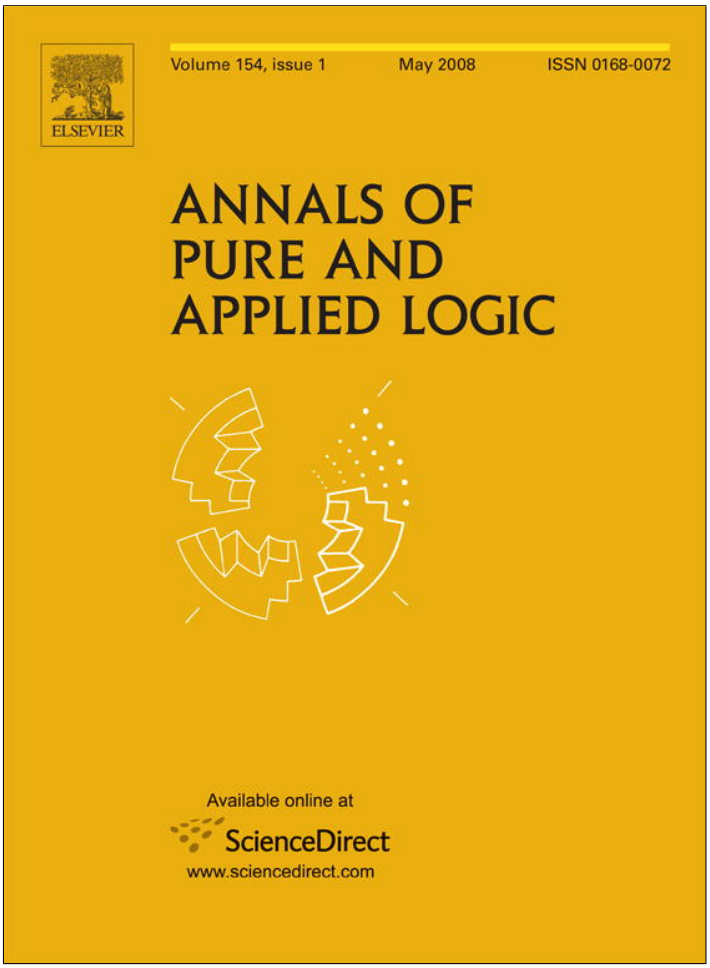


Available online at www.sciencedirect.com
ScienceDirect
Annals of Pure and Applied Logic 154 (2008) 51–69
ANNALS OF PURE AND APPLIED LOGIC
www.elsevier.com/locate/apal

On strongly jump traceable reals

Keng Meng Ng

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

Received 15 March 2007; received in revised form 13 November 2007; accepted 14 November 2007
Available online 30 January 2008

Communicated by R.I. Soare

Abstract

In this paper we show that there is no minimal bound for jump traceability. In particular, there is no single order function such that strong jump traceability is equivalent to jump traceability for that order. The uniformity of the proof method allows us to adapt the technique to showing that the index set of the c.e. strongly jump traceables is Π^0_4-complete.

MSC: primary 03D25; secondary 68Q30

Keywords: Strongly jump traceable; Completeness; Minimal bound

1. Introduction

One of the fundamental concerns of computability theory is in understanding the relative difficulty of computational problems as measured by Turing reducibility ($\leq_T$). The equivalence classes of the preordering $\leq_T$ are called Turing degrees, and it is long recognized that the fundamental operator on the structure of the Turing degrees is the jump operator. For a set A, the halting problem relative to A is denoted by A', and if $\mathbf{a}$ is the Turing degree of A, then $\mathbf{a}'$ would denote the Turing degree of A'. As it is well known, if $\mathbf{a} \leq_T \mathbf{b}$ then $\mathbf{a}' \leq_T \mathbf{b}'$, but the jump operator is not injective on the degrees.

The concern of this paper is computational lowness. Here a set A is low relative to the Turing jump if $A' \equiv_T \emptyset'$. A low set A is indistinguishable from the empty set as far as the jump operator is concerned. We would expect that low sets resemble computable sets, and there is a long and rich literature exploring this idea. This seems particularly true for the computably enumerable sets, and we mention a few well-known examples.

Soare [19] showed that if A is c.e. low, then the lattice of c.e. supersets of A is isomorphic to the lattice of c.e. sets modulo finite sets. Robinson [13] extended the Sack's Splitting Theorem [14] by showing that any c.e. degree can be split above a low one. A generalization of Lachlan's Non-Diamond Theorem [7] by Ambos-Spies [1] over low degrees further show that the interval $[\boldsymbol{a}, \mathbf{0}']$ is structurally very similar to $[\mathbf{0}, \mathbf{0}']$ when $\boldsymbol{a}$ is low.

Shore and Slaman [15] showed that there is no Slaman triple below a c.e. low (in fact, low_2) degree, while in [16] they proved that every high degree (A is high iff $A' \equiv_T \emptyset''$) bounds a Slaman triple. This gives an elementary property

E-mail address: Keng.Meng.Ng@mcs.vuw.ac.nz.

0168-0072/$ - see front matter
doi:10.1016/j.apal.2007.11.014

which separates the high and low (low_2) c.e. degrees. Further work by Soare (no low set is speedable [18]), and Downey and Jockusch (every low Boolean algebra is isomorphic to a computable one [4]) further demonstrate that the low sets behave just like the computable ones. In an amazing paper, Slaman and Solovay [17] showed that every set A which was low for EX learning, is also low (in fact 1-generic below $\emptyset'$). It is not important in this paper what lowness in terms of EX learning means; what is important is that Slaman and Solovay's result demonstrate that lowness for various notions of computation can be intertwined. In this case, they demonstrate a relationship between a lowness concept from the theory of inductive inference and another seemingly unrelated lowness concept from computability theory. This idea is further explored in this paper, where we will investigate lowness for Kolmogorov complexity and its relationships.

Because of these results and perhaps also because the "Robinson trick" (see Soare [21], Chapter XI) was so well understood, the low computably enumerable sets were thought to be reasonably understood in terms of their degree-theoretical properties.

However, recent work has again turned the spotlight on this class, and demonstrated that low c.e. sets have an astonishing theory. This is due to their relationship with another computational lowness notion – lowness in terms of Kolmogorov complexity – and have been shown to be related to the widely studied class of the K-trivials.

The class of K-trivial reals was first introduced in [22]. They are the reals α such that for some constant c, $K(\alpha\restriction_n) \leq K(n) + c$ for every n.[1] That is, the K-trivials have got very low initial segment complexity, similar to the computable ones. Solovay [22] showed however, that there are noncomputable reals which are K-trivial. In spite of Solovay's theorem, the resemblance they bear with the computable reals makes one wonder if they are related to the low sets. Recent work have shown that this is indeed the case. In particular Nies [11,12] showed that every c.e. K-trivial was superlow,[2] and Cholak, Downey and Greenberg [2] discovered that a certain class of reals exhibiting very strong "lowness properties" form a proper natural subclass of the K-trivials.

This brings us to the fundamental notion of traceability. An order function h is one which is total computable, nondecreasing and unbounded. A set A is said to be *jump traceable* with respect to an order h, if there is a computable g, such that for all x, $|W_{g(x)}| \leq h(x)$, and $J^A(x) \in W_{g(x)}$.[3] A is said to be jump traceable, if it is jump traceable with respect to some order h. This is a variation of the concepts of computable traceability (Terwijn and Zambella [23]), and c.e. traceability (Ishmukhametov [6]). The class of jump traceable reals was introduced by Nies [11] to study lowness properties. He showed that in the c.e. case, jump traceability and superlowness were the same, but were different outside of the c.e. sets.

In further work, Nies [11,12] showed that every K-trivial real was jump traceable and hence superlow, with an order function of growth rate $\sim h(n) = n \log n$. Inspired by these results, Figueira, Nies and Stephan [5] went on to study the notion of strong jump traceability. We say that A is *strongly jump traceable*, if it is jump traceable with respect to all order functions. Figueira, Nies and Stephan showed the existence of a noncomputable strongly jump traceable c.e. set, and characterized c.e. strong jump traceability via the notion of well approximability: A set A is *well approximable*, if for every order function h, A can be effectively approximated with less than $h(x)$ many changes at each input x. Figueira, Nies and Stephan showed that if A is c.e., then A is strongly jump traceable if and only if A' is well approximable.

In an upcoming work, Cholak, Downey and Greenberg [2] show that the c.e. strongly jump traceables form a proper subclass of the K-trivials. This is the first example of a combinatorial property which implies K-triviality. They show that like the K-trivials, the c.e. strongly jump traceables are also closed under $\oplus$, and constructed a K-trivial c.e. real which is not jump traceable with respect to a bound of size $\sim h(n) = \log \log n$.

Figueira, Nies and Stephan [5] also studied the role that the size of the bound h has on jump traceability. They showed that for any order function h, there is always some set A which is jump traceable, but not jump traceable via h. That is, no matter how fast an order h grows, there is always some jump traceable set A for which h grows too slowly still, for the purpose of jump tracing A. Hence, unlike the case of computable traceability, strong jump traceability was actually different from jump traceability. Figueira, Nies and Stephan [5] also asked if there was a minimal bound for jump traceability: Is there an order function h, such that every set A which is jump traceable via h is already strongly jump traceable? In Theorem 2.1, we answer the question in the negative:

[1] $K(\sigma)$ is the prefix-free Kolmogorov complexity of the string σ.

[2] A is superlow, if $A' \equiv_{tt} \emptyset'$.

[3] $J^A(x)$ denotes the value of the universal jump function $\{x\}^A(x)$, partial computable in A.

Theorem 2.1. *For any given order function h, there is a c.e. set A and an order function $\tilde{h}$, such that A is jump traceable via h, but not jump traceable via $\tilde{h}$.*

In particular, there is no single order function such that strong jump traceability is the same as jump traceability for that order.

Nies observed that the K-trivial reals form the first example of a natural nontrivial Σ^0_3 ideal in the c.e. Turing degrees. Had there been a slowest order we could use to define strong jump traceability, the strongly jump traceables would also have to be Σ^0_3. We have already learnt from Cholak, Downey and Greenberg [2] that there are K-trivials which are not strongly jump traceable, and so the following Theorem 3.3 would show that in terms of the complexity of the classes, the strongly jump traceables are as complex as they could be, and in fact differ from the K-trivials as much as they possibly can:

Theorem 3.3. *The set $\{e \in \mathbb{N} : W_e$ is strongly jump traceable$\}$ is Π^0_4-complete.*

We will use the technique in the proof of Theorem 2.1 to prove Theorem 3.3. As a corollary, we get the result of Cholak, Downey and Greenberg [2] that not all K-trivials are strongly jump traceable.

We would like to make a further remark. Cholak, Downey and Greenberg showed in [2] that the c.e. strongly jump traceables form a nonprincipal ideal. In fact, they showed something stronger: There is an effective procedure Λ, such that given any order function h, we have another order function $\Lambda(h)$ growing slower than h, such that if A and B are c.e. and jump traceable with respect to $\Lambda(h)$, their join $A \oplus B$ will be jump traceable with respect to h. For each order function h, we define

$$\mathcal{H}_h := \{\Lambda^n(h) : n > 0\},$$

where Λ^n denotes iterating the Λ-procedure n times. We say that A is $\mathcal{H}_h$-jump traceable, if A is jump traceable with respect to all $g \in \mathcal{H}_h$.

Clearly for each order function h, the class of c.e. sets A which are $\mathcal{H}_h$-jump traceable forms an ideal. When h grows slow enough ($h(n) \ll \log\log n$), the ideal it generates is contained in the ideal of the K-trivials. Since we can always diagonalize against all the functions in each $\mathcal{H}_h$, it follows from Theorem 2.1 that there are infinitely many intermediate ideals lying between the ideals of the K-trivials, and the strongly jump traceables.

2. There is no minimal bound for jump traceability

We show that there is no minimal bound for jump traceability, answering a question of Figueira, Nies and Stephan [5]. In particular, there is no single order function such that strong jump traceability is the same as jump traceability for that order.

Theorem 2.1. *For any given order function h, there is a c.e. set A and an order function $\tilde{h}$, such that A is jump traceable via h, but not jump traceable via $\tilde{h}$.*

2.1. Requirements

We build a c.e. set A, and a trace $\{V_e\}_{e\in\mathbb{N}}$ for $J^A(e)$ satisfying the following requirements:

$\mathcal{N}_e$: Trace $J^A(e)$ into V_e, with $|V_e| \leq h(e)$,

$\mathcal{P}_e$: For some x, either $|T^e_x| > e$, or else $J^A(x) \notin T^e_x$.

Here, we let $\{T^e_x\}_{x\in\mathbb{N}}$ be the eth trace in some effective enumeration of all traces $\{T^0_x\}_{x\in\mathbb{N}}, \{T^1_x\}_{x\in\mathbb{N}}, \ldots$, and $J^A(e)[s]$ be the value of the universal A-jump function $\{e\}^A(e)[s]$ at stage s. We let the use of $J^A(e)[s]$ (if convergent) be $j(e, s)$. Note that if h is an order function, we always assume that its range takes positive values. When we say that we pick a *fresh* number x at stage s, we mean that we choose x to be the least number $x > s$, and $x >$ any number used or mentioned so far.

2.2. Description of strategy

We have two types of requirements to handle — the negative requirements $\mathcal{N}_e$ which want to impose a restraint on A each time it sees a computation $J^A(e)[s]\downarrow$, and the positive ones $\mathcal{P}_e$ which makes enumerations into A to force a particular trace T^e_x to fill up.

The way that a particular $\mathcal{N}_e$ works is described below. Since we have to ensure that $|V_e| \leq h(e)$, we have to make sure that restraints imposed by $\mathcal{N}_e$ are not obeyed on most $h(e) - 1$ many occasions. It is convenient to think of V_e as being made up of boxes, which we will fill with values (which are current values of $J^A(e)[s]$). By the time we use up all of the $h(e)$ many boxes allowed for V_e, we have to ensure that the last value we enumerated is correct. We start the construction with $\mathcal{N}_e$ initially being a $(h(e) - 1)$-box. (This is represented in the formal construction by the variable $size(e, s)$, and represents the number of injuries $\mathcal{N}_e$ can still sustain). We also say that $\mathcal{N}_e$ is an *original* $(h(e)-1)$*-box*. Every time the restraint that $\mathcal{N}_e$ is holding is breached by some positive action, we would decrease $size(e, s)$ by 1, since a new value for $J^A(e)$ might appear in future. By the time $\mathcal{N}_e$ becomes a 0-box (corresponding to $|V_{e,s}| = h(e)$), we will have to ensure that no enumerations made in future can destroy the current $J^A(e)[s]$ computation. That is, the restraint imposed by $\mathcal{N}_e$ while it is a 0-box must be obeyed by every positive requirement.

Let us now turn our attention to the positive requirement $\mathcal{P}_e$. We describe how to meet such a requirement. $\mathcal{P}_e$ would attempt to defeat the eth trace by doing the following. It will control the value of the universal jump function $J^A(x)$ of A at some location x. The recursion theorem supplies us with such an index x, and $\mathcal{P}_e$ will enumerate axioms into the Turing functional Ψ_e^A with index x, with use $u(e, s)$ on A. Each time the value $\Psi_e^A(x)[s]$ shows up in the trace T_x^e, we would put the use $u(e, s)$ into A to cancel all the previous axioms, and enumerate a new axiom $\langle x, y, A_s\restriction_{u(e,s)}\rangle$ for $\Psi_e^A(x)$ (with a fresh y). After doing this at most e times, we would be able to meet the requirement $\mathcal{P}_e$: recall that we have to build an order function $\tilde{h}$ globally, and all we have to do is to ensure that we define $\tilde{h}(x) = e$. On the other hand, the negative requirements would be imposing various restraints on $\mathcal{P}_e$, as described in the previous paragraph, for the sake of making A superlow. At times we would have to initialize $\mathcal{P}_e$, due to these restraints. For instance, if some $\mathcal{N}_k$ is in a state of being a 0-box (these boxes have the highest priority), with restraint larger than $u(e, s)$, then we would have to make $\mathcal{P}_e$ abandon the current index, and begin to enumerate a new functional with a new index x'. To ensure the success of $\mathcal{P}_e$, we would have to make sure that it is initialized only finitely often. In fact, to guarantee that $\tilde{h}$ is computable, we have to know in advance a bound for the number of times that $\mathcal{P}_e$ will need to be initialized. This is because we could then know how many different indices to set aside for $\mathcal{P}_e$, and hence define $\tilde{h} = e$ on these indices.

The construction will only require finite injury, with dynamic assignment of priority amongst the requirements. As we will see, the main obstacle we are facing is in having to arrange priority between the positive and negative requirements, such that we can limit the number of initializations to each $\mathcal{P}_e$ to an amount that can be predetermined. Let us consider the case when the given h satisfies $h(0) = h(1) = h(2) = h(3) = 1$ and $h(4) = 3$. Note that in general, if the given h grows very slowly, then it becomes much harder for numbers to enter A because there are more small boxes to consider. Consider a requirement $\mathcal{P}$ that wants to diagonalize some trace by enumerating into A twice. Suppose we arrange the requirements in the order:

$$\mathcal{N}_0(h = 1) < \mathcal{N}_1(h = 1) < \mathcal{N}_2(h = 1) < \mathcal{N}_3(h = 1) < \mathcal{P} < \mathcal{N}_4(h = 3).$$

For $\mathcal{P}$ to succeed at a particular index x, its cycle for that x has to be

> *Phase* 1: Set $J^A(x)[s_1]\downarrow$. Wait for the corresponding value to show up in the trace. If it does, put the use into A to reset $J^A(x)$.
> *Phase* 2: Set $J^A(x)[s_2]\downarrow$ again and wait for the value to show up in the trace. When it does, put the use into A to reset $J^A(x)$, set a new axiom for $J^A(x)$, and we are done.

If $\mathcal{P}$ gets blocked in phase 2, it will be initialized and will have to *start with a new index* x' *in phase* 1. Why is this a problem in the above example? When $\mathcal{P}$ is in phase 1, it will have a follower appointed pointing at A, which it will put in A when realized. But in the meantime we might have $\mathcal{N}_4$ imposing an A-restraint above the $\mathcal{P}$-follower. This is due to the fact that $\mathcal{N}_4$ has seen $J^A(4)$ converge with a large A-use, and $\mathcal{N}_4$ has put that value into the trace we are building for $J^A(4)$.

Suppose next, the $\mathcal{P}$-follower gets realized. It will then enumerate the $\mathcal{P}$-follower it has appointed and enter phase 2, injuring $\mathcal{N}_4$ in the process. Remember that $\mathcal{N}_4$ is allowed two mistakes (i.e. it is an original 2-box), and now it has used up one of them. Therefore, in future it is only allowed one more mistake (i.e. it has now been *promoted* to a 1-box).

$\mathcal{P}$ is now waiting in phase 2 for its follower to be realized. It might be the case that $\mathcal{N}_0$ now imposes A-restraint larger than $\mathcal{P}$'s follower, forcing $\mathcal{P}$ to be initialized and start again in phase 1. This looks bad, because the process could be repeated with $\mathcal{N}_1$ in the same manner, and $\mathcal{N}_4$ can be promoted yet again, now to a 0-box. When $\mathcal{N}_4$ next

imposes A-restraint, being a 0-box, its restraint has to be obeyed by everyone, including $\mathcal{P}$ above it. Again we could create any number of 0-boxes in this way, and in turn use them to produce even more 0-boxes further down the list of requirements, and we are faced with the same problem.

The solution is to arrange priority between $\mathcal{P}$ and the negative requirements dynamically. This priority ordering depends on whether $\mathcal{P}$ is in first phase, or in second phase. If $\mathcal{P}$ is in the first phase, we place $\mathcal{P}$ above (stronger priority than) all $\mathcal{N}_e$ which are currently at least 2-boxes, and place $\mathcal{P}$ below (weaker priority than) all $\mathcal{N}_e$ which are currently 0 or 1-boxes. If $\mathcal{P}$ is in the second phase then we place it above all $\mathcal{N}_e$, other than those that are currently 0-boxes. At the beginning of the construction, before anything is done, we have the ordering:

$$\underbrace{\mathcal{N} < \cdots}_{0\text{-boxes},h=1} < \underbrace{\mathcal{N} < \cdots}_{1\text{-boxes},h=2} < \mathcal{P}_{\text{(phase 1)}} < \underbrace{\mathcal{N} < \cdots}_{2\text{-boxes},h=3} < \underbrace{\mathcal{N} < \cdots}_{3\text{-boxes},h=4} < \cdots.$$

When $\mathcal{P}$ enters phase 2, the situation becomes

$$\underbrace{\mathcal{N} < \cdots}_{0\text{-boxes},h=1} < \mathcal{P}_{\text{(phase 2)}} < \underbrace{\mathcal{N} < \cdots}_{1\text{-boxes},h=2} < \underbrace{\mathcal{N} < \cdots}_{1\text{-boxes},h=3} < \underbrace{\mathcal{N} < \cdots}_{2\text{-boxes},h=3} < \underbrace{\mathcal{N} < \cdots}_{2\text{-boxes},h=4} < \underbrace{\mathcal{N} < \cdots}_{3\text{-boxes},h=4} < \cdots.$$

If $\mathcal{P}$ gets initialized while in phase 2 due to one of the 0-boxes, the ordering becomes

$$\underbrace{\mathcal{N} < \cdots}_{0\text{-boxes},h=1} < \underbrace{\mathcal{N} < \cdots}_{1\text{-boxes},h=2} < \underbrace{\mathcal{N} < \cdots}_{1\text{-boxes},h=3} < \mathcal{P}_{\text{(phase 1)}} < \underbrace{\mathcal{N} < \cdots}_{2\text{-boxes},h=3} < \underbrace{\mathcal{N} < \cdots}_{2\text{-boxes},h=4} < \underbrace{\mathcal{N} < \cdots}_{3\text{-boxes},h=4} < \cdots.$$

We claim that this solves the problem, namely that we can count the number of times $\mathcal{P}$ is forced to be initialized. The ability to perform this counting is essential for $\tilde{h}$ to be computable. First, note that *no new* 0*-boxes are ever created*, unless $\mathcal{P}$ is permanently satisfied at the same time. That is, the only 0-boxes present are those original ones — namely, those $\mathcal{N}$ with $h = 1$.

The counting of injuries to $\mathcal{P}$*:* whilst in the second phase, $\mathcal{P}$ can only by initialized by a 0-box, we have already observed that these must be original 0-boxes. In the first phase, $\mathcal{P}$ would be initialized:

(1) either by some $\mathcal{N}$ with $h = 1$ or 2 (i.e. the original 0 and 1-boxes), or
(2) a promoted 1-box.

Suppose case 2 happens at stage s. The only reason why a 2-box is promoted to a 1-box, is because it was injured by $\mathcal{P}$ and $\mathcal{P}$ moved from the first phase to the second phase, at some previous stage $t < s$. But now at stage s, $\mathcal{P}$ is back in the first phase, which means that at some time between t and s, $\mathcal{P}$ must have been initialized while in phase 2. This can only be done by some $\mathcal{N}$ with $h = 1$, i.e. one of the original 0-boxes, since these are the only requirements stronger than $\mathcal{P}$ in phase two. This means that the largest k such that some $\mathcal{N}$ with $h = k + 1$ (original k-box), is ever promoted to a 1-box, is at most $S := 2 + h^{-1}(1)$, where $h^{-1}(i) = \#$ of y such that $h(y) = i$. That is, if $k > S$ then no $\mathcal{N}$ which is originally a k-box, can ever get promoted to a box size of 1. Therefore, the number of times that $\mathcal{P}$ can be injured, has bounds of $h^{-1}(1)$ from phase 2, and $\sum_{i \leq S+1} h^{-1}(i)$ while in phase 1.

2.3. The general strategy

The requirement $\mathcal{P}_e$ will need to enumerate e many times without being initialized; its action will be divided into e many phases. In the discussion above, the $\mathcal{P}$ being considered is just $\mathcal{P}_2$. In the example above we had the three numbers $C_0^1 = 0$, $C_0^2 = 1$ and $C_1^2 = S$, which are called *thresholds*.[4] These are the critical numbers which we use to determine priority between $\mathcal{P}_2$ and the negative requirements. In phase 1, $\mathcal{P}_2$ would be injured by C_0^2-boxes (and smaller ones). In phase 2, $\mathcal{P}_2$ would be injured by C_0^1-boxes. To prevent the different positive requirements from interfering with each other, we ensure that no new C_1^2-boxes are ever created by the actions of $\mathcal{P}_3, \mathcal{P}_4, \ldots$. Thus, $\mathcal{P}_3$ would be injured by C_1^3-boxes in phase 1, by C_0^3-boxes in phase 2, and by C_1^2-boxes in phase 3. The values C_0^3, C_1^3, C_2^3 are defined inductively.

[4] These values are chosen for discussion purposes, and are slightly different in the formal construction. This is because we had not taken into account the other positive requirements.

Each time $\mathcal{P}_e$ is initialized, its threshold would be reset to C^e_{e-2}. With each enumeration that $\mathcal{P}_e$ makes, we will decrease the threshold accordingly ($C^e_{e-3}, \ldots, C^e_0, C^{e-1}_{e-2}$). When moving from phase 1 to 2, a $(C^e_{e-2}+1)$-box can be promoted to a C^e_{e-2}-box but this newly promoted box cannot initialize $\mathcal{P}_e$ while in phase 2 or higher, for its threshold has now decreased to C^e_{e-3} or less. The only way to initialize $\mathcal{P}_e$ after it has made m enumerations, would be through a C^e_{e-2-m}-box (or less). So as long as we keep the critical thresholds values $C^{e-1}_{e-2}, C^e_0, \ldots, C^e_{e-2}$ sufficiently spaced out, we will be all right.

2.4. Notations for the formal construction

Let $size(e, s)$ denote the size of the $\mathcal{N}_e$-box at stage s, i.e. the number of injuries that $\mathcal{N}_e$ can still sustain. At the beginning of the construction, $size(e, 0)$ is set to $h(e)-1$. Each time $\mathcal{N}_e$ is injured, we reduce $size(e, s)$ by 1, and when $size(e, s)$ reaches 0, the $J^A(e)$-computation will have to be preserved forever. During the construction, all parameters retain their assigned values, unless they are initialized or reassigned a different value. We append $[s]$ to an expression to refer to the evaluation of the expression at stage s. On the other hand if the context is clear we drop the stage number from the expression. For example $size(e, s)$ becomes simply $size(e)$.

For each n, let $h^{-1}(n) := \{x \in \mathbb{N} \mid h(x) = n\}$, where h is the order function given. Since h is an order, the set $h^{-1}(n)$ and the value $|h^{-1}(n)|$ are both computable for each n. For each e, let $p(e, s)$ be a counter which records the number of different values the requirement $\mathcal{P}_e$ has managed to put into the trace $T^e_{x(e,s)}$, for the current $x(e, s)$. Note that $p(e, s)+1$ is also the phase number of $\mathcal{P}_e$, as used in the discussion in Section 2.2. Let $S(n) = \sum_{r=1}^{n} r|h^{-1}(r)|$. That is, $S(n)$ denotes the maximum number of different values $j(k, s)$ can take, for the set of k's such that $h(k) \leq n$. This number $S(n)$ is used to give a bound on the number of times a particular $\mathcal{P}_e$ can get injured by some $\mathcal{N}_k$ which is an original $(n-1)$-box or less.

We now define the sequence of numbers $\{C^n_k \mid n \geq 0 \wedge k < n\}$ and the functions $|\mathcal{P}_n|, \tilde{I}_n, I_n : \mathbb{N} - \{0\} \mapsto \mathbb{N}$ as follows. We start with $C^0_{-1} = 0$, and in general for $n \geq 1$, we have:

$$
\begin{aligned}
\tilde{I}_n &= n + \sum_{r=1}^{n-1} r I_r, \\
C^n_0 &= (C^{n-1}_{n-2} + 2) + \sum_{r=1}^{n-1} |\mathcal{P}_r|, \\
&\vdots \\
C^n_{k+1} &= (C^n_k + 2) + \sum_{r=1}^{n-1} |\mathcal{P}_r| + n\tilde{I}_n + nS(C^n_k), \\
I_n &= S(C^n_{n-1}) + \tilde{I}_n, \\
|\mathcal{P}_n| &= (n-1)I_n + 1.
\end{aligned}
$$

We fix the convention $C^n_{-1} = C^{n-1}_{n-2}$. We will explain what each of the symbols represent, for $n \geq 1$. The sequence $C^n_{-1}, C^n_0, \ldots, C^n_{n-2}$ are the different critical threshold values when $\mathcal{P}_n$ is in phase $n, n-1, \ldots, 1$ respectively. The number C^n_{n-1} represents the largest number $b+1$ such that a b-box can interfere with the action of $\mathcal{P}_n$. We obviously do not want $\mathcal{P}_{n+1}$ to promote any box to a b-box, to keep the effects of different $\mathcal{P}$ disjoint; for this reason the critical threshold values of $\mathcal{P}_{n+1}$ are all larger than $b+1$. I_n represents the total number of times $\mathcal{P}_n$ may be initialized, while $\tilde{I}_n$ is the number of times that $\mathcal{P}_n$ may be initialized by the actions of a higher priority $\mathcal{P}_r$. Lastly, $|\mathcal{P}_n|$ is the maximum number of enumerations $\mathcal{P}_n$ may make into A; it may make at most $n-1$ enumerations before it is initialized, plus one more to make it permanently satisfied.

To control the value of the universal jump function $J^A(x)$ of A at different locations x, we will have an infinite list of indices (of Turing functionals) supplied by the recursion theorem. We let Ψ^A_e denote the functional that the requirement $\mathcal{P}_e$ is enumerating at stage s, with index $x(e, s)$ chosen from the list. The value of $x(e, s)$ will change from time to time, more specifically, whenever the requirement $\mathcal{P}_e$ is initialized. At these stages, $\mathcal{P}_e$ will start the enumeration of a new functional, with a new index taken from the list. The use of the functional with index $x(e, s)$

that $\mathcal{P}_e$ is enumerating at stage s is denoted by $u(e, s)$. Thus $u(e, s)$ is a number targeted at A, which we will have to put into A before we can enumerate a new axiom into Ψ_e^A.

In Lemma 2.4(iii) we show that the number of times $\mathcal{P}_e$ can be initialized is bounded by I_e. Thus to define the function $\tilde{h}$, we do the following. We reserve the first $I_1 + 1$ many indices for use by $\mathcal{P}_1$, the next $I_2 + 1$ many indices for $\mathcal{P}_2$, and so on. The function $\tilde{h}$ is defined to have value e on all the indices reserved for $\mathcal{P}_e$. Hence $\tilde{h}$ is clearly an order, and by Lemma 2.4, $\mathcal{P}_e$ only uses indices $x(e)$ which are reserved for it, i.e. it never runs out of indices. Hence, $\tilde{h}(\lim_s x(e, s)) = e$, and it follows that A is not jump-traceable via $\tilde{h}$.

When we *initialize* a requirement $\mathcal{P}_e$ at stage s, we do the following. Do nothing if $p(e, s) = e$ (in this case, $\mathcal{P}_e$ has already been permanently satisfied, and needs to to nothing else). Otherwise, set $x(e, s + 1)$ to be the next value reserved for $\mathcal{P}_e$, and reset the counter $p(e, s + 1)$ to 0.

Definition 2.2. We say that a requirement $\mathcal{P}_e$ *requires attention at stage* s, if $p(e, s) < e$ and one of (ATT1)–(ATT3) holds.

(ATT1) $\mathcal{P}_e$ has been initialized at some stage $t < s$, and has not received attention[5] at any stage u such that $t < u < s$.

(ATT2) All of the following hold:
- There is a computation in Ψ_e^A which currently apply with use $u(e, s)$ and value $r = \Psi_e^A(x(e, s))[s]$,
- For some $k < s$, we have $J^A(k)[s] \downarrow$ with use larger than $u(e, s)$,
- If $e = 1$, we find that $size(k, s) = 0$. Otherwise for $e > 1$, we find that

$$size(k, s) \leq \begin{cases} C^e_{e-2-p(e,s)}, & \text{if } p(e, s) \leq e - 2, \\ C^{e-1}_{e-2}, & \text{if } p(e, s) = e - 1. \end{cases}$$

(ATT3) There is a computation in Ψ_e^A which currently applies with use $u(e, s)$ and value $r = \Psi_e^A(x(e, s))[s]$, such that r has shown up in the trace $T^e_{x(e,s)}$.

If (ATT1) holds, then $\mathcal{P}_e$ has just been initialized, and we need to set $\mathcal{P}_e$ on a new index $x(e)$. If (ATT2) holds, the restraint on $\mathcal{P}_e$ has increased beyond $u(e, s)$, and we would need to initialize it. This would be due to some high priority box blocking $\mathcal{P}_e$. If (ATT3) holds, then the follower $u(e, s)$ has been realized, and we will need to take a positive action to defeat the trace $\{T^e_x\}_{x\in\mathbb{N}}$.

2.5. *Construction of A and* $\{V_e\}_{e\in\mathbb{N}}$

At stage $s = 0$, initialize all requirements and set $size(e, 0) = h(e) - 1$ for all e. At stage $s > 0$, we do the following:

- For all $e < s$ such that $J^A(e)[s] \downarrow$, we enumerate the value $J^A(e)[s]$ into V_e.
- Pick the smallest $e < s$ such that $\mathcal{P}_e$ requires attention at stage s. Take the appropriate action listed below, and declare that $\mathcal{P}_e$ has *received attention* at stage s.

  (1) *(ATT1) holds*: we enumerate a computation $\Psi_e^A(x(e, s))[s] \downarrow = s$ with use $u(e, s) > s$ and $u(e, s) >$ any value previously chosen as a use.

  (2) *(ATT1) fails and (ATT2) holds*: initialize requirement $\mathcal{P}_k$ for all $k \geq e$.

  (3) *(ATT1) and (ATT2) fails but (ATT3) holds*: do all of the following.
  - For each $k < s$ such that $J^A(k)[s] \downarrow$ with use $j(k, s) > u := u(e, s)$, we decrease $size(k, s)$ by 1; these are the boxes which will be promoted.
  - Enumerate u into A to clear the Ψ_e^A-axioms.
  - Increase $p(e, s)$ by 1 to enter the next phase.
  - Define a new computation $\Psi_e^A(x(e))[s] \downarrow = s$ with fresh use $u(e, s)$.
  - Initialize all requirements $\mathcal{P}_k$ for $k > e$.

[5] This will be defined in the construction.

2.6. Verification

In the first lemma we show that the maximum number of different restraints held by any original b-box for any $b < n$ is at most $S(n)$. This is needed for various counting arguments to follow. If X is a set and k is a number then $X\restriction_k$ denotes the string $X(0)\cdots X(k)$.

Lemma 2.3. (i) *For each k, $|\{A_s\restriction_{j(k,s)}: J^A(k)[s]\downarrow \ \wedge\ k < s\}| \leq h(k)$.*
(ii) *For each n, $|\{\langle A_s\restriction_{j(k,s)}, k\rangle : J^A(k)[s]\downarrow \ \wedge\ h(k) \leq n \ \wedge\ k < s\}| \leq S(n)$.*
(iii) *For all e, $\mathcal{N}_e$ is satisfied. Hence, A is jump traceable via h.*

Proof. (i): Suppose on the contrary, there are stages $s_0 < \cdots < s_{h(k)}$ all larger than the number k, such that $A_{s_i}\restriction_{j(k,s_i)} \neq A_{s_{i+1}}\restriction_{j(k,s_{i+1})}$ for all $i = 0,\ldots,h(k)-1$. For each i, the change $A_{s_i}\restriction_{j(k,s_i)} \neq A_{s_{i+1}}\restriction_{j(k,s_{i+1})}$ must have been caused by some positive requirement receiving attention at some stage t in which we also decrease $size(k,t)$, where $s_i \leq t < s_{i+1}$. This means that by the time we reach stage $s_{h(k)-1}$, we have $size(k, s_{h(k)-1}) = 0$, putting it at the highest priority. Hence no enumeration can be made below $j(k, s_{h(k)-1})$ at any stage $t \geq s_{h(k)-1}$, a contradiction.

(ii): By part (i).

(iii): Fix an e, and we want to argue that $\mathcal{N}_e$ is satisfied. If $J^A(e)\downarrow$ then clearly it must be enumerated into V_e after stage e. Suppose that $|V_e| > h(e)$, then $|\{J^A(e)[s] : s > e\}| > h(e)$. Each different value of $J^A(e)[s]$ corresponds to a different string $A_s\restriction_{j(k,s)}$ for the use, contradicting (i). □

The next lemma is the crucial lemma; it argues that the combinatorics we used work out fine. In particular, we argue that no original C^n_k-box can ever be promoted to a C^n_{k-1}-box.

Lemma 2.4. *Let $n > 0$.*

(i) *If $m > C^n_0$, then for all $k \in h^{-1}(m)$ and all s, $size(k,s) > C^{n-1}_{n-2}$.*
(ii) *For each $1 \leq r \leq n-1$, if $m > C^n_r$, then for all $k \in h^{-1}(m)$ and all s, $size(k,s) > C^n_{r-1}$.*
(iii) *The total number of initializations to $\mathcal{P}_n$ is bounded by I_n.*

Proof. We prove (i)–(iii) simultaneously by induction on n. Suppose the results hold for all $n' < n$. Let $\nu(e,s)$ be the critical value function

$$\nu(e,s) := \begin{cases} C^e_{e-2-p(e,s)}, & \text{if } p(e,s) \leq e-2, \\ C^{e-1}_{e-2}, & \text{if } p(e,s) = e-1. \end{cases}$$

We have the following facts:

(Fact 1): For all $n' < n$, the total number of times an enumeration is made into A by $\mathcal{P}_{n'}$, is bounded by $|\mathcal{P}_{n'}|$.
(Fact 2): The total number of stages t such that the requirement $\mathcal{P}_n$ is initialized at stage t, and $\mathcal{P}_n$ *did not* receive attention at stage t, is at most $\tilde{I}_n$. To see this, observe that at such a stage t, it must be that $\mathcal{P}_{n'}$ receives attention for some $n' < n$. At stage t, either $\mathcal{P}_{n'}$ is initialized as well, or else an enumeration is made into A. The total number of such stages t is at most $\sum_{r=1}^{n-1} I_r + \sum_{r=1}^{n-1} |\mathcal{P}_r| < \tilde{I}_n$.

(i): Suppose on the contrary that there is some $m > C^n_0$ such that $size(k,s) \leq C^{n-1}_{n-2}$ for some s and some $k \in h^{-1}(m)$. We may assume that $size(k,s) = C^{n-1}_{n-2}$, since the parameter $size(k)$ is never decreased by more than one at any stage. Since $size(k,0) \geq C^n_0$, there is a stage $s' < s$ such that $size(k,s') = C^n_0$. Suppose $t \geq s'$ is a stage where some positive requirement $\mathcal{P}_e$ receives attention in which $size(k,t)$ is decreased. It cannot be that $e > n$, because $\nu(e,t) \geq C^n_0 \geq size(k,t)$; $\mathcal{P}_e$'s actions are kept from interfering with $\mathcal{P}_n$'s. If $e = n$, then $p(e,t)$ has to be $e-1$ and hence there can only be at most one such stage t where $\mathcal{P}_n$ receives attention and decreases $size(k,t)$ (since $p(e,t)$ is increased to e at stage t, and $\mathcal{P}_n$ becomes permanently satisfied after that). Hence, the number of times $size(k,t)$ can be decreased after stage s', is at most $1 + \sum_{r=1}^{n-1} |\mathcal{P}_r|$. This shows that the smallest value $size(k,t)$ can take at any stage $t \geq s'$, is $C^{n-1}_{n-2} + 1 > size(k,s)$, a contradiction.

(ii): The proof is more or less similar to (i), with more cases involved. Suppose on the contrary that there is some $m > C^n_r$ such that $size(k,s) = C^n_{r-1}$ for some s and some $k \in h^{-1}(m)$. Let $s' < s$ be such that $size(k,s') = C^n_r$. Suppose $t \geq s'$ is a stage where some positive requirement $\mathcal{P}_e$ receives attention in which $size(k,t)$ is decreased. Similar to (i), we will now count the maximum number of such stages t. It is clear that $e \not> n$, and there can be at

most $\sum_{r=1}^{n-1} |\mathcal{P}_r|$ many stages t where $\mathcal{P}_e$ receives attention in which $size(k,t)$ is decreased and $e < n$. Lastly if t is a stage where $\mathcal{P}_e$ receives attention, $size(k,t)$ is decreased and $e = n$, we must have $p(e,t) > n-2-r$. We split the counting into the following cases:

- *Case 1: t is a stage such that $\mathcal{P}_n$ receives attention in which $size(k,t)$ is decreased and $p(n,t) = n-1$.*
  As in (i), there can only be one such t.
- *Case 2x: t is a stage such that $\mathcal{P}_n$ receives attention in which $size(k,t)$ is decreased and $p(n,t) = x$, where $n-1-r \leq x \leq n-2$.*
  At stage t, $p(n,t)$ is set to $x+1$. In order that there is a next stage such that Case 2x applies again, $\mathcal{P}_n$ will have to be initialized — we can blame this on a small sized box. There are (by Fact 2) at most $\tilde{I}_n$ many times where $\mathcal{P}_n$ can be initialized by some other $\mathcal{P}$. If $\mathcal{P}_n$ is initialized at a stage $t' > t$ where it receives attention, then $p(e,t') \geq x+1 > n-1-r$. Hence there is some $k' < t'$ such that $J^A(k')[t'] \downarrow$, and

$$size(k',t') \leq \begin{cases} C^e_{r-2}, & \text{if } r > 1, \\ C^{e-1}_{e-2}, & \text{if } r = 1. \end{cases}$$

  In any case by (i) and induction hypothesis of (ii), we have $h(k') \leq C^e_{r-1}$ (blaming a box of small size), and by Lemma 2.3(ii) there can only be at most $\tilde{I}_n + S(C^e_{r-1})$ many t's where Case 2x applies.

Putting together the above calculations, we see that the smallest value $size(k,t)$ can take at any stage $t \geq s'$, is $C^n_r - \sum_{r=1}^{n-1} |\mathcal{P}_r| - 1 - n(\tilde{I}_n + S(C^e_{r-1})) = C^n_{r-1} + 1 > size(k,s)$, a contradiction.

(iii): There are (by Fact 2) at most $\tilde{I}_n$ many times where $\mathcal{P}_n$ can be initialized without it receiving attention. If $\mathcal{P}_n$ is initialized at a stage t where it receives attention, there is some $k < t$ such that $J^A(k)[t] \downarrow$, and

$$size(k,t) \leq \begin{cases} C^n_{n-2}, & \text{if } n > 1, \\ C^0_{-1}, & \text{if } n = 1. \end{cases}$$

It follows by (i) and (ii) that $h(k) \leq C^n_{n-1}$, and by Lemma 2.3(ii) there can be at most $\tilde{I}_n + S(C^n_{n-1}) = I_n$ initializations to $\mathcal{P}_n$. □

Lemma 2.5. *For all e, $\mathcal{P}_e$ is satisfied.*

Proof. Fix an e, and let $x := \lim_{s\to\infty} x(e,s)$, which exists. Suppose that $J^A(x) \downarrow \in T^e_x$. Let $s_0 > e$ be a stage large enough so that we have $x(e,s_0) = x$, $J^A(x) \in T^e_x[s_0]$, and $\mathcal{P}_e$ never requires attention after stage s_0. Hence, it must be the case that $\Psi^A_e(x)[s_0] \downarrow = J^A(x)$, and also that $p(e,s_0) = e$ (else (ATT3) holds at stage s_0). This means that there are e many different stages t (before stage s_0) where $\mathcal{P}_e$ receives attention, and $p(e,t)$ is increased by 1. At each such stage t we also enumerate a new value for $\Phi^A_e(x)[t]$, and wait for it to be traced. By stage s_0, we have $|T^e_x[s_0]| > e$. □

Lemma 2.4(iii) actually establishes that $\mathcal{P}_e$ never uses a forbidden index (an index not meant for it). Hence $\tilde{h}(x(e,s)) = e$ for all e and s, and it follows that A is not jump-traceable via $\tilde{h}$.

3. The c.e. strongly jump traceable sets are Π^0_4-complete

It is easy to see that the index set concerned is Π^0_4:

Lemma 3.1. *The set $\{e \in \mathbb{N} : W_e$ is strongly jump traceable$\}$ is Π^0_4.*

Proof. W is strongly jump traceable $\Leftrightarrow \forall e\,(h_e$ is an order $\Rightarrow \exists k \forall x\, Q(e,k,x))$, where the predicate Q is

$$Q(e,k,x) = g_k(x) \downarrow \text{ and } |W_{g_k(x)}| \leq h_e(x) \text{ and } J^W(x) \downarrow \Rightarrow J^W(x) \in W_{g_k(x)},$$

and $\{h_e\}_{e\in\mathbb{N}}$ and $\{g_k\}_{k\in\mathbb{N}}$ are effective listing of all partial computable functions. Clearly "h_e is an order" is a Π_2 fact, while $Q(e,k,x)$ is a Π_2 predicate. □

To perform the coding of a Π^0_4 set into the index set of the c.e. strongly jump traceable sets, we will need to use the following lemma; the complete proof can be found in Nies [10]:

Lemma 3.2. *If S is a Π^0_4 set, there is a u.c.e. (uniformly computably enumerable) sequence $X_{y,e,p}$ of initial segments of ω, such that*

$$y \in S \Rightarrow \forall e \exists p (X_{y,e,p} = \omega),$$
$$y \notin S \Rightarrow \text{For almost all } e \text{ and } p, |X_{y,e,p}| < \infty.$$

Proof. If S is a Π^0_4 set, then there is a u.c.e. sequence $\bar{X}_{y,e,p}$ of initial segments of ω, such that $y \in S \Leftrightarrow \forall e \exists p (\bar{X}_{y,e,p} = \omega)$. Observe that it is easy for us to define a new u.c.e. sequence $Y_{y,e,p}$, such that if it is the case that for some e, $|\bar{X}_{y,e,p}| < \infty$ for all p, then we have $|Y_{y,e,p}| < \infty$ for all p and all $e' \geq e$. On top of doing that, we also need to ensure that each sequence $\{Y_{y,e,p}\}_{p \in \mathbb{N}}$ has got at most one infinite set. To do this, we replace each sequence $\{Y_{y,e,p}\}_{p \in \mathbb{N}}$ by the sequence $\{X_{y,e,\langle p,b\rangle}\}_{p,b \in \mathbb{N}}$, where for each p, the set $X_{y,e,\langle p,b\rangle}$ is allowed to copy $Y_{y,e,p}$ if $b = |Y_{y,e,0} \cup \cdots \cup Y_{y,e,p-1}|$. Clearly for each y and e, $X_{y,e,r}$ is infinite for at most one r. □

We will devote the rest of this section to the proof of

Theorem 3.3. *The set $\{e \in \mathbb{N} : W_e$ is strongly jump traceable$\}$ is Π^0_4-complete.*

3.1. Requirements and an overview

We let S be a Π^0_4 set, and let $\{X_{y,e,p}\}$ be the corresponding sequence in Lemma 3.2. Fix a y, and we shall build a c.e. set A, such that

$$\forall e \exists p (X_{y,e,p} = \omega) \Rightarrow A \text{ strongly jump traceable,}$$
$$\forall^\infty e,\ p(|X_{y,e,p}| < \infty) \Rightarrow A \text{ is not jump traceable via some order.}$$

The requirements are:

$$\mathcal{R}_{e,p} : \text{If } |X_{y,e,p}| = \infty \text{ and } h_e \text{ is an order, then make } A \text{ jump traceable via } h_e.$$

The requirement $\mathcal{R}_{e,p}$ also has a positive role, namely

$$\mathcal{R}_{e,p} : \text{If } |X_{y,e,p}| < \infty, \text{ make } A \text{ not strongly jump traceable by defeating } \{T^k_x\}_{x \in \mathbb{N}} \text{ for some } k, \text{ with respect to some order.}$$

Here, we let $\{h_e\}_{e \in \mathbb{N}}$ be an effective list of all partial computable functions such that $\forall e \forall n > 0\ (0 < h_e(n) \leq n)$. We shorten notation, and write $X_{e,p}$ for $X_{y,e,p}$, since y is fixed in the construction. We identify sets with their characteristic functions, and initial segments of functions with finite strings. That is, when we write $A\restriction_k$, we mean the finite string $A(0)A(1)\cdots A(k-1)$.

Basically, Lemma 3.2 helps us to arrange the coding requirements on the construction tree, which is a tree of strategies, in the style of a $\emptyset'''$-priority argument. This method is originally due to Lachlan [8], and is also presented in Chapter XIV.4 of [21]. The reader is assumed to be familiar with standard tree arguments, a good exposition on this topic can be found in [20]. The true path of the construction is defined as usual, as the leftmost path visited infinitely often during the construction.

We will use the construction in Theorem 2.1 as an atomic strategy in this construction. The discussion on how this is to be carried out will be developed over the next few pages. For now, we first give the reader a brief preview of how this is to be done. Basically we want to code a given Π^0_4 set S into the index set of the strongly jump traceable sets. For each y, we perform a separate construction, and uniformly produce a set A at each construction. The trouble is that $y \in S$ or $y \notin S$ is a Π^0_4/Σ^0_4 fact. We arrange for guesses to take place on the construction tree, and with the help of Lemma 3.2, the true path of the construction will reflect whether or not $y \in S$. However, the construction, having to be effective, can only act on approximations to the true path. Hence there will be some stages where y looks like it is in S; at these stages we try and make A strongly jump traceable. At other stages, y will look like it is out of S. At these stages, we will try and make A not jump traceable via some order function $\tilde{h}$ which we build. However, $y \notin S$, being a Σ^0_4 fact, can still cause us to have finitely many objects exhibiting infinitary behaviour. Each of these will force us to make A jump traceable via some order h_e. Therefore, at each stage of the construction where y looks like it is out

of S, we have to make A jump traceable via some collection $h_{e_0}, \ldots, h_{e_k}$ of orders, and run the previous construction to make A *not* jump traceable via some $\tilde{h} \ll \min\{h_{e_0}, \ldots, h_{e_k}\}$.

The facts "$X_{y,e,p} = \omega$" and "$X_{y,e,p} < \infty$" are Π^0_2 and Σ^0_2 facts respectively, so each such statement can be measured at a single node on the construction tree, measuring infinitary or finitary behaviour. Lemma 3.2 says that the Π^0_4 fact "$y \in S$" can be broken down into a Π^0_2 statement regarding the true outcomes of nodes on the construction tree. That is, $y \in S$ would be equivalent to the fact that "$\forall e \exists p$ such that the node measuring $|X_{y,e,p}|$ has true infinitary outcome". Similarly, if $y \notin S$ then "for almost all e, p, the node measuring $|X_{y,e,p}|$ has true finitary outcome". Hence $y \in S$ or $y \notin S$ will determine the true path of the construction, i.e. which nodes are visited infinitely often and which are not. We will arrange the strategies on the construction tree to align our actions with the true path.

3.2. Description of strategies

Each node α on the construction tree has two different strategies. Suppose α is assigned the requirement $\mathcal{R}_{e,p}$, then α will have to test if $|X_{e,p}| = \infty$. Each time some number enters $X_{e,p}$, α will have to direct its efforts towards making A h_e-jump traceable (corresponding to applying negative restraints on A). This is represented by the infinitary outcome. If some time has elapsed with no changes in $|X_{e,p}|$, then α will have to try and make A not strongly jump traceable, by attempting to defeat some trace $\{T_x\}_{x\in\mathbb{N}}$ (corresponding to taking positive action on A). This is represented by the finitary outcome f. Each node α has a dual role — at expansionary stages when $|X_{e,p}|$ increases, α pursues the *negative strategy*, while at nonexpansionary stages α pursues the *positive strategy*.

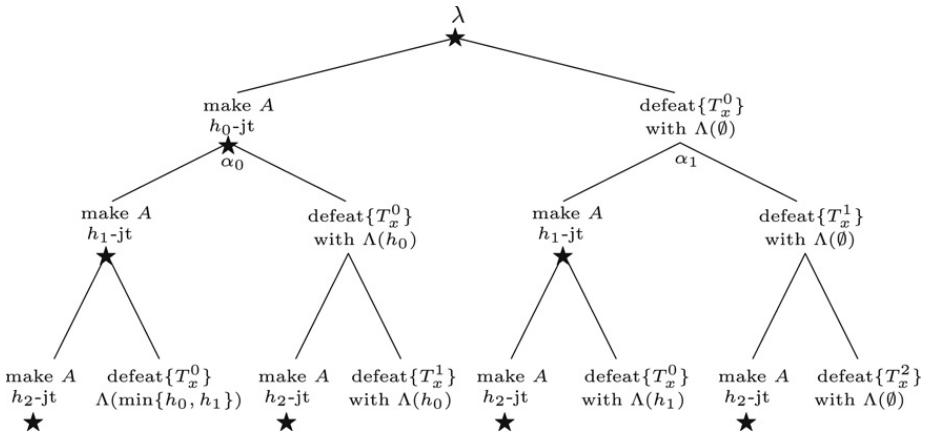
The negative α-strategy (to ensure h_e-jump traceablility) is the usual. It splits its task into infinitely many substrategies $ST_0, ST_1, \ldots$. For each $k \in \mathbb{N}$, the kth substrategy ST_k works by the following: it waits for $h_e(k) \downarrow$, and when $J^A(k)[s]$ next converges, we would enumerate the value into V^α_k (the sequence $\{V^\alpha_x\}_{x\in\mathbb{N}}$ is build at α), and restraint A on the use. At this time, we set $size^\alpha_k = h_e(k) - 1$ (to indicate that V^α_k can take at most that many more injuries). When the $size^\alpha_k = 0$, V^α_k is totally filled and any restraint α imposes for it must be permanent. If we arrange for each substrategy ST_k to be assigned to an entire level below α, we immediately meet with a technical obstacle. Recall that in Theorem 2.1, the positive requirements had priority (relative to some ST_k), which was determined dynamically. This would not be easy to arrange on a tree of strategies.

Note that we could however, arrange for *all of* the α substrategies to be carried out at α itself. This means that α could impose an ever increasing restraint on the positive strategies below it, even though each substrategy ST_k contributes a finite amount. To get around this problem, we arrange for there to be infinitely many restraint functions $r_0, r_1, \ldots$, where r_k is the restraint function for ST_k, and let different positive strategies be restrained by a different r_k. Suppose each positive strategy below α only wants to enumerate once. We could then let the first positive strategy obey restraint r_0, the second positive strategy obey restraints $\max\{r_0, r_1\}$, and so on. This would be fine if h is the identity order function (otherwise we just adjust accordingly). For details of this, see Theorem 7.3 of [3], where this idea was used to give a direct construction of a noncomputable c.e. set which is strongly jump traceable. Carrying out all the α-substrategies at α has the effect of complicating the mechanism at a single node, but simplifies the global considerations and the notations. Each node α on the construction tree builds an entire sequence $\{V^\alpha_x\}_{x\in\mathbb{N}}$ — each α makes a separate attempt to ensure jump traceability at some order.

In Theorem 2.1 we had shown that there is an effective procedure Λ, such that given any order function h, the procedure $\Lambda(h)$ outputs a set A and an order function $\tilde{h}$ such that A is h-jump traceable but not $\tilde{h}$-jump traceable. We will attempt to repeat this construction Λ at each node α. At stages when α runs its positive strategy, we would make α diagonalize some trace $\{T_x\}_{x\in\mathbb{N}}$. The same atomic strategy is used for this: we can take control of $J^A(x)$ for some x. Enumerate $J^A(x)[s] \downarrow = s$ with use u. Each time $J^A(x)[s]$ appears in T_x, we put u into A and set $J^A(x)[s'] \downarrow$ with another value. If we do this n times (for some chosen n), we would succeed in diagonalizing against $\{T_x\}$ at some order $\tilde{h}$, which we build.

The positive and negative strategies clearly conflict with each other. Why is it not possible to simultaneously run the positive and negative strategies of all requirements? This is equivalent to making A strongly jump traceable, *and* not $\tilde{h}$-jump traceable for some $\tilde{h}$. The trouble is that the eth positive strategy has to obey restraints set up for the sake of making A jump traceable via $\min\{h_0, \ldots, h_e\}$. Even though the threshold values defined in Section 2.4 are fixed in advance, it is not possible to compute $|h^{-1}_k(r)|$ for every k and r. We really have to make guesses as to whether or not each h_k is an order, and so the $\tilde{h}$ built this way will have to be computable in a $\emptyset''$-oracle. This is explored in [9], where the relativization of strong jump traceability is studied.

To make sure $\tilde{h}$ is computable, we will build a different version of $\tilde{h}$ at each of infinitely many nodes on the construction tree. These nodes are called *top nodes*. If τ is a top node, and lives below nodes assigned to requirements $\mathcal{R}_{e_0,p_0},\ldots,\mathcal{R}_{e_j,p_j}$, such that τ believes that $h_{e_0},\ldots,h_{e_j}$ are all orders, then the positive strategy of τ would repeat the construction Λ and make A not jump traceable with respect to the order function $\Lambda(g)$, where $g=\min\{h_{e_0},\ldots,h_{e_j}\}$. The following shows the tree of strategies, and how the strategies may be arranged. In the following diagram, λ represents the root node (the empty string). The left branch represents the infinitary outcome which will be visited during expansionary stages, while the right branch represents finitary outcomes at non-expansionary stages. For instance, at λ-expansionary stages, we will run the negative λ-strategy of making A jump traceable (abbreviated below by jt) via h_0. At non- λ-expansionary stages we will run the positive λ-strategy of trying to defeat the trace $\{T^0_x\}_{x\in\mathbb{N}}$, by applying the procedure $\Lambda(\emptyset)$.



The top nodes are marked out in the diagram above using the star symbol. At each of these top nodes, a new application of Λ is started, with a new order function $\tilde{h}$. For instance, λ is a top node where Λ is applied at non-λ-expansionary stages to diagonalize the trace $\{T^0_x\}_{x\in\mathbb{N}}$. The λ-counterexample to strong jump traceability would be the order function $\tilde{h}_\lambda=\Lambda(\emptyset)$, since λ has no stronger priority order functions to respect. The node α_0 gets to act at λ-expansionary stages, and therefore will have to live in harmony with λ's negative strategy. Thus α_0 will start its own version of Λ by building the counterexample $\tilde{h}_{\alpha_0}=\Lambda(h_0)$, at non-$\alpha_0$-expansionary stages. On the other hand, the node α_1 would continue λ's positive strategy by diagonalizing the trace $\{T^1_x\}_{x\in\mathbb{N}}$ at the same order $\tilde{h}_\lambda$, at non-α_1-expansionary stages. α_1 is said to be a *child* of λ. λ's other children are $\alpha^\frown f, \alpha^\frown f^\frown f,\ldots$ (where $^\frown$ is the string concatenation operator), which will all help to carry out the λ-version of procedure Λ, by diagonalizing the traces $\{T^2_x\}, \{T^3_x\},\ldots$ respectively, at their nonexpansionary stages.

If $y\in S$, then there will be infinitely many nodes α on the true path with true infinitary outcome. At these nodes we would succeed in making A jump traceable via arbitrarily slow growing orders, and so A would be strongly jump traceable. On the other hand, if $y\notin S$, then there is a maximal node τ^- on the true path which has true infinitary outcome, and all of its successors $\tau, \tau^\frown f, \tau^\frown f^\frown f,\ldots$ will have true finitary outcome. In this case, τ is the *final top node*. Each of the τ-children will stop running its negative strategy after a finite number of stages have elapsed. The set A would not be jump traceable via the counterexample h_τ.

There is a major modification to the basic construction Λ we have to make to ensure it runs smoothly on a tree of strategies. Suppose τ is a top node running a version of Λ, at the order $\tilde{h}_\tau$. Let $\alpha_1\subset\alpha_2\subset\cdots$ be the τ-children nodes, where α_n is devoted to the diagonalization of $\{T^n_x\}$ at nonexpansionary stages.

There are two instances where we have to choose a new index x for α_n. The first happens when some $\beta\subset\tau$ increases A-restraint while running its negative strategy — as we have seen in Theorem 2.1, this is expected and τ is perfectly prepared for it by choosing the thresholds $C^n_0,\ldots,C^n_{n-1}$ to be sufficiently spaced out. The second case happens due to the fact that requirements are now arranged on a tree: at each α_i-expansionary stage (for some $i\le n$), we would also have to choose a new index x' for α_n because stronger requirements have now acted and may need to be respected. Even though this happens only finitely often, we do not know how often. Thus it now becomes impossible for us to know how many indices to set aside for α_n.

The solution is to divide the indices into intervals called *regions*. The mth region contains all the indices x such that $\tilde{h}_\tau(x)=m$, based on the calculations in Section 2.4. Instead of getting α_n to use indices from the nth region all

the time, we would make α_n move on to a new region number n', at each α_i-expansionary stage, $i \leq n$. Thus, each time α_n has to choose a new index due to reasons that τ is not prepared for, we would start with the smallest index in a fresh region. This ensures that α_n never runs out of indices.

3.3. Construction tree layout

Due to technical reasons, the construction will take place on the full ternary tree $3^{<\omega}$, instead of a binary tree as discussed Section 3.2. This is because before a node can begin its positive strategy, it would have to first compute all the relevant threshold values, and thus it would have to know which functions are real orders.

Nodes of length $\langle e, p\rangle$ are assigned the requirement $\mathcal{R}_{e,p}$, with three outcomes: $\infty\infty < \infty f < f$. Outcome f means that we believe $X_{e,p}$ is finite, and we have to start the positive strategy. Outcome $\infty\infty$ means that $X_{e,p}$ is infinite, and h_e is an order. Thus, we have to continue with the negative strategy of making A h_e-jump traceable. Finally, outcome ∞f means that we believe $X_{e,p}$ is infinite, but h_e is not an order. In this case, we have to initialize every node which believes that $X_{e,p}$ is finite, and do nothing else. Let $\alpha <_{left} \beta$ denote that α is strictly to the left of β (i.e. there is some $i < \min\{|\alpha|, |\beta|\}$ such that $\alpha\restriction_i = \beta\restriction_i$ and $\alpha(i) < \beta(i)$). The construction tree grows downwards. We write "α is a $\mathcal{Q}$-node" to denote the fact that α is assigned the requirement $\mathcal{Q}$.

3.4. Notations

If α is a $\mathcal{R}_{e,p}$-node, we let $order(\alpha) = e$. We also let $Z^-(\alpha) := \{\beta \subset \alpha \mid \beta^\frown\infty\infty \subseteq \alpha\}$, which are all the nodes running the negative strategy extended by α. Similarly we let $Z^+(\alpha) := \{\beta \subset \alpha \mid \beta^\frown f \subseteq \alpha\}$, which are the nodes attempting the positive strategy extended by α. For each node α, we define $trace(\alpha)$ by the following: $trace(\lambda) = 0$, and for α of positive length, we let α^- be the predecessor of α, and set

$$trace(\alpha) = \begin{cases} 1 + trace(\alpha^-), & \text{if } \alpha(|\alpha^-|) = f, \\ 0, & \text{otherwise.} \end{cases}$$

$trace(\alpha)$ denotes the number k such that α needs to defeat the kth trace $\{T^k_x\}_{x\in\mathbb{N}}$ for its positive strategy. We say that α is a *top node*, if $trace(\alpha) = 0$. For any node α on the tree, we define $\tau(\alpha)$, the *top of* α to be the maximal $\tau \subseteq \alpha$ such that τ is a top node. We say that α is a *child* of τ if α has top τ. Thus the children of τ are exactly $\tau \subset \tau^\frown f \subset \tau^\frown f^\frown f \subset \cdots$.

Each $\mathcal{R}_{e,p}$-node α has a number of parameters associated with it. The parameters used for its positive strategy are the following:

- If α is a top node, it will build a partial order function $\tilde{h}_\alpha$. We will get a chance to extend $dom(\tilde{h}_\alpha)[s] := \{x \in \mathbb{N} : \tilde{h}_\alpha(x)[s]\downarrow\}$, whenever one of α's children begins its positive strategy.
- $x(\alpha, s)$, which denotes the index of the functional that α is currently enumerating at stage s. It will try and cause $T^{trace(\alpha)}_{x(\alpha,s)}$ to fill up with numbers.
- $u(\alpha, s)$, which is the use of the most recent computation α has enumerated into $J^A_{x(\alpha,s)}$. This is a number pointing at A, which α may decide at a later stage to put into A, when the trace $T^{trace(\alpha)}_{x(\alpha,s)}$ increases in size.
- $region(\alpha, s)$, which denotes the number of elements that α has to try to fill $T^{trace(\alpha)}_{x(\alpha,s)}$ up with.
- $attempt(\alpha, s)$, this is a counter which reflects the progress of α in its positive strategy. This plays the same role as the parameter $p(e, s)$ in Theorem 2.1.

The parameters associated with the negative strategy of α are the following:

- At α, we build a uniformly c.e. sequence $\{V^\alpha_k\}_{k\in\mathbb{N}}$. This will trace J^A in the event that $\infty\infty$ is its true outcome.
- We let $size^\alpha_k[s]$ denote the size of the V^α_k-box at stage s, similar to Theorem 2.1. It records the number of injuries the V^α_k-box can still take. At the beginning, $size^\alpha_k$ is set to $h_e(k) - 1$, and will be reduced by 1 each time a $J^A(k)$-computation is injured after being traced in V^α_k.

For $e \in \mathbb{N}$, we define the length of convergence for h_e at stage s, to be

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

We will sometimes write $l(\alpha, s)$ instead of $l(e, s)$. For each $n, s \in \mathbb{N}$, let

$$S(\alpha, n)[s] = \sum_{r=1}^{n} r \cdot |\{k < l(\alpha, s) : size_k^{\alpha}[s] = r-1\}|.$$

Furthermore, if τ is a top node, we let

$$S(\tau, n)[s] = \sum_{\beta \in Z^-(\tau)} S(\beta, n)[s].$$

This has the same intended purpose as the parameter $S(n)$ of Theorem 2.1, with two marked differences. First, each τ has to now consider the total effect of all $\{V_k^{\beta}\}$ for every $\beta \subset \tau$ which it believes will make A jump traceable via $h_{order(\beta)}$ (unlike in Theorem 2.1, where we only had a single order to consider). Second, because the strategy of τ is based on the fact that its guesses are correct, so τ has to wait for various computations to converge before it can proceed further. Therefore, threshold values will have to be computed during the construction itself, and their values will depend on the current situation. In particular, the cardinality in the sum of $S(\alpha, n)[s]$ is computed using *current* $(r-1)$-*boxes*, instead of using original $(r-1)$-boxes as in Theorem 2.1.

Suppose τ is a top node. The parameters $\{C_{n,k}^{\tau} \mid n > 0 \wedge k < n\}$ and $\{I_n^{\tau} \mid n > 0\}$ helps us keep track of the threshold values, and as mentioned above, will be computed during the construction. These parameters are all set to $\uparrow$ initially. The values $C_{n,1}^{\tau}, \ldots, C_{n,n-1}^{\tau}$, and I_n^{τ} are associated with the nth region, similar to their counterparts in Theorem 2.1.

There are infinitely many indices $\nu_0 < \nu_1 < \cdots$ set aside for use by τ-children. If α is a child of τ, then $x(\alpha)$ will be chosen from this list. $\tilde{h}_{\tau}$ will be set to a constant value over each interval $\{x \in \mathbb{N} \mid \nu_{i-1} < x \leq \nu_i\}$, for $i \in \mathbb{N}$. Thus, whenever we refer to $x(\alpha)$ or $\tilde{h}_{\tau}$, we mean the values modulo intervals partitioned by the ν_i's. That is, $\tilde{h}_{\tau}(i)$ will refer to the (common) value of $\tilde{h}_{\tau}(x)$ for $\nu_{i-1} < x \leq \nu_i$, and we write $x(\alpha) = i$ instead of $x(\alpha) = \nu_i$. We let Ψ_{α}^{A} denote the functional α is enumerating at stage s, with index $x(\alpha, s)$. The value of $x(\alpha, s)$ will change from time to time, specifically at those stages when α is initialized or reset (the meaning of these two terms will be explained soon). At these stages, α will start enumeration of a new functional, with a new index. α will pick the new index according to the following:

- If it is the case that α is reset, we increment $x(\alpha)$ by 1.
- If α is initialized, it will be asked to start on a fresh region n. In this case, we will set $x(\alpha) = n$.

For a node α and stage s, we define $threshold(\alpha, s)$ by

$$threshold(\alpha, s) = C_{r,r-2-a}^{\tau(\alpha)},$$

where $r = region(\alpha, s)$ and $a = attempt(\alpha, s)$. This refers to the current threshold value that α has to obey. When we *initialize* α at stage s, we do the following:

- Pick a fresh number n for $region(\alpha)$.
- Set $attempt(\alpha) = 0$.
- Set $x(\alpha) = n$.
- Set $u(\alpha) = \uparrow$.
- Set $C_{n,-1}^{\tau(\alpha)} = n$, and $C_{n,0}^{\tau(\alpha)} = n+2$.

That is, α has to restart its negative strategy due to reasons that $\tau(\alpha)$ had not foreseen. If α is injured because of activity above $\tau(\alpha)$, we will *reset* α (at stage s) by doing the following: If $attempt(\alpha, s) = region(\alpha, s)$ (α's negative strategy has succeeded) or $x(\alpha) = region(\alpha, s) + I_{region(\alpha,s)}^{\tau(\alpha)}$ (i.e. α has run out of indices), do nothing. Otherwise increase $x(\alpha)$ by 1, set $u(\alpha) = \uparrow$, and set $attempt(\alpha) = 0$.

Definition 3.4. We say that a node α *requires positive attention* at stage s, if $attempt(\alpha, s) < region(\alpha, s)$, and one of the following (ATT0)–(ATT3) holds.

(ATT0) One of $C^{\tau(\alpha)}_{n,1}, \ldots, C^{\tau(\alpha)}_{n,n-1}$, or $I^{\tau(\alpha)}_n$ has not yet received a value, where $n = \mathit{region}(\alpha, s)$.

(ATT1) There is no computation in Ψ^A_α which currently applies.

(ATT2) All of the following hold:
- there is a computation in Ψ^A_α which currently applies with use $u(\alpha, s)$,
- there is some β and k such that $\beta \in Z^-(\alpha)$, and $k < l(\beta, s)$, and we have $J^A(k)[s]\downarrow$ with use larger than $u(\alpha, s)$,
- $\mathit{size}^\beta_k[s]\downarrow \leq \mathit{threshold}(\alpha, s)$.

(ATT3) There is a computation in Ψ^A_α which currently apply with use $u(\alpha, s)$ and value $r = \Psi^A_\alpha(x(\alpha, s))[s]$, such that r has shown up in the trace $T^{\mathit{trace}(\alpha)}_{x(\alpha,s)}$.

If (ATT0) holds, then α is not yet ready to start its positive strategy; wait until all the relevant parameters have been defined. If (ATT1) holds, we need to place an axiom into Ψ^A_α. If (ATT2) holds, the restraint on α from β above has increased beyond $u(\alpha, s)$ — there is some high priority box blocking the positive strategy of α. We have to reset α. If (ATT3) holds, we will need to take positive action to defeat the $\mathit{trace}(\alpha)$th trace.

3.5. Construction of A

At each stage s of the construction, we will define the approximation to the true path of the construction, δ_s of length $< s$. We say that α *is visited* at stage s, if $\delta_s \supset \alpha$. We will also state the actions to be taken by the nodes on δ_s. At stage $s = 0$, set $\delta_s = \lambda$ and do nothing else.

Let $s > 0$, and assume that $\alpha = \delta_s\restriction_d$ has been defined for $d < s$. Suppose α is a $\mathcal{R}_{e,p}$-node. We now have to determine which of the three outcomes to take. Check if $|X_{e,p}|$ has increased since the last visit to α. If not, let $\delta_s(d) = f$. Otherwise, we let $t < s$ be the stage number of the most recent visit to α in which $\delta_t(d) \neq f$. If:

- $l(\alpha, t) < l(\alpha, s)$, and
- $h_e(l(\alpha, s)) > C^\tau_{n,r}$ for every $n, r \in \mathbb{N}$, and every top node $\tau \supseteq \alpha^\frown\infty\infty$,

we will let $\delta_s(d) = \infty\infty$. Otherwise let $\delta_s(d) = \infty f$. Let s^- be the stage number of the previous visit to α. If:

- s^- does not exist (i.e. this is the first visit to α), or
- $\delta_t <_{\mathit{left}} \alpha$ for some $s^- < t < s$, or
- some $\sigma \in Z^+(\alpha)$ enumerates in A between the two visits to α at s^- and s,

we will initialize α and set $V^\alpha_k = \emptyset$ for all k, and set $\mathit{size}^\alpha_k = h_e(k) - 1$ for all $k < l(\alpha, s)$. Next, we will take the corresponding actions depending on the outcome $\delta_s(d)$ of α determined above:

(1) $\delta_s(d) = \infty\infty$: we run the negative strategy for α. For all $k < l(\alpha, s)$ such that $J^A(k)[s]\downarrow$, we enumerate the value $J^A(k)[s]$ into V^α_k. For all $k < l(\alpha, s)$ such that size^α_k has not yet been assigned a value, we do the update $\mathit{size}^\alpha_k = h_e(k) - 1$.

(2) $\delta_s(d) = \infty f$: do nothing.

(3) $\delta_s(d) = f$: we run the positive strategy for α. If $\delta_{s^-}(d) \neq f$, we initialize α. Next, if α does not require positive attention, we do nothing. Otherwise, take the appropriate action listed below, and declare that α has *received positive attention* at stage s.
   (a) *(ATT0) holds*: if α has just been initialized at this current stage, terminate the definition of δ_s, do nothing else and go to stage $s + 1$. Otherwise, do the following:
   - Pick the smallest $r < n := \mathit{region}(\alpha, s)$ such that $C^{\tau(\alpha)}_{n,r}\uparrow$, and set $C^{\tau(\alpha)}_{n,r} = C^{\tau(\alpha)}_{n,r-1} + 2 + n + nS(\tau(\alpha), C^{\tau(\alpha)}_{n,r-1})[s]$.
   - If $C^{\tau(\alpha)}_{n,r}\downarrow$ for all $r < n$, set $I^{\tau(\alpha)}_n = S(\tau(\alpha), C^{\tau(\alpha)}_{n,n-1})[s]$. Additionally, we set $\tilde{h}_{\tau(\alpha)}(x) = n$ for all $x \leq n + I^{\tau(\alpha)}_n$ such that $x \notin \mathit{dom}(\tilde{h}_{\tau(\alpha)})[s]$.
   - We terminate the definition of δ_s, and go to stage $s + 1$.

   (b) *¬(ATT0)∧(ATT1)*: we enumerate a computation $\Psi^A_\alpha(x(\alpha, s))[s]\downarrow = s$ with fresh use $u(\alpha, s)$.

   (c) *¬(ATT0)∧¬(ATT1)∧(ATT2)*: reset α.

(d) $\neg(ATT0)\wedge\neg(ATT1)\wedge\neg(ATT2)\wedge(ATT3)$: for each k such that there is some $\beta \in Z^-(\alpha)$ and $k < l(\beta, s)$, such that $J^A(k)[s]\downarrow$ with use $j(k,s) > u := u(\alpha, s)$, we decrease $size_k^\beta[s]$ by 1. Enumerate u into A, and increase $attempt(\alpha, s)$ by 1.

This ends the construction. The purpose of terminating the definition of δ_s under step 3(a), is because we do not want the other $\tau(\alpha)$-children to pick their threshold values before α finishes with its own. The true path of the construction is defined as the leftmost path visited infinitely often during the construction.

3.6. Verification

The following fact is rather obvious, but important: if a node σ is initialized and $region(\sigma)$ set to n, then σ can make at most $1 + (n-1)(I_n^{\tau(\sigma)} + 1)$ many enumerations into A before it is next initialized.

Since δ_s can sometimes have length $< s$, we have to see that the true path of the construction exists, and is infinitely long: suppose α is a node visited infinitely often, and $\delta_t <_{left} \alpha$ for finitely many t. We need to see that only finitely often, α is visited and the definition of δ_s is terminated at α. This is an issue only when α stops playing the outcomes $\infty\infty$ and ∞f after a finite number of stages. Clearly each $\sigma \in Z^+(\alpha)$ enumerates into A only finitely often, by the above fact. Hence α is initialized only finitely often, and so step 3(a) of the construction applies only finitely often.

Thus, for each node α on the true path, we can let $True(\alpha)$ be the least stage t, such that:

- α is visited at stage t,
- for all $u > t \Rightarrow \delta_u \not<_{left} \alpha$,
- no $\sigma \in Z^+(\alpha)$ enumerates into A after stage t.

Clearly if $\alpha \subset \beta$ are both on the true path, then $True(\alpha) \leq True(\beta)$.

Lemma 3.5. *Let α be a $\mathcal{R}_{e,p}$-node on the true path:*

(i) *If $|X_{e,p}| < \infty$, then f is the true outcome of α.*
(ii) *If $|X_{e,p}| = \infty$, but h_e is not an order, then ∞f is the true outcome of α.*
(iii) *If $|X_{e,p}| = \infty$, and h_e is an order, then $\infty\infty$ will be the true outcome of α.*

Proof. Clearly (i) holds because if $X_{e,p}$ is finite, then eventually no new numbers show up and α will always play outcome f at each visit past a certain stage. For (ii) and (iii), observe that h_e is an order iff $l(\alpha, s) \to \infty$. (ii) is clear enough, so we prove (iii). Suppose that $|X_{e,p}| = \infty$, and h_e is an order, but α never plays the outcome $\infty\infty$ after stage s.

Let $s_0 > s$ be the least stage such that $h_e(l(\alpha, s_0)) > C^\tau_{n,r}$ for every top $\tau \supseteq \alpha^\frown\infty\infty$ and $n, r \in \mathbb{N}$. The number s_0 exists because no top $\tau \supseteq \alpha^\frown\infty\infty$ is ever visited after stage s. Let $s_1 > s_0$ be the least stage such that $l(\alpha, s_1) > l(\alpha, s_0)$. We also let $s_2 > s_1$ be the least such that $X_{e,p}[s_2] - X_{e,p}[s_1] \neq \emptyset$. Finally let $t \geq s_2$ be the least stage such that α is visited at stage t.

If α is visited at any stage u, $s_1 \leq u < s_2$, then α must play outcome f. To see this, suppose not, and let u^- be the previous visit to α prior to stage u. Note that $u^- \not< s_1$, otherwise $l(\alpha, u^-) < l(\alpha, s_1) \leq l(\alpha, u)$. But we also cannot have $s_1 \leq u^- < u$, otherwise $X_{e,p}[u] \supsetneq X_{e,p}[u^-] \supseteq X_{e,p}[s_1]$.

Since α does not play outcome f at any stage u, $s_1 \leq u < s_2$, it follows that $\delta_t \supset \alpha^\frown\infty\infty$, a contradiction. □

The following is the analogue of Lemma 2.3, applied to the current situation. In part (i) we show that the maximum number of different restraints that a current b-box can hold is at most $b + 1$. Part (ii) says that the maximum number of different restraints held by any current b-box for $b < n$ of stronger priority, is at most $S(\tau, n)[s]$.

Lemma 3.6. (i) *Let β be on the true path, with true outcome $\infty\infty$. Let $k \in \mathbb{N}$, and t_0 be a stage $\geq True(\beta)$ such that $size_k^\beta[t_0]\downarrow$. Then,*

$$|A_s\restriction_{j(k,s)}: J^A(k)[s]\downarrow \ \wedge\ \delta_s \supset \beta^\frown\infty\infty \ \wedge\ s \geq t_0| \leq 1 + size_k^\beta[t_0].$$

(ii) *Let τ be a top node on the true path, $n \in \mathbb{N}$, and $t_1 \geq True(\tau)$. Then, the number of pairs $\langle A_s\restriction_{j(k,s)}, k\rangle$ for which*

(a) $J^A(k)[s]\downarrow$,
(b) *For some* $\beta \in Z^-(\tau)$, *we have* $size_k^\beta[t_1] < n$,
(c) $\delta_s \supset \tau$ *and* $s \geq t_1$,
is at most $S(\tau, n)[t_1]$.

Proof. (i): The proof is the same as in Lemma 2.3(i); we include it here for the sake of completeness. Suppose that there are stages $t_0 \leq s_0 < s_1 < \cdots < s_m$ such that $A_{s_i}\upharpoonright_{j(k,s_i)} \neq A_{s_{i+1}}\upharpoonright_{j(k,s_{i+1})}$ for all $i = 0, \ldots, m-1$ (where $m = 1 + size_k^\beta[t_0]$). For each i, the change $A_{s_i}\upharpoonright_{j(k,s_i)} \neq A_{s_{i+1}}\upharpoonright_{j(k,s_{i+1})}$ must have been caused by some $\sigma \supseteq \beta^\frown\infty\infty$ receiving positive attention at some stage t where $s_i \leq t < s_{i+1}$. It must also be the case that we had decreased $size_k^\beta$ at stage t. This means that by the time we reach stage s_{m-1}, we have $size_k^\beta = 0$. Hence at all stages $t \geq s_{m-1}$, no $\sigma \supseteq \beta^\frown\infty\infty$ can make an enumeration below $j(k, s_{m-1})$, a contradiction.
(ii): Using part (i). □

The following is the counterpart to Lemma 2.4. We show that if $C^\tau_{n,r}[s_0]$ has been defined, then no box of a current size at least $C^\tau_{n,r}[s_0]$ can be promoted to a $C^\tau_{n,r-1}[s_0]$-box.

Lemma 3.7. *Let* α *be on the true path with true outcome* f, $\tau = \tau(\alpha)$, *and* $\lim_{t\to\infty} region(\alpha, t) = n$. *Then, the following are true.*

(i) *For each* $0 \leq r \leq n-1$, $\beta \in Z^-(\tau)$, *each stage* s_0 *such that* $C^\tau_{n,r}[s_0]\downarrow$, *and* k *such that* $size_k^\beta[s_0] \geq C^\tau_{n,r}$, *we have*

$$\forall t\,(t \geq s_0 \Rightarrow size_k^\beta[t] > C^\tau_{n,r-1}).$$

(ii) *The total number of times which* α *can be reset after its last initialization, is bounded by* I^τ_n.

Proof. (i): We proceed by induction on r. Suppose the results hold for all $r' < r$.

Suppose on the contrary that there is some $\beta \in Z^-(\tau)$, stage s_0 and number k such that $size_k^\beta[s_0]\downarrow \geq C^\tau_{n,r}$, and some $s > s_0$ such that $size_k^\beta[s] \leq C^\tau_{n,r-1}$. We may as well assume that $size_k^\beta[s_0] = C^\tau_{n,r}$. Since $C^\tau_{n,r}[s_0]\downarrow$, it follows there is some stage $\bar{s_0} \leq s_0$, such that α is visited at stage $\bar{s_0}$ in which $C^\tau_{n,r}$ receives its definition.

Suppose $t \geq s_0$ is a stage where some σ enumerates into A, resulting in a decrease in $size_k^\beta$. It is clear that:

(i) $\sigma <_{left} \alpha$,
(ii) $\sigma \supseteq \alpha^\frown\infty\infty$ or $\sigma \supseteq \alpha^\frown\infty f$,
(iii) $\sigma \in Z^+(\alpha)$,

cannot hold, lest α gets initialized after stage s_0. It cannot also be the case that:

(iv) $\sigma^\frown\infty\infty \subseteq \alpha$ or $\sigma^\frown\infty f \subseteq \alpha$,
(v) $\sigma >_{left} \alpha$,
(vi) $\sigma \supseteq \alpha^\frown f$,

because otherwise $region(\sigma, t)$ has to be larger than $C^\tau_{n,r}$ when σ enumerates at stage t. This is because α is visited at stage $\bar{s_0} \leq s_0$, and before $C^\tau_{n,1}, \ldots, C^\tau_{n,r}$ receives their definition, we never get to visit any node $\delta \supseteq \alpha^\frown f$. This means that $size_k^\beta[t] \leq C^\tau_{n,r} < region(\sigma, t) \leq threshold(\sigma, t)$, a contradiction.

This leaves only the case when $\sigma = \alpha$; that is, the only node that can bring down $size_k^\beta$ after stage s_0, is α itself. We want to count the number of possible such stages t. At stage t, we must have $C^\tau_{n,r} \geq size_k^\beta[t] > threshold(\alpha, t) = C^\tau_{n,n-2-attempt(\alpha,t)}$, which puts $attempt(\alpha, t) > n-2-r$. As before, we split the counting into the cases:

– *Case* x: t *is a stage where* α *makes an enumeration which results in a decrease in* $size_k^\beta$, *and* $attempt(\alpha, t) = x$, *where* $n-1-r \leq x \leq n-1$.

If $x = n-1$, then $attempt(\alpha)$ will be increased to n and α never enumerates again. So, suppose that $x < n-1$ (and hence $r > 0$). In order for Case x to apply again, α has to be reset at some (least) stage $t' > t$, where $attempt(\alpha, t') \geq x+1 > n-1-r$. This means that for some β' and k' with $\beta' \in Z^-(\tau)$, we have $J^A(k')[t']\downarrow$ and $size_{k'}^{\beta'}[t'] \leq threshold(\alpha, t') \leq C^\tau_{n,r-2}$.

We firstly claim that $size_{k'}^{\beta'}[\bar{s_0}]\downarrow$. Suppose not. Since $r > 0$, it follows that $h_{order(\beta')}(l(\beta', \bar{s_0})) > C^{\tau}_{n,r-1}$, and that $h_{order(\beta')}(k') > C^{\tau}_{n,r-1}$. Applying induction hypothesis (on $r-1$) gives us a contradiction.

We can further conclude that $size_{k'}^{\beta'}[\bar{s_0}] < C^{\tau}_{n,r-1}$ (by applying induction hypothesis on $r-1$), and by Lemma 3.6(ii), there can be at most $1 + S(\tau, C^{\tau}_{n,r-1})[\bar{s_0}]$ many stages t where Case x applies (by associating each t with the string $A_{t'}\restriction_{j(k',t')}$).

Considering all the different cases, we see that the smallest value $size_k^{\beta}$ can take, is $C^{\tau}_{n,r} - 1 > C^{\tau}_{n,r-1}$, if $r = 0$. On the other hand if $r > 0$, the smallest value $size_k^{\beta}$ can take, is $C^{\tau}_{n,r} - 1 - n(1 + S(\tau, C^{\tau}_{n,r-1})[\bar{s_0}]) = C^{\tau}_{n,r-1} + 1 > C^{\tau}_{n,r-1}$.

(ii): Let t_0 be the stage where I^{τ}_n receives its definition. If α is reset at stage $t > t_0$, then for some β and k with $\beta \in Z^-(\tau)$, we have $J^A(k)[t]\downarrow$ and $size_k^{\beta}[t] \leq threshold(\alpha, t) \leq C^{\tau}_{n,n-2}$. By part (i), it follows that $size_k^{\beta}[t_0]\downarrow$, and furthermore that $size_k^{\beta}[t_0] < C^{\tau}_{n,n-1}$. Again by Lemma 3.6(ii), there can be at most $S(\tau, C^{\tau}_{n,n-1})[t_0] = I^{\tau}_n$ many such stages t (by associating each t with the string $A_t\restriction_{j(k,t)}$). □

Lemma 3.8. *$y \in S \Leftrightarrow A$ is strongly jump traceable.*

Proof. ($\Rightarrow$): Suppose $y \in S$, and we let $h = h_e$ be an order function. There is some p such that $|X_{e,p}| = \infty$, so let α be the node on the true path assigned the requirement $\mathcal{R}_{e,p}$. By Lemma 3.5, α has true outcome $\infty\infty$. Fix a $k \in \mathbb{N}$. If $J^A(k)\downarrow$, then clearly it will be enumerated into V^{α}_k after stage $True(\alpha)$. Each distinct value in V^{α}_k corresponds to a string $A_s\restriction_{j(k,s)}$, where $J^A(k)[s]\downarrow$, $\delta_s \supset \alpha^\frown\infty\infty$ and $s \geq True(\alpha)$ and $l(\alpha, s) > k$, so by Lemma 3.6(i) it follows that $|V^{\alpha}_k| \leq h_e(k)$.

($\Leftarrow$): Suppose now $y \notin S$. By Lemma 3.5, let τ be the maximal top node on the true path. $\tilde{h}_\tau$ is total because all of τ's children α are on the true path, and each α will extend $dom(\tilde{h}_\tau)$ at least once. Furthermore, each time α defines a piece of $\tilde{h}_\tau$, it picks a value larger than anything used before, so that $\tilde{h}_\tau$ is an order.

Let $\alpha \supseteq \tau$ be on the true path, and let $n = \lim_{t\to\infty} region(\alpha, t)$. By Lemma 3.7(ii), α never runs out of indices after its last initialization. We let $x = \lim_{t\to\infty} x(\alpha, t)$, and clearly we have $\tilde{h}_\tau(x) = n$. Suppose that $J^A(x)\downarrow \in T_x^{trace(\alpha)}$. Let $s_0 > True(\alpha)$ be large enough so that $J^A(x) \in T_x^{trace(\alpha)}[s_0]$, and $x(\alpha, s_0) = x$. It must be the case that $\Psi^A_\alpha(x)[s_0]\downarrow = J^A(x)$ (because any axiom $\langle x, y, \sigma\rangle$ we enumerate into Ψ^A_α after stage s_0, must have $y \geq s_0 > J^A(x)$). Hence, it must be the case that $attempt(\alpha, s_0) = n$, otherwise (ATT3) would hold and we would destroy the correct axiom in Ψ^A_α. This means that there are n many different stages t (before stage s_0) where α receives positive attention, and $attempt(\alpha, t)$ is increased by 1. After each such stage t we also enumerate a new value for $\Psi^A_\alpha(x)$, and wait for it to be traced. By stage s_0, we have $|T_x^{trace(\alpha)}[s_0]| \geq n$. □

The proof of Theorem 3.3 is complete, upon observation that the construction of A is uniform in y, by using a uniform version of the recursion theorem.

References

[1] K. Ambos-Spies, An extension of the nondiamond thoerem in classical and α-recursion theory, Journal of Symbolic Logic 49 (1984) 586–607.

[2] P. Cholak, R. Downey, N. Greenberg, Strong jump-traceability 1 : The computably enumerable case, Advances in Mathematics (in press).

[3] R. Downey, The sixth lecture on algorithmic randomness, in: Proceedings of the IMS Workshop on Computational Aspects of Infinity, Singapore, 2007 (in press).

[4] R. Downey, C. Jockusch, Every low boolean algebra is isomorphic to a recursive one, Proceedings of the American Mathematical Society 122 (3) (1994) 871–880.

[5] S. Figueira, A. Nies, F. Stephan, Lowness properties and approximations of the jump, in: Proceedings of the Twelfth Workshop of Logic, Language, Information and Computation, WoLLIC 2005, in: Electronic Lecture Notes in Theoretical Computer Science, vol. 143, 2006, pp. 45–57.

[6] S. Ishmukhametov, Weak recursive degrees and a problem of spector, Recursion theory and complexity (Kazan, 1997) 2 (1997) 81–87.

[7] A. Lachlan, Embedding nondistributive lattices in the recursively enumerable degress, in: Conference in Mathematical Logic, London, in: Lecture Notes in Mathematics, vol. 255, 1970, pp. 149–177.

[8] A. Lachlan, A recursively enumerable degree which will not split over all lesser ones, Annals of Mathematical Logic 9 (1975) 307–365.

[9] K.M. Ng, Beyond strong jump traceability (in preparation).

[10] A. Nies, On a uniformity in degree structures, in: Complexity, Logic and Recursion Theory, in: Lecture Notes in Pure and Applied Mathematics, Feb. 1997, 1997, pp. 261–276.

[11] A. Nies, Reals which compute little, CDMTCS Research Report 202, The University of Auckland, 2002.

[12] A. Nies, Lowness properties and randomness, Advances in Mathematics 197 (2005) 274–305.
[13] R. Robinson, Interpolation and embedding in the recursively enumerable degrees, Annals of Mathematics 93 (2) (1971) 285–314.
[14] G. Sacks, On the degrees less than $\mathbf{0}'$, Annals of Mathematics 77 (2) (1963) 211–231.
[15] R. Shore, T. Slaman, Working below a low_2 recursively enumerable degree, Archive of Mathematical Logic 29 (1990) 201–211.
[16] R. Shore, T. Slaman, Working below a high recursively enumerable degree, Journal of Symbolic Logic 58 (1993) 824–859.
[17] T. Slaman, R. Solovay, When oracles do not help, in: Fourth Annual Conference on Computational Learning Theory, 1971, pp. 379–383.
[18] R. Soare, Computational complexity, speedable and levelable sets, Journal of Symbolic Logic 42 (1977) 545–563.
[19] R. Soare, Automorphisms of the lattice of recursively enumerable sets, part 2 : Low sets, Annals of Mathematical Logic 22 (1982) 69–107.
[20] R. Soare, Tree arguments in recursion theory and the $\emptyset'''$-priority method, Proceedings of Symposia in Pure Mathematics 42 (1985) 53–106.
[21] R. Soare, Recursively enumerable sets and degrees, in: Perspectives in Mathematical Logic, Springer-Verlag, 1987.
[22] R. Solovay, Draft of paper (or series of papers) on chaitin's work, unpublished notes, 215 pages, 1975.
[23] S. Terwijn, D. Zambella, Algorithmic randomness and lowness, Journal of Symbolic Logic 66 (2001) 1199–1205.