PROCEEDINGS OF THE
AMERICAN MATHEMATICAL SOCIETY
Volume 137, Number 6, June 2009, Pages 2099–2109
S 0002-9939(09)09761-5
Article electronically published on January 22, 2009

K-TRIVIAL DEGREES AND THE JUMP-TRACEABILITY HIERARCHY

GEORGE BARMPALIAS, ROD DOWNEY, AND NOAM GREENBERG

(Communicated by Julia Knight)

ABSTRACT. For every order h such that $\sum_n 1/h(n)$ is finite, every K-trivial degree is h-jump-traceable. This motivated Cholak, Downey and Greenberg to ask whether this traceability property is actually equivalent to K-triviality, thereby giving the hoped for combinatorial characterisation of lowness for Martin-Löf randomness. We show however that the K-trivial degrees are properly contained in those that are h-jump-traceable for every convergent order h.

1. INTRODUCTION

An important question in algorithmic randomness is whether there is a combinatorial characterisation of the class of K-trivial reals (equivalently, of the Turing degrees of such reals). This important class was shown by Nies and his co-authors (see [10, 12]) to coincide with a number of lowness notions such as lowness for Martin-Löf randomness and lowness for K. However, all known definitions of this class involve effective randomness or measure. It is still hoped that new light will be shed on this class by finding a measure-free definition, one which involves only standard notions of computability. We refer to such a definition as "combinatorial".

The main example of a combinatorial characterisation of a class which was first defined by randomness and lowness is the class of reals which are low for Schnorr randomness (for a general introduction to randomness notions and lowness, see the texts [4, 12]). In a pair of papers, Terwijn and Zambella [14] and then Kjos-Hanssen, Nies and Stephan [7] showed that a degree is low for Schnorr randomness if and only if it is computably traceable.

Traceability is a notion which was borrowed from set theory by Terwijn and Zambella. In the context of computability, we say that a partial function $\psi\colon \omega \to \omega$ is *traced* by a sequence of finite sets $\langle T_n \rangle$ of natural numbers if for every $n \in \operatorname{dom} \psi$ we have $\psi(n) \in T_n$. If the sequence $\langle T_n \rangle$ is given effectively, then we can think of ψ as being "weakly computable"; it may not quite be computable, but for every n, there are only finitely many possibilities for $\psi(n)$, which are given effectively. The more effective the trace $\langle T_n \rangle$, the closer ψ is to being computable: for the notion above we assume that the list $\langle T_n \rangle$ is computable (that is, canonical indices for

Received by the editors March 20, 2008, and, in revised form, September 11, 2008.
2000 *Mathematics Subject Classification.* Primary 03D80.
Key words and phrases. K-triviality, Turing degrees, jump traceability, superlinear orders.
All authors were supported by the Marsden Fund of New Zealand.

finite sets are given effectively). A standard weakening is that of c.e. traceability; here the trace is a *c.e. trace*, which means that the sequence $\langle T_n \rangle$ is uniformly c.e.

In the search for a combinatorial definition of the K-trivial degrees (see [8, 11]), Figueira, Nies and Stephan [5] modified Nies's class of jump-traceable reals [9] and introduced the notion of *strong jump-traceability*. We give the definitions.

Definition 1.1. (1) An *order* is a computable, non-decreasing and unbounded function $h\colon \omega \to \omega \setminus \{0\}$.
(2) A trace $\langle T_n \rangle$ *obeys* an order h if for all n, $|T_n| \leqslant h(n)$.
(3) A real $A \in 2^\omega$ is *jump-traceable* if the universal A-partial computable function J^A is traceable by some c.e. trace which obeys some order.
(4) A real $A \in 2^\omega$ is *strongly jump-traceable* if for every order h, J^A is traceable by some trace which obeys h.

The point here was that unlike the previously used notions of computable traceability and c.e. traceability, when tracing partial functions, the orders which bound the size of the traces do make a difference. For instance, there are $2^{\aleph_0}$ many jump-traceable reals, but by a result of Downey and Greenberg [3], there are only countably many strongly jump-traceable reals; indeed they are all Δ^0_2.

All K-trivial reals are jump-traceable (Nies [9]). Cholak, Downey and Greenberg [2], however, showed that while all c.e. strongly jump-traceable reals are K-trivial, there are K-trivial reals which are not strongly jump-traceable, so this attempt at giving a measure-free characterisation of K-triviality fails.

We have mentioned, though, that (at least among c.e. sets, which is not a restriction when dealing with K-triviality), the K-trivial reals lie somewhere between strong and plain jump-traceability (for c.e. degrees, jump-traceability is equivalent to superlowness). So possibly a less naive approach would be to investigate *for which orders h* are K-trivial reals h-jump traceable. For this to be meaningful, we first need to define this notion.

One is tempted to make the following definition: a real $A \in 2^\omega$ is h-jump-traceable if J^A is traceable by a trace which obeys h. Indeed some authors have taken this approach and have interesting results; for example, Ng [13] showed that there are two c.e. sets A_0 and A_1 which join up to $\emptyset'$ and such that J^{A_i} can be traced by a trace which obeys the identity function.

The drawback of this definition, though, is that it depends on the particular choice of universal function J, and so is not degree invariant, let alone downward closed in the Turing degrees, as is expected from a notion of computational weakness such as traceability. We suggest a degree-invariant definition.

Definition 1.2. Let h be an order. A Turing degree $\mathbf{a}$ is *h-jump-traceable* if every function which is $\mathbf{a}$-partial computable has a c.e. trace which obeys h.

We mention that Ng's construction cannot be used to construct id-jump-traceable degrees (this would imply our main result).

Now what is the relationship between K-triviality and h-jump-traceability for various orders h? As mentioned above, Nies showed that all K-trivial reals are jump-traceable; indeed they are all $n \log^2 n$-jump-traceable (see [12]). We improve this result in Theorem 1.3 below. On the other hand, Cholak, Downey and Greenberg [2] showed that every c.e. degree which is $\sqrt{\log n}/9$-jump-traceable is K-trivial (again see [12]).

The decanter argument which shows that K-triviality implies superlowness actually yields the following.

Theorem 1.3. *Let h be an order such that $\sum_n 1/h(n)$ is finite. Then every K-trivial degree is h-jump-traceable.*

Proof. Let A be K-trivial and let Φ be a Turing functional.

Let k_0 be such that $\sum_{n>k_0} 1/f(n) < 1$ and let (q_i) be a computable sequence of finite binary rationals such that $q_i > 1/f(i)$ for all $i \in \mathbb{N}$ and $\sum_{n>k_0} q_i < 1$. For each $n > k_0$ define a prefix-free set of strings C_n such that $\mu(C_i) = q_i$ and $[C_i] \cap [C_j] = \emptyset$ for all $i \neq j$ with $i, j > k_0$. Define a prefix-free machine M as follows: for each $j > k_0$, $s \in \mathbb{N}$ and $\sigma \in C_j$ let $M^A(\sigma)[s] \downarrow$ iff $\Phi^A(j)[s] \downarrow$. By a lemma in [10, 12] (an application of the golden run construction) there is a computable increasing function g such that

$$\sum_{x,r} \{c(x,r) \mid x \text{ is least s. t. } A \upharpoonright g(r+1) \neq A \upharpoonright g(r+2)\ \} < 1, \tag{1.1}$$

where

$$c(x,r) = \sum \{2^{-|\sigma|} \mid M^A(\sigma)[g(r+1)] \downarrow\ \wedge\ x < \text{use}(M^A(\sigma)[g(r+1)]) < g(r)\}.$$

Now we can enumerate a trace (T_i) for Φ^A by proceeding as follows for each $r \in \mathbb{N}$: at stage $g(r+1)$ for each $i < g(r+1)$, if $\Phi^A(i) \downarrow$ with use $< g(r)$, enumerate $\Phi^A(i)$ into T_i. By (1.1) for each $i > k_0$ we have $(|T_i| - 1) \cdot q_i < 1$; hence $|T_i| \leq f(i)$. □

Notice that the function $n \log^2 n$ satisfies the condition of Theorem 1.3. This observation prompted Cholak, Downey and Greenberg to ask whether this level of jump-traceability actually provided the sought-after combinatorial definition of K-triviality. We show that it doesn't.

Theorem 1.4. *There is a (c.e.) degree* **a** *which is h-jump-traceable for every order h such that $\sum_n 1/h(n)$ is finite but is not K-trivial.*

The convergence of $\sum 1/h(n)$ is not a particularly easy assumption to work with. We instead work with a wider class of orders, the superlinear ones.

Definition 1.5. An order h is called *superlinear* if $\lim_n h(n)/n = \infty$.

Lemma 1.6. *If h is an order and $\sum_n 1/h(n) < \infty$, then h is superlinear.*

Proof. Suppose that $\lim_s \frac{h(n)}{n} = \infty$ does not hold. Then there exists some $c \in \mathbb{N}$ such that for infinitely many n we have $h(n) < c \cdot n$. So there exists a sequence $\langle n_k \rangle$ such that for all k, $h(n_k) < c \cdot n_k$ and $n_{k+1} > 2n_k$ (so $1 - n_k/n_{k+1} \geqslant 1/2$). Since h is non-decreasing, for all $x \in (n_{k-1}, n_k]$ we have

$$\frac{1}{c \cdot n_k} \leqslant \frac{1}{h(n_k)} \leqslant \frac{1}{h(x)}$$

and so

$$\sum_x \frac{1}{h(x)} = \sum_k \sum_{x=n_{k-1}+1}^{n_k} \frac{1}{h(x)} \geqslant \sum_k \frac{n_k - n_{k-1}}{c \cdot n_k} \geqslant \frac{1}{c} \sum_k \frac{1}{2} = \infty. \qquad \square$$

Hence, instead of proving Theorem 1.4 directly, we prove the following:

Theorem 1.7. *There is a (c.e.)* **a** *which is not K-trivial but which is h-jump-traceable for every superlinear order h.*

This still leaves us wondering whether some other level of the jump-traceability hierarchy might characterise K-triviality. Orders h such that $\sum 2^{-h(n)}$ is finite may play a role. A particular test question is the following: is every K-trivial degree id-jump-traceable?

2. Proof of Theorem 1.7

2.1. **The individual requirements.** We enumerate a c.e. set A which will have the required degree.

Making A not K-trivial. By the equivalence of K-triviality and lowness for randomness, it is sufficient to make A not low for ML randomness. In [6] (see [1] for a different proof) it was shown that A is low for ML-randomness iff every $\Pi^0_1(A)$ class of positive measure contains an unrelativised Π^0_1 class of positive measure; equivalently, every $\Sigma^0_1(A)$ class G^A such that $\mu(G^A) < 1$ is contained by some unrelativised Σ^0_1 class V such that $\mu(V) < 1$ (here μ denotes the standard Lebesgue (or fair coin) measure on the Cantor space 2^ω). Hence we build a $\Sigma^0_1(A)$ class G^A and meet the requirement P, which states that G^A is not contained in any Σ^0_1 class of measure less than 1 (in fact, we can modify the construction to ensure that if V is any Σ^0_1 class which contains G^A, then $V = 2^\omega$).

To achieve P, we enumerate all Σ^0_1 classes of measure less than 1 effectively as $\langle V_j \rangle_{j\in\mathbb{N}}$; for each j we enumerate a $\Sigma^0_1(A)$ class G^A_j and ensure that $\sum_j \mu(G^A_j) < 1$. The requirement is

$$P_j\colon G^A_j \not\subseteq V_j.$$

So if we let $G^A = \bigcup_j G^A_j$, then the requirement P is met.

The strategy for meeting P_j, if there are no restraints around, is very simple. Suppose that we want to ensure that $\mu(G^A_j) \leqslant q_j$, where q_j is a binary rational (if, for example, we let $q_j = 2^{-(j+2)}$, then the requirement $\sum_j \mu(G^A_j) < 1$ will be satisfied). We partition the space 2^ω into $1/q_j$ many clopen sets $C_1, C_2, \ldots, C_{1/q_j}$. In order to do so, we first enumerate C_k into G^V_j with some (large) use $u = u_k$. We then wait for C_k to appear completely in V_j (by compactness of C_k, if $C_k \subseteq V_j$, then there is a finite stage s at which $C_k \subseteq V_j[s]$). If this never happens, then we keep waiting (and don't do anything else); in this case we'll have $G^A_j = C_k$, so $\mu(G^A_j) = q_j$ as required and $G^A_j \not\subseteq V_j$, so P_j is met. If, on the other hand, we see at some stage that $C_k \subseteq V_j$, then we take C_k out of G^A_j (by enumerating u into A) and move on to C_{k+1}. We can never get to $k = 2^n$, for otherwise $V_j = 2^\omega$ and certainly doesn't have measure less than 1.

Note that if P_j decides to work with clopen pieces of size q_j, then it may end up enumerating $1/q_j$ many numbers into A.

Making A h-jump-traceable. Let $\langle \Phi_e, h_e \rangle$ be an effective enumeration of all pairs consisting of a functional and a partial computable, non-decreasing function from $\omega \to \omega \setminus \{0\}$ whose domain is an initial segment of ω. For each e we need to build a c.e. trace $\langle T^e_i \rangle_{i\in\mathbb{N}}$ which obeys h_e and meets the requirement

Q_e**:** If h_e is a superlinear order, then $\langle T^e_i \rangle$ traces Φ^A_e.

The strategy for meeting Q_e is also straightforward: whenever a computation $\Phi^A_e(i)$ is discovered, we impose a restraint on A to preserve this computation and enumerate the value in T^e_i.

The conflict between the P and Q requirements is now clear: the P requirements need to enumerate numbers into A in order to keep G_j^A small, whereas the Q requirements need to restrain A from changing in order to keep T_i^e small.

2.2. **The private priority list; restraint on P_j.** Here we describe the interaction between a single requirement P_j which has to respect a single requirement Q_e; we assume that h_e is a total, superlinear order. To understand the situation better, we realise that Q_e can be broken up into infinitely many subrequirements:

$Q_{e,i}$**:** $|T_i^e| \leqslant h_e(i)$; and if $\Phi_e^A(i)\downarrow$, then $\Phi_e^A(i) \in T_i^e$.

We imagine all of the subrequirements $Q_{e,i}$ being ordered (naturally) in a "private" priority list. Somewhere in this list, we need to place P_j:

$$Q_{e,0}, \ldots, Q_{e,k-1}, P_j, Q_{e,k}, Q_{e,k+1}, \ldots.$$

P_j acts as we described before, except that it has to respect the restraints imposed by $Q_{e,0}, \ldots, Q_{e,k-1}$. Each one of these subrequirements acts at most once, because they can never be injured. When P_j is restrained, it may not be able to take out a clopen set from G_j^A as the A-use used in enumerating that piece into G_j^A may be smaller than the use of the computation $\Phi_e^A(i)$ which is being protected. In that case, P_j is forced to leave the clopen piece in G_j^A and move on to the next piece. Now note that if $Q_{e,i}$ (for some $i < k$) prevents P_j from extracting some clopen piece C from G_j^A, then the A-uses which P_j picks later are greater than the use of the computation $\Phi_e^A(i)$, which as mentioned above is never injured. Hence $Q_{e,i}$ will not prevent P_j from extracting a different piece C' from G_j^A, so the total number of clopen pieces that P_j may be prevented from extracting from G_j^A is at most k. So if P_j needs to keep the measure of G_j^A below q_j, then it needs to work with clopen pieces of size q_j/k (we may assume that k is a power of 2).

As analysed above, this means that P_j may make enumerate up to k/q_j many numbers into A. Each such enumeration may injure any of the subrequirements $Q_{e,k}, Q_{e,k+1}, \ldots$ by removing a computation $\Phi_e^A(i)$ whose value was already enumerated into T_i^e. To keep these subrequirements happy, we need to ensure that they are allowed to make T_i^e have a size greater than k/q_j, which, since h_e is non-decreasing, is equivalent to requiring that $h_e(k) > k/q_j$.

The existence of such a number k, indeed cofinitely many such numbers, is guaranteed by the assumption that h_e is superlinear. So in this simple case, the tension can be resolved.

2.3. **Initialisations.** Let us now see how one Q_e interacts with all of the requirements P_i. Again we assume that h_e is a total, superlinear order and break Q_e into subrequirements; then we place all of the P_i on Q_e's private priority list:

$$Q_{e,0}, \ldots, Q_{e,k_0-1}, P_0, Q_{e,k_0}, \ldots, Q_{e,k_1-1}, P_1, Q_{e,k_1}, Q_{e,k_1+1}, \ldots.$$

Suppose that the quota for requirement P_j (that is, the bound required on the measure of G_j^A) is q_j.

There are three ways to deal with this situation.

The naive (or brute force) approach proceeds as in the previous section. We require that $h_e(k_0) > k_0/q_0$. For each $i \in [k_0, k_1)$ we see that $Q_{e,i}$ will protect at most k_0/q_0 many computations, and so P_1 will be restrained something like $k_1 k_0/q_0$ many times, so we require something like $h_e(k_1) > k_0 k_1/(q_0 q_1)$, which is again possible by superlinearity. This approach yields terribly complicated inequalities.

An improvement on the naive approach is obtained by observing that, in fact, each P_j is restrained at most k_j many times. The reason is that if $j' < j$, then at any given time, the A-use used by $P_{j'}$ is smaller than the A-use used by P_j, and so any action by $P_{j'}$ already removes from G_j^A the current clopen piece. If $i < k_j$ and $Q_{e,i}$ permanently restrains P_j from extracting a piece from G_j^A, then only this piece can be "blamed" on $Q_{e,i}$. This is because the computation $\Phi_e^A(i)$ of $Q_{e,i}$ can only be injured by some $P_{j'}$ for some $j' < j$, hereby extracting that very clopen piece from G_j^A.

The reason we would rather adopt the third approach, which is initialising P_j and moving the locations k_j along the private priority list, is that it is necessary for meeting a different task entirely, namely injuring P-nodes by Q-nodes which are lexicographically left of those P-nodes in the tree of strategies we will use in the full construction. Since we have to use this blunt tool, it can subsume the gentler approaches for the issue in hand.

The main idea is this. When we choose to place P_j at location k_j of Q_e's private list, we assume that all the stronger $P_{j'}$ (for $j' < j$) have already ceased all action. Under this assumption, no subrequirement $Q_{e,i}$ (for $i < k_j$) can be later injured, and so, by the arguments from Section 2.2, subrequirements can only be responsible for at most one clopen piece remaining stuck in G_j^A. It follows that we can let P_j use pieces of size q_j/k_j.

Of course, this assumption may be wrong. Thus if some $P_{j'}$ (where $j' < j$) acts later, we initialise P_j. This has some consequences. The first is that while we can intialise the requirement P_j (and let it start with a fresh G_j^A), we cannot initialise G_j^A itself; after all, we are building a global set G^A which must be open. So before declaring the "new" G_j^A to be empty, we need to "pour" the old G_j^A into the global G^A. This means that we need to shrink the quota of P_j so that repeated injury will not make the measure of G^A too large. So at the beginning, we actually choose q_j in such a way that $\sum_j q_j < 1/2$, and each time P_j is injured, we halve the current value of q_j. Thus in total, P_j does not add more than $2q_j$ much measure to G^A.

Updating q_j implies that we need to update k_j as well, since we at least need the inequality $h_e(k_j) > k_j/q_j$, which becomes stricter when q_j shrinks. Another reason we would like to update k_j each time P_j is initialised is the fact that we do not wish to keep injuring subrequirements $Q_{e,i}$ which were already maltreated by P_j; we do not want these injuries to accumulate so as to make T_i^e too large, so each time we initialise P_j, we move it beyond any $Q_{e,i}$ which has already been injured by P_j.

But what inequality do we ask for? Suppose that we install P_j at location k_j. Let $i \geqslant k_j$. We already decided that P_j will work with pieces of size q_j/k_j (for the current value of q_j) and so will enumerate at most k_j/q_j many numbers into A, making T_i^e have size possibly $k_j/q_j + 1$. But it is not sufficient to require that $h_e(k_j) > k_j/q_j$, because the requirements $P_0, P_1, \ldots, P_j$ may act in backwards order: first P_j doing its worst, enumerating k_j/q_j many numbers into A; then P_{j-1} (indeed injuring P_j and putting the next manifestation of P_j beyond $Q_{e,i}$, but nevertheless enumerating k_{j-1}/q_{j-1} numbers on its own), and so on. In the case of a single Q_e, this can easily be solved by requiring $h_e(k_j) > (j+1)k_j/q_j$, which is in turn greater than $\sum_{j' \leqslant j} k_{j'}/q_{j'}$. But in the general construction we will need a more complex approach, which we describe later.

2.4. **Considering several Q.** Suppose now that we want to work for all requirements Q_e that matter, namely the requirements Q_e for e such that h_e is total and superlinear. We need to place the requirements P_j on the private lists of the various Q_e's. As we will see shortly, it is only possible for each P_j to respect finitely many Q_e's, so we make a global priority list of the requirements Q_e and P_j. If P_j is stronger than Q_e, then P_j does not appear on Q_e's private list and every action of P_j completely initialises Q_e (which means that we start with a new trace $\langle T_i^e \rangle$). If Q_e is stronger than Q_e, then P_j appears on Q_e's list.

We need to modify our calculations a bit. Fix some P_j, and let E be the finite set of indices e for which Q_e is tended to and stronger than P_j. If $e \in E$ and the current location of P_j on Q_e's private list is $k_{e,j}$, then the number of clopen sets which can be stuck in G_j^A is at most $k = \sum_{e \in E} k_{e,j}$, and this determines the size (q_j/k) of the clopen sets which P_j uses and hence the desired inequalities, $h_e(k_{e,j}) > (j+1)k/q_j$. The point here is that the value of one $k_{e,j}$ is relevant for the inequality $h_{e'}(k_{e',j}) > \dots$ of *every* $e' \in E$, so these numbers cannot be chosen independently. For simplicity, we declare that all of these locations $k_{e,j}$ are equal (to a single number k_j). So P_j uses clopen pieces of size $q_j/k_j|E|$ (actually, in the construction, we use $2^{|E|}$ instead of $|E|$ just to make sure that we have a power of 2).

2.5. **Guessing superlinearity; the tree of strategies.** The construction of the previous subsection would be the complete one if the set of e such that h_e is a total, superlinear order were computable. It is not. In fact, while the property of h_e being a (total) order is Π^0_2 and so can be guessed appropriately, the property of being superlinear is not Π^0_2, indeed not even Δ^0_3 (but Π^0_3). However, the collection of e such that $\liminf_n h_e(n)/n = \infty$ is Π^0_2. Guessing for that collection is not good enough, because to find the appropriate k_j, we need $h_e(k_j) > c \cdot k_j$ for some constant c and *all* $e \leqslant j$ which we believe are relevant. The solution is to adopt a "tree philosophy" and guess, at level e, whether $\liminf_n h_e(n)/n = \infty$, where the limit inferior is only taken over the set of n which have been approved and passed on by the previous $h_{e'}$ which are guessed to be large. That is, say, for example, that $\liminf h_0(n)/n = \infty$; the guessing process extracts an infinite set H_0 such that the n^{th} number k of H_0 satisfies $h_0(k) > nk$. We then test h_1 relative to H_0: we try to find a subset $H_1 \subseteq H_0$ such that the n^{th} number k of H_1 satisfies $h_1(k) > nk$. Since $H_1 \subseteq H_0$, we'll of course also get $h_0(k) > nk$.

The point is that if h_e is truly superlinear, then this process of guessing will not cause us to miss h_e and we will be able to find the infinite set H_e, relative to any infinite set H_{e-1} which will be handed to us. This will ensure that if we really need to tend to Q_e, we will in fact do so.

We use a simple tree of strategies; nodes on level $2e$ work for requirement Q_e and have outcomes `fin` and `inf`, and nodes on level $2j+1$ work for requirement P_j and only have a single outcome. Nodes γ which work for some P_j build their own version of G_j^A, which we call G_γ^A (at the end we let G^A be the union of all the versions of all G_γ^A). We issue quotas q_γ such that

$$\sum_\gamma q_\gamma < 1/2. \tag{2.1}$$

Similarly, every node α which works for some Q_e builds a trace $\langle T_i^\alpha \rangle$ for Φ_e^A and has a private priority list which we indicate by $Q_{e,0}, Q_{e,1}, \dots$.

We note that our method of initialising P_j's also takes care of initialising those P_γ which lie to the right of the current accessible node (this is why we introduced it in the first place). The original problem is that if a P_j-node γ extends the finite outcome of a Q_e-node α, then γ believes that Q_e can be safely ignored, and so the location k_γ (which is chosen with respect to those $Q_{e'}$ which γ does not ignore) may not be sufficiently large to accommodate α's private priority list. So when the infinite outcome of α is accessible, *all* the subrequirements on α's list must have priority over γ. Of course, γ did not take into account this amount of restraint, which may make G^A_γ too large. Initialisation takes care of this concern.

Now we need to describe a problematic configuration which explains why the approach we outlined above for determining which inequalities $h_e(k_\gamma)$ should be satisfied for the construction to work is actually insufficient. Consider three nodes, α_0 (working for Q_{e_0}), α_1 (working for Q_{e_1}) and γ (working for P_j) such that $\alpha_0 * \mathtt{inf} \subset \alpha_1 * \mathtt{inf} \subseteq \gamma$. Suppose that we wish to define k_γ; we plan to let γ use clopen pieces of size $m_\gamma := q_\gamma/(4k_\gamma)$. Suppose for simplicity that γ is the strongest P-node on the tree.

The issue is of timing. While waiting for h_{e_1} to converge and yield large values, we follow the node $\alpha_1 * \mathtt{fin}$ (assuming that h_{e_0} converges much more quickly than h_{e_1}). Various P-nodes extending $\alpha_1 * \mathtt{fin}$ only need to consider α_0's private list and are set up along that list, and as the P-nodes act, they increase the size of various $T^{\alpha_0}_i$. When γ is finally accessible, the only candidates for k_γ may be small relative to the i for which $T^{\alpha_0}_i$ is already large, as h_{e_1} has not caught up to the length of convergence of h_{e_0}. So the inequality $h_{e_0}(k_\gamma) > m_\gamma$ is no longer sufficient to prevent the bust of these $Q_{e_0,i}$. This situation can repeat indefinitely.

The solution is for $Q_{e_0,i}$ to be proactive and make sure it never gets injured too often by making sure that not too many P-nodes ever get placed before it. Surprisingly, the most elegant way to achieve this is indirectly, by requiring that the k_γ satisfy even more stringent inequalities than was indicated above. For this purpose, we attach, for every P-node γ, a constant c_γ such that

$$\sum_{\gamma \in \mathcal{P}} 1/c_\gamma < 1/4, \tag{2.2}$$

where $\mathcal{P}$ is the set of all P-nodes on the tree of strategies; and we require, when setting k_γ, that for all nodes Q_e nodes α which P_γ needs to respect, that $h_e(k_\gamma) > c_\gamma m_\gamma$. This ensures that $|T^\alpha_i| \leqslant h_e(i)$, because if Γ is the (finite) set of nodes which injure α, then

$$|T^\alpha_i| \leqslant \sum_{\gamma \in \Gamma} m_\gamma \leqslant \sum_{\gamma \in \Gamma} \frac{h_e(k_\gamma)}{c_\gamma} \leqslant h_e(i) \sum_{\gamma \in \Gamma} \frac{1}{c_\gamma} \leqslant h_e(i)$$

as is required. (Again we use here the fact that only one "version" of each γ can injure α, because if γ is injured after T^α_i is started, then future values of k_γ will be larger than i.)

2.6. **Construction.** We are now ready to give the formal construction.

For any node γ, we let I_γ be the collection of indices e such that there is some node α which works for Q_e such that $\alpha * \mathtt{inf} \subseteq \gamma$.

Let $\forall x \exists y R(e, x, y)$ be a Π^0_2 condition which states that h_e is a total order. At stage s we let $\ell(e)[s]$ be the least x such that there is no $y < s$ for which $R(e, x, y)$ holds.

At stage s we build the path of accessible nodes and act as follows.

Suppose that γ, a node which works for P_j, is accessible at stage s. Let r be the last stage before s at which γ was initialised (0 if γ was never initialised).

(1) If k_γ is not defined: look for a number $k > r$ (but $k < s$) which is a power of 2 such that for all $e \in I_\gamma$ we have $h_e(k) > c_\gamma 2^{|I_\gamma|} k/q_\gamma[s]$. If such a number is not found, terminate this stage. Otherwise, declare such a number to be the new k_γ.

Divide 2^ω into clopen subsets, each of measure $q_\gamma/k_\gamma 2^{|I_\gamma|}$. Terminate this stage.

(2) If there is a clopen piece C which is currently processed by γ, do the following (if not, go to (3)). If $C \not\subseteq V_j[s]$, let the successor of γ be accessible (and move to act for it). Otherwise, let u be the use of enumerating C into G^A_γ. If there is some α, working for Q_e, such that $\alpha * \mathtt{inf} \subseteq \gamma$ (so $e \in I_\gamma$), and some $i < k_\gamma$ such that $\Phi^A_e(i)\downarrow\ [s]$ with use greater than u and value $\Phi^A_e(i)$ already traced in T^α_i, then just go to (3). Otherwise, enumerate u into A and go to (3).

(3) Pick a clopen piece C which hasn't yet been processed. Pick a large number u and enumerate C into G^A_γ with use u. End this stage.

Next, suppose that α, a node which works for Q_e, is accessible at stage s. Let r be the last stage at which either α was initialised or at which $\alpha * \mathtt{inf}$ was accessible. If both $\ell(e)[s] > r$ and there is some number $k > r$, $k < s$ which is a power of 2 and such that for every $e' \in I_\alpha \cup \{e\}$ we have $h_{e'}(k) > rk$, then we let $\alpha * \mathtt{inf}$ be accessible next. Otherwise, we let $\alpha * \mathtt{fin}$ be accessible.

If $\alpha * \mathtt{inf}$ is accessible, then before we act for it, we update α's traces: for every $i < r$ such that $\Phi^A_e(i)\downarrow\ [s]$, we enumerate $\Phi^A_e(i)[s]$ into T^α_i.

At the end of the stage, we initialise all nodes which lie above the last accessible node or to the right of it. Initialising a node γ which works for P_j means setting $q_\gamma[s+1] = q_\gamma[s]/2$, making k_γ undefined, and resetting G^A_γ to be empty (by "emptying" it into the global G^A). Initialising a node α which works for Q_e means resetting the trace $\langle T^\alpha_i \rangle$.

2.7. **Verification.** The following lemmas show that the construction satisfies the requirements of Theorem 1.7.

Lemma 2.1 (True Path)**.**

- *If β is the leftmost node of its length that is visited infinitely often, then for every $m \in \mathbb{N}$ there is some $k \in \mathbb{N}$ such that $h_e(k) > m \cdot k$ for all $e \in I_\beta$.*
- *The leftmost infinitely often visited path f of the tree is infinite.*

Proof. The first clause follows by the construction and the definition of I_β. The second follows from the first, because if β is a P-node on the true path, then the first clause ensures that the construction cannot be stuck in step (1) at all but finitely many β-stages. □

Lemma 2.2 (Properties of the P nodes)**.** *Let γ be a P node on the true path. Then:*

- *After some stage, γ is never initialized again and k_γ reaches a limit.*
- *During an interval $[s, t]$ of stages where γ is not initialized, it can make at most $k_\gamma[s]2^{|I_\gamma|}/q_\gamma[s]$ enumerations into A.*

- *At each stage s, the set $G^A_\gamma[s]$ contains at most $k_\gamma[s] \cdot |I_\gamma|$ clopen sets.*
- *At each stage s, each clopen set in $G^A_\gamma[s]$ is of measure $q_\gamma[s]/k_\gamma[s]2^{|I_\gamma|}$. So $\mu(G^A_\gamma) \leq q_\gamma[s]$.*

Proof. By induction on the length of γ. The first clause holds because, by induction, nodes preceding γ stop initialising γ after some stage. For the second clause, note that each time a P node which precedes γ completes a cycle (i.e. goes from step 3 to step 2), the measure of the corresponding set V increases by a constant amount, namely $q_\gamma[s]/k_\gamma[s]2^{|I_\gamma|}$. Hence there can be at most $k_\gamma[s]2^{|I_\gamma|}/q_\gamma[s]$ enumerations into A by γ. For the third clause note that $|G^A_\gamma|$ increases only when some computation of some Q node above γ with index in I_γ for some argument $< k_\gamma$ involves a use below the use of G^A_γ. But there are at most $|I_\gamma| \cdot k_\gamma$ such computations. The last clause follows from the construction and the third clause. □

Lemma 2.3 (Satisfaction of the Q requirements)**.** *If α is a Q_d-node on the true path and $\lim_n h_\alpha(n)/n = \infty$, then:*

- *$\Phi^A_\alpha(k) \in T^\alpha_k$ for all k such that $\Phi^A_\alpha(k)\downarrow$*
- *$|T^\alpha_k| \leq h_\alpha(k)$ for all $k \in \mathbb{N}$.*

Proof. For each $m \in \mathbb{N}$ there are infinitely many $k \in \mathbb{N}$ such that $h_e(k) > k \cdot m$ for all $e \in I_\alpha$. Since $\lim_n h_\alpha(n)/n = \infty$, for each $m \in \mathbb{N}$ there are infinitely many $k \in \mathbb{N}$ such that $h_e(k) > k \cdot m$ for all $e \in I_\alpha \cup \{d\}$. Hence there are infinitely many α-expansionary stages (i.e. stages where the outcome `inf` of α is accessed), and according to the construction for every k, the value $\Phi^A_\alpha(k)$ (if it is defined) will be recorded in T^α_k. This establishes the first clause.

For the second clause, let s_0 be the last stage at which α was initialised. Then $|T^\alpha_k[s_0]| = \emptyset$. By the initialisation of nodes to the left of the approximation of the true path, after stage s_0 only P nodes below $\alpha * \mathtt{inf}$ can ever enumerate a number into A which is less than the use of a computation $\Phi^A_\alpha(k)$ whose value is in T^α_k. Let us call such an enumeration (α, k)-*violating* and suppose, for a contradiction, that $|T_\alpha(k)| > h_\alpha(k)$ for some $k \in \mathbb{N}$. According to the discussion above, this means that $h_\alpha(k)$ many (α, k)-violating enumerations into A occurred after stage s_0, which were caused by P nodes below $\alpha * \mathtt{inf}$. Let $(\gamma_i)_{i<t}$ be the P nodes which where involved in the first $h_\alpha(k)$ such enumerations. We note that no γ_i can perform (α, k)-violating enumerations both before a stage where it was injured and after that. This is because after an injury of γ_i the parameter k_{γ_i} receives a large value. The parameters $k_{\gamma_i}, q_{\gamma_i}$ mentioned below refer to the interval of stages where γ_i performed the (α, k)-violating enumerations. According to the construction (in particular the choice of k_{γ_i} at step 1 of the P strategy) we have

$$h_t(k) > c_{\gamma_i} 2^{|I_{\gamma_i}|} k_{\gamma_i}/q_{\gamma_i} \tag{2.3}$$

for all $i < t$. By Lemma 2.2, each γ_i can make at most $k_{\gamma_i}2^{|I_{\gamma_i}|}/q_{\gamma_i}$ enumerations into A during any interval of stages at which it is not injured, so, at most $k_{\gamma_i}2^{|I_{\gamma_i}|}/q_{\gamma_i}(\alpha, k)$-violating enumerations. Hence $\sum_{i<t} k_{\gamma_i}2^{|I_{\gamma_i}|}/q_{\gamma_i} \geq h_t(k)$. But by (2.3), (2.1) and (2.2) we have $\sum_{i<t} k_{\gamma_i}2^{|I_{\gamma_i}|}/q_{\gamma_i} < \sum_{i<t} h_\alpha(k)/c_i \leq h_\alpha(k)/2$, which is a contradiction. □

Lemma 2.4. *The set A is not low for random.*

Proof. Let G^A be the union of all versions of G^A_γ for all P-nodes γ. By (2.1), the fourth clause of Lemma 2.2, and since q_γ is halved each time γ is initialized, we

have $\mu(G^A) = \sum_{\gamma \in \mathcal{P}} \mu(G^A_\gamma) < 1$. We show that for all $e \in \mathbb{N}$, $G^A \not\subseteq V_e$. Let γ be a P_e-node on the true path; let s_0 be a stage after which γ is never initialised. If $G^A_\gamma \subseteq V_e$, the strategy γ would proceed in forcing the whole space 2^ω into V_e by ensuring that successive small intervals belong to V_e, but this is impossible since $\mu(V_e) < 1$. □

References

[1] George Barmpalias, Andrew E. M. Lewis, and Mariya Soskova. Randomness, lowness and degrees. *J. of Symbolic Logic*, 73(2):559–577, 2008. MR2414465

[2] Peter Cholak, Rod Downey, and Noam Greenberg. Strong jump-traceability, I: The computably enumerable case. *Adv. Math.*, 217:2045–2074, 2008. MR2388085

[3] Rod Downey and Noam Greenberg. Strong jump-traceability, II: The general case, in preparation.

[4] Rod Downey and Denis Hirshfeldt. *Algorithmic Randomness and Complexity*. Springer-Verlag, in preparation, 2009.

[5] Santiago Figueira, André Nies, and Frank Stephan. Lowness properties and approximations of the jump. In *Proceedings of the 12th Workshop on Logic, Language, Information and Computation (WoLLIC 2005)*, volume 143 of Electron. Notes Theor. Comput. Sci., pages 45–57 (electronic), Elsevier, Amsterdam, 2006. MR2270232

[6] Bjørn Kjos-Hanssen. Low for random reals and positive-measure domination. *Proc. Amer. Math. Soc.*, 135(11):3703–3709 (electronic), 2007. MR2336587 (2008g:03070)

[7] 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. MR2201451 (2006j:68051)

[8] Joseph S. Miller and André Nies. Randomness and computability: Open questions. *Bull. Symbolic Logic*, 12(3):390–410, 2006. MR2248590 (2007c:03059)

[9] A. Nies. Reals which compute little. In *Logic Colloquium '02*, Lecture Notes in Logic, 27, pages 261–275, Assoc. Symbol. Logic, La Jolla, CA, 2006. MR2258710 (2007i:03048)

[10] André Nies. Lowness properties and randomness. *Adv. Math.*, 197(1):274–305, 2005. MR2166184 (2006j:68052)

[11] André Nies. Eliminating concepts. In *Proceedings of the IMS workshop on computational aspects of infinity, Singapore*, to appear, 2008.

[12] André Nies. *Computability and Randomness*. Oxford University Press, in preparation, 2009.

[13] Keng-Meng Ng. Beyond jump traceability, preprint.

[14] S. Terwijn and D. Zambella. Algorithmic randomness and lowness. *J. Symbolic Logic*, 66:1199–1205, 2001. MR1856736 (2002j:03044)

School of Mathematics, Statistics and Computer Science, Victoria University, P.O. Box 600, Wellington, New Zealand

E-mail address: George.Barmpalias@mcs.vuw.ac.nz

School of Mathematics, Statistics and Computer Science, Victoria University, P.O. Box 600, Wellington, New Zealand

E-mail address: downey@mcs.vuw.ac.nz

School of Mathematics, Statistics and Computer Science, Victoria University, P.O. Box 600, Wellington, New Zealand

E-mail address: greenberg@mcs.vuw.ac.nz