The Sixth Lecture on Algorithmic Randomness*

Rod Downey
School of Mathematics, Statistics, and Computer Science
Victoria University
PO Box 600 Wellington
New Zealand

April 17, 2007

Abstract

This paper follows on from the author's *Five Lectures on Algorithmic Randomness*. It is concerned with material not found in that long paper, concentrating on Martin-Löf lowness and triviality. We present a hopefully user-friendly account of the decanter method, and discuss recent results of the author with Peter Cholak and Noam Greenberg concerning the class of strongly jump traceable reals introduced by Figueira, Nies and Stephan.

1 Introduction

This paper is a follow-up to the author's paper *Five Lectures on Algorithmic Randomness*, Downey [8], and covers material not covered in those lectures. In particular, I plan to look at lowness which was the basis of one of my lectures in Nijmegen, and to try to make accessible the *decanter* method whose roots come from the paper Downey, Hirschfeldt, Nies and Stephan [12], and whose full development was in Nies [26]. The class of K-trivial reals has turned out to be a remarkable "natural" class with really fascinating

*Research supported by the Marsden Fund of New Zealand. Special thanks to Noam Greenberg for providing extensive corrections.

properties and connections with other areas of mathematics. For instance, K-trivial reals allow us to solve Post's problem "naturally" (well, reasonably naturally), without the use of a priority argument, or indeed without the use of requirements. Also, Kučera and Slaman [19] have used this class to solve a longstanding question in computable model theory, namely given a noncomputable Y in a Scott set S, there exists in S an element X Turing incomparable with Y.

As well I will report on recent work of the author with Peter Cholak and Noam Greenberg [4] on the computably enumerable *strongly jump traceable* reals which form a *proper* subclass of the K-trivials.

2 Notation

I will keep the notation identical to that of Downey [8] and Downey and Hirschfeldt [11]. As a brief reminder, I will use C for plain complexity and K for prefix free complexity. We will be working in Cantor space 2^ω with uniform measure $\mu([\sigma]) = 2^{-|\sigma|}$, where the subbasic clopen sets are $[\sigma] =_{\rm def} \{\sigma\alpha : \alpha \in 2^\omega\} : \sigma \in 2^{<\omega}\}$. Elements of 2^ω will be referred to as *reals*. For any other notations and the like, we refer the reader to either [8] or [11].

3 Plain complexity characterizations of computable sets

The first result establishing the fact that the complexity of initial segment of a real can have significant impact on its algorithmic complexity is due to Loveland. It concerned conditional complexity. It states that if all initial segments of a real are as compressible as they can be then the real must be computable. Notice that if α is a computable real then since there is a computable function g such that $g(n) = \alpha \upharpoonright n$ for all n, $C(\alpha \upharpoonright n|n)$ is a constant: we read n then apply g.

Theorem 3.1 (Loveland [22])**.** *Suppose that there is e such that for all x, $C(\alpha \upharpoonright x|x) \leq e$. Then α is computable. Moreover for each e there are only finitely many α with $C(\alpha \upharpoonright x|x) \leq e$ for all x. (Indeed the same is true even if we replace "for all x" by "for all x in some infinite computable set.".)*

Proof. (Sketch) The main idea in the proof below is that the possibilities for α reduce to a Π^0_1 class with a finite number of paths, and hence all such α all will be computable. If $C(\alpha \upharpoonright x|x) \leq e$ for all x, then there are at most $f = \mathcal{O}(2^e)$ many programs of size e or less. There is a maximum collection of such programs $p_1, \ldots, p_m$ which are hit infinitely often.

Working above some length (taken to be 0 for simplicity), inductively form the Π^0_1 class of strings σ as follows. For the first step, find a set of m strings $\sigma_1, \ldots, \sigma_m$ of the same length ℓ such that all of them appear to have $C(\sigma_i \upharpoonright p|p) \leq e$ for all $p \leq \ell$, and $C(\sigma_i|\ell) < e$ via p_i. For the second step, we simply use extensions of these σ_i in the same way. Then this will generate a Π^0_1 class of width at most m which must contain α. Therefore α is computable. □

The same proof will also work for any usual complexity measure such as K or KM or Km etc in place of C *for the conditional case.*

A mild generalization of Loveland's Theorem is the result of Chaitin, which gives another information-theoretical characterization of computable sets.

Theorem 3.2 (Chaitin [3])**.** *Suppose that $C(\alpha \upharpoonright n) \leq C(n) + \mathcal{O}(1)$ for all n (or for an infinite computable set of n), or $C(\alpha \upharpoonright n) \leq \log n + \mathcal{O}(1)$, for all n. Then α is computable (and conversely). Furthermore for a given constant $\mathcal{O}(1) = d$, there are only finitely many ($\mathcal{O}(2^d)$) such α.*

In the same way that Loveland's Theorem needed a basic finiteness condition to work, the same holds for Chaitin's theorem. The following is the main lemma. Let $D : \Sigma^* \mapsto \Sigma^*$ be partial computable. Then we define *D-description* of σ to be a pre-image of σ.

Lemma 3.3 (Chaitin [3])**.** *For all d and D, there is a number $f(d)$ such that for each $\sigma \in \Sigma^*$,*

$$|\{q : D(q) = \sigma \wedge |q| \leq C(\sigma) + d\}| \leq f_D(d).$$

That is, the number of D-descriptions of length $\leq C(\sigma) + d$, is bounded by an absolute constant depending upon d, D alone (and not on σ)

Proof. Let σ be given, and $k = C(\sigma) + d$. For each m there are at most $2^{k-m} - 1$ strings with at least 2^m D-descriptions of length at most k, since there are $2^k - 1$ strings in total. Given k and m we can effectively list strings

σ with at least 2^m D-descriptions of length below k, uniformly in k, m. (Wait till you see 2^m q's of length $\leq k$ with $D(q) = \nu$ and and then put ν on the list $L_{k,m}$.) The list $L_{k,m}$ has length $\leq 2^{k-m}$.

If σ has at least 2^m D-descriptions of length at most k, then σ can be specified by

- m
- a string q of length 2^{k-m},

the latter indicating the position of σ in $L_{k,m}$. This description has length bounded by $2\log m + k - m + c$ where c depends only upon D. If we choose m large enough so that $2\log m + k - m + c < k - d$, we can then get a description of σ of length $< k - d = C(\sigma)$. If we let $f(d)$ be 2^n where n is the least m with $2\log m + c + d < m$ then we are done. □

In the next lemma, we will apply Lemma 3.3 with D being the universal prefix-free machine. The next lemma tells us that there are relatively few string with short descriptions, and the number depends on d alone.

Lemma 3.4 (Chaitin [3])**.** *There is a computable h depending only on d ($h(d) = \mathcal{O}(2^d)$) such that, for all n,*

$$|\{\sigma : |\sigma| = n \wedge C(\sigma) \leq C(n) + d\}| \leq h(d).$$

Proof. Consider the partial computable function D defined via $D(p)$ is $1^{|U(p)|}$. Then let $h(d) = f_D(d)$, with f given by the previous lemma. Suppose that $C(\sigma) \leq C(n) + d$, and pick the shortest p with $U(p) = \sigma$. Then p is a D-description of n and $|p| \leq C(n) + d$. Thus there at most $f(d)$ many $p's$, and hence σ's. □

Now we can use the same Π^0_1 class argument to establish Chaitin's theorem, Theorem 3.2. We will have the width determined by Lemma 3.4, and hence if $C(\alpha \upharpoonright n) \leq \log n + c$ for all n, then α will be computable. This is because there will always be a length between k and 2^k where $C(n) = \log n$.

The same argument will also show that for monotone complexity (or any other where C is a special case) if $Km(\alpha \upharpoonright n) \leq \log n + c$ for all n, then α is computable.

Frank Stephan (see [11]) has shown that for left c.e. reals, α and β, if for all n,

$$C(\alpha \upharpoonright n) \leq C(\beta \upharpoonright n) + O(1),$$

then $\alpha \leq_T \beta$.

4 K-trivials are Δ^0_2, and there are few of them

Chaitin's Theorem 3.2 shows that if, for all n, $C(\alpha \upharpoonright n) \leq C(n) + O(1)$, then α is computable. A compact way of expressing this is to use $\alpha \leq_Q \beta$, for a measure of complexity Q, iff $Q(\alpha \upharpoonright n) \leq Q(\beta \upharpoonright n) + O(1)$. Then Chaitin's Theorem says that $\alpha \leq_C 1^\omega$ iff α is computable. Chaitin asked if the same result held for K in place of C. We define the following for K, for other measures of relative complexity, there are similar notions (most of which coincide with being computable!).

Definition 4.1 (Downey, Hirschfeldt, Nies and Stephan [12])**.** *We will call a real α K-*trivial *if $\alpha \leq_K 1^\omega$.*

The first limitation on the complexity of a K-trivial real was given by Chaitin. It relies on a analog of Lemma 3.4 above for prefix-free Kolmogorov complexity.

Theorem 4.2 (Chaitin [3], Zambella [38])**.** *For any prefix-free machine D, there is a constant d such that for all c and all σ,*

$$|\{\nu : D(\nu) = \sigma \wedge |\nu| \leq K(\sigma) + c\}| \leq d2^c.$$

The proof of Theorem 4.2 uses the Coding Theorem (Levin [21]) (see Downey [8], Downey and Hirschfeldt [11]). Essentially, if there were too many elements in $\{\nu : D(\nu) = \sigma \wedge |\nu| \leq K(\sigma) + c\}$, then we could use the total measure they provide to make an even shorter description of σ, that being a contradiction. Here is our analog of Lemma 3.4.

Theorem 4.3. *The set $KT(d, n) = \{\sigma : K(\sigma) < K(|\sigma|) + d\}$ has at most $O(2^d)$ many strings of length n.*

Proof. We build a prefix-free machine V. Suppose that $U_s(\nu) = \sigma \wedge |\nu| \leq K_s(|\sigma|) + d$. Then define $V(\nu) = |\sigma|$. Then V is prefix-free as U is. Moreover, by Theorem 4.2,

$$|\{\nu : V(\nu) = |\sigma| \wedge |\nu| \leq K(|\sigma|) + d\}| \leq c2^d.$$

But then by construction, there is a constant p such that $KT(d, n)$ has at most $pc2^d$ many members. □

Now we can run the proof that all C-trivial reals are computable using K in place of C. However, we do not know any value of K[1]. The key thing we use in the C case is that between k and 2^k there is a C-random length which will have complexity $\log p$. Using $\emptyset'$ as an oracle we *can* know K. The same argument will show the following.

Theorem 4.4 (Chaitin [3])**.** *If α is K-trivial, then $\alpha \leq_T \emptyset'$.*

This proof has the following corollary, first observed by Zambella. To state Zambella's result, we let $KT(d)$ denote the class of reals which are K-trivial with constant d, meaning that for all n,

$$K(\alpha \upharpoonright n) \leq K(n) + d.$$

Corollary 4.5 (Zambella [38])**.** *The number of elements in $KT(d)$ is $\leq b2^d$ for some constant b independent of d.*

The reader should note that to be K-trivial, it is enough that the real be K-trivial on an infinite computable set. That is, the following piece of folklore is true.

Proposition 4.6. *Suppose that we have a computable set $A = \{a_1, a_2, \dots\}$ listed in increasing order of magnitude, and for all i, $K(A \upharpoonright a_i) \leq K(a_i) + O(1)$. Then A is K-trivial.*

Proof. Let $h(n) = a_n$, be computable. Then $K(n) = K(h(n)) + O(1) = K(a_n) + O(1)$. Notice that $K(A \upharpoonright n) \leq K(a_n) + O(1)$, since to compute $A \upharpoonright n$, take the program for $A \upharpoonright h(n)$, then reconstruct n from $h(n)$ and truncate $A \upharpoonright h(n)$ to get $A \upharpoonright n$. Then $K(A \upharpoonright n) \leq K(A \upharpoonright h(n)) + O(1) \leq K(h(n)) + O(1) \leq K(n) + O(1)$. □

Zambella's Theorem, Theorem 4.5, leads one to speculate as to how many K-trivials there are. This has been investigated in unpublished work of Downey, Miller and Yu.

[1]It is also true that we don't know the value of $C(\sigma)$ for any *given* σ. *However,* we do know that there is some random length between k and 2^k for each k, and such a length m will have maximum complexity which in the C case *is* m. In the K case, we also know that there is some random length, but we don't know what its complexity is, as its complexity would be $m + K(m)$, *bounded* by $m + 2\log m$.

Definition 4.7 (Downey, Miller, Yu [14]). *Let*

$$G(d) = |KT(d)|.$$

G seems a strangely complicated object. We calculate some arithmetical bounds on G. To do this we will need the following combinatorial result.

Theorem 4.8 (First Counting Theorem, [14]). *(i)* $lim_c \frac{G(c)}{2^c} = 0$.

(ii) Indeed,

$$\sum_{c\in\mathbb{N}} \frac{G(c)}{2^c} \text{ is finite.}$$

Proof. We define $G(c,n) = |KT(c,n)|$.

Lemma 4.9. $\sum_{c\in\mathbb{N}} \frac{G(c)}{2^c} \leq \liminf_n \sum_c \frac{G(c,n)}{2^c}$.

The proof of Lemma 4.9 is almost immediate. Any finite partial sum on the left represents a finite number of K-trivials. For sufficiently large n each of these reals is isolated, so that the sum on the right exceeds that of the left.

Lemma 4.10. *There is a finite q such that, for all n,*

$$\sum_{c\in\mathbb{N}} \frac{G(c,n)}{2^c} \leq q.$$

Proof. (Of Lemma 4.10) By definition of $G(\cdot,\cdot)$ we have that for some constant q, for all n,

$$\sum_{c\in\mathbb{N}} G(c,n)2^{-K(n)-c} \leq 2\sum_{\sigma\in 2^n} 2^{-K(\sigma)}.$$

The first inequality follows from definition of $G(\cdot,\cdot)$. On the other hand, K is a minimal universal computably enumerable semi-measure (by the Coding Theorem, see [8, 11]), and hence there is a constant q such that for every n,

$$2\sum_{\sigma\in 2^n} 2^{-K(\sigma)} \leq q2^{-K(n)},$$

since the quantity on the left is a computably enumerable semi-measure. □

The proof is now finished by putting together Lemmas 4.9 and 4.10. □

Before we turn to analyzing the Turing complexity of G, we point out that the result above is relatively sharp in in the following sense.

Theorem 4.11 (Second Counting Theorem, [14])**.**

$$G(b) = \Omega(\frac{2^b}{b^2}).$$

Proof. For any string σ, we know that

$$K(\sigma 0^r) \leq^+ K(\sigma) + K(0^r) \leq^+ K(\sigma) + K(0^{|\sigma|+r}).$$

Now if we choose σ of length below $b - 2\log b$, we know that $K(\sigma) \leq b$, and hence $K(\sigma 0^r) \leq K(|\sigma 0^r|) + b$. There are $\Omega(\frac{2^b}{b^2})$ such strings σ. □

Theorem 4.12 (Downey, Miller, Yu [14])**.** $G \not\leq_T \emptyset'$.

Proof. [2] Assume that G is Δ^0_2, and hence by the Limit Lemma, $G(n) = \lim_s G(n,s)$, where this time $G(n,s)$ denotes a computable approximation to $G(n)$. Also we assume that we know k such that that $0^\omega \in KT(k)$.

Using Kraft-Chaitin, we build a machine M with coding constant d, which we know in advance. M looks for the first $c \geq k, 2^d$ such that

$$\frac{G(c)}{2^c} < 2^{-d}.$$

We can approximate the c in a Δ^0_2 manner, so that $c = \lim_s c_s$, where c_s is the stage s approximation of c.

The key idea behind the definition of M is that M wants to ensure that there are at least 2^{c-d} $KT(c)$ reals. If it does this, then we have a contradiction.

At stage s, M will pick (at most) 2^{c_s-d} strings of length s extending those of length $s-1$ already defined, and promise to give them each M-descriptions of length $K(s)+c_s-d$ (hence they are $KT(c_s)$ *strings*). It clearly has enough room to do this, since

$$2^{c_s-d}2^{-(K(s)+c_s-d)} = 2^{-K(s)}.$$

[2] If the reader is unfamiliar with the use of Kraft-Chaitin to build prefix free Turing machines, then they might delay the reading of this proof until looking at the first construction of the next section.

At stage s, if c_s has a new value then all of M's old work is abandoned and M starts building $2^{c_s-d}\ KT(c_s)$ reals which branch off of 0^ω starting at length s. If c_s has the same old value, then M continues building the $2^{c_s-d}\ KT(c_s)$ reals. Eventually c_s stabilizes and M gets to build his $2^{c-d}\ KT(c)$ reals, contradicting the definition of c. □

We remark that a crude upper bound on G is that it is computable in $\mathbf{0}'''$. It is unknown if $G \leq_T \emptyset''$, or even if this question is machine dependent.

5 The basic construction : limiting damage and quanta recycling

Solovay was the first to construct a (Δ^0_2) K-trivial real. This method was adapted by Zambella [38] and later Calude and Coles [6] to construct a c.e. K-trivial real. In [12], Downey, Hirschfeldt, Nies, and Stephan gave a new construction of a K-trivial real, and this time the real was strongly c.e., that is, the characteristic function of a c.e. set. (Independently, this had been found by Kummer in an unpublished manuscript.) As we will later see this is a priority-free and later requirement free solution to Post's problem.

Theorem 5.1. (Downey, Hirschfeldt, Nies, and Stephan [12], Calude and Coles [6], after Solovay [33]) *There is a noncomputable c.e. set B such that $B \leq_K 1^\omega$.*

Proof. The proof below is surely becoming pretty well known. We only give it for completeness. As we will later see, the only way to to construct a K-trivial real is to build a machine to demonstrate the fact that it is K-trivial. By that, if B is the relevant target set, we must show that $K(B \upharpoonright n) \leq K(n) + O(1)$. In most constructions, the $O(1)$ term is the overhead of the Recursion, or at least the s-m-n theorem.

We view this as a game: The opponent will give descriptions of n's in the construction. That is, at each stage s, he will say $K_s(n) = p$, say, by giving some string ν of length p, (and $n \leq s$) *shorter than* $K_{s-1}(n)$ and have $U(\nu) = n[s]$. It is *our* job to build a machine M, by using Kraft-Chaitin, saying saying there is a description of $B \upharpoonright n$ of length $p+1$ (the "+1" option here being for technical reasons). We enumerate an axiom $\langle p, B \upharpoonright n\rangle$.

Notice that to be a Kraft-Chaitin set we only need that the sum of 2^{-p} (indeed $2^{-(p+1)}$) for such requests is below 1. In the case that B was, say,

computable, no problem, for we are only making request of *exactly the same size* as the opponent has used. So our requests (to make M) are bounded by $\frac{1}{2}$ the amount he spends, namely $\frac{\Omega}{2}$.

Unfortunately, B being noncomputable, is not actually given to us. We will only know B_s, where $B = \cup_s B_s$; that is, approximations to the real B. Here is the main asymmetry:

We might describe $\langle p, B_s \upharpoonright n \rangle$ *but* it might be that to make B noncomputable, we will need $B_{s+1} \upharpoonright n \neq B_s \upharpoonright n$, since we enumerate n into $B_{s+1} - B_s$. *This will then entail issuing new descriptions for parts of B which have changed since the last time. In particular, we will need to describe $B_{s+1} \upharpoonright m$ for $m \in [n, s]$.*

Thus we have a *cost* of

$$c(n, s) = \sum_{n \leq m \leq s} 2^{-(K_s(m)+1)}.$$

This cost is *not* chargeable to the opponent, and hence we will have to make sure that this is limited. Thus, if we are meeting a requirement R_e saying $\overline{W_e} \neq B$, we would be prepared to do this if the cost was less than, say, $2^{-(e+1)}$.

Thus we can define

$$B_{s+1} = B_s \cup \{n : n \in W_{e,s} \wedge W_{e,s} \cap B_s = \emptyset \wedge n \geq 2e \wedge \sum_{n \leq m \leq s} 2^{-K_s(m)} \leq 2^{-(e+1)}\}.$$

In that case B will be noncomputable and we can make it K-trivial since the overall cost of the injuries is bounded by the amount we can charge against the opponent $\frac{\Omega}{2}$, plus the amount we are not compensated for, namely, $\sum_{e \in \omega} 2^{-(e+1)} = \frac{1}{2}$. The reader should realize that the $\overline{B} \neq W_e$ since $\lim_n \sup_s c(n, s) \to 0$. □

Actually, this construction can be made to have *no* visible requirements: In particular, if t is any computable function which dominated the overheads of the Recursion Theorem, then if we define, akin to the Dekker deficiency set,

$$B_{t(s+1)} = B_{t(s)} \cup \{b_n^{t(s)}, \ldots, b_{t(s)}^{t(s)}\},$$

if $K_{t(s+1)}(m) < K_{(t(s)}(m)$ for all $n \leq m \leq t(s)$, where $\overline{B_s} = \{b_j^s : s \in \omega\}$, then B is automatically K-trivial and noncomputable. Space limitations mean

that we will not discuss this variation here. hence, we refer the reader to Downey, Hirschfeldt, Nies, Terwijn [13], or Downey and Hirschfeldt [11] for more details.

6 Lowness

A very similar construction to that of the last section is due to Kučera and Terwijn [20], involving *lowness*. Let R be the collection of random sets for some randomness concept. Then R^X is the class obtained for the same concept using X as an oracle.

Definition 6.1. *We say that a set X is* R-low *if $R = R^X$. We say that X is low for R-tests, R_0-low if, for every R^X-test $\{U_n^X : n \in \mathbb{N}\}$, there is a R-test $\{V_n : n \in \mathbb{N}\}$ such that $\cap_n V_n \supseteq \cap_n U_n$.*

For instance, a set X is Martin-Löf low if the collection of reals Martin-Löf random relative to X is the same as the collection of Martin-Löf random reals. Hence X is no help in making reals non-random. Notice also that if X is R_0-low then it is automatically R-low, but the converse is not clear. However, since there is a universal Martin-Löf test, the collections Martin-Löf low and Martin-Löf low for tests coincide[3].

Martin-Löf low sets were first studied by Kučera and Terwijn [20], answering a question of van Lambalgen [37].

Theorem 6.2 (Kučera and Terwijn [20])**.** *There is a noncomputable c.e. set A that is Martin-Löf low.*

Proof. We give an alternative proof to that in [20], taken from Downey [7]. It is clear that there is a primitive recursive function f, so that $\{U^A_{f(n)} : n \in \omega\}$ is the universal Martin-Löf test relative to A. Let I^A_n denote the corresponding Solovay test. Then X is A-random iff X is in at most finitely many I^A_n. We show how to build a $\{J_n : n \in \omega\}$, a Solovay test, so that for each $[p] \in I^A_n$ is also in J_n. This is done by simple copying: if $p < s$ (as a number) is in $\cup_{j \leq s} I^{A_s}_j$ is not in $J_i : i \in s$, add it. Clearly this "test" has the desired property of covering I^A_n. We need to make A so that the "mistakes" are not too big. This is done in the *same* way as the construction of a K-trivial.

[3]Not treated in these notes are lowness notions for other kinds of randomness such as Schnorr and Kurtz randomness. To the author's knowledge there is no notion of randomness where the potentially two lowness notions have not turned out to coincide.

The crucial concept comes from Kučera and Terwijn: Let $M_s(y)$ denote the collection of intervals $\{I_n^{A_s} : n \leq s\}$ which have $A_s(y) = 0$ in their use function. Then we put $y > 2e$ into $A_{s+1} - A_s$ provided that e is least with $A_s \cap W_{e,s} = \emptyset$, and

$$\mu(M_s(y)) < 2^{-e}.$$

It is easy to see that this can happen at most once for e and hence the measure of the total mistakes is bounded by $\Sigma 2^{-n}$ and hence the resulting test is a Solovay test. The only thing we need to prove is that A is noncomputable. This follows since, for each e, whenever we see the some y with $\mu(M_s(y)) \geq 2^{-e}$, such y will *not* be added and hence this amount of the A-Solovay test will be protected. But since the total measure is bounded by 1, this cannot happen forever. □

A related concept is the following.

Definition 6.3. *We call a set X* low for K *if there is a constant d such that for all σ, $K^X(\sigma) \geq K(\sigma) - d$.*

This notion is due to An. A. Muchnik who, in unpublished work, constructed such a real. It is evident that the same method of proof (keeping the measure of the injury of the uses down) will establish the following.

Theorem 6.4 (Muchnik, unpubl.)**.** *There is a noncomputable c.e. set X that is low for K.*

The reader cannot miss the similarities in the proofs of the existence of K-trivials and K-lows and even low for K reals. These are all notions of K-antirandomness, and should somehow be related.

We will soon see, in some deep work of Nies, that these classes all *coincide*!

6.1 K-trivials solve Post's problem

The basic method introduced in this section, the *quanta pushing* or more colorfully, the *decanter*[4] method, is the basis for almost all results on the K-antirandom reals[5]. In this section we will look only at the basic method to

[4]The *decanter* method was introduced in [12] and is a linear one. It is used to prove, for example, that K-trivials solve Post's problem. The later extension, discussed later, where trees of strategies are used is due to Nies [26] and is refereed to as the *golden run* method.

[5]At least in this author's opinion, and at the present time.

aid the reader with the more difficult nonuniform applications in subsequent sections. The following result proves that they are a more-or-less natural solution to Post's problem.

Theorem 6.5 (Downey, Hirschfeldt, Nies, Stephan [12]). *If a real α is K-trivial then α is Turing incomplete.*

6.2 The Decanter Method

In this section we will motivate a very important technique for dealing with K-trivial reals now called the decanter technique, or the golden run machinery. It evolved from attempted proofs that there were Turing complete K-trivial reals, the blockage being turned around into a proof that no K-trivial real was Turing complete in the time-honoured symmetry of computability theory. Subsequently many of the artifacts of the original proof were removed and streamlined particularly by Nies and we have now what appears to be a generic technique for dealing with this area. The account below models the one from Downey, Hirschfeldt, Nies and Terwijn [13].

The following result shows that K-trivials solve Post's Problem. This will be greatly improved in the later sections.

The proof of Theorem 6.5 below runs the same way whether A is Δ^0_2 or computably enumerable. We only need the relevant approximation being $A = \cup_s A_s$ or $A = \lim_s A_s$.

6.3 The first approximation, *wtt*-incompleteness

The fundamental tool used in all of these proofs is what can be described as *amplification*. Suppose that A is K-trivial with constant of triviality b, and we are building a machine M whose coding constant within the universal machine U is known to be d.

Now the import of these constants is that if we describe n by some KC-axiom $\langle p, n\rangle$ meaning that we describe n by something of length p, and hence has *weight* 2^{-p}, then in U we describe n by something of length $p + d$ and hence the opponent at some stage s must eventually give a description of $A_s \upharpoonright n$ of length $p + b + d$. The reader should think of this as meaning that the opponent has to play *less quanta* than we do for the same effect.

What has this to do with us? Suppose that we are trying to claim that A is *not* K-trivial. Then we want to *force* U to issue too many descriptions

of A, by using up all of its quanta.

The first idea is to make the opponent play many *times on the same length and hence amount of quanta.*

The easiest illustration of this method is to show that no K-trivial is *wtt*-complete.

Proposition 6.6. *Suppose that A is K-trivial then A is wtt-incomplete.*

Proof. We assume that we are given $A = \lim_s A_s$, a computable approximation to A. Using the Recursion Theorem, we build a c.e. set B, and a prefix free machine M. We suppose that $\Gamma^A = B$ is a weak truth table reduction with computable use γ. Again by the Recursion Theorem, we can know Γ, γ and we can suppose that the coding constant is d and the constant of triviality is b as above.

Now, we pick $k = 2^{b+d+1}$ many followers $m_k < \cdots < m_1$ targeted for B and wait for a stage where $\ell(s) > m_1$, $\ell(s)$ denoting the length of agreement of $\Gamma^A = B[s]$.

At this stage we will *load* an M-description of some *fresh, unseen* $n > \gamma(m_1)$ (and hence bigger than $\gamma(m_i)$ for all i) of size 1, enumerating an axiom $\langle 1, n\rangle$. The translation of course is that at some stage s_0 we must get a description of $A_{s_0} \upharpoonright n$ in U of length $b + d$ or less. That is, at least $2^{-(b+d)}$ must enter the domain of U devoted to describing this part of A.

At the first such stage s_0, we can put m_1 into $B_{s_0+1} - B_{s_0}$ causing a change in $A \upharpoonright n - A_{s_0} \upharpoonright n$. (We remark that in this case we could use any of the m_i's but later it will be important that the m_i's enter in reverse order.) Then there must be at some stage $s_1 > s_0$, with $\ell(s_1) > m_1$, a new $A_{s_1} \upharpoonright n \neq A_{s_0} \upharpoonright n$ *also* described by something of length $b + d$. Thus U must have at least $2^{-(b+d)}$ more in its domain. If we repeat this process one time for each m_i then eventually U runs out of quanta since $2^{-(b+d)}k > 1$. □

6.4 The second approximation: impossible constants

The argument above is fine for weak truth table reducibility. There are clearly problems in the case that Γ is a *Turing* reduction, for example.

That is, suppose that our new goal is to show that no K-trivial is Turing complete. The problem with the above construction is the following.

When we play the M-description of n, we have used all of our quanta available for M to describe a single number. *Now it is in the opponents power to move the use $\gamma(m_1, s)$ (or $\gamma(m_k, s)$ even) to some value bigger than*

n before *even it decides to match our description of* n. Thus it costs him very little to match our M-description of n:

He moves $\gamma(m_k, s)$ then describes $A_s \upharpoonright n$ and we can no longer cause *any* changes of A below n, as all the Γ-uses are too big.

It is at this point that we realize it is pretty dumb of us to try to describe n in one hit. All that really matters is that we load lots of quanta beyond some point were it is measured many times. For instance, in the *wtt* case, we certainly could have used many n's beyond $\gamma(m_1)$ loading each with, say, 2^{-e} for some small e, and only attacking once we have amassed the requisite amount beyond $\gamma(m_1)$.

This is the idea behind our second step.

Impossible assumption: We will assume that we are given a Turing reduction $\Gamma^A = B$ and the overheads of the coding and Recursion Theorem result in a constant of 0 for the coding, and the constant of triviality is 0.

Hence we have $\Gamma^A = B[s]$ in some stage by stage manner, and moreover when we enumerate $\langle q, n\rangle$ into M then the opponent will eventually enumerate something of length q into U describing $A_s \upharpoonright n$. Notice that with these assumptions, in the wtt case we'd only need one follower m. Namely in the wtt case, we could load (e.g.) $\frac{7}{8}$ onto n, beyond $\gamma(m)$ and then put m into B causing the domain of U to need $\frac{7}{4}$ since we count $A_s \upharpoonright n$ for two different $A_s \upharpoonright n$-configurations.

In the case that γ is a Turing reduction, the key thing to note is that we still have the problem outlined above. Namely if we use the dumb strategy, then the opponent will change $A_s \upharpoonright \gamma(m, s)$ moving some γ-use *before* he describes $A_s \upharpoonright n$. Thus he only needs to describe $A_s \upharpoonright n$ *once*.

Here is where we use the drip feed strategy for loading. What is happening is that we really have called a procedure $P(\frac{7}{8})$ asking us to load $\frac{7}{8}$ beyond $\gamma(m)$ and then use m to count it twice. It might be that whilst we are trying to load some quanta, the "change of use" problem might happen, a certain amount of "trash"will occur. This trash corresponds to axioms enumerated into M that do not cause the appropriate number of short descriptions to appear in U. We will need to show that this trash is small enough that it will not cause us problems.

Specifically, we would use a procedure $P(\frac{7}{8}, \frac{1}{8})$ asking for twice counted quanta (we call this a 2-*set*) of size $\frac{7}{8}$ but only having trash bounded by $\frac{1}{8}$.

Now, $\frac{1}{8} = \sum_j 2^{-(j+4)}$. Initially we might try loading quanta beyond the current use $\gamma(m, s_0)$ in lots of 2^{-4}. If we are successful in reaching our target

of $\frac{7}{8}$ before A changes, then we are in the *wtt*-case and can simply change B to get the quanta counted twice.

Now suppose we load the quantum 2^{-4} on some $n_0 > \gamma(m, s_0)$. The opponent might, at this very stage, move $\gamma(m, s)$ to some new $\gamma(m, s_1) > n_0$, at essentially no cost to him. We would have played 2^{-4} for no gain, and would throw the 2^{-4} into the trash. Now we would begin to try to load anew $\frac{7}{8}$ beyond $\gamma(m, s_1)$ *but this time we would use chunks of size* 2^{-5}. Again if he moved immediately, then we would trash that quanta and next time use 2^{-6}. Notice that if we assume that $\Gamma^A = B$ this movement can't happen forever, lest $\gamma(m, s) \to \infty$.

On the other hand, in the first instance, perhaps we loaded 2^{-4} beyond $\gamma(m, s_0)$ and he did not move $\gamma(m, s_0)$ at that stage, but simply described $A \upharpoonright n_0$ by some description of size 4. At the next step, we would pick another n beyond $\gamma_(m, s_0) = \gamma(m, s_1)$ and try again to load 2^{-4}. If the opponent *now* changes, then we lose the *second* 2^{-4} but he *must* count the *first* one (on n_0) twice. That is, whenever he actually does not move $\gamma(m, s)$ then he must match our description of the current n, and this *will* later be counted *twice* since either *he* moves $\gamma(m, s)$ over it (causing it to be counted twice) *or* we put m into B making $\gamma(m, s)$ change.

Thus, for this simplified construction, each time we try to load, he either matches us(in which case the amount will contribute to the 2-set, and we can return $2^{-\text{current } \beta}$ where β is the current number being used for the loading to the target, *or* we lose β, but gain in that $\gamma(m, s)$ moves again, and we put β in the trash, but make the next $\beta = \frac{\beta}{2}$.

If $\Gamma^A = B$ then at some stage $\gamma(m, s)$ must stop moving and we will succeed in loading our target $\alpha = \frac{7}{8}$ into the 2-set. Our cost will be bounded above by $\frac{7}{8} + \frac{1}{8} = 1$.

6.5 The less impossible case

Now we will remove the simplifying assumptions. The key idea from the *wtt*-case where the use is fixed but the coding constants are nontrivial, is that we must make the changes beyond $\gamma(m_k)$ a $k-$set. Our idea is to combine the two methods to achieve this goal. For simplicity, suppose we pretend that the constant of triviality is 0, but now the coding constant is 1. Thus when we play 2^{-q} to describe some n , the opponent will only use $2^{-(q+1)}$.

Emulating the wtt-case, we would be working with $k = 2^{1+1} = 4$ and would try to construct a 4-set of changes. What we will do is break the task

into the construction of a 2-set of a certain weight, a 3-set and a 4-set of a related weight in a coherent way.

We view these as procedures P_j for $2 \leq j \leq 4$ which are called in in reverse order in the following manner.

Our overall goal begins with, say $P_4(\frac{7}{8}, \frac{1}{8})$ asking us to load $\frac{7}{8}$ beyond $\gamma(m_4, s_0)$ initially in chunks of $\frac{1}{8}$, this being a 4-set.

To do this we will invoke the lower procedures. The procedure P_j ($2 \leq j \leq 4$) enumerates a j-set C_j. The construction begins by calling P_4, which calls P_3 several times, and so on down to P_2, which enumerates the 2-set C_2 and KC set L of axioms $\langle q, n \rangle$.

Each procedure P_j has rational parameters $q, \beta \in [0, 1]$. The *goal* q is the weight it wants C_j to reach, and the *garbage quota* β is how much it is allowed to waste.

In the simplified construction, where there was only one m, the goal was $\frac{7}{8}$ and the β evolved with time. The same thing happens here. P_4's goal *never* changes, and hence can never be met lest U use too much quanta. Thus A cannot compute B.

The main idea is that procedures P_j will ask that procedures P_i for $i < j$ do the work for them, with eventually P_2 "really" doing the work, but the the goals of the P_i are determined inductively by the garbage quotas of the P_j above. Then if the procedures are canceled before completing their tasks then the amount of quanta wasted is acceptably small.

We begin the construction by starting $P_4(\frac{7}{8}, \frac{1}{8})$. Its action will be to

1. Choose m_4 large.

2. Wait until $\Gamma^A(m_4) \downarrow$.

When this happens, P_4 will call $P_3(2^{-4}, 2^{-5})$. Note that here the idea is that P_4 is asking P_3 to enumerate the 2^{-4}'s which are the current quanta bits that P_4 would like to load beyond m_4's current Γ-use. (The actual numbers being used here are immaterial except that we need to make them converge, so that the total garbage will be bounded above.

Now if, while we are waiting, the Γ-use of m_4 changes, then we will go back to the beginning. But let's consider what happens on the assumption that this has not yet occurred.

What will happen is that $P_3(2^{-4}, 2^{-5})$ will pick some m_3 large, wait for $\Gamma(m_3)$ convergence, and then it will invoke $P_2(2^{-5}, 2^{-6})$, say. This will pick its own number m_2 again large, wait for $\Gamma^A(m_2) \downarrow$ and finally now we will

get to enumerate something into L. Thus, at this very stage we would try to load 2^{-5} beyond $\gamma(m_2, s)$ in chunks of 2^{-6}.

Now whilst we are doing this, many things can happen. The simplest case is that nothing happens to the uses, and hence, as with the wtt case, we would successfully load this amount beyond $\gamma(m_2, s)$. Should we do this then we can enumerate m_2 into B and hence cause this amount to be a 2-set C_2 of weight 2^{-5} and we have reached our target.

This would return to P_3 which would realize that it now has 2^{-5} loaded beyond $\gamma(m_3, s)$, and it would like another such 2^{-5}. Thus it would again invoke $P_2(2^{-5}, 2^{-6})$. If it did this successfully, then we would have seen a 2-set of size 2^{-5} loaded beyond $\gamma(m_3, s)$ (which is unchanged) and hence if we enumerate m_3 into B we could make this a 3-set of size 2^{-5}, which would help P_4 towards its goals.

Then of course P_4 would need to invoke P_3 again and then down to P_2. The reader should think of this as "wheels within wheels within wheels" spinning ever faster.

Of course, the problems all come about because uses can change. The impossible case gave us a technique to deal with that. For example, if only the outer layer P_2 has its use $\gamma(m_2, s)$ change, then as we have seen, the amount already matched would still be a 2-set, but the latest attempt would be wasted. We would reset its garbage quota to be half of what it was, and then repeat. Then we could rely on the fact that (assuming that all the other procedures have m_i's with stable uses) $\lim_s \gamma(m_2, s) = \gamma(m_2)$ exists, eventually we get to build the 2-set of the desired target with acceptable garbage, build ever more slowly with ever lower quanta.

In general, the inductive procedures work the same way. Whilst waiting, if uses change, then we will initialize the lower procedures, reset their garbages to be ever smaller, but not throw away any work that has been successfully completed. Then in the end we can argue by induction that all tasks are completed.

Proof of Theorem 6.5. We are now ready to describe the construction. Let $k = 2^{b+d+1}$ The method below is basically the same for all the constructions with one difference as we later see.

As in the wtt case, our construction will build a k-set C_k of weight $> 1/2$ to reach a contradiction.

The procedure P_j $(2 \leq j \leq k)$ enumerates a j-set C_j. The construction begins by calling P_k, which calls P_{k-1} several times, and so on down to P_2,

which enumerates L (and C_2).

Each procedure P_j has rational parameters $q, \beta \in [0,1]$. The *goal* q is the weight it wants C_j to reach, and the *garbage quota* β is how much it is allowed to waste.

We now describe the procedure $P_j(q,\beta)$, where $1 < j \leq k$, and the parameters $q = 2^{-x}$ and $\beta = 2^{-y}$ are such that $x \leq y$.

1. Choose m large.
2. Wait until $\Gamma^A(m)\downarrow$.
3. Let $v \geq 1$ be the number of times P_j has gone through step 2.

   $j = 2$: Pick a large number n. Put $\langle r_n, n\rangle$ into L, where $2^{-r_n} = 2^{-v}\beta$. Wait for a *stage* t such that $K_t(n) \leq r_n + d$, and put n into C_1. (If M_d is a prefix-free machine corresponding to L, then t exists.)

   $j > 2$: Call $P_{j-1}(2^{-v}\beta, \beta')$, where $\beta' = \beta 2^{j-k-w-1}$) and w is the number of P_{j-1} procedures started so far.

   In any case, if $\text{weight}(C_{j-1}) < q$ then repeat step 3, and otherwise return.
4. Put m into B. This forces A to change below $\gamma(m) < \min(C_{j-1})$, and hence makes C_{j-1} a j-set (if we assume inductively that C_{j-1} is a $(j-1)$-set). So put C_{j-1} into C_j, and declare $C_{j-1} = \emptyset$.

If $\gamma^A(m)$ changes during the execution of the loop at step 3, then cancel the run of all subprocedures, and go to step 2. Despite the cancellations, C_{j-1} is now a j-set because of this very change. (This is an important point, as it ensures that the measure associated with numbers already in C_{j-1} is not wasted.) So put C_{j-1} into C_j, and declare $C_{j-1} = \emptyset$.

This completes the description of the procedures. The construction consists of calling $P_k(\frac{7}{8}, \frac{1}{8})$ (say). It is easy to argue that since quotas are inductively halved each time they are injured by a use change, they are bounded by, say $\frac{1}{4}$. Thus L is a KC set. Furthermore C_k is a k-set, and this is a contradiction since then the total quanta put into U exceeds 1. □

The following elegant description of Nies' is taken from [13]:

We can visualize this construction by thinking of a machine similar to Lerman's pinball machine (see [34, Chapter VIII.5]). However, since we

enumerate rational quantities instead of single objects, we replace the balls in Lerman's machine by amounts of a precious liquid, say 1955 Biondi-Santi Brunello wine. Our machine consists of decanters $C_k, C_{k-1}, \ldots, C_0$. At any stage C_j is a j-set. We put C_{j-1} above C_j so that C_{j-1} can be emptied into C_j. The height of a decanter is changeable. The procedure $P_j(q, \beta)$ wants to add weight q to C_j, by filling C_{j-1} up to q and then emptying it into C_j. The emptying corresponds to adding one more A-change.

The emptying device is a hook (the $\gamma^A(m)$-marker), which besides being used on purpose may go off finitely often by itself. When C_{j-1} is emptied into C_j then $C_{j-2}, \ldots, C_0$ are spilled on the floor, since the new hooks emptying $C_{j-1}, \ldots, C_0$ may be much longer (the $\gamma^A(m)$-marker may move to a much bigger position), and so we cannot use them any more to empty those decanters in their old positions.

We first pour wine into the highest decanter C_0, representing the left domain of L, in portions corresponding to the weight of requests entering L. We want to ensure that at least half the wine we put into C_0 reaches C_k. Recall that the parameter β is the amount of garbage $P_j(q, \beta)$ allows. If v is the number of times the emptying device has gone off by itself, then P_j lets P_{j-1} fill C_{j-1} in portions of size $2^{-v}\beta$. Then when C_{j-1} is emptied into C_j, at most $2^{-v}\beta$ much liquid can be lost because of being in higher decanters $C_{j-2}, \ldots, C_0$. The procedure $P_2(q, \beta)$ is special but limits the garbage in the same way: it puts requests $\langle r_n, n\rangle$ into L where $2^{-r_n} = 2^{-v}\beta$. Once it sees the corresponding $A \upharpoonright n$ description, it empties C_0 into C_1 (but C_0 may be spilled on the floor before that because of a lower decanter being emptied).

6.6 K-trivials form a robust class

It turns out that the K-trivials are a remarkably robust class, and coincide with a host of reals defined in other ways. This coincidence also has significant degree-theoretical implications. For example, as we see, not only are the K-trivials Turing incomplete, but are closed downwards under Turing reducibility and form a natural Σ^0_3 ideal in the Turing degrees.

What we need is another view of the decanter method. In the previous section it was shown that K-trivials solve Post's Problem. Suppose however, we actually applied the method above to a K-trivial and a partial functional Γ. Then what would happen would be that for some i the procedure P_i *would not return.* This idea forms the basis for most applications of the decanter method, and the run that does not return would be called a *golden run.* This

idea is from Nies [26]

For instance, suppose that we wanted to show Nies' result that all K-trivials are superlow.

Let A be K-trivial. Our task is to build a functional $\Gamma^K(e)$ computing whether $\Phi_e^A(e)\downarrow$. For ease of notation, let us denote $J^A(e)$ to be the partial function that computes $\Phi_e^A(e)$. Now the obvious approach to this task is to monitor $J^A(e)[s]$. Surely if $J^A(e)[s]$ never halts then we will never believe that $J^A(e)\downarrow$. However, we are in a more dangerous situation when we see some stage s where $J^A(e)[s]\downarrow$. If we define $\Gamma^K(e) = J(e)[s]$ then we risk the possibility that this situation could repeat itself many times since it is in the opponent's power the changes $A_s \upharpoonright \varphi_e(e,s)$.

Now if *we* were building A, then we would know what to do. We should *restrain* A in a familiar way and hence with finite injury A is low. However, the *opponent* is building A, and *all we know is that A is K-trivial.*

The main idea is to load up quanta beyond the use of the e-computation, before we change the value of $\Gamma^A(e)[s]$, that is changing our belief from divergence to convergence. Then, if A were to change on that use after we had successfully loaded, it would negate our belief and causing us to reset $\Gamma^K(e)$. Our plan is to make sure that *it would cost the opponent, dearly.*

As with the proof above, this cannot happen too many times for any particular argument, and in the construction to be described, there will be a golden run which does not return. The interpretation of this non-returning is that the $\Gamma_R^K(e)$ *that this run R builds* will actually work. Thus, the construction of the lowness index is *non-uniform.*

Thus, the idea would be to have *tree* of possibilities. The height of the tree is $k = 2^{b+d+1}$, and the tree is ω branching. A node $\sigma^\frown i$ denotes the action to be performed for $J^A(i)$ more or less assuming that now σ is the highest priority node that does not return.

At the top level we will be working at a procedure $P_k(\frac{7}{8},\frac{1}{8})$ yet again, and we know in advance that this won't return with a k-set of that size.

What we do is distribute the tasks out to the successors of λ, the empty node. Thus outcome e would be devoted to solving $J^A(e)$ via a k-set. It will be given quanta, say, $2^{-(e+1)}\alpha_k$ where $\alpha_k = \frac{7}{8}$. (Here this choice is arbitrary, save that it is suitably convergent. For instance, we could use the series $\sum_{n\in\mathbb{N}}\frac{1}{n^2}$ which would sharpen the norms of the wtt-reductions to n^2.) To achieve its goals, when it sees some apparent $J^{(}A)(e)\downarrow[s]$, It will invoke $P_k(2^{-(e+1)}\alpha_k, 2^{-e}\beta_k)$ where $\beta_k = \frac{1}{8}$. We denote this procedure by $P_k(e, 2^{-(e+1)}\alpha_k, 2^{-(e+1)}\beta_k)$. This procedure will look for a k-set of the appro-

priate size, which, when it achieves its goal, the version of Γ at the empty string says it believes.

Again, notice that if this P_k returns (and this is the idea below) 2^{e+2} many times then $P_k(\alpha_k, \beta_k)$ would return, which is impossible.

As with the case of Theorem 6.5, to achieve its goals, *before it believes that* $J^A(e) \downarrow [s]$, it needs to get its quanta by invoking the team via nodes below e. These are, of course, of the form $e^\frown j$ for $j \in \omega$. They will be asked to try to achieve a $k-1$ set and $P_k(e, 2^{-(e+1)}\alpha_k, 2^{-(e+1)}\beta_k)$ by using $P_{k-1}(e^\frown j, 2^{(e+1)}2^{-(j+1)}\beta_k, 2^{-(j+1)}2^{-(e+1)}\beta_{k-1})$, with $\beta_{k-1} << \beta_k$ and j chosen appropriately. Namely we will choose those j, say, with $j > \varphi_e^A(e)[s]$. That is, these j will, by convention, have their uses beyond $\varphi_e^A(e)[s]$ and hence will be working similarly to the "next" m_i "down" in the method of the incompleteness proof of Theorem 6.5. Of course such procedures would await $\varphi_e^A(j)[s]$ and try to load quanta in the form of a $k-1$ set beyond the $\varphi_j^A(j,s)$ $(> \varphi_e^A(e))$, the relevant j-use.

The argument procedures working in parallel work their way down the tree. As above when procedure $\sigma^\frown i$ is injured because the $J^A(t)[s]$ is unchanged for all the uses of $t \in \sigma$, yet $J^A(i)[s]$ changes before the procedure returns then we reset all the garbage quotas in a systematic way, so as to make the garbage quota be bounded.

Now the argument is the same. There is some m least σ of length m, and some final α, β for which $P_m(e, \alpha, \beta)$ is invoked and never returns. Then the procedure built at σ will be correct on all $j > \varphi_e^A(e)$. Moreover, we can always calculate how many times it would be that some called procedure would be invoked to fulfill $P_m(\alpha, \beta)$. Thus the can bound the number of times that we would change our mind on $\Gamma^K_{P_m(\alpha,\beta)}(i)$ for any argument i. That is, A is superlow. In fact, as Nies pointed out, this gives a little more. Recall that an *order* is a computable nondecreasing function with infinite limit.

Definition 6.7 (Nies [24, 26])**.** *We say that a set B is jump traceable iff there is a computable order h and a weak array (or not necessarily disjoint c.e. sets) $\{W_{g(j)} : j \in \mathbb{N}\}$, such that $|W_{g(j)}| \leq h(j)$, and $J^B(e) \in W_{g(e)}$.*

Theorem 6.8 (Nies [24, 26])**.** *Suppose that A is K-trivial. Then A is jump traceable.*

The proof is to observe that we are actually constructing a trace. A mild variation of the proof above also shows the following.

Theorem 6.9 (Nies [24]). *Suppose that A is K-trivial. Then there exists a K-trivial computably enumerable B with $A \leq_{tt} B$.*

Proof. (sketch) Again the golden run proof is more or less the same, our task being to build B. This is farmed out to outcomes e in the ω-branching tree, where we try to build $\Gamma^B(e) = A(e)$. Again at level j, the size of the use will be determined by the number of times the module can act before it returns enough quanta to give the node above the necessary $j-1$ set. This is a computable calculation. Now when the opponent seeks to load quanta beyond $\gamma^j(e,s)$ before we believe this, we will load matching quanta beyond e for A. The details are then more or less the same. □

Other similar arguments show that K-triviality is basically a *computably enumerable* phenomenon[6]. That is, the following is true.

Theorem 6.10 (Nies [26]). *The following are equivalent*

(i) A is K-trivial.

(ii) A has a Δ^0_2 approximation $A = \lim_s A_s$ which reflects the cost function construction. That is,

$$\{\sum_{x \leq y \leq s} \frac{1}{2} c(y,s) : x \text{ minimal with } A_s(x) \neq A_{s-1}(x)\} < \frac{1}{2}.$$

6.7 More characterizations of the K-trivials

We have seen that the K-trivials are all jump traceable. In this section, we sketch the proofs that the class is characterized by other "antirandomness" properties.

Theorem 6.11 (Nies and Hirschfeldt [26]). *Suppose that A is K-trivial. Then A is low for K.*

Corollary 6.12. *The following are equivalent:*

(i) A is K-trivial.

[6] To the author's knowledge, K-triviality is the only example of a fact in computability theory that relies purely on enumerations. It would appear, for instance, that forcing the existence of a K-trivial is impossible.

(ii) A is low for Martin-Löf randomness.

(iii) A is low for K.

Proof. The corollary is immediate by the implication (iii)→(ii)→(i). We prove Theorem 6.11. Again this is another golden run construction. This proof proceeds in a similar way to that showing that K-trivials are low, except that $P_{j,\tau}$ calls procedures $P_{j-1,\sigma}$ based on computations $U^A(\sigma) = y[s]$ (since we now want to enumerate requests $\langle |\sigma| + d, y\rangle$), and the marker $\gamma(m, s)$ is replaced by the use of this computation. That is, we wish to believe a computation, $U^A(\sigma) = y[s]$ and to do so we want to load quanta beyond the use $u(\sigma, s)$. This is done more or less exactly the same way, beginning at P_k and descending down the nodes of the tree. Each node ν will this time build a machine $\hat{U}_\nu$, which will copy ν-believed computations; namely those for which we have successfully loaded the requisite $|\nu|-$set. We need to argue that for the golden ν, the machine is real. The garbage is bounded by, say, $\frac{1}{8}$ by the way we reset it. The machine otherwise is bounded by U itself. □

Corollary 6.13 (Nies [26])**.** *The K-trivials are closed downward under $\leq_T$.*

From Downey [8], we know that real addition + induces a join on the Solovay (and hence K- and C-) degrees of left c.e. reals. It is not hard to show that the following also holds.

Theorem 6.14 (Downey, Hirschfeldt, Nies and Stephan [12])**.** *Suppose that α and β are K-trivial. Then so is $\alpha + \beta$, and hence $\alpha \oplus \beta$.*

Proof. Assume that α, β are two K-trivial reals. Then there is a constant c such that $K(\alpha \upharpoonright n)$ and $K(\beta \upharpoonright n)$ are both below $K(n) + c$ for every n. By Theorem 4.3 there is a constant d such that for each n there are at most d strings $\tau \in \{0,1\}^n$ satisfying $K(\tau) \leq K(n) + c$. Let $e = n^*$ be the shortest program for n. One can assign to $\alpha \upharpoonright n$ and $\beta \upharpoonright n$ numbers $i, j \leq d$ such that they are the i-th and the j-th string of length n enumerated by a program of length up to $|e| + c$.

Let U be a universal prefix-free machine. We build a prefix-free machine V witnessing the K-triviality of $\alpha + \beta$. Representing i, j by strings of the fixed length d and taking $b \in \{0,1\}$, $V(eijb)$ is defined by first simulating $U(e)$ until an output n is produced and then continuing the simulation in order to find the i-th and j-th string α and β of length n such that both are generated by a program of size up to $n + c$. Then one can compute

$2^{-n}(\alpha + \beta + b)$ and derive from this string the first n binary digits of the real $\alpha + \beta$. These digits are correct provided that e, i, j are correct and b is the carry bit from bit $n+1$ to bit n when adding α and β – this bit is well-defined unless $\alpha + \beta = z \cdot 2^{-m}$ for some integers m, z, but in that case $\alpha + \beta$ is computable and one can get the first n bits of $\alpha + \beta$ directly without having to do the more involved construction given here. Notice that $\alpha \oplus \emptyset \leq_K \alpha$ trivially, and hence if α and β are K-trivial, $\alpha \oplus \beta$ will also be K-trivial. $\square$

Since the K-trivials are closed downwards in $\leq_T$, we have the following.

Corollary 6.15 (Nies [26]). *The K-trivials form a Σ^0_3 ideal in the Turing degrees.*

The K-trivial form the only known such ideal. It is known ([12]) that the ideal is not principal, and that there is a low_2 computably enumerable degree above this ideal (Nies, see Downey and Hirschfeldt [11] for a proof). (It is known that no low c.e. degree can be above the K-trivials, but it is quite possible that there is a Δ^0_2 low degree above them all, perhaps even a Martin-Löf random one. This is an apparently difficult open question.)

In passing we mention two further characterizations of the K-trivials. First we can define B be a *base of a cone of Martin-Löf randomness* if $B \leq_T A$ where A is B-random. Kučera was the first to construct such a noncomputable set B.

Theorem 6.16 (Hirschfeldt, Nies and Stephan [17]). *A is K-trivial iff A is a base of a cone of Martin-OLöf randomness.*

Finally, in recent work there has been a very surprising new characterization of K-triviality. We will say that A is *low for weak $2-$randomness tests* iff for all Π^A_2 nullsets $\mathcal{N}$, there is a Π^0_2 nullset $\mathcal{M} \supseteq \mathcal{N}$.

Theorem 6.17 (Downey, Nies, Weber, Yu [15]+ Nies [29]+Miller [23]). *A is K-trivial iff A is low for weak $2-$randomness.*

7 A proper subclass

7.1 Jump traceability and strong jump traceability

Again we return to the theme of jump traceability. Recall that that a set B is jump traceable iff there is a computable order h and a weak array

$\{W_{g(j)} : j \in \mathbb{N}\}$, such that $|W_{g(j)}| \leq h(j)$, and $J^B(e) \in W_{g(e)}$. We would say that A is *jump traceable via the order* h. Recall that Nies [26] showed that if A is K-trivial, then A is jump traceable.

Such considerations lead Figueira, Nies and Stephan to investigate a new class of reals.

Definition 7.1 (Figueira, Nies and Stephan [16])**.** *We say that A is* strongly jump traceable *iff for* all *(computable) orders h, A is jump traceable via h.*

Nies' Theorem shows that if A is K-trivial then it is jump traceable via $h(e)$ about $e \log e$. Interestingly, Figueira, Nies and Stephan [16] showed that there are $2^{\aleph_0}$ many reals which are jump traceable at order $h(e) = 2^{2^e}$. Using a rather difficult argument, Downey and Greenberg proved the following.

Theorem 7.2 (Downey and Greenberg, unpubl.)**.** *For $h(e) = \log \log e$ all reals jump traceable with order $h(e)$ are Δ^0_2 and hence there are only countable many.*

It is not altogether clear that *strongly* jump traceable reals should exist.

Theorem 7.3 (Figueira, Nies and Stephan [16])**.** *There exist c.e. promptly simple strongly jump traceable sets.*

Proof. The following proof is due to Keng Meng Ng. It is really a Π^0_2 argument since the guess that a particular partial computable function is an order is a Π^0_2 fact. There is a promptly simple c.e. set A, which is strongly jump traceable.

Requirements

We build an c.e. set A satisfying the following requirements :

$$\begin{array}{rcl} \mathcal{P}_e & : & W_e \text{ is infinite } \Rightarrow \exists x, s(x \in W_{e,\text{ats}} \wedge x \in A_{s+1}), \\ \mathcal{N}_e & : & h_e \text{ is an order } \Rightarrow A \text{ is jump traceable via } h_e. \end{array}$$

Here, h_e is the e^{th} partial computable function of a single variable. The negative requirement $\mathcal{N}_e$ will build the sequence $V_{e,0}, V_{e,1}, \cdots$ of c.e. sets such that $|V_{e,i}| \leq h_e(i)$ and $J^A(i) \in V_{e,i}$ for all i, if h_e is an order.

Strategy

We will describe the strategy used to satisfy $\mathcal{N}_0$. The general strategy for $\mathcal{N}_e$ is similar. Suppose that h_0 is an order function, our aim is to build

the uniformly c.e. sequence $V_{0,0}, V_{0,1}, \cdots$. Consider the sequence of intervals $I_1, I_2, \cdots$, initial segments of $\mathbb{N}$ such that $h_0(x) = n$ for all $x \in I_n$.

For $i \in I_1$, whenever $J^A(i)[s] \downarrow$ with use $u(i)$, we would enumerate the value $J^A(i)[s]$ into $V_{0,i}$ and preserve the value $J^A(i)[s]$ by preventing any positive requirement from enumerating an $x \in A \upharpoonright_{u(i)}$. If $i' \in I_2$ and $J^A(i')[s'] \downarrow$ with use $u(i')$ at stage s', we will also enumerate $J^A(i')[s']$ into $V_{0,i'}$. Since $V_{0,i'}$ can take two values, it is therefore not essential that the computation $\{i'\}^A(i')[s']$ at stage s' be preserved forever. We could allow $\mathcal{P}_0$ to make an enumeration below $u(i')$ (but above $\max\{u(k) \mid k \in I_0\}$), and block all other positive requirements $\mathcal{P}_1, \mathcal{P}_2, \cdots$ from enumerating below $u(i')$. When a new value $J^A(i')[s'']$ appears after $\mathcal{P}_0$ acts, it will be put into $V_{0,i'}$ and the computation $\{i'\}^A(i')[s'']$ preserved forever.

In general, we would allow the requirements $\mathcal{P}_0, \cdots, \mathcal{P}_{n-2}$ to enumerate below the use of $J^A(x)[t]$ at any stage t and $x \in I_n$. This ensures that for $x \in I_n$ there will be at most n values placed in $V_{0,x}$. Therefore, $\mathcal{N}_0$ will impose different restraint on each positive requirement $\mathcal{P}_0, \mathcal{P}_1, \cdots$. In particular, the restraint imposed by $\mathcal{N}_0$ on $\mathcal{P}_e$ at a stage s is $r_0(e, s) >$ the use of any computation $J^A(x)[s]$, where $x \in I_1 \cup \cdots \cup I_{e+1}$.

The above strategy is designed to work in the case where h_0 is an order. At a stage s we could compute the values $h_0(0), \cdots, h_0(s)$ up to s steps, and use the values computed to see if h_0 looks like an order. If the first $l(s)$ (the length of convergence) many convergent values do not form a non-decreasing sequence, then we could cease all action for $\mathcal{N}_0$. On the other hand, if h_0 is a non-decreasing function such that $\lim_{n\to\infty} h_0(n) = m$, then at every stage s, the first $l(s)$ many convergent values of h_0 will always form a non-decreasing sequence. This would result in the positive requirements $\mathcal{P}_{m-1}, \mathcal{P}_m, \cdots$ having restraint $\to \infty$. To prevent this situation, we declare a number x to be active (at some stage s), if the length of convergence $l(s) > x$, and $h_0(x) < h_0(l(s))$. The requirement $\mathcal{N}_0$ would only act on those numbers i which are active, for if h_0 is indeed an order, it does no harm for us to wait until i becomes active before making enumerations into $V_{0,i}$.

Construction of A. We arrange the requirements in the order $P_0 < N_0 < P_1 < N_1 < \cdots$. Let $J^A(i)[s]$ denote the value of $\{i\}_s^{A_s}(i)$ if it is convergent, with the use $u(i, s)$, and $V_{e,i}[s]$ denote $V_{e,i}$ in the s^{th} stage of formation. For each e, let the length of convergence of h_e at stage s be defined as

$$l(e, s) = \max\{y \leq s \mid (\forall x \leq y)\ (h_{e,s}(x) \downarrow\ \wedge\ h_e(x) \geq h_e(x-1))\}.$$

A number i is said to be e-active at stage s, if $i < l(e, s)$ and $h_e(i) <$

$h_e(l(e,s))$. That is, a number i will become e-active when the length of convergence of h_e exceeds i, and we have received further confirmation that $h_e(x)$ is not an eventually constant function. For each $k < e$, we let $r_k(e,s)$ be the restraint imposed by $\mathcal{N}_k$ on the e^{th} positive requirement $\mathcal{P}_e$ at stage s, defined as follows

$$r_k(e,s) = \max\{u(i,s) \mid i \text{ is } k\text{-active at stage } s \ \wedge \ h_k(i) \leq e+1\}.$$

We will let $r(e,s) := \max\{r_k(e,s) \mid k < e\}$. At stage s, we say that $\mathcal{P}_e$ requires attention if $A \cap W_{e,s} = \emptyset$, and $\exists x \in W_{e,s} - W_{e,s-1}$ such that $x > 2e$ and $x > r(e,s)$.

The construction at stage s involves the following actions :

(i) Pick the least $e < s$ such that $\mathcal{P}_e$ requires attention, and enumerate the least $x > \max\{2e, r(e,s)\}$ and $x \in W_{e,s} - W_{e,s-1}$ into A. For each $k \geq e$, we set $V_{k,i}[s+1] = \emptyset$ for all i.

(ii) For each $e < s$ we do the following for the sake of $\mathcal{N}_e$: For every currently e-active number i, we enumerate $J^A(i)[s]$ (if convergent) into $V_{e,i}[s]$.

Verification

Firstly note that for each e,

$$\exists^\infty s \ \ J^A(e)[s]\downarrow \ \Rightarrow J^A(e)\downarrow . \tag{1}$$

To see this for each e, pick an index $i > e$ such that $\forall n(h_i(n) = n)$. Since each positive requirement only enumerates at most one element into A, let s be a stage such that no $\mathcal{P}_k$ for any $k \leq i$ ever receives attention after stage s, and e is i-active after stage s. If $J^A(e)[t]\downarrow$ infinitely often, let $t > s$ be such that $J^A(e)[t]\downarrow$. Then, $A_t \upharpoonright_{u(e,t)} = A \upharpoonright_{u(e,t)}$ and so $J^A(e)\downarrow$. Note that this implies A is low, but A will actually be jump traceable and hence superlow.

It is not difficult to see that (1) implies $\lim_{s\to\infty} r(e,s) < \infty$ for every e : Fix a $k < e$, and we will argue that $\lim_{s\to\infty} r_k(e,s) < \infty$. There can only be finitely many numbers i such that i eventually becomes k-active with $h_k(i) \leq e+1$, and for each such i, $u(i,s)$ has to be bounded (or undefined) by (1).

Hence every $\mathcal{P}_e$ will be satisfied, and A is coinfinite and promptly simple. Next, we fix an e where h_e is an order. Fix an i, and we shall show that $|V_{e,i}| \leq h_e(i)$ and $J^A(i) \in V_{e,i}$. Let s be the last stage at which $V_{e,i}[s]$ is reset

to $\emptyset$, and $t \geq s$ be the smallest stage such that i becomes e-active at stage t. Enumerations into $V_{e,i}$ start only after stage t (where i becomes active), and furthermore no requirement $\mathcal{P}_k$ for any $k > h_e(i) - 2$ can ever enumerate a number $x < u(i, t')$ at any stage $t' > t$. This means there are at most $h_e(i)$ many different values in $V_{e,i}$. Lastly if $J^A(i) \downarrow$ then $J^A(i) = J^A(i)[t'']$ for some $t'' > t$, and $J^A(i)[t'']$ is enumerated into $V_{e,i}[t'']$ at that stage. □

Notice that the proof is yet another cost function construction. (Here the cost is how many things can potentially be put into some $V_{e,i}$.) Indeed, at first blush, it would seem that we are only getting the same class as the K-trivials.

Theorem 7.4 (Cholak, Downey, Greenberg [4])**.** *Suppose that A is c.e. and strongly jump traceable. Then A is K-trivial. Indeed if A is c.e. and jump traceable via an order of size $\sqrt{\log e}$, then A is K-trivial.*

Proof. Let A be strongly jump-traceable. As with the proof that K-trivials are low for K, we will need to cover U^A by an oracle-free machine, obtained via the Kraft-Chaitin theorem. When a string σ enters the domain of $U^A[s]$ we need to decide whether we believe the A-computation that put σ in dom U^A. In the "K-trivial implies low for K" proof we put weight beyond the use of the relevant computation to enable us to certify it. In this construction, we will use another technique.

To test the $\sigma \in$ dom$U^A[s]$ computation, let the use be u and let $\rho = A \upharpoonright u$ (at that stage). The naive idea is to pick some input x and define a function Ψ^A (which is partial computable in A) on the input x, with A-use u and value ρ. This function is traced by a trace $\langle T_x \rangle$; only if ρ is traced do we believe it is indeed an initial segment of A and so believe that $U^A(\sigma)$ is a correct computation. We can then enumerate $(|\sigma|, U^A(\sigma))$ into a Kraft-Chaitin set we build and so ensure that $K(U^A(\sigma)) \leq^+ |\sigma|$.

However, we need to make sure that the issued commands are a KC set. This would of course be ensured if we only believed correct computations, as $\mu(\text{dom } U^A)$ is finite. However, the size of most T_x is greater than 1, and so an incorrect ρ may be believed. We need to limit the mass of the errors.

A key new idea from Cholak, Downey and Greenberg [4] is that, rather than treat each string σ individually, we batch strings up in pieces of mass. When we have a collection of strings in dom U^A whose total mass is 2^{-k} we verify A up to a use that puts them all in dom U^A. The greater 2^{-k} is, the more stringent the test will be (ideally, in the sense that the size of T_x is

smaller). We will put a limit m_k on the amount of times that a piece of size 2^{-k} can be believed and yet be incorrect. The argument will succeed if

$$\sum_{k<\omega} m_k 2^{-k}$$

is finite.

The reader should realize that once we use an input x to verify an A-correct piece, it cannot be used again for any testing, as $\Psi^A(x)$ becomes defined permanently. Following the naive strategy, we would need at least 2^k many inputs for testing pieces of size 2^{-k}. Even a single error on each x (and there will be more, as the size of T_x has to go to infinity) means that $m_k \geq 2^k$ is too large. The rest of the construction is a combinatorial strategy: which inputs are assigned to which pieces in such a way as to ensure that the number of possible errors m_k is sufficiently small. The strategy has two ingredients.

First, we note that two pieces of size 2^{-k} can be combined into a single piece of size $2^{-(k-1)}$. So if we are testing one such piece, and another piece, with comparable use, appears, then we can let the testing machinery for $2^{-(k-1)}$ take over. Thus, even though we need several testing locations for 2^{-k} (for example if a third comparable piece appears), at any stage, the testing at 2^{-k} is really responsible for at most one such piece.

The naive reader would imagine that it is now sufficient to let the size of T_x (for x testing 2^{-k}-pieces) be something like k and be done. However, the opponent's spoiling strategy would be to "drip-feed" small mass that aggregates to larger pieces only slowly (this is similar to the situation in decanter constructions.) In particular, fixing some small 2^{-k}, the opponent will first give us k pieces (of incomparable use) one after the other (so as to change A and remove one before giving us a new one.) At each such occurrence we would need to use the input x devoted to the first 2^{-k} piece, because at each such stage we only see one. Once the amount of errors we get from using x for testing is filled (T_x fills up to the maximum allowed size) the opponent gives us one correct piece of size $2^{-(k-1)}$ and then moves on to gives us k more incorrect pieces which we test on the next x. Overall, we get k errors on *each* x used for 2^{-k}-pieces. As we already agreed that we need something like 2^k many such x's, we are back in trouble.

Every error helps us make progress as the opponent has to give up one possible value in some T_x; fewer possible mistakes on x are allowed in the future. The solution is to make every single error count in our favour in all

future testings of pieces of size 2^{-k}. In other words, what we need to do is to maximize the benefit that is given by a single mistake; we make sure that a single mistake on *some* piece will mean one less possible mistake on *every* other piece.

In the beginning, rather than just testing a piece on a single input x, we test it simultaneously on a large set of inputs and only believe it is correct if the use shows up in the trace of every input tested. If this is believed and more pieces show up then we use them on other large sets of inputs. If, however, one of these is incorrect, then we later have a large collection of inputs x for which the number of possible errors is reduced. We can then break up this collection into 2^k many smaller collections and keep working only with such x's.

This can be geometrically visualized as follows. If the naive strategy was played on a sequence of inputs x, we now have an m_k-dimensional cube of inputs, each side of which has length 2^k. In the beginning we test each piece on one hyperplane. If the testing on some hyperplane is believed and later found to be incorrect then from then on we work in that hyperplane, which becomes the new cube for testing pieces of size 2^{-k}; we test on hyperplanes of the new cube. If the size of T_x for each x in the cube is at most m_k then we never "run out of dimensions".

Further details can be found in Cholak, Downey and Greenberg [4], and Downey and Hirschfeldt [11]. □

We remark *en passant* that it is unknown if this result is true with the hypothesis that A is c.e. is removed. Downey and Greenberg have conjectured that the answer is yes.

But finally we have an example of a class of reals, defined by cost functions, where we actually get a *proper* subclass of the K-trivials.

Theorem 7.5 (Cholak, Downey, Greenberg [4])**.** *The c.e. strongly jump traceables form a* proper *subclass of the K-trivials. Again this is true at tracing order* $\log\log e$.

Proof. The easiest way to understand this proof is that the reader realize the following: since the K-trivials are closed under $\leq_T$, there must be ones of minimal degree, and hence not n-c.e. for any n. How would we make such a real directly? Now we have already seen that the only way to make K-trivials is to use a cost function construction. Thus in the basic construction, we will pick some n and monitor $c(n, s)$, the weight of the tail at s. If this

was simply making A noncomputable, should that weight be too large, we would abandon this n and choose some $n' > s$.

Now if we are to make A not k-c.e. for any k, then we would need to perhaps put n into and take it out of A many times, perhaps $k+1$ times.

The *problem* is that each time we change $A_s \upharpoonright n$, we must pay some *uncompensated* price $c(n,s)$. The basic idea, the reader will recall, is to keep this price bounded. The plan is to use a decanter kind of idea. Suppose, for instance, $k = 2$ so we need 3 attacks on some n. We would have an overall cost we are willing to pay of, say, $2^{-(e+1)}$, for this requirement R_e. Then the idea is to think of this cost as, initially, $[2^{-(e+2)}, 2^{-(e+4)}, 2^{-(e+6)}]$ where the *first* attack via some n will only cost *at most* $2^{-(e+6)}$. If we see that this cost exceeded at some stage s, *before* the first attack, we could abandon this attack at no cost choosing a new $n' > s$.

However, should we have done the first attack, we would choose a new $n' > s$ *but* give n' the quotas
$[2^{-(e+2)}, 2^{-(e+4)}, 2^{-(e+9)}]$ (if the attack was abandoned *before* the second attack occurred, *or* give n' the quotas $[2^{-(e+2)}, 2^{-(e+6)}, 2^{-(e+9)}]$ should this happen *after* we did the second attack. (In general the third attack gets numbers of the form $2^{-(e+3p)}$ and the second $2^{-(e+2t)}$ (or any suitably convergent series). This is very similar to the decanter method.

Now in our argument, we need to make A K-trivial, yet not strongly jump traceable. Again we will have a suitable slowly growing order h about $\log\log e$ is enough. To kill some possible trace $W_{g(e)}$ we will control parts of the jump. Suppose that we are dealing with part where he is supposed to be able to jump trace with at most k members of $W_{g(e)}$. (This is some e we can put things into the jump.)

Then the idea is simple. We would pick some $a_s > s$ and put e into the jump with axiom s, e, a_s, saying $J^A(e) = s$ if $a_s \notin A$. Once this appears in $W_{g(e)}[s']$ for some $s' > s$, we can then remove this from $J^A(e)[t]$ defining it to be t instead of s by simply putting a_s into $A_{t+1} - A_t$ and this new value having a new use a_{t+1}. This would need to repeat itself at most $k+1$ times. Each stage would cost us $c(a_t, t)$. Then, the argument is the same as the one above. We simply need a combinatorial counting argument to calculate how long the interval where $h(e) = k$ needs to be that we *must* succeed on some follower.

Details again can be found in Cholak, Downey, Greenberg [4]. □

Notice the property of being strongly jump traceable is something closed

downwards under $\leq_T$. Until very recently, it is an interesting open question as to whether they form an ideal. Keng Meng Ng constructed a strongly jump traceable c.e. set whose join with any strongly jump traceable set is strongly jump traceable. That construction was a careful analog of the construction of an *almost deep* c.e. degree by Cholak, Groszek and Slaman [5], which was a c.e. degree $\mathbf{a}$ such that for all low c.e. degree $\mathbf{b}$, $\mathbf{a}\cup$ was also low. The proof relied on a characterization of a c.e. set X being strongly jump traceable due to Nies, Figueira and Stephan: X is strongly jump traceable iff X' is *well-approximable* meaning that for all computable orders h, there is a computable enumeration $X'(z) = \lim_s f(z,s)$ with $|\{s : f(z,s+1) \neq f(z,s)\}| \leq h(z)$.

Theorem 7.6 (Ng [32])**.** *There is a promptly simple c.e. set A, such that if W is a strongly jump traceable c.e. set, then $A \oplus W$ is strongly jump traceable.*

In an earlier version of the present paper we sketched a proof of Ng's Theorem. Recently, Cholak, Downey and Greenberg indeed verified that the c.e. strongly jump traceable degrees form an ideal. It is this last result whose proof we will sketch.

Theorem 7.7 (Cholak, Downey and Greenberg [4])**.** *Suppose that A and B are c.e. and strongly jump traceable. Then so is $A \oplus B$.*

Proof. (Sketch) Actually something more is proven. It is shown that given an order h we can construct a slower order k such that if A and B are jump traceable via k then $A \oplus B$ is jump traceable via h. The opponent must give us $W_{p(x)}$ jump tracing A and $W_{q(x)}$ jump tracing B, such that

$$|W_{p(x)}|, |W_{p(x)}| < k(x),$$

for all x. It is *our* task to construct a trace V_z tracing $J^{A\oplus B}(z)$ with $|V_z| < h(z)$. There are two obstacles to this task. We will treat them in turn.

Fundamentally, what happens is that we see some *apparent* jump computation $J^{A\oplus B}(x) \downarrow [s]$. The question is, *should we believe this computation?* The point is that we only have at most $h(x)$ many slots in the trace V_x to put possible values. (We will think of the V_x as a *box* of height x.) The opponent can change the computation by changing either A or B after stage s on the use.

Our solution is to build another part of the jump to test the A and B parts. Of course these locations are given by the Recursion Theorem. There

will be many parts devoted to a single x. For each x the strategies will operate separately. We will denote parts of the jump for testing the A use by $a(x,i)$ and for the B side $b(x,i)$ for $i \in \omega$. (The reason for the large number will be seen later. It is kind of like a decanter of *infinite depth*, and is because of the *noncompletion* problem we need to solve.)

The basic idea is this: when stage s occurs for some a and b we will define

$$J^B[s](b) = j_B(x,s) \text{ and } J^A[s](a) = j_A(x,s),$$

where $j_C(x,s)$ denotes the C-use of the $J^{A\oplus B}(x)[s]$ computation.

Now of course this is not quite correct. We can do this, but *before* the real jump returns either of these computations, they can go away since A or B ranges on the relevant use. In particular, although we know that real jump values must occur in $W_{p(x)}$ and $W_{q(x)}$, the jump computations we have *purported* to define can become divergent on account of the relevant oracle changing. We will call this the *noncompletion problem*, and discuss its solution later.

To demonstrate the first idea, we will assume that this problem will not arise, so that the procedures return. That is, we see $j_A(x,s)$ occur in it trace: $W_{p(a)}$ and similarly for B. This would happen at some sage $t > s$ where without loss of generality, we can assume the $J^{A\oplus B}(x)$ computation is still around. Then at such a stage we would be prepared to believe it, by putting its present value v_1 into the first slot of V_x.

The simplest case is that we actually were working in the situation where the $W_{p(a)}$ and $W_{q(b)}$ were of size 1 (1-boxes) then we would be done. The computation for $A \oplus B$ *is* correct.

In the more general case we would have, say, the A and B boxes of, say, size 2, and the $A \oplus B$ one of size, say, 3. We will, as seen below, *manufacture* 1-boxes, if necessary.

Now, if the $A \oplus B$ computation is wrong, at least one of the A or B ones are too. We have arranged that the size of the V_x will be much bigger than that of the A and B. If both sides are wrong, or are shown wrong then there is are false jump computations in both of the $W_{p(a)}$ and $W_{q(b)}$. In that case then the boxes are now, in effect, 1-boxes as the top slot is filled with a false jump computation. Then the next time we get a $A \oplus B$ computation, we would be safe, assuming that we get a return.

If only one side, say the A side, is incorrect, then we have come to the first problem. The B-box $W_{q(b)}$ actually is returning correctly a jump computation

and is thus useless for testing more computations. The next test would involve possibly the same a but would need a different b, and this could alternate.

The idea is to use more than one a, b as we now see. At the beginning we could use *two* A boxes and *two* B boxes of size 2. Suppose that, as above, we get a return on all of them and the A side was wrong. Then now we have *two promoted* 1-boxes. Then the next time we test a $A \oplus B$ computation, we could use only one of them and another B-box of size 2. Since the A-computation now must be correct, if the believed computation is wrong, it must be the B side which wrong the next time, now creating a new B-1-box. Finally the third time we test, we would have two 1-boxes.

This is all very fine, but the fact that we might not get a return into the boxes causes really deep problems. The problem is that we might enumerate into the boxes two A and B configurations and the computation might occur in only one side, say the A side, before the $A \oplus B$ computation vanishes. We have not used up any V_x slot, *but the probe is that the A side might be correct and hence that box is now useless.* The reader should not that this is even a problem if we were dealing with 1-boxes. What could happen is that the change side (before return) could alternate rendering the boxes corresponding to the (correct) other side useless, and we would run out of small boxes before the oracle decided to return correctly.

The idea is to use a decanter-like strategy to get rid of this problem. Initially, to test the 2 boxes for A and B we begin at 3-boxes. These will be *metaboxes* in the sense that they are amalgams of some large number of 3 boxes. Say, $W_{p(a_1)}, \ldots, W_{p(a_n)}$ and $W_{q(b_1)}, \ldots, W_{q(b_n)}$. Before we believe the $A \oplus B$ computation at x, we begin by testing the A side, then the B side one box at a time, alternating. Thus $W_{p(a_1)}$, $W_{q(b_1)}$, $W_{p(a_2)}$, and so forth, only moving on to the next box if the previous one returns, and hence the $A \oplus B$ computation remains unchanged. Now two things can happen.

The first possibility is that we get to the end of this process. It is only then that we would move to the 2-boxes and try to test as above. If we actually get to the end of the of this final procedure with no $A \oplus B$ change, then we would then return and use a slot of V_x.

The other possibility is that at some stage of this process one of A or B changes on the x-use. Suppose that this is A. The key new idea is then if this is in the the last 2-box testing phase, then we have *created* many new 2-boxes, since the top slot of all of the $W_{p(a_i)}$ are filled with false jump computations of A. (In this case we have also created new A 1-boxes.) Box promotion is to

a lesser extent true for the case where this fails in the first phase. We have also created A 1-boxes

Notice that all of the B boxes used in this processes are likely now useless. The function k will be slow enough growing that there will be plenty more 3 boxes for later work. Then the idea is that at the next try we would begin at the 4 boxes and recursively travel down to the 2-boxes only when we get returns from the higher boxes. (But at the 3 box stage, we would be using half of the now promoted original A metabox.) The key thing to observe is that for a correct computation, we must eventually make progress as the killed side always promotes. This can only happen finitely often since the height of the relevant metabox is fixed and the killed side will promote that box.

The details are a little messy but this is the general idea. Full details can be found in the paper [4]. □

Several questions suggest themselves. First is it true that each strongly jump traceable is bounded by a c.e. strongly jump traceable? We have seen that they form an ideal in the Turing degrees? How complicated is the ideal in the c.e. case? It would seem likely that it could be Π^0_4 complete.

Finally we remark that there are several other examples of cost function constructions in the literature, whose relationship with the strongly jump traceables and the K-trivials is not yet clear. For example Nies has proven the following.

Theorem 7.8 (Nies [28])**.** *There exists a c.e. set A such that for all B, if B is random then $A \oplus B$ does not compute $\emptyset'$.*

We know that such sets must be K-trivial. The question is whether this can be reversed. Barmpalias [1] has related material here, extending Nies' Theorem above. Finally, we will call a set A *almost complete* if $\emptyset'$ is K-trivial relative to A, and $A \leq_T \emptyset'$. Such sets can be constructed from the K-trivial construction and the pseudo-jump theorem. Hirschfeldt[7] (unpubl) has shown that there are c.e. (necessarily K-trivial) reals below all such *random* reals. The question is whether they coincide with the K-trivials.

[7] A simpler proof was found by Hirschfeldt and Miller and appears in Nies [30].

8 What have I left out this time?

While I hope that this is the last lecture in the series, I should point out that there is a lovely series of results concerning lowness for other randomness notions. Terwijn and Zambella [36] characterized lowness for Schnorr randomness tests in terms of *computable traceability*, and this was extended by Kjos-Hanssen, Nies and Stephan [18] to the class of Schnorr randoms. Nies [26] proved that only the computable reals are low for computable randomness, and Downey and Griffiths [10], and later Stephan and Yu [35] investigated lowness for Kurtz randomness. To treat these and other related results properly would take another paper, and hence I will simply refer the reader to the source papers, or to Downey-Hirschfeldt [11], or Nies [31], and the recent survey Nies [30] for details.

References

[1] Barmpalias, G., *Random non-cupping revisited,* J. Complexity. Vol. 22 (2006) 850-857

[2] Chaitin, G., *A theory of program size formally identical to information theory*, Journal of the Association for Computing Machinery 22 (1975), pp. 329-340.

[3] Chaitin, G. *Information-theoretical characterizations of recursive infinite strings,* Theoretical Computer Science, vol. 2 (1976), 45-48.

[4] Cholak, P., R. Downey and N. Greenberg, *On strongly jump traceable reals*, in preparation.

[5] Cholak, P., M. Groszek, and T. Slaman, *An almost deep degree*, J. Symbolic Logic, Vol. 66, No.2 (2001), 881–901

[6] Calude, C., and Coles, R. *Program size complexity of initial segments and domination relation reducibility,* in *Jewels are Forever*, (J. Karhümaki, H. Mauer, G. Paŭn, G. Rozenberg, eds.) Springer-Veralg, New York, 1999, 225-237.

[7] Downey, R., *Some Computability-Theoretical Aspects of Reals and Randomness,* in *The Notre Dame Lectures* (P. Cholak, ed) *Lecture Notes in Logic* Vol. 18, Association for Symbolic Logic, 2005, 97-148.

[8] Downey, R., *Five Lectures on Algorithmic Randomness,* to appear proceedings of *Computational Prospects of Infinity* World Scientific.

[9] Downey, R. and N. Greenberg, *Strongly jump traceable reals are* Δ^0_2*,* in preparation.

[10] Downey, R. and E. Griffiths, *Schnorr randomness*, Journal of Symbolic Logic, Vol 69 (2) (2004), 533-554.

[11] Downey, R. and D. Hirschfeldt, *Algorithmic Randomness and Complexity*, Springer-Verlag, in preparation.

[12] Downey, R., D. Hirschfeldt, A. Nies, and F. Stephan, *Trivial reals*, extended abstract in *Computability and Complexity in Analysis* Malaga, (Electronic Notes in Theoretical Computer Science, and proceedings, edited by Brattka, Schröder, Weihrauch, FernUniversität, 294-6/2002, 37-55),July, 2002. Final version appears in (Downey, R., Ding Decheng, Tung Shi Ping, Qiu Yu Hui, Mariko Yasuugi, and Guohua Wu (eds)), *Proceedings of the 7th and 8th Asian Logic Conferences*, World Scientific, 2003, viii+471 pages.

2003, 103-131.

[13] Downey, R., D. Hirschfeldt, A. Nies, and S. Terwijn, *Calibrating randomness,* Bulletin Symbolic Logic. Vol. 12 (2006), 411-491.

[14] Downey, R., J. Miller, and L. Yu, *On the quantity of* K*-trivial reals,* in preparation.

[15] Downey, R., A. Nies, R. Weber, and L. Yu, *Lowness and* Π^0_2 *nullsets*, Journal of Symbolic Logic, Vol. 71 (2006), 1044-1052.

[16] Figueira, S., A. Nies, and F. Stephan, *Lowness properties and approximations of the jump,* in *Proceedings of the Twelfth Workshop of Logic, Language, Information and Computation (WoLLIC 2005),* Electronic Lecture Notes in Theoretical Computer Science, Vol. 143 (2006), 45-57.

[17] Hirschfeldt, D, A. Nies, and F. Stephan, *Martin-Löf oracles*, in preparation.

[18] Kjos-Hanssen, B., A. Nies, and F. Stephan, *Lowness for the class of Schnorr random sets,* to appear, SICOMP.

[19] Kučera, A., and T. Slaman, *Turing Incomparability in Scott Sets*, Proc. Amer. Math. Soc.. to appear

[20] Kučera, A., and S. Terwijn, *Lowness for the class of random sets,* Journal of Symbolic Logic, vol. 64 (1999), 1396-1402.

[21] Levin, L., *Some Theorems on the Algorithmic Approach to Probability Theory and Information Theory*, Dissertation in Mathematics, Moscow, 1971.

[22] Loveland, D. *A variant of the Kolmogorov concept of complexity*, Information and Control, vol. 15 (1969), 510-526.

[23] Miller, J., personal communcation.

[24] Nies, A., *Reals which compute little*, *Proceedings of Logic Colloquium 2002,* (Chatzidakis, Z, Koepke, P. and Pohlers, W., editors), Lecture Notes in Logic 27, Springer-Verlag (2002), 261-275.

[25] Nies, A., *Low for random reals: the story*, unpublished.

[26] Nies, A., *Lowness properties and randomness*, Advances in Mathematics, Vol. 197 (2005), 274-305.

[27] Nies, A., *Each Low(CR) set is computable*, typeset manuscript, January 2003.

[28] Nies, A., *Non-cupping and randomness*, to appear, Proceedings of the AMS.

[29] Nies, A., personal communication.

[30] Nies, A., *Eliminating concepts*, to appear *proceedings of Computational Prospects of Infinity*, World Scientific.

[31] A. Nies, *Computability and Randomness*, to appear.

[32] Ng, Keng Meng, *On strongly jump traceable reals.* in preparation.

[33] Solovay, R., *Draft of paper (or series of papers) on Chaitin's work*, unpublished notes, May, 1975, 215 pages.

[34] Soare, R., *Recursively enumerable sets and degrees* (Springer, Berlin, 1987).

[35] Stephan, F. and L. Yu, *Lowness for weakly 1-generic and Kurtz-random* in *Theory and Applications of Models of Computation: Third Internationa l Conference, TAMC 2006, Beijing, China, May 15-20, 2006*, Proceedings. Springer LNCS, 3959:756-764, 2006.

[36] Terwijn, S., and D. Zambella, *Algorithmic randomness and lowness,* Journal of Symbolic Logic, vol. 66 (2001), 1199-1205.

[37] van Lambalgen, M., *Random Sequences,* Ph. D,. Diss. University of Amsterdam, 1987.

[38] Zambella, D., *On sequences with simple initial segments*, ILLC technical report, ML-1990-05, University of Amsterdam, 1990.