TRANSACTIONS OF THE
AMERICAN MATHEMATICAL SOCIETY
Volume 367, Number 3, March 2015, Pages 1771–1796
S 0002-9947(2014)06089-3
Article electronically published on November 12, 2014

INHERENT ENUMERABILITY OF STRONG JUMP-TRACEABILITY

DAVID DIAMONDSTONE, NOAM GREENBERG, AND DANIEL D. TURETSKY

Abstract. We show that every strongly jump-traceable set obeys every benign cost function. Moreover, we show that every strongly jump-traceable set is computable from a computably enumerable strongly jump-traceable set. This allows us to generalise properties of c.e. strongly jump-traceable sets to all such sets. For example, the strongly jump-traceable sets induce an ideal in the Turing degrees; the strongly jump-traceable sets are precisely those that are computable from all superlow Martin-Löf random sets; the strongly jump-traceable sets are precisely those that are a base for $\mathrm{Demuth}_{\mathrm{BLR}}$ randomness; and strong jump-traceability is equivalent to strong superlowness.

1. Introduction

An insight arising from the study of algorithmic randomness is that anti-randomness is a notion of computational weakness. While the major question driving the development of effective randomness was "what does it mean for an infinite binary sequence to be random?", fairly early on Solovay [27] defined the notion of K-trivial sets, which are the opposite of Martin-Löf random sequences in that the prefix-free Kolmogorov complexity of their initial segments is as low as possible. While Chaitin [4, 5] showed that each K-trivial set must be Δ^0_2, a proper understanding of these sets has only come recently through work of Nies and his collaborators (see for example [9, 15, 21, 22]). This work has revealed that K-triviality is equivalent to a variety of other notions, such as lowness for Martin-Löf randomness, lowness for K, and being a base for 1-randomness. These other notions express computational weakness, either as the target of a computation or as an oracle: they either say that a set is very easy to compute, or is a weak oracle and cannot compute much.

The computational weakness of K-trivial sets is reflected in more traditional measures of weakness studied in pure computability theory. For example, every K-trivial set has a low Turing degree. Recent developments in both pure computability and in its application to the study of randomness have devised other notions of computational weakness, and even hierarchies of weakness, and attempted to calibrate K-triviality with these notions. One such attempt uses the hierarchy of *jump-traceability*.

While originating in set theory (see [26]), the study of traceability in computability was initiated by Terwijn and Zambella [28, 29].

Definition 1.1. A *trace* for a partial function $\psi\colon \omega \to \omega$ is a sequence $T = \langle T(z)\rangle_{z<\omega}$ of finite sets such that for all $z \in \operatorname{dom}\psi$, $\psi(z) \in T(z)$.

Received by the editors October 25, 2011 and, in revised form, January 27, 2013.
2010 *Mathematics Subject Classification.* Primary 03D25, 03D28, 03D32.
All authors were supported by the Marsden Fund of New Zealand, the first and the third as postdoctoral fellows. The second author was also supported by a Rutherford Discovery Fellowship.

Thus, a trace for a partial function ψ indirectly specifies the values of ψ by providing finitely many possibilities for each value; it provides a way of "guessing" the values of the function ψ. Such a trace is useful if it is easier to compute than the function ψ itself. In some sense the notion of a trace is quite old in computability theory. W. Miller and Martin [19] characterised the hyperimmune-free degrees as those Turing degrees $\mathbf{a}$ such that every (total) function $h \in \mathbf{a}$ has a computable trace (the more familiar, but equivalent, formulation, is in terms of domination). In the same spirit, Terwijn and Zambella used a uniform version of hyperimmune-freeness to characterise lowness for Schnorr randomness, thereby giving a "combinatorial" characterisation of this lowness notion.

In this paper we are concerned not with how hard it is to compute a trace, but rather, how hard it is to enumerate it.

Definition 1.2. A trace $T = \langle T(z) \rangle$ is *computably enumerable* if the set of pairs $\{(x, z) : x \in T(z)\}$ is c.e.

In other words, if, uniformly in z, we can enumerate the elements of $T(z)$. It is guaranteed that each set $T(z)$ is finite, and yet if T is merely c.e., we do not expect to know when the enumeration of $T(z)$ ends. Thus, rather than using the exact size of each element of the trace, we use effective bounds on this size to indicate how strong a trace is: the fewer options for the value of a function, the closer we are to knowing what that value is. The bounds are known as order functions; they calibrate rates of growth of computable functions.

Definition 1.3. An *order function* is a non-decreasing, computable and unbounded function h such that $h(0) > 0$. If h is an order function and $T = \langle T(z) \rangle$ is a trace, then we say that T is an *h-trace* (or that T is *bounded by* h) if for all z, $|T(z)| \leqslant h(z)$.

In addition to measuring the sizes of c.e. traces, order functions are used to define uniform versions of traceability notions. For example, *computable traceability*, the uniform version of hyperimmune-freeness used by Terwijn and Zambella, is defined by requiring that traces for functions in a hyperimmune-free degree $\mathbf{a}$ are all bounded by a single order function.

Zambella (see Terwijn [28]) observed that if A is low for Martin-Löf randomness, then there is an order function h such that every function computable from A has a c.e. h-trace. This was improved by Nies [21], who showed that one can replace total by partial functions. In some sense it is natural to expect a connection between uniform traceability and K-triviality; if every function computable (or partial computable) from A has a c.e. h-trace, for some slow-growing order function h, then the value $\psi(n)$ of any such function can be described by $\log n + \log h(n)$ many bits.

Following this, it was a natural goal to characterise K-triviality by tracing, probably with respect to a family of order functions. While partial results have been obtained [1,16], this problem still remains open. The point is that while K-triviality has been found to have multiple equivalent definitions, all of these definitions use analytic notions such as Lebesgue measure or prefix-free Kolmogorov complexity in a fundamental way, and the aim is to find a purely combinatorial characterisation for this class.

An attempt toward a solution of this problem led to the introduction of what seems now a fairly fundamental concept, which is not only interesting in its own right, but now has been shown to have deep connections with randomness.

Definition 1.4 (Figueira, Nies, and Stephan [11])**.** Let h be an order function. An oracle $A \in 2^\omega$ is *h-jump-traceable* if every A-partial computable function has a c.e. h-trace. An oracle is *strongly jump-traceable* if it is h-jump-traceable for every order function h.

Figueira, Nies, and Stephan gave a construction of a non-computable strongly jump-traceable c.e. set. Their construction bore a strong resemblance to the construction of a K-trivial c.e. set. J. Miller and Nies [18] asked if strong jump-traceability and K-triviality coincided.

Cholak, Downey, and Greenberg [6] answered this question in the negative. They showed however that one implication holds, at least for c.e. sets: the strongly jump-traceable c.e. sets form a proper subclass of the c.e. K-trivial sets. They also showed that restricted to c.e. sets, the strongly jump-traceable sets share a pleasing feature with the K-trivials, in that they induce an ideal in the c.e. Turing degrees.

In view of these results it might seem that strong jump-traceability might be an interesting artifact of the study of randomness, but as it turned out, the class of c.e., strongly jump-traceable sets has been shown to have remarkable connections with randomness. Greenberg, Hirschfeldt, and Nies [12] proved that a c.e. set is strongly jump-traceable if and only if it is computable from every superlow random sets, if and only if it is computable from every superhigh random set. Greenberg and Turetsky [14] complemented work of Kučera and Nies [17] and showed that a c.e. set is strongly jump-traceable if and only if it is computable from a Demuth random set, thus solving the Demuth analogue of the random covering problem, which was only recently solved [2].

The restriction to c.e. sets appeared to be a major technical drawback. The major tool introduced in [6] for working with strongly jump-traceable oracles, called the *box-promotion* method, works well for c.e. oracles; but technical difficulties restricted its application to other sets. Early on, Downey and Greenberg showed that all strongly jump-traceable sets are Δ^0_2, and more recently in [7], they showed that all such sets are in fact K-trivial, giving a full implication, not restricted to c.e. sets. In this paper we show how to overcome the difficulties in adapting the box-promotion method to work with arbitrary strongly jump-traceable oracles and to yield the following definitive result.

Theorem 1.5. *Every strongly jump-traceable set is computable from a c.e., strongly jump-traceable set.*

This shows that strong jump-traceability, much like K-triviality, is *inherently enumerable*. It cannot be obtained by devising a suitable notion of forcing, but essentially, only through a computable enumeration. While it is impossible for every strongly jump-traceable set to be c.e., as this notion is closed downward in the Turing degrees, Theorem 1.5 says this downward closure is the only reason for the existence of non-c.e., strongly jump-traceable sets.

Theorem 1.5 has a slew of corollaries. It enables us to extend characterisations of c.e. strong jump-traceability to all strongly jump-traceable sets.

Corollary 1.6. *The Turing degrees of strongly jump-traceable sets form an ideal in the Turing degrees.*

Proof. The Turing degrees of c.e., strongly jump-traceable sets form an ideal in the c.e. Turing degrees [6]. □

Figueira, Nies, and Stephan introduced a notion seemingly stronger than strong jump-traceability, called *strong superlowness*, which can be characterised using plain Kolmogorov complexity.

Corollary 1.7. *A set is strongly jump-traceable if and only if it is strongly superlow.*

Proof. Figueira, Nies, and Stephan [11] showed that every strongly superlow set is strongly jump-traceable, and that the notions are equivalent on c.e. sets. Strong superlowness is also closed downward in the Turing degrees. □

Unlike K-triviality, strong jump-traceability has both combinatorial and analytic characterisations.

Corollary 1.8. *A set is strongly jump-traceable if and only if it is computable from all superlow Martin-Löf random sets.*

Proof. In [12] it is shown that every set computable from all superlow 1-random sets is strongly jump-traceable, and that every c.e., strongly jump-traceable set is computable from all superlow 1-random sets. □

We remark that the results of [12] imply that every strongly jump-traceable set is computable from all superhigh random sets, but we do not yet know if all sets computable from all superhigh random sets are strongly jump-traceable.

Another connection between strong jump-traceability and randomness passes through a notion of randomness stronger than Martin-Löf's, introduced by Demuth. As mentioned above, the Demuth analogue of the incomplete Martin-Löf covering problem was solved by Greenberg and Turetsky, giving yet another characterisation of c.e. strong jump-traceability. This characterisation cannot, of course, be extended to all sets, since every Demuth random is computable from itself. The analogue of the covering problem for all sets is the notion of a *base* for randomness: a set A is a *base* for a relativisable notion of randomness $\mathcal{R}$ if A is computable from some $\mathcal{R}^A$-random set. Hirschfeldt, Nies, and Stephan [15] showed that a set is a base for Martin-Löf randomness if and only if it is K-trivial. On the other hand, while every base for Demuth randomness is strongly jump-traceable (Nies [24]), these two notions do not coincide (Greenberg and Turetsky [14]). However, this relies on the full relativisation of Demuth randomness. Recent work of Bienvenu, Downey, Greenberg, Nies, and Turetsky [3] discovered a partial relativisation of Demuth randomness, denoted $\text{Demuth}_{\text{BLR}}$, which is better behaved than its fully-relativised counterpart.

Corollary 1.9. *A set is strongly jump-traceable if and only if it is a base for* $\text{Demuth}_{\text{BLR}}$ *randomness.*

Proof. Nies [24] showed that every set which is a base for Demuth randomness is strongly jump-traceable. An examination of his proof, though, shows that he uses the hypothesis of being a base for Demuth randomness by building a Demuth test, and this test has a computable bound. In other words, his proof shows that every set which is a base for $\text{Demuth}_{\text{BLR}}$ randomness is strongly jump-traceable.

In the other direction, by [14], every c.e., strongly jump-traceable set A is computable from a Demuth random set, and by [3], each such set is also low for $\text{Demuth}_{\text{BLR}}$ randomness, and so in fact computable from a $(\text{Demuth}_{\text{BLR}})^A$-random set, in other words, is a base for $\text{Demuth}_{\text{BLR}}$ randomness. Again this notion is downwards closed in the Turing degrees. □

Our proof of Theorem 1.5 utilises a concept of independent interest, that of a *cost function*. Formalised by Nies (see [23]), cost function constructions generalise the familiar construction of a K-trivial set (see [10]) or the construction of a set low for K (Mučnik; see [8]). Indeed, the key to the coincidence of K-triviality with lowness for K is the fact that K-triviality can be characterised by obedience to a canonical cost function.

In this paper, we define a *cost function* to be a Δ^0_2, non-increasing function from ω to the non-negative real numbers $\mathbb{R}^+$. A cost function c satisfies the *limit condition* if its limit $\lim_x c(x)$ is 0. A *monotone approximation* for a cost function c is a uniformly computable sequence $\langle c_s \rangle$ of functions from ω to the non-negative rational numbers $\mathbb{Q}^+$ such that:

- each function c_s is non-increasing; and
- for each $x < \omega$, the sequence $\langle c_s(x) \rangle_{s<\omega}$ is non-decreasing and converges to $c(x)$.

Here we use the standard topology on $\mathbb{R}$ to define convergence, rather than the discrete topology which is usually used to define convergence of computable approximations of Δ^0_2 sets and functions. A cost function is called *monotone* if it has a monotone approximation. In this paper, *we are only interested in monotone cost functions which satisfy the limit condition*, and so when we write "cost function", unless otherwise mentioned, we mean "monotone cost function satisfying the limit condition".

If $\langle A_s \rangle$ is a computable approximation of a Δ^0_2 set A, then for each $s < \omega$, we let x_s be the least number x such that $A_{s-1}(x) \neq A_s(x)$. If $\langle c_s \rangle$ is a monotone approximation for a cost function c, then we write $\sum c_s(A_s)$ for $\sum c_s(x_s)$. It is understood that if $A_s = A_{s-1}$, then no cost is added at stage s to the sum $\sum c_s(A_s)$.

Definition 1.10. A Δ^0_2 set A *obeys* a cost function c if there is a computable approximation $\langle A_s \rangle$ of A and a monotone approximation $\langle c_s \rangle$ of c such that the sum $\sum_{s<\omega} c_s(A_s)$ is finite.

Nies [25] showed that obedience does not depend on the monotone approximation for c; that is, if A obeys c, then for any monotone approximation $\langle c_s \rangle$ for c, there is a computable approximation $\langle A_s \rangle$ of A for which the sum above is finite. See Proposition 3.2 below. However, different approximations for A may cause the sum to be infinite.

Unlike K-triviality, strong jump-traceability cannot be characterised by a single cost function; one way to see this is by considering the complexity of the index-set of strong jump-traceability, which is Π^0_4-complete (Ng [20]). Greenberg and Nies [13] isolated a class of cost functions which together characterised strong jump-traceability on the c.e. sets. Benignity is an effective witness for the limit condition. It is a generalisation of the additive property of the canonical cost function for K-triviality.

Let $\langle c_s \rangle$ be a monotone approximation for a cost function c, and let $\epsilon > 0$ be rational. We define an auxiliary sequence of markers $m_1(\epsilon), m_2(\epsilon), \ldots$, by letting $m_1(\epsilon) = 0$, and given $m_r(\epsilon)$, letting $m_{r+1}(\epsilon)$ be the least $s > m_r(\epsilon)$ such that $c_s(m_r(\epsilon)) \geqslant \epsilon$, if there is such a stage s; otherwise, $m_{r+1}(\epsilon)$ is undefined. The fact that $\lim c_s = c$ and that $\lim c = 0$ shows that the sequence $\langle m_r(\epsilon) \rangle$ must be finite, and so we can let $r(\epsilon) = r_{\langle c_s \rangle}(\epsilon)$ be the last r such that $m_r(\epsilon)$ is defined.

Definition 1.11. A cost function c is *benign* if it has a monotone approximation $\langle c_s \rangle$ for which the function $\epsilon \mapsto r_{\langle c_s \rangle}(\epsilon)$ is bounded by a computable function.

Note that if $\langle c_s \rangle$ witnesses that c is benign, then the last value $m(\epsilon) = m_{r(\epsilon)}(\epsilon)$ need not be bounded by a computable function; it is ω-computably approximable (ω-c.e.).

Greenberg and Nies showed that a c.e. set is strongly jump-traceable if and only if it obeys all benign cost functions. Much like obeying the canonical cost function captures the dynamics of the decanter and golden run methods which are used for working with K-trivial oracles, this result shows that benign cost functions capture the dynamics of the box-promotion method when applied to c.e., strongly jump-traceable oracles.

Greenberg, Hirschfeldt, and Nies [12] showed that every set, not necessarily c.e., which obeys all benign cost functions, must be strongly jump-traceable. In this paper we show that obeying benign cost functions in fact characterises strong jump-traceability on all sets.

Theorem 1.12. *A set is strongly jump-traceable if and only if it obeys every benign cost function.*

The fact that every K-trivial set is computable from a c.e. one is also deduced using obedience to the canonical cost function. It is easy to see that if a computable approximation $\langle A_s \rangle$ witnesses that A obeys a cost function c, then the associated change-set, which records the changes in this approximation for A, is a c.e. set which computes A and also obeys the cost function c. Hence, Theorem 1.12 almost gives us Theorem 1.5; the connection between benign cost functions and strong jump-traceability established in [13] shows now that if A is a strongly jump-traceable set, and h is an order function, then there is an h-jump-traceable c.e. set which computes A. (We note that this result implies all the corollaries above). We get Theorem 1.5 by showing:

Theorem 1.13. *There is a benign cost function c such that for any Δ^0_2 set A obeying c, there is a c.e. set W computing A, which obeys all cost functions that A obeys.*

Theorem 1.5 is an immediate consequence of the conjunction of Theorems 1.12 and 1.13. We prove Theorem 1.12 in Section 2 and Theorem 1.13 in Section 3.

2. Strongly jump-traceable sets obey benign cost functions

In this section we prove Theorem 1.12. As we mentioned above, one direction of the theorem is proved in [12]. For the other direction, we are given a strongly jump-traceable set A, and a benign cost function c, and show that A obeys c.

2.1. **Discussion.** Our departure point is a simplified version of the original argument showing that every strongly jump-traceable set is Δ^0_2. Suppose that we are given a strongly jump-traceable set A, and we wish to find a computable approximation $\langle A_s \rangle$ for A. The idea is to *test* binary strings, potential initial segments of A. For example, to determine $A(0)$, we try to test both strings $\langle 0 \rangle$ and $\langle 1 \rangle$, and hopefully get an indication which one is an initial segment of A. Our belief about which one is correct may change from time to time, but we need to make sure that it changes only finitely many times, and eventually settles on the correct value.

While we fluctuate between $\langle 0\rangle$ and $\langle 1\rangle$, we also test strings of length 2, and match up our guess for which string of length 2 is an initial segment of A with the current guess about which string of length 1 is an initial segment of A. Again, our belief about strings of length 2 may change several times, indeed many more than the changes between $\langle 0\rangle$ and $\langle 1\rangle$, but eventually it should settle to the correct value.

How do we test strings of a given length? We define a functional Ψ, fix an order function h, which will be designed to grow sufficiently slowly as to enable the combinatorics of the construction, and by the recursion theorem (or by using a universal trace), we have a c.e. trace $\langle T(z)\rangle$ for the partial function Ψ^A, bounded by h. To test, for example, all strings of a length ℓ on some input z, we define $\Psi^\sigma(z) = \sigma$ for every string σ of length ℓ. We then only believe strings which show up in the trace $T(z)$. If $h(z) = 1$ then we are done, since only one string may show up in $T(z)$, and the correct string $A\upharpoonright \ell$ must appear in $T(z)$. However, h must be unbounded, and once we tested a string σ on some input z, we cannot test any extensions of σ on the same input; for the functional Ψ must be kept consistent. What do we do, then, if $h(z) > 1$, and more than one string of length ℓ shows up in $T(z)$?

This is where *length promotion* comes into place. Suppose that initially, we use inputs z such that $h(z) = \ell$ to test strings of length ℓ (such inputs are sometimes called *ℓ-boxes*). So when we test strings of length 3 on 3-boxes, of the eight possibilities, we believe at most three. At first, we believe the first string of length 3 which shows up in the relevant trace component, say $\langle 000\rangle$. If another string shows up, say $\langle 001\rangle$, we move to test the length 3 on 2-boxes which we have reserved for this occasion — we *promote* length 3 to 2-boxes. We then believe the first string of length 3 which shows up in the trace for the 2-boxes, say $\langle 010\rangle$. If another string shows up, say $\langle 011\rangle$, we promote length 3 to 1-boxes. Since trace components for 1-boxes can contain only a single element, we know that the first string of length 3 to show up in this component must be correct. In general, the promotion mechanism ensures that we have an approximation for A for which there are at most ℓ changes in our belief about $A\upharpoonright \ell$. We emphasize again, though, that we have only finitely many n-boxes for each n, and so the main point of the argument is arranging the testing so that only finitely many lengths are ever promoted to be tested on n-boxes, and moreover, that we can compute in advance a bound on the number of such lengths. As explained in greater detail below, this is done by making sure that all the lengths that are ever promoted from level n to level $n-1$ interact in some way in some n-box and "fill it up", so that in fact no more than n many lengths can be promoted beyond level n.

Let $\langle c_s\rangle$ be a monotone approximation for c which witnesses that c is benign; let $m_r(\epsilon)$ be the associated markers. To construct a computable approximation $\langle A_s\rangle$ for A for which the sum $\sum_s c_s(A_s)$ is finite, we need, roughly, to give a procedure for guessing initial segments of A such that for all n, the number of stages at which we change our mind about a guess for $A\upharpoonright m_r(2^{-n})$, for any $r < 2^{-n}$, is bounded by (say) n. The computable bound on $r(2^{-n})$, the number of lengths we need "test at level n", allows us to apportion, in advance, sufficiently many n-boxes to deal with all of these lengths, even though which lengths are being tested at level n is not known in advance. The fact that the lengths themselves are not known in advance necessitates a first step of "winnowing" the strings of new lengths $m_r(2^{-n})$, so that instead of dealing with $2^{m_r(2^{-n})}$ many strings, we are left with at most n such

strings. This is done by testing all strings of the given length on an n-box reserved for this length, as described above.

This technique of length promotion is a close cousin of the older technique of box promotion. If A were a c.e. set, we could use some enumeration of it in combination with the above tracing to determine our guesses for $A\upharpoonright \ell$; we would only believe a string at stage s if that string happened to be $A_s\upharpoonright \ell$. If we later stopped believing that string, it would mean that the approximation to A had changed away from $A_s\upharpoonright \ell$, and so that string is now known to be wrong. Any n-box for which the relevant trace contains this string is effectively now an $(n-1)$-box; this is the fundamental mechanic of box promotion. In our setting, however, we are working with arbitrary sets (we already know, in fact, that they are Δ^0_2, but we will not use this in the construction), and so we cannot know that any given string is wrong. We will see incompatible strings occur in the traces, and so we will know at least one of them is wrong, but we will not know which. The advantage of length promotion is that in this case, we do not need to know which is wrong.

As is the case with all promotion constructions, the heart of the proof is in the precise combinatorics which tell us which strings are tested on which boxes. One main point is that while we need to prepare n-boxes for the possibility that lengths tested at higher levels are promoted all the way down to level n, the number of such promotions must be computably bounded in n, and cannot rely on the computable bound on $r(2^{-(n+1)})$, $r(2^{-(n+2)}),\ldots$. That is, the number of promotions must be tied to the size (or level) of the boxes, and not on the number of lengths that may be tested at that level. We shall achieve this by ensuring that if k lengths are promoted from $n+1$-boxes to n-boxes, then some trace component for an $n+1$-box contains at least $k+1$ elements, and thus $k\leqslant n$.

Consider, for example, the following situation: at some level n, we are testing two lengths, $\ell_1<\ell_2$, and tests have returned positively for strings σ_0 and σ_1 of length ℓ_1, and strings τ_0 and τ_1 of length ℓ_2. If, to take an extreme situation for an example, the strings $\sigma_0,\sigma_1,\tau_0,\tau_1$ are pairwise incomparable, then when apportioning boxes on which to test the strings, we could have arranged that there was a single input z on which they were *all* tested; then z is an n-box with 4 elements in its trace component after 2 lengths have been promoted, as desired. If, on the other hand, τ_0 extends σ_0 and τ_1 extends σ_1, then we cannot test τ_0 on boxes on which we already tested σ_0, and the same holds for τ_1 and σ_1. We do not want, though, to let both lengths be promoted while only putting 2 elements into any given n-box. In this case our action depends on timing:

- If σ_0 and σ_1 appear before τ_0 and τ_1 appear, we promote the length ℓ_1. We do not promote ℓ_2, unless another string of length ℓ_2 appears. If no such new string appears, then our belief about which of σ_0 or σ_1 is an initial segment of A will dictate which of τ_0 or τ_1 we believe too.
- If τ_0 and τ_1 appear before we see both σ_0 and σ_1, then we promote the length ℓ_2. In this case, certainly our belief about which of τ_0 or τ_1 is an initial segment of A would tell us whether to believe σ_0 or σ_1.

In the first case, an important observation is that if another string ρ of length ℓ_2 appears, then ρ cannot extend both σ_0 and σ_1. If ρ does not extend σ_0, say, then we can test σ_0,ρ and τ_1 all on one box, and so the corresponding component will eventually contain 3 elements, justifying the promotion of both lengths ℓ_1 and ℓ_2. Of course, during the construction, we need to test strings on a large number of

boxes, to allow for all possible future combinations of sets of strings involving the ones being tested, including strings of future lengths not yet observed.

2.2. **Construction.** As mentioned above, let $\langle c_s \rangle$ be a monotone approximation for c which witnesses that c is benign; let $m_r(\epsilon)$ be the associated markers. We force these markers to cohere in the following way. For $n < \omega$ and $s < \omega$ let

$$l_s(n) = \max\left(\{n\} \cup \left\{m_r(2^{-q}) \,:\, q \leqslant n \;\&\; r \leqslant s \;\&\; m_r(2^{-q}) \leqslant s\right\}\right).$$

Note that $m_r(\epsilon)$ is strictly increasing in r, and so $m_r(2^{-q}) \leqslant s$ already implies $r \leqslant s$. The value $l_s(n)$ is, roughly, a length at which the cost function c_s becomes small, relative to 2^{-n}. We summarise the properties of the functions l_s in the following lemma.

Lemma 2.1.

(1) *Each function l_s is non-decreasing, with $n \leqslant l_s(n) \leqslant \max\{n, s\}$.*
(2) *For each n, the sequence $\langle l_s(n) \rangle_{s<\omega}$ is non-decreasing, and takes finitely many values. Indeed, the function*

$$n \mapsto \#\{l_s(n) \,:\, s < \omega\}$$

is computably bounded.
(3) *For all n and s, $c_s(l_s(n)) < 2^{-n}$.*

We fix a computable function g bounding the function $n \mapsto \#\{l_s(n) \,:\, s < \omega\}$.

For $n \geqslant 1$, let $\alpha(n) = \binom{n}{0} + \binom{n}{1} + \binom{n}{2}$ be the number of subsets of $\{1, 2, \ldots, n\}$ of size at most 2. We partition ω into intervals M^1,I^1, M^2, I^2,...; the interval M^n has size $\alpha(n)^{n+g(n)}$ and the interval I^n has size $n + g(n)$. We define an order function h by letting $h(x) = n$ for every $x \in M^n \cup I^n$.

As mentioned, we enumerate a functional Ψ. Either by using the recursion theorem (as was done in [6]) or by using a universal trace (as in [13]), we obtain a number $o \in \omega$ and a c.e. trace $T = \langle T(z) \rangle$ for Ψ which is bounded by $\max\{h, o\}$.

Each level $n \geqslant o$ will list an increasing sequence of lengths $\ell^n_1, \ell^n_2, \ldots$ which will be tested at level n. The list is dynamic – we may extend it during the construction. However, we will need to ensure that the length of the list is bounded by $n + g(n)$.

The testing of lengths at level n will be in two parts:

A. *Initial testing* of all strings of length ℓ^n_k will be performed on a reserved input from the interval I^n. We thus enumerate the elements of I^n as $\{y^n_1, y^n_2, \ldots, y^n_{n+g(n)}\}$; the input y^n_k is reserved for initial testing of all strings of length ℓ^n_k. We note here that as the list of lengths $\ell^n_1, \ell^n_2, \ldots$ may not necessarily reach its maximal length $n+g(n)$, it is possible that some inputs y^n_k will never be used. This is one reason for the fact that Ψ^A will be a *partial function*. In this way we use the full hypothesis of strong jump-traceability of A; we cannot hope to make Ψ^A total, and so the proof would not work for merely c.e. traceable oracles.

B. The main bulk of the testing of strings of length ℓ^n_k would be performed on inputs from M^n. To maximise the interaction between the various lengths (to justify promotion as described above, we need to test large antichains of strings on inputs from M^n), we organize M^n in a particular fashion. Let $D(n) = \{1, 2, \ldots, n+g(n)\}$, and let $P(n)$ be the collection of all subsets of $\{1, 2, \ldots, n\}$ of size at most 2. We enumerate the elements of M^n as $z_\nu = z^n_\nu$, where ν ranges over all functions from $D(n)$ to $P(n)$. Note that $|M^n| = |P(n)|^{|D(n)|}$. Geometrically-oriented readers can

envision M^n as a discrete $|D(n)|$-dimensional cube, each of whose sides has length $|P(n)| = \alpha(n)$; the index ν gives the Cartesian coordinates of z_ν in M^n.

The construction begins at stage o. At stage $s \geqslant o$, we act in turn on level s, level $s-1, \dots,$ down to level o. The action at level n consists of: (1) extending the sequence of lengths $\langle \ell^n_k \rangle$; (2) testing strings on the n-cube M^n; and (3) if $n > o$, promoting lengths to be tested on level $n-1$.

Let $n \in [o, \dots, s]$. The action at level n at stage s is as follows:

1. If $n < s$ and some lengths have just been promoted from level $n+1$, we append them to the list of lengths $\ell^n_1, \ell^n_2, \dots$ tested at level n, ordered by magnitude (we will make sure that the promoted lengths are longer than lengths already tested at level n).

If $l_s(n)$ is greater than the lengths currently tested at level n (including the lengths which have just been promoted), we add it too to the list of lengths tested at level n.

We are assuming now that at every stage, the number of lengths tested at level n is at most $n + g(n)$. We will prove this later (Section 2.3).

For each length ℓ^n_k which was added to the list, we test all strings of length ℓ^n_k on y^n_k. This means we define $\Psi^\sigma(y^n_k) = \sigma$ for every string σ of length ℓ^n_k.

2. Suppose that ℓ^n_k is defined by stage s. We list the elements of $T(y^n_k)$ by $\sigma^n_k(1), \sigma^n_k(2), \dots$ as they appear. Because $n \geqslant o$ and $y^n_k \in I^n$, we have $|T(y^n_k)| \leqslant n$, so the list has length at most n.

Suppose that $\sigma^n_k(i)$ has appeared in $T(y^n_k)$. Recall that $P(n)$ is the collection of subsets of $\{1, 2, \dots, n\}$ of size at most 2. For every $\nu\colon D(n) \to P(n)$ such that $i \in \nu(k)$, we want to test $\sigma^n_k(i)$ on $z_\nu = z^n_\nu$. Fix such ν. We need to ensure that Ψ remains consistent; the point is that there may be strings comparable with $\sigma = \sigma^n_k(i)$ which are already tested on z_ν. To test σ on z_ν while keeping Ψ consistent, we define $\Psi^\tau(z_\nu) = \tau$ for every extension τ of σ of length s which does not extend any string already tested on z_ν.

Using other notation, we let $Z_{\nu,s}$ be the collection of strings ρ for which we defined $\Psi^\rho(z_\nu) = \rho$ by the end of stage s, and let $\mathcal{Z}_{\nu,s} = [Z_{\nu,s}]$ be the clopen subset of Cantor space 2^ω determined by the set of strings $Z_{\nu,s}$ (the collection of all infinite extensions of strings in $Z_{\nu,s}$). Testing a string σ on z_ν at stage s means adding strings of length s to $Z_{\nu,s-1}$ so as to keep $Z_{\nu,s}$ an antichain, but ensuring that $[\sigma] \subseteq \mathcal{Z}_{\nu,s}$.

3. For $\nu\colon D(n) \to P(n)$, we may assume that $T_s(z_\nu) \subseteq Z_{\nu,s}$. (Otherwise, simply ignore all other values, acting as though $T_s(z_\nu)$ were replaced by $T_s(z_\nu) \cap Z_{\nu,s}$.) We let $\mathcal{T}_s(z_\nu) = [T_s(z_\nu)]$ be the clopen subset of Cantor space determined by $T_s(z_\nu)$.

Let $k \leqslant n + g(n)$ such that ℓ^n_k is defined by stage s, and let $i \leqslant n$ such that $\sigma^n_k(i)$ is defined by stage s, that is, $T(y^n_k)$ already contains at least i many elements by stage s. The test of $\sigma^n_k(i)$ is *successful* if for all ν such that $i \in \nu(k)$, that is, for all ν such that σ was tested on z_ν, we have $[\sigma] \cap \mathcal{T}_s(z_\nu) \neq \emptyset$. In other words, if some string which is comparable with σ appears in $T(z_\nu)$ by stage s.

For the purpose of the following definition, let $\ell^n_0 = 0$. We say there is a *conflict* at length ℓ^n_k (and level n) if there are two strings $\sigma_0 = \sigma^n_k(i)$ and $\sigma_1 = \sigma^n_k(j)$ of length ℓ^n_k, both of whose tests are successful by stage s, such that $\sigma_0 \restriction \ell^n_{k-1} = \sigma_1 \restriction \ell^n_{k-1}$. We note, for future reference, that if there is a conflict at length ℓ^n_k at stage s, then this conflict persists at every later stage.

At stage s, if $n > o$, then we promote to level $n-1$ all lengths ℓ_k^n for which there is a conflict at stage s, and which are longer than any length already tested at level $n-1$.

These instructions determine our action for level n at stage s, and so completely describe the construction.

2.3. **Justification.** Before we show how the construction gives us the desired approximation for A, we first need to show that we can actually implement the construction. We need to prove that we have allocated sufficiently many n-boxes to each level n. Since the tests run on M^n are winnowed through I^n, it is immediate that we have allocated sufficiently many n-boxes to M^n; it remains only to show that we have allocated sufficiently many n-boxes to I^n. That is, we must show that the list of lengths $\langle \ell_k^n \rangle$ tested at level n has length at most $n + g(n)$.

For $n \geqslant o$ and $s < \omega$, let $k_s(n)$ be the number of lengths tested at level n by the end of stage s. That is, at the end of stage s, the lengths ℓ_k^n are defined for $k \leqslant k_s(n)$. We need to show that for all s, $k_s(n) \leqslant n + g(n)$.

There are two streams contributing lengths to test at level n: lengths promoted from level $n+1$, and lengths of the form $l_s(n)$. Of the latter, there are at most $g(n)$ many. Hence, it remains to show that there are at most n many lengths that are promoted by level $n+1$. Shifting indices, we show that level n promotes at most $n-1$ many lengths.

Indeed, we show the following:

Lemma 2.2. *Let $n \geqslant o$ and let $s \geqslant o$. Then there are at most $n-1$ many lengths ℓ_k^n at which there is a conflict (for level n) at stage s.*

To prove Lemma 2.2, fix $n \geqslant o$ and $s \geqslant o$. Let N be the number of lengths at which there is a conflict (at level n) at the end of stage s. We show that there is some $\nu \colon D(n) \to P(n)$ such that $|T_s(z_\nu)| - 1 \geqslant N$. Using the fact that $n \geqslant o$ and $z_\nu \in M^n$ we see that $|T_s(z_\nu)| \leqslant n$, which establishes the desired bound.

In order to define ν, we define an increasing sequence of antichains of strings, indexed in reverse $C_{k_s(n)+1} \subseteq C_{k_s(n)} \subseteq C_{k_s(n)-1} \subseteq \cdots \subseteq C_1$, starting with $C_{k_s(n)+1} = \emptyset$. Each set C_k consists of strings of lengths $\ell_{k'}^n$ for $k' \geqslant k$. Let $k \in \{1, \ldots, k_s(n)\}$; we assume that C_{k+1} has been defined, and we show how to define C_k.

The definition is split into two cases. First, suppose that there is no conflict at stage s in length ℓ_k^n. In this case, we let $C_k = C_{k+1}$; it will follow from the definition below that $\nu(k) = \emptyset$.

We assume then that there is a conflict in length ℓ_k^n at stage s. Let $\sigma_0 = \sigma_k^n(i)$ and $\sigma_1 = \sigma_k^n(j)$ be a pair witnessing this conflict. We let C_k be a maximal antichain from $C_{k+1} \cup \{\sigma_0, \sigma_1\}$ containing C_{k+1}. In other words, if neither σ_0 nor σ_1 are comparable with any string in C_{k+1}, then we let $C_k = C_{k+1} \cup \{\sigma_0, \sigma_1\}$; otherwise, if either σ_0 or σ_1 is incomparable with all the strings in C_{k+1}, then we let C_k be one of $C_{k+1} \cup \{\sigma_0\}$ or $C_{k+1} \cup \{\sigma_1\}$, making sure that we choose so that C_k is an antichain; and finally, if both σ_0 and σ_1 are comparable with strings in C_{k+1}, then we let $C_k = C_{k+1}$.

Now given the sequence of sets C_k, we can define the index function ν:

- For $k \in \{1, 2, \ldots, k_s(n)\}$, we let
$$\nu(k) = \{i \leqslant n \,:\, \sigma_k^n(i) \in C_k\}\,.$$
- For $k \in \{k_s(n)+1, \ldots, n + g(n)\}$, we let $\nu(k) = \emptyset$.

Since the strings in C_k of length ℓ_k^n are precisely the strings in $C_k \setminus C_{k+1}$, we see that for all k, $\nu(k)$ is indeed a set of size at most 2, so ν is a function from $D(n)$ to $P(n)$. The point of this definition is that the strings tested on z_ν are precisely the strings in C_1.

Letting $\ell_0^n = 0$ again, for $k \in \{1, \dots, k_s(n) + 1\}$, let

$$D_k = \left\{\sigma \restriction \ell_{k-1}^n : \sigma \in C_k\right\},$$

and let

$$p_k = |C_k| - |D_k|.$$

Note that $p_{k_s(n)+1} = 0$, and that unless C_1 is empty, $|C_1| = p_1 + 1$.

Claim 2.3. *For all $k \leqslant k_s(n)$, $p_k \geqslant p_{k+1}$.*

Proof. For every string τ in D_k which has no extension in D_{k+1}, there is an extension σ of τ in $C_k \setminus C_{k+1}$. Therefore,

$$p_k - p_{k+1} = |C_k| - |C_{k+1}| + |D_{k+1}| - |D_k| \geq 0. \qquad \square$$

Claim 2.4. *If ℓ_k^n has a conflict at stage s, then $p_k > p_{k+1}$.*

Proof. Let σ_0 and σ_1 be the strings that were chosen at step k to witness that ℓ_k^n has a conflict at stage s. By definition of having a conflict, $\sigma_0 \restriction \ell_{k-1}^n = \sigma_1 \restriction \ell_{k-1}^n$; we let τ denote this string.

There are three cases. In all three cases, we note that every string in D_k other than possibly τ has an extension in D_{k+1}.

If $C_k = C_{k+1} \cup \{\sigma_0, \sigma_1\}$ then we need to show that $|D_k| \leqslant |D_{k+1}| + 1$, which follows from the fact we just mentioned, that every string in D_k other than τ has an extension in D_{k+1}.

In the second case, we assume that C_k is obtained from C_{k+1} by adding one string, say σ_0; we need to show that $|D_k| \leqslant |D_{k+1}|$. But σ_1 is comparable with some string in C_{k+1}, and in fact must be extended by some string in C_{k+1}. Hence $\sigma_1 \in D_{k+1}$, i.e. τ is extended by some string in D_{k+1}, and therefore every string in D_k is extended by some string in D_{k+1}.

Finally, suppose that $C_{k+1} = C_k$; we need to show that $|D_{k+1}| \leqslant |D_k| - 1$. Since both σ_0 and σ_1 are comparable with elements of C_k, both are elements of D_{k+1}, and so τ has two extensions in D_{k+1}, while every other string in D_k has an extension in D_{k+1}. $\square$

Hence $p_1 \geqslant N$. If C_1 is empty, then $N = 0$, so we may assume that C_1 is non-empty, and so $|C_1| = p_1 + 1$ is at least one more than N. Then Lemma 2.2, and with it our justification for the construction, is completed once we establish the following claim.

Claim 2.5. $|T_s(z_\nu)| \geqslant |C_1|$.

Proof. We show that $T_s(z_\nu)$ contains only strings which are extensions of strings in C_1, and that each string in C_1 has an extension in $T_s(z_\nu)$.

Recall that we let $Z_{\nu,s}$ be the collection of strings that were actually tested on z_ν by stage s, that, is, the collection of strings ρ for which we defined $\Psi^\rho(z_\nu) = \rho$ by the end of stage s.

Our instructions (and the definition of ν) say that the strings tested on z_ν are precisely the strings in C_1. Since C_1 is an antichain, this means that at any stage t before some string σ is tested on z_ν, we have $[\sigma] \cap \mathcal{Z}_{\nu,t} = \emptyset$, and so when testing σ,

we only add extensions of σ to $Z_{\nu,s}$. Since we assumed that $T_s(z_\nu) \subseteq Z_{\nu,s}$, we see that all strings in $T_s(z_\nu)$ are extensions of strings in C_1.

Let $\sigma \in C_1$. Then $\sigma = \sigma^n_k(i)$ for some k and i is part of a pair of strings witnessing that there is a conflict at length ℓ^n_k (and level n) by stage s. So the test of σ on M^n is successful by the end of stage s. Since σ is tested on z_ν, we have $[\sigma] \cap \mathcal{T}_s(z_\nu) \neq \emptyset$. Since no proper initial segment of σ is tested on z_ν, this means that some extension of σ is an element of $T_s(z_\nu)$. □

2.4. **The approximation of A.** We now show how to find a computable approximation for A witnessing that A obeys c.

For $n \geqslant o$, let $k(n) = \lim_s k_s(n)$ be the number of lengths ever tested at level n.

Lemma 2.6. *For all $n \geqslant o$ and all $k \leqslant k(n)$, the string $A \upharpoonright \ell^n_k$ is eventually successfully tested at level n.*

Proof. Let s_0 be the stage at which the length ℓ^n_k is first tested at level n. Let $\rho = A \upharpoonright \ell^n_k$. At stage s_0, we define $\Psi^\rho(y^n_k) = \rho$, and so $\Psi^A(y^n_k) = \rho$. Since T traces Ψ^A, we have $\rho \in T(y^n_k)$; this is discovered by some stage $s_1 > s_0$. At stage s_1 we test ρ on elements z_ν of M^n. Fix such an input z_ν. We need to show that $[\rho] \cap \mathcal{T}(z_\nu)$ is non-empty.

At stage s_1, we enumerate strings into Z_ν to ensure that $[\rho] \subseteq \mathcal{Z}_\nu$. Hence $A \in \mathcal{Z}_\nu$, in other words, $z_\nu \in \operatorname{dom} \Psi^A$. Since T traces Ψ^A, we have $\Psi^A(z_\nu) \in T(z_\nu)$. All axioms of Ψ are of the form $\Psi^\tau(z) = \tau$ for binary strings τ, so $\tau = \Psi^A(z_\nu)$ is an initial segment of A, and so is comparable with ρ. Then $[\tau] \subseteq \mathcal{T}(z_\nu)$ implies that $[\rho] \cap \mathcal{T}(z_\nu) \neq \emptyset$. □

For $n \in [o, \dots, s]$, let $\ell^n[s] = \ell^n_{k_s(n)}$ be the longest length tested at level n at the end of stage s. Then for all $s \geqslant o$, $\ell^o[s] \leqslant \ell^{o+1}[s] \leqslant \cdots \leqslant \ell^s[s] = s$, because if we let $\ell^n[s] = l_s(n)$ at stage s, then (Lemma 2.1) $l_s(n+1) \geqslant l_s(n)$ and so we define $\ell^{n+1}[s] = l_s(n+1)$ if this length is longer than previous lengths tested at level $n+1$. Also, since at stage s we test $s = l_s(s)$ at level s, we see that for all $s \geqslant n$, $\ell^n[s] \geqslant n$.

For $n \geqslant o$, we let $\ell^n = \ell^n_{k(n)} = \lim_s \ell^n[s]$ be the longest length ever tested at level n. Let $\rho^* = A \upharpoonright \ell^o$. Let $s_o > o$ be a stage sufficiently late so that $\ell^o[s_o] = \ell^o$ and the string ρ^* is successfully tested at level o by stage s_o.

We note that other than specifying ρ^*, the construction is uniform (in the computable index for $\langle c_s \rangle$). The reason for the non-uniform aspect of the construction is the overhead o charged by the recursion theorem; if we had access to 1-boxes, the construction would be completely uniform.

Let $s \geqslant s_o$ and $n \geqslant o$. A string σ of length $\ell^n[s]$ is *n-believable* at stage s if:

- σ extends ρ^*; and
- for all $m \in [o, n]$, and for all $k \leqslant k_s(m)$, the string $\sigma \upharpoonright \ell^m_k$ is successfully tested at level m by the end of stage s.

Lemma 2.6 shows that for all n, the string $A \upharpoonright \ell^n$ is n-believable at almost every stage.

Claim 2.7. *Let $s \geqslant s_o$. For every $n \geqslant o$, there is at most one string which is n-believable at stage s.*

Proof. By induction on n. For $n = o$ this is clear, because ρ^* has length $\ell^o[s]$ for all $s \geqslant s_o$.

Let $n > o$, and suppose that there is at most one string which is $(n-1)$-believable at stage s. Suppose, for contradiction, that there are two strings τ_0 and τ_1 which are both n-believable at stage s. Then both $\tau_0 \upharpoonright \ell^{n-1}[s]$ and $\tau_1 \upharpoonright \ell^{n-1}[s]$ are $(n-1)$-believable at stage s, and so are equal. Let k be the least index such that $\tau_0 \upharpoonright \ell^n_k \neq \tau_1 \upharpoonright \ell^n_k$. Of course k exists, since $\tau_0 \neq \tau_1$ are both of length $\ell^n_{k_s(n)}$, and $\ell^n_k > \ell^{n-1}[s]$. In other words, ℓ^n_k is longer than any length tested at level $n-1$ at stage s. But then the strings $\tau_0 \upharpoonright \ell^n_k$ and $\tau_1 \upharpoonright \ell^n_k$ witness that there is a conflict at length ℓ^n_k at stage s, and so we would promote ℓ^n_k to be tested at level $n-1$ by the end stage s, contradicting the assumption that ℓ^n_k is not tested at level $n-1$ at stage s. □

We can now define the computable approximation for A. We define a computable sequence of *stages*: the stage s_o has been defined above; we may assume that $s_o \geqslant o+1$. For $t > o$, given s_{t-1}, we define s_t to be the least stage $s > s_{t-1}$ at which there is a t-believable string σ_t. So $s_{t-1} \geqslant t$. We let $A_t = \sigma_t \hat{} 0^\omega$. The fact that $A \upharpoonright \ell^n$ is n-believable at almost every stage (and that $\ell^n \geqslant n$) implies that $\lim_t A_t = A$.

For $t \geqslant o$, let x_t be the least number x such that $A_t(x) \neq A_{t-1}(x)$. It remains to show that $\sum_{t>o} c_t(x_t)$ is finite. For all $n \geqslant 0$, let

$$S_n = \left\{ t > o : c_t(x_t) \geqslant 2^{-n} \right\}.$$

Then $\sum c_t(x_t) < \infty$ will follow from any polynomial bound on $|S_n|$. Let $n > o$, and let $t \in S_n$. Let $s = s_{t-1}$, and let $\overline{s} = s_t$. Since $t \leqslant s$ and $\langle c_s \rangle$ is monotone, we have $c_s(x_t) \geqslant c_t(x_t) \geqslant 2^{-n}$. Since $c_s(l_s(n)) < 2^{-n}$ (Lemma 2.1), and the function c_s is monotone, we have $x_t < l_s(n)$. So $A_t \upharpoonright l_s(n) \neq A_{t-1} \upharpoonright l_s(n)$.

Suppose that $t > n$. Then the strings σ_t and σ_{t-1} are at least as long as $\ell^{t-1}[s]$ which is not smaller than $\ell^n[s]$, which in turn is not smaller than $l_s(n)$, by the instruction for testing $l_s(n)$ at level n at stage s if it is a large number. So we actually have $\sigma_t \upharpoonright \ell^n[s] \neq \sigma_{t-1} \upharpoonright \ell^n[s]$.

Let $m \leqslant n$ be the least such that $\sigma_t \upharpoonright \ell^m[s] \neq \sigma_{t-1} \upharpoonright \ell^m[s]$; since both σ_t and σ_{t-1} extend ρ^* we have $m > o$. Let $k \leqslant k_s(n)$ be the least such that $\sigma_t \upharpoonright \ell^m_k \neq \sigma_{t-1} \upharpoonright \ell^m_k$; the minimality of m implies that $\ell^m_k > \ell^{m-1}[s]$.

Let $\tau_0 = \sigma_{t-1} \upharpoonright \ell^m_k$ and $\tau_1 = \sigma_t \upharpoonright \ell^m_k$. So τ_0 and τ_1 are distinct. Since σ_{t-1} is $(t-1)$-believable at stage s, and $m \leqslant n \leqslant t-1$, the string τ_0 is successfully tested at level m by stage s, and similarly, τ_1 is successfully tested at level m by stage $\overline{s}$. Thus there is a conflict at length ℓ^m_k at stage $\overline{s}$, which implies that $\ell^{m-1}[\overline{s}] \geqslant \ell^m_k$. We observed that $\ell^m_k > \ell^{m-1}[s]$, and so there is no conflict at level ℓ^m_k at stage s.

This means that if t and u are two stages in S_n, and $u > t > n$, then there is some $m \leqslant n$ and some length $\ell = \ell^m_k$ at which there is no conflict at stage s_{t-1} but there is a conflict at stage $s_t \leqslant s_{u-1}$. Lemma 2.2 states this can happen, for each m, at most $m-1$ times, and so overall, there are at most $\binom{n}{2}$ many stages greater than n in S_n; that is, $|S_n| \leqslant n + \binom{n}{2}$. This gives a polynomial bound on $|S_n|$ and completes the proof. □

3. A c.e. set computing a given set

In this section we give a proof of Theorem 1.13: we construct a benign cost function c such that for any Δ^0_2 set A obeying c, there is a c.e. set W computing A which obeys all cost functions that A obeys.

3.1. **A simplification.** Even though the cost function c works for any Δ^0_2 set A, we may assume that we are given a particular computable approximation $\langle A_s \rangle$ to a Δ^0_2 set A which obeys c, and define c using the approximation.

To see why this seemingly circular construction is in fact legal, we enumerate as $\left\langle \left\langle A^k_s \right\rangle_{s<\omega} \right\rangle_{k<\omega}$ all partial sequences of uniformly computable functions; we think of $\left\langle A^k_s \right\rangle_{s<\omega}$ as the k^{th} potential computable approximation for a Δ^0_2 set A^k.

For each $k < \omega$, we define a benign cost function c^k, together with a monotone approximation $\langle c^k_s \rangle$ for c^k and a computable function g^k which together witness that c^k is benign; all of these, uniformly in k. The important dictum is: *even if* $\langle A^k_s \rangle$ *is not total, we must make* $\langle c^k_s \rangle$ *and* g^k *total.* We ensure that $c^k(x) \leqslant 1$ for all x and k.

Once these are constructed, we let $c = \sum_{k<\omega} 2^{-k} c^k$.

Lemma 3.1. *c is a benign cost function.*

Proof. For $s < \omega$ let $c_s(x) = \sum_{k<s} 2^{-k} c^k_s(x)$. Then $\langle c_s \rangle$ is a monotone approximation of c. For benignity, the point is that since $c^k \leqslant 1$, only finitely many c^k can contribute more that ϵ to c. We note that for all k, if $\mathcal{I}$ is a set of disjoint intervals of ω such that for all $[x, s) \in \mathcal{I}$ we have $c^k_s(x) \geqslant \epsilon$, then $|\mathcal{I}| \leqslant g^k(\epsilon)$. Fix $\epsilon > 0$, and let $m_1(\epsilon), m_2(\epsilon), \cdots, m_{r(\epsilon)}(\epsilon)$ be the sequence of markers associated with $\langle c_s \rangle$. Let $\mathcal{I}$ be the set of intervals $[m_i(\epsilon), m_{i+1}(\epsilon))$ for $i < r(\epsilon)$. For all $[x, s) \in \mathcal{I}$ we have

$$\epsilon \leqslant c_s(x) = \sum_{k<s} 2^{-k} c^k_s(x).$$

Let $K = -\log_2(\epsilon) + 1$. Since $c^k_s(x) \leqslant 1$ for all k, we have

$$\sum_{k>K} 2^{-k} c^k_s(x) \leqslant \frac{\epsilon}{2}.$$

It follows that for some $k \leqslant K$ we have $c^k_s(x) \geqslant \epsilon/4$. Hence

$$r(\epsilon) \leqslant \sum_{k \leqslant K} g^k\left(\frac{\epsilon}{4}\right)$$

which is a computable bound on $r(\epsilon)$. □

Suppose that a Δ^0_2 set A obeys c. By Nies's result from [25], which is repeated as Proposition 3.2 below, there is a computable approximation $\langle A_s \rangle$ for A such that $\sum c_s(A_s) \leqslant 1$. This means that for all k, $\sum c^k_s(A_s) \leqslant 2^k$; in particular for k such that $\langle A_s \rangle = \langle A^k_s \rangle$. Thus, it suffices to construct $\langle c^k_s \rangle$ and g^k, uniformly in k, such that if $\langle A^k_s \rangle$ is indeed a Δ^0_2 approximation for a set A, and $\sum c^k_s(A^k_s) \leqslant 2^k$, then there is some c.e. set W computing A which obeys all cost functions that A obeys. The construction for each k is independent.

In the remainder of the construction, we omit the index k; we assume that we are given a partial sequence $\langle A_s \rangle$ and construct total $\langle c_s \rangle$ and g with the desired property. Although we have to make $\langle c_s \rangle$ and g total and c_s bounded by 1, regardless of the partiality of $\langle A_s \rangle$, we note that unless $\langle A_s \rangle$ is total and is a computable approximation for a set A, and $\sum c_s(A_s) \leqslant 2^k$, the construction of W need not be total.

3.2. **More on cost functions.** Given the approximation $\langle A_s \rangle$ for A, we need to test whether A obeys a given cost function d, with a given approximation $\langle d_s \rangle$. But of course it is possible that $\sum d_s(A_s)$ is infinite, while some other approximation for A witnesses that A obeys d. Any other approximation can be compared with the given approximation $\langle A_s \rangle$, and so it suffices to examine a speed-up of the given approximation.

Further, it suffices to test cost functions bounded by 1. This is all ensured by the following proposition. A version of this proposition appears in [25], but as we give it in slightly different form, we include a proof for completeness.

Proposition 3.2. *Let B be a Δ^0_2 set which obeys a cost function d. For any monotone approximation $\langle d_s \rangle$ of d, there is a computable approximation $\langle B_s \rangle$ of B such that $\sum d_s(B_s)$ is finite.*

Moreover, if $\langle B_s \rangle$ is a given computable approximation of B, then there is an increasing computable function h such that $\sum d_s(B_{h(s)}) \leqslant 1$.

Proof. Fix a computable approximation $\langle B_s \rangle$. It is sufficient to find a computable function h such that $\sum_s d_s(B_{h(s)})$ is finite. Then, we can decrease the sum by any finite amount by omitting finitely many initial stages and using monotony of $\langle d_s \rangle$: let y_s be the least x such that $B_{h(s+1)}(x) \neq B_{h(s)}(x)$, so $\sum_s d_s(B_{h(s)}) = \sum_s d_s(y_s)$. Let $r < \omega$ such that $\sum_{s \geqslant r} d_s(y_s) \leqslant 1$. Let $\overline{h}(s) = h(s + r)$; then $\sum_s d_s(B_{\overline{h}(s)}) = \sum_s d_s(y_{s+r})$ which is bounded by $\sum_s d_{s+r}(y_{s+r}) \leqslant 1$ by the monotony of $\langle d_s \rangle$.

Let $\langle e_s \rangle$ be a monotone approximation of d and $\langle \widehat{B}_s \rangle$ be a computable approximation of B such that $\sum e_s(\widehat{B}_s)$ is finite. We define increasing sequences $\langle t_s \rangle$ of stages and $\langle x_s \rangle$ of numbers (lengths) as follows. We let $t_{-1} = x_{-1} = 1$. For $s \geqslant 0$, given t_{s-1} and x_{s-1}, we search for a pair (t, x) such that $t > t_{s-1}$, $x > x_{s-1}$ and

- $d_t(x) < 2^{-(s+1)}$;
- $B_t \upharpoonright x = \widehat{B}_t \upharpoonright x$; and
- For all $y < x_{s-1}$, $2e_t(y) \geqslant d_t(y)$.

Such a pair (t, x) exists because $\lim e_s = \lim d_s$, $\lim \widehat{B}_s = \lim B_s$, and $\lim_x d(x) = 0$ (the limit condition for d). We let (t_s, x_s) be the least such pair that we find. We note that for all $s \geqslant 0$, $s < t_{s-1}$. We now let $h(s) = t_{s+1}$ for all $s \geqslant 0$.

We claim that

$$\sum d_s(B_{h(s)}) \leqslant 2 \sum e_s(\widehat{B}_s) + \sum_s 2^{-s},$$

which is finite. For let $s > 0$, and let y_s be the least number y such that $B_{h(s)}(y) \neq B_{h(s-1)}(y)$; so $\sum d_s(B_{h(s)}) = \sum d_s(y_s)$. There are two cases.

If $y_s \geqslant x_{s-1}$, then $d_{t_{s-1}}(y_s) < 2^{-s}$, and by monotonicity, since $t_{s-1} \geqslant s$, we have $d_s(y_s) < 2^{-s}$.

In the second case, we have $y_s < x_s, x_{s+1}$, and so $B_{h(s-1)}(y_s) = B_{t_s}(y_s) = \widehat{B}_{t_s}(y_s)$ and $B_{h(s)}(y_s) = B_{t_{s+1}}(y_s) = \widehat{B}_{t_{s+1}}(y_s)$. Hence $\widehat{B}_{t_{s+1}}(y_s) \neq \widehat{B}_{t_s}(y_s)$. There is some stage $t \in (t_s, t_{s+1}]$ such that $\widehat{B}_t(y_s) \neq \widehat{B}_{t-1}(y_s)$. Let z be the least number such that $\widehat{B}_t(z) \neq \widehat{B}_{t-1}(z)$. Since $s \leqslant t_s$, we have $d_s(y_s) \leqslant d_{t_s}(y_s)$. Since $y_s < x_{s-1}$, we have $d_{t_s}(y_s) \leqslant 2e_{t_s}(y_s)$. Again by monotonicity, we have $e_{t_s}(y_s) \leqslant e_t(y_s)$. And since $z \leqslant y_s$, we have $e_t(y_s) \leqslant e_t(z)$. Overall, we get

$$d_s(y_s) \leqslant 2e_t(z),$$

and $e_t(z)$ is a summand in $\sum e_s(\widehat{B}_s)$, which is counted only against s, as $h(s-1) < t < h(s)$. □

To ensure that the set W obeys every cost function that A obeys, we need to monitor all possible cost functions. So we need to list them: we need to show that they are uniformly Δ^0_2, indeed with uniformly computable monotone approximations. This cannot be done effectively, because the limit condition cannot be determined in a Δ^0_2 fashion. However, we will not need the limit condition during the construction, only during the verification, and so we list monotone cost functions which possibly fail the limit condition.

Lemma 3.3. *There is a list $\langle d^e\rangle_{e<\omega}$ of all monotone cost functions (which possibly fail the limit condition) bounded by 1, such that from an index e we can effectively obtain a monotone approximation $\langle d^e_s\rangle_{s<\omega}$ for d^e. We may assume that $d^e_s \leqslant 1$ for all e and s, and that for all e, s and $x \geqslant s$ we have $d^e_s(x) = 0$.*

Proof. The idea is delaying. In this proof we do not assume that cost functions satisfy the limit condition, but we do assume that they are total. We need to show that given a partial uniformly computable sequence $\langle d_s\rangle$ we can produce, uniformly, a total monotone approximation $\langle \widehat{d}_s\rangle$ of a cost function $\widehat{d}$ such that if $\langle d_s\rangle$ is a monotone approximation of a cost function d bounded by 1, then $\widehat{d} = d$. To do this, while keeping monotonicity, for every $s < \omega$ we let $t(s) \leqslant s$ be the greatest $t \leqslant s$ such that after calculating for s steps, we see $d_u(x)$ converge for all pairs (u, x) such that $u \leqslant t$ and $x \leqslant t$, each value $d_u(x)$ is bounded by 1, and the array $\langle d_u(x)\rangle_{u,x\leqslant t}$ is monotone (non-increasing in x and non-decreasing in u). We let $\widehat{d}_s(x) = d_{t(s)}(x)$ for all $x \leqslant t(s)$, and $\widehat{d}_s(x) = 0$ for all $x > t(s)$. □

3.3. **Discussion.** Returning to our construction, recall that we are given a partial approximation $\langle A_s\rangle$ and a constant k, and need to produce a (total) monotone approximation $\langle c_s\rangle$ of a cost function c and a computable function g witnessing that c is benign; and we need to ensure that if $\langle A_s\rangle$ is a total approximation of a Δ^0_2 set A and $\sum c_s(A_s) \leqslant 2^k$, then there is a c.e. set W computing A which obeys every cost function that B obeys.

The main tool we use is that of a *change set*. For any computable approximation $\langle B_s\rangle$ of a Δ^0_2 set B, the associated *change* set $W(\langle B_s\rangle)$ consists of the pairs (x, n) such that there are at least n many stages s such that $B_{s+1}(x) \neq B_s(x)$. The obvious enumeration $\langle W_s\rangle$ of W enumerates a pair (x, n) into W_s if there are at least n many stages $t < s$ such that $B_{t+1}(x) \neq B_t(x)$. It is immediate that the change set is c.e. and computes B. It is also not hard to show that for any monotone approximation $\langle d_s\rangle$ for a cost function we have

$$\sum d_s(W_s) \leqslant \sum d_s(B_s),$$

and so if $\langle B_s\rangle$ witnesses that B obeys $d = \lim d_s$, then $\langle W_s\rangle$ witnesses that W obeys d as well. Nies used this argument to show that every K-trivial set is computable from a c.e. K-trivial set.

Thus if $A = \lim A_s$ (if it exists) obeys some cost function d, we immediately get a c.e. set computing A which also obeys d. The difficulty arises when we consider more than one cost function. The point is that different cost functions obeyed by B would require *faster* enumerations of B, and the associated change sets may have distinct Turing degrees. In general, it is not the case that the change set for a

given enumeration of a Δ^0_2 set B would obey all cost functions obeyed by B. For an extreme example, it is not difficult to devise a computable approximation for the empty set for which the associated change set is Turing complete. The point is that a faster approximation of a Δ^0_2 set may undo changes to some input $B(x)$, whereas the change set for the original approximation must record the change to $B(x)$ (and also its undoing), and must pay costs associated with such recordings.

The idea of our construction is to let W be the change set of some speed-up of the approximation $\langle A_s \rangle$. We define an increasing partial computable function f. If $\langle A_s \rangle$ is total, approximates A, and $\sum c_s(A_s) \leqslant 2^k$, then f will be total, and we will let W be the change set of the approximation $\langle A_{f(s)} \rangle$. Roughly, the role of f would be to ensure that not too many undone changes in some $A(x)$ would be recorded by W and associated costs paid. To be more precise, we discuss our requirements in detail.

Let $\langle d^i \rangle_{i<\omega}$ be a list of cost functions (possibly failing the limit condition) bounded by 1, as given by Lemma 3.3 (with associated approximations $\langle d^i_s \rangle$), and let $\langle h^j \rangle_{j<\omega}$ be an effective list of all partial computable functions whose domain is an initial segment of ω and which are strictly increasing on their domain. To save indices, we renumber the list of pairs $\langle d^i, h^j \rangle_{i,j<\omega}$ as $\langle d^e, h^e \rangle_{e<\omega}$.

Let $e < \omega$. The requirement S^e states that if h^e is total and $\sum d^e_s(A_{(f \circ h^e)(s)}) \leqslant 1$, then there is some total increasing computable function q^e such that $\sum d^e_s(W_{q^e(s)})$ is finite.

First, we explain why meeting the requirements is sufficient. Let d be a cost function (with the limit condition) obeyed by A. Let M be a positive rational bound on d, and let $\widehat{d} = d/M$. Since summation is linear, A obeys $\widehat{d}$. Let i be such that $\widehat{d} = d^i$. As f is total, the sequence $\langle A_{f(s)} \rangle$ is a computable approximation of A. By Proposition 3.2, there is an increasing computable function h such that $\sum d^i_s(A_{(f \circ h)(s)}) \leqslant 1$. There is an index e such that $\langle d^e_s \rangle = \langle d^i_s \rangle$ and $h^e = h$. Then requirement S^e ensures that W obeys $\widehat{d}$. By linearity again, W obeys d as well.

We now discuss how to use the cost function c to help meet a requirement S^e. Suppose, for now, that f is the identity function, and that $q^e = h^e$. Let $u = h^e(t)$ be a stage in range h^e. Let z be the least such that $A_{u+1}(z) \neq A_u(z)$. Then we have to enumerate a pair (z, i) into W_{u+1}. This, in turn would mean that $\sum d^e_s(W_{q^e(s)})$ will increase by at most $d^e_t(z)$. To keep $\sum d^e_s(W_{q^e(s)})$ bounded, we need to *charge* this cost to some account. There are two possible accounts: the sum $\sum c_s(A_s)$ and the sum $\sum d^e_s(A_{h^e(s)})$.

Ideally, we define $c_{u+1}(z) \geqslant d^e_t(z)$. Let $v = h^e(t+1)$ (which we may assume is greater than $u+1$). There are two possibilities:

- If $A_v(z) \neq A_u(z)$ (for example, if $A(z)$ does not change back between stages $u+1$ and v), then a cost of $d^e_{t+1}(z) \geqslant d^e_t(z)$ is added to the sum $\sum d^e_s(A_{h^e(s)})$.
- If $A_v(z) = A_u(z)$ then $A_v(z) \neq A_{u+1}(z)$ and so a cost of at least $c_{u+1}(z)$ is added to the sum $\sum c_s(A_s)$. Since we defined $c_{u+1}(z) \geqslant d^e_t(z)$, again a cost at least as large as that facing $\sum d^e_s(W_{q^e(s)})$ is borne by $\sum c_s(A_s)$.

It is important to note that our action at stage $u+1$ to assuage requirement S^e does not require us to wait until we see $v = h^e(t+1)$; it allows us to keep defining c (and f) even if h^e is partial.

The catch is that we used the values of A_u and A_{u+1} in order to define c_{u+1}. Our commitment to make $\langle c_s \rangle$ total even if $\langle A_s \rangle$ is not, means that our definition of $\langle c_s \rangle$ must be *quicker* than the unfolding of the values of $\langle A_s \rangle$. For $s < \omega$, let $\overline{s}$ be the greatest number below s such that $A_u(x)$ has converged by stage s for all $u, x \leqslant \overline{s}$. Usually, $\overline{s}$ will be much smaller than s. At stage s we need to define c_s, but can read the values of $\langle A_u \rangle$ only for $u \leqslant \overline{s}$.

This is where the function f comes into play. The speed-up of the approximation of A that it allows us to define can be used to prevent unwanted elements from entering W, if A changes back. We return to the situation above, this time with f growing quickly, but still with $q^e = h^e$. Suppose that $n = h^e(t)$ and $u = f(n)$, and s is a stage with $\overline{s} = u + 1$. We see that $A_{u+1}(z) \neq A_u(z)$, and so at stage s we see that we would have liked to define $c_{u+1}(z) \geqslant d^e_t(z)$. But s is much greater than $u+1$; at stage $u+1$, we were not aware of this situation, and so kept $c_{u+1}(z)$ small. At stage s we would like to rectify the situation by defining $s = f(n+1)$ and $c_s(z) \geqslant d^e_t(z)$. Let $v = f(h^e(t+1))$, which is presumably greater than s. We now have two possibilities:

- If $A_s(z) = A_u(z)$, that is, $A(z)$ changed back from its value at stage $u+1$, then the change in $A(z)$ between stages u and $u+1$ need not be recorded in W. In this case, W pays no cost related to z, and so we do not need to charge anything to anyone.
- Otherwise, the change in $A(z)$ from u to $u+1$ persists at stage s, and is recorded in W, which pays at most $d^e_t(z)$. If $A_v(z) = A_s(z)$, then this change persists until stage v, and so the cost is paid by the sum $\sum d^e_t(A_{(f \circ h^e)(t)})$. If $A_v(z) = A_u(z)$, then $A(z)$ must have changed at some stage *after stage* s, and so the cost can be charged to $\sum c_s(A_s)$.

All is well, except that we did not consider yet another commitment of ours, which is to make c benign (and in fact, to make the bound g uniformly computable from the index k for the partial approximation $\langle A_s \rangle$). The idea is to again charge increases in $c_s(z)$ to either the sum $\sum c_s(A_s)$ or the sums $\sum d^e_t(A_{(f \circ h^e)}(t))$. Then every time c increases beyond ϵ, one of these two sums would increase by at least ϵ. Since these sums are bounded by 2^k and 1, respectively, this would ensure $r(\epsilon) \leqslant (1 + 2^k)/\epsilon$, and thus c would be benign.

In the scenario above, before defining $c_s(z) \geqslant d^e_t(z)$, we would like to have evidence that $A_s(z) \neq A_u(z)$, so the cost would actually be paid by one of the sums. To avoid this seeming circularity, we "drip feed" cost in tiny yet increasing steps. In the scenario above, at stage $s_0 = s$, we would increase $c_{s_0}(z)$ by a little bit – not all the way up to $d^e_t(z)$ – and wait for a stage $s_1 > s_0$ at which we see what $A_{s_0}(z)$ is (that is, for a stage s_1 such that $\overline{s}_1 \geqslant s_0$). If $A_{s_0}(z) = A_u(z)$ then we can let $f(n+1) = s_0$. We increased $c_{s_0}(z)$ by something comparable to $c_{u+1}(z)$, and the change in $A(z)$ between stage $u+1$ and stage s shows that this amount was added to $\sum c_s(A_s)$. If $A_{s_0}(z) \neq A_u(z)$, then we increase $c_{s_1}(z)$ again (we can double it), but again not necessarily all the way up to $d^e_t(z)$, and repeat, *while delaying the definition of* $f(n+1)$. Also, since there are infinitely many requirements S^e, we have to scale our target, so that only finitely many such requirements affect the ϵ-increases in $\langle c_s \rangle$; that is, instead of a target of $d^e_t(z)$, we look for $c(z)$ to reach $2^{-(e+1)} d^e_t$.

The last ingredient in the proof is the function q^e – we have not yet explained why we need q^e to provide an even faster speed-up of $\langle W_s \rangle$, compared with $\langle A_{(f \circ h^e)(s)} \rangle$.

Now the point is that as slow as the definition of f is, the function h^e shows its values even more slowly. After all, even if $\langle A_s \rangle$ and f are total, many functions h^e are not. In the scenario above, there may be several stages added to the range of f before we see that $h^e(t) = n$. This means that in trying to define $f(n+1)$, as above, we may suddenly see more requirements S^e worry about more inputs z, as more stages enter the range of $f \circ h^e$. The argument regarding the scenario above breaks down if the stage $v = f(h^e(t+1))$ is not greater than the stage s.

We use the function q^e to mitigate this problem. To keep our accounting straight, we need to make the range of q^e contained in the range of h^e (otherwise we might introduce more changes which we will not be able to charge to the sum $\sum d^e_t(A_{(f \circ h^e)(t)})$). In our scenario above, we now assume that $n = h^e(t)$ is in the range of q^e. The key now is that by delaying the definition of $q^e(t)$, we may assume that $A(z)$ does not change between stage $u = f(n)$ and the last stage currently in the range of f; we use here the assumption that $\langle A_s \rangle$ indeed converges to A. And so the strategy above can work, because even though we declared new values of f beyond u, at the time we declare that $n \in \operatorname{range} q^e$, we see that these new values would not spoil the application of our basic strategy.

3.4. **Construction.** Let $\langle A_s \rangle$ be a uniformly computable sequence of partial functions, and let k be a constant. As mentioned above, for all $s < \omega$, we let $\overline{s}$ be the greatest number below s such that for all x and u bounded by $\overline{s}$, $A_u(x)$ converges at stage s.

We define a uniformly computable sequence $\langle c_s \rangle$. We start with $c_0(z) = 2^{-z}$ for all $z < \omega$. At every stage s, we measure our approximation for $\sum c_s(A_s)$; this, of course, would be the sum of the costs $c_u(x_u)$, where $u, x_u \leqslant \overline{s}$ and x_u is the least $x \leqslant \overline{s}$ such that $A_u(x_u) \neq A_{u-1}(x_u)$. If at stage s our current approximation for this sum exceeds 2^k, we halt the construction, and let $c = c_s$.

Otherwise, we let $\widehat{s} \leqslant s$ be greatest such that $c_{\widehat{s}} \neq c_{\widehat{s}-1}$ ($\widehat{s} = 0$ if there is no such stage). So $c_s = c_{\widehat{s}}$. Stage $\widehat{s} - 1$ is the last stage before s at which we took some action toward assuaging the fears of various requirements S^e, which is a step toward defining a new value of f. By the beginning of stage $s > 0$, the function f is defined (and increasing) on inputs $0, 1, \ldots, m_s$; we start with $f(0) = 0$. We will ensure that $f(m_s) \leqslant \overline{s}$.

For all $e < \omega$, we define a function q^e. To begin with, q^e is defined nowhere. Once we see that $h^e(0)\!\downarrow$, say at stage s^e, we define $q^e(0) = h^e(0)$. Henceforth, at the beginning of stage $s > s^e$, the function q^e is defined on $0, 1, \ldots, t^e_s$, is increasing, and the range of q^e is contained in both the range of h^e and the domain of f. That is, $q^e(t^e_s) \leqslant m_s$.

Similarly to measuring the sum $\sum c_s(A_s)$, for each e, we measure the sum $\sum d^e_t(A_{(f \circ h^e)(t)})$; at stage s we add the costs $d^e_t(y_t)$, where $t < s$ is such that $h^e(t) \leqslant m_s$, and y_t is the least such that $A_{f(h^e(t))}(y) \neq A_{f(h^e(t-1))}(y)$. The requirement S^e is only *active* at stage s if this sum, as calculated at this stage, is bounded by 1.

Let $s > 0$. If $\overline{s} = \overline{s-1}$, or $\overline{s} \leqslant \widehat{s}$, then we let $c_{s+1} = c_s$ and do not change f (so $m_{s+1} = m_s$). Suppose otherwise. Let $e < m_s$ and $z < m_s$. We say that the requirement S^e is *worried about* z at stage s if S^e is active at stage s, t^e_s is defined (that is, $s > s^e$), and:

- $A_{\overline{s}}(z) \neq A_{f(q^e(t^e_s))}(z)$; and
- $c_s(z) < 2^{-(e+1)} d^e_{t^e_s}(z)$.

Now there are two cases:

(1) If for all $e < m_s$ and $z < m_s$, the requirement S^e is not worried about z at stage s, then we add $m_s + 1$ to $\operatorname{dom} f$ by letting $f(m_s + 1) = \overline{s}$ (so $m_{s+1} = m_s + 1$). Note that indeed $f(m_{s+1}) \leqslant \overline{s+1}$. We also let $c_{s+1} = c_s$. In the discussion, we used s for the new value of f instead of $\overline{s}$, but since $\overline{s} \geqslant \widehat{s}$, $c_s = c_{\overline{s}}$, and so $\overline{s}$ will suffice.

(2) Otherwise, we let z be the least number about which some requirement S^e (with $e < m_s$) is worried at stage s. For all $y \leqslant m_s$ we let $c_{s+1}(y) = \max\{c_s(y), 2c_s(z)\}$. For $y > m_s$ we let $c_{s+1}(y) = c_s(y)$. We do not change f, so $m_{s+1} = m_s$.

This determines c_s, f, and m_{s+1} at the end of stage s. If $m_{s+1} = m_s$ then this is the end of the stage. Otherwise, we now possibly make changes to the functions q^e. Let $e < s$ such that s^e has been observed, that is, such that t^e_s is already defined. If there is some $n \in (q^e(t^e_s), m_{s+1}]$ such that n is observed to be in the range of h^e at stage s, and such that $A_{n'} \upharpoonright (t^e_s + 1)$ is constant for $n' \in [f(n), f(m_{s+1})]$, then we extend $\operatorname{dom} q^e$ by letting $q^e(t^e_s + 1)$ be the least such n; so $t^e_{s+1} = t^e_s + 1$. Note that we can enquire about the values of $A_n \upharpoonright (t^e_s + 1)$ because $f(m_{s+1}) = \overline{s}$ and $t^e_s \leqslant m_s < \overline{s}$.

If there is no such n, then we leave q^e unchanged (so $t^e_{s+1} = t^e_s$). This concludes the construction.

3.5. **Verification.** The sequence $\langle c_s \rangle$ is total. Each function c_s is non-increasing, and its limit is 0. For all z, $c_{s+1}(z) \geqslant c_s(z)$.

Lemma 3.4. *For all s and z, $c_s(z) \leqslant 1$.*

Proof. By induction on s. The point is that if at stage s we let $c_{s+1}(y) = 2c_s(z)$ for z as described in the construction, then for some $e < m_s$ we have

$$c_s(z) < 2^{-e-1} d^e_{t^e_s}(z) \leqslant 1/2,$$

because by assumption $d^e_t(z) \leqslant 1$ for all t and z. So

$$c_{s+1}(y) = 2c_s(z) \leqslant 1. \qquad \square$$

Let $c = \lim c_s$. We will soon show that $\langle c_s \rangle$ witnesses that c is benign, and so c satisfies the limit condition. Moreover, the benignity bound for $\langle c_s \rangle$ is uniformly computable from k.

Fix $\epsilon > 0$ rational. Let N be such that $2^{-N} < \epsilon \leqslant 2^{-N+1}$. Let $m_1(\epsilon), \ldots, m_{r(\epsilon)}(\epsilon)$ be the markers associated with $\langle c_s \rangle$. After the current paragraph, we will never mention these again, so there should be no cause for confusion with the parameters m_s of the construction. We would like to state that $c_{m_{i+1}(\epsilon)-1}(m_i(\epsilon)) < \epsilon$ for all i, but unfortunately, this is only eventually true. Note that because we began by defining $c_0(z) = 2^{-z}$, the first several of these markers are predetermined: $m_i(\epsilon) = i - 1$ for all $i \leqslant N$, and so our desired inequality fails for $m_i(\epsilon)$ with $i < N$. However, by a straightforward induction one can see that $c_s(s) = 2^{-s}$ for all s, and thus $c_{m_{i+1}(\epsilon)-1}(m_i(\epsilon)) < \epsilon$ for all $i \geqslant N$. Consequently, we define the set

$$S = \{m_i(\epsilon) - 1 : i \geqslant N\}.$$

Then for $s_0 < s_1$ sequential elements of S, we have

$$c_{s_1}(s_0 + 1) < \epsilon \leqslant c_{s_1+1}(s_0 + 1).$$

So if $s \in S$ is not the least element, we know that at stage s we defined $c_{s+1} \neq c_s$ because some strategy e was worried about some element $z < m_s$ (here m_s is the parameter from the construction) with $c_s(z) \geqslant 2^{-N}$. We wish to separate out the elements of S according to which strategy was worried, so we define the set $J(e)$ to consist of those $s \in S$ other than the least element of S such that there is a $z < m_s$ with $c_s(z) \geqslant 2^{-N}$ and e worried about z at stage s. Note that multiple strategies could be worried at the same stage, so $J(e)$ and $J(e')$ might not be disjoint for $e \neq e'$. This will not concern us.

Lemma 3.5. *$J(e)$ is empty for all $e \geqslant N - 1$.*

Proof. If $s \in J(e)$, then there is some z with $c_s(z) \geqslant 2^{-N}$ and e worried about z at stage s. By the definition of "worry", $c_s(z) < 2^{-(e+1)} d^e_{t^e_s}(z)$. But d^e is bounded by 1, so $2^{-N} < 2^{-(e+1)}$, and thus $e < N - 1$. □

Lemma 3.6. *Fix $e < N - 1$. If I is an interval of stages such that t^e_s is defined and constant on I, then $|I \cap J(e)| < N - e$.*

Proof. Let t be the unique value taken by t^e_s on I. Then for $s \in I \cap J(e)$, from the definition of "worry", there is some $z' < m_s$ with $c_s(z') < 2^{-e+1} d^e_t(z')$. Since $d^e_t(z) = 0$ for all $z > t$, we know that $z' \leqslant t$.

Now, let $I \cap J(e) = \{s_1 < s_2 < \dots\}$. We claim by induction that $c_{s_i+1}(z) \geqslant 2^{-N+i}$ for all $z < t$. We begin with $i = 1$. Since $s_1 \in J(e)$, we know that $c_{s_1+1}(z) \geqslant 2^{-N+1}$ for all $z \leqslant m_{s_1}$. Since $t \leqslant m_{s_1}$, this includes all $z \leqslant t$.

For the inductive step, since $s_{i+1} \in J(e)$, there is some $z' \leqslant t$ about which e is worried at stage s_{i+1}. Let $z'' < m_{s_{i+1}}$ be the element chosen by the construction for defining $c_{s_{i+1}+1}$. Since z'' is chosen least, we know that $z'' \leqslant z'$, so by monotonicity and the inductive hypothesis we have $c_{s_{i+1}}(z'') \geqslant c_{s_i+1}(z'') \geqslant 2^{-N+i}$. So $c_{s_{i+1}+1}(z) \geqslant 2^{-N+i+1}$ for all $z \leqslant t$.

Now, if there were some $s_{N-e} \in I \cap J(e)$, by the definition of "worry" we would have that $c_{s_{N-e}}(z') < 2^{-(e+1)} d^e_t(z')$ for some $z' \leqslant t$. But $d^e_t(z') \leqslant 1$ and $c_{s_{N-e}}(z') \geqslant c_{s_{N-e-1}+1}(z') \geqslant 2^{-(e+1)}$, a contradiction. □

Lemma 3.7. *Fix $e < N - 1$. Suppose $\langle [\ell_i, r_i) \rangle$ is a sequence of finite half-open intervals such that:*

1. *For each i, t^e_s is defined and constant on $[\ell_i, r_i)$;*
2. *$t^e_{\ell_i} \neq t^e_{r_i}$;*
3. *For each i, $r_i \leqslant \ell_{i+1}$; and*
4. *For each i, $[\ell_i, r_i) \cap J(e) \neq \emptyset$.*

Then the sequence has length at most $2^{N-e-1} + 2^{N+k}$.

Proof. The point is that each such interval contributes a distinct charge to either $\sum c_s(A_s)$ or $\sum d^e_s(A_{(f \circ h^e)(s)})$.

For each i, let $t_i = t^e_{\ell_i}$, $n_i = q^e(t_i)$ and $k_i = q^e(t^e_{r_i})$. Note that $n_i = q^e(t^e_s)$ for all $s \in [\ell_i, r_i)$. Fix some $s_i \in [\ell_i, r_i) \cap J(e)$. Then there is some $z < m_{s_i}$ with $A_{\overline{s_i}}(z) \neq A_{f(n_i)}(z)$ and $2^{-N} \leqslant c_{s_i}(z) < 2^{-(e+1)} d^e_{t_i}(z)$. Consider $A_{f(k_i)}(z)$.

If $A_{f(k_i)}(z) = A_{\overline{s_i}}(z)$, then $A_{f(k_i)}(z) \neq A_{f(n_i)}(z)$, and n_i, k_i are in the range of h^e. So there must be some s with $n_i \leqslant h^e(s-1) < h^e(s) \leqslant k_i$ and $A_{(f \circ h^e)(s-1)}(z) \neq A_{(f \circ h^e)(s)}(z)$. Since the range of q^e is contained in the range of h^e and both are increasing, we have $s \geqslant t_i$. So an amount of at least

$$d^e_s(z) \geqslant d^e_{t_i}(z) \geqslant 2^{e+1} c_{s_i}(z) \geqslant 2^{-N+e+1}$$

is added to $\sum d^e_s(A_{(f\circ h^e)(s)})$. Since $n_i < h^e(s) \leqslant k_i$, the charges of this sort for distinct i are disjoint, and so this case can occur at most 2^{N-e-1} times.

Otherwise, $A_{f(k_i)}(z) \neq A_{\overline{s_i}}(z)$. By the same argument as in Lemma 3.6, note that $z \leqslant t_i$. So by q^e's choice of k_i (which was made at stage $r_i - 1$), we know that $f(k_i) > \overline{s_i}$. So there is some $s \in (\overline{s_i}, f(k_i)]$ with $A_{s-1}(z) \neq A_s(z)$. By assumption we have $\overline{s_i} \geqslant \widehat{s_i}$, so $c_{s_i} = c_{\overline{s_i}}$. So an amount of at least

$$c_s(z) \geqslant c_{\overline{s_i}}(z) = c_{s_i}(z) \geqslant 2^{-N}$$

is added to $\sum c_s(A_s)$. Since n_i is in the domain of f by stage s_i, we know $f(n_i) \leqslant \overline{s_i}$. It follows that $f(n_i) < s \leqslant f(k_i)$, and thus the charges of this sort for distinct i are disjoint, and so this case can occur at most 2^{N+k} times. □

Lemma 3.8. $r(\epsilon) \leqslant N^2 \cdot 2^{N+k+1} + N$.

Proof. Recall that $r(\epsilon) = (N-1) + |S|$, and every element of S save the least element occurs in some $J(e)$. Thus

$$r(\epsilon) \leqslant N + \sum_e |J(e)|.$$

By Lemma 3.5, we may restrict the sum to $e < N - 1$. To bound the size of $J(e)$, partition $[s^e, \infty)$ into maximal intervals upon which t^e_s is constant. Now, restrict to those which contain an element of $J(e)$. This collection may contain a single interval of the form $[\ell, \infty)$, but all the rest will be of the form required for Lemma 3.7. So there can be at most $2^{N-e-1} + 2^{N+k} + 1$ such intervals. By Lemma 3.6, each such interval contains at most $N - e - 1$ many elements of $J(e)$.

We thus have

$$\begin{aligned} r(\epsilon) &\leqslant N + \sum_{e<N-1} |J(e)| \\ &\leqslant N + \sum_{e<N-1} (N-e-1)(2^{N-e-1} + 2^{N+k} + 1) \\ &\leqslant N + \sum_{e<N-1} N \cdot 2^{N+k+1} \\ &\leqslant N + N^2 \cdot 2^{N+k+1}. \end{aligned}$$

□

It follows that c is benign with the limit condition.

We now assume that the sequence $\langle A_s \rangle$ is total and converges to a limit A, and that $\sum c_s(A_s) \leqslant 2^k$. The construction is never halted.

Lemma 3.9. *The function f is total.*

Proof. Suppose, for contradiction, that f is not total; at some stage s^* we define the last value $m^* = m_{s^*} + 1$ on which f is defined, and for all $s > s^*$ we have $m_s = m^*$. No function q^e is extended after stage s^*, so for all $e < \omega$, the value t^e_s for all $s > s^*$ is fixed.

Because $\langle A_s \rangle$ is total, the function $s \mapsto \overline{s}$ is unbounded. So there are infinitely many stages $s > s^*$ for which $\overline{s} > \widehat{s}$; let T be the collection of these stages. By assumption, at each stage $s \in T$ there is some number $z < m^*$ about which some requirement S^e (for $e < m^*$) worries at that stage. For $e < m^*$ and $z < m^*$, let $T(e, z)$ be the collection of stages $s \in T$ at which S^e worries about z. There are some $e < m^*$ and $z < m^*$ such that $T(e, z)$ is infinite.

Let $t = t^e_s$ for $s \in T$. At each stage $s \in T(e,z)$ we have $c_s(z) < \delta_e d^e_t(z)$. At stage s we define $c_{s+1}(z) = 2c_s(y)$ for some $y \leqslant z$, and $c_s(y) \geqslant c_s(z)$. We note that $c_s(z) > 0$ because $c_0(z) = 2^{-z}$. This quickly (i.e. in z steps) leads to a contradiction. □

We let W be the change set of $\langle A_{f(n)} \rangle$. Then W computes A. It remains to show that every requirement S^e is met. Fix $e < \omega$. Suppose that h^e is total, and that $\sum d^e_t(A_{(f \circ h^e)(t)}) \leqslant 1$.

Lemma 3.10. *The function q^e is total.*

Proof. Suppose, for contradiction, that q^e is not total; a final value t^* is added to $\operatorname{dom} q^e$ at some stage s_0, so $t^e_s = t^*$ for all $s > s_0$. We note that S^e is active at every stage.

Let $s_1 > s_0$ be a stage such that for all $s \geqslant s_1$, $A_s \restriction t^* + 1 = A \restriction t^* + 1$. Let $k < \omega$ such that $f(h^e(k)) > s_1$. By Lemma 3.9, there is some stage $s > s_1$ such that $m_{s+1} > m_s$ and $m_s > h^e(k)$. Then at stage s we are instructed to define $q^e(t^*+1)$, a contradiction. □

The following lemma concludes the proof of Theorem 1.13.

Lemma 3.11. *The sum $\sum d^e_t(W_{q^e(t)})$ is finite.*

Proof. For $t \in \omega$, let y_t be the least number such that $W_{q^e(t)}(y_t) \neq W_{q^e(t-1)}(y_t)$. We need to show that $\sum d^e_t(y_t)$ is finite.

Fix t. We may assume that $y_t < t$, for otherwise $d^e_t(y_t) = 0$. Let $y_t = (z_t, k)$ for some $k < \omega$. So $z_t \leqslant y_t$ (using the standard pairing function), and $A_{f(n)}(z_t) \neq A_{f(n-1)}(z_t)$ for some $n \in (q^e(t-1), q^e(t)]$. Taking the least such n, we have $A_{f(n-1)}(z_t) = A_{f(q^e(t-1))}(z_t)$ and so $A_{f(n)}(z_t) \neq A_{f(q^e(t-1))}(z_t)$.

Now there are two possibilities: either $A_{f(n)}(z_t) = A_{f(q^e(t))}(z_t)$, or not.

In the first case, we have $A_{f(q^e(t))}(z_t) \neq A_{f(q^e(t-1))}(z_t)$. Since both $q^e(t-1)$ and $q^e(t)$ belong to the range of h^e, and $q^e(x) \geqslant h^e(x)$ for all x, we see that there is some $x \geqslant t$ such that $A_{f(h^e(x))}(z_t) \neq A_{f(h^e(x-1))}(z_t)$. This means that an amount of at least $d^e_x(z_t) \geqslant d^e_t(z_t)$ is added to the sum $\sum d^e_t(A_{(f \circ h^e)(t)})$, at a stage x such that $h^e(x) \in (q^e(t-1), q^e(t)]$. Thus the charges for distinct such stages t are disjoint. This shows that the total contribution to the sum $\sum d^e_t(W_{q^e(t)})$ by stages t falling under the first case is at most 1.

In the second case, let s be the stage at which n is added to $\operatorname{dom} f$, that is, $n = m_{s+1} > m_s$. The main point is that $t^e_s = t - 1$. For $q^e(t) > n = m_s$, so $t^e_s < t$ or t^e_s is undefined if $s < s^e$. But if $t^e_s < t-1$ or $s < s^e$, then $q^e(t-1)$ is not defined before stage s. Let $u \geqslant s$ be the stage at which $q^e(t-1)$ is defined. The number m_{u+1} is in the range of h^e, and $m_{u+1} \geqslant m_{s+1} = n$. Since $z_t \leqslant t-1$, the condition for defining $q^e(t-1)$ at stage u implies that $A_{f(m)}(z_t)$ is constant for all $m \in [q^e(t-1), m_{u+1}]$, but $A_{f(q^e(t-1))}(z_t) \neq A_{f(n)}(z_t)$.

At stage s, S^e is not worried about z_t (note that $z_t \leqslant t-1 = t^e_s < m_s$). The requirement S^e is active at stage s. We have $f(n) = \overline{s}$, and $t^e_s = t-1$, and $A_{f(q^e(t-1))}(z_t) \neq A_{f(n)}(z_t)$, which rewriting gives $A_{f(q^e(t^e_{\overline{s}}))}(z_t) \neq A_{\overline{s}}(z_t)$. So the only reason that S^e does not worry about z_t at stage s is that $c_s(z_t) \geqslant \delta_e d^e_t(z_t)$. Now $f(n) = \overline{s} > \widehat{s}$, and $c_s(z_t) = c_{\widehat{s}}(z_t)$; so altogether, we see that $c_{f(n)}(z_t) \geqslant \delta_e d^e_t(z_t)$.

Because this is the second case, we have $A_{f(n)}(z_t) \neq A_{f(q^e(t))}(z_t)$, so there is some stage $u \in (f(n), f(q^e(t))]$ such that $A_u(z_t) \neq A_{u-1}(z_t)$. As $u > f(n)$ we have $c_u(z_t) \geqslant c_{f(n)}(z_t) \geqslant \delta_e d^e_t(z_t)$. So an amount of at least $\delta_e d^e_t(z_t)$ is added to the

sum $\sum c_s(A_s)$. We have $u \in (f(q^e(t-1)), f(q^e(t))]$ so the charges for distinct t are disjoint. So the total amount contributed to the sum $\sum d^e_t(W_{q^e(t)})$ by stages t falling under the second case is bounded by $2^k/\delta_e$, which is finite. □

References

[1] George Barmpalias, Rodney Downey, and Noam Greenberg, *K-trivial degrees and the jump-traceability hierarchy*, Proc. Amer. Math. Soc. **137** (2009), no. 6, 2099–2109, DOI 10.1090/S0002-9939-09-09761-5. MR2480292 (2009k:03073)

[2] Laurent Bienvenu, Adam R. Day, Noam Greenberg, Antonín Kučera, Joseph S. Miller, André Nies, and Dan Turetsky, *Computing K-trivial sets by incomplete random sets*, Bull. Symb. Log. **20** (2014), no. 1, 80–90, DOI 10.1017/bsl.2013.3. MR3230826

[3] Laurent Bienvenu, Rod Downey, Noam Greenberg, André Nies, and Dan Turetsky, *Characterizing lowness for Demuth randomness*, J. Symb. Log. **79** (2014), no. 2, 526–560, DOI 10.1017/jsl.2013.21. MR3224979

[4] Gregory J. Chaitin, *Information-theoretic characterizations of recursive infinite strings*, Theoret. Comput. Sci. **2** (1976), no. 1, 45–48. MR0413595 (54 #1709)

[5] Gregory J. Chaitin, *Nonrecursive infinite strings with simple initial segments*, IBM Journal of Research and Development **21** (1977), 350–359.

[6] Peter Cholak, Rodney Downey, and Noam Greenberg, *Strong jump-traceabilty. I. The computably enumerable case*, Adv. Math. **217** (2008), no. 5, 2045–2074, DOI 10.1016/j.aim.2007.09.008. MR2388085 (2008k:03087)

[7] Rod Downey and Noam Greenberg, *Strong jump-traceability II: K-triviality*, Israel J. Math. **191** (2012), no. 2, 647–665, DOI 10.1007/s11856-011-0217-z. MR3011490

[8] Rodney G. Downey and Denis R. Hirschfeldt, *Algorithmic randomness and complexity*, Theory and Applications of Computability, Springer, New York, 2010. MR2732288 (2012g:03001)

[9] Rodney G. Downey, Denis R. Hirschfeldt, André Nies, and Frank Stephan, *Trivial reals*, Proceedings of the 7th and 8th Asian Logic Conferences, Singapore Univ. Press, Singapore, 2003, pp. 103–131, DOI 10.1142/9789812705815_0005. MR2051976 (2005a:03089)

[10] Rodney Downey, Denis R. Hirschfeldt, André Nies, and Sebastiaan A. Terwijn, *Calibrating randomness*, Bull. Symbolic Logic **12** (2006), no. 3, 411–491. MR2248591 (2007j:03055)

[11] Santiago Figueira, André Nies, and Frank Stephan, *Lowness properties and approximations of the jump*, Ann. Pure Appl. Logic **152** (2008), no. 1-3, 51–66, DOI 10.1016/j.apal.2007.11.002. MR2397489 (2009e:03076)

[12] Noam Greenberg, Denis R. Hirschfeldt, and André Nies, *Characterizing the strongly jump-traceable sets via randomness*, Adv. Math. **231** (2012), no. 3-4, 2252–2293, DOI 10.1016/j.aim.2012.06.005. MR2964638

[13] Noam Greenberg and André Nies, *Benign cost functions and lowness properties*, J. Symbolic Logic **76** (2011), no. 1, 289–312, DOI 10.2178/jsl/1294171001. MR2791349 (2012d:03096)

[14] Noam Greenberg and Daniel D. Turetsky, *Strong jump-traceability and Demuth randomness*, Proc. Lond. Math. Soc. (3) **108** (2014), no. 3, 738–779, DOI 10.1112/plms/pdt040. MR3180595

[15] Denis R. Hirschfeldt, André Nies, and Frank Stephan, *Using random sets as oracles*, J. Lond. Math. Soc. (2) **75** (2007), no. 3, 610–622, DOI 10.1112/jlms/jdm041. MR2352724 (2008g:68051)

[16] Rupert Hölzl, Thorsten Kräling, and Wolfgang Merkle, *Time-bounded Kolmogorov complexity and Solovay functions*, Mathematical foundations of computer science 2009, Lecture Notes in Comput. Sci., vol. 5734, Springer, Berlin, 2009, pp. 392–402, DOI 10.1007/978-3-642-03816-7_34. MR2539508 (2011f:68076)

[17] Antonín Kučera and André Nies, *Demuth randomness and computational complexity*, Ann. Pure Appl. Logic **162** (2011), no. 7, 504–513, DOI 10.1016/j.apal.2011.01.004. MR2781092 (2012g:03113)

[18] Joseph S. Miller and André Nies, *Randomness and computability: open questions*, Bull. Symbolic Logic **12** (2006), no. 3, 390–410. MR2248590 (2007c:03059)

[19] Webb Miller and D. A. Martin, *The degrees of hyperimmune sets*, Z. Math. Logik Grundlagen Math. **14** (1968), 159–166. MR0228341 (37 #3922)

[20] Keng Meng Ng, *On strongly jump traceable reals*, Ann. Pure Appl. Logic **154** (2008), no. 1, 51–69, DOI 10.1016/j.apal.2007.11.014. MR2413929 (2010b:03045)

[21] André Nies, *Lowness properties and randomness*, Adv. Math. **197** (2005), no. 1, 274–305, DOI 10.1016/j.aim.2004.10.006. MR2166184 (2006j:68052)
[22] André Nies, *Eliminating concepts*, Computational prospects of infinity. Part II. Presented talks, Lect. Notes Ser. Inst. Math. Sci. Natl. Univ. Singap., vol. 15, World Sci. Publ., Hackensack, NJ, 2008, pp. 225–247, DOI 10.1142/9789812796554_0012. MR2449467 (2010a:68067)
[23] André Nies, *Computability and randomness*, Oxford Logic Guides, vol. 51, Oxford University Press, Oxford, 2009. MR2548883 (2011i:03003)
[24] André Nies, *Computably enumerable sets below random sets*, Ann. Pure Appl. Logic **163** (2012), no. 11, 1596–1610, DOI 10.1016/j.apal.2011.12.011. MR2959662
[25] André Nies, *Calculus of cost functions.* In preparation.
[26] Jean Raisonnier, *A mathematical proof of S. Shelah's theorem on the measure problem and related results*, Israel J. Math. **48** (1984), no. 1, 48–56, DOI 10.1007/BF02760523. MR768265 (86g:03082b)
[27] Robert M. Solovay, Draft of paper (or series of papers) related to Chaitin's work. IBM Thomas J. Watson Research Center, Yorktown Heights, NY, 215 pages, 1975.
[28] Sebastiaan A. Terwijn, *Computability and Measure*. PhD thesis, University of Amsterdam, 1998.
[29] Sebastiaan A. Terwijn and Domenico Zambella, *Computational randomness and lowness*, J. Symbolic Logic **66** (2001), no. 3, 1199–1205, DOI 10.2307/2695101. MR1856736 (2002j:03044)

Department of Mathematics, University of Wisconsin, Madison, Wisconsin 53706
E-mail address: ddiamondstone@gmail.com
URL: http://math.wisc.edu/~diamondstone/

School of Mathematics, Statistics and Operations Research, Victoria University of Wellington, Wellington, New Zealand
E-mail address: greenberg@msor.vuw.ac.nz
URL: http://homepages.mcs.vuw.ac.nz/~greenberg/

Kurt Gödel Research Center, University of Vienna, 1090 Vienna, Austria
E-mail address: turetsd4@univie.ac.at
URL: http://tinyurl.com/dturetsky