BEYOND STRONG JUMP TRACEABILITY

KENG MENG NG

Abstract. Strong jump traceability has been studied by various authors. In this paper we study a variant of strong jump traceability by looking at a partial relativization of strong jump traceability. We discover a new subclass $\mathcal{H}$ of the c.e. K-trivials with some interesting properties. These sets are computationally very weak, but yet contain a cuppable member. Surprisingly they cannot be constructed directly using cost functions, and is the first known example of a subclass of the K-trivials which does not contain any promptly simple member. Furthermore there is a single c.e. set which caps every member of $\mathcal{H}$, demonstrating that they are in fact very far away from being promptly simple.

1. Introduction

The last few years have seen an extraordinarily fruitful interaction between the areas of algorithmic information theory and computability theory. A highlight in the programme of trying to understand algorithmic information has been the work on various notions of lowness. A notion of lowness is one indicating weakness as an oracle. Intuitively, a notion of lowness means that a set does not give any extra power to certain operations when it is used as an oracle; it's role as an oracle can be dispensed with. The classical example is where operation is the Turing jump, where a set A is low if $A' \equiv_T \emptyset'$. That is, a low set A is indistinguishable from the empty set as far as the jump operator is concerned. We would of course expect that low sets resemble computable sets, and there have been many results in literature demonstrating this, especially amongst the computably enumerable sets. We refer to Soare [19], Chapter XI for many illustrations of the use of lowness for the Turing jump.

The notion of lowness can be considered for many other concepts where relativization makes sense. A beautiful example is the paper of Slaman and Solovay [17] who studied the notion of lowness for computational learning theory. They showed that for a certain notion of learning, called EX learning, *low for EX learning coincides with being low and 1-generic.* It is not important for this paper what EX learning is, but the point here is that we have a remarkable coincidence between a lowness notion for learning theory and notions from classical computability.

The present paper was motivated by the beautiful interactions of lowness and simplicity in Kolmogorov complexity with concepts from classical computability. The work of Terwijn and Zambella [21], Kjös-Hanssen, Nies and Stephan [10], Bedregal and Nies [2] showed that the notion of being low for Schnorr randomness[1] coincided with a natural combinatorial notion: being computably traceable. Here A being computably traceable means that A is "uniformly hyperimmune-free" in that there is a computable function h such that for each $f \leq_T A$, there exists a computable sequence of canonical finite sets $D_{g(x)}$ with $|D_{g(x)}| = h(x)$, and such that $f(x) \in D_{g(x)}$ for all x.

Date: February 3, 2010
Mathematics Subject Classifications: 03D25, 03D32.

[1]Recall that a Schnorr test is a Martin-Löf test $\{U_i : i \in \omega\}$ with $\mu(U_i) = 2^{-i}$.

The most well-known work on lowness relative to Kolmogorov complexity is the work of Nies [13, 15] showing the coincidence of the K-trivial reals, the low for K reals[2], and the reals low for Martin-Löf randomness[3].

The class of K-trivial reals was first introduced in [20]. They are the reals α such that for some constant c, $K(\alpha\upharpoonright_n) \leq K(n) + c$ for every n[4]. That is, every initial segment of a K-trivial real contains no more information than just its own length. Solovay [20] was the first to show that there are non-computable reals which are K-trivial. This construction has been simplified and is now known as a *cost function construction*. The K-trivials have aroused great interest in recent years, and are related to various other classes defined independently. Subsequently, more characterizations of the K-trivial reals were found, such as the bases of Martin-Löf randomness [8], and the reals low for weak 2-randomness.

These characterizations show that the notion of ML-lowness is a very robust one. It is very natural to ask if the class relates to notions from classical computability in any meaningful way. For example, is there a computability-theoretical characterization like the one for EX-lowness? Is there a combinatorial characterization in terms of classical computability like the one for Schnorr lowness?

In [6], Downey, Hirschfeldt, Nies and Stephan showed that the K-trivial reals are natural solutions to Post's problem in the sense that they have incomplete Turing degrees. This is by the Decanter method, whose fanciful name is derived from the fact that we amplify small mistakes made by the opponent. For a good description of this method, we refer the reader to [5]. Again, this method is based on the fact that K-trivial sets resemble $\emptyset$; we can ask for certification that a certain initial segment does not change. Nies [14, 15] then showed that every c.e. K-trivial was superlow (A is superlow, if $A' \equiv_{tt} \emptyset'$). Thus K-triviality is essentially an enumerable phenomenon. This connects K-triviality (as a notion of lowness in terms of randomness) with the traditional notion of lowness (in terms of Turing jump operators). These results go some way towards understanding ML-lowness in classical terms.

Efforts towards a combinatorial classification came from the results above. Nies' proof that K-trivial sets are superlow actually showed that their *jump* has a tracing property similar to computable traceability. An order function h is one which is total computable, non-decreasing 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 $J^A(x) \in W_{g(x)}$. Here, $J^A(x)$ denotes the value of the universal function $\{x\}^A(x)$ partial computable in A. A is 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 [21]), and c.e. traceability (Ishmukhametov [9]). The class of jump traceable sets was introduced by Nies [14] initially to study lowness properties relating to K. 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 [14, 15], Nies showed that every K-trivial real was jump traceable with an order function of growth rate $\sim h(n) = n \log n$.

Nies' results suggested that perhaps K-triviality is related to the growth rate of orders for jump tracing. This insight lead to Figueira, Nies and Stephan [7] studying the notion of *strong jump traceability*. We say that A is *strongly jump traceable*, if it is jump traceable with respect to *all* (computable) order functions. Figueira, Nies and Stephan used a cost function construction to construct a non-computable strongly jump traceable c.e. set. Figueira, Nies and Stephan characterized the

[2] A is low for K if $\exists c \forall \sigma (K(\sigma) \leq K^A(\sigma) + c)$.

[3] A is ML-low, if every A-random set is already 1-random.

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

c.e. strongly jump traceable sets via the notion of well approximability: a set X is *well approximable*, if for every order function h, X 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. Hence, one can view c.e. strong jump traceability as a natural strengthening of lowness or even of superlowness. Perhaps this was the hoped for characterization of K-triviality in classical and also combinatorial terms, a question explicitly asked in Miller and Nies [11].

In recent work, Cholak, Downey and Greenberg [4] showed that the c.e. strongly jump traceable sets form a *proper* subclass of the K-trivials. In fact, they proved that if A is c.e. and jump traceable at order $\sim h(n) = \sqrt{\log n}$, then A is K-trivial. This is the first example of a combinatorial property which implies K-triviality. They showed that like the K-trivials, the c.e. strongly jump traceable sets 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 $\sim h(n) = \log \log n$. Thus, for the first time we have a combinatorial notion, generated by a cost function construction, giving a proper subclass of the ideal of K-trivials.

Many questions suggest themselves. How does this idea relate to other notions from Kolmogorov complexity such as ML-cuppability and the like? Is this the limit of the cost function construction? Are there any natural proper subclasses of this ideal? Two possible characterizations have been suggested here: Greenberg suggested that A is K-trivial iff A is jump traceable for all orders h with $\sum_{n\in\omega} \frac{1}{h(n)} < \infty$. Cholak, Downey and Greenberg suggested that this should be $\sum_{n\in\omega} 2^{-h(n)} < \infty$. Is there some combinatorial characterization of the class of K-trivials in terms of orders between the Nies' bound and the Cholak-Downey-Greenberg bound? What else can be said about the class of strongly jump traceable reals, as they seem a very interesting class in their own right.

This paper and an earlier one [12] are devoted to the constellation of questions above. In [12], Ng showed that the index set $\{e \in \mathbb{N} : W_e \text{ is strongly jump traceable}\}$ is Π^0_4-complete. It was also shown that there is no minimal bound for jump traceability, that is, there is no single order function such that strong jump traceability is the same as jump traceability for that order. This is different in the case of c.e. traceability or computable traceability. Nies observed that the K-trivial reals form a natural nontrivial Σ^0_3 ideal in the c.e. Turing degrees. Therefore in terms of the complexity of the classes, the strongly jump traceable sets are as complex as they could be, and in fact differ from the K-trivials as much as they possibly can.

In Section 2 we show that not every c.e. set jump traceable at identity is K-trivial, disproving one of the conjectures towards a combinatorial characterization of ML-lowness. We do this by constructing two c.e. sets which are both jump traceable at identity, but whose join has complete Turing degree. This result shows that any hope of giving a combinatorial characterization of K-triviality in terms of jump traceability, must include at least some functions which grow slower than the identity.

Another goal of the present paper is to introduce a proper subclass of the strongly jump traceable reals. This class has a number of very interesting properties and is the first class constructed by a cost function type construction which cannot, for instance, be promptly simple, nor can it be carried out below any c.e. degree. We believe its study may have significant implications in classical computability.

We will study a variant of strong jump traceability by relativizing (partially) the concept of traceability. In Section 3, we define what it means for a c.e. set A to be strongly jump traceable by another set X, denoted by the relation "$A \in SJT(X)$". We study this binary relation, keeping in mind the interactions with computational

lowness. The main result in Section 3 is Theorem 3.5, where we construct two c.e. sets A and B which are both strongly jump traceable, but $A \notin SJT(B)$. This is somewhat contrary to intuition, since if B is computationally very weak, we would expect that $SJT(B)$ is exactly the class of strongly jump traceable c.e. sets.

Theorem 3.5 provides us with the encouragement that we need to take this study further. We define a new class of c.e. sets $\mathcal{H} := \bigcap\{SJT(W) \mid W \text{ is c.e.}\}$, and call these sets *hyper jump traceable*. These sets are all strongly jump traceable, with an even more marked resemblance to the computable sets. In Section 4 we construct a non-computable hyper jump traceable c.e. set, by using a $\emptyset'''$-priority argument. We also show that such sets can be cuppable. Thus, there is a fundamental property which separates them from the computable sets. In Section 5 we show that no hyper jump traceable c.e. set can be low cuppable, and this implies that the hyper jump traceable c.e. sets cannot be constructed using a single cost function. This forms the first known example of a subclass of the K-trivials free of promptly simple sets. Moreover, they form the first class of sets which are constructed without any cupping notions in mind, but which *can* be cuppable, yet *cannot* be promptly simple.

In Section 6 we show that no c.e. set A is strongly jump traceable relative to *all* Δ^0_2 sets, apart from the computable sets. Therefore in some sense, such a variation of strong jump traceability is the best possible amongst the c.e. sets. In Section 7 we show that not only is every hyper jump traceable c.e. set cappable, but there is also a single capping companion for the entire class $\mathcal{H}$. Hence, $\mathcal{H}$ is far away from being promptly simple.

Our notation is fairly standard, and follow Soare [19]. Unless otherwise stated, $\{T^e_x\}_{x\in\mathbb{N}}$ is the e^{th} c.e. trace in some effective enumeration of all c.e. traces. For an oracle $X \subseteq \mathbb{N}$ and $n \in \mathbb{N}$ where $J^X(n)\downarrow$, we write $j^X(n)$ to denote the use. We append $[s]$ to long expressions to refer to the value of that expression as evaluated at stage s.

2. Not every c.e. set jump traceable at identity is K-trivial

It is known that the K-trivial reals form an extremely robust class, having different characterizations. Existing results tell us that in the c.e. case, we have A is jump traceable at order $\sqrt{\log n} \Rightarrow A$ is K-trivial $\Rightarrow A$ is jump traceable at order $n \log^2 n$. This suggests that the K-trivials might have a combinatorial characterization using order functions having growth rates somewhere between the two extremes (e.g. the identity function). Greenberg, as well as Cholak, Downey and Greenberg put forward the conjectures:

Conjecture 2.1. *A is K-trivial if and only if A is jump traceable at all order functions h such that $\sum_{n=1}^{\infty} h(n)^{-1} < \infty$.*

Conjecture 2.2. *A is K-trivial if and only if A is jump traceable at all order functions h such that $\sum_{n=1}^{\infty} 2^{-h(n)} < \infty$.*

We show that not every c.e. set jump traceable at identity is K-trivial, hence giving a negative answer to Conjecture 2.1. We do this by constructing a pair of c.e. sets which are both jump traceable at identity, and whose join has complete Turing degree. Since the K-trivial sets are closed under $\oplus$, at least one of the two constructed sets is not K-trivial. This result says that any class of order functions characterizing the K-trivials must contain a member which does not dominate the identity function, and hence Conjecture 2.2 is still open.

Bickford and Mills introduced the concept of superlowness, and in [3] constructed a pair of c.e. superlow sets which joins to $\emptyset'$. A calculation of the underlying order

function yields a growth rate of around $\sim n^2$; in the following theorem we will improve this to the identity function.

Theorem 2.3. *There are c.e. sets A_0 and A_1 which are both jump traceable at identity, such that $\emptyset' \leq_T A_0 \oplus A_1$.*

Proof. We construct traces $\{T^i_x\}_{x>0}$ for J^{A_i}, $i = 0, 1$, as well as a Turing functional Γ ensuring that $\emptyset' = \Gamma^{A_0 \oplus A_1}$. The stage s use of this computation is recorded by $\gamma(x, s)$, which we think of as a moving marker pointing at a particular number not yet in $A_0 \cup A_1[s]$. To move the marker $\gamma(x)$, we would have to remove the Γ-axiom we had previously set; to do this we have to put[5] $\gamma(x)$ into either A_0 or A_1. If $\gamma(x)$ is not enumerated then it is not allowed to move. We fix an enumeration $\emptyset'[s]$ of the halting problem, such that at most one number is enumerated per stage. When we pick a *fresh* number at stage s, we mean a number $> s$ and $>$ any number used so far. The construction involves a careful monitoring of box sizes. We think of each T^i_x as being made up of locations, or "boxes", which we will fill with current values of $J^{A_i}(x)[s]$. As more elements enter T^i_x, the "box size" (i.e. the number of available free slots) goes down. This box intuition is convenient, and references of this sort will be made throughout this paper. This concept also appears in [4], where the *box promotion method* was used to prove several results.

In order to avoid ambiguity, we assume that if $J^\rho(x)[s] \downarrow$ for any string ρ and x, s, then this fact can be seen at the beginning of stage s. Also, we want to distinguish between the two computations $J^\rho(x)$ and $J^{\rho'}(x)$ which may have the same value but have different use $\rho \neq \rho'$. Therefore, instead of enumerating potential values of $J^{A_i}(x)$ into T^i_x, we will enumerate the use of these potential computations (as finite strings) into T^i_x. As long as we ensure that $J^\rho(x) \downarrow$ for some $\rho \subset A_i \Rightarrow \rho \in T^i_x$, then A_i would be jump traceable via some suitable transformation of $\{T^i_x\}$. This feature is not necessary, but useful as it allows us to avoid considering different cases. We say that a computation $J^\rho(x) \downarrow$ *has been i-traced* if ρ has been enumerated in T^i_x.

2.1. **Description of strategy.** We think of each T^i_x as a collection of x many "slots", which we will each fill with potential jump computations. Each of these slots is also called a "box". Initially T^i_x starts off as having box size of $x - 1$; we also say it is a $(x - 1)$-box. $x - 1$ represents the fact that T^i_x has not yet been injured, and hence can take $x - 1$ many more injuries. Each time a potential jump computation is enumerated in T^i_x and subsequently destroyed, we decrease the box size by one. To ensure jump traceability, all we need to do is to ensure that at all times, no box of size 0 is injured on either side.

To make this compatible with coding requirements, we pick a number $forbid(e)$ for each marker $\gamma(e)$, and then ensure that every time we see $\gamma(e) < j^{A_i}(w)$ for some w of small T^i_w-box size $\leq forbid(e)$ on the A_i-side, we move $\gamma(e)$ out of the way. We do this by enumerating $\gamma(e)$ on the other side, into A_{1-i}, and then pick a fresh marker location for $\gamma(e) > j^{A_i}(w)$. If small box sizes appear on both sides A_0 and A_1, we pick any of the two sides to injure; this does not matter as both of these computations must have newly converged, and we would not have traced either of these computations yet (we arrange for computations to be traced at the end of each stage, so that only the surviving computations are traced). If we ensure that at all times, $0 < forbid(1) < forbid(2) < \cdots$, then no traced 0-boxes can ever be injured. This ensures jump traceability of both A_0 and A_1.

The above plan is general enough to work for all orders h, and yet we know that if h is sufficiently slow-growing, we cannot code $\emptyset'$ into the join of two sets jump traceable at order h. We want to show that the identity $h(n) = n$ grows quickly

[5]Hence the use of $\Gamma^{A_0 \oplus A_1}(x)$ is really $2\gamma(x)$. $\gamma(x)$ marks the part of A_0 and A_1 which is accessed by the computation.

enough to allow coding. The problem we have clearly not yet considered, is the fact that $\gamma(e)$ might not settle. Let us consider $\gamma(1)$, and suppose that $forbid(1) = F$ (which never changes since $\gamma(1)$ is of the highest priority). Now $\gamma(1)$ might be put into A_0 because some box of size $\leq F$ has been filled with a convergent computation with use above $\gamma(1)$ on the A_1-side. Our strategy ensured that there was no box *which is already traced*, and of size $\leq F$ converging above $\gamma(1)$ on the A_0-side. However, there might be boxes of size $F+1$ with convergent computations above $\gamma(1)$ on the A_0-side, which have already been traced. Thus, in trying to avoid boxes of small size on the A_1-side, we might create more boxes of size F on the A_0 side. These new small-sized boxes we created on the A_0 side might come back and force us to create more boxes of size F on the A_1-side, and so on. Hence $\gamma(1)$ never settles since it spends all of its time avoiding these small boxes.

This obstacle cannot be overcome if h is sufficiently slow growing. The intuition is that if h grows very slowly, then every time we avoid a single box on one side, we will create a lot of new F-boxes on the other side. However, if h is the identity, then the creation of new F-boxes is controlled; avoiding a single box on one side creates (more or less) just a single extra F-box on the other side. We make this more precise: we want to argue that at identity $h(n) = n$, the marker $\gamma(1)$ is only moved finitely often. $\gamma(1)$ moves only to avoid an F-box T^i_w, and once $\gamma(1)$ moves to its new location the same box T^i_w is never a problem again on the A_i side, since the computation will be preserved forever. We wait until we have done this $F+1$ times on each side, at some stage s. We can ensure that at each stage t we only make traces into T^i_x for $x < t$, hence at stage s, T^i_s is empty and hence must be an $(s-1)$-box for both i (this is where we make use of the fact that h is identity). In future, how many times may T^0_s be injured? As described above, each time T^0_s is injured, we brush aside some T^1_w, of current size $\leq F$. We may assume that $w < s$, otherwise T^1_s will become an F-box before T^0_s does, in which case we argue with T^1_s in place of T^0_s. Each of these T^1_w only blocks $\gamma(1)$ once, and there are altogether only $s-1-(F+1)$ many of these boxes to consider, since at stage s we had already had $F+1$ many of these boxes converging below $\gamma(1)$. Therefore, T^0_s can be reduced to at most a $(F+1)$-box, hence neither of T^0_s nor T^1_s can become an F-box. Hence this process stops, and $\gamma(1)$ eventually settles. An inductive argument can be applied for $\gamma(e)$.

Note this observation works just as well if $h(n) = n - c$ for any constant c, but no longer works if h is unbounded away from the identity, e.g. $h(n) = \frac{n}{2}$. Any attempt to improve Theorem 2.3 will have to address the issue of *amplification*. That is, the opponent might pursue some sort of strategy along the lines of the Decanter or box promotion methods, and amplify small errors repeatedly. In this construction, we stop the opponent from doing this by isolating his actions on each of our requirements. We do so by setting up barriers; if the opponent has caused us to make an error on $\gamma(x)$ then we isolate his actions on $\gamma(x+1), \gamma(x+2), \cdots$, so that errors we make on these cannot be amplified to cause us grief on $\gamma(x)$. The identity function grows just fast enough to permit us to do that. The formal construction and verification fills in the rest of the details.

2.2. **Notations.** We record the size of T^i_k by the parameter $size_i(k, s)$, where $k > 0$. This is initially set to $size_i(k, 0) = k - 1$ for all $k \in \mathbb{N} - \{0\}$ and $i = 0, 1$. At times we decrease $size_i(k)$ when T^i_k is injured, and when $size_i(k)$ reaches 0 then the use of $J^{A_i}(k)$ must be preserved when the computation next converges. We ensure $size_i(0, s) \leq size_i(1, s) \leq size_i(2, s) \leq \cdots$ for each i, s. Thus when a T^i_x box is injured we will have to decrease $size_i(k)$ for all k such that $size_i(k) = size_i(x)$; we pretend that all these boxes are injured as well. Formally, during the construction, before we enumerate a number x into A_i at stage s, we will *update the box size* by

doing the following: let $X = \{k < s : J^{A_i}(k)[s] \downarrow$ with use $> x$, and has already been i-traced$\}$. For every k' such that $size_i(k') = size_i(k)$ for some $k \in X$, we decrease $size_i(k')$ by 1. That is, if we decrease $size_i(k)$ for $k \in X$, we will also decrease $size_i(k')$ for all k' in the "same block" as k. We have a variable $forbid(x, s)$ which represents the region forbidden to $\gamma(x)$ - i.e. the largest number w such that $\gamma(x)$ is not allowed to be below the use of $J^{A_i}(y)$ for $i = 0, 1$ and any T^i_y with box size $\leq w$.

2.3. **The construction.** At stage 0 initialize the parameters according to the following. Set $size_i(k) = k - 1$ for all $k > 0$ and $i = 0, 1$. Set $forbid(x) = x$ for all x, and do nothing else. Suppose now that $s > 0$. There are three parts to the construction:

(1) First, check if any corrections to Γ is necessary: if there is some $x \in \emptyset'[s] - \emptyset'[s-1]$, we update the box size and enumerate $\gamma(x)$ (if defined) into A_0. If no corrections to Γ is necessary, we extend the definition of Γ by setting $\Gamma^{A_0 \oplus A_1}(x)[s] \downarrow = \emptyset'(x)[s]$ for the least $x < s$ where no axioms currently apply; do this with a fresh $\gamma(x)$-use.
(2) Pick the least $x < s$ such that $\gamma(x, s) \downarrow$ and $\gamma(x, s) < j^{A_i}(w, s)$ for some i and w such that $size_i(w) \leq forbid(x, s)$, and coding is not yet done (i.e. $x \notin \emptyset'[s]$). Note that the $J^{A_i}(w)[s]$ computation currently blocking $\gamma(x)$ might not have been i-traced, but we will need to take action for it anyway. Update the relevant box size, and enumerate $\gamma(x, s)$ into A_{1-i} (choose the smaller i, if we have a choice) where the above holds. Set $forbid(y) = s + y - x$ for all $y > x$.
(3) For each $i = 0, 1$, and each $k < s$ such that $J^{A_i}(k)[s] \downarrow$, such that A_i had not changed below the use due to (1) and (2) above, we enumerate the use into T^i_k. Note that at stage s we only trace jump values $J^{A_i}(k)$ for $k < s$.

2.4. **Verification.** It is clear that for each $i = 0, 1$, and $x > 0$, if $J^{A_i}(x) \downarrow$, then $A_i \upharpoonright_{j^{A_i}(x)} \in T^i_x$. Suppose that for some i, x, we have $|T^i_x| > x$. There are strings $\rho_0, \rho_1, \cdots, \rho_x$ which have been enumerated into T^i_x in that order. By the use principle, for each $j < x$, there must be a change in $A_i \upharpoonright_{|\rho_j|}$ between the enumerations of ρ_j and ρ_{j+1}. The first such change would cause $size_i(x)$ to be decreased by 1, and therefore when ρ_{x-1} is enumerated, we must already have $size_i(x) \leq 0$. Suppose ρ_{x-1} is traced under step 3 of the construction, at some stage t. Hence $A_i \upharpoonright_{|\rho_{x-1}|}$ was unchanged throughout stage t. If $\gamma(0)$ was defined and below $|\rho_{x-1}|$, then we would have enumerated $\gamma(0)$ under step (2); in any case at the end of stage t, $\gamma(0)$ must be undefined, so there can be no markers below $A_i \upharpoonright_{|\rho_{x-1}|}$ (which have not yet been coded). This is a contradiction as there cannot be any change below $A_i \upharpoonright_{|\rho_{x-1}|}$ after stage t. Hence, A_0 and A_1 are both jump traceable with respect to the identity function.

Next, we have to establish the crucial fact: for each e, $\gamma(e)$ eventually settles. Suppose the result holds for all $e' < e$. Hence $F = \lim_s forbid(x, s)$ exists. Let s_0 be a stage where $forbid(x)$ has settled. For each $s \geq s_0$, we let $\Theta_i[s] = \{x : size_i(x, s) \leq F\}$ (this is an initial segment of $\mathbb{N}$, since we keep $size$ non-decreasing), and let $\bar{\Theta}_i[s] = \{x < s : size_i(x, s) > F\}$ be the rest. We further split $\Theta_i[s]$ into two parts: $\Theta^-_i[s] = \{x \in \Theta_i[s] : J^{A_i}(x)[s] \downarrow$ and has been i-traced$\}$, and the rest goes in $\Theta^+_i[s] = \Theta_i[s] - \Theta^-_i[s]$. Basically, Θ^-_i contains all those boxes with high e-priority, which no longer need to be considered by $\gamma(e)$, while Θ^+_i contains all those high priority boxes which might act and block $\gamma(e)$ at any time in the future.

Suppose for a contradiction, that $\gamma(e)$ moves infinitely often after s_0. Hence $e \notin \emptyset'$. Define the function $gap_i(s) = |\Theta^-_i[s]|$, where the expression is evaluated at the end of stage s. A contradiction is derived using the following lemmas. The

first lemma says that each time a marker $\gamma(e)$ is moved, we never promote a box forbidden to it:

Lemma 2.4. *Let s be a stage, such that $\gamma(r)$ was enumerated in A_i at stage s. If $size_i(k)$ was decreased from $a+1 \to a$ in the corresponding update of box size, then necessarily $a \geq forbid(r)$.*

Proof. Suppose the lemma holds for all stages $s' < s$. We prove the case for stage s. There must be some k' such that $J^{A_i}(k')[s] \downarrow$ with use $> \gamma(r)$, which has already been i-traced. That use must have been i-traced in some stage $t < s$, which means that A_i had not changed below that use in the entire stage t. We must have $\gamma(r)[s] = \gamma(r)[t]$, which means that $size_i(k')[t] > forbid(r,t) = forbid(r,s)$, lest $\gamma(r)$ be moved at stage t. If $size_i(k')[t] = a+1$ then we are done, so assume that $size_i(k')$ is decreased from $a+2$ to $a+1$ between stages t and s, and this has got to be due to some $\gamma(r')$ being moved, for some $r' > r$. By induction hypothesis, we have $a+1 \geq forbid(r') > forbid(r,s)$. □

Lemma 2.5. *There is $s_1 > s_0$ such that $gap_i(s_1) > F$ for both $i = 0, 1$.*

Proof. Note that for all $s > s_0$, we have $gap_i(s+1) \geq gap_i(s)$ for both i (by Lemma 2.4). Furthermore each time the marker $\gamma(e)$ was put in A_{1-i}, it must be under step 2, so each enumeration corresponds to an increase in gap_i. If $gap_1(\cdot)$ is bounded, then only finitely many enumerations are made into A_0. Hence no box of current size $\leq F+1$ can ever be promoted once enumeration of $\gamma(e)$ into A_0 stops, by Lemma 2.4, since only markers $\gamma(e+1)$ and above can be enumerated on the A_0-side. This means that $\Theta_0[s]$ reaches a limit, say θ. Since obviously $gap_0(s) \leq \theta$, it follows that $gap_0(\cdot)$ is bounded as well, contradicting the fact that $\gamma(e)$ is moved infinitely often. Hence, neither gap_0 nor gap_1 can be bounded. □

Lemma 2.6. *For all $t \geq s_1$ and both i, we have $size_i(s_1, t) > F$.*

Proof. Note that $size_i(s_1, s_1) = s_1 - 1$. Suppose on the contrary, that at some stage $s_2 > s_1$ and some i, we update box size causing $size_i(s_1)$ to become $= F$. Since box sizes are updated on only one side each time, we assume that we still have $size_{1-i}(s_1, s_2) > F$. Each time $size_i(s_1)$ is decreased between s_1 and s_2, it has to be due to $\gamma(e)$ being enumerated into A_i, as we may assume that $forbid(e') > s_1$ at the end of stage s_1 for all $e' > e$. Each time $\gamma(e)$ enters A_i, we must also have a corresponding decrease in $|\Theta^+_{1-i}|$. That is, there is some number which drops out of Θ^+_{1-i} (and goes into Θ^-_{1-i}), and this has to be one of the numbers in the set $\Theta^+_{1-i}[s_1] \cup \bar{\Theta}_{1-i}[s_1]$. Hence, $size_i(s_1)$ can be decreased at most $|\Theta^+_{1-i}[s_1] \cup \bar{\Theta}_{1-i}[s_1]| = s - 1 - |\Theta^-_{1-i}[s_1]| = s - 1 - gap_{1-i}(s_1) < s - 1 - F$ times during the interval from s_1 to s_2. Since $size_i(s_1, s_1) = s_1 - 1$, a contradiction follows. □

The above lemmas show that for both i, Θ_i is of bounded size. We get a contradiction to the fact that $\gamma(e)$ is moved infinitely often. Lastly, it is not hard to see that Γ is total, and gives a correct reduction. □

As mentioned at the beginning of this section, the search for a combinatorial characterization for the K-trivials remains open. Our result raises a number of interesting questions. How slowly can we allow h to grow, such that there are c.e. sets A and B which are both jump traceable at order h, but $A \oplus B$ is of complete Turing degree? We know from Cholak, Downey and Greenberg [4] that h cannot be arbitrarily slow growing. On the other hand our strategy is rather specific, and does not even work for the order $\frac{1}{2}n$.

We know that not every c.e. set jump traceable at identity is K-trivial, but is it true that every c.e. set which is K-trivial is jump traceable at identity? An analysis of the Decanter method in Downey [5] shows that if A is K-trivial, then A is jump traceable at all orders h such that $\sum_{n=1}^{\infty} h(n)^{-1} < \infty$ (see Barmpalias, Downey and Greenberg [1], Theorem 1.3). We do not know if our method adapts to the case of $h(n) = n$. We also mention that Barmpalias, Downey and Greenberg [1] have refuted Conjecture 2.1 by a direct construction:

Theorem 2.7 (Barmpalias, Downey and Greenberg [1])**.** *There is a c.e. set A which is not K-trivial, but for every order h such that $\lim_n \frac{h(n)}{n} = \infty$ and for every e, the function Φ_e^A is traceable at bound h.*

In fact, our result implies Theorem 2.7. To see this, note that we can code $\{\Phi_e^A(x)\}_{e,x\in\mathbb{N}}$ efficiently into a single partial A-computable function Ψ, by letting $\Psi^A(n) = \Phi_e^A(x)$ where $2^e(2x+1) = n$. It is clear that this coding of pairs (e, x) is injective. Hence we apply the proof above to produce a non-K-trivial c.e. set A where Ψ^A is traced at identity. We can then uniformly obtain, for each e, a trace for Φ_e^A with bound $2^e(2x+1)$. Any order h has to dominate this function lest $\liminf \frac{h(n)}{n}$ is bounded above by 2^{e+1}.

3. The first glimpse of hyper jump traceability

Part of the motivation of this paper is to study variations of strong jump traceability, by looking at relativizations. We are interested in studying these notions in conjunction with the lowness properties of a set A. These motivations give rise to the following definition.

Definition 3.1. We say that A is *strongly jump traceable by X*, if for every X-computable order h^X, there is a total computable function g, such that for all x, we have $|W^X_{g(x)}| \leq h^X(x)$ and $J^A(x) \in W^X_{g(x)}$.

We let $SJT(X)$ denote the class of c.e. sets strongly jump traceable by X. Note that this is not a true relativization, which would call for $J^{A\oplus X}$ to be traced instead of just J^A. There is no difference if $X = \emptyset$; in both cases we have the class of strongly jump traceable c.e. sets. However things behave a little differently if we consider an arbitrary $X >_T \emptyset$. Having X as an oracle helps us in the sense that we have more traces at our disposal (now we have all X-c.e. traces instead of just c.e. traces). However, we also suffer a drawback because we now have to trace J^A respecting more order functions (we have to respect all X-computable orders instead of just computable orders). Therefore it is not immediately clear that if A is strongly jump traceable, then it must also be strongly jump traceable by all sets X. Some sets X might produce very slow growing X-order functions which we cannot trace using only X-c.e. traces. The only thing we are sure of is that $A \leq_T X \Rightarrow A \in SJT(X)$. In fact, in Theorem 3.5 we produce a strongly jump traceable c.e. set A such that $A \notin SJT(X)$ for some c.e. X.

The first indication that we can further extend the notion of strong jump traceability in a meaningful way, comes from the investigation of the transitivity of the binary relation "$A \in SJT(X)$".

Lemma 3.2. *Suppose that $g(x)$ is a total computable function. Then, there is a total computable function $\alpha(x, n)$, such that for all numbers x and n, and for any set A,*

$$J^A(\alpha(x,n)) = \begin{cases} n^{th} \text{ value in the enumeration of } W^A_{g(x)}, & \text{if } |W^A_{g(x)}| \geq n, \\ \uparrow, & \text{if } |W^A_{g(x)}| < n. \end{cases}$$

Proof. By the *s-m-n* Theorem. □

Suppose we wanted to show that the relation $A \in SJT(X)$ is transitive. We might make a first attempt at proving transitivity below $\emptyset$, and then try and relativize the proof. That is, we can try first of all to show that if A is strongly jump traceable in X, and X is strongly jump traceable, then A is also strongly jump traceable. While this is true (Corollary 3.4), the proof does not actually relativize to give transitivity in the general setting. The following Theorem 3.3 gives the correct relativization:

Theorem 3.3. *Suppose that $A \in SJT(B)$ and $B \in SJT(C)$, such that $C \leq_T B$. Then, $A \in SJT(C)$.*

Proof. Let h be a C-order. Define

$$p(x) = \sqrt{h(x)} \text{ (rounded down).}$$

Then, p is a B-computable order, and so there is some total computable g, such that for all x,

(i) $|W^B_{g(x)}| \leq p(x)$, and
(ii) $J^A(x) \in W^B_{g(x)}$.

Let α be the function from Lemma 3.2. Define the C-computable function q by : $q(z) = \sqrt{h(x)}$, where x is the least number such that $z \leq \alpha(x, i)$ for some $i \leq p(x)$. Then, q is a C-order, and so there is some total computable g', such that for all x,

(i) $|W^C_{g'(x)}| \leq q(x)$, and
(ii) $J^B(x) \in W^C_{g'(x)}$.

Finally, we define the uniformly C-c.e. sequence $\{V_x\}_{x\in\mathbb{N}}$, by letting

$$V_x = \bigcup_{i=1}^{p(x)} W^C_{g'(\alpha(x,i))}$$

We can easily see that for all x,

(i) $|V_x| \leq p(x)q(x) \leq h(x)$, and
(ii) $J^A(x) \in V_x$,

thus giving a trace for J^A from C. □

Corollary 3.4. *If $A \in SJT(B)$, and B is strongly jump traceable, then A is strongly jump traceable.*

We need the extra condition of $C \leq_T B$ to ensure that we do not get more order functions as we pass from one side to the other of the binary relation. Corollary 3.4 is interesting on its own, in that it says that any set which is "extremely low" over another which is extremely low, must itself be also extremely low. This brings us back to the question as to whether we can remove the assumption "$C \leq_T B$" in Theorem 3.3. We provide a negative answer to this:

Theorem 3.5. *There are strongly jump traceable c.e. sets A and B, such that $A \notin SJT(B)$.*

This theorem has a few interesting consequences. Firstly it shows that "$A \in SJT(X)$" is not a transitive relation, and so this relation cannot be used to generate a degree structure in the usual way. It also answers a question regarding the relationship of SJT with upper and lower Turing cones. In particular, while every set is strongly jump traceable by any set which computes it, it is however not true that every set is strongly jump traceable by a set which it computes. This is not

even true for sets which join to it, i.e. if $A \equiv_T A_0 \oplus A_1$ then $A \in SJT(A_0)$ is not true in general.

Thirdly, one might expect that if a set B is very low in terms of its power when serving as an oracle, then every set which is strongly jump traceable is also strongly jump traceable by B. This is not the case. Lastly and perhaps most importantly, this theorem shows that the class of strongly jump traceable c.e. sets and the class $\mathcal{H} := \bigcap\{SJT(W) \mid W \text{ is c.e.}\}$ are different. This suggests that we can look to $\mathcal{H}$ as a possible candidate for an even stronger version of computational lowness. This will be taken further in the next section, and will form the central theme for the rest of the paper. For now, let us return to the proof of Theorem 3.5.

3.1. **Requirements.** We build the c.e. sets A and B, and a Turing functional Ψ to meet the requirements :

$$\begin{aligned} \mathcal{N}_e &: \text{If } h_e \text{ is an order, make } A \oplus B \text{ jump traceable relative to } h_e. \\ \mathcal{P}_e &: \text{Defeat the } e^{th} \text{ trace. That is, for some } x, \text{ either} \\ &\quad |T^e_x| \geq \Psi^B(x), \text{ or else } J^A(x) \notin T^e_x. \end{aligned}$$

Here, we let $\{T^e_x\}_{x\in\mathbb{N}}$ be the e^{th} c.e. B-trace, in some effective listing of all traces computing with an oracle. For simplicity we drop the oracle B from the notations. We let $J^X(e)[s]$ be the value of the universal jump function $\{e\}^X(e)[s]$ at stage s. The use of $J^X(e)[s]$ (if convergent) is $j^X(e,s)$. We also let $\{h_e\}_{e\in\mathbb{N}}$ be an effective list of all partial computable functions.

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 $x >$ 1+any number used or mentioned so far. We drop the stage number from the notations if the context is clear.

3.2. **Discussion of a related result.** The proof of this theorem uses the ideas in [12] closely. We will describe the combinatorics involved in the discussion that follows. We begin by looking at the strategy used to prove the following:

Proposition 3.6 (Ng [12])**.** *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 $\tilde{h}$.*

We build a trace $\{V_x\}_{x\in\mathbb{N}}$ to trace J^A with respect to h. This strategy corresponds to negative action on A (it wants to preserve A), handled by negative requirement $\mathcal{N}_e$ which wants to impose some amount of restraint on A each time it sees a computation $J^A(e)[s]\downarrow$. On the other hand we are also building an order function $\tilde{h}$, and we have to try and diagonalize against the e^{th} trace with respect to $\tilde{h}$. This action is positive on A (it wants to change A), and will be handled by the positive requirement $\mathcal{P}_e$, which makes enumerations into A to force T^e_x to fill up for some x.

Initially V_x starts off as an *original* $(h(e)-1)$*-box*. Every time the restraint that V_x is holding is injured by some positive action, we say that V_x receives a *promotion* in size, i.e. it now has a smaller box size. By the time $\mathcal{N}_e$ becomes a 0-box (corresponding to $|V_x[s]| = h(e)$), we will have to ensure that no positive action in future can destroy the current $J^A(e)[s]$ computation. This ensures that the negative requirements are satisfied.

Let us now turn our attention to the positive requirement $\mathcal{P}_e$. We first describe how to meet such a requirement in isolation. $\mathcal{P}_e$ would attempt to defeat the e^{th} 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 an infinite computable list of indices x, for which we are allowed to enumerate axioms for $J^X(x)$ on any oracle $X \subset \mathbb{N}$. $\mathcal{P}_e$ will pick one of these x and enumerate (possible)

axioms for $J^A(x)$ with use $u(e,s)$ on A. Each time the value $J^A(x)[s]$ shows up in the trace T^e_x, we would put the use $u(e,s)$ into A to cancel all the previous axioms, and enumerate a new axiom $\langle x, y, A_s \restriction_{u(e,s+1)} \rangle$ for $J^A(x)$ (with fresh y and $u(e,s+1)$). 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 previously, 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 pre-determined. 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-sized boxes to consider. Consider a requirement $\mathcal{P}$ that wants to diagonalize against 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 2 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 1 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. Firstly, 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.

We now describe the strategy in the general setting. 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*. 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^2_0-boxes (and smaller ones). In phase 2, $\mathcal{P}_2$ would be injured by C^1_0-boxes. To prevent the different positive requirements from interfering with each other, we ensure that no new C^2_1-boxes are ever created by the actions of $\mathcal{P}_3, \mathcal{P}_4, \cdots$. Thus, $\mathcal{P}_3$ would be injured by C^3_1-boxes in phase 1, by C^3_0-boxes in phase 2, and by C^2_1-boxes in phase 3. The values C^3_0, C^3_1, C^3_2 are defined inductively.

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 (from $C^e_{e-3}, \cdots$ down to C^e_0 and 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 $C^{e-1}_{e-2}, C^e_0, \cdots, C^e_{e-2}$ sufficiently spaced out, we will be alright.

The crucial point is that these threshold values can be determined in advance, and depend only on the given h. These thresholds are used in the definition of $\tilde{h}$. Hence there is a general procedure Λ, such that given an order function h, $\Lambda(h)$ outputs the order $\tilde{h}$ satisfying the statement of Proposition 3.6. This procedure Λ is used as an atomic strategy in the proof of:

Proposition 3.7 (Ng [12])**.** *The set* $\{e \in \mathbb{N} : W_e$ *is strongly jump traceable*$\}$ *is* Π^0_4*-complete.*

We will need to make use of the procedure Λ, to calculate the relevant threshold values in Section 3.6. We will describe how this is done in the next section.

3.3. **Description of strategy.** We now return to the proof of Theorem 3.5. How are we going to make use of the strategy described above in Section 3.2? It is impossible to make A strongly jump traceable, yet at the same time make A not jump traceable relative to a computable order function $\tilde{h}$. However if we allow $\tilde{h}$ to be computable in a c.e. oracle, we can combine the positive and negative requirements stated in Section 3.1. We describe how to do this. To make A and B strongly jump traceable, we make $A \oplus B$ strongly jump traceable.

We describe exactly how we intend to carry out the strategy for a single $\mathcal{N}_e$. We need to make $A \oplus B$ jump traceable respecting h_e. Suppose α is a node on the construction tree which is assigned the requirement $\mathcal{N}_e$. It splits its task into infinitely many substrategies $ST_0, ST_1, \cdots$. For each $k \in \mathbb{N}$, the k^{th} substrategy ST_k works by the following: it waits for $h_e(k)\downarrow$, and when $J^{A\oplus B}(k)[s]$ next converges, we would enumerate the value into V^α_k (the sequence $\{V^\alpha_x\}_{x\in\mathbb{N}}$ is built at α), and restraint $A \oplus B$ on the use. At this point in time, we set $size^\alpha_k = h_e(k) - 1$ (where $size^\alpha_k$ denotes the corresponding α-box size). When $size^\alpha_k = 0$, 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 the discussion in Section 3.2, 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 $r_0, r_1, \cdots$, where r_k is the restraint function for ST_k, and let different positive strategies below α, 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 works in the case when h is the identity order function (otherwise we just make suitable adjustments). For more details on this, see Theorem 7.3 of [5].

Now we describe the positive requirements. We first consider (the weaker case of) only making A strongly jump traceable. We also make A not jump traceable relative to B via the order $\tilde{h} = \Psi^B$. They work in almost the same way as described above in Section 3.2, with two main differences (due to the fact that we now have an oracle B):

(1) To defeat the e^{th} trace, we want it to fill up with trash. We can stimulate responses from T^e_x by emulating the computation $J^A(x)$ at some x. However, when some number enters T^e_x, it will have a corresponding B-use. To keep the number in T^e_x, we have to now preserve B on this use. This gives the positive node an extra responsibility - holding B on some use. This extra restraint is finitary and can be very easily made compatible with the rest of the construction.
(2) $\tilde{h}$ is now build globally. Suppose a positive node σ needs to make the statement of $\mathcal{P}_e$ true at some x, with $\tilde{h}(x) = \Psi^B(x) = 2$. Now we have the ability to change Ψ^B whenever we want. σ might start off by picking a follower x and setting $\Psi^B(x)[s] = 2$. At a later stage σ might be blocked by an increased A-restraint and hence cannot proceed with diagonalization with x anymore. It will then pick a fresh $x' > x$ and clear all $\Psi^B(y)$ axioms for $y > x$ which may have been set by other positive nodes in the meantime. σ can do this by enumerating $\psi(x)$ into B, and redefine $\Psi^B(y) = 2$ for all $x \leq y \leq x'$, and proceed with diagonalization with new follower x'. It is not hard to arrange things so that the A-restraint on σ is finite, so that this injury only happens finitely often.

We have no problems putting together the positive and negative requirements, if we only needed to make A strongly jump traceable, because each time σ needs a new follower for diagonalization, we simply change B to create more followers for σ. The problem now is that to make $A \oplus B$ strongly jump traceable, the negative requirement $\mathcal{N}_e$ also needs to set up restraints to protect B. The following summarizes the actions of the positive and negative nodes:

$\mathcal{P}_e$	Puts numbers into A and B. Prevents numbers from entering B.
$\mathcal{N}_e$	Prevents numbers from entering A and B.

Each enumeration into B made by σ will now cause boxes above σ to be promoted as well. Thus, in the process of trying to create more followers for diagonalization, σ will further promote boxes which will return and block σ when it starts the next round of diagonalization.

To take a first step towards solving this problem, we will first ensure that σ makes less enumerations into B. Enumerations into B will have to obey restraints from above, since we have boxes tracing $J^{A\oplus B}$. To cut down on the number of enumerations made into B, σ will have to do a little more work. Before σ begins any diagonalization attempt, it will first compute the relevant thresholds $C_{n,-1}, \cdots, C_{n,n-1}$ for some selected n. These thresholds will be computed based on the procedure $\Lambda(\min\{h_{e_0}, \cdots, h_{e_k}\})$, where σ believes that $h_{e_0}, \cdots, h_{e_k}$ are orders. Of course these values will not return if the h_{e_i} are not true orders, but we get to change our mind on Ψ^B, so we can carry on with the rest of the construction while waiting for these values to return. If and when these values return, σ will then adjust Ψ^B accordingly by changing B once, and setting aside enough followers x with $\Psi^B(x) = n$. This ensures that σ only changes B once.

Note that σ will not be able to change B whenever it wants, because we might have boxes of small size blocking B. On the other hand, we have to be careful when we allow σ to change B, because whenever σ changes B, it will promote some boxes above σ (remember these are boxes tracing $J^{A\oplus B}$). If we are not careful, σ will promote boxes to take on very small sizes and affect the other positive nodes above σ.

In view of these considerations, we will arrange for σ to have a separate threshold value, called $setter(\sigma)$, which is to be used only when σ is in the preliminary stage of setting the parameters needed for diagonalization. This number is chosen to be large, so that σ does not interfere with the positive requirements above σ (i.e. σ does not promote any boxes to have size $< setter(\sigma)$). When σ is computing the thresholds it needs for diagonalization, it will monitor all boxes currently of size $\leq setter(\sigma)$. Once any of them imposes an $A \oplus B$ restraint above $\psi(x)$, σ will have to abandon its current set of parameters, and pick new $n' > n$ and $x' > x$. So long as no new $setter(\sigma)$-boxes are created, σ will eventually be able to begin diagonalization.

A final technical point to consider: σ makes an enumeration into B before it actually begins diagonalization. Therefore, boxes are already promoted by σ before σ begins its Λ-strategy. Fortunately, this sort of promotion due to changes in B only happens once, so when σ actually begins diagonalization, it knows that a current b-box it encounters was actually a $(b+1)$-box. The threshold values can be pre-chosen to take this into account.

3.4. **Construction tree layout.** The construction takes place on a subtree of the full binary tree. Nodes of length $2e$ are assigned the requirement $\mathcal{N}_e$, with outcomes $\infty <_{left} f$. This stands respectively for the Π^0_2 (Σ^0_2) fact that h_e is (is not) an order function. The positive requirements have to act based on guesses to these outcomes, and have to deal with increased $(A \oplus B)$-restraint. Nodes of length $2e+1$ are assigned the requirement $\mathcal{P}_e$, with a single outcome 0; these nodes are each involved in only finitely much action.

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 $\alpha(i) <_{left} \beta(i)$. We say that α *is a* $\mathcal{Q}$*-node* if α is assigned the requirement $\mathcal{Q}$. $\mathcal{N}_e$ nodes are *negative nodes* wanting to impose restraint on $A \oplus B$, while $\mathcal{P}_e$-nodes are *positive nodes* wanting to put things into A and B. Note that even though we consider a $\mathcal{P}_e$-node to have primarily a positive role, there will be times when it will want to preserve a segment of B. This happens when thrash gets filled up in the B-trace, and we want to keep it filled. There will only be however, a finite amount of B-restraint due to this.

3.5. **Notations used in the formal construction.** At each $\mathcal{N}_e$-node α, we build a u.c.e. sequence $\{V^\alpha_x\}_{x\in\mathbb{N}}$; the purpose is to trace $J^{A\oplus B}$ in the event that h_e turns out to be an order. We define the length of convergence for h_e at stage s, to be:

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

Sometimes we will write $l(\alpha, s)$ in place of $l(e, s)$, and h_α in place of h_e. Since h_e is (partial) computable, hence it is clear that $l(e, s)$ is non-decreasing over time, and $l(e, s) \to \infty$ iff h_e is an order. We let $size^\alpha_k[s]$ denote the *size of the* V^α_k*-box at stage* s. 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 $J^{A\oplus B}(k)$-computation is injured after being traced in V^α_k. When $size^\alpha_k$ reaches 0, the restraint that α imposes for $A \oplus B$ must be obeyed by all.

Each $\mathcal{P}_e$-node α is in charge of diagonalizing the e^{th} B-trace. We do this by enumerating a functional Δ_α^A with index x for some x, which simulates the value of $J^A(x)$. If at any point in time we want give up all axioms enumerated into Δ_α^A, we have to pick a new index $x' > x$ and work on $J^A(x')$. There are a few other parameters associated with α:

- We let $x(\alpha, s)$ denote the index of the functional which α is enumerating at stage s, with A-use $\delta(\alpha, s)$. These indices are chosen from an infinite list of indices supplied by the Recursion Theorem.
- $region(\alpha, s)$ denotes the number n, such that α will define $\Psi^B(x) = n$ for every index x that it uses after stage s. In other words, this represents the number of elements α has to enumerate into A, and is also the amount of unwanted thrash that α has to fill T_x^e up with, at some x.
- We record the number of elements that α has managed to force into $T_{x(e)}^e$ by the parameter $attempt(\alpha, s)$. When this parameter value reaches $region(\alpha)$, then α's work would be done and is permanently satisfied.
- $setter(\alpha, s)$ denotes the smallest number q, such that α is allowed to promote $(q+1)$-boxes while setting the axioms for Ψ^B.

The stage s B-use of the functional $\Psi^B(x)$ that we are enumerating is denoted by $\psi(x, s)$. To ensure that Ψ^B is total and non-decreasing, we adopt the following convention: when we define $\Psi^B(x)[s] \downarrow = y$ with use u at a particular stage s, we mean that we enumerate the axioms $\langle x', y, B_s\restriction_{u+1}\rangle$ for all $x' \leq x$, where no other axioms currently apply with input x'. Also set $\psi(x', s) = u$ for all such x'. Note that Ψ^B is maintained globally, so this ensures that its axioms are consistent regardless of which portion of the construction tree we visit.

If α is a negative node, then for each $n, s \in \mathbb{N}$, we let

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

For a positive node σ, we let

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

where $Z^-(\sigma) := \{\beta \subset \sigma \mid \beta \text{ is negative, and } \beta^\frown\infty \subseteq \sigma\}$; these are all the negative nodes $\beta \subset \sigma$ which have to trace $J^{A\oplus B}$ at a certain order. Informally, the value $S(\alpha, n)[s]$ denotes the maximum number of different values $j^{A\oplus B}(k, s)$ can take, for the set of k's such that $size_k^\alpha[s] < n$. The number $S(\sigma, n)[s]$ is used to provide a bound on the number of times σ can be blocked by some current $(n-1)$-box (or less) of some negative $\beta \subset \sigma$. The combinatorics here is similar to that which is used in [12]; we have attempted to retain the notations and terminologies from [12] as much as we can in order not to confuse the reader, and for more information we refer the reader to [12]. The threshold values will be computed during the construction itself, and their values will depend on the current observed situation.

The parameters $\{C_{n,k} \mid n > 0 \ \wedge \ k < n\}$ and $\{I_n \mid n > 0\}$ helps us keep track of the threshold values, and as mentioned above, will only be computed during the construction. These parameters are all set to $\uparrow$ initially. The order function Ψ^B will have domain divided into intervals which we call *regions*. The n^{th} region will consist of all the numbers x such that $\Psi^B(x) = n$; this is related to the parameter $region(\alpha, s)$ mentioned above in the following way: the values $C_{n,-1}, C_{n,0}, \cdots, C_{n,n-1}$, and I_n are associated with the n^{th} region, and will get their values evaluated and assigned by the positive node α such that $region(\alpha) = n$. Once α sets these parameter values, it will use them to define $\Psi^B = n$ on the n^{th} region. Informally, $C_{n,k}$ represents the critical threshold value of α, when

$attempt(\alpha) = n - 2 - k$. I_n represents a bound on the number of indices that α needs to use for the n^{th} region.

We describe the use of indices in some detail: there will be infinitely many indices $\nu_0 < \nu_1 < \cdots$ set aside for use by the positive nodes. If α is a positive node then $x(\alpha)$ is always chosen from this list. Once α finishes computing all of the threshold values it needs, it will set Ψ^B to a constant value over each interval $\{x \in \mathbb{N} \mid \nu_{i-1} < x \leq \nu_i\}$, for all ν_i in the $region(\alpha)^{th}$ region. Thus, whenever we refer to $x(\alpha)$ or Ψ^B, we are referring to the values modulo the intervals partitioned by the ν_i's. That is, $\Psi^B(i)$ will refer to the (common) value of $\Psi^B(x)$ for all $\nu_{i-1} < x \leq \nu_i$, and we write $x(\alpha) = i$ instead of $x(\alpha) = \nu_i$.

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

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

where $r = region(\alpha, s)$ and $a = attempt(\alpha, s)$. This represents the current threshold value that α has to obey, based on its progress in its atomic strategy.

When we *initialize a negative node* α at stage s, we set $V^\alpha_x = \emptyset$ for all x, and set $size^\alpha_x[s] = h_\alpha(x) - 1$ for all $x < l(\alpha, s)$. As for a positive node α, there are three ways in which the atomic strategy of α may be restarted.

(1) *A (full) initialization to* α: we set $region(\alpha), x(\alpha), setter(\alpha)$ and $\delta(\alpha)$ all $\uparrow$, and set $attempt(\alpha) = 0$. In this case, α has to restart its strategy due to reasons that it had not foreseen, for example due to an incorrect guess, or some positive node acting above α. All parameter values are canceled.

(2) *A self-imposed initialization*: we set $region(\alpha), x(\alpha)$ and $\delta(\alpha)$ all $\uparrow$, and set $attempt(\alpha) = 0$. This happens if the following scenario takes place. While α is waiting to compute its threshold values, some higher priority box of size $\leq setter(\alpha)$ imposes a restraint on $A \oplus B$, above $\psi(x(\alpha))$. Since α always has to respect boxes of size at most $setter(\alpha)$ when setting its Ψ^B-axioms, this means that the current position of $x(\alpha)$ is too small. So, we have to abandon the current $x(\alpha)$ value, *as well as* the current $region(\alpha)$ value, since α is no longer able to effect changes in Ψ^B below the current $region(\alpha)^{th}$-region.

Such initializations will be performed only when α is visited. We want to distinguish between full and self-imposed initializations because we have to be careful about when we cancel the $setter(\alpha)$ parameter.

(3) *A reset of* α: the third way to disrupt the atomic strategy of α, is through the activity of some node in $Z^-(\alpha)$. α is fully prepared for injury of this sort, and would have reserved enough indices in the $region(\alpha)^{th}$ region for this. In this case we do the following. If $attempt(\alpha, s) = region(\alpha, s)$ (i.e. α is permanently satisfied) or $x(\alpha) = region(\alpha, s) + I_{region(\alpha,s)}$ (i.e. α has run out of indices), do nothing. Otherwise increase $x(\alpha)$ by 1, set $\delta(\alpha) = \uparrow$, and set $attempt(\alpha) = 0$.

If σ is a positive node and $k \in \mathbb{N}$, we define the *k-restraint function on* σ at stage s to be

$$r(\sigma, k, s) = \max\{j^{A\oplus B}(p, s) \mid J^{A\oplus B}(p)[s] \downarrow \text{ for some } \beta \in Z^-(\sigma) \wedge p < l(\beta, s) \wedge size^\beta_p[s] \downarrow \leq k\}.$$

That is, $r(\sigma, k)$ represents the total amount of restraint on $A \oplus B$ imposed on σ, by some current k-box (or less) above σ.

We say that a positive $\mathcal{P}_e$-node α *requires attention* at stage s, if $attempt(\alpha, s) < region(\alpha, s)$ provided they are defined, and one of the following (A0)-(A4) holds:

(A0) $region(\alpha) \uparrow$.

(A1) $region(\alpha)\downarrow = n$ for some n, and one of $C_{n,-1}, C_{n,0}, \cdots, C_{n,n-1}$, or I_n has not yet received an assignment.
(A2) There is no computation in Δ_α^A which currently apply.
(A3) There is a computation in Δ_α^A which currently apply with use $\delta(\alpha, s)$, such that $\delta(\alpha, s) < r(\alpha, threshold(\alpha))[s]$.
(A4) There is a computation in Δ_α^A which currently apply with use $\delta(\alpha, s)$ and value $r = \Delta_\alpha^A(x(\alpha))[s]$, such that r has shown up in the trace $T^e_{x(\alpha,s)}$.

(A0)-(A4) can be thought of to be the state of the atomic strategy of α. When α has just been initialized, (A0) holds and we will pick a fresh value for $region(\alpha)$. Once that has been done, (A1) will now apply and we will have to wait for all the relevant parameters to be defined. α will next move on to (A2) and begin diagonalization with Δ_α^A. After that, α will move to either (A3) or (A4); if (A3) holds then the restraint on α from above has increased beyond $\delta(\alpha, s)$ - there is some high priority box blocking the strategy of α and we have to reset α. If (A4) holds then the opponent has responded by filling $T^e_{x(\alpha)}$ up. We can now turn the contents of $T^e_{x(\alpha)}$ into thrash by enumerating $\delta(\alpha, s)$ into A.

Let α be a positive node, and $p, s \in \mathbb{N}$. We *make α promote all boxes with use p at stage s* by doing the following: for each k such that there is some $\beta \in Z^-(\alpha)$ and $k < l(\beta, s)$, such that $J^{A\oplus B}(k)[s]\downarrow$ with use $j^{A\oplus B}(k, s) > p$, we decrease $size_k^\beta[s]$ by 1. That is, this action adjusts the size of all boxes with use larger than p, because an enumeration of p into A or B is imminent.

3.6. **The Construction.** 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, and equivalently that s is an α-stage, if $\delta_s \supset \alpha$. The nodes along δ_s will get to act at stage s. At stage $s = 0$, initialize all nodes and set $\delta_s = \langle\rangle$.

Suppose $s > 0$, and assume that $\alpha = \delta_s\restriction_d$ has been defined for $d < s$. We first consider the case where α is an $\mathcal{N}_e$-node. If

- $l(\alpha, s^-) < l(\alpha, s)$ where s^- is the previous α-stage, and
- $h_e(l(\alpha, s)) > C_{n,r}$ for every $n, r \in \mathbb{N}$ such that $C_{n,r}\downarrow$ and $n = region(\sigma)$ for some $\sigma \supseteq \alpha^\frown\infty$,

then we say that stage s is α-*expansionary*, otherwise it is non-α-expansionary. If stage s is α-expansionary, do the following: for all $k < l(\alpha, s)$ such that $J^A(k)[s]\downarrow$, we enumerate the value $J^A(k)[s]$ into V_k^α. For all $k < l(\alpha, s)$ such that $size_k^\alpha$ has not yet been assigned a value, we update the size and set $size_k^\alpha = h_e(k) - 1$. Set $\delta_s(d) = \infty$. On the other hand if stage s is non-α-expansionary, we set $\delta_s(d) = f$, and do nothing else.

Now assume that α is a $\mathcal{P}_e$-node. Let $\delta_s(d) = 0$. If α does not require positive attention, we do nothing. Otherwise, initialize all nodes $\beta \supset \alpha$, take the appropriate action listed below, and declare that α has *received attention* at stage s. Let x be the least such that (Ax) holds.

- $x = 0$: Pick a fresh number n for $region(\alpha)$, and set $x(\alpha) = n$. Also set $C_{n,-1} = n$ and $C_{n,0} = n + 3$. Set $\Psi^B(n)[s]\downarrow = n$ with fresh ψ-use. Furthermore if $setter(\alpha)$ is undefined, assign $setter(\alpha) = n$.
- $x = 1$: go down the following list, pick the first that applies, and perform the action stated.
  - If $\psi(x(\alpha), s) < r(\alpha, setter(\alpha))[s]$, do a self-imposed initialization of α.
  - If there is a smallest $q < n = region(\alpha)$ such that $C_{n,q}\uparrow$, set $C_{n,q} = C_{n,q-1} + 3 + n + nS(\alpha, C_{n,q-1})[s]$.
  - Otherwise if $C_{n,q}\downarrow$ for all $q < n$, set $I_n = S(\alpha, C_{n,n-1})[s]$. Adjust the size of boxes by making α promote all boxes with use $\psi(x(\alpha), s)$.

Additionally, we put $\psi(x(\alpha), s)$ into B to clear the definition of $\Psi^B(x')$ for all $x' \geq x(\alpha)$. Set $\Psi^B(n + I_n)[s] \downarrow = n$ with fresh ψ-use.

- $x = 2$: we enumerate a computation $\Delta^A_\alpha(x(\alpha))[s] \downarrow = s$ with fresh use $\delta(\alpha, s)$.
- $x = 3$: reset α.
- $x = 4$: adjust the size of boxes by making α promote all boxes with use $\delta(\alpha, s)$. Enumerate $\delta(\alpha, s)$ into A, and increase $attempt(\alpha, s)$ by 1.

This concludes the definition of δ_s. Finally, we initialize all nodes $\beta >_{left} \alpha$, and proceed to the next stage. The true path of the construction is defined as usual to be the leftmost path visited infinitely often during the construction. If α is visited and receives attention under (A0) or (A1), we say that α *is setting its parameters*. That is, α is computing the initial values of various parameters it needs to start its atomic strategy. When α finishes setting its parameters, it will mark the end of the process with an enumeration into B, to adjust Ψ^B. Thereafter, α will begin diagonalization under (A2)-(A4), and never return to (A0)-(A1) again unless it is initialized.

3.7. **Verification.** The main bulk of the verification will focus on showing that various counting arguments work. Again we follow [12] closely. In the following lemma we are going to count the number of different strings ρ, such that J^ρ can be traced by a negative node. This affects the combinatorics used by the positive nodes. In (i) below, we show that the maximum number of different restraints that a current b-box can put up 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 σ-priority, is at most $S(\sigma, n)$. Because the part of the oracle accessed during computations can be different while possibly having the same length at different stages, we focus on strings ρ for J^ρ, rather than the length of the use $j^{A\oplus B}$.

Lemma 3.8. (i) *Let β be a negative node on the true path with true outcome ∞. Let $k \in \mathbb{N}$, and t_0 be a stage after which β is never initialized*[6]*, such that $size^\beta_k[t_0] \downarrow$. Then,*

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

(ii) *Let σ be a positive node on the true path, $n \in \mathbb{N}$, and t_1 be a stage after which σ is never initialized. Then, the number of pairs $\langle (A \oplus B)\restriction_{j(k)} [s], k\rangle$ for which*
   (a) *$J^{A\oplus B}(k)[s] \downarrow$,*
   (b) *for some $\beta \in Z^-(\sigma)$, we have $size^\beta_k[t_1] < n$,*
   (c) *$\delta_s \supset \sigma$ and $s \geq t_1$,*
is at most $S(\sigma, n)[t_1]$.

Proof. This is the same as Lemma 3.6 of [12], we include the proof here for the benefit of the reader.

(i): this should be easy to see, because informally the statement says that if we currently have a b-box, then there can only be at most $b+1$ many possible observed versions ρ of the use for J^ρ, since we can have a different observed use only if the box is promoted.

Formally, we suppose for a contradiction, that there are stages $t_0 \leq s_0 < s_1 < \cdots < s_m$ such that $(A \oplus B)\restriction_{j(k)} [s_i] \neq (A \oplus B)\restriction_{j(k)} [s_{i+1}]$ for all $i = 0, \cdots, m-1$ (where $m = 1 + size^\beta_k[t_0]$). For each i, the change $(A \oplus B)\restriction_{j(k)} [s_i] \neq (A \oplus B)\restriction_{j(k)} [s_{i+1}]$ must have been caused by some positive node $\beta' \supseteq \beta^\frown\infty$ receiving attention at some stage t where $s_i \leq t < s_{i+1}$. Hence β' must have promoted the box V^β_k and

[6] We have not yet shown that such a stage must exist. For this lemma we assume its existence.

decreased $size_k^\beta$ while it is receiving attention 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 $t \geq s_{m-1}$ including s_{m-1} itself, no node β' extending $\beta^\frown\infty$ is allowed to make an enumeration into A or B below $j(k, s_{m-1})$, since $r(\beta', y, t) \geq j(k, s_{m-1})$ for all $y \in \mathbb{N}$, a contradiction.

(ii): Using part (i). □

Lemma 3.9. *Let σ be a positive node, receiving attention at some stage s, and let N be the largest parameter value of σ at stage s. Suppose $\beta \in Z^-(\sigma)$ and $k \in \mathbb{N}$. Then, the V_k^β-box cannot be promoted to become an N'-box after stage s (for all $N' \leq N$) by a node either extending σ, or to the right of σ.*

The statement of the lemma is rather long, but is really quite intuitive. What it says is the following. Take σ to be a positive node receiving attention at stage s, where it sets one of its parameter values $= N$ at stage s. Consider the V_k^β-boxes (these have higher priority than σ). It would be bad if a lot of new boxes of this type are promoted to have a small box size $\leq N$, since this messes up the combinatorics set up by σ. The lemma says that any such promotion cannot be due to nodes of lower priority than σ. Therefore, once σ asserts control during the construction, any such promotion which creates boxes of small size have to be due to σ's actions alone; this isolates the effects of the other positive nodes.

Proof of Lemma 3.9. Let σ' be a possible counterexample for the promotion of V_k^β. Since σ' is initialized at stage s, this means that $setter(\sigma')$ and $threshold(\sigma')$ must be larger than N whenever they are defined after stage s. This makes promotion of the type mentioned impossible, because $r(\sigma', N+1)$ will have to be obeyed. □

Next, we will argue inductively that along the true path, all requirements are satisfied. We prove the following simultaneously by induction on $|\alpha|$, where α is on the true path:

(I1) if α is a $\mathcal{Q}$-node, then the statement of $\mathcal{Q}$ is satisfied,
(I2) if α is positive then it requires attention finitely often.

Suppose (I1) and (I2) holds for all $\beta \subset \alpha$, and α is on the true path. Firstly, suppose that α is an $\mathcal{N}_e$-node, and that h_e is an order. By induction hypothesis, α is initialized only finitely often, so we can consider the true version of $\{V_k^\alpha\}$. Fix a $k \in \mathbb{N}$. If $J^A(k)\downarrow$, then clearly it will be enumerated into V_k^α at a large enough α-expansionary stage. Each distinct value in V_k^α corresponds to a string $(A \oplus B)\restriction_{j(k)[s]}$ (i.e. the use) in the statement of Lemma 3.8(i), with $t_0 =$ least stage after which α is never initialized. Hence, it follows that $|V_k^\alpha| \leq h_e(k)$.

Now suppose that α is a $\mathcal{P}_e$-node. Let s_0 be the least α-stage after which α is never initialized, and $q = setter(\alpha, s_0)$ be the final value.

Lemma 3.10. *There are only finitely many self-imposed initializations to α.*

Proof. Note that once α finishes setting its parameters after s_0, then (A0) and (A1) will never apply to α again, so the lemma is clearly true. So, we may assume for the sake of argument, that α never finishes setting its parameters, hence never enumerates anything (into B). We argue that $r(\alpha, q)$ eventually settles, for a contradiction. We first claim that for each $\beta \in Z^-(\alpha)$, and each $p > l(\beta, s_0^+)$ such that s_0^+ is a stage after s_0 with $h_\beta(l(\beta, s_0^+)) > q+1$, we have $size_p^\beta > q$ whenever it is defined: when $size_p^\beta$ first gets defined, it is certainly larger than q. If the V_p^β box is to be promoted after stage s_0^+, it has to be due to some positive node σ. Clearly σ cannot be on top or to the left of α, lest α is initialized. Also $\sigma \neq \alpha$ by assumption. Thus we have $\sigma \supset \alpha$ or $\sigma >_{left} \alpha$. Applying Lemma 3.9, it follows that $size_p^\beta > q$ always holds. Hence only finitely many boxes contribute to $r(\alpha, q)$, and by Lemma 3.8(ii) this means that $r(\alpha, q)$ is bounded, a contradiction. □

Hence, α will eventually finish setting its parameters, with the final value for $n = \lim region(\alpha)$. By Lemma 3.9 boxes can be promoted to small sizes only by α itself. We now argue that amongst the different threshold values $C_{n,-1}, C_{n,0}, \cdots$, box sizes are also kept disjoint. That is, once $C_{n,r}[t_0]$ has been defined, then no box of a current size at least $C_{n,r}[t_0]$ can be promoted to size $C_{n,r-1}[t_0]$.

Lemma 3.11. (i) *For each $0 \leq r \leq n-1$, $\beta \in Z^-(\alpha)$, each stage t_0 such that $C_{n,r}[t_0] \downarrow$, and k such that $size_k^\beta[t_0] \downarrow \geq C_{n,r}$, we have $\forall t(t \geq t_0 \Rightarrow size_k^\beta[t] > C_{n,r-1})$.*

(ii) *The total number of times which α can be reset after it sets its parameters, is bounded by I_n.*

Proof. (i): This follows Lemma 3.7 of [12] closely. We proceed by induction on r. Suppose the results hold for all $r' < r$. Suppose the statement fails for some $\beta \in Z^-(\alpha)$, t_0 and k. Let $s > t_0$ such that $size_k^\beta[s] \leq C_{n,r-1}$, and we may as well assume that $size_k^\beta[t_0] = C_{n,r}$. Since $C_{n,r}[t_0] \downarrow$, it follows there is some α-stage $\bar{t_0} \leq t_0$, such that $C_{n,r}$ receives its definition.

Suppose $t \geq t_0$ is a stage where some σ promotes V_k^β. It is clear from Lemma 3.9 that σ has to be α. We want to count the number of such stages t where α promotes V_k^β. At stage t, promotion happens because α needs to enumerate into A or B. The latter case only happens once after stage s_0. In order for promotion to take place at stage t due to enumeration into A, we must have $C_{n,r} \geq size_k^\beta[t] > threshold(\alpha, t) = C_{n,n-2-attempt(\alpha,t)}$, which means that $attempt(\alpha, t) > n-2-r$. We split the counting into the possible cases z for $n-1-r \leq z \leq n-1$:

Case z: t is a stage where α promotes V_k^β, and $attempt(\alpha, t) = z$.

If $z = n-1$, then $attempt(\alpha)$ will be increased to n and α never enumerates again. So, suppose that $z < n-1$ (and hence $r > 0$). In order for Case z to apply again, α has to be reset at some (least) α-stage $t' > t$, where $attempt(\alpha, t') \geq z+1 > n-1-r$. This must be due to some small box size blocking α. That is for some $\beta' \in Z^-(\alpha)$ and k', we have $J^{A\oplus B}(k')[t'] \downarrow$ and $size_{k'}^{\beta'}[t'] \leq threshold(\alpha, t') \leq C_{n,r-2}$.

We claim that $size_{k'}^{\beta'}[\bar{t_0}] \downarrow$. Suppose not. Since $r > 0$, it follows that $h_{\beta'}(l(\beta', \bar{t_0})) > C_{n,r-1}$ being a β'-expansionary stage, and therefore $h_{\beta'}(k') > C_{n,r-1}$. Applying induction hypothesis (on $r-1$) gives us a contradiction. The above not only shows that at stage $\bar{t_0}$, $size_{k'}^{\beta'}[\bar{t_0}]$ must be defined, but in fact that $size_{k'}^{\beta'}[\bar{t_0}] < C_{n,r-1}$. Applying Lemma 3.8(ii), there can be at most $1 + S(\alpha, C_{n,r-1})[\bar{t_0}]$ many stages t where Case z applies (by associating each t with the string $(A \oplus B)\restriction_{j(k')}[t']$).

Totalling the effects from all the different cases, we see that the smallest value $size_k^\beta$ can take, is $C_{n,r} - 1 - 1 > C_{n,r-1}$, if $r = 0$. On the other hand if $r > 0$, the smallest value $size_k^\beta$ can be reduced to, is $C_{n,r} - 1 - n(1 + S(\alpha, C_{n,r-1})[\bar{t_0}]) - 1 = C_{n,r-1} + 1 > C_{n,r-1}$.

(ii): this follows by a similar counting argument as (i). □

By Lemma 3.11(ii), α never runs out of indices after it is done setting parameters, so let $x = \lim x(\alpha)$. After a large enough amount of time has passed, the only reason why α requires attention will be under (A2) or (A4). When $attempt(\alpha)$ reaches n, α will no longer require attention, so (I2) is true. We now show (I1) holds. Clearly $\Psi^B(x) \downarrow = n$. Suppose that $J^A(x) \downarrow = w$ and $w \in T_x^e$. Let $s_1 > s_0$ be large enough so that $w \in T_x^e[s_1]$, and that $\Delta_\alpha^A(x)[s_1] \downarrow = w$. Observe that it must be the case that $attempt(\alpha, s_1) = n$, otherwise we would destroy the correct axiom in Δ_α^A at the next α-stage. Hence there are n many different stages $t \leq s_1$ and $attempt(\alpha, t)$

is increased by 1. After each such stage t we also enumerate a new value for $\Delta^A_\alpha(x)$, and wait for it to be traced. Thus we have $|T^e_x| \geq n$, since each time α sees a new value appearing in T^e_x, it initializes all nodes of lower priority. So, (I1) is true. This completes the induction.

The last thing we have to do, is to argue that Ψ^B is a B-order, in the sense that it is total, non-decreasing and unbounded. This follows from the fact that for every positive node α on the true path, we have $\Psi^B(\lim x(\alpha)) \downarrow = \lim region(\alpha)$. This ends the proof of Theorem 3.5.

4. A cuppable hyper jump traceable c.e. set

Theorem 3.5 raises a fundamental issue. Can there possibly be (non-computable) sets A which resemble the computable sets so strongly, such that given *any* set X not computing A, we are able to make A strongly jump traceable by X? As we have seen, it is generally *harder* to jump trace a set A by a non-computable oracle X, compared to simply jump tracing A with no oracle, because we have to deal with more (slow-growing) X-orders.

Theorem 3.5 shows that such behaviour has to be at least distinct from strong jump traceability. To what extremes can this concept be taken? Is it reasonable to require that A is strongly jump traceable by *all* sets?

Definition 4.1. We say that a c.e. set A is *hyper jump traceable*, if A is strongly jump traceable by every c.e. set W.

That is, $\mathcal{H}$ is the class of all hyper jump traceable c.e. sets. This defines a class with very strong similarities to $\emptyset$: not only can we approximate J^A with respect to all computable order functions, but also for all c.e. sets W, we can W-approximate J^A with respect to all W-computable order functions. It is clear that every hyper jump traceable c.e. set is strongly jump traceable, and the class $\mathcal{H}$ is closed downwards.

The rest of this paper will be devoted to studying $\mathcal{H}$. In Theorem 4.2, we first give a sketch of the construction of a non-computable c.e. set which is hyper jump traceable. Hence $\mathcal{H}$ is a new subclass of the strongly jump traceable and K-trivial sets. We then construct a hyper jump traceable set which is cuppable. Thus, whilst the c.e. hyper jump traceable sets resemble $\emptyset$ very closely, there is at least a fundamental property which separates them from the computable sets. In Theorem 5.1 we show that no construction of a c.e hyper jump traceable set is compatible with making it low cuppable. In Section 6, we show that no c.e. set A can be strongly jump traceable relative to *all* Δ^0_2 sets, apart from the computable sets. In Section 7 we show that there is a single capping companion for the entire class $\mathcal{H}$.

Theorem 4.2. *There is a c.e. hyper jump traceable set $A >_T \emptyset$.*

Sketch of the construction. There are two categories of requirements to be considered here. We have the positive requirements making A non-computable, as well as the negative requirements of the form

$$\mathcal{N}: \text{ If } h^W \text{ is an order, build a trace } \{T^W_x\}_{x\in\mathbb{N}} \text{ for } J^A \text{ respecting } h^W.$$

The standard strategy for building a non-computable strongly jump traceable set (without considering oracles) is the following. Suppose the opponent shows us $h^\emptyset(x)[s] \downarrow = 2$ for some x. We could then safely trace $J^A(x)[s]$ into T_x (the trace we build), and then impose the A-restraint $j(x,s)$. This is alright because in future, we can still allow one positive requirement to act below the restraint $j(x,s)$, so this restraint is not absolute. The main trouble now is that h^W might not be computable; the opponent has much more freedom now in the sense that he could show us an (incorrect) segment $h^W\restriction_n$, and then later on changes his mind on

that segment. In particular, we might first be shown $h^W(x)[s] \downarrow = 2$ for some x, and when we decide to trace $J^A(x)[s]$, the opponent would then tell us that in fact, $h^W(x) = 1$. In that case, the A-restraint $j(x, s)$ now becomes of the highest priority, and no positive action below it is allowed. If the opponent tricks us infinitely often in this way, we would not be able to get anything into A.

However, just like the opponent, we are also gifted with some degree of freedom to change our mind on whatever we are building. We could "wrap" our trace T_x^W axioms around the opponent's axioms for h. In particular, when the opponent first shows us $h^W(x)[s] \downarrow = 2$, we would enumerate $J^A(x)[s]$ into T_x^W with W-use $u(x)[s] =$ use of $h^W(x)[s]$. When he changes his mind and later on shows us $h^W(x)[s] \downarrow = 1$, he has to change $W\upharpoonright_{u(x)[s]}$; the same W-change would also remove the value $J^A(x)[s]$ from our trace T_x^W. This allows us to open up a gap by dropping A-restraint momentarily before tracing $J^A(x)[t]$ the next time. When the opponent next shows us $h^W\upharpoonright_{n+1} [s'] = 1^n 2$ (for some $n > x$), we would set our T_x^W-use to be the same as $u(n)[s']$. In this way, if the opponent tries to show us initial segments of h^W of the form $1^m 2$ for infinitely many m's, we would also have ensured that infinitely many gaps are open, in which positive requirements can act.

The above obstacle can be turned into a construction of a noncomputable order g, such that no c.e. set (other than the computable sets) can be jump traced respecting g using only a plain sequence $\{T_x\}$. Therefore if we consider jump traceability via non-computable orders, we will need a suitable oracle to help us build the trace. The above describes a basic module of $\mathcal{N}$. The requirement $\mathcal{N}$ now has to be split into infinitely many subrequirements, and each subrequirement will run several of these basic modules. For each y (corresponding to a basic $\mathcal{N}$-module), we need to guess whether or not there are infinitely many n's such that the opponent shows us an initial h^W-segment of the form $\sigma y^n(y+1)$ for some string σ. The infinitary outcome of a subrequirement corresponds to infinitely many A-gaps open (i.e. $\liminf$ of the restraint is 0), while the finitary outcome means that the restraint on A eventually settles down.

We can arrange these requirements on a tree of strategies, in the style of a degenerate $\emptyset'''$-priority argument. The reader is assumed to be familiar with standard tree arguments, and a good exposition on this topic can be found in [18]. Deciphering the outcome of an $\mathcal{N}$-requirement requires a $\emptyset'''$-oracle, because "h^W is an order" is a Π^0_3 fact, and our $\mathcal{N}$-strategies depend crucially on this fact. The full construction is given in Theorem 4.3.

On a side note we remark that this strategy will meet with fatal problems if W is a Δ^0_2 set. This is because the opponent could challenge us as above, and wait for us to enumerate $J^A(x)[s]$ into T_x^W at stage s. Suppose the next thing he does is to change W, hence emptying T_x^W for us, and we would follow up by opening up an A-gap below $j(x, s)$, according to the plan. At the next stage the opponent could *recover* W back to the configuration at stage s, i.e. restore W back to the state before the A-gap was opened. We would be in trouble, because T_x^W now contains the $J^A(x)[s]$ (having been brought back by the opponent), and the opponent could now make $J^A(x) \neq J^A(x)[s]$. Since A is c.e. there is no way for us to get the correct $J^A(x)$ into our trace, unless we increase the size of T_x^W illegally. In Section 6, we show that this obstacle cannot be overcome - no c.e. set is strongly jump traceable by every Δ^0_2 set. It is not known however, if we can build an $A \leq_T \emptyset'$, such that A is strongly jump traceable by every Δ^0_2 set. □

Theorem 4.3. *There is a c.e hyper jump traceable set A, and an incomplete c.e. set C, such that $\emptyset' \leq_T A \oplus C$.*

4.1. **Requirements.** We code $\emptyset'$ into $A \oplus C$ by building the reduction $\emptyset' = \Gamma^{A\oplus C}$, and act for it globally, i.e. outside of the construction tree. There are two types of requirements with conflicting interests, namely the ones making A hyper jump traceable:

$$\mathcal{N}_e \quad : \quad \text{If } h_e^{W_e} \text{ is an order, then build a trace } \{T_x^{W_e}\}_{x\in\mathbb{N}} \text{ for } J^A \text{ respecting } h_e^{W_e},$$

and the ones making C incomplete, by building an auxiliary c.e. set D:

$$\mathcal{R}_e \quad : \quad \Phi_e^C \neq D.$$

Here, $\langle W_e, \Phi_e, h_e\rangle_{e\in\mathbb{N}}$ is an effective list of all triples such that W_e is a c.e. set, and Φ_e, h_e are Turing functionals. Also let $J^A(x) = \{x\}^A(x)$. The stage s use of the computations $J^A(x), \Phi_e^C(x)$ and $h_e^{W_e}(x)$ are denoted by $j(x,s), \varphi_e(x,s)$ and $u_e(x,s)$ respectively.

Since we are concerned only with the functions $h_e^{W_e}$ that are total, it does no harm for us to assume that for $x < y$, we have $u_e(x,s) < u_e(y,s)$ whenever they are both defined. 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. We will also drop the stage number from the notations if the context is clear. All parameters will retain their assigned values until initialized or reassigned. We append $[s]$ to an expression to mean the value of the expression as evaluated at stage s.

At each requirement $\mathcal{N}_e$, we will build a u.c.e. trace $\{T_x^{W_e}\}_{x\in\mathbb{N}}$ such that for all $x \in \mathbb{N}$, we have (if $h_e^{W_e}$ is an order)

(i) $|T_x^{W_e}| \leq h_e^{W_e}(x)$, and
(ii) $J^A(x)\downarrow \Rightarrow J^A(x) \in T_x^{W_e}$.

As discussed in Theorem 4.2, we divide the requirement $\mathcal{N}_e$ into the subrequirements $\mathcal{N}_{e,0}, \mathcal{N}_{e,1}, \cdots$, where $\mathcal{N}_{e,i}$ is responsible for tracing $J^A(x)$ for all x such that $m_e^{i+1} < h_e^{W_e}(x) \leq m_e^{i+2}$, where $m_e^i = 2^{2\langle e,i\rangle+2}$. This number is chosen based on technical reasons, which will become clear later.

4.2. **Description of strategy.** In this construction we replace the non-computable requirements with the (stronger) cupping requirements. This consists of two parts - the coding of $\emptyset'$ into $A \oplus C$, as well as the incompleteness strategies $\mathcal{R}$. The coding of the Turing complete set is done outside the tree, and is not affected by the movement of the accessible string δ_s (the accessible string determines which nodes get to act during stage s of the construction). This involves maintaining a set of markers $\gamma(0) < \gamma(1) < \cdots$, where the current values of $\gamma(n), \gamma(n+1), \cdots$ are dumped into the set C whenever some n enters $\emptyset'$. New values will be chosen for these markers, and we say that these markers are *moved*. More details will be given in Section 4.4; at this stage the reader will only need to note that the coding strategy can only move markers by dumping them into C.

The second part to cupping involves building a set D not computable in C. The atomic strategy for an $\mathcal{R}$-requirement is the standard Friedberg strategy: it picks a number z not yet in D, and waits for $\Phi^C(z)[s]\downarrow = 0$. When that happens, we enumerate z into D and preserve C; at this point we say that we *believe in the* Φ^C*-computation*. In order for the strategy to succeed, we have to ensure that $\mathcal{R}$ believes in only finitely many wrong Φ^C-computations. To limit the number of times a believed computation may be injured and hence become incorrect, $\mathcal{R}$ will first pick a number b, and then only believe in a Φ^C-computation if $\varphi(z,s) < \gamma(b,s)$; to do this it may have to enumerate $\gamma(b,s)$ into A to move the marker $\gamma(b)$ above $\varphi(z,s)$.

Let us consider an $\mathcal{R}$-strategy below a subrequirement $\mathcal{N}_{e,0}$ of $\mathcal{N}_e$. Suppose that $\mathcal{N}_{e,0}$ is assigned to the node α, and is responsible for tracing $J^A(x)$ for all the x such that $h_e^{W_e}(x) = 1$. The locations in $\{T_x^{W_e}\}$ which traces these values are called 1-*boxes*, and once these 1-boxes are filled with a value, the restraint $\mathcal{N}_{e,0}$ imposes for it has to be respected everywhere. At all times $\mathcal{N}_{e,0}$ is in charge of a certain number of 1-boxes; however since $h_e^{W_e}$ may not be computable, the number of 1-boxes it is responsible for may increase with time. Recall that $\mathcal{N}_{e,0}$ will have two outcomes, the infinitary ∞ outcome which opens a gap each time $\mathcal{N}_{e,0}$ is entrusted with more 1-boxes to trace, and the finitary outcome f which $\mathcal{N}_{e,0}$ will take if the situation in $h_e^{W_e}$ is stable.

There will be two corresponding versions of the $\mathcal{R}$ strategy assigned to the nodes $\beta_0 = \alpha^\frown\infty$ and $\beta_1 = \alpha^\frown f$ below α. These two use a different set of parameters, i.e. b_{β_0}, z_{β_0} for β_0 and b_{β_1}, z_{β_1} for β_1. How may β_1 be affected by α? Each time α increases A-restraint due to activity in one of its 1-boxes (remember these boxes are of the highest priority), we would have to move the marker $\gamma(b_{\beta_1})$ above the A-restraint, by dumping into C. However β_1 might encounter infinitely many 1-box restraints, but if that is the case then β_0 will be visited infinitely often, and hence be the one to meet $\mathcal{R}$. To ensure that $\gamma(b_{\beta_1})$ settles, we have to pick a new value for b_{β_1} each time α opens a gap and is shown with even more 1-boxes to trace. Hence, β_1 (or β_0, depending on the true outcome of α) will eventually settle on some final value for b_{β_1}, and receive a finite amount of restraint from α above (in the case of β_0 there is no restraint from above). We want to see that it succeeds in meeting $\mathcal{R}$ with some z.

It is possible for β_1 to believe in a false computation, if some marker $\gamma(k)$ for $k < b_{\beta_1}$ is dumped into C after we believe. Thus, it is possible for β_1 to make more than one enumeration into A. Each A-enumeration that β_1 makes has to respect the restraint from 1-boxes above it (i.e. from α), but ignores the restraints imposed by boxes below it. In particular, $\mathcal{N}_{e,1}$ might occupy a level below β_1, and be in charge of a number of 2-boxes, which all get *promoted*[7] to 1-boxes due to the positive actions of β_1. When these locations next get traced, *their restraints have to be obeyed even by* β_1 *above*, because $\mathcal{N}_{e,1}$ while of a lower local priority, actually has a higher global priority than β_1. Therefore, the blockages on A that β_1 has to consider become slightly more complicated; in particular β_1 has to search through the entire sequence $\{T_x^{W_e}\}$ to see which boxes have been promoted and must now be obeyed.

This brings up another problem. Notice that there might be many levels below β_1, which are assigned some other incompleteness requirement $\mathcal{R}'$, say at some $\sigma \supset \beta_1$. When σ acts, it may in turn promote $\{T_x^{W_e}\}$-boxes living *below* it, causing these boxes to become 1-boxes. Hence, infinitely many 1-boxes may be promoted in this way down the tree; β_1 now has to deal with all these boxes, which is bad because β_1 is on the true path and there is no other backup strategy for $\mathcal{R}$.

Before we proceed further, let us pause for a moment and think why this does not happen in Theorem 4.2. The positive requirements were simple - each level on the tree assigned a positive requirement only needed to make at most one enumeration into A, for the sake of making A non-computable. Therefore, we could use this fact to help us assign the $\mathcal{N}_{e,i}$-strategies on the tree. For instance, we could do the following assignment of strategies in Theorem 4.2:

[7]That is, a location $T_y^{W_e}$ with $h_e^{W_e}(y) = 2$ is now filled with a single wrong value, reducing the size of the tracing location.

level of the construction tree	strategy assigned
0	handle 1-boxes
1	non-computable strategy
2	handle 2-boxes
3	non-computable strategy
4	handle 3-boxes

For instance, the 3-boxes can be handled safely at level 4 because each level (levels 1 and 3) above it running the positive strategy can make at most one enumeration each. Hence the 3-boxes are at most promoted to a 1-box, after which levels 1 and 3 no longer need to act.

We can use this idea to help us out in this current construction. Even though the nodes assigned the incompleteness strategies enumerate more than once into A, we can make them behave just like a non-computable strategy relative to each $\mathcal{N}_{e,i}$. To illustrate this, we arrange the strategies in this manner:

level of the construction tree	strategy assigned
0	handle 1-boxes
1	incompleteness strategy
2	handle 2-boxes
3	incompleteness strategy
4	handle 3-boxes

If the incompleteness strategy at level 3 enumerates into A, promoting boxes at level 4 in its current run, we make sure that at its next run when it has to enumerate into A again, it will not promote any boxes at level 4 which it has promoted before. Hence levels 1 and 3 will look like they are running a non-computable strategy when viewed by level 4, even though they are making many enumerations into A. Thus, level 3 can promote boxes at levels $4, 6, 8$,etc but this has no effect on level 1.

To help us implement this idea, for each node σ assigned to an incompleteness strategy, we keep track of the *injury number* of σ. This represents the largest number x such that σ has previously injured a box location $T_x^{W_e}$, and helps σ remember which boxes not to promote in its next run. Note that the injury number does not need to increase until σ makes the next enumeration into A, hence lifting $\gamma(b)$ above the φ-use, and believing the Φ^C-computation and ending its current run. Therefore, in its current run, there is only finitely much restraint it has to obey from below (due to those boxes it has previously promoted), before it is next allowed to believe in a Φ^C-computation. Note also that in this construction, we are exploiting the fact that C need not be low (in fact, cannot be low): we dump markers into C until all blockages on A have been overcome.

4.3. **Construction tree layout.** The construction takes place on a subtree of the full binary tree. Nodes of even length $|\alpha| = 2e$ are assigned the requirement $\mathcal{R}_e$ with the single outcome 0. Nodes of length $|\alpha| = 2\langle e, i\rangle + 1$ are assigned the requirement $\mathcal{N}_e$ if $i = 0$, and the subrequirement $\mathcal{N}_{e,i-1}$ if $i > 0$. Nodes assigned the requirement $\mathcal{N}_e$ have only one outcome 0. The subrequirement $\mathcal{N}_{e,i}$ of $\mathcal{N}_e$ has the two outcomes $\infty <_{left} f$. The infinitary outcome represent phases when $\mathcal{N}_{e,i}$ is imposing no restraint (during A-gaps), while the finitary outcome means that $\mathcal{N}_{e,i}$ is holding some A-restraint.

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 $\alpha(i) <_{left} \beta(i)$). We say that α *is a* $\mathcal{Q}$*-node*, if α is assigned the requirement $\mathcal{Q}$. α is an *incompleteness node*, if α is an $\mathcal{R}_e$-node for some e. We say that α is a *top node*, if α is an $\mathcal{N}_e$-node for some e, and that α is an i*-bottom node of* τ if α is an $\mathcal{N}_{e,i}$-node for some e, and $\tau \subset \alpha$ where τ

is an $\mathcal{N}_e$-node. If α is a bottom node, then $\tau(\alpha)$ denotes its top node. α and β are *sibling nodes* if for some i, they are both i-bottom nodes of the same top. If τ is an $\mathcal{N}_e$-node, denote $Cone_i^\tau := \{\beta : \beta \supset \tau \ \wedge \ |\beta| \leq 2\langle e, i+1\rangle\}$, that is, all the nodes lying strictly in between τ and its i-bottom nodes.

Each top $\mathcal{N}_e$-node τ does not really make any guesses as to whether or not $h_e^{W_e}$ is an order. Rather, this is done at each layer below τ. We distribute the A-restraints amongst its bottom nodes, and τ's role is to coordinate the actions of its bottom nodes.

4.4. **Notations.** The functional Γ reducing $\emptyset'$ from $A \oplus C$ will be build globally. If a $\Gamma^{A\oplus C}(x)$ axiom is set at stage s, it will have use denoted by $2\gamma(x,s)$, so that the first $\gamma(x,s)$ many bits of both A and C are involved. We ensure that at all times, if $x < y$ and both axioms are set, then $\gamma(x,s) < \gamma(y,s)$. To ensure this, we always enumerate markers by dumping. That is, each time during the construction when we say that we enumerate $\gamma(x,s)$ into A or C, we mean to say that we enumerate $\gamma(y,s)$ into A or C, for every $y \geq x$ for which the Γ-axioms have been set. We fix an enumeration $\{\emptyset'_s\}_{s\in\mathbb{N}}$ of the halting problem, in which at most one element is enumerated at each stage.

If σ is an $\mathcal{R}_e$-node, it will pick a number z_σ and attempt to make $\Phi_e^C(z_\sigma) \neq D(z_\sigma)$. It does this by picking a number b_σ, and then it will enumerate z_σ into D at a stage s only after it succeeds in ensuring that $\gamma(b_\sigma, s) > \varphi_e(z_\sigma, s)$. At each top $\mathcal{N}_e$-node τ, we measure the length of convergence of $h_e^{W_e}$ at stage s by

$$l_\tau(s) = \max\{y < s \mid (\forall x \leq y) \ \ h_e^{W_e}(x)[s] \downarrow \geq h_e^{W_e}(x-1)[s] \\ \wedge \ h_e^{W_e}(y)[s] > h_e^{W_e}(y-1)[s]\}.$$

We will build a uniformly c.e. sequence of sets $\{T_x^\tau\}_{x\in\mathbb{N}}$ at τ. For simplicity we drop the oracle W_e from the notation. At times a bottom node α of τ will enumerate a number r into T_x^τ for some x, with a W_e-use denoted by $t_x^\tau(s)$. In future if W_e doesn't change below this use t_x^τ, then the number r stays in T_x^τ. On the other hand if for some $t > s$, we have $W_{e,t}\restriction_{t_x^\tau(s)} \neq W_{e,s}\restriction_{t_x^\tau(s)}$, then the number r is removed from T_x^τ, i.e. $r \notin T_x^\tau[t]$.

We let r_i^τ record the restraint imposed on A by τ, for the sake of its i-bottom nodes. That is, a single restraint function is used for each entire level. For each incompleteness node $\sigma \supset \tau$, we let I_σ^τ denote the *injury number* that σ has to respect when considering which side (A or C) to enumerate into. That is, σ cannot make any enumeration into $A\restriction_{j(i)}$ for any $i < I_\sigma^\tau$, for the sake of limiting injuries to T_i^τ. If α is an i-bottom node of τ, define the parameter

$$marker_\alpha(s) = \text{least } y < l_\tau(s) \text{ such that } h_e^{W_e}(y)[s] \geq m_e^{i+2} + 1.$$

This marks out the first place y such that $h_e^{W_e}(y)[s] \geq m_e^{i+2} + 1$, and $h_e^{W_e}\restriction_{y+1}[s]$ looks like an order. If no such y exists, let $marker_\alpha(s) \uparrow$. The purpose of this definition is to let α wrap its trace axioms around the value $u_e(marker_\alpha)[s]$. If β is a sibling node of α, then for all s, $marker_\alpha(s) = marker_\beta(s)$. We also write $marker_{e,i}(s)$ in place of $marker_\alpha(s)$, and let $marker_{e,-1} = 0$. If W_e has settled on the part which is accessed by α, we would be able to run the strategy at level $|\alpha|$ without interruption. Such a stage s is said to be *α-fixed*, or *(e,i)-fixed*. That is, we have $marker_{e,i}[s] \downarrow$, and $W_{e,s}\restriction_u = W_e\restriction_u$, where $u = u_e(marker_{e,i})[s]$.

When we *initialize* an incompleteness node σ, we set $z_\sigma, b_\sigma \uparrow$. When we initialize a top node τ, we clear the sequence $\{T_x^\tau\}_{x\in\mathbb{N}}$ of all axioms, and start building it again from scratch. Set $r_i^\tau = 0$ for all i, and set $I_\sigma^\tau = 0$ for each incompleteness node $\sigma \supset \tau$. During the construction, when we *increase* (or *decrease*) a parameter value P to x, we mean that we set $P = x$, unless P is already larger (or smaller) than x, in which case we do nothing.

4.5. **The construction.** At stage $s = 0$, initialize all nodes, and do nothing. Let $s > 0$. We build the stage s approximation to the true path, δ_s of length s inductively. We say that α is visited at stage s, if $\delta_s \succ \alpha$; equivalently we say that s is an α-stage. Assume that $\alpha = \delta_s{\restriction}_d$ has been defined for $d < s$. There are three possibilities for α.

(1) *α is an $\mathcal{R}_e$-node* : let $\delta_s(d) = 0$. Pick the first case from the list that applies, and act for α accordingly:
   (a) if b_α and z_α are undefined, pick fresh values for them,
   (b) if $D(z_\alpha) \neq \Phi_e^C(z_\alpha)[s]$, do nothing,
   (c) if $z_\alpha \in D$, pick a fresh value and reassign z_α,
   (d) all the above fails, i.e. $\Phi_e^C(z_\alpha)[s] \downarrow = 0 = D(z_\alpha)$. If $\gamma(b_\alpha, s) \downarrow$, we now have to decide which side (A or C) to enumerate $\gamma(b_\alpha, s)$ into. If
      - $\gamma(b_\alpha, s) > \max\{r_i^\tau \mid \tau \subset \beta^\frown f \subseteq \alpha$ for some top node τ, and some i-bottom node β of $\tau\}$, and
      - $\gamma(b_\alpha, s) > \max\{j(x, s) \mid x < I_\alpha^\tau$ for some top node $\tau \subset \alpha\}$,

   both holds, then we enumerate z_α into D, and for each top node $\tau \subset \alpha$, we increase the injury number I_α^τ to be $1+$ the largest n such that $J^A(n)[s] \downarrow \in T_n^\tau$, with use $j(n, s) > \gamma(b_\alpha, s)$ (if no such n exists, do nothing). Enumerate $\gamma(b_\alpha, s)$ into A. Otherwise if one of the above does not hold, i.e. $\gamma(b_\alpha, s)$ is blocked on the A side, then we enumerate $\gamma(b_\alpha, s)$ into C, and do nothing else.

(2) *α is an $\mathcal{N}_e$-node* : let $\delta_s(d) = 0$, and s^- be the previous α-stage (if this is the first α-stage, do nothing). Check if there is a least i such that $marker_{e,i}(s^-) \downarrow$, and $W_{e,s^-}{\restriction}_{u_e(marker_{e,i})[s^-]} \neq W_{e,s}{\restriction}_{u_e(marker_{e,i})[s^-]}$. If no such i exists, do nothing. Otherwise consider the least such i - we know that all work previously done at level $\mathcal{N}_{e,i}$ and below have now been undone. So, we can decrease I_σ^α to $marker_{e,i-1}$ for every incompleteness node $\sigma \supset \alpha$.

(3) *α is an $\mathcal{N}_{e,i}$-node* : let s^- be the previous α-stage. Look at :
   (a) s^- exists, i.e. we have visited α before.
   (b) $marker_\alpha(s^-)$ and $marker_\alpha(s)$ are both defined and equal.
   (c) $W_{e,s^-}{\restriction}_{u_e(marker_\alpha)[s^-]} = W_{e,s}{\restriction}_{u_e(marker_\alpha)[s^-]}$.

   If one of the above 3(a), 3(b) or 3(c) fails, then $h_e^{W_e}$ has not yet stabilized on the part which is required for α; we have not reached an α-fixed stage. In this case drop A-restraint by letting $\delta_s(d) = \infty$, and do nothing else.

   Otherwise if all of 3(a)-(c) holds, then let $\delta_s(d) = f$, and do the following. For each x such that $J^A(x)[s] \downarrow$ and $marker_{e,i-1}(s) \leq x < marker_\alpha(s)$, we enumerate the value $J^A(x)[s]$ into $T_x^{\tau(\alpha)}$ (unless the value is already in there), with W_e-use $t_x^{\tau(\alpha)}(s) = u_e(marker_\alpha)[s]$. Increase the restraint $r_i^{\tau(\alpha)}$ to $j(x, s)$.

This ends the definition of δ_s. At the end of stage s, we initialize all nodes $\beta >_{left} \delta_s$. We take actions for coding. If there is some $x \in \emptyset'_s - \emptyset'_{s-1}$, such that $\Gamma(x)[s]$-axioms currently apply, we enumerate $\gamma(x, s)$ into C. Otherwise, pick the least x for which no $\Gamma(x)[s]$-axioms apply. Set $\Gamma^{A \oplus C}(x)[s] \downarrow = \emptyset'_s(x)$ with fresh use $2\gamma(x, s)$. Go to the next stage.

4.6. **Verification.** The true path of the construction is defined as usual to be the leftmost path visited infinitely often. If τ is a top node and σ is an incompleteness node and $x \in \mathbb{N}$, we say that σ *injures* T_x^τ at stage s, if at stage s, σ enumerates into A below the use of a convergent computation $J^A(x)[s] \downarrow = n$, where $n \in T_x^\tau$.

We first show that if τ is on the true path, then each location T^τ_x is injured from above[8] only finitely often:

Lemma 4.4. *Let τ be an $\mathcal{N}_e$-node on the true path, and $i \in \mathbb{N}$. Suppose s is an (e,i)-fixed stage such that τ is never initialized after stage s. Then for each $x < marker_{e,i}$, and each $\sigma \in Cone^\tau_i$, σ is allowed to injure T^τ_x at most once after stage s.*

Proof. If σ injures T^τ_x at some stage $t \geq s$, the same action also sets $I^\tau_\sigma > x$. The only way for σ to injure T^τ_x again, is for I^τ_σ to drop to $\leq x$. Since τ is never initialized this must be due to τ decreasing the injury number T^τ_σ under step 2 of the construction. Since t is a τ-stage, this can only happen if $W_{e,t}\restriction_u \neq W_e\restriction_u$, which is impossible. □

Lemma 4.5. *$\Gamma^{A\oplus C}$ is total and equals $\emptyset'$. In particular for each x, $\gamma(x,s)$ eventually settles.*

Proof. We prove the above by induction on x. Fix an x, and let s_0 be a stage after which $\gamma(x-1)$ is never moved. The only interesting case to consider is when some σ on the true path has picked $b_\sigma = x$. We assume that σ is never initialized after s_0, and that no $\sigma' \subset \sigma$ enumerates after stage s_0, since $b_{\sigma'} < b_\sigma$. If it is the case that σ enumerates $\gamma(x)$ into A after s_0, then we are done. We assume that no enumeration is ever made into A by σ, and that step 1(d) of the construction applies at infinitely many σ-stages. We want to argue that the two conditions in step 1(d) are eventually satisfied, for a contradiction.

The first condition: fix a top node τ, and an i-bottom β of τ such that $\tau \subset \beta^\frown f \subseteq \sigma$. We argue that $\lim_{s\to\infty} r^\tau_i[s] < \infty$. Since β has true outcome f, hence there is a β-fixed stage, so eventually at every τ-stage t large enough, we have $\delta_t(|\beta|) = f$. Once we reach such a stage in the construction, the restraint r^τ_i can only be destroyed by some node $\sigma' \in Cone^\tau_i$. However r^τ_i only increases if $j(w,t)$ gives a new value for some w, and there are only finitely many w to consider at level $|\beta|$. By Lemma 4.4, each of these T^τ_w-boxes are injured only finitely often, so the restraint function r^τ_i is bounded.

Now we look at the second condition: fix a top $\mathcal{N}_e$-node $\tau \subset \sigma$. Since σ never enumerates into A, the injury number I^τ_σ does not increase. Hence, I^τ_σ settles down at a smallest value, say p at some stage t_0. Fix some $w < p$, and we want to show that $j(w,t)$ has only a finite effect on σ. In particular we want to show that the set $\{j(w,t) \mid t \text{ is a } \sigma\text{-stage where step 1(d) applies}\}$ is bounded. If t is a σ-stage such that $J^A(w)[t]\downarrow$ and step 1(d) applies, we would enumerate $\gamma(x,t)$ into C to move $\gamma(x)$ above $j(w,t)$. Since all smaller markers have already settled, it follows that A will never change below $j(w,t)$ after stage t.

Hence the two conditions in step 1(d) will eventually be satisfied, and σ will eventually find no reason to enumerate $\gamma(x)$ into C - it will have to enumerate $\gamma(x)$ into A instead, a contradiction. □

We next show that A is hyper jump traceable. Fix an $\mathcal{N}_e$-node τ on the true path, such that $h^{W_e}_e$ is an order. If α is a τ-bottom node on the true path, then α has true outcome f, as there will be an α-fixed stage. Let s_0 be the stage where τ is initialized for the last time, hence the true version of $\{T^\tau_x\}_{x\in\mathbb{N}}$ is built after s_0. Fix an x, and let i be such that $m^{i+1}_e < h^{W_e}_e(x) \leq m^{i+2}_e$; this can be done for almost all x, and our task is to show that $|T^\tau_x| \leq m^{i+1}_e$, and if $J^A(x)\downarrow$, then $J^A(x) \in T^\tau_x$. Let $x^+ =$ least such that $h^{W_e}_e(x^+) > m^{i+2}_e$, i.e. $x^+ = \lim_s marker_{e,i}[s]$. Let α be the i-bottom node of τ on the true path. Let s_1 be the least stage α-fixed stage $> s_0$.

[8]We think of the location T^τ_x as being built at the level of the i-bottom nodes, where $x < marker_{e,i}$.

It is not hard to see that $T^\tau_x = \emptyset$ at the beginning of stage s_1: if not then some j-bottom node of τ had made a trace at some stage t, with $s_0 < t < s_1$, with use $u_e(marker_{e,j})[t]$. If W_e changes below $u_e(marker_{e,j})[t]$ after stage t, the trace value would be removed. But if W_e did not change then we must have $j = i$ to match the situation at stage s_1, but this means that $t < s_1$ is (e, i)-fixed, a contradiction. So, $T^\tau_x = \emptyset$ at the beginning of s_1. Anything that we put in T^τ_x during or after stage s_1, will be put in with W_e-use $u_e(x^+)$, and thus will stay in T^τ_x forever. If $J^A(x)\downarrow$, then its value will clearly be placed into T^τ_x after stage s_1, so we certainly have $J^A(x) \in T^\tau_x$. Now we argue that:

Lemma 4.6. $|T^\tau_x| \leq m_e^{i+1}$.

Proof. Firstly, observe that there are at most $2^{|\alpha|}$ many τ-stages $t \geq s_1$ such that $\delta_t(|\alpha|) = \infty$, because each sibling node of α can be visited with outcome ∞ at most once, on or after the α-fixed stage s_1. Let $t \geq s_1$ be a stage where $J^A(x)[t]$ is traced into T^τ_x, by some i-bottom node of τ. The same action also increases r^τ_i beyond $j(x,t)$. Which incompleteness node σ can enumerate into $A\restriction_{j(x,t)}$ after stage t? Obviously $\sigma \not<_{left} \tau$, and every node to the right of τ has to obey r^τ_i. If $\sigma \subset \tau$ then by Lemma 4.5 it only makes finitely many enumerations into A, so we may assume x is large enough so as not to be affected by σ. This leaves the case when $\sigma \supset \tau$; hence either $\sigma \in Cone^\tau_i$, or else $|\sigma| > |\alpha|$. In the former case there can only be a total of $|Cone^\tau_i|$ many injuries to r^τ_i by Lemma 4.4, while the latter case contributes at most $2^{|\alpha|}$ many injuries since r^τ_i is obeyed at $\delta_t(|\alpha|)^\frown f$-stages. Hence there are at most $1 + 2^{|\alpha|} + |Cone^\tau_i| = m_e^{i+1}$ many different values in T^τ_x. $\square$

We show that C is Turing incomplete. Let σ be an $\mathcal{R}_e$-node on the true path, and let $b = \lim b_\sigma$. By Lemma 4.5 there is a stage s_0 after which 1(d) never applies at σ-stages. After s_0 the parameter z_σ can get reassigned at most once since 1(d) never applies, so let $z = \lim z_\sigma$. Observe that $D(z) \neq \Phi^C_e(z)$. Hence, A is cuppable, and this concludes the proof of Theorem 4.3.

5. No c.e. hyper jump traceable set can be promptly simple.

In this section, we show that no c.e. hyper jump traceable set can be promptly simple. Thus, while there can be cuppable hyper jump traceable c.e. sets, none of them can be low cuppable. The construction of a non-computable c.e. hyper jump traceable set in Section 4 was by using a $0'''$-priority argument, and does not seem to allow prompt enumeration for the positive requirements. In the following theorem we show that this must indeed be the case.

A non-computable member of $\mathcal{H}$ was constructed without using direct cost functions. $\mathcal{H}$ is therefore the first known example of a subclass of the c.e. K-trivials, which is completely free of the promptly simple sets. Other known subclasses of the c.e. K-trivials, such as the strongly jump traceable, ML-non-cuppable, ML-coverable sets, and $\mathcal{C}^\diamond$ for certain Σ^0_3-null classes $\mathcal{C}$, are all shown to be non-trivial by constructing a promptly simple member using cost functions (see Chapter 8 of [16]).

Therefore $\mathcal{H}$ forms a subclass of the cappable c.e. sets. In Theorem 7.1 we will extend this result to show that there is a single capping companion for the entire class $\mathcal{H}$.

Theorem 5.1. *Suppose A is c.e. and promptly simple. Then, there is a c.e. set C, such that $A \notin SJT(C)$.*

5.1. **Requirements.** We build the c.e. set C, and a Turing functional Ψ to meet the requirements :

$$\mathcal{R}_e \quad : \quad \text{Defeat the } e^{th} \ C\text{-trace. That is, for some } x, \text{ either } |T^e_x| \geq \Psi^C(x), \text{ or else } J^A(x) \notin T^e_x.$$

Here, we let $\{T^e_x\}_{x\in\mathbb{N}}$ be the e^{th} C-trace, in some effective listing of all traces computing with an oracle. For ease of notations we suppress all mention of C in T^e_x.

5.2. **Description of strategy.** The strategy for this theorem is based on the fact that we could force A to change "promptly", otherwise known as prompt permitting. Suppose we were only given A non-computable, and we wanted to build a c.e. set C such that $A \notin SJT(C)$. The false proof would go something like this. We want to build a C-order Ψ^C, and defeat every possible C-trace $\{T^C_x\}_{x\in\mathbb{N}}$ via Ψ^C. We take control of $J^A(x)$ for some x, and start by setting $\Psi^C(x) = 1$, and enumerate $J^A(x)[s_0] \downarrow = s_0$ with use u. Once s_0 shows up in T^C_x, we record the string $A_{s_0}\upharpoonright_u$. If A changes in future below u, we could set $J^A(x)$ different and be done. There is no guarantee that A will ever change below u, of course, so while waiting for the (potential) change to happen, we have to pick a larger $x' > x$ and repeat. Of course other requirements might have already defined $\Psi^C(y) = 2$, say, for some $x < y < x'$, so we have to enumerate $\psi(x)$ into C to kill the axiom and set $\Psi^C\upharpoonright_{[x,x']} = 1$. Eventually A has to change below one of the use u since it is non-computable, and when it does so we would be done.

Since we know that there is a non-computable hyper jump traceable set, which part of the above goes wrong? The opponent plays the non-computable A, which means that he has to change $A\upharpoonright_u$ below one of the uses we specify, but he doesn't have to do so *promptly*. In particular, he could put $J^A(x)[s_0]$ into T^C_x with a C-use larger than that of $\psi(x)$, i.e. he wraps his axioms around ours. The opponent then waits for us to pick $x' > x$ and for us to put $\psi(x)$ into C. We needed to do that to ensure that Ψ^C is an order, but we have also unwittingly helped the opponent to clear T^C_x temporarily. The opponent could now seize the opportunity and change $A\upharpoonright_u$, and we would be left stranded.

However, if the opponent was put in charge of a promptly simple set A instead, we could force him to change $A\upharpoonright_u$ promptly. Only when the opponent fails to respond promptly, do we choose a new $x' > x$, and put $\psi(x)$ into C to clear our Ψ^C axioms (as well as the T^C_x axioms). This is alright because eventually the opponent has to respond with an A-change *before* we clear the C-axioms.

5.3. **The construction.** We arrange our requirements in the order $\mathcal{R}_0 \prec \mathcal{R}_1 \prec \cdots$. At each requirement $\mathcal{R}_e$, we will enumerate auxiliary c.e. sets $U_{e,0}, U_{e,1}, \cdots$ to try and force A to change. By a slowdown lemma and the Recursion Theorem, we may assume that if we put numbers $x_0 < x_1 < \cdots$ into an auxiliary set U at stages $s_0 < s_1 < \cdots$ respectively, then A has to promptly permit one of them; namely there is some i such that $A_{s_i}\upharpoonright_{x_i} \neq A_{s_{i+1}}\upharpoonright_{x_i}$. We may in fact assume that A has to promptly permit infinitely many of the x_i's.

By the Recursion Theorem again, we have an infinite list of indices of functionals we will enumerate during the construction. Let $x(e,s)$ denote the index that $\mathcal{R}_e$ is using at stage s, with use $u(e,s)$. At times $\mathcal{R}_e$ will be initialized; in that case it abandons this index and use and picks a fresh one from the list (not used before). We will let $region(e,s)$ record the number n, such that $\mathcal{R}_e$ wants $\Psi^C(x(e)) = n$. This also represents the number of elements $\mathcal{R}_e$ wants to force into $T^e_{x(e)}$, before it is satisfied. We let $attempt(e,s)$ keep track of the number of elements $\mathcal{R}_e$ has succeeded in forcing into $T^e_{x(e)}$; when $attempt(e) = region(e)$, its work is done. During

the construction we will also enumerate axioms into Ψ^C, maintained globally; again we denote the stage s use of $\Psi^C(x)[s]$ by $\psi(x,s)$.

At stage s of the construction, we pick the least $e < s$ such that $\mathcal{R}_e$ requires attention, i.e. one of the following applies:

(A1) $\mathit{region}(e,s)$ is currently unassigned.
(A2) $\mathit{region}(e,s)\downarrow$, but no axioms in $J^A(x(e,s))[s]$ currently apply.
(A3) $J^A(x(e,s))[s]\downarrow = r$, and $r \in T^e_{x(e,s)}[s]$ and $\mathit{attempt}(e,s) < \mathit{region}(e,s)$.

There are three cases:

- If (A1) applies, we pick a fresh number for $\mathit{region}(e)$, pick a fresh unused index $x(e)$, and set $u(e) = s$ and $\mathit{attempt}(e) = 0$. Set $\Psi^C(x(e,s))[s]\downarrow = \mathit{region}(e,s)$ with a fresh use $\psi(x(e,s),s)$. In fact, to ensure the totality of Ψ^C, we also enumerate axioms with the same C-use for all $y < x(e,s)$ where no axioms currently apply for y. Go to the next stage.
- If (A1) fails but (A2) applies, then we set $J^A(x(e,s))\downarrow = s$ with use $u(e,s)$. Go to the next stage.
- If (A1) and (A2) fails but (A3) holds then we enumerate $u(e,s)$ into $U_{e,\mathit{attempt}(e,s)}$, and set $\mathit{region}(e')\uparrow$ for all $e' > e$.
  Check if A promptly permits $u(e,s)$. If yes, increment $\mathit{attempt}(e)$ by 1, and move on to stage $s+2$. If not, we enumerate $\psi(x(e),s)$ into C to clear the definition of $\Psi^C(x')$ for all $x' \geq x(e)$, and choose a fresh unused index $x(e)$ and set $u(e) = s$ and $\mathit{attempt}(e) = 0$. Also, for all the relevant $x' \leq$ the new $x(e)$, set $\Psi^C(x')[s]\downarrow = \mathit{region}(e,s)$ with fresh $\psi(x',s)$ use. Go to stage $s+2$.

At the start of the construction all *region* parameters are unassigned, so this guarantees that $\mathcal{R}_{s-1}$ requires attention at stage s.

5.4. **Verification.**

Lemma 5.2. *For each e, $\mathit{region}(e,s)$ eventually settles.*

Proof. Suppose that $r = \mathit{region}(e-1,s_0)$ has settled. The only way for $\mathit{region}(e)$ to be reset after stage s_0, is when $\mathcal{R}_{e-1}$ receives attention, and $\neg$(A1)$\wedge\neg$(A2)$\wedge$(A3) holds for it. Suppose this happens at infinitely many stages after stage s_0. At each such stage, a number will be enumerated into one of $U_{e-1,0},\cdots,U_{e-1,r-1}$, and has to be new to the set. Choose the largest $r' < r$ such that $U_{e-1,r'}$ is infinite. Since A promptly permits infinitely many of the numbers in $U_{e-1,r'}$, eventually A promptly permits some number $u(e-1,s')$ enumerated into $U_{e-1,r'}$ at stage s', where the construction will make no further enumeration into any of $U_{e-1,r'+1},\cdots,U_{e-1,r-1}$. Clearly $r' < r-1$, in fact when $\mathcal{R}_{e-1}$ next receives attention under $\neg$(A1)$\wedge\neg$(A2)$\wedge$(A3) at some stage $s'' > s'$, we must have $r'+1 \leq \mathit{attempt}(e-1,s'') \leq r-1$. This is not possible since we would then enumerate $u(e-1,s'')$ into $U_{e-1,\mathit{attempt}(e-1,s'')}$. □

Lemma 5.3. *All requirements are satisfied, and $A \notin SJT(C)$.*

Proof. Firstly, note that Ψ^C is total, non-decreasing and unbounded (by Lemma 5.2), and we never add inconsistent Ψ^C axioms. Next, fix an $e \in \mathbb{N}$ and let $r = \lim_{s\to\infty} \mathit{region}(e,s)$ and $x = \lim_{s\to\infty} x(e,s)$. Since we never change the parameter $u(e)$ until A fails to promptly permit it, hence $J^A(x)\downarrow = y$ with a permanent axiom say, after stage s_0. We clearly have $\Psi^C(x) = r$, and if $y \in T^e_x$ then $\mathit{attempt}(e) = r$ must be true when y finally shows up in T^e_x. This corresponds to r different numbers forced into T^e_x; each time a number shows up in T^e_x, the C part of the use that puts it in T^e_x is preserved forever. □

This ends the proof of Theorem 5.1. How close to $\emptyset$ (in terms of computational power) can we make the set C in Theorem 5.1? Since each $\mathcal{R}_e$ enumerates into C only finitely often, it is easy to see that the construction can be easily modified to make C low. On the other hand it is not possible to compute an upperbound for the number of C-enumerations for each $\mathcal{R}_e$, because the opponent could refuse to give us his prompt permission for as many times as he wishes. Thus it is not clear if we can make C strongly jump traceable, or even just superlow. We conjecture that $\mathcal{H} = \bigcap\{SJT(W) \mid W \text{ is c.e. and low}\}$. Note that both classes contain no promptly simple members.

6. No c.e. set is strongly jump traceable by every Δ^0_2 set

In this section, we will show that the only c.e. sets which are strongly jump traceable by every Δ^0_2 set, are the computable ones. This property is one which is exclusive to the computable sets - the only way we can jump trace A in this way, is for J^A to be traceable via a constant bound.

Theorem 6.1. *For any c.e. set $W >_T \emptyset$, there is a set $A \leq_T \emptyset'$ such that $W \notin SJT(A)$.*

6.1. **Requirements.** We build a Δ^0_2 set A by full approximation, and an A-order Ψ^A. A tree T is a total function from finite binary strings to finite binary strings, such that for all σ, $T(\sigma^\frown 0)$ and $T(\sigma^\frown 1)$ are incompatible extensions of $T(\sigma)$. At each stage s of the construction we define the sequence of trees $T_0[s] \supseteq T_1[s] \supseteq \cdots \supseteq T_s[s]$, and ensure that for each e, $T_e := \lim_{s\to\infty} T_e[s]$ exists pointwise. In fact, we will ensure that $T_e[s] = T_e$ for some s. To make notations more consistent, for each e, s, σ, we write $T_e(\sigma)[s]$ instead of $T_e[s](\sigma)$ - the value of $T_e[s]$ applied to σ.

For each s, we define the finite string $A_s = T_s(\emptyset)[s]$. Hence by the above, $\lim_{s\to\infty} A_s(x)$ exists for all x, so that by the Limit Lemma, $A \leq_T \emptyset'$. We want to ensure that the following requirements are met:

$$\mathcal{R}_e \quad : \quad \text{Defeat the } e^{th} \text{ } A\text{-trace. That is, for some } x, \text{ either } |V^A_{e,x}| \geq \Psi^A(x), \text{ or else } J^W(x) \notin V^A_{e,x}.$$

We let $\{V^X_{e,x}\}_{x\in\mathbb{N}}$ be the e^{th} trace computing with an oracle, in some effective enumeration. We append $[s]$ to all parameters to denote the value of the parameter at stage s. To choose a *fresh* number x at stage s, means that we choose $x > s$ and larger than any number previously used or mentioned. We write 1^n to denote the finite string of n many 1's

6.2. **Description of strategy.** We remind the reader of the standard strategy in making $W \notin SJT(A)$. We have to define a single order Ψ^A, and defeat each A-trace $\{V^A_x\}_{x\in\mathbb{N}}$ by meeting the requirements above. Let us consider the things we have to do to defeat a single trace. We pick a follower ξ, and enumerate $J^W(\xi)[s] \downarrow = s$ with use u, and wait for s to be traced in V^A_ξ. When s enters $V^A_\xi[s]$, we want W to change so that we can define $J^W(\xi)$ on a new value. Recall from Theorem 5.1 that if W is promptly simple, then we can easily build A to be c.e. and low. Therefore, if the opponent wants to make things difficult for us, he has to make W (which is non-computable) respond very slowly. In particular, the opponent would do the following. He would trace s into V^A_ξ with an A-use larger than $\psi(\xi)$ (i.e. he wraps his axioms around ours), say at stage t. He knows that he has to change W below u eventually to ensure that W is non-computable, but he could wait until we enumerate $\psi(\xi)$ into A. He knows that we have to do that, because we have to pick a new $\xi' > \xi$ and start the process above again with ξ'. Only after we enumerate $\psi(\xi)$ into A, would he then change $W\upharpoonright_u$.

Note that as discussed in Theorem 4.2, if we had to make A c.e., we would not be able to succeed. However, we are allowed to make $A \in \Delta^0_2$, so we could actually restore A-traces back to the state which they were at previously. If the opponent does change $W\upharpoonright_u$ eventually, we could restore A back to what it was, A_t, at stage t. Doing so would put s back into the trace V^A_ξ.

The formal construction is given below. Trees are used to keep track of nodes which we might have to return to, if the opponent changes W at a later stage.

6.3. **Notations.** Each requirement $\mathcal{R}_e$ will make several attempts at defeating the e^{th} A-trace, which we call *attacks*. The Recursion Theorem gives us a list of indices $i_0, i_1, \cdots$ for which we can use to control $J^W(i_n)$. When $\mathcal{R}_e$ begins its i^{th} attack, it will pick an index from the list, which we denote by ξ^e_i. $\mathcal{R}_e$ will then control $J^W(\xi^e_i)$, with W-use u^e_i. Each of the parameters $\xi^e_1, \xi^e_2, \cdots$ represent a separate attack of $\mathcal{R}_e$. Basically, the construction visits $T_e(0)$ if $\mathcal{R}_e$ is performing the first attack, and visits $T_e(1)$ when it is waiting for a $W\upharpoonright_{u^e_1}$-change, i.e. the appropriate numbers have entered the trace V^σ_{e,ξ^e_1} for some $\sigma \succ T_e(0)$. When that happens, we would set $T_e(0) = \sigma$, and the construction will visit $T_e(10)$ to start the second attack, and when the opponent responds again, we will go to $T_e(110)$ to begin the third attack, and so on. Eventually $W\upharpoonright_{u^e_i}$ has to change for some i, and when that happens we can go back to $T_e(1^{i-1}0)$ and enumerate a new computation for $J^W(\xi^e_i)$. Each time $\mathcal{R}_e$ requires attention, we will move $T_r(\emptyset)$ for all $r > e$ above some branch of T_e, so that the requirements $\mathcal{R}_r$ for $r > e$ can start their attacks anew, above the branch of T_e we want restored.

Each requirement $\mathcal{R}_e$ will pick a number $region(e)$, which denotes the number r such that we want to try and make $|V^\sigma_{e,\xi^e_i}| \geq r$ (provided that numbers are always traced). Obviously this means that $\mathcal{R}_e$ will have to enumerate the axiom $\Psi^\sigma(\xi^e_i) \downarrow = r$ for each i^{th} attack it begins. Care has to be taken to ensure that no incompatible Ψ-axioms are enumerated, and that Ψ^A eventually turns out to be an order. Hence $\mathcal{R}_e$ will enumerate Ψ-axioms for the path $T_e(1^{i-1}0)$, when it is beginning the i^{th} attack. Other requirements $\mathcal{R}_k$ for $k > e$ starts above $T_e(1^{i-1}0)$, and enumerates their own Ψ-axioms above $T_e(1^{i-1}0)$. When $\mathcal{R}_e$ needs to begin a new j^{th} attack, it has to start on a new branch $T_e(1^{j-1}0)$, incompatible with $T_e(1^{i-1}0)$.

We build a Turing functional Ψ, which we think of as a c.e. set of axioms. The axioms are of the form $\langle x, y, \sigma\rangle$, which means "given input x, output the number y if $\sigma \subset X$", where X is the oracle. At stage s of the construction, when we say that we set $\Psi(x)[s] \downarrow = y$ with use σ, we mean that we enumerate the following Ψ-axioms: for each $x' \leq x$, enumerate the axiom $\langle x', y, \sigma\rangle$, if for every Ψ-axiom $\langle x', y', \sigma'\rangle$ already present, we have σ' incompatible with σ. This ensures that new axioms are always compatible with existing ones, and that along any path $X \in 2^\omega$, the function Ψ^X will be non-decreasing whenever it is defined.

For each e, i, when $\mathcal{R}_e$ begins its i^{th} attack, we would pick a fresh index for ξ^e_i, and set $\Psi(\xi^e_i)[s] \downarrow = region(e)$, with use $T_e(1^{i-1}0)$. It will then carry out its i^{th} attack by setting $J^W(\xi^e_i) \downarrow$, and then wait for the value to be traced. Each time a new value appears in the trace (on some use extending $T_e(1^{i-1}0)$), we increment the counter $attempt(e, i)$ by 1. When the counter $attempt(e, i)$ reaches $region(e)$ for *any* attack i, then $\mathcal{R}_e$ is permanently satisfied, and need not be considered any further (unless it is later initialized).

We say that an attack of $\mathcal{R}_e$ is *e-pending* if it is waiting for some number to enter the appropriate trace; at any time at most one attack of $\mathcal{R}_e$ is pending, and any e-pending attack will have priority over all the other attacks of the same requirement. When we *initialize* a requirement $\mathcal{R}_k$, we set $region(k), \xi^k_i, u^k_i$ undefined for all i,

set $attempt(k,i) = 0$ for all i, and remove all k-pending status from the attacks of $\mathcal{R}_k$. We let $Full(\sigma)$ be the full tree above σ, i.e. $Full(\sigma)(\nu) = \sigma^\frown\nu$ for all ν. All parameters retain their assigned values until reassigned or initialized.

6.4. **The construction.** At stage $s = 0$, we make all parameters undefined, and set $T_0[0] = Id$. At stage $s > 0$ we define the trees $T_0[s] \supseteq \cdots \supseteq T_s[s]$. For each $e \leq s$, we do the following. If there is some i such that $attempt(e,i) \geq region(e)$, i.e. the i^{th} attack has succeeded, then let $T_e[s] = T_e[s-1]$, and go to the next e. Otherwise, do the following:

(1) If $region(e)$ is currently undefined, pick a fresh follower for it.
(2) If there is currently no e-pending attack, we set $T_e[s] = T_e[s-1]$ and begin a new attack by doing the following. Pick the least i such that $\xi_i^e \uparrow$, and start the i^{th} attack. We pick fresh followers for ξ_i^e and u_i^e, and set $\Psi(\xi_i^e)[s] \downarrow = region(e)$ with use $T_e(1^{i-1}0)[s]$. We initialize $\mathcal{R}_k$, and set $T_k[s] = Full(T_e(1^{i-1}0)[s])$ for all $e < k \leq s$. Declare i to be the attack of $\mathcal{R}_e$ that is now e-pending. Go to next stage.
(3) If there is a currently e-pending attack i, pick the action which applies:
   (a) $J^W(\xi_i^e)[s] \uparrow$: we set $J^W(\xi_i^e) \downarrow = s$ with use u_i^e, and set $T_e[s] = T_e[s-1]$. We initialize $\mathcal{R}_k$ and set $T_k[s] = Full(T_e(1^{i-1}0)[s])$ for $e < k \leq s$. Go to the next stage.
   (b) $J^W(\xi_i^e)[s] \downarrow$, *but for some* $\sigma \supset T_e(1^{i-1}0)[s-1]$, *we have* $|V^\sigma_{e,\xi_i^e}[s]| > attempt(e,i)$. *(All searches and computations are limited to a use and a search time of* s*)*: hence there is some extension σ which gives us an increase in the size of the trace. Do the following.
      - Set $T_e(1^{i-1}0^\frown\nu)[s] = \sigma^\frown\nu$ for all ν, and $T_e(\eta)[s] = T_e(\eta)[s-1]$ everywhere else.
      - Increase the value of $attempt(e,i)$ to $|V^\sigma_{e,\xi_i^e}[s]|$.
      - If $attempt(e,i)$ is now at least as big as $region(e)$, we initialize $\mathcal{R}_k$ and set $T_k[s] = Full(T_e(1^{i-1}0)[s])$ for all $e < k \leq s$, and go to the next stage. Otherwise we have to decide which attack is going to be e-pending next: firstly, remove the e-pending status from i. Next, pick the least j, if there is one, such that $\xi_j^e \downarrow$ and $J^W(\xi_j^e) \uparrow$, and declare j to be e-pending.
   (c) *Neither (a) nor (b) holds*: Do nothing, let $T_e[s] = T_e[s-1]$, and go to the next e.

6.5. **Verification.** It follows directly from the steps in the construction that at each stage s, the trees $T_0[s] \supseteq \cdots \supseteq T_s[s]$ are all defined. Let $A = \lim_{s\to\infty} T_s(\emptyset)[s] \leq_T \emptyset'$ (by Lemma 6.2). It is obvious that if an attack i of $\mathcal{R}_e$ is e-pending, then it remains e-pending until either $\mathcal{R}_e$ is initialized, or $attempt(e,i)$ increases. This is the reason why we have the "pending" status - to hold the construction through $T_e(1^{i-1}0)$ until we are able to find a node extending $T_e(1^{i-1}0)$, with a larger trace size. We will first show that all trees $T_e[s]$ will eventually stop changing:

Lemma 6.2. *For each e, $\mathcal{R}_e$ is initialized only finitely often, and $T_e[s] = T_e$ for some s.*

Proof. We prove the above simultaneously by induction on e. Assume that s_0 is a stage where the statement holds for $e' < e$. After s_0, $\mathcal{R}_e$ can only be initialized when we are taking actions for $\mathcal{R}_{e-1}$. Since $T_{e-1}[s_0] = T_{e-1}$, it follows that $\mathcal{R}_{e-1}$ does not begin any new attack after s_0. Let $u = \max\{u_i^{e-1}[s_0] : \text{attack } i \text{ of } \mathcal{R}_{e-1} \text{ eventually begins}\}$, and wait until $W_{s_1}\restriction_u = W\restriction_u$. After stage s_1, $\mathcal{R}_e$ is never initialized.

Next, we argue that T_e exists. Since $\mathcal{R}_e$ is initialized only finitely often, let $r = \lim region(e)$. Since every change in $T_e[s]$ corresponds to an increase in

$attempt(e,i)$ for some attack i, it follows that if T_e does not exist, then no attack remains e-pending forever, and that all attacks of $\mathcal{R}_e$ are eventually started. We assume, contrary to the statement of the lemma, that this is the case. We let $x = \limsup_{i\in\mathbb{N}} attempt(e,i)$, and we may also assume $x < r$ (in fact, no attack i ever attains $attempt(e,i) = r$), otherwise $\mathcal{R}_e$ will be permanently satisfied. Then, we claim that W is computable, for a contradiction. For each i, we can compute $W\upharpoonright_{u_i^e}$ as follow: run the construction, and wait until the first $j \geq i$ such that $attempt(e,j) = x$, say at stage $s_2(i)$.

We claim that $W_{s_2(i)}\upharpoonright_{u_i^e} = W\upharpoonright_{u_i^e}$ is correct for almost all i: suppose that $W_{s_2(i)}\upharpoonright_{u_i^e} \neq W\upharpoonright_{u_i^e}$. Therefore $J^W(\xi_j^e)[t]\uparrow$ for some $t > s_2(i)$ after the change in W. This means that attack j must become e-pending eventually, and since no attack can be forever e-pending, we must have $attempt(e,j) > x$. By the definition of x, the procedure above to compute W can fail for only finitely many i's. Hence W is computable, and this contradiction shows that T_e must exist. □

Fix a requirement $\mathcal{R}_e$. There is some attack i such that either $attempt(e,i) \geq \lim region(e)$, or i is forever e-pending. Let $i(e)$ denote this i. Note that $i(e)$ denotes the attack that has a successful run, i.e. either $V^A_{e,\xi^e_{i(e)}}$ fails to trace $J^W(\xi^e_{i(e)})$, or has size larger than $region(e)$. The following facts are not hard to verify:

(1) For every e, $\xi^e_{i(e)} < \xi^{e+1}_{i(e+1)}$, since the former is always chosen before the latter.
(2) For every e, $A \supset T_e(1^{i(e)-1}0)$.
(3) For each attack i of $\mathcal{R}_e$ that is eventually started, we have $\Psi^{T_e(1^{i-1}0)}(\xi^e_i) = r$, where $r = \lim region(e)$.

Lemma 6.3. *Ψ^A is an order, i.e. total, non-decreasing, unbounded.*

Proof. From the facts above, we have for every e, $\Psi^A(\xi^e_{i(e)}) = \lim region(e)$. This clearly means that Ψ^A is total, and unbounded. Next, we want to see that Ψ^X is non-decreasing along any path $X \in 2^\omega$. Suppose $\mathcal{R}_e$ enumerates the axiom $\langle x, y, \sigma\rangle$ at stage s; note that this only happens when $\mathcal{R}_e$ is starting a new attack i, hence $T_e(1^{i-1}0)[s] = \sigma \subset X$.

Suppose $\mathcal{R}_k$ enumerates a Ψ-axiom $\langle x', y', \sigma'\rangle$ at some stage $t > s$, where $\sigma' \subset X$. Since σ and σ' are compatible this means that $x' > x$. The only thing we have to ensure, is that $y' \geq y$. If $\mathcal{R}_k$ was initialized between s and t, then $y' > y$ so there is no problem. We may therefore assume that $\mathcal{R}_k$ was not initialized. Hence, $k \not> e$, and if $k = e$ then $\sigma' \supset T_e(1^i)[t] = T_e(1^i)[s]$ has to be incompatible with σ so that is impossible (because we start a new attack of $\mathcal{R}_e$ at stage t). Lastly, if $k < e$ then because T_e is always restarted above a branch of T_k which is k-pending, we must have $\sigma \supset T_k(1^{j-1}0)[s]$ for some pending j-attack of $\mathcal{R}_k$ at stage s. Since a new attack of $\mathcal{R}_k$ is started at stage t, we must have $\sigma' \supset T_k(1^j)[t] = T_e(1^j)[s]$ incompatible with σ, again impossible. □

Lemma 6.4. *All requirements are met.*

Proof. Fix an e, and consider the $i(e)^{th}$ attack of $\mathcal{R}_e$. Let $x = \xi^e_{i(e)}$, and $r = \lim region(e) = \Psi^A(x)$. Note that $|V^A_{e,x}| \geq |V^{T_e(1^{i(e)-1}0)}_{e,x}|$, so if $attempt(e,i(e)) \geq r$, then $|V^A_{e,x}| \geq r$ and we are done. Suppose not, i.e. $i(e)$ is forever e-pending. We must have $J^W(x)\downarrow$ and receives the definition at a stage s, when $i(e)$ is finally e-pending. Hence $J^W(x) = s \notin V^{T_e(1^{i(e)-1}0)}_{e,x}[t]$, where we have $|V^{T_e(1^{i(e)-1}0)}_{e,x}[t]| = attempt(e,i(e))[s]$ at the previous stage $t < s$, where we increased $attempt(e,i(e))$. Since $A \supset T_e(1^{i(e)-1}0)$, hence if $s \in V^A_{e,x}$, then some σ where $T_e(1^{i(e)-1}0) \subset \sigma \subset A$, will return a successful search under step 3b in the construction at a later stage

past s, which will again increase $attempt(e, i(e))$. This is not possible since $i(e)$ never loses its e-pending status after stage s. □

7. The degrees containing hyper jump traceable sets are not downwards dense

In this section, we demonstrate yet another strange behaviour exhibited by the degrees containing hyper jump traceable sets. One might expect that if a class of degrees contains very little information, then every non-computable c.e. degree would bound a member of that class. For instance, every non-computable c.e. degree bounds a strongly jump traceable set $A >_T \emptyset$. However, this is not true for the hyper jump traceable sets - they are not downwards dense despite their strong resemblance to the computable sets.

This is due to the fact that we have to consider non-computable functions, which are slightly more tricky to deal with because they may change their mind in their approximation. Such functions might in reality be eventually constant, but infinitely often will try and fool us into thinking that they are unbounded. An example is the following function f with the approximation f_s:

$x =$	0	1	2	3	4	5
$f_0(x)$	1	2	3	4	5	6
$f_1(x)$	1	1	2	3	4	5
$f_2(x)$	1	1	1	2	3	4
$f_3(x)$	1	1	1	1	2	3

Informally, a hyper jump traceable set A imposes a lot of negative restraint, so it is difficult to get elements into the set. In this respect, A is very close to being computable. Even though this is the case, the manner in which A allows numbers to enter is a little bit more complicated than the other low sets such as K-trivials or strongly jump traceable sets (both of these constructions can be described using cost functions). Roughly speaking, elements can only enter A at certain "gap phases" in which restraints on A are temporarily taken down; these windows of opportunity are synchronized with changes in the approximation to functions such as f above.

As we have already seen, the construction of a non-computable hyper jump traceable set (Theorem 4.3) is not compatible with making it promptly simple (Theorem 5.1), because of this reason. Such a construction also does not seem to be compatible with permitting below a given c.e. set $B >_T \emptyset$, again due to the same reason : when permission is given to us by B to change A, we might not be able to do so because some function h^W is currently telling us that it is a true order function (hence imposing a large amount of A-restraint). There is no way for us to figure out if it is lying or not, and when the opportunity for us to effect changes on A passes, the opponent will then reveal the true form of h^W. The resulting A-gap which is opened is of no use to us for B-permission, though we could still use it to make A non-computable. The failure of this strategy can be turned into a proof of the following theorem.

Theorem 7.1. *There is a c.e. set $A >_T \emptyset$ such that for any set X, if $\emptyset <_T X \leq_T A$, then X is not hyper jump traceable.*

An immediate corollary to this theorem, is that there is a single c.e. set which forms a minimal pair with every hyper jump traceable set:

Corollary 7.2. *There is a c.e. set $A >_T \emptyset$, such that if X is hyper jump traceable, then A and X forms a minimal pair.*

Therefore, the hyper jump traceable c.e. sets are far away from being promptly simple, in the sense that they can all be capped by a single c.e. set.

7.1. **Requirements.** We build a coinfinite c.e. set A, and ensure that it is non-computable via the usual simple requirements.

$\mathcal{P}_e$: $|W_e| = \infty \Rightarrow A \cap W_e \neq \emptyset$,

$\mathcal{N}_e$: If Φ_e^A is total, then either $\Phi_e^A = X_e$ is computable, or else X_e is not hyper jump traceable.

Here, Φ_e denotes the e^{th} Turing reduction. The stage s use of the computation $\Phi_e^A(x)$ is denoted by $\varphi_e(x,s)$. At each requirement $\mathcal{N}_e$, we will build a c.e. set C and a Turing functional h, for the sake of demonstrating that X_e is not hyper jump traceable. We ensure that h^C is non-decreasing, and will ultimately be total and unbounded unless X_e is computable. Consider an effective enumeration $\{T_{0,x}\}_{x\in\mathbb{N}}, \{T_{1,x}\}_{x\in\mathbb{N}}, \cdots$ of all possible traces with an oracle. We split $\mathcal{N}_e$ into the subrequirements $\mathcal{N}_{e,0}, \mathcal{N}_{e,1}$, where $\mathcal{N}_{e,i}$ will try and diagonalize against the i^{th} trace:

$$\mathcal{N}_{e,i} : \text{if } X_e >_T \emptyset, \text{ then for some } x, \text{ either } |T^C_{i,x}| \geq h^C(x), \text{ or } J^{X_e}(x) \downarrow \notin T^C_{i,x}.$$

7.2. **Description of strategy.** We first outline the basic strategy used to satisfy a single $\mathcal{N}_{e,i}$ requirement, which has to diagonalize the i^{th} trace $\{T^C_{i,x}\}$. Suppose $\mathcal{N}_{e,i}$ wants to try and make $|T^C_{i,x}| \geq 2$ for some x, where C is built at the main requirement $\mathcal{N}_e$. The naive try would be to do the following. We pick some x_0, and set $J^{X_e}(x_0) \downarrow = s$ with X_e-use j_0, and wait for the value s to be traced in T^C_{i,x_0}. When s shows up in T^C_{i,x_0}, we would then define $R\restriction_{j_0} = X_e\restriction_{j_0}$ where R is the procedure we are building to compute X_e. We would then repeat the above with a larger $j_1 > j_0$ and a different $x_1 > x_0$. At the end of the day if $X_e\restriction_{j_n}$ never changes after $R\restriction_{j_n}$ is set, then R correctly computes X_e; on the other hand if $X_e\restriction_{j_n}$ changes on some j_n we could go back and define $J^{X_e}(x_n) \downarrow =$ a new value, and we would be able to meet $\mathcal{N}_{e,i}$.

The above strategy works well when considering $\mathcal{N}_{e,i}$ in isolation, but cannot possibly work when combined with other $\mathcal{N}_{e,j}$'s. This is because we had not made use of the fact that $X_e \leq_T A$ - indeed we are trying to show that there can be no hyper jump traceable set. The problem shows up when the opponent takes a long time to trace $J^{X_e}(x_0)$ into T^C_{i,x_0}; in fact the opponent can delay responding until we define $h^C(y) \downarrow = 3$ for some $y > x_0$, say with C-use $u(y)$. He would then trace $J^{X_e}(x_0)$ into T^C_{i,x_0} with a C-use larger than $u(y)$. When we pick a new $x_1 > y$ and j_1 and repeat the above, we would have to define $h^C(x_1) = 2$; in order to do that we have to enumerate $u(y)$ into C and clear the opponent's trace T^C_{i,x_0} at the same time. Thus even if X_e changes below j_0 in future, we will not be able to benefit from it.

The fortunate thing is that we do not need to consider arbitrary X_e, only those X_e below A. When we set $R\restriction_{j_n}$ we could actually freeze $A\restriction_{\varphi_e(j_n)}$ at the same time, so that $X_e\restriction_{j_n}$ will not change. However we need to arrange for the A-restraint to be dropped infinitely often, since the restraint might $\to \infty$ in the case when X_e is computable. The following are the steps for doing this, at the n^{th} cycle:

Step 1: set $J^{X_e}(x_n) \downarrow$ with use j_n, and wait for the trace to appear in T^C_{i,x_n}. When it appears, we *open up a gap* by dropping all A-restraint. Go to step 2.

Step 2: we *close the gap* by doing the following. Check if $X_e\restriction_{j_n}$ has changed during the gap open phase. If yes, then success is achieved for $\mathcal{N}_{e,i}$. If not, we compute $R\restriction_{j_n} = X_e\restriction_{j_n}$, and re-impose A-restraint. Start the $n+1^{th}$ cycle from step 1 with a larger j_{n+1}, x_{n+1}.

Thus if only finitely many gaps are opened, then when the last gap is closed we have the desired X_e-change. If infinitely many gaps are opened and closed, then

R computes X_e correctly. On a construction tree, we can arrange things so that restraint is dropped at infinitely many stages. Note the similarity between this strategy, and the one used in Theorem 5.1. In both cases we wait for the opponent to respond with a trace on an index x_n. When that happens, we challenge the opponent to change X_e. If the opponent refuses to respond with a X_e-change we will *abandon this index x_n and never return to it.* In Theorem 5.1 we never need to return to an abandoned index since a prompt change in X_e is guaranteed. In our case here, we can impose restraint and force X_e to change only during gap phases - even though the indices which were abandoned previously cannot benefit from this change, we make sure that the one which is currently active can benefit from the change.

We now describe the strategy in the general case. Assume first of all that X_e is c.e.. We start by picking a number k and x_0, and set $h^C(x_0) \downarrow = k$. We now need to have k many consecutive gap phases in which X_e changes. Each run of attempts is called a *cycle*. We will need to have k many attempts $R_1, \cdots, R_k$ at computing X_e, and have restraint functions $r_1, \cdots, r_k$ for each of these procedures.

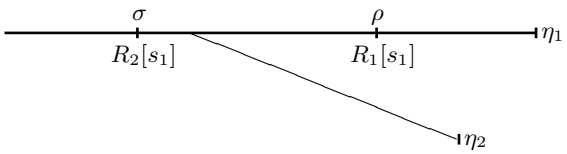
We start a new cycle by picking a fresh follower x. This follower is used throughout the current cycle, and will only be abandoned when the current cycle is ended and a new one begins. We start by running the basic strategy and waiting for it to return. That is, we wait for $J^{X_e}(x)$ to show up in $T^C_{i,x}$. When that happens, we open up a 1-gap by dropping restraint r_1. At the next expansionary stage, we close the 1-gap by the following: if there had been no X_e change below the use, then we extend the procedure R_1 and re-impose the r_1-restraint. We end the current cycle and start a new one. If there had been a change then we run the basic strategy again with r_2, R_2 in place of r_1, R_1, since we are now able to set a new $J^{X_e}(x)$ axiom. This continues on until either we get k consecutive gap phases in this current cycle (and hence $|T^C_{i,x}| \geq k$), or else some k'-gap is closed with no X_e-change, ending the cycle.

If a single subrequirement $\mathcal{N}_{e,i}$ of $\mathcal{N}_e$ opens and closes infinitely many gaps, then X_e would be computable. If k' is the largest number such that infinitely many k'-gaps are opened and closed, then $R_{k'}$ is total and computes X_e correctly. In this case $\mathcal{N}_{e,i}$ actually abandons infinitely many indices making h^C into a constant function. This is how we exploit such order functions to our advantage - infinitely often it will appear to the opponent that it is a real order.

Now let us consider the general situation when X_e is Δ^0_2. For simplicity we will only describe what goes on in cycles which only require two consecutive gap phases in which X_e changes. We will construct procedures R_1, R_2, and restraints r_1, r_2 just as in the c.e. case. Each time a 1-gap is closed, we have the two cases (as before): either no X_e-change (in which case we extend R_1, r_1 and end the cycle), or else there is an X_e-change (in which case we carry on with a 2-gap). However, when a 2-gap is closed, the opponent has one extra option. He could now recover X_e back to the use of the computation enumerated earlier before the 1-gap opened. Thus, the follower x would now be useless, since we are unable to enumerate a new (third) computation for $J^{X_e}(x)$.

The important point to note is that should this happen (say at stage t), then we will have $X_{e,t} \supset R_1$ again. We could then end the current cycle, and we would have made progress because R_1 is currently correct. Also, R_2 is never extended unless a 2-gap is closed with no X_e-change. Hence $R_2[t] \subseteq R_1[t] \subset X_{e,t}$ is correct as well. Hence if infinitely many cycles are ended with R_2 being extended, then R_2 is total and computes X_e correctly. On the other hand if $\lim |R_2| < \infty$, then almost all cycles will be ended with a recovery of X_e back to a previous configuration. In this case, R_1 will be total and computes X_e correctly.

We illustrate the above with an example. Suppose at stage s_1, we have $R_2[s_1] = \sigma \subset R_1[s_1] = \rho$. We have currently started a new cycle with a computation with use $\eta_1 \supset \rho$. Suppose this computation shows up in the trace, and we then open a 1-gap. When the 1-gap is closed, we found that X_e has changed. We then challenge the opponent with a new computation with use η_2. Note that at this stage we had only opened a 1-gap but not a 2-gap so that η_2 and ρ are incompatible extensions of σ:



When the opponent returns by tracing the new computation, we will open up a 2-gap. Note that during the 2-gap, X_e can change even below σ. When the 2-gap is closed at stage s_2, there are three possibilities:

(1) *No X_e-change, that is $X_{e,s_2} \supset \eta_2$.* Then, we extend $R_2[s_2] = \eta_2$, and restart R_1 above η_2. End the cycle.
(2) *X_e returns to the previous use, i.e. $X_{e,s_2} \supset \eta_1$.* Then, we extend $R_1[s_1] = \eta_1$. End the cycle. Note that R_2 is unchanged by this action.
(3) *Otherwise.* Then, X_{e,s_2} is new and we can enumerate a new computation with that use. The node can now stop acting, having completed the diagonalization successfully.

Note that R_1 and R_2 are always built by extension (unless R_1 is restarted due to R_2 injuring it). Hence one of the two will be total computable, and computes X_e correctly. In general, if n-gaps are opened infinitely often, then one of $R_1, \cdots, R_n$ will be total and computes X_e correctly.

7.3. **Construction tree layout.** The construction takes place on an infinite branching tree. Nodes of even length $|\alpha| = 2e$ are assigned the requirement $\mathcal{P}_e$ with a single outcome 0. Nodes of length $|\alpha| = 2\langle e, i\rangle + 1$ are assigned the requirement $\mathcal{N}_e$ if $i = 0$, and the subrequirement $\mathcal{N}_{e,i-1}$ if $i > 0$. Nodes assigned the requirement $\mathcal{N}_e$ have two outcomes $\infty <_{left} f$. They stand respectively for infinitely many, and finitely many expansionary stages. The subrequirement $\mathcal{N}_{e,i}$ of $\mathcal{N}_e$ has infinitely many outcomes $f >_{left} 1 >_{left} 2 >_{left} \cdots$, with order type ω^*. The rightmost outcome f represents the fact that X_e is non-computable and hence diagonalization will have to succeed, while the outcome n to the left of f represents the fact that X_e is computable via one of the attempts $R_1, R_2, \cdots, R_n$ at computing it. The reason why we label the outcomes as such is because during the construction, $\mathcal{N}_{e,i}$ will make a few attempts at computing X_e; each time it plays outcome n we make progress on one of the attempts $R_1, R_2, \cdots, R_n$. Note that we also identify outcome f with the natural number 0, so as to keep the notations consistent.

We order nodes on the tree lexicographically: denote $\alpha <_{left} \beta$ to mean 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 $\alpha(i) <_{left} \beta(i)$). We say that α *is a $\mathcal{Q}$-node* if α is assigned the requirement $\mathcal{Q}$. α is a *positive node*, if α is a $\mathcal{P}_e$-node for some e. We say that α is a *top node*, if α is an $\mathcal{N}_e$-node for some e, and that α is an *i-bottom node of τ* if α is an $\mathcal{N}_{e,i}$-node for some e, i, and $\tau \subset \alpha$ where τ is an $\mathcal{N}_e$-node. If α is a bottom node, then $\tau(\alpha)$ denotes the (unique) top node τ such that α is a bottom node of τ. Note that in order to keep the labeling of nodes simple, we had actually allowed some nodes to be given a label for which it never needs to act. Therefore, we need to specify when a bottom node α is an *active* node: if $\alpha \supset \tau(\alpha)^\frown\infty$, and for every $\tau(\alpha)$-bottom node β such that $\beta \subset \alpha$, we also have $\beta^\frown f \subset \alpha$. Bottom nodes which are not

active does nothing in the construction, and will always play the outcome f when visited.

7.4. **Notations.** Let α be an active $\mathcal{N}_{e,i}$-node, and $\tau = \tau(\alpha)$. We measure the length of convergence at the mother node τ:

$$l_\tau[s] := \max\{y < s \mid (\forall x < y)\Phi_e^A(x)[s] \downarrow\}.$$

α will act each time $l_\tau[s]$ gets long enough, at τ-expansionary stages. At the node τ we build a c.e. set C_τ and a Turing functional h_τ[9]. The C_τ-use of an enumerated $h_\tau(x)$-axiom (if it still applies at stage s) is denoted by $u_\tau(x, s)$.

α works by first picking a number $region_\alpha$. It will also pick a number x_α, and then set $h_\tau^{C_\tau}(x_\alpha) \downarrow = region_\alpha$. In this current run, α will attempt to diagonalize against the i^{th} trace by focussing on $T_{i,x_\alpha}^{C_\tau}$. x_α is picked from an infinite list of indices, given by the Recursion Theorem, which we will use to control $J^{X_e}(x_\alpha)$. α will also pick a number j_α, which denotes the length of X_e such that α will enumerate $J^{X_e}(x_\alpha)$-computations with that much use. The parameter $attempt_\alpha$ will record how successful α is in diagonalization - this keeps track of at least how many elements are currently in $T_{i,x_\alpha}^{C_\tau}$. At the same time, α will be trying to build several effective procedures to compute X_e. More specifically, it will be simultaneously building at any one time, $region_\alpha$ many procedures to compute X_e, and imposing various restraints for each of these procedures. These procedures are denoted by $R_{\alpha,1}, R_{\alpha,2}, \cdots$, which should be viewed as finite strings. Let $r_{\alpha,n}$ denote the A-restraint function imposed for the n^{th} procedure $R_{\alpha,n}$. Finally, we let F_α denote the state of α's strategy. This may be 0 (which means that α has not yet started on its strategy), 1 (which means that α is waiting for a number to appear in the trace), or 2 (which signifies that α has opened a certain A-gap).

A stage s is *τ-expansionary*, if either $s = 0$, or else

(1) τ is visited at stage s of the construction,
(2) $l_\tau[s] > l_\tau[s^-]$ where s^- is the previous τ-expansionary stage, and
(3) $l_\tau[s] > j_\beta$ for every active τ-bottom node β.

When we *initialize a bottom node* α, we set $region_\alpha, attempt_\alpha, x_\alpha$ and j_α undefined, and set $F_\alpha = 0$, $r_{\alpha,n} = 0$ and $R_{\alpha,n} = \emptyset$ for all n. When we *initialize a top node* τ, we set $C_\tau = \emptyset$, remove all axioms in h_τ, and initialize all τ-bottom nodes. During the construction, we will enumerate axioms for $J^{X_e}(x)$. The index x is chosen from an infinite list of indices given to us by the Recursion Theorem. When we say at a certain stage s, that $J^{X_e}(x)[s] \downarrow$, we mean that there is an axiom (previously enumerated), that currently apply at stage s.

7.5. **The construction.** At stage $s = 0$, initialize all nodes, and do nothing. Let $s > 0$. We build the stage s approximation to the true path, δ_s of length s inductively. Assume that $\alpha = \delta_s\restriction_d$ has been defined for $d < s$. There are three possibilities for α.

(1) *α is a $\mathcal{P}_e$-node*: let $\delta_s(d) = 0$, and if $W_{e,s} \cap A_s \neq \emptyset$ do nothing. Otherwise, check if there is some $x \in W_{e,s}$, where $x > 2e$, $x > \max\{t < s \mid \delta_t <_{left} \alpha\}$, and for each bottom node $\beta \subset \alpha$, we have $x > \max\{r_{\beta,n} \mid n <_{left} \alpha(|\beta|)\}$. If such an x exists, enumerate the least such into A and initialize all nodes $\beta \supset \alpha$.
(2) *α is an $\mathcal{N}_e$-node*: if s is α-expansionary, let $\delta_s(d) = \infty$. Otherwise let $\delta_s(d) = f$.

[9]We use lowercase h_τ even though it is a Turing functional; this is to respect the fact that $h_\tau^{C_\tau}$ is intended to be an order function.

(3) *α is an $\mathcal{N}_{e,i}$ node*: let $\tau = \tau(\alpha)$. If $attempt_\alpha = region_\alpha$ or α is not active, then we do nothing. Otherwise, take the relevant action below based on F_α:
   (a) $F_\alpha = 0$: pick fresh numbers for $region_\alpha$, x_α and j_α. Set $attempt_\alpha = 0$. For all $x' \leq x_\alpha$ for which no $h_\tau^{C_\tau}$-axioms currently apply, set $h_\tau^{C_\tau}(x')[s] \downarrow = region_\alpha$ with a fresh use $u_\tau(x', s)$. Also set $F_\alpha = 1$.
   (b) $F_\alpha = 1$: there are three possibilities.
      (i) $J^{X_e}(x_\alpha)[s] \uparrow$: we set $J^{X_e}(x_\alpha)[s] \downarrow = attempt_\alpha + 1$ with use σ, where $\sigma = X_{e,s}{\restriction_{j_\alpha}}$. Set $r_{\alpha,attempt_\alpha+1} = \varphi_e(j_\alpha, s)$.
      (ii) $J^{X_e}(x_\alpha)[s] \downarrow$ *and appears in* $T_{i,x_\alpha}^{C_\tau}$: set $F_\alpha = 2$, and open up an A-gap by letting $\delta_s(d) = attempt_\alpha + 1$. We also call this action *opening a k-gap*, where $k = attempt_\alpha + 1$.
      (iii) *otherwise*: do nothing since we are waiting for $J^{X_e}(x_\alpha)$ to get traced.
   (c) $F_\alpha = 2$: let s^- be the previous visit to α. Close the A-gap, and take the appropriate actions listed below. There are three possibilities.
      (i) $X_{e,s}{\restriction_{j_\alpha}} = X_{e,s^-}{\restriction_{j_\alpha}}$: (i.e. X_e recovers). We now have to abandon the current value of x_α. To do that, we first enumerate $u_\tau(x_\alpha, s)$ into C_τ to clear all relevant axioms, and then pick a fresh value for x_α. For all $x' \leq$ the newly chosen x_α for which no $h_\tau^{C_\tau}$-axioms currently apply, set $h_\tau^{C_\tau}(x')[s] \downarrow = region_\alpha$ with a fresh use $u_\tau(x', s)$.
      Next, set $r_{\alpha,attempt_\alpha+1} = \varphi_e(j_\alpha, s)$, and set $R_{\alpha,attempt_\alpha+1} = X_{e,s}{\restriction_{j_\alpha}}$. Reset $attempt_\alpha = 0$, pick a fresh value for j_α, and set $F_\alpha = 1$.
      (ii) $J^{X_e}(x_\alpha)[s] \downarrow$: (i.e. X_e goes back to a previous configuration). Pick a fresh x_α and adjust $h_\tau^{C_\tau}$ as in 3(c)(i) above. Extend $R_{\alpha,k} = X_{e,s}{\restriction_{j_\alpha}}$, where $k = J^{X_e}(x_\alpha)[s]$. Also set $r_{\alpha,k'} = \varphi_e(j_\alpha, s)$ for all k' such that $k \leq k' \leq attempt_\alpha + 1$, since such a k'-gap has been recently opened, and the restraint has to be updated. Reset $attempt_\alpha = 0$, pick a fresh value for j_α, and set $F_\alpha = 1$.
      (iii) *otherwise*: set $r_{\alpha,attempt_\alpha+1} = \varphi_e(j_\alpha, s)$. Increase $attempt_\alpha$ by 1, and set $F_\alpha = 1$. Pick a fresh value for j_α.

   In all cases other than 3(b)(ii), the outcome played is f, where all restraints $r_{\alpha,n}$ will be held.

This ends the definition for δ_s. At the end of stage s, initialize all nodes $\beta >_{left} \delta_s$. This completes the construction.

7.6. **Verification.** Define the true path of the construction TP, by the following: for all n, $TP{\restriction_n}$ is visited infinitely often, and $\delta_t <_{left} TP{\restriction_n}$ only finitely often. Note that TP exists: this is because if α is an active bottom node, then the only outcomes it can play when visited at a stage s must be one of $\{f, 1, 2, \cdots, region_\alpha[s]\}$, and $region_\alpha$ is reassigned only if α is initialized. Also, it is obvious that every node α on TP is initialized only finitely often, so we let $true(\alpha)$ be the least stage t such that α is visited at stage t, and α is never initialized after stage t.

Lemma 7.3. *A is non-computable.*

Proof. Firstly observe that if β is a bottom node on TP, then it can increase $r_{\beta,n}$ at a stage s either under step 3(b)(i), or under step 3(c) of the construction. In the latter case it must be that some outcome $\leq_{left} n$ was played at the previous visit to β. In the former case, if $s > true(\beta)$, then nothing happens until the next A-gap is opened by β, where outcome n will be played. Thus each positive node α

on the true path has to obey only a finite amount of restraint on A, and so A is non-computable. □

The next fact to establish, is that for all e, requirement $\mathcal{N}_e$ is satisfied. Let τ be an $\mathcal{N}_e$-node on TP, and suppose that $\Phi_e^A = X_e$ is non-computable. There must be infinitely many τ-expansionary stages because all τ-bottom nodes $\supset \tau^\frown f$ are not active. Hence τ has true outcome ∞. We show that every τ-bottom node α on TP must have true outcome f:

Lemma 7.4. *Suppose α is a τ-bottom node on TP, with true outcome $o <_{left} f$. Then, X_e is computable (contrary to assumption).*

Proof. Let $s_0 = true(\alpha^\frown o)$. Firstly, observe that after stage s_0, we must always have $attempt_\alpha < o$. Suppose not - then at some stage when an o-gap is closed, we increase $attempt_\alpha$ to the value o under step 3(c)(iii) of the construction. Since outcome $o+1$ cannot be played after stage s_0, it follows that F_α stays forever at 1 and thus outcome o is never played again - a contradiction. Thus, $attempt_\alpha$ is always $< o$.

At each stage $s > s_0$ such that outcome o is played, we have an opening of an o-gap. At the next visit to α, the o-gap opened at stage s will be closed. Thus, we have $j_\alpha \to \infty$. Furthermore each o-gap can only be closed under step 3(c)(i) or (ii). Hence, there is some largest $n \leq o$ such that $|R_{\alpha,n}[t]| \to \infty$. We may thus assume that s_0 is large enough so that no $R_{\alpha,m}$ is reassigned after s_0, for all $m > n$.

Claim 7.5. *Suppose that $s_1 > s_0$ is a stage in which $R_{\alpha,n}$ is assigned the value σ. Then, $\sigma \subset X_e$.*

Proof of claim. Suppose s_1 and σ are as above. We show by an induction, that if an n-gap is opened at any stage $s_2 > s_1$, then we must have

(I1) $X_{e,s_2} \supset \sigma$, and
(I2) any axioms of the form $J^\rho(x_\alpha)[s_2] = k$ enumerated by stage s_2, must satisfy $k \leq n$ and $\rho \supset \sigma$.

This is enough to guarantee that $\sigma \subset X_e$ by (I1), since Φ_e^A is total. At stage s_1 when $R_{\alpha,n}$ is assigned, it must also be that a "cycle" is ended, i.e. $attempt_\alpha$ is set to 0. Thus when the next n-gap is opened, we must have $X_{e,t} \supset \sigma$, which proves the base case for (I1). As for (I2), note that $X_{e,s_1}\restriction_{|\sigma|}$ is preserved from s_1 until the next n-gap is opened, and hence any computation set in the meantime must have use $\rho \supset \sigma$.

Now consider an arbitrary $s_2 > s_1$ such that an n-gap is opened at stage s_2, and assume (I1) and (I2) holds. Let $s_3 > s_2$ be the next time an n-gap is opened, our task is to show firstly that $\sigma \subset X_{e,s_3}$.

This current cycle must end at some stage t, where $s_2 < t < s_3$, with the closing of a k'-gap, for some $k' \geq n$ and $attempt_\alpha$ reset to 0. Consider the action taken when the cycle ends. Either step 3(c)(i) or (ii) applies to close the k'-gap. If 3(c)(i) applies, then $k' = n$ because $R_{\alpha,n+1}, R_{\alpha,n+2}, \cdots$ are not reassigned anymore. But then the restraint $r_{\alpha,n}$ is updated and protects $X_{e,t}\restriction_{|\sigma|}$ until the next n-gap is opened at s_3, and hence $\sigma \subset X_{e,s_3}$. Suppose that 3(c)(ii) applies at stage t. Thus we return to a previous configuration, and $J^\rho(x_\alpha)[t] \downarrow = k$ where $\rho \subset X_{e,t}$. Clearly $k \leq n$ because again, $R_{\alpha,n+1}, R_{\alpha,n+2}, \cdots$ are not reassigned anymore. Hence $J^\rho(x_\alpha)[s_2] = J^\rho(x_\alpha)[t] = k$ must have been enumerated before stage s_2, and from (I2), we know that $\rho \supset \sigma$. The restraint $r_{\alpha,n}$ is updated at the end of the cycle at stage t, which preserves $X_{e,t} \supset \sigma$ until stage s_3. In either case we have $\sigma \subset X_{e,s_3}$.

Next, we have to show (I2). Note that any such computation has to be set at some stage between t and s_3, where the corresponding use ρ must be the current

approximation of X_e. However after stage t, the segment $X_{e,t}\restriction_{|\sigma|}$ is protected until s_3, so clearly $\rho \supset \sigma$. □

To compute $X_e(p)$, wait until the least stage $t > s_0$ such that $|R_{\alpha,n}[t]| > p$. Then, $R_{\alpha,n}(p)[t] = X_e(p)$. □

Lemma 7.6. *$h_\tau^{C_\tau}$ is total, non-decreasing and unbounded, and for each τ-bottom node α on TP, $h_\tau^{C_\tau}(\lim x_\alpha) \downarrow = \lim region_\alpha$.*

Proof. For each τ-bottom node α on TP, we have x_α and $region_\alpha$ eventually settles at some values x and r respectively. It is clear also that $h_\tau^{C_\tau}(x) \downarrow = r$, because after α sets the $h_\tau^{C_\tau}(x)$-axiom, no other τ-bottom node $\beta \subset \alpha$ can enumerate below $u_\tau(x)$ lest β opens some gap in future, which would cause α to be initialized. Also $\beta \not\supset \alpha$ and $\beta \not>_{left} \alpha$ because otherwise β will set its C_τ-use $u_\tau(x_\beta)$ after α does. Lastly $\beta \neq \alpha$ since x_α has settled.

No inconsistent h_τ-axioms are ever enumerated. $h_\tau^{C_\tau}$ is total because there are infinitely many active τ-bottom nodes along TP, by Lemma 7.4. It is non-decreasing because axioms set under 3(a) are done so with a fresh *region*-value, while in 3(c)(i) and (ii), old axioms are first canceled before new ones replace them. □

Lemma 7.7. *X_e is not hyper jump traceable.*

Proof. Let α be an $\mathcal{N}_{e,i}$-node on TP, and let $x = \lim x_\alpha$ and $r = \lim region_\alpha$. Let $a = \lim attempt_\alpha$, which also has to settle since only finitely many gaps are opened by α. We can easily see that $|T_{i,x}^{C_\tau}| \geq a$ because each time we increase $attempt_\alpha$ after $true(\alpha)$ we must have a new value appearing in $T_{i,x_\alpha}^{C_\tau}$, whose axiom will never be removed after it shows up (unless removed by α itself). Hence if $a = r$, then we are done, so suppose that $a < r$. After α increases $attempt_\alpha$ for the last time it will be put back to state $F_\alpha = 1$. It is clear that $J^{X_e}(x) \downarrow$, since $X_e\restriction_{j_\alpha}$ is forever preserved. It also follows that $J^{X_e}(x) \notin T_{i,x}^{C_\tau}$ since the value a has already been attained by $attempt_\alpha$. □

Many questions regarding the class $\mathcal{H}$ remain open. For instance, it is not known if $\mathcal{H}$ forms an ideal. Since the relation SJT is not a true relativization, one cannot directly relativize the proof that the strongly jump traceable c.e. sets are closed under $\oplus$. A more direct approach is needed, and we expect that the box promotion methods in Cholak, Downey and Greenberg [4] would be useful. It is also not known if every hyper jump traceable set is bounded by a c.e. one, or if there is a c.e. hyper jump traceable set which is low_2 cuppable. In a different vein, we can also ask if $\mathcal{H}$ can be characterized in terms of a randomness-theoretic property. For instance, is there a class of sets $\mathcal{C}$ such that $\mathcal{H}$ is exactly the class of c.e. sets which are below every Martin-Löf member of $\mathcal{C}$?

References

[1] G. Barmpalias, R. Downey, and N. Greenberg. K-trivial degrees and the jump-traceability hierarchy. *Proceedings of the American Mathematical Society*, 137:2099–2109, 2009.

[2] B. Bedregal and A. Nies. Lowness properties of reals and hyper-immunity. *WoLLIC 2003, Electronic Lecture Notes in Theoretical Computer Science*, 84, 2003.

[3] M. Bickford and C. Mills. Lowness properties of r.e. sets. *Theoretical Computer Science.* typewritten unpublished manuscript.

[4] P. Cholak, R. Downey, and N. Greenberg. Strong jump-traceability 1 : The computably enumerable case. *Advances in Mathematics*, 217:2045–2074, 2008.

[5] R. Downey. The sixth lecture on algorithmic randomness. *Logic Colloquium '06*. To appear.

[6] R. Downey, D. Hirschfeldt, A. Nies, and F. Stephan. Trivial reals. *Proceedings of the 7th and 8th Asian Logic Conferences, World Scientific, Singapore*, pages 103–131, 2003.

[7] S. Figueira, A. Nies, and F. Stephan. Lowness properties and approximations of the jump. *Proceedings of the Twelfth Workshop of Logic, Language, Information and Computation (WoLLIC 2005), Electronic Lecture Notes in Theoretical Computer Science*, 143:45–57, 2006.
[8] D. Hirschfeldt, A. Nies, and F. Stephan. Using random sets as oracles. *Journal of the London Mathematical Society*, 75:610–622, 2007.
[9] S. Ishmukhametov. Weak recursive degrees and a problem of Spector. *Recursion theory and complexity (Kazan, 1997)*, 2:81–87, 1997.
[10] B. Kjös-Hanssen, A. Nies, and F. Stephan. Lowness for the class of Schnorr random sets. *Notre Dame Journal of Formal Logic*, 35(3):647–657, 2005.
[11] J. Miller and A. Nies. Randomness and computability: Open questions. *Bull. Symb. Logic*, 12(3):390–410, 2006.
[12] K.M. Ng. On strongly jump traceable reals. *Annals of Pure and Applied Logic*, 154:51–69, 2008.
[13] A. Nies. On a uniformity in degree structures. *Complexity, Logic and Recursion Theory, Lecture Notes in Pure and Applied Mathematics, Feb 1997*, pages 261–276, 1997.
[14] A. Nies. Reals which compute little. *CDMTCS Research Report 202, The University of Auckland*, 2002.
[15] A. Nies. Lowness properties and randomness. *Advances in Mathematics*, 197:274–305, 2005.
[16] A. Nies. *Computability and Randomness*. Oxford University Press, 2008.
[17] T. Slaman and R. Solovay. When oracles do not help. *Fourth Annual Conference on Computational Learning Theory*, pages 379–383, 1971.
[18] R. Soare. Tree arguments in recursion theory and the $\emptyset'''$-priority method. *Proceedings of Symposia in Pure Mathematics*, 42:53–106, 1985.
[19] R. Soare. *Recursively enumerable sets and degrees*. Perspectives in Mathematical Logic. Springer-Verlag, 1987.
[20] R. Solovay. Draft of paper (or series of papers) on Chaitin's work, unpublished notes, 215 pages. 1975.
[21] S. Terwijn and D. Zambella. Algorithmic randomness and lowness. *Journal of Symbolic Logic*, 66:1199–1205, 2001.

University of Wisconsin, 480 Lincoln Drive, Madison, Wisconsin 53706
E-mail address: selwynng@math.wisc.edu