction is from computability to randomness. One uses algorithmic methods to define and study randomness notions [43, 26, 27]. We will show that notions introduced in very different computability-theoretic ways coincide.

The converse interaction was discovered later. Concepts originating from randomness enrich computability theory [4, 19, 34]. We will give examples of this interaction through the study of *lowness properties* of a set A. Such a property specifies a sense in which A is close to being computable. Often this

*Partially supported by the Marsden Fund of New Zealand, grant no. 08-UOA-184
André Nies, Dept. of Computer Science, University of Auckland, Private Bag 92019, Auckland, New Zealand. E-mail: andre@cs.auckland.ac.nz.

is understood via the *weak-as-an-oracle* paradigm: A is weak in a specific sense when used as an oracle set in a Turing machine computation. Randomness-related concepts have led to two new paradigms of lowness [37, 31, 13].

The *Turing-below-many* paradigm says that A is close to being computable because it is easy for an oracle set to compute it, in the sense that the class of oracles computing A is large. Here, a class of oracles is considered large if it contains random sets of a certain kind. So far, all the sets that satisfy an instance of the Turing-below-many paradigm are Δ^0_2.

The *inertness* paradigm says that a set A is close to computable because it is computably approximable with a small number of changes. In particular, such a set is Δ^0_2 (see the Limit Lemma 2.1 below). To formalize the inertness paradigm, we use so-called *cost functions*. They measure the total number of changes of a Δ^0_2 set, and especially that of a computably enumerable set. Most examples of cost functions are based on randomness-related concepts.

In Sections 4-6, we will show that various lowness properties coincide. We introduce the K-trivial sets, and the strongly jump-traceable sets. For each class we give characterizations via all three lowness paradigms.

For some more motivation and background in non-technical language see [32]. For detailed background see [37, 8]. Most sections end with a summary and some interesting further facts. The keen student may want to prove some of these facts as exercises.

2. Some Background from Computability Theory

We assume that the reader knows the basics of computability theory, such as the notions of a computable set, a computably enumerable (c.e.) set, Turing reducibility $\leq_{\mathrm{T}}$, relativization, and (to some extent) Turing functionals. See [41] or [37, Ch. 1].

The capital letters A, B, X, Y, Z denote sets of natural numbers, simply called *sets* in what follows. For an "oracle" set A, we let $J^A(x)$ be the value on input x of a universal partial A-computable function. For instance, let $(\Phi_e)_{e\in\mathbb{N}}$ be an effective listing of all Turing functionals and let $J^A(x) = \Phi^A_x(x)$ (equality extends to the value 'undefined'). The domain of J^A is denoted A'. Thus, A' is the set of x such that $J^A(x)$ is defined, and $\emptyset'$ is (a version of) the halting problem. We say that a set A is Δ^0_2 if $A \leq_{\mathrm{T}} \emptyset'$. The following basic result of Shoenfield will be used frequently.

Lemma 2.1 (Limit Lemma).
A is $\Delta^0_2 \Leftrightarrow A(x) = \lim_s f(x,s)$ for some computable $0,1$-valued function f.

Usually we write $A_s(x)$ instead of $f(x,s)$.

Recall that $X \leq_{\mathrm{tt}} Y$ (X is truth-table below Y) if $X \leq_{\mathrm{T}} Y$ via a Turing functional Γ such that $\Gamma(Z)$ is total for all oracles Z. A variant of the Limit

Lemma says that $A \leq_{\mathrm{tt}} \emptyset'$ iff the number of changes in some computable approximation of A is computably bounded in x. Such a set is called *ω-c.e.*

Recall that a *lowness property* specifies a sense in which a set $A \subseteq \mathbb{N}$ is close to being computable. Such a property is closed downwards under $\leq_{\mathrm{T}}$. For instance, A is *low* if $A' \leq_{\mathrm{T}} \emptyset'$, that is, the Turing degree of A' is as low as possible. A is *superlow* if in fact A' is truth-table below $\emptyset'$.

Another example of a lowness property is the following. We say that a set A is *computably dominated* (or of hyperimmune-free degree) if each function f that can be computed with A as an oracle is dominated by a computable function. Outside the computable sets, this lowness property is not compatible with being low in the usual sense. In fact, the only computably dominated Δ^0_2 sets are the computable sets.

Cantor space. Finite sequences of bits will be called *strings*. The set of strings is denoted $\{0,1\}^*$. The variables x, y, z, σ, τ range over strings. We identify strings with natural numbers via a computable bijection $\{0,1\}^* \to \mathbb{N}$ (related to the binary presentation of a number).

Subsets of $\mathbb{N}$ are identified with infinite sequences of bits. They form the *Cantor space* $2^{\mathbb{N}}$, which is equipped with the product topology. For each string σ,

$$[\sigma] = \{X \colon \sigma \prec X\}$$

is the class of sets extending the string σ. The clopen classes $[\sigma]$ form a basis for the product topology. Thus, an *open class* (or set) has the form $\bigcup_{\sigma \in C} [\sigma]$ for some set C. Such a class is called *computably enumerable*, or Σ^0_1, if one can choose the set C computably enumerable.

The complements of computably enumerable open classes are called Π^0_1 *classes*. A Π^0_1 class is given as the set of paths through a computable binary tree. There are many examples of non-empty Π^0_1 classes without a computable member.

Basis Theorems for Π^0_1 classes. A *basis theorem* (for Π^0_1 classes) says that each non-empty Π^0_1 class has a member with a particular property, usually a lowness property.

Theorem 2.2 (Jockusch and Soare [16]). *Let $\mathcal{P}$ be a non-empty Π^0_1 class. Then $\mathcal{P}$ has a low member, and a computably dominated member.*

The proof of the first statement actually shows that $\mathcal{P}$ has a superlow member. In the second statement one obtains a computably dominated member A of $\mathcal{P}$ such that $A'' \leq_{\mathrm{tt}} \emptyset''$ (see [37, 1.8.38, 1.8.43]).

3. Randomness

In this section we consider the interaction from computability to randomness. We use algorithmic tools to introduce tests concepts, which determine formal

randomness notions. The tools are not only taken from computability theory on sets of natural numbers, but also from computable analysis, where the basic objects are continuous functions. We show that differentiability of certain computable functions defined on the unit interval can be used as a test notion.

3.1. Finite objects. Recall that $\{0,1\}^*$ denotes the set of strings over $\{0,1\}$. A *machine* is a partial computable function $M : \{0,1\}^* \mapsto \{0,1\}^*$. If $M(\sigma) = x$ we say that σ is an *M-description* of x.

We say that a machine M is *prefix-free* if no M-description is a proper initial segment of any other M-description. To build a prefix-free machine M, one usually specifies a set of *requests* $\langle r, y\rangle \in \mathbb{N} \times \{0,1\}^*$. Via such a request one asks that M can describe the string y with r bits. An important technical fact is that any consistent c.e. set of requests can be turned into a prefix-free machine. This result is often referred to as the Kraft-Chaitin theorem, but is called the Machine Existence Theorem in [37, 2.2.17].

Theorem 3.1. *Let L be a computably enumerable set of requests such that*

$$1 \geq \sum\{2^{-r} : \langle r, y\rangle \in L\}.$$

Then there is a prefix-free machine M such that for each request $\langle r, y\rangle \in L$, the machine M can describe y with at most r bits.

Let $(M_d)_{d\in\mathbb{N}}$ be an effective listing of the prefix-free machines. We define a prefix-free machine $\mathbb{U}$ by $\mathbb{U}(0^d 1\sigma) = M_d(\sigma)$. The machine $\mathbb{U}$ is *universal* in the sense that, if a string y has an M_d-description σ, it has a $\mathbb{U}$-description that is only longer by a constant. For a string x, we let $K(x)$ denote the length of a shortest $\mathbb{U}$-description of x:

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

The definitions given above can be generalized to the case that the computation model includes queries to an oracle set X. In this way, we define $\mathbb{U}^X(\sigma)$, $K^X(y)$, etc.

We list some facts about K. Let "$\leq^+$ denote "$\leq$" up to a constant. (For instance, we write $2n+5 \leq^+ n^2$.) Let $|x| \in \{0,1\}^*$ denote the length of a string x written in binary. The following bounds are proved by constructing appropriate prefix-free machines: for each computable function f we have $K(f(x)) \leq^+ K(x)$. In particular, we have the *lower bound* $K(|x|) \leq^+ K(x)$. Further, we have the *upper bound* $K(x) \leq^+ |x| + K(|x|)$. Since $K(|x|) \leq^+ 2\log|x|$, this upper bound is not much larger than $|x|$.

If $|x| \leq^+ K(x)$ we think of x as *incompressible*. This formalizes the intuitive notion of randomness for strings (see Section 2.5 of [37] for details).

3.2. Measure, tests, and Martin-Löf randomness. The *product measure* λ on Cantor space $2^{\mathbb{N}}$ is given by

$$\lambda[\sigma] = 2^{-|\sigma|}$$

for each string σ. If a class $\mathcal{G} \subseteq 2^{\mathbb{N}}$ is open then $\lambda\mathcal{G} = \sum_{\sigma\in B} 2^{-|\sigma|}$ where B is a prefix-free set of strings such that $\mathcal{G} = \bigcup_{\sigma\in B}[\sigma]$.

A class $\mathcal{C} \subseteq 2^{\mathbb{N}}$ is called *null* if $\mathcal{C}$ is contained in some Borel class $\mathcal{D}$ such that $\lambda\mathcal{D} = 0$. We discuss the connection of null classes and randomness. The intuition is that an object is random if it satisfies no exceptional properties. We give two examples of exceptional properties of a set Y. The first is that every other bit is zero. The second is that in the limit, there are at least twice as many zeros as ones:

$$2/3 \leq \liminf_n |\{i < n : Y(i) = 0\}|/n.$$

We would like to formalize "exceptional property" by "null class". The examples above are null classes, so they should not contain a random set. The problem is that if we do this, no set Z is random, because $\{Z\}$ itself is a null class. The solution is to consider only effective null classes. By specifying a particular notion of effectivity, we specify a notion of *tests*. To be random in this particular algorithmic sense, Z has to avoid these effective null classes, that is, to pass these tests. Since there are only countably many null classes of this type, the class of random sets in this sense will have measure 1.

Frequently test notions are based on the following fact from measure theory.

Fact 3.2. *The class $\mathcal{C} \subseteq 2^{\mathbb{N}}$ is null $\Leftrightarrow$ $\mathcal{C} \subseteq \bigcap \mathcal{G}_m$ for some sequence $(\mathcal{G}_m)_{m\in\mathbb{N}}$ of open sets such that $\lambda\mathcal{G}_m$ converges to 0.*

We obtain a type of effective null class (or test) by adding effectivity requirements to this condition characterizing null classes. We can require an effective presentation of $(\mathcal{G}_m)_{m\in\mathbb{N}}$; further, we can require fast convergence of $\lambda\mathcal{G}_m$ to 0. In this way, we obtain for instance the central randomness notion introduced by Martin-Löf in 1966 [26].

Definition 3.3. A *Martin-Löf test* (or ML-test) is a uniformly computably enumerable sequence $(\mathcal{G}_m)_{m\in\mathbb{N}}$ of open subclasses of $2^{\mathbb{N}}$ such that $\lambda\mathcal{G}_m \leq 2^{-m}$ for each m. A set Z is *Martin-Löf random* (or ML-random) if Z passes each ML-test $(\mathcal{G}_m)_{m\in\mathbb{N}}$, in the sense that Z is not a member of some $\mathcal{G}_m$.

The two properties given above (every other bit is zero, or in the limit there are at least twice as many zeros as ones) determine effective null classes in this sense. So a ML-random set does not have either of these properties.

In the following, we identify co-infinite sets with real numbers in $[0, 1)$ via the binary presentation. A natural example of a ML-random set was given by Chaitin. Consider the halting probability of the universal prefix-free machine $\mathbb{U}$:

$$\Omega = \sum\{2^{-|\sigma|} : \mathbb{U} \text{ halts on input } \sigma\}.$$

Note that this sum converges because the machine $\mathbb{U}$ is prefix-free. Chaitin proved that Ω is Martin-Löf random.

The left cut $\{q \in \mathbb{Q}\colon\ q < \Omega\}$ is computably enumerable. Since any real number is Turing equivalent to its left cut, this implies that $\Omega \leq_{\mathrm{T}} \emptyset'$. It is also not hard to show that $\emptyset' \leq_{\mathrm{T}} \Omega$. Thus Ω determines a ML-random set that is Turing equivalent to the halting problem.

Given a set Z and $n \in \mathbb{N}$, let $Z\upharpoonright_n$ denote the initial segment $Z(0)\dots Z(n{-}1)$. Schnorr's 1972 Theorem [40] says that Z is ML-random if and only if each of its initial segments is incompressible.

Theorem 3.4. *Z is ML-random $\Leftrightarrow$*
there is $b \in \mathbb{N}$ such that $\forall n\, K(Z\upharpoonright_n) > n - b$.

Levin [24] proved the analogous theorem for a variant of K called monotone string complexity.

Schnorr's Theorem yields a *universal* ML-test: Let

$$\mathcal{R}_b = \{X\colon\ \exists n\, [K(X\upharpoonright_n) \leq n - b]\}.$$

The relation "$K(x) \leq r$" is computably enumerable, so the sequence of open classes $\mathcal{R}_b$ is uniformly computably enumerable. One shows that $\lambda\mathcal{R}_b \leq 2^{-b}$. Thus, using this notation, Schnorr's Theorem says that

$$Z \text{ is ML-random } \Leftrightarrow \ Z \text{ passes the ML-test } (\mathcal{R}_b)_{b\in\mathbb{N}}.$$

So, the single test $(\mathcal{R}_b)_{b\in\mathbb{N}}$ suffices to emulate all the others.

This fact can be used to obtain ML-random sets with lowness properties. The complement of $\mathcal{R}_1$ is $\{X\colon\ \forall n\, K(X\upharpoonright_n) \geq n\}$. This is a Π^0_1 class of measure at least $1/2$. By Schnorr's Theorem, it consists entirely of ML-random sets. So we can apply the Jockusch-Soare Basis Theorems 2.2 to obtain ML-random sets satisfying lowness properties:

Example 3.5. *(i) There is a low ML-random set.*
(ii) There is a computably dominated ML-random set.

3.3. Randomness and differentiability.

A well-known theorem from analysis states that every function $f\colon\ [0,1] \to \mathbb{R}$ of bounded variation is differentiable almost everywhere (with respect to Lebesgue measure λ). In particular, this holds for every monotonic function. In the following we identify co-infinite subsets of $\mathbb{N}$ with reals in $[0,1]$ via the binary representation (we identify the set $\mathbb{N}$ with the real 1). If one also requires an effectiveness condition on the function, the reals at which it is not differentiable form a type of effective null class, and hence a test notion for reals. In the 1970s Demuth had a program to show that effective functions are well-behaved, and in particular differentiable, at random reals. For instance, in his own constructivist language he proved that if a real x is Martin-Löf random then each constructive function of bounded variation is differentiable at x [6].

We will describe a similar coincidence due to Brattka, Miller and Nies [2]. We characterize computable randomness using differentiability of non-decreasing computable functions on the unit interval. First we explain the notions involved.

Computable randomness. Martin-Löf tests are c.e. objects. For this reason Schnorr [39] maintained the point of view that ML-tests are already too powerful to be considered algorithmic. He proposed a more restricted notion of a test. His tests formalize computable betting strategies. A test in Schnorr's sense is a computable function M from $\{0,1\}^*$ to the non-negative rationals. When the player has seen $z = Z\upharpoonright_n$, she can make a bet q where $0 \leq q \leq M(z)$ on the next bit $Z(n)$. If she is right she wins q, otherwise she loses q. Thus M must satisfy the fairness condition $M(z0) + M(z1) = 2M(z)$ for each string z. She wins on Z if $M(Z\upharpoonright_n)$ is unbounded. We call a set Z *computably random* if no computable betting strategy wins.

Choose $c \in \mathbb{N}$ such that the start capital $M(\emptyset)$ is at most 2^c. Let $\mathcal{G}_r$ be the class of Z such that $M(Z\upharpoonright_k) \geq 2^{r+c}$ for some k. It is not hard to see that $(\mathcal{G}_r)_{r\in\mathbb{N}}$ forms a ML-test. If $M(Z\upharpoonright_n)$ is unbounded then $Z \in \bigcap_m \mathcal{G}_m$. This shows that computable betting strategies induce a type of effective null class. Further, each ML-random set is computably random.

Computable functions on the unit interval. A Cauchy representation of a real $x \in \mathbb{R}$ is a sequence $(q_i)_{i\in\mathbb{N}}$ of rationals converging to x such that $|q_k - q_i| \leq 2^{-i}$ for each $k \geq i$. A function $f\colon [0,1] \to \mathbb{R}$ is called *computable* if there is an effective method (i.e., a Turing functional) to transform each Cauchy representation of an $x \in [0,1]$ into a Cauchy representation of $f(x)$. Such a function is necessarily continuous. Functions from analysis such as e^x, $\sqrt{x}$ etc. are computable.

We are now able to state the result of Brattka, Miller and Nies in [2].

Theorem 3.6. *Let $x \in [0,1]$. Then x is computably random if and only if $f'(x)$ exists for each computable non-decreasing function f.*

Further research in [2] indicates that x is Martin-Löf random if and only if each computable function of bounded variation is differentiable at x. The forward implication is a variant of the aforementioned result of Demuth [6].

3.4. A notion stronger than Martin-Löf randomness. One can also argue that ML-randomness is too weak to be viewed as a formal counterpart of our intuitive idea of randomness for sets. For instance, the ML-random real Ω has a computably enumerable left cut, and is Turing equivalent to the halting problem $\emptyset'$. These properties may contradict our intuition on randomness: the halting problem is not random at all, so a random set should not match its computational strength. In fact, the set should be Turing incomparable with the halting problem. The following stronger notion was proposed by Kurtz [23].

Definition 3.7. We say that Z is *2-random* if Z is ML-random relative to the halting problem.

Clearly, a set $Z \leq_{\mathrm{T}} \emptyset'$ is not 2-random: let $\mathcal{G}_m = [Z\upharpoonright_m]$, then $(\mathcal{G}_m)_{m\in\mathbb{N}}$ is a ML-test relative to $\emptyset'$ which Z fails. It is also not hard to show that a set $Z \geq_T \emptyset'$ is not 2-random: given a Turing functional Φ, the halting problem can for each m compute k such that the measure of $\mathcal{G}_m = \{Y\colon \forall i < k\,[\Phi^Y(i) = \emptyset'(i)]\}$ is at most 2^{-m}. If $\Phi(Z) = \emptyset'$ then Z fails $(\mathcal{G}_m)_{m\in\mathbb{N}}$, which is a ML-test relative to $\emptyset'$.

For an example, note that the real $\Omega^{\emptyset'}$ is 2-random. Kurtz [23] showed, among other things, that no 2-random set Z is computably dominated (see Section 2 for the definition). In fact he obtained the stronger result that Z is c.e. relative to some set $Y <_T Z$.

Let $C(x)$ be the plain Kolmogorov complexity of a string x, without restriction to prefix-free machines. Clearly $C(x) \leq^+ |x|$. A string is incompressible in the sense of C if $C(x) > |x| - b$ for some (small) constant b. One can show that for some constant slightly larger than b, all prefixes of such a string are incompressible in the sense of K.

Our next coincidence result characterizes 2-randomness in terms of C-incompressibility of initial segments. This can be seen as a variant of Schnorr's Theorem 3.4. However, we merely need C-incompressibility of *infinitely* many initial segments to arrive at the stronger notion of 2-randomness. This suggests that C-incompressibility of a string is a condition much stronger than K-incompressibility.

The coincidence result is due to Nies, Stephan and Terwijn [36]; the harder implication "⇒" was also independently (and slightly earlier) obtained by Miller [28].

Theorem 3.8. *Z is 2-random ⇔*
there is $b \in \mathbb{N}$ such that $C(Z\upharpoonright_n) > n - b$ for infinitely many n.

Sketch of Proof (for the details see [37, Thm. 3.6.10]).
⇐: Recall that for each oracle X the domain of $\mathbb{U}^X$ is prefix-free. The plain machine M on input σ searches for a splitting $\sigma = \tau z$ such that $y = \mathbb{U}^{\emptyset'}(\tau)$ converges in $|\sigma|$ steps with the approximation of the oracle $\emptyset'$ at stage $|\sigma|$. In this case, it outputs yz. That is, M prints y followed by the rest of σ.

Now suppose that Z is not 2-random. Then by Theorem 3.4 relative to $\emptyset'$, for each d there is $r \in \mathbb{N}$ such that $K^{\emptyset'}(Z\upharpoonright_r) \leq r - d$. Let n_0 be so large that the final computation $\mathbb{U}^{\emptyset'}(\tau) = Z\upharpoonright_r$ converges in n_0 steps for some τ such that $|\tau| \leq r - d$. For each $n \geq n_0$, if the string y contains the bits of Z from position r to $n-1$, then $M(\tau y) = Z\upharpoonright_n$. Thus M can describe $Z\upharpoonright_n$ with at most $n - d$ bits for each $n \geq n_0$, whence $C(Z\upharpoonright_n) \leq n - d + O(1)$ for each $n \geq n_0$.

⇒: A function $F\colon \{0,1\}^* \to \{0,1\}^*$ is called a *compression function* if F is one-one and $|F(x)| \leq C(x)$ for each x. By the Low Basis Theorem 2.2 there is

a compression function F such that $F' \equiv_T \emptyset'$. Using this lowness of F, if Z is 2-random one can show that there is b such that $|F(Z\upharpoonright_n)| > n - b$ for infinitely many n. This implies that $C(Z\upharpoonright_n) > n - b$ for infinitely many n. □

As a corollary, in [36] we obtained a simple new proof of Kurtz's result [23] that no 2-random set is computably dominated.

Summary of Section 3. For binary strings x, we introduce plain descriptive string complexity $C(x)$, and prefix-free string complexity $K(x)$. The intuitive notion of randomness for strings can be formalized by incompressibility. One formal version of incompressibility is $|x| \leq^+ K(x)$. A stronger one is $|x| \leq^+ C(x)$.

For an infinite sequence of bits (i.e., a set), the intuitive notion of randomness corresponds to a hierarchy of mathematical randomness notions. The central one is Martin-Löf randomness; computable randomness is a weaker notion where the tests formalize the idea of a computable betting strategy; 2-randomness is the relativization of ML-randomness to $\emptyset'$.

ML-randomness and 2-randomness can be characterized via incompressibility of initial segments of the sequence. Computable randomness is implied by ML-randomness. It can be characterized by the condition that each non-decreasing computable function on the unit interval is differentiable at the corresponding real number.

Some further facts. The implications between randomness notions are proper:
2-random $\Rightarrow$ Martin-Löf random $\Rightarrow$ computably random. See [37, 3.6.2, 7.4.8].

Suppose that the set A is computable. If Z satisfies a randomness notion, then the symmetric difference $Z \Delta A$ satisfies the same notion. Further, if ρ is a computable permutation of $\mathbb{N}$, then $\rho(Z)$ satisfies the same notion (see [37, 7.6.24] for the case of computable randomness).

4. For Δ^0_2, Close to Computable = Far From Random

We will introduce a lowness property via relativized ML-randomness. Further, we will introduce the K-trivial sets which are far from random. Recall from Subsection 3.1 that $K(x)$ is the length of a shortest prefix-free description of a string x.

Definition 4.1. *Let $A \subseteq \mathbb{N}$.*

(i) A is *low for ML-randomness* (Zambella 1990, [44]) if each ML-random set is already ML-random relative to A.

(ii) A is *K-trivial* (Chaitin 1975, [4]) if each initial segment of A has prefix-free complexity no greater than the complexity of its length. That is, there is $b \in \mathbb{N}$ such that, for each n, $K(A\upharpoonright_n) \leq K(n) + b$. (Here n is written in binary.)

We will see that these two properties of sets are equivalent.

4.1. Background on the two properties.

Lowness for ML-randomness. Zambella asked whether lowness for ML-randomness implies being computable. Kučera and Terwijn [22] answered this in the negative. They proved that in fact some incomputable c.e. set is low for ML-randomness.

Kjos-Hanssen [17] characterized being low for ML-randomness using only effective topology and the uniform measure on Cantor space: A is low for ML-randomness $\Leftrightarrow$ each open class $G \subseteq 2^{\mathbb{N}}$ that is c.e. in A and has measure $\lambda\mathcal{G}$ less than 1 is contained in an open class $\mathcal{S}$ that is c.e. (without the oracle A) and still of measure $\lambda\mathcal{S}$ less than 1.

J. Miller observed that for an incomputable set A, Kjos-Hanssen's result is not constructive. An *index* for a c.e. open set $\mathcal{R}$ is a number e such that $\mathcal{R}$ is the class of sets extending some string in W_e. Assume that from an index relative to A for $\mathcal{G}$ we can effectively obtain an index for the covering class $\mathcal{S} \supseteq \mathcal{G}$. To compute $A\upharpoonright_n$, let $\mathcal{G} = 2^{\mathbb{N}} - [A\upharpoonright_n]$. Compute the index for $\mathcal{S}$. Wait for a stage when all strings y of length n except for one satisfy $[y] \subseteq \mathcal{S}$. Then $A\upharpoonright_n$ must be the remaining string.

K-triviality. This property of sets is the opposite of ML-randomness: K-trivial sets are "antirandom". For, by Schnorr's Theorem 3.4, Z is ML-random iff all values $K(Z\upharpoonright_n)$ are near their upper bound $n + K(n)$; on the other hand Z is K-trivial if the values $K(Z\upharpoonright_n)$ are at their lower bound $K(n)$ (all within constants).

Chaitin [4] was the first to study K-triviality. He showed that the number of strings of a fixed length with minimal K-complexity up to a constant b is bounded by $O(2^b)$.

Theorem 4.2 (Counting Theorem [4]). *For each $b \in \mathbb{N}$, at most $O(2^b)$ strings of length n satisfy $K(x) \leq K(n) + b$. Thus, at most $O(2^b)$ sets are K-trivial with constant b.*

The following is an easy consequence.

Theorem 4.3 ([4]). *Each K-trivial set is Δ^0_2.*

Proof. A is K-trivial via the constant b iff A is a path on the Δ^0_2 tree of strings z such that $K(x) \leq K(|x|) + b$ for each $x \preceq z$. This tree has only $O(2^b)$ paths. Therefore A is Δ^0_2 as an isolated path on a Δ^0_2 tree. □

Instigated by Chaitin, in 1975 Solovay [42] built an incomputable K-trivial set. His set was merely Δ^0_2. Calude and Coles [3] modified Solovay's construction in order to make the set c.e. In 2002, Downey, Hirschfeldt, Nies and Stephan [9] gave an easier construction of a c.e. incomputable K-trivial set. It is similar to the 1999 Kučera-Terwijn construction of a set that is low for ML-randomness.

These constructions gave rise to the cost function method described in Section 5. In Proposition 5.4 we will explain how to obtain a c.e. incomputable K-trivial set via the general Existence Theorem 5.3.

For sets A and B, let $A \oplus B$ denote the set $2A \cup 2B + 1$, namely the set which is A on the even bit positions and B on the odd positions. The K-trivial sets are closed under $\oplus$ by the following result of Downey, Hirschfeldt, Nies and Stephan.

Theorem 4.4 ([9]). *If A and B are K-trivial via b, then $A \oplus B$ is K-trivial via $3b + O(1)$.*

Proof. It is sufficient to describe each string $A \oplus B \upharpoonright_{2n}$ with $K(n)+3b+O(1)$ bits. To do this, we need to describe n only once; if we have a shortest description of n we also know its length $r = K(n)$.

We (somewhat generously) use $b+1$ bits to describe b itself, by putting the string 0^b1 at the beginning of our description of $A \oplus B \upharpoonright_{2n}$. Next, we put the prefix-free description of n. The set of strings x of length n such that $K(x) \leq r + b$ is uniformly c.e. and has size $O(2^b)$ by Chaitin's Counting Theorem 4.2. So we only need to put $b + O(1)$ further bits each to describe the positions of $A \upharpoonright_n$ and $B \upharpoonright_n$ in its enumeration. □

4.2. Coincidence of the two properties.

Theorem 4.5. *A is low for ML-randomness $\Leftrightarrow$ A is K-trivial.*

Known since 2002, this result published in [34] is now considered fundamental in the area. Nies [34] proved "⇒". The converse implication has a complicated history. Downey, Hirschfeldt, Nies and Stephan [9] showed that each K-trivial set is Turing incomplete. These ideas were later explained through the decanter model [10]. Nies combined this model with a new technique called the golden run method in order to show that the K-trivial sets are closed downward under $\leq_T$. Hirschfeldt and Nies together used the golden run method to show the stronger result that K-triviality implies being low for ML-randomess; see [34, 37].

Conceptually, lowness for ML-randomness and K-triviality are quite far apart: the former is a lowness property defined in terms of randomness, while the latter expresses being far from random. So it come at no surprise that the proof of their coincidence is hard. On the other hand, this makes the coincidence quite beneficial, because properties that are easily obtained via one definition can be very hard to obtain directly via the other. For instance, it is easy to see from the definition that each set A that is low for ML-randomness is generalized low_1, i.e., $A' \leq_{\mathrm{T}} A \oplus \emptyset'$ [22], while it takes the golden run method to see this for the K-trivials. On the other hand, for the K-trivial sets, containment in the Δ^0_2 sets and closure under $\oplus$ is not very hard to see (Theorems 4.3, 4.4). If one takes the definition via lowness for ML-randomness, containment in the

Δ^0_2 sets is much harder [33], and no direct proof is even known for the closure under $\oplus$.

Outline of the Proof. It is easiest to introduce two further properties and show the coincidence of the theorem via these properties. The implication from left to right is proved via the notion of a base for ML-randomness. The converse implication is proved via the notion of being low for K. These two notions are of independent interest.

$\Rightarrow$: Bases for ML-randomness were introduced by Kučera [21] in a different terminology.

Definition 4.6. *We say that A is a base for ML-randomness if $A \leq_{\mathrm{T}} Z$ for some set Z that is ML-random relative to A.*

Each set A that is low for ML-randomness is a base for ML-randomness. For, by the Kučera-Gács Theorem (see [37, Thm. 3.3.2]) there is a ML-random set Z such that $A \leq_T Z$. Then Z is ML-random relative to A.

It is now sufficient to show that each base for ML-randomness is K-trivial. This is a result of Hirschfeldt, Nies and Stephan [15] whose proof we follow. Suppose there are a set Z and a Turing functional Φ such that $\Phi^Z = A$ and Z is ML-random relative to A. We will build a prefix-free machine N_d for each $d \in \mathbb{N}$. We want to ensure that there is a d such that N_d can describe each $\tau \prec A$ with $K(|\tau|) + d + 2$ bits. Of course, A is unknown. Thus, given the limitation that the total measure of the N_d-descriptions must not exceed 1, we have to be judicious in deciding which strings τ receive such a description. The idea is to build uniformly c.e. open classes $\mathcal{C}^\tau_d \subseteq 2^\omega$ for $d \in \mathbb{N}$ and $\tau \in 2^{<\omega}$. Their purpose is to test whether a string τ is likely to be an initial segment of A. Roughly, τ fulfills this test if sufficiently many σ satisfy $\tau \preceq \Phi^\sigma$.

For each fixed d, the $\mathcal{C}^\tau_d$ are pairwise disjoint. If we let $\mathcal{G}_d = \bigcup_{\tau \prec A} \mathcal{C}^\tau_d$, then the following hold.

- $(\mathcal{G}_d)_{d\in\mathbb{N}}$ is a Martin-Löf test relative to A.
- If $Z \notin \mathcal{G}_d$ then $\lambda\mathcal{C}^\tau_d = 2^{-K(|\tau|)-d}$ for all $\tau \prec A$.

For a c.e. open class $\mathcal{C}$ and a stage s, let $\mathcal{C}[s] \subseteq \mathcal{C}$ denote the clopen class approximating $\mathcal{C}$ at stage s. We define N_d by enumerating a description of length $K_s(|\tau|)+d+2$ of τ at stage s whenever we have not previously enumerated such a description and $\lambda\mathcal{C}^\tau_d[s] \geq 2^{-K_s(|\tau|)-d-1}$. Since the $\mathcal{C}^\tau_d$ are disjoint for fixed d, we don't run out of descriptions. For the formal definition of the N_d we apply the Machine Existence Theorem 3.1.

Since Z is ML-random relative to A, we have $Z \notin \mathcal{G}_d$ for some d and hence $\lambda\mathcal{C}^\tau_d = 2^{-K(|\tau|)-d}$ for all $\tau \prec A$. This implies that there is an N_d-description of length $K(|\tau|) + d + 2$ of τ for all $\tau \prec A$, as desired.

To build the $\mathcal{C}^\tau_d$, as long as at a stage s we have $\lambda\mathcal{C}^\tau_d[s] < 2^{-K_s(|\tau|)-d}$, we look for strings σ such that $\tau \preceq \Phi^\sigma$ and $\lambda\mathcal{C}^\tau_d[s] + 2^{-|\sigma|} \leq 2^{-K_s(|\tau|)-d}$, and put $[\sigma]$ into $\mathcal{C}^\tau_d[s+1]$. To keep our open classes pairwise disjoint, we then ensure that no $[\sigma']$ such that σ' is compatible with σ is later put into $\mathcal{C}^\nu_d$ for any string ν.

If $Z \notin \mathcal{G}_d$, then no $[\sigma]$ with $\sigma \prec Z$ is ever put into any $\mathcal{C}^\tau_d$. This means that the measure of each $\mathcal{C}^\tau_d$ with $\tau \prec A = \Phi^Z$ must eventually exceed $2^{-K(|\tau|)-d-1}$.

$\Leftarrow$: Recall that $\mathbb{U}^A$ is the universal prefix-free machine with oracle A, and $K^A(y)$ is the length of a shortest $\mathbb{U}^A$-description of y. In general, enhancing the computational power of the universal machine by an oracle A decreases $K(y)$. We say that A is *low for* K if this is not so:

Definition 4.7. *A is* low for K *if $K(y) \leq^+ K^A(y)$ for each string y.*

This property was introduced by Andrej Muchnik Jr. in a 1999 Moscow seminar. He showed that some incomputable c.e. set is low for K. Among the properties discussed in this section, it is the most well-behaved. For instance, if a c.e. set A is low for K, we can, effectively in the constant for being low for K and the c.e. index for A, find an index for a truth table reduction showing $A' \leq_{\mathrm{tt}} \emptyset'$, that is, the superlowness of A [37, 5.1.3].

Also, lowness for K easily implies the other properties we have discussed. By Schnorr's Theorem relative to A, being low for K implies being low for ML-randomness: if Z is ML-random, then $n \leq^+ K(Z\upharpoonright_n) \leq^+ K^A(Z\upharpoonright_n)$ for each n, so Z is ML-random relative to A. To show that lowness for K implies K-triviality, one uses the finitary methods common in algorithmic information theory (see [25]): $K(A\upharpoonright_n) \leq^+ K^A(A\upharpoonright_n) \leq^+ K^A(n) \leq^+ K(n)$. The hypothesis is only used in the first inequality.

To prove that K-triviality implies being low for ML-randomness, it now suffices to show that conversely, each K-trivial set is low for K. From the formulation of this implication, one could hope that it can also be proved using finitary methods, such as manipulating inequalities involving K and K^A. However, so far no one has found such a proof.

The difficulty of proving the implication "K-trivial $\Rightarrow$ low for K" is in part explained by the fact that it is not constructive: from a constant for K-triviality and a c.e. index for a set A, one can not compute a constant via which A is low for K. See [37, 5.5.6], which goes back to [9]. In fact, one cannot even compute an index of a Turing reduction for $A' \leq_{\mathrm{T}} \emptyset'$ [37, 5.5.5]. This shows that the original implication "$\Leftarrow$" in the theorem is also not constructive.

We give an outline of the proof that K-triviality implies being low for K, using the decanter model and the golden run method; for more details see [37, Sections 5.4-5]. We already know from Theorem 4.3 that each K-trival set A is Δ^0_2, and hence has a computable approximation $(A_s)_{s\in\mathbb{N}}$ in the sense of the Limit Lemma 2.1. We now have to understand why K-triviality of A can be seen as an inertness (in particular, a lowness) property. Roughly speaking, whenever $A\upharpoonright_n$ changes, say at a stage s, a $\mathbb{U}$-description of length at most $K_s(n) + b$ of

the new version of $A\upharpoonright_n$ is needed. The measure of possible descriptions is at most 1, so this restricts the changes of A.

Turing incompleteness. First we will discuss the result of [9] that a K-trivial set A is Turing incomplete. The proof is by contradiction. We build an auxiliary c.e. set B. If A is Turing complete, then by the Recursion Theorem we are given a Turing reduction Γ such that $B = \Gamma(A)$. Let $\gamma^A(m)[s]$ denote the use u, i.e., $u-1$ is the largest oracle question asked in the computation $\Gamma^A(m)[s]$. If we put m into B then A must change below u in order to maintain $B = \Gamma(A)$ at input m.

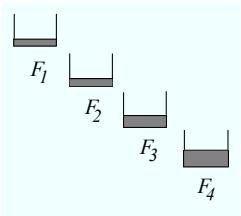
We also build a bounded request set L as in Theorem 3.1. Putting a request $\langle r, n\rangle$ into L causes $K(n) \leq r + d$, where d is the constant for the machine obtained from L (which is again known ahead of the construction by the Recursion Theorem). Hence $K(A\upharpoonright_n) \leq r + b + d$ where A is K-trivial via b.

Let $\boldsymbol{k} = 2^{b+d+1}$. If we can force $A\upharpoonright_n$ to change to a new configuration for more than $\boldsymbol{k}/2$ times, then for this n, our investment into L is overmatched by the opponent's investment into descriptions of $A\upharpoonright_n$. The idea is to do this for so many numbers n that he does not have enough resources to match us.

We can cause these changes if we have $\boldsymbol{k}/2$ numbers m with use $\gamma^A(m) \leq n$ to put into B. If Γ is a weak truth-table reduction, i.e., the use is bounded by a computable function g, we can arrange this by choosing $n \geq g(\boldsymbol{k})$. In the general case, the opponent will simply change A "early" and then redefine the use $\gamma^A(m)$ with a value beyond n. This deprives us of the possibility to cause further A-changes when we need them.

Our solution to this problem is to pool numbers n together, so that a single A-change will let us make progress on lots of them. Further, we already make partial progress based on the A-changes the opponent relies on to move up $\gamma^A(m)$. For $i \leq \boldsymbol{k}$ let us say that a set E is an *i-set* if for each $n \in E$, we put a request $\langle r_n, n\rangle$ into L, and then see descriptions of length $r_n + b + d$ of i different $A\upharpoonright_n$ configurations. The *weight* of such a set is $\sum_{n\in E} 2^{-r_n}$. If n is in a $\boldsymbol{k}$-set, then for each of the $\boldsymbol{k}$ different versions $A\upharpoonright_n$ there is a $\mathbb{U}$-description of length at most $r_n + b + d$. Hence the weight of a $\boldsymbol{k}$-set cannot exceed $1/2$.

We visualize a set of numbers n associated with requests $\langle r_n, n\rangle$ as a quantity of precious wine of the corresponding weight. The *decanter model* consists of decanters $F_1, \ldots, F_{\boldsymbol{k}}$. For instance, in the case $\boldsymbol{k} = 4$ it looks like this:



Precious wine is first poured into F_1. Decanter F_{i-1} can be emptied into decanter F_i. At any stage the content of each F_i must form an i-set. We want as much wine as possible to reach $F_{\boldsymbol{k}}$, because from $F_{\boldsymbol{k}}$ we can pour it into a glass

and drink it. Under certain circumstances we cannot ensure that the content of F_{i-1} is promoted to F_i, so we have to spill it on the floor.

When we put $\langle r_n, n\rangle$ into L, we also put n into F_1, which means that we pour a quantity 2^{-r_n} of precious wine into F_1. At a stage s, all elements n of F_{i-1}, $i \leq \boldsymbol{k}$, satisfy $n \geq \gamma^A(m)[s]$ for a specific number m associated with F_i (to be explained shortly). Once the weight of F_{i-1} passes a certain quota, we put m into B and empty F_{i-1} into F_i. Since $A\upharpoonright_{\gamma^A(m)}$ has to change to keep $B = \Gamma(A)$ correct, the content of F_i including the wine just added remains an i-set, as required.

Now we can get around the problem of an "early" A-change that would move $\gamma^A(m)$ beyond n. If A changes early, then the wine that has already reached F_{i-1} is still promoted to F_i. The only wine lost is the one currently in $F_1, \ldots, F_{i-2}$: the content of these decanters is spilled onto the floor. But the quotas of these decanters are chosen smaller and smaller as i decreases, so we can ensure that the total quantity of wine spilled has a weight of less than $1/4$.

In the construction we have many *runs* of procedures associated with a decanter F_i. Each run has a parameter m such that $\Gamma^A(m)$ converges, and a weight quota p called its *goal*. For $i > 1$ it will call a run associated with F_{i-1} with smaller quota for as many times as needed for F_{i-1} to fill to weight p. Then it puts m into B, empties F_{i-1} into F_i, and returns. If $A\upharpoonright_{\gamma^A(m)}$ changes prematurely then the current content of F_{i-1} is poured into F_i, but the run for F_i continues.

The construction starts out by running $F_{\boldsymbol{k}}$ with a quota of $3/4$. It calls $F_{\boldsymbol{k}-1}$ with a smaller quota for a number of times, and so on down to F_1.

Since Γ^A is total we can force all the A changes needed for runs to return. Hence, the single run associated with $F_{\boldsymbol{k}}$ returns. This yields a $\boldsymbol{k}$-set of weight $3/4$, which is a contradiction.

The full result. We now discuss the full result that a K-trivial set A is low for K. The basic approach is to build a bounded request set W (see Theorem 3.1) as follows: if $\mathbb{U}^A(\sigma) = y$, there is a request $\langle |\sigma| + O(1), y\rangle$ in W. Similar to the proof of the implication "$\Rightarrow$" of the present theorem, we have to judiciously choose the computations $\mathbb{U}^A(\sigma)$ existing at a stage s for which we want to issue a request. The set W has bounded resources, so we have to limit the situation that, after a computation is chosen, A changes to destroy it. We will use such an A-change to promote numbers.

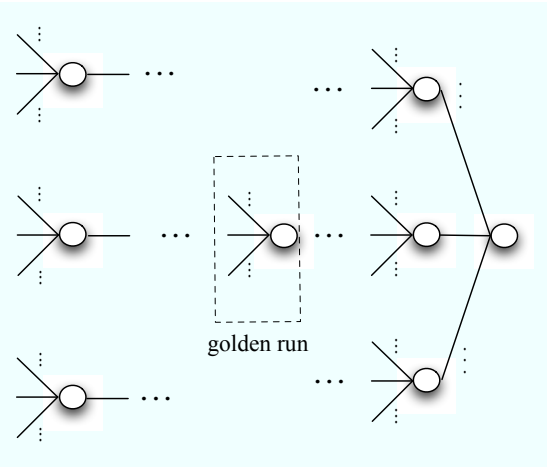
To exploit the hypothesis that A is K-trivial, as before we build a global bounded request set L. Numbers n go through levels $1, \ldots, \boldsymbol{k}$. The decanters are now arranged on a tree. While trying to fill, each decanter at a level greater than 1 builds its own bounded request set W in an attempt to show that A is low for K.

Suppose F_i is a decanter at level i where $1 < i \leq \boldsymbol{k}$. When $\mathbb{U}^A(\sigma)$ converges, F_i calls a decanter $F_{i-1,\sigma}$ at level $i-1$ that can be emptied into F_i. Its goal is

$2^{-|\sigma|}\alpha$, where α is a non-negative rational called the *garbage quota* of the run of F_i (to be explained shortly). When $F_{i-1,\sigma}$ reaches its goal, it returns. It now remains inactive, until possibly A changes below the use of $\mathbb{U}^A(\sigma)$. In this case the content of $F_{i-1,\sigma}$ becomes an i-set, so $F_{i-1,\sigma}$ can be emptied into F_i. We say that the run of $F_{i-1,\sigma}$ is *released*.

If A changes below the use of $\mathbb{U}^A(\sigma)$ before the run returns, then this run is *cancelled*, but we still can empty the current content of $F_{i-1,\sigma}$ into F_i, because the A-change turned it into an i-set.

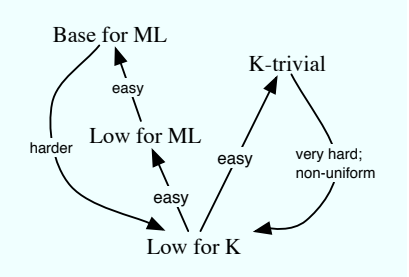
If A does not change at all, a quantity $2^{-|\sigma|}\alpha$ of garbage has been created in the form of wine that is forever stuck at the now defunct decanter $F_{i-1,\sigma}$. If we choose the α values small enough, we can make the amount of garbage tolerable.



To start the construction, we call the decanter at level $\boldsymbol{k}$ with goal 3/4 and an appropriate small garbage quota. A *golden run* is a run of a decanter F_i that is never cancelled and never returns, while all the runs of $F_{i-1,\sigma}$ it calls are cancelled or return. A golden run exists, for otherwise the decanter at level $\boldsymbol{k}$ would reach its goal 3/4, which is a contradiction.

At the golden run node F_i we can build a bounded request set W that succeeds in showing that A is low for K. Suppose the golden run of F_i has goal p and garbage quota α. Let $u \in \mathbb{N}$ be least such that $p/\alpha \leq 2^u$. When $F_{i-1,\sigma}$ returns we put $\langle |\sigma| + u + 1, y\rangle$ into W. To see that W is a bounded request set, note that we can bound by 2^u the sum of all $2^{|\sigma|}$ where σ is a $\mathbb{U}^A$-description at some stage, $F_{i-1,\sigma}$ is called, then $F_{i-1,\sigma}$ returns, and later on it is released by an A-change. If this sum exceeds 2^u then the run of F_i reaches its goal $p \leq 2^u\alpha$. So these descriptions contribute at most 1/2 to W. The weight of the descriptions σ where A does not change after the run of $F_{i-1,\sigma}$ returns is at most the measure of the domain of $\mathbb{U}^A$, whence their contribution is at most 1/2. Hence W is a bounded request set. □

Summary of Section 4. We introduce a lowness property, being low for ML-randomness, and a far-from-randomness property, K-triviality. Each K-trivial set is Δ^0_2. The K-trivials are closed under $\oplus$.



We show the equivalence of the two properties. To do so we introduce two further properties, being a base for ML-randomness and being low for K, which are also of independent interest. The diagram summarizes the implications discussed. To show that each K-trivial is low for K, we need the decanter and golden run methods.
Some further facts. We say that A is C-trivial if $C(A\upharpoonright_n) \leq^+ C(n)$. Each C-trivial set is computable (Chaitin; see [37, 5.2.20]).

Directly from the definition one can see that each set A that is low for ML-randomness is GL_1, namely $A' \leq_{\mathrm{T}} A \oplus \emptyset'$ [22]. As mentioned above, it is not hard to see from the definition that each c.e. set A that is low for K is superlow, namely $A' \leq_{\mathrm{tt}} \emptyset'$ [37, 5.1.3]. A golden run construction shows directly that each K-trivial set is superlow [37, p. 208].

5. The Inertness Paradigm and Cost Functions

In Section 4.1 we described four properties of Δ^0_2 sets that were introduced by different groups of researchers. For each property, the researchers gave a construction of a c.e. incomputable set with the property. All these constructions looked similar, which is not too surprising given that the properties later turned out to be equivalent. From 1999 on, the language of cost function was developed to formulate these constructions [22, 9, 34].

Nowadays cost functions are an indispensable tool for understanding the class of K-trivial sets and its subclasses [37, Section 5.3], [14, 35]. For instance, each K-trivial set is Turing below a c.e. K-trivial set (see Corollary 5.6 below). The only known proof of this result relies on a cost function.

5.1. Basics on cost functions. Recall the Limit Lemma 2.1: $A \leq_{\mathrm{T}} \emptyset'$ iff $A(x) = \lim_s A_s(x)$ for some 0,1-valued computable approximation $(A_s)_{s\in\mathbb{N}}$. Cost functions are used to measure the total of changes, taken over all numbers, of a computable approximation. In this way we have a formal version of the inertness paradigm from the introduction: a Δ^0_2 set is close to computable if it can be computably approximated with a small total amount of changes.

Definition 5.1. A *cost function* is a computable function

$$c : \mathbb{N} \times \mathbb{N} \to \{x \in \mathbb{Q} \colon\ x \geq 0\}.$$

We say that a cost function c satisfies the *limit condition* if

$$\lim_x \sup_s c(x, s) = 0.$$

When building a computable approximation of a Δ^0_2 set A, we view $c(x, s)$ as the cost of changing $A(x)$ at stage s. We now express that the *total* cost of changes, taken over all x, is finite [37, Section 5.3].

Definition 5.2. We say that a computable approximation $(A_s)_{s\in\mathbb{N}}$ *obeys* a cost function c if

$$\infty > \sum_{x,s} c(x, s)\, [\![x < s \ \wedge\ x \text{ is least such that } A_{s-1}(x) \neq A_s(x)]\!].$$

We say that A obeys c if *some* computable approximation of A obeys c.

Mostly we use this to construct some auxiliary object of finite "weight", such as a bounded request set in the sense of 3.1, or a so-called Solovay test in the proof of Theorem 5.10 below.

The analytic approach to restricting changes is more powerful than most combinatorial approaches. For example, call a Δ^0_2 set A *slow* if for each non-decreasing unbounded computable function h, there is a computable approximation $(A_s)_{s\in\mathbb{N}}$ of A such that $A_s \upharpoonright_n$ changes at most $h(n)$ times. Is it not hard to build a slow c.e. set that is Turing complete.

A co-infinite c.e. set is called *simple* [38] if it meets each infinite c.e. set. Clearly no such set is computable. The following theorem can be traced back to [22, 9].

Theorem 5.3. *If a cost function c satisfies the limit condition, then some simple set A obeys c.*

Proof. Let $(W_e)_{e\in\mathbb{N}}$ be an effective listing of the c.e. sets. To make A simple we meet the requirements $S_e :\ |W_e| = \infty \Rightarrow A \cap W_e \neq \emptyset$. Requirement S_e is allowed to spend at most 2^{-e}. Because of the limit condition, S_e can wait for an x to appear in W_e that is so large that S_e can afford it.

At stage s, if S_e is not satisfied yet, we look for an x, $2e \leq x < s$, such that $x \in W_{e,s}$ and

$$c(x, s) \leq 2^{-e}.$$

If so, we put the least such x into A and declare S_e satisfied.

Since a requirement S_e spends at most 2^{-e}, the total cost of changes is bounded by $\sum_e 2^{-e} = 2$. Hence A obeys c.

Suppose that W_e is infinite. As explained above, since c satisfies the limit condition, each S_e is met. A is co-infinite because we choose $x \geq 2e$. So A is simple. □

We say that a cost function $c(x, s)$ is *monotonic* if $c(x, s)$ is non-increasing in x, and non-decreasing in s. Thus, at the same stage a smaller number can only be more expensive, and the same number can only get more expensive at later stages. Most cost functions given below will be monotonic.

5.2. Applications of cost functions.

We analyze some lowness properties and their corresponding constructions, using cost functions.

5.2.1. K-triviality. Recall that a set A is K-trivial if there is a $b \in \mathbb{N}$ such that $\forall n\ K(A\upharpoonright_n) \leq K(n) + b$. We introduce a cost function $c_{\mathcal{K}}$ satisfying the limit condition such that any set obeying $c_{\mathcal{K}}$ is K-trivial. Then, by Theorem 5.3, there is a simple K-trivial set. In Theorem 5.5 we will prove that obeying $c_{\mathcal{K}}$ actually characterizes K-triviality.

To show that A is K-trivial we build an appropriate prefix-free machine M via the Machine Existence Theorem 3.1.

(a) Let $K_s(i)$ be the value of $K(i)$ at stage s. Whenever there is a new value $K_s(i)$, we give an M-description of $A_s\upharpoonright_i$ with length $K_s(i) + 1$. The combined weight of such descriptions is at most $1/2$.

(b) If $A(x)$ changes at stage s then, for all i such that $s \geq i > x$, the initial segment $A\upharpoonright_i$ gets a new M-description of length $K_s(i) + 1$. If we let

$$c_{\mathcal{K}}(x, s) = \textstyle\sum_{i=x+1}^{s} 2^{-K_s(i)},$$

then the measure of the new M-descriptions needed is $c_{\mathcal{K}}(x, s)/2$. If A obeys $c_{\mathcal{K}}$ and the total cost of changes is at most 1, this contributes a weight of at most $1/2$ in M-descriptions, so we build the desired machine. More generally, if the total cost of changes is at most 2^d for $d \in \mathbb{N}$, we choose the M-descriptions in (b) of length $K_s(i) + d + 1$. We have shown the following.

Proposition 5.4. *Suppose that A obeys the cost function $c_{\mathcal{K}}$. Then A is K-trivial.*

Note that $\sup_s c_{\mathcal{K}}(x, s) = \sum_{i>x} 2^{-K(i)}$ is bounded above by the measure of the set of strings σ such that $\mathbb{U}(\sigma) > x$. Therefore $c_{\mathcal{K}}$ satisfies the limit condition, and by Theorem 5.3 some simple set is K-trivial.

By the implication "$\Rightarrow$" of the following result, any possible construction of a K-trivial set will be similar to the one in the proof of Theorem 5.3.

Theorem 5.5 (Nies [34]). *A is K-trivial $\Leftrightarrow$ A obeys $c_{\mathcal{K}}$.*

The implication "$\Leftarrow$" is Proposition 5.4. The implication "$\Rightarrow$" is not too hard for c.e. sets ([37, 5.3.27]). For Δ^0_2 sets in general, apparently it requires the full power of the golden run method (see [37, 5.5.2]).

As an application, we show that K-triviality is closely tied to being c.e.

Corollary 5.6. *For each K-trivial set A, there is a c.e. K-trivial set $D \geq_T A$.*

Proof. Let $D = \{\langle x, i\rangle \colon A(x) \text{ changes at least } i \text{ times}\}$. Thus, when $A(x)$ changes, we put the next element in the x-th column of $\mathbb{N}$ into D. Clearly, $D(\langle x, i\rangle)$ can only change at a stage s when $A(x)$ also changes. Now $x \leq \langle x, i\rangle$ and $c_{\mathcal{K}}(y, s)$ is non-increasing in y. Thus, if A obeys $c_{\mathcal{K}}$ then D obeys $c_{\mathcal{K}}$ as well. (Note that $c_{\mathcal{K}}$ can be replaced by any monotonic cost function in this argument.) □

In [35] we introduce a cost function c_Ω simpler than $c_{\mathcal{K}}$, and show that it also characterizes K-triviality. For each stage t, let Ω_t be the measure of the domain of $\mathbb{U}$ at stage t. Now let $c_\Omega(x, s)$ be the measure of $\mathbb{U}$–descriptions converging from stage x to s, that is, $c_\Omega(x, s) = \Omega_s - \Omega_x$.

Theorem 5.7. *A is K-trivial $\Leftrightarrow$ A obeys the cost function c_Ω.*

Outline of the proof. $\Leftarrow$: Clearly $c_{\mathcal{K}}(x, s+1) - c_{\mathcal{K}}(x, s) \leq \Omega_{s+1} - \Omega_s$, which implies that $c_{\mathcal{K}}(x, s) \leq c_\Omega(x, s)$ by induction on $s \geq x$. Thus, if a set A obeys c_Ω it also obeys $c_{\mathcal{K}}$. Therefore A is K-trivial by Proposition 5.4.
$\Rightarrow$: This is a further application of the golden run method. It can also be proved directly from the foregoing Theorem. □

We say that a monotonic cost function c is *additive* if $c(x, y) + c(y, z) = c(x, z)$ for each $x < y < z$. Clearly c_Ω is additive (while $c_{\mathcal{K}}$ is not). An additive cost function c is completely determined by the non-decreasing sequence of rationals $(c(0, s))_{s\in\mathbb{N}}$ approximating the real $\sup_s c(0, s)$. Nies [35] proved that A is K-trivial iff A obeys all additive cost functions. This characterizes K-triviality of a Δ^0_2 set A purely based on effective approximations of A, and on left-c.e. reals. In contrast, the characterizations in Section 4 used machines, measure, or relativization.

5.2.2. Strongly jump traceable sets. We discuss a lowness property which is defined by purely computability-theoretic means following the weak-as-an-oracle paradigm. It properly implies K-triviality for c.e. sets. It is much stronger than slowness mentioned after Definition 5.2. We will show how it can be characterized by obeying all so-called benign cost functions.

The property is an instance of the meta-concept of traceability. The idea behind traceability is the following. The set A is computationally weak because for certain functions ψ computed with oracle A, the possible values $\psi(n)$ are contained in a finite set T_n of small size. The sets T_n are obtained effectively from n (not using A as an oracle).

Traces for functions $\omega \to \omega$ also appear in combinatorial set theory, especially forcing results related to cardinal characteristics. They are called slaloms there, and were introduced by T. Bartoszyński (see [1]).

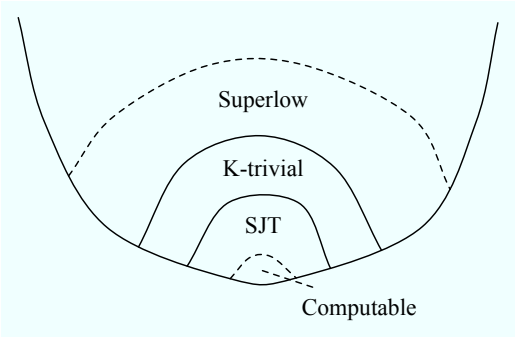
Recall that $J^A(x)$ is the value on input x of a universal A-partial computable function. We say that a computable function h with only positive values is an *order function* if it is non-decreasing and unbounded.

Definition 5.8. A *computably enumerable trace with bound* h is a uniformly computably enumerable sequence $(T_x)_{x\in\mathbb{N}}$ of finite sets such that $|T_x| \leq h(x)$ for each x.

We say that a set A is *strongly jump traceable* (SJT) if for each order function h, there is a c.e. trace $(T_x)_{x\in\mathbb{N}}$ with bound h such that, whenever $J^A(x)$ is defined, we have $J^A(x) \in T_x$.

Strong jump traceability was introduced by Figueira, Nies, and Stephan [11]. They built a simple strongly jump traceable set. Further, they show that A is SJT iff the relative Kolmogorov complexity $C^A(y)$ of a string y is not far below $C(y)$ (for each order function g we have $C(y) \leq^+ C^A(y) + g(C^A(y))$). This makes the notion an analog of being low for K.

It matters that we require *each* order function h as a bound for some trace. A much weaker notion is jump traceability, where one merely requires that there is a c.e. trace for J^A with *some* computable bound h. There is a perfect class of sets that are jump traceable as shown in [37, 8.4.4], while each SJT set is Δ^0_2 by [7].



The c.e. strongly jump traceable sets form a *proper* subclass of the c.e. K-trivial sets by Cholak, Downey, and Greenberg [5]. It is interesting to compare the two classes. Both are closed downward under $\leq_{\mathrm{T}}$. Both are closed under $\oplus$. By definition the c.e. K-trivials have a Σ^0_3 index set; in contrast, the c.e. SJTs have a Π^0_4-complete index set by Ng [30]. Thus, as already indicated by the definition, within the c.e. sets SJT is more complicated than the K-trivials as a class, even though its members are closer to being computable. Recent research of Downey and Greenberg [7] shows that in fact each SJT set (c.e. or not) is K-trivial.

Greenberg and Nies [14] characterized the c.e. SJTs according to the inertness paradigm. They specified the right class of cost functions to gauge how inert a c.e. set must be so that it is SJT. A monotonic cost function c is called *benign* if there is a computable bound $g(n)$ on the length of any finite sequence $x_0 < x_1 < \ldots < x_k$ such that $c(x_i, x_{i+1}) \geq 2^{-n}$ for each $i < k$. For instance, the cost function $c_{\mathcal{K}}$ characterizing K-triviality is benign via $g(n) = 2^n$. Further, any additive cost function is benign.

Theorem 5.9 ([14]). *Let A be c.e. Then*

A is strongly jump traceable $\Leftrightarrow$ A obeys each benign cost function.

Because of Proposition 5.4, the harder implication "$\Rightarrow$" generalizes the result of Cholak, Downey, and Greenberg [5] that SJT implies K-triviality for c.e. sets.

5.2.3. Kučera's injury free solution to Post's problem. Post [38] asked whether a c.e. set can be incomputable but also Turing incomplete. Both Friedberg and Muchnik solved the problem in 1955 by building a pair of Turing incomparable c.e. sets. To do so, they introduced the finite injury method. A further solution to Post's problem is to build a low simple set (see [41]). This construction again uses the finite injury method, because it has injury to lowness requirements. In contrast, Kučera in 1986 [20] obtained an injury-free proof of the following result, and then used it for an injury-free solution to Post's problem.

Theorem 5.10. *Suppose Y is a ML-random Δ^0_2 set. Then some simple set A is Turing below Y.*

Now let Y be the bits of Ω in the even positions. An easy direct argument involving van Lambalgen's theorem on relative randomness shows that Y is low and ML-random ([36] or [37, 3.4.10]). Therefore one can build without injury a low simple set A.

We formulate the proof of Kučera's theorem in the language of cost functions. This argument is due to Greenberg and Nies [14], and indirectly also Hirschfeldt and Miller (2006, unpublished; see Section 6 of this paper).

Proof of Theorem 5.10. Fix a computable approximation $(Y_s)_{s\in\mathbb{N}}$ of Y. We define a cost function c_Y such that, if $e < x$ and $Y_t\upharpoonright_e$ does not change for $x \leq t \leq s$, then $c_Y(x,s) < 2^{-e}$. In more detail, let $c_Y(x,s) = 2^{-x}$ for each $x \geq s$. If $x < s$, and e is least such that $Y_{s-1}(e) \neq Y_s(e)$, let $c_Y(x,s) = \max(c_Y(x,s-1), 2^{-e})$.

Fact 5.11. *If a Δ^0_2 set A obeys c_Y, then $A \leq_{\mathrm{T}} Y$ with use function bounded by the identity.*

A *Solovay test* $\mathcal{S}$ is given by an effective enumeration of strings $\sigma_0, \sigma_1, \ldots$, such that $\sum_i 2^{-|\sigma_i|} < \infty$. If Y is ML-random and $\sigma_0, \sigma_1, \ldots$ is a Solovay test, then for almost all i the string σ_i is not a prefix of Y (see [37, 3.2.19]).

To see that $A \leq_{\mathrm{T}} Y$, we enumerate a Solovay test as follows. When $A_{s-1}(x) \neq A_s(x)$ and $c_Y(x,s) = 2^{-e}$, we put the string $Y_s\upharpoonright_e$ into $\mathcal{S}$. Since A obeys c_Y, $\mathcal{S}$ is indeed a Solovay test.

Choose s_0 such that $\sigma \not\prec Y$ for any σ enumerated into $\mathcal{S}$ after stage s_0. Given an input $x \geq s_0$, using Y as an oracle, compute $t > x$ such that $Y_t\upharpoonright_x = Y\upharpoonright_x$. Then $x \in A$ implies $x \in A_t$. For, by the definition of the cost function c, at each stage $s > t$, if $c(x,s) = 2^{-e}$ (where $e \leq x$), then $Y_s\upharpoonright_e$ still has the same

value as at stage t, which is the true $Y \upharpoonright_e$. Thus, if $A_{s-1}(x) \neq A_s(x)$ we will put a prefix σ of Y into $\mathcal{S}$, contradiction. This shows Fact 5.11.

Since Y is Δ^0_2, the cost function c_Y satisfies the limit condition. Hence some simple set A obeys c_Y. So $A \leq_{\mathrm{T}} Y$. □

5.2.4. Adaptive cost functions and injury. In Theorem 5.3 we assumed that the cost function c was given in advance. In a more complicated variant, the cost function c may be defined during the construction. Such a variant is needed for the Kučera-Terwijn construction of a set that is low for ML-randomness [22], and also for Muchnik's direct construction of a set that is low for K. In the latter construction, say, $c(x, s)$ is the measure of all descriptions at stage $s-1$ such that a change at x would destroy the corresponding computation of $\mathbb{U}^A$ at stage $s-1$; that is, $c(x,s) = \sum_\sigma 2^{-|\sigma|} [\![\mathbb{U}^A(\sigma)[s-1]\downarrow \ \wedge\, x < \mathsf{use}\ \mathbb{U}^A(\sigma)[s-1]]\!]$. Extra care has to be taken now to ensure that c satisfies the limit condition. Note that this cost function is not monotonic.

If the cost function is defined during the construction, then the construction must be regarded as having injury. For instance, during the construction of a low simple set, the lowness requirements $L_e\colon\ \exists^\infty s\, J^A(e)[s-1]\downarrow \Rightarrow\ \ J^A(e)\downarrow$ are injured. The following cost function encodes the restraint imposed by L_e: if $J^A(e)$ newly converges at stage $s-1$, define $c(x,s) = \max\{c(x,s-1), 2^{-e}\}$ for each $x < \mathsf{use}\ J^A(e)[s-1]$. If A is enumerated in such a way that the total cost of changes is finite, then L_e is injured only finitely often. Thus A is low.

In contrast, a cost function c given in advance cannot be used to hide injury, because to encode a restraint that is in force at the beginning of stage s we have to know A_{s-1}.

Summary of Section 5. Cost functions arose to uniformize the constructions of Δ^0_2 sets with lowness properties. Nowadays they have turned into an important tool for understanding these lowness properties. We formulate in terms of cost functions the construction of a simple K-trivial set, and Kučera's construction of a simple set below a Δ^0_2 ML-random. We characterize the K-trivial sets and the strongly jump traceable sets in terms of obeying a class of cost functions with simple combinatorial properties: being additive for the K-trivials, and benign for the SJTs. For the K-trivials there is a universal cost function c_Ω.

Some further facts. If c is a monotonic cost function and sets A and B obey c, then $A \oplus B$ obeys c. The class of sets obeying c is closed downward under Turing reduction with use bounded by the input [35].

There is a computable enumeration $(A_s)_{s\in\mathbb{N}}$ of $\mathbb{N}$ in the order $0, 1, 2, \ldots$ such that $(A_s)_{s\in\mathbb{N}}$ does not obey $c_{\mathcal{K}}$ [37, Ex. 5.3.7]. Thus it matters in the Definition 5.2 of obedience that we require a finite total cost of changes only for *some* computable approximation.

The converse of Theorem 5.3 holds for a monotonic cost function c: if a computable approximation $(A_s)_{s\in\mathbb{N}}$ of an incomputable set A obeys c, then c satisfies the limit condition [37, Ex. 5.3.8].

6. The Turing-Below-Many Paradigm

In Theorem 5.10 we discussed the result of Kučera [20] that for every ML-random Δ^0_2 set Y there is an incomputable c.e. set $A \leq_{\mathrm{T}} Y$. If Y is Turing incomplete (i.e. $\emptyset' \not\leq_{\mathrm{T}} Y$), then A must be a base for randomness, and hence K-trivial by [15] (also see [37, 3.4.13]). Thus, for c.e. sets, being below a Turing incomplete ML-random set is a lowness property implying K-triviality. A major open question in the area is whether this property coincides with K-triviality [29, Question 4.6], [37].

Question 6.1. *Is each K-trivial set Turing below an incomplete ML-random?*

By Corollary 5.6, it is not necessary to require that the given set be c.e.

Kučera's result is our starting point for studying lowness properties of a set A according to the Turing-below-many lowness paradigm. To obtain lowness properties stronger than the ones mentioned in the previous paragraph, we strengthen the condition related to Kučera's result that $A \leq_{\mathrm{T}} Y$ for some Turing incomplete random set Y. There are two interrelated approaches:

(a) Replace the single oracle set Y by a null class $\mathcal{C} \subseteq 2^{\mathbb{N}}$ containing a ML-random set $Y \not\geq_T \emptyset'$, and require that $A \leq_{\mathrm{T}} Z$ for each ML-random set $Z \in \mathcal{C}$.

(b) Stay with a single oracle set Y, but require that it satisfy a randomness property stronger than Martin-Löf-randomness.

Both approaches lead to similar results related to strong jump traceability.

To carry out (a) the following notation is useful. For a class $\mathcal{C} \subseteq 2^{\mathbb{N}}$, let $\mathcal{C}^{\diamond}$ denote the collection of c.e. sets that are computable from all ML-random sets in $\mathcal{C}$. This "infimum" operator was implicitly introduced in unpublished work of Hirschfeldt and Miller. Each class of the form $\mathcal{C}^{\diamond}$ induces an ideal in the c.e. Turing degrees. Via cost functions Hirschfeldt and Miller showed that $\mathcal{C}^{\diamond}$ contains a simple set for each null Σ^0_3 class $\mathcal{C}$ (see [37, 5.3.15]). Since $\{Y\}$ is a Σ^0_3 class for each Δ^0_2 set Y, this strengthens Kučera's result.

A strengthening of ML-randomness as required in (b) is Demuth randomness, a notion between 2-randomness and Martin-Löf randomness that is still compatible with being Turing below $\emptyset'$ (but no longer with being above $\emptyset'$). We show that each c.e. set that is Turing below a Demuth random is strongly jump traceable. We leave open the question whether being below a Demuth random actually characterizes strong jump traceability for c.e. sets.

We give some more detail on the two approaches above.

Approach (a). By definition, the strongly jump traceable (SJT) sets are weak as an oracle. In Theorem 5.9 we discussed how to characterize the c.e. SJT sets via the inertness paradigm. Now we will characterize them via the Turing-below-many paradigm. Recall that a Δ^0_2 set Y is ω-c.e. if Y has a computable approximation with a computable bound on the number of times $Y(n)$ changes. It is

easy to obtain a ML-random ω-c.e. set. Examples are Chaitin's number Ω, or a superlow ML-random set. Thus, the following theorem of Greenberg, Hirschfeldt and Nies [13] says that a c.e. set A is strongly jump traceable iff it is Turing below many ML-random oracles.

Theorem 6.2. *Let A be c.e. Then*
A is strongly jump traceable $\Leftrightarrow$ A is Turing below each ω-c.e. ML-random set.

The implication "$\Rightarrow$" follows from Theorem 5.9: if Y is ω-c.e. then its associated cost function c_Y defined in the proof of Theorem 5.10 is benign. Since A obeys c_Y and Y is ML-random, we obtain $A \leq_{\mathrm{T}} Y$.
The implication "$\Leftarrow$" is harder. Given an order function h we want to build a c.e. trace for J^A with bound h. We threaten to build an ω-c.e. ML-random set Y such that $A \not\leq_{\mathrm{T}} Y$.

Let $(\Phi_e)_{e\in\mathbb{N}}$ be an effective list of Turing functionals. We have a tree of runs of procedures similar to the golden run method in Subsection 4.2. However, now the tree has infinitely many levels. At stage s, there is a procedure S^e_x at each level e, for each x such that $y = J^A(x)$ converges at s with use u. This procedure either shows that $A\upharpoonright_u$ is not a prefix of $\Phi_e(Y)$, or places y into a trace set T_x of size at most $h(x)$. Since $A \leq_{\mathrm{T}} Y$, at some level e there is a golden run node which always succeeds via tracing. At this node we obtain the required trace for J^A with bound h.

Since diamond classes induce ideals, as a corollary the c.e. SJT sets are closed under $\oplus$. This result was first obtained by Cholak, Downey, and Greenberg [5] who used a direct construction.

The techniques in the proof of Theorem 6.2 are very adaptable. A variant shows that the c.e. sets in SJT coincide with $\mathcal{C}^\diamond$ when $\mathcal{C}$ is the class of superlow sets. A more complex variant shows that the c.e. sets in SJT also coincide with $\mathcal{C}^\diamond$ when $\mathcal{C}$ is the class of superhigh sets Z (namely, Z' is truth-table above $\emptyset''$).

In proving the implication "$\Leftarrow$", the hypothesis is actually not needed that the given set A be c.e. for the case of superlow (and hence ω-c.e.) sets. We conclude that the same hypothesis can be discarded from the implication "$\Leftarrow$" of Theorem 5.9: suppose A obeys all benign cost functions. Then, for each ω-c.e. set Y, A obeys the benign cost function c_Y defined in the proof of Theorem 5.10. Hence $A \leq_{\mathrm{T}} Y$. Thus A is strongly jump traceable.

By [7] each SJT set is K-trivial, and hence obeys $c_{\mathcal{K}}$. However, it is not known whether the implication "$\Rightarrow$" of Theorem 5.9 works for arbitrary sets, that is, whether each SJT set obeys each benign cost function.

Approach (b). Demuth tests generalize Martin-Löf tests $(G_m)_{m\in\mathbb{N}}$ in that one can change the m-th component (a Σ^0_1 set of measure at most 2^{-m}) for a computably bounded number of times. Z fails a Demuth test if Z is in infinitely many final versions of the G_m. (For a formal definition see [37, Section 3.6].)

Greenberg [12] built a Δ^0_2 Martin-Löf random set Y such that every c.e. set computable from Y is strongly jump traceable. Subsequently, Kučera and Nies [18] showed that any Demuth random Δ^0_2 set Y serves this purpose.

Theorem 6.3. *Let Y be Demuth random. Let A be a c.e. set such that $A \leq_{\mathrm{T}} Y$. Then A is strongly jump traceable.*

The following open problem is analogous to Question 6.1.

Question 6.4. *Is each strongly jump traceable c.e. set Turing below a Demuth random?*

Acknowledgments. I thank Rod Downey, Asher Kach, Justin Moore, Eamonn O'Brien, and Christopher Porter for comments on earlier drafts of this paper.

References

[1] T. Bartoszyński. Combinatorial aspects of measure and category. *Fund. Math.*, 127(3):225–239, 1987.

[2] V. Brattka, J. Miller, and A. Nies. Computable randomness and differentiability. To appear.

[3] C. Calude and Richard J. Coles. Program-size complexity of initial segments and domination reducibility. In *Jewels are forever*, pages 225–237. Springer, Berlin, 1999.

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

[5] P. Cholak, R. Downey, and N. Greenberg. Strongly jump-traceability I: the computably enumerable case. *Adv. in Math.*, 217:2045–2074, 2008.

[6] O. Demuth. The differentiability of constructive functions of weakly bounded variation on pseudo numbers. *Comment. Math. Univ. Carolin.*, 16(3):583–599, 1975. Russian.

[7] R. Downey and N. Greenberg. Strong jump traceability II: the general case. To appear.

[8] R. Downey and D. Hirschfeldt. *Algorithmic randomness and complexity.* Springer-Verlag, Berlin. To appear.

[9] R. Downey, D. Hirschfeldt, A. Nies, and F. Stephan. Trivial reals. In *Proceedings of the 7th and 8th Asian Logic Conferences*, pages 103–131, Singapore, 2003. Singapore University Press.

[10] R. Downey, D. Hirschfeldt, A. Nies, and S. Terwijn. Calibrating randomness. *Bull. Symbolic Logic*, 12(3):411–491, 2006.

[11] S. Figueira, A. Nies, and F. Stephan. Lowness properties and approximations of the jump. *Ann. Pure Appl. Logic*, 152:51–66, 2008.

[12] N. Greenberg. A Δ^0_2 random set which only computes strongly jump-traceable c.e. sets. To appear.

[13] N. Greenberg, D. Hirschfeldt, and A. Nies. Characterizing the strongly jump traceable sets via randomness. To appear.

[14] N. Greenberg and A. Nies. Benign cost functions and lowness properties. To appear.

[15] D. Hirschfeldt, A. Nies, and F. Stephan. Using random sets as oracles. *J. Lond. Math. Soc. (2)*, 75(3):610–622, 2007.

[16] C. Jockusch, Jr. and R. Soare. Π^0_1 classes and degrees of theories. *Trans. Amer. Math. Soc.*, 173:33–56, 1972.

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

[18] A. Kučera and A. Nies. Demuth randomness and computational complexity. To appear.

[19] A. Kučera. Measure, Π^0_1-classes and complete extensions of PA. In *Recursion theory week (Oberwolfach, 1984)*, volume 1141 of *Lecture Notes in Math.*, pages 245–259. Springer, Berlin, 1985.

[20] A. Kučera. An alternative, priority-free, solution to Post's problem. In *Mathematical foundations of computer science, 1986 (Bratislava, 1986)*, volume 233 of *Lecture Notes in Comput. Sci.*, pages 493–500. Springer, Berlin, 1986.

[21] A. Kučera. On relative randomness. *Ann. Pure Appl. Logic*, 63:61–67, 1993.

[22] A. Kučera and S. Terwijn. Lowness for the class of random sets. *J. Symbolic Logic*, 64:1396–1402, 1999.

[23] S. Kurtz. *Randomness and genericity in the degrees of unsolvability.* Ph.D. Dissertation, University of Illinois, Urbana, 1981.

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

[25] M. Li and P. Vitányi. *An introduction to Kolmogorov complexity and its applications.* Graduate Texts in Computer Science. Springer-Verlag, New York, second edition, 1997.

[26] P. Martin-Löf. The definition of random sequences. *Inform. and Control*, 9:602–619, 1966.

[27] Per Martin-Löf. On the notion of randomness. In *Intuitionism and Proof Theory (Proc. Conf., Buffalo, N.Y., 1968)*, pages 73–78. North-Holland, Amsterdam, 1970.

[28] J. Miller. Every 2-random real is Kolmogorov random. *J. Symbolic Logic*, 69:907–913, 2004.

[29] J. Miller and A. Nies. Randomness and computability: Open questions. *Bull. Symbolic Logic*, 12(3):390–410, 2006.

[30] K. Ng. On strongly jump traceable reals. *Ann. Pure Appl. Logic*, 154:51–69, 2008.

[31] A. Nies. Applying randomness to computability. Series of three lectures at the ASL summer meeting, Sofia, 2009.

[32] A. Nies. Computability and randomness: Five questions. To appear.

[33] A. Nies. Low for random sets: the story. Preprint, available at `http://www.cs.auckland.ac.nz/nies/papers/`, 2005.

[34] A. Nies. Lowness properties and randomness. *Adv. in Math.*, 197:274–305, 2005.

[35] A. Nies. Calculus of cost functions. To appear.

[36] A. Nies, F. Stephan, and S. Terwijn. Randomness, relativization and Turing degrees. *J. Symbolic Logic*, 70(2):515–535, 2005.

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

[38] E. Post. Recursively enumerable sets of positive integers and their decision problems. *Bull. Amer. Math. Soc.*, 50:284–316, 1944.

[39] C.P. Schnorr. *Zufälligkeit und Wahrscheinlichkeit. Eine algorithmische Begründung der Wahrscheinlichkeitstheorie*. Springer-Verlag, Berlin, 1971. Lecture Notes in Mathematics, Vol. 218.

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

[41] R. Soare. *Recursively Enumerable Sets and Degrees*. Perspectives in Mathematical Logic, Omega Series. Springer–Verlag, Heidelberg, 1987.

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

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

[44] D. Zambella. On sequences with simple initial segments. ILLC technical report ML 1990-05, Univ. Amsterdam, 1990.