March 20, 2017

STRONG JUMP-TRACEABILITY

NOAM GREENBERG AND DAN TURETSKY

Abstract. We review the current knowledge concerning strong jump-traceability. We cover the known results relating strong jump-traceability to randomness, and those relating it to degree theory. We also discuss the techniques used in working with strongly jump-traceable sets. We end with a section of open questions.

1. Traces

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 [52] 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 [10, 9] 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 [23, 47, 48, 30]). 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 (for the latter see [39]). These other notions express computational weakness as an oracle: they say that a set is too weak to compute better-than-computable compressions, patterns in randoms, or null sets covering the reals computing it.

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.*

Traceability has its roots in set theory, where traces are referred to as *slaloms.* Bartoszyński [5] first introduced them for studying cardinal characteristics of the continuum. For example, he used the method of slaloms to prove that $\text{cofin}(\mathcal{N}) = \mathfrak{d}(\in^*)$. The study of traceability in a computability context was initiated by Terwijn and Zambella [54].

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 $z \in \operatorname{dom}\psi$, $\psi(z) \in T(z)$.

Greenberg was supported by the Marsden Fund and by a Rutherford Discovery Fellowship from the Royal Society of New Zealand, and by a Turing research fellowship "Mind, mechanism and mathematics" from the Templeton foundation. Turetsky was supported by the Austrian Science Fund FWF project I 1238.

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; and traces consisting of smaller sets do a better job of guessing the value of the function, and thus give more information about the function. In some sense the notion of a trace is quite old in computability theory. W. Miller and Martin [42] 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 [54] and Kjos-Hanssen, Nies and Stephan [37] used a uniform version of hyperimmune-freeness to characterise lowness for Schnorr randomness, thereby giving a "combinatorial" characterisation of this lowness notion; Kihara later observed that their proof was essentially an effectivization of Bartoszyński's proof.

The characterisations of hyperimmune-freeness and of lowness for Schnorr randomness demonstrate how traces can be used to capture the notion that a set is easy to approximate. Traces can also be used in a dual method by fixing the functions to be traced and varying the oracles used to perform the tracing. This can capture the notion that a set is powerful as an oracle. For example, a degree $\mathbf{a}$ is high (in the sense that $\mathbf{a}' = \mathbf{0}''$) if and only if every partial $\mathbf{0}'$-computable function has an $\mathbf{a}$-c.e. trace.

Note that traces are defined for partial functions, rather than only total functions. This is in contrast to the set-theoretic treatment of slaloms, where it is sufficient to only consider total functions; any partial function can be extended to a total function of the same set-theoretic complexity (e.g. constructible), while this fails for various computability theoretic notions of complexity.

As in the characterisation of the high degrees, often 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 (following terminology of Schnorr's); they calibrate rates of growth of computable functions.

Definition 1.3. An *order function* is a nondecreasing, 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, a degree is *computably traceable*, the uniform version of hyperimmune-freeness used by Terwijn and Zambella, if there is an order h such that every total function in the degree has a computable h-trace. As mentioned above, A set A is computably traceable if and only if every Schnorr random sequence is A-Schnorr random [54, 37]. There is also a connection between computable traceability and strong reducibilities: the truth-table degree of a set is computably traceable if and only if it is Schnorr trivial [].

Replaced by its computably-enumerable variant, *c.e. traceability* was discovered to have strong links to traditional degree theory. For example, Ishmukhametov [32] showed that a c.e. traceable Turing degree has a strong minimal cover; and in the context of the c.e. degrees, c.e. traceability is equivalent to the notion of array computability [24, 22]. Further, a weak truth-table degree is computably traceable if and only if it is anti-complex [26].

As these only consider total functions, the choice of order function turns out not to matter; if there is an order h such that every $\mathbf{a}$-computable function is traced by a computable (resp. c.e.) h-trace, then every $\mathbf{a}$-computable function is computably (resp. c.e.) traced by a g-trace for every order g. An analogous fact is used in the treatment of slaloms, which are typically taken to be bounded by n^2.

Zambella (see [53]) 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 [47], 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 ψ 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 (in the sense of Kolmogorov complexity) by approximately $\log n + \log h(n)$ 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 [3, 31] 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.

2. Strong jump-traceability

An attempt toward a solution of this problem lead 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 2.1 (Figueira, Nies, Stephan [25])**.** Let h be an order function. An oracle $A \in 2^\omega$ is *h-jump-traceable* if every partial A-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. [1]

Figueira, Nies, and Stephan characterised the strongly jump-traceable sets as those that are "lowly" for plain kolmogorov complexity C in that $C(n) \leqslant^+ C^A(n) + h(C^A(n))$ for any order function h. They 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 [41] asked if strong jump-traceability and K-triviality coincided.

[1]The definition we have given is not quite that originally presented by Figueira, Nies, and Stephan; rather than tracing all partial A-computable functions, they defined h-jump-traceability only in terms of tracing the universal partial A-computable function $J^A(e) = \Phi_e^A(e)$. While this does not change the notion of strong jump-traceability, it causes h-jump-traceability to depend on one's choice of Gödel numbering. Hence the above definition has become the standard.

Cholak, Downey and Greenberg answered this question in the negative. They showed [11, 20] that every strongly jump-traceable set is K-trivial but that the containment is proper. In fact they constructed an order h such that every h-jump-traceable set is K-trivial. Nonetheless, further research revealed remarkable similarities between the strongly jump-traceable Turing degrees and the ideal of K-trivial degrees.

2.1. **Structural results.** Nies showed [47] that every K-trivial set is computable from a c.e. one. This says that the class is "inherently" enumerable. Because it is closed downward under Turing reducibility, it cannot be the case that every K-trivial set is c.e. However bounding by c.e. sets shows that as far as constructions go, we may restrict ourselves to c.e. sets. The same holds for strongly jump-traceable sets:

Theorem 2.2 (Diamondstone, Greenberg, Turetsky [19])**.** *Every strongly jump-traceable set is computable from a c.e. one.*

In particular, unlike all other classes defined by a traceability concept, there are only countably many strongly jump-traceable sets.

The K-trivial degrees form an ideal in the Turing degrees.

Theorem 2.3 (Cholak, Downey, Greenberg [11];[19])**.** *The class of strongly jump-traceable sets is closed under join. Hence the strongly jump-traceable degrees form an ideal in the Turing degrees.*

The presentation here deviates from the historical order. First, in [11] the authors showed that the join of two c.e. strongly jump-traceable sets is strongly jump-traceable. In other words that the computably enumerable, strongly jump-traceable degrees form an ideal in the c.e. Turing degrees. Theorem 2.2 (which was proved later) is then used to extend the result to all strongly jump-traceable sets and degrees.

A similar process shows the coincidence of the class of strongly jump-traceable sets with another class defined by a lowness property. Recall that a set $A \in 2^\omega$ is *superlow* if its Turing jump A' is weak truth-table reducible to the halting problem $\varnothing'$: the use of the reduction is bounded by a computable function. Equivalently, A' has a computable approximation for which the number of mind-changes on each input is bounded by a computable function. In analogy with the definition of strong jump-traceablility, Figueira, Nies and Stephan [25] defined a set A to be *strongly superlow* if for any order function h there is a computable approximation of A' with mind-changes bounded by h. On the other hand, a set A is strongly jump-traceable if, bounded by any given computable growth-rate, one can enumerate finitely many possible values of the jump function $J^A(e) = \Phi_e^A(e)$.

Note the distinction between the jump *function* and the jump *set* $A' = \operatorname{dom} J^A$. For strong superlowness we give an approximation, which is stronger than listing values; but we only approximate the jump set rather than the jump function. That is, we approximate whether a given n is in the domain of J^A, while strong jump-traceability provides a list of values, one of which is $J^A(n)$ if n is in the domain.

Figueira, Nies and Stephan showed that every strongly superlow set is strongly jump-traceable, and that the two notions are equivalent on the c.e. sets. Hence, Theorem 2.2 implies:

Corollary 2.4 ([19])**.** *A set is strongly jump-traceable if and only if it is strongly superlow.*

As mentioned above, Cholak, Downey and Greenberg [11] showed that every c.e. strongly jump-traceable set is K-trivial; this was extended by Downey and Greenberg [20] to all sets. In hindsight, as above, the full result also follows from Theorem 2.2. Later research [3, 55, 31] further studied the connection between K-triviality and rates of growth of bounds of traces.

Unlike the treatment of total functions, or slaloms, since strong jump-traceability is defined for partial functions, we cannot replace all orders with a single order. The ultimate result along these lines is by Ng [43], who significantly extended the techniques used to separate K-triviality from strong jump-traceability to show that the hierarchy of rates of growth on traces does not collapse in the Turing degrees. In turn he utilised this to calculate the complexity of the notion.

Theorem 2.5 (Ng)**.**

(1) *For every order function h there is a (much slower) order function g such that there is a degree which is h-jump-traceable but not g-jump-traceable. Thus for no order function h do the strongly jump-traceable degrees coincide with the h-jump-traceable degrees.*
(2) *The index set of c.e., strongly jump-traceable sets is Π^0_4 complete.*

In contrast, the c.e. K-trivial sets have a Σ^0_3 index set. In further work [46], Ng defined a yet more restricted class (which he called *hyper jump-traceable*) which in contrast with strong jump-traceability does not contain a promptly simple set.

2.2. Connections to randomness.

A surprising aspect of the study of the K-trivial sets is their strong connection to randomness: after all, under the yardstick of prefix-free complexity, the K-trivial sets are the least random possible. Hirschfeldt, Nies and Stephan [30] showed that a c.e. set computable from an incomplete random set must be K-trivial.[2] The converse — that every K-trivial set is computable from an incomplete ML-random set — remained an open problem for almost a decade; it was recently resolved in the affirmative [6].

The strongly jump-traceable degrees have an even closer relationship with randomness (using Turing reducibility). A full analogue of the "covering problem" has been found, using a notion of randomness introduced by Demuth in his studies of randomness and analysis [15, 16].

Theorem 2.6 (Kučera, Nies [40]; Greenberg, Turetsky [29])**.** *A c.e. set is strongly jump-traceable if and only if it is computable from a Demuth random set.*

Related is the classification of lowness for Demuth randomness [7] and related notions of bases for randomness. We first remark that lowness for Demuth randomness can be charaterised in terms of traceability. An oracle A is low for Demuth randomness if and only if it is both hyperimmune-free and "BLR traceable": for every order h, every function $f \leqslant_{\mathrm{T}} A$ has an ω-computably approximable h-trace T — we can change our mind about the set of values $T(x)$ but the number of changes is bounded by some computable function.

A set A is called a *base for ML-randomness* if it is computable from an A-random sequence. These sets coincide with the K-trivial sets [30]. Using a partial

[2]This showed that Kučera's priority-free solution to Post's problem is K-trivial.

relativisation of Demuth randomness (the bounds on changes in tests remain computable), the same result holds for strong jump-traceability [50, 29]. Below we also discuss a connection between strong jump-traceability and a weak form of Demuth randomness.

The investigations into covering by random sets passes through understanding which random sets compute *all* elements of some class. Relevant here is an unpublished result by Hirschfeldt and J. Miller, extending Kučera's result [38] showing that Δ^0_2 random sets compute noncomputable c.e. sets. Hirschfeldt and Miller showed that if V is a null Σ^0_3 subset of Cantor space, then all random elements of V compute a fixed noncomputable c.e. set (in fact the set can be promptly simple). The solution of the covering problem is a kind of converse to Kučera's result. A converse to two instances of the Hirschfeldt-Miller result is the following:

Theorem 2.7 (Greenberg, Hirschfeldt, Nies [28, 27])**.** *The following are equivalent for a set A:*

(1) *A is computable from all superlow random sets.*
(2) *A is computable from all superhigh random sets.*
(3) *A is strongly jump-traceable.*

We remark that the implication (2) $\Longrightarrow$ (3) for all sets A (rather than only c.e. sets) uses a recent result of Day and Miller's [14]. In [27] it is shown that the implication holds for all sets which are superlow and jump-traceable, in particular, for all K-trivial sets. Day and Miller show that if A is computable from all LR-hard random sequences then it is K-trivial (to do that, they use an effective form of the Lebesgue density theorem). Simpson [51] showed that every LR-hard sequence is superhigh.

A recent result allows us to use Theorem 2.7 to create a new proof of Theorem 2.2. The following is proved using Nies's "golden run" technology:

Lemma 2.8 (Greenberg, Turetsky). For every K-trivial set A, there is a c.e. K-trivial set B such that $A \leqslant_{\mathrm{T}} B$ and the randoms which compute B are precisely those which compute A.

If A is strongly jump-traceable, then by Theorem 2.7 it is computable from all superhigh random sets. As argued in the previous paragraph, A must then be K-trivial. Taking B as in the lemma, B must also be computable from all superhigh random sets, and so by Theorem 2.7 again, B is strongly jump-traceable.

2.3. Applications to degree theory.

The study of strong jump-traceability, originally motivated by questions in algorithmic randomness, has unexpected applications in degree theory.

2.3.1. *Superlow preservation.* Cholak, Groszek and Slaman [12] constructed a low c.e. degree whose join with any low c.e. degree is low. Ng [45] constructed an analogous degree for superlowness. Related is the question about superlow cupping. In [1], Ambos-Spies, Jockusch, Shore and Soare showed that the low-cuppable c.e. degrees were precisely the promptly simple ones. Diamondstone [17] showed that some degree was low-cuppable but not superlow-cuppable.

Theorem 2.9 (Greenberg, Nies [28])**.** *Every strongly jump-traceable degree is* superlow preserving*: if $\mathbf{a}$ is strongly jump-traceable and $\mathbf{b}$ is superlow (not necessarily c.e.) then $\mathbf{a} \vee \mathbf{b}$ is superlow as well.*

Theorem 2.9 implies the results of Ng's and Diamondstone's: the latter, since the construction from [25] produces a promptly simple, strongly jump-traceable c.e. set.

2.3.2. *Pseudojump inversion.* A *pseudojump operator* is an operator which takes a set X to a set $W^X \geqslant_{\mathrm{T}} X$ which is c.e. in X. Psuedojump operators are usually assumed to have the form $W^X = \operatorname{dom} \Phi_e^X \oplus X$ for some e. These have been studied extensively, for example they were used in Jockusch and Shore's definition of the arithmetic degrees [33, 34]. In particular, they introduced the technique of pseudojump inversion, showing for example that if W is a strictly increasing pseudojump operator then $\mathbf{0}'$ contains W^X for some incomplete c.e. set X. Coles, Downey, Jockusch and LaForte [13] first put in print a question which dates back to the 80s: can any strictly increasing pseudojump operator be inverted to a minimal pair? Or merely avoiding upper cones?

The question was answered by Downey and Greenberg. Let W_{SJT} be the operator obtained by relativising the construction from [25]: for any set X it returns a set $W^X_{\mathrm{SJT}} >_{\mathrm{T}} X$ which is strongly jump-traceable relative to X.

Theorem 2.10 (Downey, Greenberg [21])**.** *W_{SJT} cannot be inverted while avoiding upper cones.*

The result is related to so-called *weak reducibilities* (see [2]). The best-known example of such is $\leqslant_{\mathrm{LR}}$, *low-for-random* reducibility, where $A \leqslant_{\mathrm{LR}} B$ if every B-random set is also A-random [47]. The analogue for strong jump-traceability is $\leqslant_{\mathrm{SJT}}$, where $A \leqslant_{\mathrm{SJT}} B$ if A is strongly jump-traceable relative to B but via a partial relativisation which is necessary to make the relation transitive. Namely, traceability is required only with respect to computable rather than B-computable bounds.

A set B is $\leqslant_{\mathrm{SJT}}$-hard if $\varnothing' \leqslant_{\mathrm{SJT}} B$. Every inversion of W_{SJT} is $\leqslant_{\mathrm{SJT}}$-hard. In the proof of Theorem 2.10, Downey and Greenberg in fact showed that there is a noncomputable c.e. set which is computable from all $\leqslant_{\mathrm{SJT}}$-hard c.e. sets. This gives rise to a new ideal in the c.e. degrees, namely those computable from all $\leqslant_{\mathrm{SJT}}$-hard c.e. degrees. Diamondstone, Downey, Greenberg and Turetsky [18] showed that this ideal contains a high degree; this is in contrast with the fact that it cannot contain promptly simple degrees (as there are SJT-hard, cappable c.e. degrees [45, 44]).

3. Techniques

We discuss some technical aspects of working with strongly jump-traceable sets.

3.1. **Box promotion.** The major technique introduced in [11] became known as the "box promotion" method. The basic idea is: (a) to use a trace to gain certification that certain initial segments of a c.e. set are correct; and (b) to amplify progress by considering many inputs at once.

Say A is strongly jump-traceable and c.e.; let $\langle A_s \rangle$ be an enumeration of A. Fix an order-function g. During a construction we observe A's behaviour and at some stage s we wonder if a certain initial segment σ of A_s is *correct* in that $\sigma \prec A$. To do that we define an auxiliary partial A-computable function ψ. Let T be a g-bounded trace for ψ. To test whether $\sigma \prec A$, we pick a "testing location" z (a number) and define $\psi(z) = \sigma$ with use σ. Now at least one of two things must happen: either $\sigma \in T(z)$; or σ is later revealed to be incorrect. In our construction we can wait for one of these events to happen; we believe that σ is correct if the former event happens first.

Of course since A is likely not computable, often we will believe incorrectly. This happens when we first see that $\sigma \in T(z)$ but only later observe that σ is incorrect. While this erroneous belief will incur some cost for us, progress is made. Once we see that σ is incorrect, the testing location z becomes free in that $\psi(z)$ becomes undefined, and we can redefine it later. On the other hand, once enumerated into $T(z)$, σ always remains an element of $T(z)$. Since $|T(z)| \leqslant g(z)$, our opponent has wasted one of their $g(z)$ many "slots" in the "box" $T(z)$. The situation is as if $g(z)$ is now smaller (by 1) compared to its original value. The "best" value is $g(z) = 1$: in this case we know that any answer must be right.

We can amplify this progress by considering aggregates of boxes. Instead of using one testing location z, we use a large finite set Z of testing locations. We test as above by letting $\psi(z) = \sigma$ with use σ for all $z \in Z$. If σ is correct then we know that $\sigma \in T(z)$ for all $z \in Z$, and so we only believe that σ is correct if we see this happen. If we later see that σ is incorrect then all of the boxes $T(z)$ for $z \in Z$ will have been promoted, usually with the same price to us as the promotion of a single box. We can then break up Z into smaller parts and work independently with each one. The benefit from the price paid for a task for one requirement is spread among many.

For example, we consider the argument showing that a c.e. strongly jump-traceable set is K-trivial. Here we must construct a prefix-free machine M and a constant b such that for every n there is a τ with $M(\tau) = A\restriction_n$ and $|\tau| \leqslant K(n) + b$. As we enumerate A we follow the right-c.e. approximation K_s of prefix-free complexity K. At stage s we see a new version of $\sigma = A_s\restriction_n$. Before we believe σ we test it on a large set of locations Z. Only if the test is successful do we choose a τ with $|\tau| = K_s - b$ and define $M(\tau) = \sigma$. The "cost" of this definition is $2^{-K_s(n)-b}$. If σ is later found to be incorrect, this cost is not "refunded" by our machine; the definition remains. By a standard argument, so long as the total cost paid is less than 1, we can arrange that there is always an available τ to select. We organise our testing locations according to the potential costs (rather than say the lengths of the strings described). That is, if $K_s(n) = k$ then the testing for σ is made on inputs from a collection Z_k satisfying $g(z) = k$ for all $z \in Z_k$. If we knew in advance that there is only one length n with $K(n) = k$, then we could let Z_k be a singleton $\{z_k\}$. We would then repeatedly test $A_s\restriction_n$ on the location z_k and know that at most $k = g(z_k)$ many values would be believed. That is, for all k we would waste a quantity of 2^{-k-b} at most k times, and so the total waste would be bounded by $\sum k2^{-k-b}$, which we can arrange to be small by appropriate choice of b.

However, there can be many lengths n with $K(n) = k$, and we need to test strings for each length. We do have a bound on the number of such lengths (for example 2^k). If we make k mistakes on each length the sum would be only bounded by $\sum k2^{-k-b}2^k$ which of course is not finite. We need to ensure that the total number of mistakes made for all of these lengths is say k, so that the sum of our costs remains $\sum k2^{-k-b}$. For this reason we start by testing each length n with $K(n) = k$ on a set $Z_{k,n}$ of size exponential in k. Once one of them $Z_{k,n}$ is promoted (and the amount 2^{-k} spent is wasted), we ignore all the other sets $Z_{k,m}$ for $m \neq n$; we break $Z_{k,n}$ up into 2^k many parts, each taking on itself the role of some $Z_{k,m}$. This can happen at most k times, so we can bound in advance the total number of inputs required (in this example 2^{k^2}, but this can be slightly improved). Achieving such a bound is necessary since we need g to be computable.

3.2. **Inverted box promotion.** A technique related to box promotion was used for the proof of Theorem 2.10. For simplicity, let us consider the weaker result that there is no minimal pair of SJT-hard c.e. sets. Given SJT-hard c.e. sets A and B we enumerate a noncomputable c.e. set E below both of them. Roughly, box promotion is used to generate simultaneous changes in A and B that would allow us to change E for the sake of making it noncomputable. Both sets A and B are enumerating h-traces for a partial Σ^0_2 function ψ that we approximate; again we define the computable order h to be sufficiently slow growing so as to make the combinatorics of the construction work.

Consider a number x which acts as a follower for a Friedberg-Muchnik requirement for E. When it is appointed, we need to choose A- and B-uses for computing $E(x)$. We define a value for $\psi(n)$ for a carefully chosen n, wait for the value to appear in the provided A- and B- traces, and appropriate the uses of these enumerations. Once x is realised and we want to enumerate it into E, we change $\psi(n)$ to some new value. If $h(n) = 1$ (we have a 1-box), then A and B need to remove the old value of $\psi(n)$ from their boxes, meaning that they have to change below the uses. Such simultaneous change would allow us to enumerate x.

Of course the situation gets complicated because we cannot always have 1-boxes. If $h(n) > 1$ then it may be that neither A nor B changes. This is actually good: the boxes tracing $\psi(n)$ have been promoted, and we can appoint a new follower x' for the same Friedberg-Muchnik requirement and start again. Sometimes, however, only one of the sets will change and provide permission; so only one of the traces promotes the box. Say only A changes. In this case we cannot enumerate x, as we do not have a B-permission. But we also need to ensure that $E(x)$ is reducible to A – we cannot keep the A-permission open indefinitely; we need to find a new use for this reduction. In this case we choose some $m < n$, wait for $\psi(m)$ to appear in the appropriate A-trace and take that use. If we later get a B-permission then we use $\psi(m)$ to give us a new B-use. Well-foundedness ensures that eventually we succeed.

The complexity of the construction is in the combinatorics involved in the interaction of followers from different requirements. These can sometimes share boxes. The construction designs the movement of the followers so that m-boxes are available when needed. This of course depends on choosing h appropriately. Another complexity is introduced by the fact that we may not have 1-boxes available (this is because of the use of the recursion theorem to obtain the traces). This is dealt with by letting some permissions remain open indefinitely. We ensure this happens for only finitely many followers. Thus, the reductions can be fixed non-uniformly. The fact that this non-uniformity is benign (only finitely many inputs need to be fixed) is important when we generalise to give the construction proving the full theorem (when we have to deal with infinitely many SJT-hard sets).

3.3. **A golden run of unbounded depth.** A reversal of the techniques used to expolit strong jump-traceability, is a technique to prove that a given set is strongly jump-traceable. As an example, we discuss the proof of the implication $(1) \Longrightarrow (3)$ of Theorem 2.7: if A is computable from all superlow ML-random sequences, then it is strongly jump-traceable. In fact, an amplification of jump-traceability does not use randomness per-se; it applies to all nonempty Π^0_1-classes.

Theorem 3.1 ([27]). *If A is a jump-traceable set computable from all superlow elements of some nonempty Π^0_1-class, then A is strongly jump-traceable.*

The implication (1) $\Longrightarrow$ (3) then follows from using a Π^0_1-class of ML-random sequences. We obtain the assumption that A is jump-traceable by showing that A is K-trivial, as it is computable from both halves of some Δ^0_2 (in fact, superlow) ML-random sequence. We also remark that such a general result is not proved for the direction (2) $\Longrightarrow$ (3); in that more complicated argument, some coding is needed in the Π^0_1-class. This coding is available for Π^0_1-classes of positive measure, but also for Medvedev-complete classes. A result, for example, is that a c.e. set is strongly jump-traceable if and only if it is computable from all superhigh PA-degrees.

The proof of Theorem 3.1 is very much non-uniform. Resembling the decanter and golden run machinery of Nies's, it uses a nested tree of procedures attempting to build an h-trace for J^A, where h is a prescribed order-function. Unlike the golden run argument, there is no a-priori bound on the depth of the nesting required.

Let $\mathcal{P}$ be a nonempty Π^0_1-class as in the assumption of the theorem. Fix an order function h. We show that J^A has a 2^h-bounded c.e. trace, which of course suffices. Paradoxically, what we do is attempt to build a superlow element $Z \in \mathcal{P}$ and try to show that $A \nleqslant_{\mathrm{T}} Z$. If our assumptions hold then this process will fail. The location at which it fails will give us a procedure for certifying initial segments of A and thus of potential computations of J^A.

The first procedure tries to diagonalise against the first Turing reduction Φ_0. Since we are aiming to build a superlow element of $\mathcal{P}$, we start following the proof of the Jockusch-Soare superlow basis theorem [35], which is approximated dynamically. We let $\mathcal{P}^0_0, \mathcal{P}^0_1, \mathcal{P}^0_2, \dots$ be the sequence of classes which is obtained: $\mathcal{P}^0_0 = \mathcal{P}$; $\mathcal{P}^0_{n+1} = \mathcal{P}^0_n$ if $n \in X'$ for all $X \in \mathcal{P}^0_n$; otherwise $\mathcal{P}^0_{n+1}$ is the class of $X \in \mathcal{P}^0_n$ for which $n \notin X'$. Our guess for what $\mathcal{P}^0_n$ is changes during the construction at most 2^n times. To believe a computation $J^A(x)\!\downarrow [s]$, with some use $\tau \prec A_s$, we will want $\tau \preccurlyeq \Phi_0(X)$ for all $X \in \mathcal{P}^0_{h(x)}$. Thus, to certify two different values, we will need two different versions of $\mathcal{P}^0_{h(x)}$; so at most $2^{h(x)}$ values will be certified by this procedure, as required.

Of course it is possible that this first attempt at building a trace for J^A does not succeed. This happens because some correct coputation $J^A(x)$ (with use $\tau \prec A$) is never certified, as not every $X \in \mathcal{P}^0_{h(x)}$ is mapped to τ by Φ_0. But what this means is that we have succeeded in ensuring that $\Phi_0(Z) \neq A$ for the hypothetical Z that we are building. We start again with

$$\mathcal{P}^1_0 = \left\{ X \in \mathcal{P}^0_{h(x)} \,:\, \tau \not\preccurlyeq \Phi_0(X) \right\}$$

which by assumption is nonempty. We then repeat the process, define $\mathcal{P}^1_1, \mathcal{P}^1_2, \dots$ following the superlow basis theorem construction, and certify computations $J^A(x)$ if all oracles in $\mathcal{P}^1_{h(x)}$ map to the use of the computation using the functional Φ_1. Of course, during the construction, we do not know whether the first attempt (using the sequence $\mathcal{P}^0_n$ and Φ_0) is actually successful or not. We see a computation $J^A(x)$, and wait for it to be certified on $\mathcal{P}^0_{h(x)}$. While we wait we need to start the second attempt as described above. That second attempt in turn may spawn a third, fourth, etc. When the computation is certified on $\mathcal{P}^0_{h(x)}$, we cancel all the other attempts and return to the first one.

In the end, we need to argue that if the assumption of the the theorem holds, then some attempt at tracing J^A succeeds: it is never cancelled and every correct computation is eventually certified. We assume not. We then show that for each k, eventually an attempt using Φ_k is started and is never cancelled; and we argue that the intersection of the classes $\mathcal{P}^0_k$ is a singleton $\{Z\}$, and we ensured that $A \not\leqslant_{\mathrm{T}} Z$. The main technical difficulty is in ensuring that Z is in fact superlow: after all, it is not actually produced by a single attempt $\langle \mathcal{P}^k_n \rangle_n$ at the superlow basis theorem. Showing that Z is superlow requires an analysis not only of the eventually stable attempts but also the cancelled ones. Here we use the assumption that A is jump-traceable.

3.4. Cost functions.

Nies introduced cost functions [49] as a means of analytically quantifying the number of changes of a computable approximation of a Δ^0_2 set. The motivation is the implicit cost function construction of a K-trivial set given for example in [23]. A cost function is a computable function $c\colon \omega^2 \to \mathbb{R}^+$. The rational number $c(x, s)$ is the cost of changing our approximation on x at stage s. An approximation $\langle A_s \rangle$ of a set A *obeys* the cost function c if the total cost $\sum_{s<\omega} c(x_s, s)$, where x_s is least such that $A_s(x) \neq A_{s-1}(x)$, is finite. A set obeys c if it has a computable approximation obeying c. The standard example is the cost function for K-triviality, isolating an aspect of the construction from [23]. The cost of changing A_s on x is $c_K(x, s) = \sum_{y>x} 2^{-K_s(y)}$. This is because changing $A_s(x)$, while trying to make it K-trivial, forces us to issue requests for new descriptions of $A_s \upharpoonright_y$ for all $y > x$, and the cost of each such request is $2^{-K_s(y)}$. The construction in [23] can then be presented in a modular fashion. First, defining of the cost function, and showing that any set that obeys the cost function c_K is K-trivial. And separately, a general lemma that says that under reasonable assumptions on a cost function c (which c_K manifestly satisfies), there is a noncomputable c.e. set which obeys c.

Similarly, several theorems regarding strongly jump-traceable sets can be understood via cost functions. For example, the proof above can be modified to show that every strongly jump-traceable set obeys the cost function c_K. Another example is computability from random sets (Theorem 2.7). As mentioned above, Kučera showed that any Δ^0_2 random set computes a noncomputable c.e. set. His argument too can be factored through cost functions. Given a Δ^0_2 random set Y one can produce a cost function c_Y, any set satisfying which is computable from Y. One then shows using a box-promotion argument that if Y is superlow then every strongly jump-traceable c.e. set obeys c_Y.

This in fact leads to a characterisation of strong jump-traceability using cost functions, much like the fact that K-trivial c.e. sets are characterised by obeying the cost function c_K. Since the set $\{e : W_e \text{ is strong jump-traceable}\}$ is Π^0_4-complete [43], while the set $\{e : W_e \text{ obeys } c\}$ can be easily seen to be Σ^0_3 for any cost function c, one cost function will not suffice. Greenberg and Nies [28] isolated a property of cost functions necessary to characterise strong jump traceability.

Definition 3.2. A cost function c is *benign* if there is a computable function $g : \omega \to \omega$ such that any sequence $x_0 < x_1 < x_2 < \dots$ satisfying $c(x_i, x_{i+1}) > 2^{-k}$ for all i has length at most $g(k)$.

The intuition is the following: suppose one is performing a dynamic construction of a c.e. set, and this construction includes a strategy which selects a witness x, waits for some event to possibly occur, and then enumerates x into the set if it happens. However, suppose further that this strategy is forbidden from enumerating an element if the cost of doing so is greater than 2^{-k}; in this case, it must discard its witness and select a new large witness to try again with. Then the number of times the strategy might need to select a new witness is bounded by $g(k)$.

The general form of the box-promotion argument given above shows that a c.e. set is strongly jump-traceable if and only if it obeys all benign cost functions. One then shows that the cost functions c_K and c_Y discussed above are benign.

Finally, the original proof of Theorem 2.2 makes essential use of benign cost functions. In [19] the authors first show that every strongly jump-traceable set, whether c.e. or not, obeys all benign cost functions. Then they show that there is a special benign cost function c^* with the property: if A is a set that obeys c^*, then there is a c.e. set $C \geqslant_{\mathrm{T}} A$ which obeys all cost functions obeyed by A.

3.4.1. *Cost functions and tests.* An alternative use of cost functions, introduced in [8], is as a bound on the rate at which the measure of the elements of a weak 2 test shrinks to zero.

Definition 3.3. For a cost function c, a *c-test* is a uniformly Σ^0_1 sequence of sets $\langle U_n \rangle_{n\in\omega}$ such that $\lambda(U_{n,s}) < b \cdot c(n, s)$ for some constant b.

A real is *captured* by a c-test if it is in the intersection of the sequence of sets.

The following result, which is basically a restatement of the Hirschfeldt-Miller result, connects the two uses of cost functions.

Lemma 3.4 ([8]). If A obeys some cost function c, then every random captured by a c-test computes A.

Restricting our attention to benign cost functions, we obtain precisely the weak Demuth tests. Here we recall that a weak Demuth test is a nested sequence $\langle U_n \rangle$ of effectively open sets with $\lambda(U_n) \leqslant 2^{-n}$ but whose index is not given effectively, but rather is computably approximated, with a computable bound on the number of mind-changes. That is, $U_n = W_{f(n)}$ where $\langle W_e \rangle$ is an effective enumeration of all Σ^0_1 classes, and f is ω-c.a. In contrast, (non-weak) Demuth test are not required to be nested, and so are used with a solovay capturing condition.

Lemma 3.5. For every weak Demuth test $\langle U_n \rangle_{n\in\omega}$ there is a benign cost function c and a c-test $\langle V_n \rangle_{n\in\omega}$ such that $\bigcap_n U_n \subseteq \bigcap_n V_n$. Conversely, for every benign cost function c and every c-test $\langle V_n \rangle_{n\in\omega}$, there is a weak Demuth test $\langle U_n \rangle_{n\in\omega}$ with $\bigcap_n V_n \subseteq \bigcap_n U_n$.

By combining these results with the cost function characterization of strongly jump-traceable sets, the following is obtained.

Theorem 3.6. *Every random which is not weakly Demuth random computes every strongly jump-traceable set.*

Proof. Suppose A is strongly jump-traceable and X is random but not weakly Demuth random. Then X is captured by some weak Demuth test, and thus by some c-test for some benign cost function c. A obeys c, and thus X computes A. □

3.5. **Strong jump-traceability and K-triviality.** We summarise some analogous properties of these two notions.

	K-trivial	SJT
Structure	c.e.-generated ideal	c.e.-generated ideal
Index set	Σ^0_3	Π^0_4
Cost functions	Additive	Benign
Random covering	Incomplete	Demuth
Base for randomness	Martin-Löf	$\text{Demuth}_{\text{BLR}}$

4. Open problems

We end this paper by describing some open problems.

4.1. **Computing with randoms.** As mentioned above, the work used for solving the covering problem for K-trivial sets gives a characterisation of the ML-random sets which compute all K-trivial sets. An analogue for strong jump-traceability is not known. The converse of Theorem 3.6 is consistent with current knowledge.

This question appears to be more difficult than the one for K-triviality. This is again because of the complexity of the ideals. The work in [8] shows that there is a "smartest" K-trivial: a K-trivial set A with the property that any random set computing A must compute all K-trivial sets. We strongly suspect the analogue does not hold from strong jump-traceability. This is also indicated by Theorem 2.6.

4.2. **Superlow preservation.** The converse to Theorem 2.9 is not known. That is, it is consistent with current knowledge that a set is strongly jump traceable if and only if it is superlow-preserving.

4.3. **SJT-hard sets.** The ideal I of c.e. degrees which lie below all $\leqslant_{\text{SJT}}$-hard c.e. degrees remains poorly understood. Significantly, we do not know how to show that it is not principal. One possible approach is to consider superhighness. While I contains a high degree, it is not known if it contains a superhigh degree. If it contains an h-superhigh degree for all order functions h then I is not principal.

Related is a question about $\leqslant_{\text{LR}}$-hardness and minimal pairs. The notion of $\leqslant_{\text{LR}}$-hardness first gained interest since it is equivalent to almost everywhere domination [36]. Barmpalias and Montalbán showed that there is an $\leqslant_{\text{LR}}$-hard c.e. degree which is half of a minimal pair [4]. Several researchers asked if there is a minimal pair of $\leqslant_{\text{LR}}$-hard c.e. degrees. This is related to the ideal I via the connection between K-triviality and tracing. In [21] it is shown that there is an order function h such that there is no minimal pair of h-superhigh sets. Further work by Turetsky showed that h can be taken to be a constant multiple of the logarithm function. Results from [3] show that if this could be improved to $h = \frac{1}{10}\log(n)$, the question about minimal pairs of $\leqslant_{\mathrm{LR}}$-hard degrees would be answered in the negative.

4.4. **Strong reducibilities.** While every strongly jump-traceable set is K-trivial, it is unknown if the binary relation $\leqslant_{\mathrm{SJT}}$ implies $\leqslant_{\mathrm{LR}}$. This is known to hold with the additional assumption of Turing reducibility: if $A \leqslant_{\mathrm{SJT}} B$ and $B \leqslant_T A$, then $A \leqslant_{\mathrm{LR}} B$, simply by relativizing the proof that strongly jump-traceable sets are K-trivial. However, this proof seems to be inherently dynamic: a construction is performed relative to B, and the proof relies on the fact that A can compute the stages of the construction, which allows A to use the trace (relative to B) to define the function to be traced. An affirmative answer to the general question would likely amount to a "static" proof of the unrelativised result: an argument which does not use the trace for defining the partial function being traced.

References

[1] Klaus Ambos-Spies, Carl G. Jockusch, Jr., Richard A. Shore, and Robert I. Soare. An algebraic decomposition of the recursively enumerable degrees and the coincidence of several degree classes with the promptly simple degrees. *Trans. Amer. Math. Soc.*, 281(1):109–128, 1984.

[2] George Barmpalias. Algorithmic randomness and measures of complexity. *Bull. Symbolic Logic*, 19(3):318–350, 2013.

[3] George Barmpalias, Rod Downey, and Noam Greenberg. K-trivial degrees and the jump-traceability hierarchy. *Proc. Amer. Math. Soc.*, 137(6):2099–2109, 2009.

[4] George Barmpalias and Antonio Montalbán. A cappable almost everywhere dominating computably enumerable degree. In *Proceedings of the Third International Conference on Computability and Complexity in Analysis (CCA 2006)*, volume 167 of *Electron. Notes Theor. Comput. Sci.*, pages 17–31 (electronic). Elsevier, Amsterdam, 2007.

[5] Tomek Bartoszyński. Additivity of measure implies additivity of category. *Trans. Amer. Math. Soc.*, 281(1):209–213, 1984.

[6] 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(1):80–90, 2014.

[7] Laurent Bienvenu, Rod Downey, Noam Greenberg, André Nies, and Dan Turetsky. Characterizing lowness for Demuth randomness. *J. Symb. Log.*, 79(2):526–560, 2014.

[8] Laurent Bienvenu, Noam Greenberg, Antoní n Ku ˇ cera, André Nies, and Dan Turetsky. Coherent randomness tests and computing the K-trivial sets. *J. Eur. Math. Soc. (JEMS)*, 18(4):773–812, 2016.

[9] Gregory J. Chaitin. Information-theoretic characterizations of recursive infinite strings. *Theoret. Comput. Sci.*, 2(1):45–48, 1976.

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

[11] Peter Cholak, Rodney G. Downey, and Noam Greenberg. Strong jump-traceabilty I: The computably enumerable case. *Adv. Math.*, 217(5):2045–2074, 2008.

[12] Peter Cholak, Marcia Groszek, and Theodore Slaman. An almost deep degree. *J. Symbolic Logic*, 66(2):881–901, 2001.

[13] Richard J. Coles, Rodney G. Downey, Carl G. Jockusch, Jr., and Geoffrey Laforte. Completing pseudojump operators. *Ann. Pure Appl. Logic*, 136(3):297–333, 2005.

[14] Adam R. Day and Joseph S. Miller. Density, forcing, and the covering problem. *Math. Res. Lett.*, 22(3):719–727, 2015.

[15] Osvald Demuth. Some classes of arithmetical real numbers. *Comment. Math. Univ. Carolin.*, 23(3):453–465, 1982.

[16] Osvald Demuth. Remarks on the structure of tt-degrees based on constructive measure theory. *Comment. Math. Univ. Carolin.*, 29(2):233–247, 1988.

[17] David Diamondstone. Promptness does not imply superlow cuppability. *J. Symbolic Logic*, 74(4):1264–1272, 2009.

[18] David Diamondstone, Rod G. Downey, Noam Greenberg, and Daniel D. Turetsky. High degrees computable from all sjt-hard degrees. In preparation.

[19] David Diamondstone, Noam Greenberg, and Daniel D. Turetsky. Inherent enumerability of strong jump-traceability. *Trans. Amer. Math. Soc.*, 367(3):1771–1796, 2015.

[20] Rod Downey and Noam Greenberg. Strong jump-traceability II: K-triviality. *Israel J. Math.*, 191(2):647–665, 2012.

[21] Rod Downey and Noam Greenberg. Pseudo-jump inversion, upper cone avoidance, and strong jump-traceability. *Adv. Math.*, 237:252–285, 2013.

[22] Rod Downey, Carl G. Jockusch, and Michael Stob. Array nonrecursive degrees and genericity. In *Computability, enumerability, unsolvability*, volume 224 of *London Math. Soc. Lecture Note Ser.*, pages 93–104. Cambridge Univ. Press, Cambridge, 1996.

[23] Rodney G. Downey, Denis R. Hirschfeldt, André Nies, and Frank Stephan. Trivial reals. In *Proceedings of the 7th and 8th Asian Logic Conferences*, pages 103–131, Singapore, 2003. Singapore Univ. Press.

[24] Rodney G. Downey, Carl G. Jockusch, Jr., and Michael Stob. Array nonrecursive sets and multiple permitting arguments. In *Recursion theory week (Oberwolfach, 1989)*, volume 1432 of *Lecture Notes in Math.*, pages 141–173. Springer, Berlin, 1990.

[25] Santiago Figueira, André Nies, and Frank Stephan. Lowness properties and approximations of the jump. *Ann. Pure Appl. Logic*, 152(1-3):51–66, 2008.

[26] Johanna N. Y. Franklin, Noam Greenberg, Frank Stephan, and Guohua Wu. Anti-complex sets and reducibilities with tiny use. *J. Symbolic Logic*, 78(4):1307–1327, 2013.

[27] Noam Greenberg, Denis R. Hirschfeldt, and André Nies. Characterizing the strongly jump-traceable sets via randomness. *Adv. Math.*, 231(3-4):2252–2293, 2012.

[28] Noam Greenberg and André Nies. Benign cost functions and lowness properties. *J. Symbolic Logic*, 76(1):289–312, 2011.

[29] Noam Greenberg and Daniel D. Turetsky. Strong jump-traceability and Demuth randomness. *Proc. Lond. Math. Soc. (3)*, 108(3):738–779, 2014.

[30] Denis R. Hirschfeldt, André Nies, and Frank Stephan. Using random sets as oracles. *J. Lond. Math. Soc. (2)*, 75(3):610–622, 2007.

[31] Rupert Hölzl, Thorsten Kräling, and Wolfgang Merkle. Time-bounded Kolmogorov complexity and Solovay functions. In *Mathematical foundations of computer science 2009*, volume 5734 of *Lecture Notes in Comput. Sci.*, pages 392–402. Springer, Berlin, 2009.

[32] Shamil Ishmukhametov. Weak recursive degrees and a problem of Spector. In *Recursion theory and complexity (Kazan, 1997)*, volume 2 of *de Gruyter Ser. Log. Appl.*, pages 81–87. de Gruyter, Berlin, 1999.

[33] Carl G. Jockusch, Jr. and Richard A. Shore. Pseudojump operators. I. The r.e. case. *Trans. Amer. Math. Soc.*, 275(2):599–609, 1983.

[34] Carl G. Jockusch, Jr. and Richard A. Shore. Pseudojump operators. II. Transfinite iterations, hierarchies and minimal covers. *J. Symbolic Logic*, 49(4):1205–1236, 1984.

[35] Carl G. Jockusch, Jr. and Robert I. Soare. Degrees of members of Π^0_1 classes. *Pacific J. Math.*, 40:605–616, 1972.

[36] Bjørn Kjos-Hanssen, Joseph S. Miller, and Reed Solomon. Lowness notions, measure and domination. *J. Lond. Math. Soc. (2)*, 85(3):869–888, 2012.

[37] Bjørn Kjos-Hanssen, André Nies, and Frank Stephan. Lowness for the class of Schnorr random reals. *SIAM J. Comput.*, 35(3):647–657 (electronic), 2005.

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

[39] Antonín Kučera. On relative randomness. *Ann. Pure Appl. Logic*, 63(1):61–67, 1993. 9th International Congress of Logic, Methodology and Philosophy of Science (Uppsala, 1991).

[40] Antonín Kučera and André Nies. Demuth randomness and computational complexity. *Ann. Pure Appl. Logic*, 162(7):504–513, 2011.

[41] Joseph S. Miller and André Nies. Randomness and computability: open questions. *Bull. Symbolic Logic*, 12(3):390–410, 2006.

[42] Webb Miller and Donald A. Martin. The degrees of hyperimmune sets. *Z. Math. Logik Grundlagen Math.*, 14:159–166, 1968.

[43] Keng Meng Ng. On strongly jump traceable reals. *Ann. Pure Appl. Logic*, 154(1):51–69, 2008.

[44] Keng Meng Ng. On very high degrees. *J. Symbolic Logic*, 73(1):309–342, 2008.

[45] Keng Meng Ng. *Computability, traceability and beyond*. PhD thesis, Victoria University of Wellington, 2009.
[46] Keng Meng Ng. Beyond strong jump traceability. *Proc. Lond. Math. Soc. (3)*, 102(3):423–467, 2011.
[47] André Nies. Lowness properties and randomness. *Adv. Math.*, 197(1):274–305, 2005.
[48] André Nies. Eliminating concepts. In *Computational prospects of infinity. Part II. Presented talks*, volume 15 of *Lect. Notes Ser. Inst. Math. Sci. Natl. Univ. Singap.*, pages 225–247. World Sci. Publ., Hackensack, NJ, 2008.
[49] André Nies. *Computability and randomness*, volume 51 of *Oxford Logic Guides*. Oxford University Press, Oxford, 2009.
[50] André Nies. Computably enumerable sets below random sets. *Ann. Pure Appl. Logic*, 163(11):1596–1610, 2012.
[51] Stephen G. Simpson. Almost everywhere domination and superhighness. *MLQ Math. Log. Q.*, 53(4-5):462–482, 2007.
[52] 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.
[53] Sebastiaan A. Terwijn. *Computability and Measure*. PhD thesis, University of Amsterdam, 1998.
[54] Sebastiaan A. Terwijn and Domenico Zambella. Computational randomness and lowness. *J. Symbolic Logic*, 66(3):1199–1205, 2001.
[55] Daniel Turetsky. A K-trivial set which is not jump traceable at certain orders. *Inform. Process. Lett.*, 112(13):544–547, 2012.

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

Department of Mathematics, Victoria University of Wellington, Wellington, New Zealand
E-mail address: dan.turetsky@vuw.ac.nz
URL: http://tinyurl.com/dturetsky