SPLITTING INTO DEGREES WITH LOW COMPUTATIONAL STRENGTH

ROD DOWNEY AND KENG MENG NG

Abstract. We investigate the extent to which a c.e. degree can be split into two smaller c.e. degrees which are computationally weak. In contrast to a result of Bickford and Mills that $\mathbf{0'}$ can be split into two superlow c.e. degrees, we construct a SJT-hard c.e. degree which is not the join of two superlow c.e. degrees. We also prove that every high c.e. degree is the join of two array computable c.e. degrees, and that not every high$_2$ c.e. degree can be split in this way. Finally we extend a result of Downey, Jockusch and Stob by showing that no totally ω-c.a. wtt-degree can be cupped to the complete wtt-degree.

1. Introduction

A classic result in computability theory is the result of Sacks which states that every c.e. set can be split into a pair of disjoint c.e. sets. Moreover, as Sacks also observed, these sets can be made low. Thus the low c.e. degrees generate the c.e. degrees under join[1].

In the last fifty or so years, there have emerged a large number of lowness notions associated with c.e. (as well as other classes of) sets and degrees. Some notable examples include the array computable degrees, the superlow degrees, the K-trivial degrees, the contiguous degrees, the strongly jump traceable degrees, the totally ω-c.a. degrees and a whole infinite hierarchy introduced by Downey and Greenberg [4]. These concepts will, where necessary, be defined later in context.

It is natural to ask the extent to which the global structure of the c.e. degrees relates to these lowness classes. For example, the celebrated work of Nies and others shows that the K-trivial degrees form an ideal ([10, 6]) in the superlow degrees, where A is superlow if $A' \equiv_{tt} \emptyset'$. So, in particular, not every c.e. degree is the join of two K-trivial degrees.

Most of the lowness concepts are concerned with how hard the set is to approximate. Thus A is called *array computable* if for each $f \leq_T A$, f has a uniformly ω-c.a. approximation. That is, there is a computable function h such that for all $f \leq_T A$, there is a computable function g with $f(x) = \lim_s g(x,s)$ where $|\{s : g(x,s+1) \neq g(x,s)\}| < h(x)$. We note

Date: March 31, 2018.

The first author thanks the Marsden fund for support of this research. The second author is partially supported by the grants MOE-RG131/17 and MOE2015-T2-2-055.

[1]All sets and degrees mentioned in this paper are c.e. unless otherwise stated.

here that this is not the original definition of array computability; it is a characterization due to Downey, Jockusch and Stob [7]. In fact, one can easily see that any choice of an order function will do for h, so typically we use $h(n) = n$.

Array computability is the uniform version of being totally ω-c.a., which has a similar definition, except that h can now depend on f. In other words, every function computable from A has an ω-c.a. approximation, hence the name "totally ω-c.a.". The totally ω-c.a. degrees form the first level of a hierarchy based on ordinals due to Downey and Greenberg [4]. Each superlow degree is array computable, and each array computable degree is totally ω-c.a.

Many results for low degrees do transfer to superlow ones. For instance, the low basis theorem which states that every nonempty Π^0_1 class has a low member, can be easily seen to be replaced by the superlow basis theorem (see for example, Downey and Hirschfeldt [5, Section 2.19.3]). Another example relevant to this paper is the result of Bickford and Mills [1] that $\mathbf{0'}$ is the join of two superlow c.e. degrees.

The first question that follows naturally from these results is of course: Is every c.e. degree the join of two superlow degrees? It is reasonable to believe or conjecture that this holds. The proof of Bickford and Mills uses the fact that $\mathbf{0'}$ is the top c.e. degree and can produce enough changes to allow the join of two c.e. sets to be complicated while maintaining that each set is itself close to being computable. This is fundamentally impossible if the degree we want to split is Turing incomplete. We prove the following:

Theorem 3.1. *There are c.e. degrees which cannot be split into the join of two superlow c.e. degrees.*

The reader might be tempted to guess that some assurance of computational power, such as highness, or being close to $\emptyset'$ in some sense, might still allow the given degree to be split in a similar way. We are able to show that this intuition is false. Indeed we show that the constructed degree can be extremely close to $\mathbf{0'}$. More specifically, such sets B can be *SJT-hard*, or even *ultrahigh*. This means that $\emptyset'$ is strongly jump traceable relative to B, as we will define in the relevant later section.

Nevertheless, we can recover part of the Bickford and Mills' result for $\mathbf{0'}$ for the high degrees:

Theorem 2.1. *Every high c.e. degree is the join of two array computable (and low) c.e. degrees.*

One might then be tempted to revise the conjecture to say that *every* c.e. degree is the join of two array computable degrees. Again, we show that this is false. In fact, we prove something stronger:

Theorem 4.1. *There is a high$_2$ c.e. degree* $\mathbf{a}$ *which is not the join of two totally ω-c.a. degrees.*

There are also low degrees where this is true.

The last section briefly explores the notion of "strong cupping"; cupping under the strong reducibility $\leq_{wtt}$. We know that the Turing degree of $\emptyset'$ is the join of two superlow degrees in the Turing degrees, but it is easy to show that this is not true in the wtt-degrees (see [7]). In our section, we prove something stronger. No c.e. set of totally ω-c.a. degree can even be wtt-cupped to the complete wtt-degree.

Theorem 5.1. *No totally ω-c.a. set can be wtt-cupped. That is, if $\emptyset' \leq_{wtt} A \oplus D$ and A is totally ω-c.a., then $\emptyset' \leq_{wtt} D$.*

Again, we remind the reader that everything in this paper is c.e. unless otherwise stated. We freely identify finite strings with the code numbers for them; this allows for functions mapping $\mathbb{N} \mapsto \mathbb{N}$ to have finite strings as outputs. 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.

2. Every high c.e. degree can be split into two array computable c.e. degrees.

In this section, we prove Theorem 2.1, that every high c.e. degree can be split into two low array computable c.e. degrees.

Theorem 2.1. *Every high c.e. degree is the join of two array computable c.e. degrees. Furthermore the two degrees can be made low.*

2.1. **Requirements.** Let C be a high c.e. set. We build two c.e. sets A_0 and A_1, and the Turing reduction Δ such that $C = \Delta^{A_0 \oplus A_1}$. As in convention, we let lowercase Greek alphabets denote use functions. The reduction Δ is built by movable markers, with the usual marker rules. That is, $\delta(n)[s] \downarrow$ iff there is a $\Delta(n)$-computation which currently applies at s. Before a marker is lifted or moved, it has to be first made undefined. A marker $\delta(n)$ which is defined at s will be made undefined if some number $x \leq \delta(n)[s]$ is enumerated into A_0 or A_1.

Recall Martin's famous characterization of the high degrees as those degrees $\mathbf{a}$ where every $\mathbf{a}$-computable function is dominant. In particular, as C is of high degree, the principal function $p(n)$ of $\overline{C}$ is dominant. We use this fact to ensure $A_i \leq_T C$ via high permitting. We make sure that if we enumerate x into A_i at stage s then $p_{s-1}(n) \neq p_s(n)$ for some $n \leq x$; this clearly ensures that $A_i \leq_T C$. Since every c.e. traceable set is array computable, we meet the requirements

$\mathcal{R}_{e,i}$: If $\Phi_e^{A_i}$ is total, then build a c.e. trace $T(x)$ for $\Phi_e^{A_i}(x)$ with at most $h(x)$ many mind changes.

Here h is a computable function to be defined later. We do not arrange for priority amongst the requirements $\mathcal{R}_{e,i}$. Each $\mathcal{R}_{e,i}$ is split into modules $M_{e,i,0}, M_{e,i,1}, \cdots$, with $M_{e,i,k}$ being in charge of tracing $\Phi_e^{A_i}(k)$. We write $M_{\langle e,i,k\rangle}$ instead of $M_{e,i,k}$. We arrange instead for priority amongst the modules, and we order the modules in the order $M_0 < M_1 < \cdots$ (lower numbers have higher priority). The module M_n will disengage the $\delta(n)$-marker before tracing a convergent computation. In order to distinguish between two computations $\Phi_e^{A_i}(x)[s]$ and $\Phi_e^{A_i}(x)[t]$ with different use, but yet having the same output, we will trace in $T(x)$ the use of computations (as finite strings), instead of their output values.

2.2. **Description of strategy.** The key driving force behind many theorems which degree splits $\emptyset'$, is the idea of *disengagement of markers*. For instance, if we wanted to show that $\emptyset' \equiv_T A_0 \oplus A_1$ where A_0 and A_1 are superlow, we would wait for jump computations $J^{A_0}(x)$ to converge, and then enumerate the marker $\delta(x)$ into the other side A_1 to lift the use $\delta(x) >$ use of $J^{A_0}(x)$. (Here we write $J^X(n) := \Phi_n^X(n)$). Only then do we believe the jump computation $J^{A_0}(x)$ and trace it; this ensures that we make computably bounded many mistakes in tracing the jump computations. The completeness of $\emptyset'$ allows us to enumerate $\delta(x)$ whenever we want.

Let us suppose that we wanted to show instead that *every* c.e. degree C was the join of two superlow degrees (which is of course false). In addition to having to disengage markers, we now have to build A_0 and A_1 below C. In particular, before we are allowed to disengage $\delta(x)$ from below the use of some jump computation $J^{A_i}(x)$, we have to wait for a C-change. The reader might imagine that if C was close to the Halting problem, such as being high, then this would be enough. Informally the highness of C allows us to force C to change below almost every challenge we issue it. A plausible plan would then be the following: when we see a jump computation $J^{A_i}(x)$ converge and we need to disengage $\delta(x)$, we will issue a challenge for C to change below $p(\delta(x))$. Since C will permit at almost all of these challenges, hence we will be able to disengage almost every marker we wished, so that almost every convergent jump computation is traced.

The above plan cannot work, since of course not every high c.e. set is the join of superlow sets. The problem is, for instance $J^{A_i}(x)$ converges before $J^{A_i}(y)$ for $x > y$. When we challenge $p(\delta(x))$ to change, we have to simultaneously challenge $p(\delta(y))$ to change. This is because we have to define some total computable function f which p will dominate, so we cannot leave $f(\delta(y)) \uparrow$. The problem is that we are forced to issue a challenge for $p(\delta(y))$ when we were not ready; when $p(\delta(y))$ changes later to witness the domination of p, we are left with a dilemma: if we choose not to lift $\delta(y)$ to a fresh value, and keep $\delta(y)$ the same, then we lose the ability to challenge $p(\delta(y))$ to change again, if $J^{A_i}(y)$ converges later. On the other hand if we always lift $\delta(y)$ to a new value and $J^{A_i}(y)$ never converges, then $\delta(y)$ will be driven to infinity.

However note that the above problem is really due to the fact that we were trying to trace partial functions. If we were only required to trace total functions, then this issue does not arise. We would challenge $p(\delta(y))$ to change only when $\Phi^{A_i}(x)$ has converged for every $x \leq y$. Each module $M_{e,i,x}$ will run the following basic strategy: first it waits for $\Phi_e^{A_i}(z)$ to converge for every $z \leq x$. Then we enumerate a challenge $f_{e,i}(\delta(e,i,x)) \downarrow$ and larger than $p(\delta(e,i,x))$. Then we wait for C to permit, namely for $p(\delta(e,i,x))$ to enter C. When this happens we enumerate $\delta(e,i,x)$ into A_{1-i} to disengage the marker and then trace the computation.

The basic strategy is clear. We now address the technical issues arising from combining the different modules. The only numbers we enumerate into A_0 or A_1 are marker values. There are two reasons why we enumerate $\delta(n)$; the first is due to coding when n has entered C. This happens at most once for each $n = \langle e,i,x\rangle$. The other reason is due to the module M_n when C has permitted on M_n's challenge. Note that C can take as long as it likes before permitting M_n's challenge. In the meantime M_n will have a convergent computation $\Phi_e^{A_i}(x)$ which is waiting for disengagement; we call computations of this type *pending computations*. Each time a pending computation is destroyed, we might get C-permission to disengage $\delta(n)$ before $\Phi_e^{A_i}(x)$ has re-converged. In this case we have to move $\delta(n)$ to a fresh value anyway, because we want to able to allow M_n to issue a new challenge when $\Phi_e^{A_i}(x)$ converges again. Hence we have to ensure that M_n preserves pending computations while it is waiting for disengagement.

There are two reasons why a module M_n wants to limit the changes in A_0 or A_1. The first is for us to count the number of injuries to a traced computation; this is to make A_i c.e. traceable. The second is to ensure that pending computations are destroyed only finitely often; as mentioned above this is to ensure that δ-markers settle. We only care about having an effective bound for injuries of the first type. We do not need, and in fact cannot have an effective bound for injuries of the second type.

During the construction each module acts according to the basic strategy above. A traced computation can be destroyed by any action. A pending computation can only be destroyed by actions of lower priority modules. To determine an upperbound on the number of times a traced computation may be destroyed, we calculate how many times each marker $\delta(n)$ is moved, where $n = \langle e,i,x\rangle$. It is moved once due to coding. Suppose it is moved by M_n. If this action successfully disengages $\delta(n)$ from below a pending M_n-computation, then we get a bound by working inductively. On the other hand when $\delta(n)$ was moved, it might be that $\Phi_e^{A_i}(x) \uparrow$. In this case we can also proceed inductively since pending computations are only destroyed by the actions of higher priority requirements. Lastly when $\delta(n)$ was enumerated it might be that $\Phi_e^{A_i}(x) \downarrow$, but we are not able to disengage $\delta(n)$ because some higher priority requirement $M_{n'}$ has a pending computation with use larger than $\delta(n)$. In this case we enumerate $\delta(n)$ into A_i destroying the convergent $\Phi_e^{A_i}(x) \downarrow$ without tracing it. We then lift $\delta(n)$

above the use of the pending $M_{n'}$-computation. If this computation is never again injured, then $M_{n'}$ will never block M_n from subsequent disengagement attempts. If on the other hand the pending $M_{n'}$-computation is injured it has to be through the movement of one of $\delta(0), \cdots, \delta(n-1)$; and we can use the inductive bounds on these.

2.3. **Notations.** We write $\delta(e,i,k)$ instead of $\delta(\langle e,i,k\rangle)$. Each requirement $\mathcal{R}_{e,i}$ defines a computable function $f_{e,i}$ that serves as a challenge for C-permitting. We also build the trace $T_{e,i}(x)$ for $\Phi_e^{A_i}(x)$. We define $\ell_{e,i}[s] = \max\{y < s \mid (\forall x < y)\Phi_e^{A_i}(x)[s]\downarrow\}$.

A computation is said to be *traced* at a stage s, if it is $\Phi_e^{A_i}(x)[s]\downarrow$ for some e,i,x, and the use of the computation is in $T_{e,i}(x)$. A computation $\Phi_e^{A_i}(x)[s]$ is *pending* at a stage s, if $\ell_{e,i}[s] > x$ for some e,i,x, and $f_{e,i}(\delta(e,i,x))[s]\downarrow$ and it is not yet traced. Basically, pending computations are those convergent computations where a challenge has already been issued for C to change, and we are waiting for the chance to disengage the appropriate δ-marker from below the use. In both cases we also say that $M_{e,i,x}$ has been traced or is pending, for the appropriate $M_{e,i,x}$. The use of a pending $M_{e,i,x}$-computation is simply $\varphi_e^{A_i}(x)[s]$.

At a stage s, we say that $M_{e,i,x}$ *requires attention* if $\ell_{e,i}[s] > x$, $\delta(e,i,x)[s]\downarrow$, $M_{e,i,x}$ is not traced and $f_{e,i}(\delta(e,i,x))[s]\uparrow$. A fresh number at stage s is a number larger than s, and not yet mentioned so far.

2.4. **The construction.** At stage s, the actions are divided into the following phases. Only the modules M_n for $n < s$ are considered.

(1) *challenge phase:* we look for the least module $M_{e,i,x}$ requiring attention (if any). For every $z \leq \delta(e,i,x)[s]$ such that $f_{e,i}(z)$ is not yet defined, we set $f_{e,i}(z)[s]\downarrow$ to a fresh number and larger than $p(\delta(e,i,x))[s]$.
(2) *coding phase:* look for the least n such that $p_{s-1}(n) \neq p_s(n)$ (i.e. $p_{s-1}(n)$ has entered C). We need to enumerate $\delta(p_{s-1}(n))$ into either A_0 or A_1: find the highest priority $M_{e,i,x}$ that is pending, such that $\delta(p_{s-1}(n)) <$ the use of the pending computation. Enumerate $\delta(p_{s-1}(n))$ into A_{1-i}, and into A_0 if $M_{e,i,x}$ does not exist.
(3) *disengagement phase:* let n be as above, i.e. the least such that $p_{s-1}(n)$ has entered C. Find the highest priority $M_{e,i,x}$ such that $\delta(e,i,x)\downarrow \geq n$, $M_{e,i,x}$ is not traced and $f_{e,i}(\delta(e,i,x))[s]\downarrow$. If such $M_{e,i,x}$ exist, we have to enumerate $\delta(e,i,x)$ into either A_0 or A_1; the decision as to which side to enumerate is based on the following: find some $M_{e',i',x'}$ (of the highest priority) which is of priority no lower than $M_{e,i,x}$ such that $M_{e',i',x'}$ is pending, and $\delta(e,i,x) <$ the use of the pending computation. If $M_{e',i',x'}$ exists, enumerate $\delta(e,i,x)$ into $A_{1-i'}$ otherwise enumerate it into A_0.

(4) *tracing phase:* for every module $M_{e,i,x}$ such that $\ell_{e,i}[s] > x$, and either $\delta(e,i,x)\uparrow$ or $\delta(e,i,x)\downarrow > \varphi_e^{A_i}(x)$, we enumerate the use of the computation into $T_{e,i}(x)$.
(5) *extension phase:* find the first m such that $\delta(m)$ is undefined. Give $\delta(m)$ a fresh value and set $C_s(m) = \Delta^{A_0 \oplus A_1}(m)$ with use $2\delta(m)+1$.

2.5. **Verification.** It is obvious that $A_0 \oplus A_1 \leq_T C$ by permitting via $p(n)$, because $\delta(p_{s-1}(n))[s] > p_{s-1}(n) \geq n$ for any n and s. We next prove the crucial lemma.

Lemma 2.2. *There is a computable function $\tilde{h}$ such that the number of times $\delta(n)$ is enumerated into A_0 or A_1 is at most $\tilde{h}(n)$.*

Proof. We proceed by induction on $n = \langle e,i,x\rangle$, and define $\tilde{h}$ inductively. Let $\tilde{h}(0) = 1+1 = 2$; it is not hard to see (by following a similar argument as the inductive step below) that this bound is sufficient. This is because $\delta(0)$ is enumerated under the coding phase at most once, and enumerated under case (i) or (ii) below at most once. Case (iii) is not possible.

Now consider $n = \langle e,i,x\rangle > 0$, and suppose that $H = 1 + \tilde{h}(0) + \cdots + \tilde{h}(n-1)$ have all been defined. $\delta(n)$ can be enumerated under the coding phase at most once. Suppose $\delta(n)$ is enumerated under the disengagement phase, at stage s. At s one of the following three scenarios must hold:

(i): there is some $M_{n'}$ of highest priority $n' \leq n$ which is pending at s, and $\delta(n) <$ use of the $M_{n'}$-pending computation.
(ii): not (i) and $\ell_{e,i}[s] > x$.
(iii): otherwise.

If we enumerate $\delta(n)$ under scenario (i) and some n', then in order for us to enumerate $\delta(n)$ again under (i) and the same n', we must have one of $\delta(0),\cdots,\delta(n-1)$ being enumerated. Hence the total number of times we enumerate $\delta(n)$ due to (i) is at most $(n+1)H$. Note that $\ell_{e,i}[s]$ might not be larger than x. Suppose we enumerate $\delta(n)$ under scenario (ii). Then $M_{e,i,x}$ is pending at s since $\ell_{e,i}[s] > x$, so the enumeration of $\delta(n)$ does not kill the pending $M_{e,i,x}$-computation, because of the failure of (i). Hence we would trace $\Phi_e^{A_i}(x)[s]$ under the tracing phase at s , and we never enumerate $\delta(n)$ under (ii) again unless one of $\delta(0),\cdots,\delta(n-1)$ is enumerated. The total number of times we enumerate under scenario (ii) is at most H. Consider scenario (iii); we claim that this is not possible.

Let t_1 be a stage where we enumerate $\delta(n)$ under (iii), and we want to get a contradiction. At t_1, $f_{e,i}(\delta(e,i,x))[t_1]$ is defined. Since uses are always chosen fresh, we let $t_0 < t_1$ be the stage where $f_{e,i}(d)$ is defined, where $d = \delta(e,i,x)[t_1]$. The action of defining $f_{e,i}(d)$ immediately makes $M_{e,i,x}$ pending. This pending computation has to be destroyed, say at t' between t_0 and t_1, in order for (i) and (ii) to fail at t_1. At t' there has to be a pending $M_{n'}$-computation of strictly higher priority $n' < n$ such that $\delta(n) <$ the use of the pending $M_{n'}$-computation. Otherwise M_n is the highest priority module being considered at t', and the action will not be

allowed to injure M_n. We let N be the module of the highest priority with a pending computation with use larger than d between t' and t_1. Clearly $N \geq n' > n$. Before t_1, no marker $\leq d$ can be enumerated, lest $\delta(n)$ is moved. Hence at t_1, M_N cannot be traced, $f_{b_0,b_1}(\delta(N))$ is defined where $N = \langle b_0, b_1, b_2\rangle$, and furthermore $\ell_{b_0}^{A_{b_1}}[t_1] > b_2$ also holds because M_N is the highest priority module with the above-mentioned properties (hence cannot be injured). Hence at t_1, M_N is pending and in fact will still have pending computation with use larger than d. This shows that (i) must hold at t_1. The resulting contradiction shows that stage t_1 does not exist, and (iii) cannot be possible. Totalling up the figures obtained from (i) and (ii), we can safely define $\tilde{h}(n) = 1 + (n+2)H$. □

Therefore each $\delta(n)$ eventually becomes defined and settles. Hence $\Delta^{A_0 \oplus A_1}$ is total, and equals C.

Lemma 2.3. *Both A_0 and A_1 are low.*

Proof. Let $Q(z,t) = 1$ if $\ell_{e,i}[t] > x$, where $z = \langle e, i, x\rangle$, and let $Q(z,t) = 0$ otherwise. As usual we assume the hat trick. Suppose for every $z' < z$, both $\delta(z')$ and $Q(z',t)$ have settled. Also assume that no number less than any convergent $\Phi_{e'}^{A_{i'}}(x')$ is ever enumerated (in either side) where $\langle e', i', x'\rangle < z$. Let $z = \langle e, i, x\rangle$. Suppose $\ell_{e,i}[t] > x$ at infinitely many stages t. If $M_{e,i,x}$ is ever traced then the traced computation would be preserved forever (since $\delta(0), \cdots, \delta(z-1)$ have settled). Hence $Q(z) = 1$ forever. Suppose $M_{e,i,x}$ is never traced. Then $M_{e,i,x}$ will be pending eventually at one of these stages t. It is simple to verify that A_i cannot change below the use of the pending $M_{e,i,x}$-computation at t ever again. Hence $Q(z)$ settles.

The fact that A_0 and A_1 are low follows from the fact that $Q(\langle g(e), i, g(e)\rangle)$ settles, where we define $\Phi_{g(e)}^{X}$ to be total if $e \in X'$ and is empty if $e \notin X'$. □

Again we draw the reader's attention to the following fact. We are only interested in getting an effective bound for the number of times a traced computation can be destroyed. We do not need to compute a bound for the number of times a *pending computation* can be destroyed; indeed there cannot be an effective bound for the number of changes in $Q(z)$, otherwise A_0 and A_1 would be superlow. Observe that $Q(z)$ can change from 1 to 0 because some higher priority pending computation is blocking a coding attempt; the number of times this happens, though finite, will depend on the use of higher priority pending computations which we cannot expect to know beforehand.

We now prove that A_0 and A_1 are both c.e. traceable. It is obvious that $|T_{e,i}(x)| \leq h(e,i,x)$ where $h(0) = 1$ and $h(n+1) = 1 + \tilde{h}(0) + \cdots + \tilde{h}(n)$. Finally suppose that $\lim_s \ell_{e,i}[s] = \infty$. We suppose there are infinitely many x such that $\Phi_e^{A_i}(x) \notin T_{e,i}(x)$ (more precisely the use). Fix one such x, and let $d = \lim \delta(e,i,x)$. Suppose d was picked under the extension phase at some stage s. Clearly $M_{e,i,x}$ is not traced at s because $\delta(e,i,x)$ has settled

down at s. In fact $M_{e,i,x}$ can never be traced at any stage after s. At s we picked d fresh so $f_{e,i}(d)[s] \uparrow$. It is not hard to see that $M_{e,i,x}$ has to receive attention under the challenge phase at some time t after s, where we will define $f_{e,i}(d) > p_t(d)$. We claim that none of $p_t(0), \cdots, p_t(d)$ gets enumerated into C after t: if one of them enters C then we will enumerate d (or some number smaller than d) into A_0 or A_1, violating the fact that $\delta(e, i, x)$ is stable. This means that infinitely often $f_{e,i}(d)$ is defined and is $> p(d)$, which means that $f_{e,i}$ is total computable and witnesses the fact that p fails to be dominant. This contradiction shows that almost all $\Phi_e^{A_i}(x)$ have to be traced.

3. An ultrahigh c.e. degree which cannot be split into two superlow c.e. degrees

We show that not every c.e. degree can be split into two superlow ones. By Sacks splitting theorem, every c.e. set can be split (as sets) into two disjoint low sets. On the other hand, as we mentioned earlier, Bickford and Mills [1] used disengagement of markers to prove that the complete c.e. Turing degree can be split into two superlow degrees.

Notice how, with this result, the fact that $\emptyset'$ allows us to move all markers is what enables us to force superlowness of the A_i using additional coding; this is weakly reflected in the previous section where the highness allowed for the movement of almost all markers, *but* with a weaker result of array computability rather than superlowness. Array computability is concerned with total functions which works well with highness. In both proofs, the problem is that in general we have no idea whether smaller codings will happen killing computations we like to preserve. This is what we exploit in the next proofs.

We give an example of an incomplete c.e. degree which cannot be split in such a way; in fact we can make the example very close to $\emptyset'$.

We show that Sacks' splitting cannot be achieved for superlow sets; in fact we cannot even do a degree split. It is easy to see that this counterexample can be made low, and hence gives another proof that the low c.e. sets and the superlow c.e. sets are not elementarily equivalent.

In the "highness" hierarchy, there are various notions of being "very high". For example, a set A is called *superhigh* if $A' \equiv_{tt} \emptyset''$. Recall that A being superlow is equivalent to having each partial A-computable function f being traced with a small number of possibilities. That is, there is an order h and a weak array $W_{g(x)}$ of finite sets with $|W_{g(x)}| < h(x)$ and $f(x) \in W_{g(x)}$, for all x. A is called *strongly jump traceable* if for *any* order h, and any partial A-computable function f, there is a collection $W_{g(x)}$ of finite sets with $|W_{g(x)}| < h(x)$ and $f(x) \in W_{g(x)}$, for almost all x.

In the same way that there are stronger lowness notions than being superlow, there are also stronger highness notions than being superhigh. Being superhigh means that for some choice of an order function h, every partial

function f relative to $\emptyset'$ can be traced relative to A by a family $W^A_{g(x)}$ of finite sets with $|W^A_{g(x)}| < h(x)$ and $f(x) \in W_{g(x)}$, for almost all x. We can call A *SJT-hard* if the above definition works for any order h and A is *ultrahigh* if it works for any A-computable order h^A. That is, for any partial f relative to $\emptyset'$ and any order h^A, there is a family $W^A_{g(x)}$ of finite sets with $|W^A_{g(x)}| < h(x)$ and $f(x) \in W_{g(x)}$, for almost all x.

SJT-hard sets were used by Downey and Greenberg to give an example of a pseudo-jump operator which could not avoid an upper cone. The c.e. SJT-hard sets form a collection with a non-zero degree below all of them. (Downey and Greenberg [3].) The ultrahigh degrees were introduced by Ng in his thesis [12] and the paper [11]. Thus, SJT-hard (and ultrahigh) sets resemble $\emptyset'$ so much that they cannot avoid an upper cone.

Recall that $\mathbf{0'}$ can be split into two superlow c.e. degrees. We construct an ultrahigh c.e. set which cannot be split in this way; this shows that splitting into superlow degrees cannot be achieved even if we consider sets which resemble $\emptyset'$ very closely.

Theorem 3.1. *There is an ultrahigh c.e. degree which cannot be split into two superlow c.e. degrees.*

3.1. **Requirements.** We build a c.e. set A satisfying the following requirements

$$\begin{array}{rcl} \mathcal{N}_e & : & \text{If } A = \Phi_e^{W_e \oplus V_e} \text{ and } W_e \oplus V_e = \Delta_e^A, \text{ then one of } W_e \text{ or } V_e \\ & & \text{is not superlow.} \\ \mathcal{P}_e & : & \text{If } \Phi_e^A \text{ is an order, make } \emptyset' \text{ } A\text{-jump traceable via } \Phi_e^A, \end{array}$$

Here, we let $\langle \Phi_e, \Delta_e \rangle$ denote the e^{th} pair of Turing reductions, and $J^{\emptyset'}(k)$ denote the value of the universal $\emptyset'$-partial recursive function $\{k\}^{\emptyset'}(k)$. $\langle W_e, V_e \rangle$ is the e^{th} pair of c.e. sets. Both kinds of requirements above will be further divided into subrequirements; we will describe how to do this in the following section.

3.2. **Making A not the join of superlow sets.** Since the superlow c.e. sets are exactly the jump traceable c.e. sets, we will diagonalize against all the possible traces for the jump $J^{\emptyset'}$. The requirement $\mathcal{N}_e$ is divided into subrequirements $\mathcal{N}_{e,i,j}$, and $\mathcal{N}_e$ is satisfied in a non-uniform manner depending on the outcomes of the individual subrequirements (i.e. we ensure that one of W_e or V_e is not superlow, but we do not know which). Basically each $\mathcal{N}_{e,i,j}$ aims to make *either* J^{W_e} not traced by T_i, *or* it makes J^{V_e} not traced by T_j. Here, we let $T_0, T_1, \cdots$ be an effective list of all possible c.e. traces. That is, T_e is a uniform sequence of c.e. sets $T_e(0), T_e(1), \cdots$. Also associated with each sequence T_e is a partial computable function t_e of a single variable. We say that W is jump traceable via T_e, if for all x, $|T_e(x)| < t_e(x)$ and whenever $J^W(x) \downarrow$ implies that $J^W(x) \in T_e(x)$. Every superlow c.e. set will be jump traceable via some T_e.

If every $\mathcal{N}_{e,i,j}$ is satisfied, then clearly $\mathcal{N}_e$ itself will be satisfied: if $\forall i \exists j$ such that $\mathcal{N}_{e,i,j}$ succeeds in the first alternative, then W_e is not superlow. Otherwise V_e will not be superlow. We first describe the $\mathcal{N}_{e,i,j}$-strategy in isolation, then we will describe the ultrahigh strategy, and finally we will see how to put both strategies together. We see the $\mathcal{N}_{e,i,j}$-strategy as having primarily a negative role blocking elements from entering A. However, like many other strategies which tries force every set in a constructed Turing degree avoid a certain property, there is a certain amount of finite positive action. Examples include constructing a Turing degree free of semi- or hemi-maximal sets, and a d.c.e. degree which is not of c.e. degree.

Consider an arbitrary $\mathcal{N}$-requirement (we drop all indices, since we are considering it in isolation). Suppose $A \equiv_T W \oplus V$ via the reductions Φ, Δ. The basic plan to make either J^W not traced by T or J^V not traced by T is the following. By the Recursion Theorem, we control parts of the jump. To wit, we are supplied with an infinite list of indices x for which we are able to build the x^{th} Turing functional $J^X(x)$. We pick a number of indices $\eta_0, \eta_1, \cdots, \eta_m$ and we will use $J^X(\eta_i)$ to diagonalize against the trace $T(\eta_i)$. If we were merely trying to make A itself not superlow (this is weaker than $\mathcal{N}$), we could do the following. We wait for $t(\eta_0) \downarrow$ and then start the following cycle. Set $J^A(\eta_0)[s] \downarrow = s$ and wait for s to be enumerated in $T(\eta_0)$. When s enters $T(\eta_0)$ (at $t > s$) we change A to make the previous axioms invalid, and then set $J^A(\eta_0)[t] \downarrow = t$ and repeat. This cycle repeats at most $t(\eta_0)$ times, so we only need to change A (for the sake of a single requirement) finitely often.

However we need to make A not of the same Turing degree as the join of superlow sets. We need to simultaneously run two of above strategy (one for W and one for V). Each change in A will make progress towards *one of* the two strategies. Hence we will just need to repeat the above cycle more times; in this case we need to freeze A in between cycles, so that any progress towards either side does not become undone. Specifically, the modified strategy is:

(1) pick a number of followers (targeted for entry into A) $x_1, \cdots, x_N$ such that $x_{i+1} > \delta(\varphi(x_i))$ for all i. These followers will be enumerated in decreasing order of magnitude. Freeze A below all these use.
(2) define $J^W(\eta_0) \downarrow$ and $J^V(\eta_1) \downarrow$ both with use $\varphi(x_N)$. Wait for these values to enter $T(\eta_0)$ and $T(\eta_1)$ respectively.
(3) enumerate x_N into A. Then one of W or V has to change below $\varphi(x_N)$. If we get a W-change, repeat steps (2)-(3) with the next index $J^V(\eta_2)$ and next follower x_{N-1}. If we get a V-change, repeat steps (2)-(3) with the same indices $J^W(\eta_0)$ and $J^V(\eta_1)$, and next follower x_{N-1}.

If we have many consecutive V-changes without a W-change, then we would have made $J^V(\eta_i) \notin T(\eta_i)$ for some i. If on the other hand we are interrupted with a W-change at each index $J^V(\eta_1), \cdots J^V(\eta_m)$, then we have

made $J^W(\eta_0) \notin T(\eta_0)$. It is important that once we start on the diagonalization, we freeze A below the uses, and also we enumerate the followers $x_N, \cdots, x_1$ in decreasing order of magnitude; this is to ensure that we keep the W-use of $J^W(\eta_0)$ above $\varphi(x)$ for any follower x not yet in A.

We draw the reader's attention to the following fact. Step 1 consists of many individual actions. Namely we first pick x_0 and wait for $\delta(\varphi(x_0))\downarrow$, before selecting x_1, and so on. We could have increased the A-restraint to $\delta(\varphi(x_0))$ the moment we see it converge, so that we never need to re-select x_1. However, for technical reasons which will be explained later, we will not do this. Observe that we only need to increase A-restraint once all of the x_i have been picked. For instance if A changes below $\delta(\varphi(x_0))$ while we were waiting for $\delta(\varphi(x_1))$ to converge, then we would pick a new value for x_1 above the new use $\delta(\varphi(x_0))$. If infinitely often we do this, then $\delta(\varphi(x_0))$ would $\to \infty$, and we would not need to act for $\mathcal{N}$ after all.

It is easy to see that we can combine all of the different $\mathcal{N}$-requirements in a finite injury argument. These requirements guarantee that A is of intermediate Turing degree ($\mathbf{0} < deg_T(A) < \mathbf{0}'$). It is easy to see that we can throw in lowness requirements to make A low, but clearly not superlow (because we do not know a forti the number of injuries). This strategy also admits variations; for instance it is not hard to see that one can make A not of the same Turing degree as the join of two sets W, V where W is low and V is superlow. This will involve considering all the possible lowness indices, and a straightforward modification of the above strategy together with a standard infinite injury argument will settle the issues.

3.3. **Making A ultrahigh.** The construction takes place on a tree, which grows downwards. We order the nodes lexicographically. We use $\supset$ for string extension. We let $\alpha <_L \beta$ denote that α is strictly to the left of β. If $\mathcal{R}$ is a requirement, we say that α is an $\mathcal{R}$-node, if α is assigned the requirement $\mathcal{R}$. A negative node is a $\mathcal{N}_{e,i,j}$-node for some e, i, j, while an ultrahigh node is a $\mathcal{P}_{e,k}$-node for some e, k. A top node or a mother node is a $\mathcal{P}_e$-node for some e. We call the $\mathcal{N}$-nodes negative, even though their actions are not solely to prevent numbers from entering A; they do themselves enumerate numbers into A (though only finitely often before being injured).

We now describe the basic strategy to make A ultrahigh; see Ng [11] for more details. If Φ_e^A is an order, the requirement $\mathcal{P}_e$ will build an A-u.r.e. sequence $\{V_k^A\}_{k\in\mathbb{N}}$ such that for all k, $|V_k^A| \leq \Phi_e^A(k)$ and $J^{\emptyset'}(k) \in V_k^A$. To do this, $\mathcal{P}_e^A$ will divide the task into infinitely many substrategies, or modules. The k^{th} module will be responsible for building V_k^A. To build V_k^A, we define a functional $\Psi^A(k,n)$ for $n < \Phi_e^A(k)$, and let $V_k^A = \cup_n \Psi^A(k,n)$. Suppose α is an ultrahigh node assigned the k^{th}-module. If $J^{\emptyset'}(k)\downarrow$, then α has to ensure that $\Psi^A(k,n) = J^{\emptyset'}(k)$ for some n. Even though $J^{\emptyset'}(k)\downarrow$, it might be that the computation $J^{\emptyset'}(k)[s]$ converges to many different values before settling on the final correct value, and we have absolutely no control over when this happens. As with making A high, we have to allow for

opportunities to redefine $\Psi^A(k,n)$ for various n. As is customary we now let α have two outcomes; ∞ to the left of f, which guesses respectively that there are infinitely (and finitely) many different $J^{\emptyset'}(k)[s]$-values observed during the α-stages.

During the construction, we decide the α-outcome based on the following rules. If $J^{\emptyset'}(k) \uparrow [s]$ at an α-stage s, or $J^{\emptyset'}(k) \downarrow [s] \neq J^{\emptyset'}(k)[s-1]$, then we play the outcome ∞ to record the belief that either $J^{\emptyset'}(k) \uparrow$ or α had previously traced a wrong value. At such stages α will make $\Psi^A(k,n)$ undefined (for some previously defined n) by changing A below the ψ-use. If we observe that $J^{\emptyset'}(k)[s]$ has converged and there had been no change below the $\emptyset'$-use for a sufficiently long time, then we will play the α-outcome f. In this case, we believe that the current convergent $J^{\emptyset'}(k)[s]$ is correct, and we make α define $\Psi^A(k,n)$ on some large n (if none of the other ns record the correct value) with some fresh A-use. Clearly if ∞ is the true outcome of α, then infinitely many ψ-uses will be enumerated into A; negative requirements extending $\alpha^\frown\infty$ will never believe any A-computation until all the relevant ψ-uses have been enumerated by α. That is, negative nodes always wait for a *believable computation* before proceeding.

We remind the reader on how to make the set A superhigh. One method is to construct a binary functional $\Gamma^A(k,x)$ such that for all $k \in \omega$, $Tot(k) = \lim_{x\to\infty} \Gamma^A(k,x)$ with $|\{x : \Gamma^A(k,x) \neq \Gamma^A(k,x+1)\}|$ bounded by some computable function. Then $Tot \leq_{tt} A'$ by employing the relativized version of Shoenfield's Limit Lemma in the tt-case. We remark that constructions of superhigh sets in this way, combined with some form of negative requirements restraining elements from entering the set, will usually end up with an exponential bound (usually like $h = 2^{2k}$). Since every superhigh c.e. set A is also $\emptyset'$-jump traceable, it follows that we can also perform the above construction in terms of the tracing of $J^{\emptyset'}$. Instead of defining the sequence $\Gamma^A(k,x)$, we can build a trace $\Psi^A(k,x)$ for $J^{\emptyset'}(k)$, where $x < 2^{2k}$. There will usually be 2^{2k} many highness nodes at level k in charge of tracing $J^{\emptyset'}(k)$; each of the nodes on this level will be in charge of a different location $\Gamma^A(k,x)$.

The situation in our case is a little bit different, for we have to do much better than just making A superhigh. The trouble is that at the level of α, we are only allowed to define $\Psi^A(k,n)$ at $\Phi_e^A(k)$ many values of n. The function $\Phi_e^A(k)$ may grow quite slowly (almost certainly will be sub-exponential); in this case, each module will have very few chances to make mistakes when tracing $J^{\emptyset'}(k)$. The actions of a higher priority $\mathcal{N}$-strategy may force α to make a mistake. In particular, if the $\mathcal{N}$-strategy is already holding some restraint on A, then this restraint must be obeyed by α. Therefore the number of higher priority $\mathcal{N}$-requirements we allow above a particular module will be determined by Φ_e^A. In particular, if $\Phi_e^A(k) < 2^n$, then there cannot be more than n many such $\mathcal{N}$-strategies.

To achieve this, we arrange for several modules to be grouped into a *block*. The b^{th} block is denoted by $\mathcal{B}_b$. The contents of $\mathcal{B}_b$ will change from time to time, during the construction, and depends on $\Phi_e^A[s]$. The subrequirement $\mathcal{P}_{e,b}$ is assigned the block $\mathcal{B}_b$. We arrange for exactly n many $\mathcal{N}$-strategies before block $\mathcal{B}_n$. We say that α is a $\mathcal{B}$-node, if it is assigned the block $\mathcal{B}$. We explain why such an arrangement of blocks will work. We claim that if $k \in \mathcal{B}_1$, then either $\Psi^A(k,0)$ or $\Psi^A(k,1)$ will contain the correct $J^{\emptyset'}(k)$ (if convergent) value. Suppose this block is operating at level k', and $\beta_0, \cdots, \beta_M$ are all the negative nodes (on the same level) placed above $\mathcal{B}_1$, and β_0 is the leftmost node accessed infinitely often during the construction. We will need to coordinate the actions amongst all the nodes at level k'. As mentioned above, in a usual high or superhigh construction, even though all of these highness nodes are on the same level, and working for the same (sub)-requirement, they will make separate attempts at tracing $J^{\emptyset'}(k)$ by defining and undefining $\Psi^A(k,r)$ for distinct r's. This clearly cannot work here. Due to the arrangement of the blocks we may have much more than just 2 nodes at level k', but being in $\mathcal{B}_1$ we are only allowed to define $\Psi^A(k,0)$ and $\Psi^A(k,1)$. Therefore, all nodes at level k' will have to be allowed access to the definition of $\Psi^A(k,0)$ and $\Psi^A(k,1)$. The heuristics is that in a superhigh construction, we only care about having a computable bound. In an ultrahigh construction, we need to *save on the trace locations*.

At a level devoted to a negative requirement, we need to ensure that we have only one active restraint at a time for the entire level. More specifically, once some β_m puts up a restraint to protect a disagreement, all the blocks $\mathcal{B}_1, \mathcal{B}_2, \cdots$ will have to respect this restraint. If $J^{\emptyset'}(k)$ later changes such that $\Psi^A(k,0)$ is wrong, then we will have to use $\Psi^A(k,1)$ to record the new $J^{\emptyset'}(k)$-value. Note that as long as the restraint on β_m remains in force, and does not increase, we will always be able to correct $\Psi^A(k,1)$ if we need to, since this ψ-use is always larger than the restraint. To make sure that each negative level only imposes a single restraint at a time, we need more coordination amongst the negative nodes $\beta_0, \cdots, \beta_M$ on the same level. Once some negative node β_m puts up a restraint, then every β_i to the right of β_m will also hold the same restraint as β_m, and does not put up any new restraint of its own. Consequently every β_i to the right will run the basic β_m-strategy, and enumerate the β_m-followers (instead of its own followers). This happens even though we might be currently on the right of β_m. We call this *acting on β_m's behalf*. This remains true until we visit left of β_m, in which case a new restraint may be put up (by some β to the left of β_m). Thus if β_0 is the leftmost node visited infinitely often, and if β_0 ever puts up any restraint, then this restraint will be in force forever, so that $\Psi^A(k,1)$ will record the correct $J^{\emptyset'}(k)$-value. If β_0 never gets to put up a restraint, then $\Psi^A(k,0)$ will record the correct $J^{\emptyset'}(k)$-value. Each time β_0 is visited we will clear $\Psi^A(k,1)$ of any value it is currently recording. This is

the reason why in the atomic strategy for $\mathcal{N}$, we only put up an A-restraint when all the followers have been picked.

The above extends to the n^{th} block as well, and generally we need $\Psi^A(k,0)$, $\cdots, \Psi^A(k,2^n-1)$ for every k in $\mathcal{B}_n$, since we need to record all possible restraint-states that the higher priority negative requirements are in. Note that due to the arrangement of the blocks, generally the n^{th} negative requirement will be placed at a level much larger than $2n$, and so there can be much more than 2^{2n} many different versions of the negative requirement. Hence the number of mistakes in which a module in $\mathcal{B}_n$ will make does not depend on the level at which it is operating at (as is the case in a high or superhigh construction), but rather on the *number of levels of negative requirements* which are placed before it.

3.4. **Technical considerations on the tree.** In this section we describe the technical aspects when combining the ultrahigh strategies with the negative strategies on the tree. Suppose β_0 is on the true path, and β_i to the right of β_0 is acting on behalf of β_0. Because β_0 and β_i are at different positions on the construction tree, we have to consider issues which will arise from enumerating β_0-followers when we are at the right of β_0. The first issue is the difference in believable computations. Remember that β_0 only puts up a restraint if all the β_0-followers have seen computations which are β_0-believable. In future when β_i enumerates these followers on behalf of β_0, we cannot expect that upon recovery, these followers will have β_0-believable computations. However note that the only reason why we enumerate any β_0-follower is to produce $W \oplus V$-changes. Hence even if the recovered computations were not β_0-believable (and may never become β_0-believable), it does no damage to β_0's strategy, since we would have obtained our desired $W \oplus V$-change.

The above issue arises because we had to allow more than one node on some level l devoted to some block $\mathcal{B}_n$ access the same trace location $\Psi^A(k,m)$ for some $m < 2^n$. This creates another curious situation when considering the interaction between two such levels $l < l'$. This is best illustrated by an example: take a node α on level l, and $\beta_0 \supset \alpha^\frown\infty$ and $\beta_1 \supset \alpha^\frown f$ both be on level l'. At some stage s, we might visit β_0, which decides to play outcome f. We will then make a trace $\Psi^A(k',m')$ for some m'. Later on, we might then visit α, where we also make a trace $\Psi^A(k,m)$. Note that the use $\psi(k,m) > \psi(k',m')$ since it was traced later. If finally β_1 is visited and sees a new $J^{\emptyset'}(k')$-value, it will (and indeed must) record this new value at the same trace location $\Psi^A(k',m')$, because we really do not want to use a new valuable trace location simply because we were blocked by another ultrahigh strategy. Hence, β_1 would have to clear $\Psi^A(k',m')$ by enumerating the use, which in turn clears the location $\Psi^A(k,m)$ as well. This creates the situation where a lower priority requirement β_1 takes an action which "injures" a higher priority requirement α. Once again we compare this with the case of a high or superhigh construction; this situation

described does not happen there because at stage t when β_1 decides to trace the new $J^{\emptyset'}(k')$-value, it will simply use another trace location to record this value (instead of clearing the old one).

We note that this situation is easily resolved by the tree mechanism, and in fact poses no real problem at all. The above situation of β_1 injuring α may repeat infinitely often, but *only if α has infinitely many expansionary stages*. In this case, α gets its trace cleared infinitely often by β_1, but it would not matter since $J^{\emptyset'}(k)\uparrow$. On the other hand if $J^{\emptyset'}(k)\downarrow$, then there will only be finitely many α-expansionary stages. Eventually at one of the $\alpha^\frown f$-stages, we would make a $\Psi^A(k,m)$ definition before any $\Psi^A(k',m')$ definition is made, after which the action of β_1 will no longer affect α.

Such arrangements might also affect nodes of higher priority, but which are further down the construction tree. Suppose now we consider α_0 to the left of α_1, which are both on the same level l and working for some block $\mathcal{B}_n$. Suppose we visit $\alpha_0{}^\frown f$ and we make a trace $\Psi^A(k,m)$. If later on we visit α_1 but we see a new $J^{\emptyset'}(k)$-value, we will need to clear the same trace $\Psi^A(k,m)$ with a use set by α_0 earlier. This will cause nodes extending $\alpha_0{}^\frown f$ to be injured; again this is due to a lower priority α_1. In particular, this might cause some negative node working on level $l' > l$ to be initialized. In this case, when we next visit α_0, we will allow $\alpha_0{}^\frown\infty$ to be visited once (regardless of the current $J^{\emptyset'}(k)$-status). This gives the version of the negative node on level l' extending $\alpha^\frown\infty$ a chance to act.

We describe another situation where nodes on the right are allowed to injure nodes to the left. Suppose β is a negative node to the right of another negative node $\tilde{\beta}$ on the same level, and β was allowed to act on behalf of $\tilde{\beta}$. Because $\tilde{\beta}$-followers were picked very early, so this action will injure nodes of every kind which are of lower $\tilde{\beta}$-priority. This can include nodes which are to the left of β, and even β itself. However this only happens finitely often to nodes on the true path (once all historical followers have been enumerated, we are safe).

An alternative presentation of the proof would have been the use of a pinball machine; indeed a pinball machine eliminates the above situation where lower priority nodes injure higher priority ones. However there are advantages of the tree method over the pinball machine in this case. This will be made clear in the following section. Basically the tree can handle the dynamic variations in block sizes easily, while in a pinball machine we would have to dynamically generate or modify the pinball machine when the distance between the gates (negative requirements) increases.

3.5. **Tree layout and notations.** So far we have only dealt with a single fixed order Φ_e^A. This is too simple for two reasons: firstly in the actual construction, we have to deal with infinitely many of these orders. Secondly, the values of Φ_e^A can only be approximated during the construction; however A will change during the construction. These changes will occur during the construction due to the other requirements, and we cannot hope to block

changes in A to make Φ_e^A a computable order. Therefore it is possible that blocks get shuffled around; we have to constantly adjust the size of blocks.

It will be difficult to dynamically change the assignment of requirements to the nodes extending τ. Instead, we will choose a presentation which fixes the labels on the construction tree, albeit at the cost of working with an ω-branching construction tree. To wit, the entire block $\mathcal{B}_n$ will be assigned to a single node α. Suppose that $k_0, \cdots, k_m$ are the modules in the same block. Then α will be responsible for tracing all of $J^{\emptyset'}(k_i)$ for $i \leq m$. We need to equip α with many outcomes; in fact we need outcomes which describes whether or not $J^{\emptyset'}(k_0), \cdots, J^{\emptyset'}(k_m)$ are currently believed to be convergent. Thus, the outcomes of α will be finite strings of length $m+1$ with word $\{\infty, f\}$. We want the outcome $x_0 \cdots x_m$ to be the true outcome of α if and only if for every $i \leq m$, $J^{\emptyset'}(k_i) \downarrow \Leftrightarrow x_i = f$. Suppose that $J^{\emptyset'}(k_0) \uparrow$ and $J^{\emptyset'}(k_1) \uparrow$. Then, we want to arrange for $\infty\infty$ to be played infinitely often. If we simply play the natural outcome which codes the current state of convergence of $J^{\emptyset'}(k_0)[s], \cdots, J^{\emptyset'}(k_m)[s]$, then the trouble will be that $J^{\emptyset'}(k_0)$ and $J^{\emptyset'}(k_1)$ takes turns to show us a new value. Then, we will cycle through the outcomes $f\infty$ and ∞f, when in fact we really want to record this as the outcome $\infty\infty$.

In order to record the outcomes correctly, at each α-stage, we will play the outcome in the following way. We want to make it as difficult as possible to output outcome f for any of the k_i's. After all if it is truly the case that $J^{\emptyset'}(k_i) \downarrow$, we can afford to delay for as long as we want before playing an outcome f for k_i (and hence begin the tracing of the correct $J^{\emptyset'}(k_i)$). Therefore at an α-stage s, we only output outcome f for k_i if $J^{\emptyset'}(k_i) \downarrow [s]$ has been stable for a long time (since some stage t), and for every other k_j such that $J^{\emptyset'}(k_j)[t] \downarrow$ and is different from $J^{\emptyset'}(k_j)[s]$, we would have output ∞ for all of these k_j's before s. It is clear that this prevents the situation above where we cycle through $f\infty$ and ∞f, and furthermore if $J^{\emptyset'}(k_i)$ truly does converge, then we will eventually always output outcome f for k_i.

We now define the priority tree. If $|\alpha| = 2\langle e, i, j\rangle$ we assign the requirement $\mathcal{N}_{e,i,j}$ to it. There are two outcomes for α, w to the left of d. The main requirement $\mathcal{N}_e$ is satisfied via the actions of $\mathcal{N}_{e,i,j}$, and we do not require its presence on the tree. Nodes α of length $2\langle e, k\rangle + 1$ are assigned requirement $\mathcal{P}_e$ if $k = 0$, and to the subrequirement $\mathcal{P}_{e,k}$ if $k > 0$. $\mathcal{P}_e$-nodes are called *mother nodes*, with two outcomes $\infty <_L f$. If α is an $\mathcal{P}_{e,k}$-node and $\tau \subset \alpha$ is a $\mathcal{P}_e$-node, we say that τ is the *mother node of* α, and we denote $\tau = \tau(\alpha)$. If $\alpha <_L \beta$ are both $\mathcal{P}_{e,k}$-nodes such that $\tau(\alpha) = \tau(\beta)$, we say that α is a *left sibling node* of β.

The outcomes of a $\mathcal{P}_{e,k}$-node are $0 <_L 1 <_L 2 <_L \cdots$. Outcome 0 is a distinguished outcome, placed to the extreme left signifying the Σ^0_3-outcome of a global win. The other outcomes each represent the code number of a finite sequence of pairs $(n_0, x_0), (n_1, x_1), \cdots, (n_j, x_j)$ where the n_i's are distinct natural numbers, and $x_i \in \{\infty, f\}$, in some effective coding $\langle\cdot\rangle$ with

range $\mathbb{N} \setminus \{0\}$. The only restriction we demand on the coding is to ensure that if σ and η are two such sequences, then $\langle \sigma, (n, \infty), \eta \rangle < \langle \sigma, (n, f), \eta \rangle$. For instance, we could just code the sequence $(0, f), (1, \infty), (2, f)$ as $2^1 3^0 5^1$. We say that m *specifies the pair* (k, x), if m is an outcome where (k, x) appears. We say that m *specifies the finite set* $A \subset \mathbb{N}$ if $A = \{k \mid m \text{ specifies } (k, f) \text{ or } (k, \infty)\}$.

To measure if Φ_e^A is an order, we define *the nondecreasing length of convergence* for Φ_e^A at stage s to be

$$\ell_e[s] = \max\{x < s : (\forall y \leq x)\Phi_e^A(y)[s] \downarrow \geq \Phi_e^A(y-1)[s] \downarrow\}.$$

This is measured at mother nodes, and we also write $\ell_\tau[s]$ in place of $\ell_e[s]$. We say that s is a τ-*expansionary stage* if $s = 0$ or $\ell_\tau[s] > \ell_\tau[s^-]$, where $s^- < s$ is the largest τ-expansionary stage before s. Generally if s is an α-stage, then we let s^- denote the previous α-expansionary stage, if α is a mother node, and the previous α-stage otherwise.

If τ is an $\mathcal{P}_e$-node, we define $m_b^\tau = \langle e, b \rangle - \langle e, 0 \rangle$. This is basically the number of levels j between τ and its b^{th} block devoted to a negative requirement. A b-daughter node α of τ (i.e. a daughter node assigned $\mathcal{B}_b^\tau$) will be allowed $2^{m_b^\tau}$ many boxes at its level. We let $M_b^\tau = \{x : 2^{m_b^\tau} < x \leq 2^{m_{b+1}^\tau}\}$. Again if α is a $\mathcal{P}_{e,b}$-node with top τ, we write m_α^τ and M_α^τ in place of m_b^τ and M_b^τ. Note that $M_1^\tau, M_2^\tau, \cdots$ gives a fixed partition of a cofinite segment of $\mathbb{N}$. The b^{th} block $\mathcal{B}_b^\tau$ will contain all the modules k such that $\Phi_e^A(k) \in M_b^\tau$; this changes with time. Hence with each $\mathcal{B}_b^\tau$-node, we associate the parameter

$$L_\alpha[s] = \{k < \ell_\tau[s] : \Phi_e^A(k)[s] \in M_\alpha^\tau\}.$$

That is, $L_\alpha[s]$ gathers all the modules of τ which should be put into block $\mathcal{B}_b^\tau$ at stage s. All sibling nodes of α will have the same list L_α at all times, so they will all run the same modules.

Note that if Φ_e^A is an order, then every τ-block will eventually settle on a finite set, and $L_\alpha[s]$ also reaches a limit for every daughter node α of τ. On the other hand, it might be possible for us to have the other possibility, in which Φ_e^A is a constant function, but at all stages appear to be an order. For instance, Φ_e^A at various stages may look like $12345\cdots$, $11234\cdots$, $11123\cdots$, $11112\cdots$ and so on. In this case, at some α, $\lim_s L_\alpha[s]$ will be a cofinite set. The other possibility which will give rise to a similar situation is when $\Phi_e^A(x) \uparrow$, but converges infinitely often. This will also cause some $L_\alpha[s]$ to change infinitely often at some block. This is why we need each daughter node α of τ to have a special outcome 0, which is placed to the left of all the other α-outcomes. The role of outcome 0 will be described later.

If α is a $\mathcal{P}_{e,b}$-node, we denote the *label of* α by its position on the tree. This is denoted by lb_α, which is a finite string of length $m_\alpha^{\tau(\alpha)}$ with word $\{w, d\}$, and is determined by the outcomes of the negative nodes lying between $\tau(\alpha)$ and α, when read off starting from τ. For example, if $\mathcal{N}_2, \mathcal{N}_3$ and $\mathcal{N}_4$ lies between $\tau(\alpha)$ and α, with outcomes w, d and w respectively, then $lb_\alpha = wdw$. Again we can partially order all such labels in $\{w, d\}^{<\omega}$ lexicographically,

with w to the left of d; we also use $<_L$ for this ordering. If τ is a top node, and $\delta \supset \tau$ is any node (including negative nodes), we can extend the above definitions to define m^τ_δ and lb^τ_δ. In the above, lb_α is just a special case of $lb^{\tau(\alpha)}_\alpha$.

The point of introducing labels for an ultrahigh node α, is to easily express the following actions. Each α will be in charge of defining $\Psi^\tau(k, lb_\alpha)$ for every $k \in L_\alpha$. Here, Ψ^τ is a functional built at τ; we drop all mention of oracle A. In the end we let $V^\tau_k = \{\Psi^\tau(k, lb_\alpha) : \tau = \tau(\alpha) \text{ and } k \in L_\alpha\}$, so that $|V_k| \leq 2^{m^\tau_\alpha}$. We refer to each value $\Psi^\tau(k, \sigma)$ as a *box*, which we will fill with a number. Note that in this construction, the label or pointer lb_α is fixed; unlike the construction of a cappable ultrahigh set in [11], where the label changes dynamically. Each ultrahigh node α will start work only if it is active. The label lb_α specifies the particular box at which α will fill or clear for every $k \in L_\alpha$. α will fill the box $\Psi^\tau(k, lb_\alpha)$ each time it plays an outcome specifying (k, f), and will clear $\Psi^\tau(k, lb_\alpha)$ each time the outcome specifies (k, ∞).

At some stage s, when we say that we *fill* $\Psi^\tau(k, \sigma)$ *with use* u, we mean that we enumerate the axiom $\langle\langle k, \sigma\rangle, J^{\emptyset'}(k)[s], A_s \upharpoonright u + 1\rangle$ into Ψ^τ. The s^{th} stage use of the computation $\Psi^\tau(k, \sigma)[s]$ is denoted by $u^\tau_{k,\sigma}[s]$. To *clear*, or to *empty* $\Psi^\tau(k, \sigma)$ *at a stage* s, means that we enumerate $u^\tau_{k,\sigma}[s]$ into A. As usual we associate finite strings with their code numbers. During a stage s of the construction in which we visit an ultrahigh node α, and we filled some box for the sake of α, we will say that the box was *filled by* α *at stage* s. This remains true until the box is next emptied.

We say that α *is active* at stage s, if τ allows α to run the modules in its instruction list L_α. That is, $\alpha \supset \tau^\frown\infty$ and we have all of the following:

- (AC.1): $\Phi^A_e(\ell_\tau)[s] \notin M^\tau_\alpha$.
  [Hence we do not expect new numbers to appear in the instruction list $L(\alpha)$, unless there is an A-change.]
- (AC.2): $L_\alpha[s] \neq \emptyset$.
  [Otherwise α has nothing to do.]
- (AC.3): there is some largest $t < s$ which are both α-stages, $L_\alpha[t] = L_\alpha[s]$, and for every $k \leq 1 + \max L_\alpha[s]$, the computation $\Phi^A_e(k)[s]$ has persisted for at least two visits to α (i.e. $A \upharpoonright \varphi_e(k)[t] = A \upharpoonright \varphi_e(k)[s]$).
  [This ensures that the true path is well-defined. That is, we ensure that α switches from being active to inactive each time there is a change in some use in L_α.]

If α is active then it will play an outcome specifying L_α. We also say that α is *permanently active* at stage s, if it is active at every visit to α after stage s. We now define what it means for a computation measured by a negative node α to be believable.

If α is active, then α will continue tracing $J^{\emptyset'}(k)$-values for all $k \in L_\alpha[s]$. The uses of these traces are chosen fresh (in particular, larger than the Φ^A_e-uses), so that when α next plays the outcome 0, all of the previous traces

would have been automatically cleared. This is important because each time the $\mathcal{B}^\tau_b$-node α plays outcome 0, there would be some rearrangement of τ-modules amongst $\mathcal{B}^\tau_b, \mathcal{B}^\tau_{b+1}, \cdots$. For instance, the k^{th} module may have moved from $\mathcal{B}^\tau_{b+1}$ to $\mathcal{B}^\tau_b$. It is important that when the $\mathcal{B}^\tau_b$-nodes begin the next tracing phase, they start with a clean slate.

Definition 3.2. For a negative node α, we say that a computation $\Delta^A(n)[s]$ with use u is α-*believable* at stage s, if all the following hold:

- for every ultrahigh node $\beta \subset \alpha$, and every k such that $\alpha(|\beta|)$ specifies (k, ∞), the box $\Psi^{\tau(\beta)}(k, lb_\beta)$ is either empty or has use $> u$.
- for every ultrahigh node β which has been visited prior to s, such that $\tau(\beta)^\frown\infty \subset \beta^\frown 0 \subseteq \alpha$, and all k, the box $\Psi^{\tau(\beta)}(k, lb_\beta)$ is either empty or has use $> u$.
- every box of lower $\alpha^\frown w$-priority is either empty, or has use $> u$.

At the negative node α, we measure the *believable length of agreement* for $A \equiv_T W_e \oplus V_e$ to be:

$$\ell^b_\alpha[s] = \max\{y < s \mid (\forall x < y)\ \ A_s(x) = \Phi_e^{W_e \oplus V_e}(x)[s] \wedge (\forall z < \varphi_e(x)[s])\Delta^A_e(z)[s] = (W_e \oplus V_e)(z)[s]\}.$$

We require that all the Δ^A_e-computations mentioned are α-believable at s. We distinguish this from the *plain length of agreement*, which we denote simply as $\ell_\alpha[s]$, where the definition is as above except that we do not require the Δ^A_e-computations to be α-believable. If $x < \ell_\alpha[s]$ or $\ell^b_\alpha[s]$, then we denote the nested use by $\delta_e(\varphi_e(x))$. This is not to be confused with the length of convergence when τ is a mother node, which we also denote by ℓ_τ.

By the Recursion Theorem, we are supplied with an infinite list of indices x for which we are able to build the x^{th} Turing functional $J^X(x)$. When a negative node α *picks a fresh index* at stage s, we will pick for α an index from this list which is a fresh number. As usual, a fresh number at stage s is a number larger than s, and larger than anything mentioned so far. We now define the parameters needed for a negative node α. The indices used are: the special index η^α_0 aimed at W'_e, and the others denoted by $\eta^\alpha_1, \eta^\alpha_2, \cdots, \eta^\alpha_{t_i(\eta^\alpha_0)}$ which are all aimed at V'_e. The followers are denoted by $x^\alpha_1, x^\alpha_2, \cdots, x^\alpha_N$ where $N = t_i(\eta^\alpha_0)t_j(\eta^\alpha_{t_i(\eta^\alpha_0)})$. Note that N will be used throughout the construction and verification if everything converges. The indices and followers are picked in increasing order, i.e. $x^\alpha_p < x^\alpha_q$ and $\eta^\alpha_p < \eta^\alpha_q$ whenever $p < q$. We assume that $t_0, t_1, \cdots$ are all non-decreasing functions. Associated with each negative α is a state F_α. This may be either 1, 2 or 3 depending on how far along the basic strategy α is. $F_\alpha = 1$ means that α is still picking the indices η^α_n. State 2 means that α has picked its indices and is now in the process of setting up its followers. In state 3 means that α has begun diagonalization. Note that if α is in state 3, then it cannot tolerate any small A-change unless it is initialized.

During the construction, if s is an α-stage and some computation is convergent at s, then we say that this computation has *persisted for at least two visits to* α, if this computation is also convergent at the previous visit to α, and there has been no change below the use. Given a node α on the tree, and a box $\Psi^\tau(x,\sigma)$, we say that the box *is of lower* α*-priority*, if $\tau^\frown\infty \subseteq \alpha$ and $\sigma >_L lb^\tau_\alpha$.

When we initialize a node α at s, we do the following. If α is a mother node we make $\Psi^\tau(-)$ and u^τ_- undefined at all arguments. Ultrahigh nodes are not initialized in this way; the boxes they control will be cleared when a negative node with a higher priority label is visited. If α is a negative node, we make all the η^α_- and x^α_- undefined, and set $F_\alpha = 1$.

3.6. **The construction.** At stage 0, initialize every node and clear every box. At stage $s > 0$, we inductively define the stage s approximation to the true path, δ_s of length s, and we state the actions taken by the nodes on δ_s. Suppose that $\alpha = \delta_s \upharpoonright d$ has been defined for $d < s$. We will also define $o = \delta_s(d)$, the α-outcome. There are three cases:

(1) α *is a* $\mathcal{P}_e$*-node.* We let $o = \infty$ if s is α-expansionary, and $o = f$ otherwise. Initialize all the nodes to the right of $\alpha^\frown o$ and clear every box of lower $\alpha^\frown o$-priority.

(2) α *is a* $\mathcal{P}_{e,i}$*-node.* We first figure out what the outcome is at the current stage s. If α is currently not active, we let $o = 0$. Otherwise, for each $k \in L_\alpha[s]$, we have to let o specify either (k,∞) or (k,f). Check if the following holds:
   - (OUT.1): the box $\Psi^{\tau(\alpha)}(k, lb_\alpha)$ is either unoccupied, or it is occupied by the correct value (i.e. equals to $J^{\emptyset'}(k)[s]$).
   - (OUT.2): $J^{\emptyset'}(k)[s]\downarrow$, and this computation has persisted for at least two visits to α. We let $t < s$ be the least stage such that the same computation $J^{\emptyset'}(k)[s]\downarrow$ also applies at t (i.e. $J^{\emptyset'}(k)[t]\downarrow$ with $\emptyset'_t \upharpoonright j(k)[t] = \emptyset'_s \upharpoonright j(k)[t]$).
   - (OUT.3): we also require that for every $k' \in L_\alpha[s] - \{k\}$, such that $J^{\emptyset'}(k')[t]\downarrow$ and $\emptyset'_t \upharpoonright j(k')[t] \neq \emptyset'_s \upharpoonright j(k')[t]$, that the current stage s is not the first α-stage after the change. That is, there is a smaller α-stage u such that $t < u < s$, and $\emptyset'_t \upharpoonright j(k')[t] \neq \emptyset'_u \upharpoonright j(k')[t]$.

   If all of the above hold for k, we let o specify (k,f), otherwise we let o specify (k,∞). Specifically we let $o =$ code of the pairs $(k_0,x_0),\cdots,(k_p,x_p)$, where k_i's are precisely all the distinct elements of $L_\alpha[s]$, and the x_i's are determined above. We explain the choices for the conditions OUT.1-3 above - these are implemented purely for technical reasons. OUT.1 ensures that if the box is occupied by the wrong value, then we immediately force outcome (k,∞) to clear it. The combination of OUT.2 and 3 ensures that the true path is consistent with the "truth of outcome", as explained in the earlier section. We now take actions for α. Do all the following

- we clear all the boxes which need clearing. For each k such that o specifies (k, ∞), we clear $\Psi^{\tau(\alpha)}(k, lb_\alpha)$. Also clear every box of lower $\alpha^\frown o$-priority. Finally for each ultrahigh node $\beta >_L \alpha$ such that $\tau(\beta) \subset \alpha$, and $lb_\beta \not<_L lb_\alpha^{\tau(\beta)}$, and x such that the $\Psi^{\tau(\beta)}(x, lb_\beta)$-box is currently full and was filled by β, we clear it. Basically this last action clears all the boxes filled by nodes to the right.
- fill all the boxes which need filling, unless some box which is expected to be occupied is not. That is, if there is some ultrahigh node $\xi \subset \alpha$ such that the ξ-outcome along α specifies (k', f) for some k', but the box $\Psi^{\tau(\xi)}(k', lb_\xi)$ is currently empty, then we skip the following action for α: for every k such that o specifies (k, f), we fill $\Psi^{\tau(\alpha)}(k, lb_\alpha)$ with a fresh use (unless the box is already full). We do this to ensure that the use of α-boxes are always larger than the use of ξ-boxes.
- initialize all the nodes to the right of $\alpha^\frown o$. If this is not the first α-stage, we also initialize all the negative nodes to the left of α which have believed in incorrect computations. For each sibling node $\beta <_{left} \alpha$ or $\beta = \alpha$, and for each outcome $o' > 0$ such that
  - either o' and o do not specify the same set of numbers, or
  - there is some k such that o' specifies (k, f) and o specifies (k, ∞),

  we initialize every node extending $\beta^\frown o'$. Note that this is possible even if o' is placed to the left of o. We injure nodes to the left via this action; we have to show during the verification that nodes on the true path are injured by action to the right only finitely often.

(3) *α is an $\mathcal{N}_{e,i,j}$-node.* If no left sibling of α is in state 3, we do the following, depending on the state F_α:

- $F_\alpha = 1$: look for the least n such that η_n^α is undefined. If $n = 0$, we pick a fresh index for η_0^α. If $n > 0$ and $t_i(\eta_0^\alpha)\downarrow$, pick fresh indices for $\eta_1^\alpha < \cdots < \eta_{t_i(\eta_0^\alpha)}^\alpha$ and set $F_\alpha = 2$.
- $F_\alpha = 2$: if one of $t_j(\eta_1^\alpha), \cdots, t_j(\eta_{t_i(\eta_0^\alpha)}^\alpha)$ has not yet converged, do nothing. Otherwise do the following. We want to get rid of the followers x where there has been a change in A below $\delta_e(\varphi_e(x))$. Look for the smallest n such that $x_n^\alpha \downarrow$ and $\ell_\alpha^b[s^-] > x_n^\alpha$ and $A \upharpoonright \delta_e(\varphi_e(x_n^\alpha))[s^-]$ has changed between s^- and s. Such a change is possible due to lower priority requirements acting. If such n exists, pick the least one and make all $x_{n+1}^\alpha, x_{n+2}^\alpha, \cdots$ undefined. Note that we *do not* undefine x_n^α itself.
  Next, pick the least $n \leq N$ such that x_n^α is undefined. If $n = 0$ we give x_n^α a fresh value, otherwise we give x_n^α a fresh value only if $\ell_\alpha^b[s] > x_{n-1}^\alpha$. Finally if there is no such n (i.e. all the followers have been defined) and $\ell_\alpha^b[s] > x_N^\alpha$, we set $F_\alpha = 3$,

initialize every node extending α and clear every box of lower $\alpha^\frown w$-priority.

- $F_\alpha = 3$: if $\ell_\alpha[s] \leq x^\alpha_N$, do nothing else (we wait for recovery). Also if $|T_i(\eta^\alpha_0)| \geq t_i(\eta^\alpha_0)$ or $|T_j(\eta^\alpha_m)| \geq t_j(\eta^\alpha_m)$ for some $m > 0$, we do nothing else (since the traces do not obey the bounds). Otherwise we let $k = |T_i(\eta^\alpha_0)| + 1$, and we will play the index η^α_0 and η^α_k. Also let x be the largest of the α-followers that has not yet entered A.
  We check if either of $J^{W_e}(\eta^\alpha_0)[s]$ or $J^{V_e}(\eta^\alpha_k)[s]$ is undefined; if so define it (or them) to give output s with use $\varphi_e(x)[s]$, and do nothing else. Otherwise we have both convergent. In this case, we check if both $J^{W_e}(\eta^\alpha_0)[s] \in T_i(\eta^\alpha_0)$ and $J^{V_e}(\eta^\alpha_k)[s] \in T_j(\eta^\alpha_k)$. If both are in the respective traces, we enumerate x into A and initialize every node extending α. Otherwise do nothing.

Note that we use the believable length of agreement when setting up the followers in state 2, and we only talk about the plain length of agreement while diagonalizing in state 3. This is because in state 3 all we really want are $W_e \oplus V_e$-changes that follow the enumeration of a follower x into A, and we are not really interested in the actual recovery of the computation $\Delta^A_e(\varphi_e(x))$. Now we decide the α-outcome. If $F_\alpha = 3$ let $o = d$, otherwise the outcome is w. Initialize all the nodes to the right of $\alpha^\frown o$ and clear every box of lower $\alpha^\frown o$-priority.

If there is some left sibling of α in state 3, we let $\tilde{\alpha}$ be the leftmost one. We take actions under state 3 above, replacing α by $\tilde{\alpha}$. The only difference is that if we enumerate some x into A, we will initialize every node $\beta \supset \tilde{\alpha}$ as well as $\beta >_L \tilde{\alpha}$. This may include nodes β of higher priority than α, and can even be α itself. In the verification we will show that this injury is finite if α is on the true path. We will call this action α *enumerates x on behalf of $\tilde{\alpha}$*. When α acts on behalf of $\tilde{\alpha}$, we are only waiting for the plain length of agreement $\ell_{\tilde{\alpha}} = \ell_\alpha$ to recover; it is unreasonable to wait for all these computations to become $\tilde{\alpha}$-believable while we were at α (nor is it necessary to do so). Let the outcome o of α be d, initialize every node to the right of $\alpha^\frown o$ and clear every box of lower $\alpha^\frown o$-priority.

3.7. **Verification.** It is easy to see that there is a leftmost path visited infinitely often: if α is an ultrahigh node and is never permanently active, then 0 is the leftmost outcome it plays infinitely often; on the other hand if α does become permanently active, then L_α eventually settles and α will have to play one of the $2^{|L_\alpha|}$ many possible outcomes specifying L_α. Let TP be the true path of the construction, defined as usual to be the leftmost path visited infinitely often. We let $True_\alpha$ be the true stage of α, i.e. least α-stage s such that $(\forall t > s)(\delta_t \not<_L \alpha)$. Clearly if $\alpha \subseteq \beta \subset TP$ then $True_\alpha \leq True_\beta$. If we ever visit left of a node α we will initialize α in the same stage. It is

not hard to verify also that if $\alpha \subset \beta$ and α is initialized, then β is initialized by the same action as well.

We have two (possibly conflicting) concepts of "priority" for a node α on the construction tree. The first is determined by the position of α on the tree. The second is determined by the label of α (relative to some top node τ). It is possible for a node α to be physically left of another node α', but yet has a lower priority label than the label of α'. In the next lemma, we reconcile these conflicts; in particular we show that the only way for us to first visit α, and then visit α' later in the construction, is for us to injure α in the meantime.

Lemma 3.3. *Suppose α, α' are negative nodes which are siblings, s is a $\alpha^\frown d$-stage and $t > s$ is a $\alpha'^\frown w$-stage, and α has been visited prior to s. Then there is some $s \leq s' \leq t$ such that α is initialized at s'.*

Proof. If $\alpha' <_L \alpha$ then we are done, so assume that $\alpha' \geq_L \alpha$. Observe that the only way for any negative node to get out of state 3 is for it to be initialized. At stage s either $F_\alpha = 3$ or some left sibling of α is in state 3. If the former holds then α has to be initialized to make $F_\alpha \neq 3$ before α' is visited. Suppose the latter holds, and at s α acts on behalf of $\tilde{\alpha} <_L \alpha$. Similarly $\tilde{\alpha}$ has to be initialized at some s' between s and t. If this initialization of $\tilde{\alpha}$ was *not* because $\tilde{\alpha}$ believed in an incorrect computation, then it is trivial to verify that the same action would also initialize α to the right of $\tilde{\alpha}$. Therefore the initialization to $\tilde{\alpha}$ was because $\tilde{\alpha}$ had previously believed in an incorrect computation. That is, $\tilde{\alpha}$ extends some $\tilde{\beta}^\frown o'$ for some ultrahigh node $\tilde{\beta}$ and outcome $o' > 0$, which is deemed to be incorrect by $\delta_{s'} \upharpoonright |\tilde{\beta}|$ visited at s'. Now α also extends some $\beta^\frown o$ for $|\beta| = |\tilde{\beta}|$. If $\tau(\beta) \neq \tau(\tilde{\beta})$ then by the fact that the nodes $\delta_{s'} \upharpoonright |\tilde{\beta}|$ and $\tilde{\beta}$ are siblings, we have α initialized at s' since α is strictly right of $\delta_{s'}$. Hence we may assume that $\tau(\beta) = \tau(\tilde{\beta})$. We analyze o' and o. At s when $\beta^\frown o$ was visited we had not initialized $\tilde{\alpha}$, which implies that o and o' specifies the same set (hence $o > 0$), and whenever o' specifies (k, f) we must also have o specifies (k, f). This means that at s', $\delta_{s'} \upharpoonright |\tilde{\beta}|$ would also consider β and o to be incorrect, and so α extending this will be initialized at s'. □

For the next lemma 3.4, we let α be a $\mathcal{P}_{e,i}$-node and $\tau = \tau(\alpha)$ such that $\alpha \supset \tau^\frown \infty$. Assume α lies on the true path. Let $L := \lim_s L_\alpha[s]$, which may not exist. It is not hard to see that

- Φ_e^A is an order $\Rightarrow$ L exists and $L = \{k \mid \Phi_e^A(k) \in M_i^\tau\}$.
- α becomes permanently active iff L exists, $L \neq \emptyset$ and for all $k \leq 1 + \max L$, $\Phi_e^A(k) \downarrow$.
- If α never becomes permanently active, then 0 is the true outcome of α.

In the next lemma, we show that the true α-outcome is consistent with the "truth of outcome". This is instrumental in showing later that nodes on the true path are injured by action on the right only finitely often. In (ii) we

show that if $J^{\emptyset'}(k)\downarrow$, then not only will the true outcome specify (k,f), but there will also be only finitely many outcomes (k,∞) being played at the whole level. This ensures that boxes filled by a positive node on the true path, never becomes emptied if $J^{\emptyset'}(k)\downarrow$.

Lemma 3.4. *Suppose that α becomes permanently active.*

(i) *Let $L=\{k_0,\cdots,k_p\}$. The true outcome of α is $(k_0,x_0),\cdots,(k_p,x_p)$ where for each i, $x_i=\infty$ iff $J^{\emptyset'}(k_i)\uparrow$.*

(ii) *For each $k\in L$ such that $J^{\emptyset'}(k)\downarrow$, there can only be finitely many stages t such that $\delta_t\supset\tau^\frown\infty$ and $\delta_t(|\alpha|)$ specifies (k,∞).*

(iii) *For all but finitely many stages t, whenever $\delta_t\supset\tau^\frown\infty$ then*
- *if t is the first visit to $\delta_t\upharpoonright|\alpha|$ then $\delta_t(|\alpha|)=0$, and*
- *if t is not the first visit to $\delta_t\upharpoonright|\alpha|$ then $\delta_t(|\alpha|)$ specifies L, and furthermore $\delta_t(|\alpha|)\not<_L(k_0,x_0),\cdots,(k_p,x_p)$.*

Proof. (i) We partition L into the two parts, $L_\infty:=\{k\in L\mid J^{\emptyset'}(k)\uparrow\}$, and $L_f:=L-L_\infty$. We let $s_0>True_\alpha$ be a large enough α-stage, such that

- L_α has settled,
- α is always active when visited,
- for each $k\in L_f$, $J^{\emptyset'}(k)\downarrow$ has stabilized on the correct use, and
- for each $k\in L_f$, the box $\Psi^\tau(k,lb_\alpha)$ is not occupied by an incorrect value. This is possible because if the box is occupied by a number other than $J^{\emptyset'}(k)$, then α will play an outcome specifying (k,∞) every time it is visited until the box is cleared.

At every visit to α after s_0, α will play an outcome specifying L. To prove that the true outcome of α is $(k_0,x_0),\cdots,(k_p,x_p)$, we do it in two parts. Firstly, we show that for each $k\in L_f$, there are only finitely many stages such that α plays an outcome specifying (k,∞). Secondly, we will show that there are infinitely many stages such that α plays an outcome specifying *all of* $\{(k,\infty)\mid k\in L_\infty\}$.

To prove the first part, we fix a $k\in L_f$. Let $s_1\leq s_0$ be the least such that $J^{\emptyset'}(k)[s_1]\downarrow$ on the correct use. For each $k'\in L_\infty$ such that $J^{\emptyset'}(k')[s_1]\downarrow$, there must be a change in $\emptyset'$ below the k'-use after stage s_1. Wait until all these changes occur, for all these $k'\in L_\infty$, and we let $s_2>s_0$ be a large enough α-stage after these changes. We claim that after stage s_2, any outcome played by α has to specify (k,f): Pick $k'\in L-\{k\}$, such that $J^{\emptyset'}(k')[s_1]\downarrow$ with $\emptyset'_{s_1}\upharpoonright j(k')[s_1]\neq\emptyset'_{s_2}\upharpoonright j(k')[s_1]$. We want to show that s_2 is not the first α-stage after the change. If $k'\in L_f$ then s_0 would be a better candidate instead of s_2, since any such change has to occur by stage s_0; on the other hand if $k'\in L_\infty$ then it follows by the choice of s_2 large.

To prove the second part, we let $s_3>s_0$ be an α-stage. We show that there is a stage $s_4\geq s_3$ such that α plays an outcome specifying *all of* $\{(k,\infty)\mid k\in L_\infty\}$. We may assume that $\tilde{L}:=\{k\in L_\infty\mid J^{\emptyset'}(k)[s_3]\downarrow\}\neq\emptyset$ (otherwise we can let $s_4=s_3$). For each $k\in\tilde{L}$, there is a least α-stage $s_4^k>s_3$ such that $\emptyset'$ would have changed below the use of $j(k)[s_3]$. Let $\tilde{k}$

be some member of $\tilde{L}$ with the largest $s_4^{\tilde{k}}$, and let $s_4 = s_4^{\tilde{k}}$. Thus s_4 is the largest α-stage amongst all the first changes. We argue that this choice of s_4 works, in particular we fix a $k \in L_\infty$ such that $J^{\emptyset'}(k)[s_4]\downarrow$, and we want to show that the outcome played by α at stage s_4 specifies (k, ∞). Assume the computation $J^{\emptyset'}(k)[s_4]\downarrow$ has persisted for two visits (otherwise it is trivial). Hence $k \neq \tilde{k}$, and therefore condition (OUT.3) in the construction is not met when deciding the outcome for k, because s_4 is the first α-stage after a $j(\tilde{k})$-change. Hence (k, ∞) is played at stage s_4.

(ii) This is proved in a similar way as in the first part of (i). Fix a $k \in L$ such that $J^{\emptyset'}(k)\downarrow$, and let s_1, s_2 be as in (i). The stage s_2 works for (i), but not for us in this case; we have to wait until every right sibling node α' of α which had been visited prior to stage s_2, is visited again at least one more time (if ever). Say this happens by stage $s_5 > s_2$. Now the argument in (i) can be used to show that at every stage $t > s_5$ such that $\delta_t \supseteq \tau^\frown\infty$, we either have $\delta_t(|\alpha|) = 0$, or else $\delta_t(|\alpha|)$ specifies (k, f).

(iii) Immediate from (ii). □

The next lemma tells us that each time some computation in L_α changes, certain τ-boxes will automatically be cleared. That is, if k is not eventually in L_α, then all the $\Psi^\tau(k, \gamma)$-boxes with $|\gamma| = m_i^\tau$, which are incorrect coding locations for the k^{th} row, will be unoccupied in the limit. Part (ii) covers the case when k eventually does enter L_α (and is in L_α at every α-stage), but yet α has true outcome 0. This might be possible if we have infinitely many uses for $\Phi_e^A(k)[t]$, so that $\Phi_e^A(k)\uparrow$, but $\Phi_e^A(k)[t] \in M_i^\tau$ whenever it converges. Every time α plays outcome 0, this box might be occupied; unlike in (i), because some sibling node of α to the right might fill the box after each change in the use for $\Phi_e^A(k)[t]$. In this case we get a global win at τ, and τ would not care which boxes are filled; however this is important for the negative nodes in the construction tree extending $\alpha^\frown 0$ which might be on the true path. These negative nodes might see Δ-computations which they would like to freeze, but there is currently a use of some τ-box below the δ-use, which will later be cleared (by the lemma below). Hence these negative node would know not to believe in these Δ-computations until these τ-boxes are cleared.

Lemma 3.5. *Let α be a $\mathcal{P}_{e,i}$-node and $\tau = \tau(\alpha)$ such that $\alpha \supset \tau^\frown\infty$. Assume α lies on the true path and s is an α-stage which is not the first one.*

(i) *If α plays outcome 0 at stage s, then for every number $k \notin L_\alpha[s]$ and string γ of length m_i^τ, the box $\Psi^\tau(k, \gamma)$ has to be empty at stage s.*
(ii) *For every k and string γ of length m_i^τ, if the box $\Psi^\tau(k, \gamma)$ gets filled by the actions at stage s (if not already full), then the box will have to be emptied at least once before α can next play outcome 0.*

Proof. (i) Suppose the contrary, let $k \notin L_\alpha[s]$ and γ of length $|\gamma| = |lb_\alpha|$ be such that the box is full with use $u_{k,\gamma}^\tau[s]\downarrow$. This box must have been filled

at some stage $t < s$, in which $\delta_t(|\alpha|)$ was active and $k \in L_\alpha[t]$. The fact that k is no longer in $L_\alpha[s]$, means that A has to change below t. Hence this change will happen after $u^\tau_{k,\gamma}[s] > t$ was set, a contradiction. In fact, we don't need that α plays outcome 0; this will always be true if k goes out of L_α.

(ii) Again the box must have been filled when $k \in L_\alpha[t]$. Before α can next play outcome 0 after stage s, there must be a change some computation in $L_\alpha[t]$, which would clear the box before the next $\alpha^\frown 0$-visit. Note that we are not claiming that the box is *empty* at the next $\alpha^\frown 0$-stage after s; just that the box would be emptied at least once in between. □

Now we will show that despite the fact that nodes can be injured by action on the right, this only happens finitely often to nodes on the true path.

Lemma 3.6. *Every node on the true path is initialized finitely often.*

Proof. We proceed by induction on the length. Let α be on true path, and suppose no node $\beta \subset \alpha$ gets initialized anymore. We consider all possibilities, i.e. we suppose α gets initialized by the actions of some node β. We may assume that $\beta \not<_L \alpha$. Since β cannot be a top node, we divide the proof into two parts. Firstly suppose that β is a negative node. If β is not acting on behalf of some sibling node, then we must have $\beta \subset \alpha$ and so this happens only finitely often (because β never picks new followers). Suppose β was acting on behalf of some sibling node $\tilde{\beta} <_L \beta$. We must have either $\tilde{\beta} \subset \alpha$ or else $\tilde{\beta} <_L \alpha$. If we let $X[t]$ be the collection of all numbers x such that x is a follower (at stage t) of some negative node in state 3, which is either to the left of α or it is extended by α. Observe that for all large enough t, $X[t]$ is non-increasing, i.e. $X[t+1] \subseteq X[t]$. Now each time α is initialized by β acting on behalf of some $\tilde{\beta}$, this event will also correspond to some number in $X[t]$ being enumerated into A. So, this only happens finitely often (since $X[t]$ is finite).

Now for the second part we assume that β is an ultrahigh node. Hence β initializes α because it thought that α had believed in incorrect computations. So α extends some $\tilde{\beta}^\frown o'$ for some outcome $o' > 0$ and some sibling $\tilde{\beta} <_L \beta$ or $\tilde{\beta} = \beta$. Let o be the β-outcome played at stage t, when this initialization took place (at a very large stage t). Since $o' > 0$ and is the true outcome of $\tilde{\beta}$, it follows that $\tilde{\beta}$ will become permanently active, and by Lemma 3.4, and the fact that t is not the first visit to β, we must have both o and o' specifies the same set. Hence there is some k where o' specifies (k, f) and o specifies (k, ∞), which is a contradiction. So after a large enough t no ultrahigh node can initialize α. □

As an important consequence to Lemmas 3.3 and 3.6, we have that if $i < j$ then for almost every stage t, we have

$$lb^{TP\restriction i}_{TP\restriction j} \leq_L lb^{\delta_t\restriction i}_{\delta_t\restriction j}.$$

This will be assumed in the rest of the proof.

Lemma 3.7. *Along the true path, all the negative requirements are satisfied.*

Proof. Take a $\mathcal{N}_{e,i,j}$-node α on the true path, and suppose that α is never initialized after some stage s_0. Assume also that t_i and t_j are total, and $|T_i(x)| < t_i(x)$ and $|T_j(x)| < t_j(x)$ for all x. Also assume that $A = \Phi_e^{W_e \oplus V_e}$ and $W_e \oplus V_e = \Delta_e^A$. If any of the assumptions above are not met, then $\mathcal{N}_{e,i,j}$ is satisfied vacuously. It is easy to see that $\ell_\alpha \to \infty$.

Claim 1. There is a leftmost node β, such that $|\beta| = |\alpha|$ and $F_\beta = 3$ at almost all stages. If there is a left sibling node of α with this property, then we are done. Otherwise at almost every visit to α, we must in fact have *no* left sibling of α is at state 3. Since α is never initialized after s_0, hence its state never decreases. We claim that $\lim F_\alpha = 3$. If not, then $\lim F_\alpha = 1$ or 2. It is obvious that $\lim F_\alpha = 1$ is not possible, so at some point all the indices will be picked and $\lim F_\alpha = 2$.

We argue inductively that for every $0 \leq i \leq N$, x_i^α will become eventually defined forever, and also that $\liminf \ell_\alpha^b > x_i^\alpha$. x_i^α will clearly be defined (and never again undefined) when $\Delta_e(\varphi_e(x_{i-1}^\alpha))$ has settled, and is believable. To see that ℓ_α^b has $\liminf$ greater than x_i^α, we suppose that there is there is some box with use $u_{k,\sigma}^\tau < \delta_e(\varphi_e(x_i^\alpha))$ where $\delta_e(\varphi_e(x_i^\alpha))$ is the correct use. We show that this box must be emptied at the next visit to α (if not before), which gives a contradiction. To see this, suppose the second option in Definition 3.2 fails (the first option fails is trivial). We apply Lemma 3.5(ii) to get a contradiction. Suppose the third option fails. Since α always play outcome w when visited, hence at the next α-stage this box will be cleared. This contradiction says that ℓ_α^b is almost always larger than x_i^α. Hence there will be a stage where we give F_α state 3, a contradiction.

Let s_2 be the final stage where β moves from state 2 to 3. It is easy to see that for all i, $x_{i+1}^\beta > \delta_e(\varphi_e(x_i^\beta))[s_2]$ and $\ell_\beta^b[s_2] > x_i^\beta$. By Lemma 3.3, for any $i < |\beta|$ and all $t \geq s_2$, we must have $lb_{\beta^\frown d}^{\beta\restriction i} \leq_L lb_{\delta_t\restriction 1+|\beta|}^{\delta_t\restriction i}$. We can in fact say something more: if $\tau \subset \beta$ and a node $\xi \supset \tau$ is visited at some stage after s_2, such that $|\xi| > |\beta|$. Then we in fact have $lb_{\beta^\frown w}^\tau <_L lb_\xi^\tau$. This is because $lb_{\beta^\frown d}^\tau = (lb_\beta^\tau)^\frown d >_L (lb_\beta^\tau)^\frown w = lb_{\beta^\frown w}^\tau$.

Claim 2. After s_2, no A-change below $q := \delta_e(\varphi_e(x_N^\beta))[s_2]$ is possible apart from an enumeration of some x_i^β. Any such change has to be produced by the actions of some node ξ at stage $t \geq s_2$. Let o be the outcome played by ξ at t. Suppose ξ was clearing a box of lower $\xi^\frown o$-priority. The box would be $\Psi^\tau(k,\sigma)$ for some $\tau \subset \xi$ and k, and $\sigma >_L lb_{\xi^\frown o}^\tau$. We may assume that $\tau \subset \alpha, \beta$, otherwise the box is only filled after s_2 and poses no threat to us. By the above observation about the priority of labels, we must either have $lb_{\xi^\frown o}^\tau >_L lb_{\beta^\frown w}^\tau$ or else $lb_{\xi^\frown o}^\tau \subset lb_{\beta^\frown w}^\tau$. Thus σ is strictly to the right of $lb_{\beta^\frown w}^\tau$, which means that the box $\Psi^\tau(k,\sigma)$ is of lower $\beta^\frown w$-priority. Hence it would be cleared by β at s_2 itself, and the only possibility is if $\xi = \beta$ and $t = s_2$. By β-believability this action does not change A below q.

Suppose on the other hand, ξ is a negative node which changed $A \upharpoonright q$ because it was clearing a box of lower $\xi^\frown w$-priority (as it moved from state 2 to 3). At t, ξ is visited and it moved from state 2 to 3. Also, the outcome of ξ at t is d. As above, the box being cleared is $\Psi^\tau(k,\sigma)$ for some $\tau \subset \xi, \alpha, \beta$ and k, and $\sigma >_L lb^\tau_{\xi^\frown w} = lb^\tau_\xi{}^\frown w$. If $|\xi| > |\beta|$ then it is easy to see once again that the box $\Psi^\tau(k,\sigma)$ is of lower $\beta^\frown w$-priority. We suppose on the other hand that $|\xi| \leq |\beta|$; and show that this is impossible. Since at t, ξ would initialize every node extending it, we must in fact have $\beta <_L \xi$. We may assume $lb^\tau_{\xi^\frown d} \subseteq lb^\tau_{\beta^\frown d}$ (because the alternative $lb^\tau_{\xi^\frown d} >_L lb^\tau_{\beta^\frown d}$ is treated as above), it is not hard to verify that after s_2, ξ to the right of β can never play outcome w again. However observe that at s_2, ξ would be initialized since it is to the right of β. The only way for us to have stage t after s_2, is for ξ to promote its state first from 1 to 2 at some stage t' strictly between s_2 and t. At such a stage t', ξ must be acting for its own sake, and be having outcome w, a contradiction. This shows that the box $\Psi^\tau(k,\sigma)$ is of lower $\beta^\frown w$-priority, and will be cleared at s_2, a contradiction to the existence of ξ and t.

Now we can assume ξ changed A for a different reason (other than clearing boxes of lower priority). Hence ξ has to be either an ultrahigh or a negative node. Suppose first that ξ is a negative node. At t, ξ can either be acting for itself, or it is acting on behalf of some sibling $\tilde{\xi}$. Consider first the case when ξ is acting for itself. Since ξ cannot be initialized at s_2, so $\xi \not\supset \beta$ and $\xi \not>_L \beta$. ξ also cannot be to the left of β, and if $\xi \subset \beta$ then it would initialize β after s_2. The only possibility is $\xi = \beta$, which means that one of the x^β_i would be enumerated. If ξ was acting on behalf of $\tilde{\xi}$ then a similar argument follows with $\tilde{\xi}$ in place of ξ.

Now suppose that ξ is an ultrahigh node, so that $u^{\tau(\xi)}_{k,lb_\xi}[t]$ is enumerated into A, for some k such that o specifies (k,∞). Again we may assume that $\tau(\xi) \subset \beta$ otherwise $u^{\tau(\xi)}_{k,lb_\xi}[t]$ is picked after s_2. If $|\xi| > |\beta|$ then we have $lb_\xi >_L lb^{\tau(\xi)}_{\beta^\frown w}$, so it would have been cleared by β at s_2, and so has use at stage t larger than q. On the other hand suppose $|\xi| < |\beta|$. Let $\tilde{\xi} = \beta \upharpoonright |\xi|$. Since β cannot be initialized by ξ at stage t, the outcome of $\tilde{\xi}$ along β is either 0 or else it specifies (k,∞). In both cases, the use cannot be less than q, again due to β-believability.

Lastly suppose that ξ is an ultrahigh node clearing a box filled by some ultrahigh node ξ' to its right. The box is again of the form $\Psi^{\tau(\xi')}(x, lb_{\xi'})$ for some x and $lb_{\xi'} \not<_L lb^{\tau(\xi')}_\xi$. Note that ξ' must have filled this box at a stage strictly before s_2. Again we may assume that $\tau(\xi') \subset \beta$; we divide into three cases. Firstly if $|\xi| < |\beta|$, then $\beta \upharpoonright |\xi|$ will be strictly to the left of ξ', and at s_2 when $\beta \upharpoonright |\xi|$ is visited we cannot have $lb_{\xi'} <_L lb^{\tau(\xi')}_{\beta\upharpoonright|\xi|}$, otherwise we will also have $lb_{\xi'} <_L lb^{\tau(\xi')}_\xi$. This means that the box $\Psi^{\tau(\xi')}(x, lb_{\xi'})$ will be cleared by the action of $\beta \upharpoonright |\xi|$ at s_2, which is a contradiction. For the second scenario

we assume that $|\xi| > |\beta| > |\xi'|$. Then we have $lb_{\xi}^{\tau(\xi')} >_L lb_{\beta^\frown w}^{\tau(\xi')}$. In this case, $\beta \upharpoonright |\xi'|$ is visited at s_2 and is strictly to the left of ξ', and again we cannot have $lb_{\xi'} <_L lb_{\beta\upharpoonright|\xi'|}^{\tau(\xi')}$, so that the box $\Psi^{\tau(\xi')}(x, lb_{\xi'})$ will also be cleared by the action of $\beta \upharpoonright |\xi'|$ at s_2. Lastly both $|\xi'|, |\xi| > |\beta|$. Then neither $lb_{\xi'} <_L lb_{\beta^\frown w}^{\tau(\xi')}$ nor $lb_{\xi'} \supseteq lb_{\beta^\frown w}^{\tau(\xi')}$ is possible since $lb_{\beta^\frown w}^{\tau(\xi')} <_L lb_{\xi}^{\tau(\xi')}$. Since ξ' is long, we also cannot have $lb_{\xi'} \subset lb_{\beta^\frown w}^{\tau(\xi')}$. The last case is $lb_{\beta^\frown w}^{\tau(\xi')} <_L lb_{\xi'}$, in which case the box $\Psi^{\tau(\xi')}(x, lb_{\xi'})$ is of lower $\beta^\frown w$-priority, so that it will be cleared by β at s_2. This concludes the verification of claim 2.

Note that we are not claiming that $\delta_e(\varphi_e(x_N^\beta))[t] = q$ for every t. In fact, the nested use $\delta_e(\varphi_e(x_N^\beta))[t]$ can become larger than q when x_N^β enters A. Subsequently we may have other nodes changing A below $\delta_e(\varphi_e(x_N^\beta))[t]$, but always does so above q. By Claim 2 we have that at every stage t after s_2 and for every k, whenever $x_k^\beta \notin A_t$ then $\varphi_e(x_k^\beta)[t] = \varphi_e(x_k^\beta)[s_2]$ and $\delta_e(\varphi_e(x_k^\beta))[t] = \delta_e(\varphi_e(x_k^\beta))[s_2]$, and furthermore the respective sets have not changed up till these uses. This is because if $W_e \oplus V_e$ changes without being permitted by us, then β (or any sibling acting on behalf of β) immediately becomes inactive to preserve the disagreement.

We also have infinitely many chances to act for β, or act on behalf of β. Suppose some largest x_c^β is never enumerated into A. Let $k = |T_i(\eta_0^\beta)| + 1$ (the actual size), and $k \leq t_i(\eta_0^\beta)$. Since $\varphi_e(x_c^\beta)$ settles, it follows that both $J^{W_e}(\eta_0^\beta)$ and $J^{V_e}(\eta_k^\beta)$ are eventually defined. This means that one of the two final values will not be in the respective traces, so $\mathcal{N}_{e,i,j}$ is satisfied. We may therefore assume that every one of the followers is enumerated into A. Let t_c be the stage where x_c^β is enumerated into A. We prove the crucial claim:

Claim 3. Between t_{c+1} and t_c, either $T_i(\eta_0^\beta)$ has a new number enumerated into it, or else some number strictly between t_{c+1} and t_c is enumerated into one of $T_j(\eta_1^\beta), \cdots, T_j(\eta^\beta_{t_i(\eta_0^\beta)})$. Let $k = |T_i(\eta_0^\beta)[t_{c+1}]|+1$, and suppose that no new number enters $T_i(\eta_0^\beta)$ between the two stages. Now both $J^{W_e}(\eta_0^\beta)[t_{c+1}]$ and $J^{V_e}(\eta_k^\beta)[t_{c+1}]$ are defined, and are in the respective traces. Since these computations were defined before t_{c+1} (say at t'), it follows that they have use at least as large as the use $\varphi_e(x_{c+1}^\beta)[t']$. By the remarks after Claim 2, it follows that $\varphi_e(x_{c+1}^\beta)[t'] = \varphi_e(x_{c+1}^\beta)[t_{c+1}]$. This is the crucial part where we use Claim 2. There must be a recovery of l_β some time between t_{c+1} and t_c (in which β or some sibling acting on behalf of β got to act). At that stage we have either W_e-change (hence $J^{W_e}(\eta_0^\beta)\uparrow$) or V_e-change (hence $J^{V_e}(\eta_k^\beta)\uparrow$). Now we cannot have a W_e-change, because otherwise we give $J^{W_e}(\eta_0^\beta)$ a new value which will be in $T_i(\eta_0^\beta)[t_c]$. Hence we must have a V_e-change at recovery, and we will give it a new value which will be reflected in $T_j(\eta_k^\beta)$.

By a simple counting argument and Claim 3, it follows that we cannot have enumerated all of $x_1^\beta, \cdots, x_N^\beta$, giving a contradiction. This shows that $\mathcal{N}_{e,i,j}$ is satisfied. □

Lemma 3.8. *Along the true path of construction, all the positive requirements are satisfied.*

Proof. Let τ be a $\mathcal{P}_e^A$-node on the true path, such that Φ_e^A is an order. The true τ-outcome is clearly ∞. We will show that $\{V_k^\tau\}_{k\in\mathbb{N}}$ traces $J^{\emptyset'}$ correctly. We fix a number k, and let i be such that $\Phi_e^A(k) \in M_i^\tau$, and α be the version of $\mathcal{P}_{e,i}^A$ on the true path with true outcome o. Hence α will eventually be responsible for tracing $J^{\emptyset'}(k)$, when L_α settles.

We first of all show that $|V_k^\tau| < \Phi_e^A(k)$. Take a box in the k^{th} row, i.e $\Psi^\tau(k,\sigma)$ for some σ. If this is filled by a node β where $|\beta| \neq |\alpha|$, then β must have done so when k was in L_β. Since k has to leave L_β, hence the box must be cleared before that. Therefore any such box $\Psi^\tau(k,\sigma)$ on row k can only be defined permanently by a node at level $|\alpha|$. There are at most $2^{m_i^\tau}$ many distinct labels of the nodes at that level, so $\Psi^\tau(k,\sigma)$ is only defined permanently in at most $2^{m_i^\tau} < \Phi_e^A(k)$ many different σ. Next, suppose that $J^{\emptyset'}(k)\downarrow$. We want to show that $J^{\emptyset'}(k) \in V_k^\tau$. We know α will become permanently active and o will specify (k,f). It is enough to prove the following claim (by induction):

Claim 1. For any ultrahigh node α along the true path and any k such that the true α-outcome specifies (k,f), the box $\Psi^{\tau(\alpha)}(k, lb_\alpha)$ receives a permanent definition with value $J^{\emptyset'}(k)$. We proceed by induction on the length of α. Take α on the true path with true outcome o specifying (k,f) for some k, and $\tau = \tau(\alpha)$. Clearly $J^{\emptyset'}(k)\downarrow$; let s_0 be a large $\alpha^\frown o$-stage. By induction hypothesis at s_0, α would not be stopped from putting $J^{\emptyset'}(k)$ into the box $\Psi^\tau(k, lb_\alpha)$, unless of course the box $\Psi^\tau(k, lb_\alpha)$ itself is occupied. In that case it has to be occupied by the correct value, since α only plays an outcome specifying (k,∞) finitely often. This can in fact be said about every $\alpha^\frown o$-stage after s_0; we must have the box $\Psi^\tau(k, lb_\alpha)$ occupied (possibly filled by α). The only issue is whether we can get it to be occupied permanently without being made undefined infinitely often. We assume for a contradiction that infinitely often we have the box $\Psi^\tau(k, lb_\alpha)$ being made undefined infinitely often. In particular, there are infinitely many α-stages s and some stage $t \geq s$ such that an enumeration of a number $p \leq u^\tau_{k,lb_\alpha}[s]$ is made at stage t. In the following, s is an arbitrary $\alpha^\frown o$-stage.

- p is the use of a box $\Psi^{\tau'}(k',\sigma)$ which is filled by a node ξ with $|\xi| > |\alpha|$. We claim that if $\Psi^{\tau'}(k',\sigma)$ was filled after s_0, then it would not be possible for it to do the above. In particular, assume that $p \leq u^\tau_{k,lb_\alpha}[s]$ is enumerated at t for some $s_0 \leq s \leq t$ and s is an $\alpha^\frown o$-stage. Say ξ is an ultrahigh node which filled p, which must be at or before s. ξ cannot be to the right of $\alpha^\frown o$, because at s, α would clear the ξ-box. $\xi \not\supseteq \alpha$ because ξ is long. So $\xi \supseteq \alpha^\frown o$ is the only

possibility left. Since ξ would have filled p strictly before s, and also the fact that when ξ filled p (at $s' < s$), we ensured that $u^{\tau}_{k,lb_\alpha}[s'] \downarrow$, we have $u^{\tau}_{k,lb_\alpha}[s'] < p \leq u^{\tau}_{k,lb_\alpha}[s]$, which means an A-change below p will take place before s, a contradiction.

- p is the use of a box $\Psi^{\tau'}(k', \sigma)$ of lower $\beta^\frown o'$-priority (when $\beta^\frown o'$ is visited). We cannot have $\tau' <_L \alpha$ because we never visit left of α. We cannot have $\tau' >_L \alpha$ because τ' is initialized at s. We cannot have $\tau' \supset \alpha$ as well, because then the box will be filled by some τ'-daughter of longer length than α. Suppose now that $\tau' \subset \alpha$. We have $\sigma >_L lb^{\tau'}_{\beta^\frown o'}$. If $|\sigma| \geq |lb^{\tau'}_{\alpha^\frown o}|$, then σ has to be filled by a τ'-daughter node of a longer length than α, which is already considered above. So we must have $|\sigma| < |lb^{\tau'}_{\alpha^\frown o}|$, and by Lemma 3.3 we also have $\sigma >_L lb^{\tau'}_{\alpha^\frown o}$ and so the box $\Psi^{\tau'}(k', \sigma)$ also of lower priority than $\alpha^\frown o$. This means the box is cleared at s, and by considering labels, p cannot be picked by α itself. Hence p is picked strictly after s, which is a contradiction.
- Suppose p is enumerated by an ultrahigh node β. The ultrahigh node β can be enumerating p either because p is the use of some box filled by a node to its right, or it is enumerating p because p is the use of a box which its outcome is telling it to clear. Suppose the latter holds. p is the use of some box, which is filled by some ultrahigh node ξ at some stage $t' \leq s$. We claim that if p is filled after s_0, then this will lead to a contradiction. We draw attention once again to the fact that at s, α will always clear all boxes it needs to clear, before filling the box $\Psi^{\tau}(k, lb_\alpha)$. There are now three cases depending on the position of ξ; remember that the box $\Psi^{\tau(\xi)}(k', lb_\xi)$ is filled by ξ and has to be cleared by β which is a sibling node of ξ. We want to get a contradiction in all three cases:
  - (C1) $\xi \subseteq \alpha$: then ξ is along the true path, so by Lemma 3.5(ii) the true outcome of ξ cannot be 0. Hence ξ will become permanently active and the true outcome of ξ has to specify either (k', ∞) or (k', f). By Lemma 3.4, the true outcome of ξ has to specify (k', ∞), since β has to play (k', ∞). This means that at s itself p would have to be cleared, a contradiction. This applies even if $\xi = \alpha$.
  - (C2) $\xi >_{left} \alpha$: at s, we would have cleared the box with use p when α is visited. This happens before α fills any box, so is a contradiction.
  - (C3) $\xi \supset \alpha$: then $|\xi| > |\alpha|$, and we have been through this case.

  Suppose now the former holds, i.e. β is enumerating p because it is the use of a box filled by some node ξ to its right. In this case, it is not hard to see that the only possibilities for ξ are (C2) and (C3) above, and the same argument there applies.

- lastly p is enumerated by a negative node β. β can either be enumerating some follower (of its own or some left sibling), or it could be clearing a box of lower $\beta^\frown w$-priority as it moved from state 2 to 3. Suppose the latter holds, i.e. β is clearing the box $\Psi^{\tau'}(k', \sigma)$. If $|\beta| > |\alpha|$ then the argument will be similar to the case when a box of lower $\beta^\frown o'$-priority is cleared. Suppose $|\beta| < |\alpha|$, and in fact we may assume that $lb^{\tau'}_{\beta^\frown d} \subseteq lb^{\tau'}_\alpha$ (because the alternative $lb^{\tau'}_{\beta^\frown d} >_L lb^{\tau'}_\alpha$ can be treated as above). This is now dealt with in a similar fashion as one of the cases in Lemma 3.7. After s, β to the right of α can never play outcome w again. At s, β would be initialized since it is to the right of α. β must promote its state first from 1 to 2 at some stage t' strictly between s and t. At such a stage t', β must be acting for its own sake, and be having outcome w, a contradiction.

  Assume further that s_0 is large enough such that p cannot be enumerated under the situations described above. We consider the following well-ordering of nodes on the construction tree, by letting $\zeta_1 <_P \zeta_2$ iff either $\zeta_1 <_L \zeta_2$ or else $\zeta_1 \subset \zeta_2$. Let ξ be the least negative node wrt $<_P$ extending $\alpha^\frown o$, and has a follower enumerated into A after s_0 (this may be due to ξ or some sibling node acting on behalf of ξ); if ξ does not exist then we are done. If only finitely many ξ-followers are enumerated into A, let $s_1 > s_0$ be the first $\alpha^\frown o$-stage after the last enumeration of a ξ-follower. Otherwise if infinitely many ξ-followers are enumerated into A, then ξ must be to the right of the true path; let $s_1 > s_0$ be an $\alpha^\frown o$-stage such that we moved left of ξ.

  We claim that no p below $u^\tau_{k,lb_\alpha}[s_1]$ can ever enter A after s_1. This can only be due to the enumeration of some β-follower for some $\beta \supseteq \alpha^\frown o$ (again this can be due to either β or some β-sibling). If ξ was to the right of the true path then it is not hard to see that no β extending $\alpha^\frown o$ can have a follower enumerated at or after s_1. In the other case if the final ξ-follower was enumerated prior to s_1, then every negative node extending $\alpha^\frown o$ and $>_P \xi$ would be initialized when the ξ-follower was enumerated, and have to pick their followers larger than $u^\tau_{k,lb_\alpha}[s_1]$. This is because if a negative node is initialized, then within the same stage, it will not pick new followers (being in state 1).

Hence the box $\Psi^\tau(k, lb_\alpha)$ will receive a permanent definition. This completes the induction step for Claim 1. Hence A is ultrahigh. □

4. Not every high$_2$ c.e. degree can be split into totally ω-c.a. degrees.

In Section 2, we showed that every high c.e. degree could be split into lower c.e. degrees which are array computable. One might ask if high permitting was necessary; a weaker highness property such as nonlow$_2$ might

possibly suffice. Perhaps even every (non-zero) degree was the join of two array computable degrees. We show that this is not the case - highness is in fact necessary for splitting into array computable halves:

Theorem 4.1. *There is a high$_2$ c.e. degree $\boldsymbol{a}$ such that whenever $\boldsymbol{a} = \boldsymbol{a_0} \cup \boldsymbol{a_1}$, then one of $\boldsymbol{a_0}$ and $\boldsymbol{a_1}$ is not totally ω-c.a.*

We divide the discussion of the proof of the theorem in the following parts. First we discuss in Section 4.1, the plan on how to build a c.e. set A and make $deg(A)$ not contain the join of totally ω-c.a. sets. We then discuss in Section 4.2 why injecting highness requirements into the construction will fail, and discuss why high$_2$ requirements would be compatible.

4.1. **Making $deg(A)$ not contain the join of any totally ω-c.a. sets.** We now need to satisfy the requirements

$$\mathcal{N}_e : \text{If } A = \Gamma_e^{W_e \oplus V_e} \text{ and } W_e \oplus V_e = \Delta_e^A, \text{ then one of } W_e \text{ or } V_e \text{ is not totally } \omega\text{-c.a..}$$

We drop subscript e for the purpose of the following discussion. To satisfy $\mathcal{N}$ we need to ensure that there are *total* functions F^W, G^V computable from W and V respectively, such that one of F^W or G^V is not ω-c.a.. We remark that the proof to follow is *nonuniform* and necessarily so.

We fix an effective enumeration $\langle a_i, b_i \rangle$ of all possible ω-c.a. approximations. That is, each $a_i(-,-)$ is a total computable function, and $b_i(-)$ is a partial computable function. We say that a function F is *i-approximated* if b_i is total and for every x, the number of mind changes on $a_i(x,-)$ is bounded by $b_i(x)$, and $\lim_s a_i(x,s) = f(x)$. Every ω-c.a. function is i-approximated for some i.

The general strategy for making a set X not superlow is very similar to making X not totally ω-c.a.. Indeed it is easy to see that if X is a c.e. set, then X is superlow iff for every (partial) function Φ^X, there is some i such that Φ^X is i-approximated at every place it is defined. Our strategy in Section 3 was able to construct *partial functions* F^W and G^V such that one of them was not i-approximated for every i.

To demonstrate, for example, F^W is not totally ω-c.a., the idea would be to defeat all pairs a_i, b_i, by finding some argument x where we can change $F^W(x)$ more than $b_i(x) = k$ many times; waiting for the approximation $a_i(x,s)$ to agree with the current approximation $F^W(x)[s]$ before each such change. This is all as expected. In the superlow case, F^W need only be partial and only defined on some arguments.

The extra difficulty posed by having to ensure totality is the following: suppose we have selected n_1 and n_2 with the intention of making one of $F^W(n_1)$ or $G^V(n_2)$ change often. We have to define $F^W(n_1)$ and $G^V(n_2)$ even though the respective b_i-bounds might not converge. We might therefore set the use of $F^W(n_1)$ and $G^V(n_2)$ to be above $\varphi(x_1), \cdots, \varphi(x_k)$, but later after more of b_i has converged we might find that k was insufficient,

because we might have to force changes in W or V more than k times. We have to again exploit the fact we are allowed a nonuniform argument; that is, we are allowed to build more than one pair $\langle F^W, G^V\rangle$ of functions as possible candidates to meet $\mathcal{N}$.

The requirement $\mathcal{N}$ will build a pair of functions $\langle F^W, G^V\rangle$ which will be total in the case when the $\mathcal{N}$-hypothesis is true. The requirement $\mathcal{N}$ is divided into infinitely many subrequirements $\mathcal{N}_{i,j}$, and each of these subrequirements $\mathcal{N}_{i,j}$ will build a pair of functions $\langle F^W_{i,j}, G^V_{i,j}\rangle$. The subrequirement $\mathcal{N}_{i,j}$ will itself be divided into infinitely many sub-subrequirements $\mathcal{N}_{i,j,k,l}$, which are each aiming to ensure that, if the $\mathcal{N}$-hypothesis is correct, then either F^W is not i-approximated, or G^V is not j-approximated, or $F^W_{i,j}$ is not k-approximated, or $G^V_{i,j}$ is not l-approximated.

The atomic action for $\mathcal{N}_{i,j,k,l}$ is the following.

(1) Pick a follower (agitator) x for A, and fresh numbers n_1, n_2, n_3, n_4. Wait for $\gamma(\delta(x))[s]\downarrow$, and then define all of
$$F^W(n_1), G^V(n_2), F^W_{i,j}(n_3), \text{ and } G^V_{i,j}(n_4)$$
convergent with use $\gamma(\delta(x))[s]$. Freeze A. The role of the follower will be to induce changes into A and thus, indirectly, into one of W or V.

(2) Wait for all of $b_i(n_1), b_j(n_2), b_k(n_3), b_l(n_4)$ to converge. Enumerate x into A and wait for $W \oplus V$-change.

(3) One of W or V will have changed, and the following is completely symmetric. So suppose that W changed in step 2, so that now both $F^W(n_1)$ and $F^W_{i,j}(n_3)$ are undefined. Now pick $x_1, \cdots, x_{b_i(n_1)}$ ($b_1(n_1)$ many new agitators) such that $x_{m+1} > \gamma(\delta(x_m))$ for all m; define $F^W(n_1)$ on new use $\gamma(\delta(x_{b_i(n_1)}))$. Pick a fresh follower $\tilde{n}_2 > n_2$, and define $G^V(\tilde{n}_2)$ with use $\gamma(\delta(x_{b_i(n_1)}))$. Leave $F^W_{i,j}(n_3)\uparrow$. Increase restraint on A. Note that now, if we put the agitators into A in reverse order, provided that *all* the changes occur in W, we have enough to now kill a_1, b_1 on n_1.

(4) Wait for $b_j(\tilde{n}_2)$ to converge. Once it converges run the basic strategy in Section 3 to try and make F^W not i-approximated at input n_1. That is, we put the followers in reverse order each time we see a reconvergence of the $a_1(n_1)$ approximation agreeing with $F^W(n_1)[s]$. This process will succeed unless it is interrupted by a V-change. Go to next step if this happens.

(5) Now pick fresh followers $y_1, \cdots y_M$ (where $M = b_j(\tilde{n}_2) + b_k(n_3)$) such that $y_{m+1} > \gamma(\delta(y_m))$ for all m; define $F^W_{i,j}(n_3)$ and $G^V(\tilde{n}_2)$ with use $\gamma(\delta(y_{b_j(\tilde{n}_2)}))$. Increase restraint on A. Run the basic strategy in Section 3 to make either $F^W_{i,j}$ not k-approximated at input n_3, or G^V not j-approximated at input $\tilde{n}_2$.

If the $\mathcal{N}$-hypothesis holds, then all γ, δ-uses will converge and the appropriate followers will be chosen when required. Hence we will not wait forever

in steps 1 and 3. If we wait forever in step 2 then we would succeed trivially at one of $F^W, G^V, F^W_{i,j}$ or $G^V_{i,j}$. If we wait forever in step 4 then we either succeed trivially at $G^V(\tilde{n}_2)$, or we succeed at $F^W(n_1)$. If we ever get to step 5 then we will succeed at either $F^W_{i,j}(n_3)$ or $G^V(\tilde{n}_2)$.

The basic module succeeds at one of $F^W, G^V, F^W_{i,j}$ or $G^V_{i,j}$ as described above. Note that regardless of the outcome of the basic strategy, F^W and G^V will be defined at every input picked by the strategy. Furthermore if the basic strategy succeeds at neither F^W nor G^V, then both $F^W_{i,j}(n_3)$ and $G^V_{i,j}(n_4)$ are defined. Therefore the functions F^W, G^V built at $\mathcal{N}$ will be total regardless of the outcomes of individual sub-subrequirements. We analyze how the $\mathcal{N}$-requirement will be met. If $\forall i \exists j,k,l$ such that $\mathcal{N}_{i,j,k,l}$ succeeds at F^W, then F^W is not ω-c.a.. If $\exists i \forall j \exists k,l$ such that $\mathcal{N}_{i,j,k,l}$ succeeds at G^V, then G^V is not ω-c.a.. If neither of the alternatives above hold, then we must have some i_0, j_0 such that $\forall k,l$, $\mathcal{N}_{i_0,j_0,k,l}$ succeed at neither F^W nor G^V; hence both $F^W_{i_0,j_0}$ and $G^V_{i_0,j_0}$ will be total. In this situation if $\forall k \exists l$ such that $\mathcal{N}_{i_0,j_0,k,l}$ succeeds at $F^W_{i_0,j_0}$, then $F^W_{i_0,j_0}$ is not ω-c.a.; otherwise $G^V_{i_0,j_0}$ is not ω-c.a..

4.2. **Making A high$_2$.** Suppose we wanted to combine the requirements in Section 4.1 with highness requirements. Since each atomic node $\mathcal{N}_{i,j,k,l}$ only increases A-restraint finitely often, and changes A finitely often, it might seem trivial to combine this with highness requirements. However we know this to be impossible from Theorem 2.1, so what goes wrong? The conflict is between an $\mathcal{N}_{i,j,k,l}$-node σ, and a highness node η such that $\sigma \supseteq \eta * \infty$; i.e. σ believes that η wants to code an infinite column into A. Now σ will pick its follower(s) x_σ only at an η-expansionary stage; at such a stage s, η would have enumerated all the contents of a column up to a large number $s' > s$, and x_σ chosen less than s'. The trouble is that $\gamma(\delta(x_\sigma))$ may later converge to a value larger than s'. Remember we had to ensure that the functions $F, G, F_{i,j}, G_{i,j}$ had to be total; therefore we cannot afford to wait until σ is next visited and $\Gamma(\Delta(x_\sigma))$ becomes σ-believable before defining $F, G, F_{i,j}, G_{i,j}$ (note that we could do this in Theorem 3.1 because the functions built there were not required to be total). (Essentially $F, G, F_{i,j}, G_{i,j}$ need to be built at a single mother location τ on the tree on the true path. If this location is above the highness requirement, it cannot know when a computation associated with some $\mathcal{N}$ requirement below the highness requirement but associated with the mother ν. Hence at the next η-expansionary stage, η would be unable to code below $\gamma(\delta(x_\sigma))$ without injuring σ. This could happen for infinitely many different sub-subrequirement nodes extending $\eta * \infty$; which would be bad for η.

The inability to combine highness requirements with certain other requirements can sometimes be avoided if *capricious destruction* helps us make overall progress in the global setting; by doing so we get a high$_2$ set instead of a high set. For instance, no c.e. degree which does not bound a minimal

pair can be high, a result of Cooper [2], but one can overcome the difficulty of coordinating the local permitting with high coding by capriciously destroying certain computations with pending action. Thus it is possible to make a high_2 c.e. degree bounding no minimal pairs. The reason for this is that high_2-ness is like infinitely many highness requirements, but only almost all of them need succeed. It is possible to all some to fail to be met if this makes progress on the greater goal, typically by forcing some functional to diverge. We refer the reader to Downey, Lempp and Shore [8] for a more detailed discussion of a high_2 construction. We will only briefly describe the main points here.

To make A high_2 we need to code $Cof = \{e : dom(\varphi_e) \text{ is cofinite}\}$ into A''. This is done by the Double Limit Lemma. That is, we will need to build a functional $\hat{H}(e,i,t)$ such that for each e, $\lim_i \lim_t \hat{H}(e,i,t) = \text{Cof}(e)$. The idea is to split the $\hat{H}$ requirements into subrequirements H_τ built at a e-mother node τ as follows. A typical $\mathcal{P}_e$-mother node τ in the construction will measure if $e \in Cof$, and constructs a functional $H^A_\tau(i,t)$ such that H^A_τ is total and computable from A, and $e \in Cof$ iff $\lim_i \lim_t H^A_\tau(i,t) = 1$, and $e \notin Cof \Leftrightarrow \lim_i \lim_t H^A_\tau(i,t) = 0$. If we could do this, then to compute if $e \in Cof$, we observe that A'' can compute the true path of construction; so we can find the version of τ on the true path and then ask about the double limit.

The mother node τ will guess at the three quantifier question "$e \in Cof$" by breaking it up into infinitely many two quantifier questions, and approximating these individual questions at sub-nodes below it. To wit, each sub-node $\eta = \eta(i)$ will measure if there is $x > i$ such that $\varphi_e(x)\uparrow$. η will have an expansionary stage whenever it is visited and sees more of φ_e convergent above input i. η will then change A below some $H^A_\tau(i,-)$-use to try and redefine more of $H^A_\tau(i,-)$ to 1. At non-expansionary stages, η will extend $H^A_\tau(i,t)$ to more inputs t with output 0. Eventually η will want to ensure that if there are infinitely many η-expansionary stages then it records the $\lim_t H^A_\tau(i,t)$ as 1, otherwise it wants to record $\lim_t H^A_\tau(i,t)$ as 0

Note that in a high construction, this would be replaced by making sure $\lim_i H(i) = \text{Tot}(i)$ at nodes σ for e, *every* σ has to record the limit faithfully. However in a high_2 construction, we only need the limit $\lim_t H^A_\tau(i,t)$ to be recorded correctly at almost every level i below τ. We now discuss how to exploit this to make our construction work. Recall that the conflict was between a $\mathcal{N}_{i,j,k,l}$-node σ, and a high_2 node η such that $\sigma \supseteq \eta * \infty$. Let $\tau(\eta)$ be the mother node of η, and $\tau(\sigma)$ be the master[2] $\mathcal{N}$-node of σ. We describe two typical situations that will arise in the construction:

Case 1. $\tau(\sigma) \subset \tau(\eta) \subset \eta$*:* in this situation σ picks its follower x_σ as usual at η-expansionary stages. However whenever $\tau(\sigma)$ is visited and witnesses the convergence of $\gamma(\delta(x_\sigma))$, it will pre-empt η and immediately enumerate

[2]We refer to this mother as the master to avoid conflict in the discussion to follow. This is where the functional F, G are all defined.

all pending η-markers $< \gamma(\delta(x_\sigma))$ into A and make more of $H^A_{\tau(\eta)}(\eta, -) = 1$. That is, we attempt to force σ-correctness by removing any coding markers associated with η from below the use of the computations at σ, and capriciously destroy the $\Gamma(\Delta(x_\sigma))$-computation in the hope of clearing the η-markers from below the use. We then travel a link $(\tau(\sigma), \sigma)$ down to σ. This happens even at non-η-expansionary stages. If the link between $\tau(\sigma)$ and σ is travelled infinitely often and never removed, then we will be forced to make $\lim_t H^A_{\tau(\eta)}(\eta, t) = 1$ even though η might want to record the limit as 0; however observe that in this situation $\gamma(\delta(x_\sigma))$ tend to ∞, so that we have a global $\tau(\sigma)$-win. In the cone below σ, no other daughter node of $\tau(\eta)$ will be forced to have an incorrect limit. On the other hand once the link between $\tau(\sigma)$ and σ is removed, then before a new link can be formed between $\tau(\sigma)$ and another $\sigma' \supset \eta * \infty$, we must have a new η-expansionary stage. Therefore if we skip over η infinitely often by jumping from τ to σ' (for different σ''s), then we forced to make $\lim_t H^A_{\tau(\eta)}(\eta, t) = 1$ without a global $\tau(\sigma)$-win; however in this case η also wants to record the limit as 1, so we are fine. The point is that $\lim_t H^A_{\tau(\eta)}(i, t)$ will be forced to record an incorrect value for finitely many i.

Case 2. $\tau(\eta) \subset \tau(\sigma) \subset \eta$*:* We ensure that the following feature is present in the tree architecture: Suppose there are daughter nodes η_1, η_2 of $\tau(\eta)$ such that $\tau(\sigma) \subset \eta_1 * w_1 \subset \eta_2 * w_2$, and that $w_1 \neq w_2$. That is, we will have come to believe that e is not cofinite or the reverse. Whenever we see this we will restart a new version of $\tau(\sigma)$ below $\eta_2 * w_2$ (as we do in a global win); along the true path $\tau(\sigma)$ will be restarted finitely often because the sub-nodes of $\tau(\eta)$ can only have finitely much alternation in their true outcomes. Therefore if σ is an active sub-node of $\tau(\sigma)$, and $\tau(\sigma)$ is the final version of the requirement on the true path, then we may delay extending the definition of F^W, G^V until the next σ-stage when the uses become η-believable. Moreover, this feature means that in the configuration described all nodes η_i with $\tau(\eta) \subset \tau(\sigma) \subset \eta_i \subset \sigma$ will be saying the same thing, either $\eta_i * \infty$ (guessing cofinite *all from the same point onwards generated by some* $\eta_k * \infty \subseteq \tau(\sigma)$), or $\eta_i * f$. In the former case, when η_k looks correct, *all of the* η_i with $\tau(\sigma) \subset \eta_i \subset \sigma$, *also* look correct, and hence will have their coding markers enumerated. Thus when we travel the link either σ looks η_i correct, or we gain on the plan on proving that some $\tau(\sigma)$ hypothesis is false.

The formal construction below require links, and *scouting reports*. That is, before following a link from τ down to σ, we check to see where we would go if the link was not present; if we wanted to go left of σ, then we cancel the link and let the construction take us there. The construction will have the feature that there is a true path, and a *genuine true path* (GTP) which is the leftmost set of nodes actually visited infinitely often. This methodology was first introduced by Downey and Stob [9].

4.3. **The Priority Tree.** We mention that there is an issue with Case 1 above. In this case, it could be that we might create some link $(\tau(\sigma), \sigma)$

down to σ. and fail to clear the use of the computations of σ but drive one of the uses to infinity. Evidently this will need some kind of outcome g of the node σ drawing attention to the fact that we have a permanent link, and now we can ignore the requirements between $\tau(\sigma)$ and σ. Note that this means that information on the leftmost path can be false, but not on the genuine true path. Fortunately in this construction, we only have the situation where links can be permanent, and not infinitely often created. Thus a σ node like this has outcomes $\sigma * d$ (we diagonalize at σ, meaning that we successfully get through the whole diagonalization process of the basic module in a σ-correct way), this being stronger than $\sigma * g$ (meaning that we have killed $\tau(\sigma)$) and then right to $\sigma * w$ meaning that one of the a_z, b_z fail to live up to their responsibilities).

The priority tree is then constructed in the inductive way, using, say, lists as per Soare [13]. It is enough to introduce mother nodes τ for high_2-ness in order of e, so that if $e < f$ then $\tau_e \subset \tau_f$. Then, if η is a subrequirement of e, we ask that $\tau(\eta) \subset \eta$.

Now we similarly have the $\tau(\sigma)$ masters above the child σ-nodes.

As with typical $\mathbf{0'''}$ arguments, if we have an outcome $\sigma * g$ as above, then below g we would restart all nodes not associated with $\tau(\sigma)$ between $\tau(\sigma)$ and σ, and have *no* nodes associated with $\tau(\sigma)$ below $\sigma * g$. That is because we have a global win of the hypothesis of $\tau(\sigma)$ at σ.

In the same spirit, if we have an outcome $\eta_i * w_2$ which is an alternation of outcome for $\eta_j * w_1 \subseteq \eta_i * w_2$ with $\tau(\eta_i) = \tau(\eta_j)$ for the longest such τ, then we would restart all requirement of lower global priority than τ (i.e. $e(\nu) > e(\tau)$) below $\eta_i * w_2$.

With the rules above, generate the priority tree PT.

4.4. **The Construction.** The construction, as usual, is run in stages and substages. At each stage s, we begin at the empty string λ, and work our way down the tree to get the apparent GTP.

Suppose we hit a node ν

If ν is an η-node, see if its hypothesis is correct. If so then for all η-nodes with $\eta * \infty \preceq \eta'$ enumerate their coding markers into A, and we put $\eta * \infty$ as the next node on GTP_s. For any such η' if later we visit them then we will additionally put $\eta' * \infty$ on GTP. Move to the next substage to process $\eta * \infty$.

If ν is a σ-node, and has not yet been processed, assign 4 fresh followers $n_1, \ldots, n_4$ as in the basic module, and create a link $(\tau(\sigma), \sigma)$ back to σ's master. This concludes the stage.

If ν is a σ-node and all followers associated with it are σ-correct, and the module is in a waiting stage for the a, b's to respond, then play $\sigma * w$.

If ν is a σ-node and we are ready to attack, attack as in the basic module, linking back to $\tau(\sigma)$).

If ν is a σ-node and we have played previously $\sigma * d$ with σ-correct computations, and since that stage σ has not been initialized, then play $\sigma * d$.

If ν is $\tau(\sigma)$. If ν is not expansionary, play outcome $\nu * f$. If ν is τ-expansionary via τ-correct computations, then if there is no link from ν play $\nu * \infty$.

If there is a link (τ, σ), look at the nodes between τ and σ. If we were to pretend the link is not there, see if you would move left of σ. If this is the case, erase the link, and play $\tau * \infty$.

If not then we will travel the link. If the computations generated by the current state of the σ module (i.e. the basic module, as performed at σ) are σ-correct, perform the next step of the module. If this step actually meets σ, then play outcome $\sigma * d$, from σ. This ends the stage. Declare σ as met. If this step generates a new follower, generate the new follower, and then the stage ends at σ. If the computations are σ-correct, but we are now awaiting the $a_i, b_i, a_{i,j}, b_{i,j}$ to respond, erase the link, and play outcome $\sigma * w$.

If the computation is not σ-correct, then for any $\eta * \infty$ with $\tau \subseteq \eta * \infty \subseteq \sigma$, and $e(\tau(\eta)) > e(\tau(\sigma)$, capriciously enumerate all of η's unrestrained coding markers $\leq s$ into A, and play outcome $\sigma * g$.

If ν is a $\tau(\eta))$ node, then it only has one outcome on the tree, say, $\tau * o$ since this simply records the fact that this is the place we are considering e.

This ends the construction.

4.5. **Verification.** We define the GTP to be that leftmost collection of strings ν such that ν is visited infinitely often in the course of the construction. We prove by simultaneous induction that GTP is infinite,

(i) for each e there are only finitely many τ-nodes with $\tau \subset GTP$ and $e(\tau) = e$ (we call the longest on the *final* $\tau(e)$-master or -mother,
(ii) for each $\tau(\eta)$ mother there are only finitely many $\eta * w_1$ $\eta * w2$ nodes $\subseteq GTP$ with $w1 \neq w2$, (Note that this does not say "on GTP", but $\subseteq GTP$.)
(iii) $\sigma \subset GTP$ then one of $\sigma * w, \sigma * d, \sigma * g$ is on GTP. In the first two cases σ is met. In the final case $\tau(\sigma)$ is met and there are no σ'-nodes below $\sigma * g$ with $\tau(\sigma') = \tau$.
(iv) If $\eta \subset GTP$ one of $\eta * \infty$ or $\eta * f$ is a prefix of GTP. In the former case almost all of η's codings happen. In the latter, η is only finitely active.
(v) If $\tau \subset GTP$ is a final master node on GTP, then either $\tau * f \subset GTP$, or
  (a) either there is a $\sigma \supset \tau$ with $\tau(\sigma) = \tau$, and a link (τ, σ) for almost all stages, or
  (b) there are infinitely many $\tau * \infty$ stages, and all links with top τ are eventually traversed to their bottoms, σ' and either σ' is met, or infinitely often we are left of σ'.

We begin with λ which is on GTP. We might as well suppose that λ is a τ-master node and look at (v), which is the most complex hypothesis. Note that τ cannot be initialized by anything as it has highest priority. If $\tau * f \subseteq$

GTP_s, for almost all stages, then almost all stages are not expansionary for $\tau = \lambda$'s functionals, and hence $e(\tau)$ is met, and $\tau * f \subseteq GTP$.

If there are infinitely many τ-expansionary stages, at each such stage we will either play $\tau * \infty$ or traverse a link (τ, σ). This link can only be destroyed by playing some node ρ strictly left of σ (found by a scouting report), or σ playing outcome $\sigma * w$ or $\sigma * d$. In the former case we will play outcome $\tau * \infty$. In the case $\sigma * w$, we will have all followers σ-correct but one of the relevant a_i, b_j's fail to respond. At such at stage, the link will be removed, and the next τ-expansionary stage will be a $\tau * \infty$ stage. The same is true for the next τ-expansionary stage after we play $\sigma * d$. The only other possibility is that the link (τ, σ) is permanent. It is never erased by nodes left of σ but one of the finite number of followers associated with σ fails to achieve σ-correctness. In this case we see that $\sigma * g$ is played at each τ-expansionary stage, and hence $\sigma * g \subset GTP$. This means that we meet $e(\tau)$ at σ. Below $\sigma * g$ all the requirements of lower priority than η are restarted, and only a finite number of η with $\tau * \infty \subseteq \eta * \infty \subseteq \sigma$ might have the wrong outcome by capricious enumeration. Note that this establishes (v) and a bit more.

Similar arguments work for (iii). If $\sigma \subseteq GTP$, then from some stage onwards, when we visit σ we will act. The first time we do this we begin the analog of the basic module for σ. We link back to σ's master τ. For the basic $n_1, \ldots, n_4$ we will not erase this link until these have σ-correct computations. If that never happens, then, as above $\sigma * g \subseteq GTP$. If it does happen, then we will play $\sigma * w$ each time we visit σ unless realization occurs, and we wish to act. At such a stage, we again would link back to τ and the same reasoning says that either we get σ-correct computations for all the new followers, or the link is permanent, and in that case, again $\sigma * g \subset GTP$. Once they are established σ-correctly, then we will again play $\sigma * w$ until the a, b functions respond, and the cycle repeats, so that in the end we see that (iii) holds.

Now notice that for each $\tau(\eta)$ mother, if we have $\eta_k * w1 \subseteq \eta_j * w2 \subseteq GTP$ with $w1 \neq w2$, suppose that $w1 = \infty$ and $w2 = f$. This means that η_i is saying that from its point of view all the numbers greater than $i(\eta_k)$ are in $\mathrm{Cof}(e)$. How can this happen if $w2 = f$ since it believes that this is not the case for $i(\eta_j) > i(\eta_k)$, by the way we construct the priority tree? This can only happen if there is a permanent link over $\eta_k * \infty$ and this is *not a correct outcome*. Such a link must come from a master τ to some $\sigma \subset \eta_j * f$. For this to happen, it must be that $e(\tau(\sigma)) < e(\tau(\eta))$. Hence this can only happen finitely many times. For the final $\tau(\eta)$-mother it is only possible that $w1 = f$ and $w2 = \infty$ and then for *all* nodes $\eta_p \subset GTP$ below $\eta_j * w2$ it will be that $\eta_p * \infty \subset GTP$. Thus (ii) holds. This reasoning also proves (iv).

The above concludes the verification, and the argument we gave for λ also works for the final e master node τ.

5. Strong cupping

In this final section we look at stronger notions of cupping. We know that $\emptyset'$ is the join of two superlow degrees in the Turing degrees, but it is easy to show that this is not true in the wtt-degrees. That is because Downey, Jockusch and Stob [7] showed that the array computable wtt-degrees form an ideal in the c.e. wtt degrees, not containing $\emptyset'$. In this last section we prove something stronger. No c.e. set of totally ω-c.a. degree can even be wtt-cupped to the complete *wtt*-degree.

Theorem 5.1. *No totally ω-c.a. set can be wtt-cupped. That is, if $\emptyset' \leq_{wtt} A \oplus D$ and A is totally ω-c.a., then $\emptyset' \leq_{wtt} D$.*

Proof. Suppose A, D and Φ are given such that $\emptyset' = \Phi^{A\oplus D}$ and the use φ is computable. We first describe the strategy required if we assumed instead that A was superlow. We would want to compute $\emptyset'$ from D; let us consider only $\emptyset'(0)$. We would associate 0 with some $g(0)$ for some (partial) function $g \leq_T A$ which we get to build. Since A is superlow, we can request for certification that A is correct on the current use of $g(0)$; furthermore we have a computable bound on the number of wrong certifications which may be given, say with bound h. Therefore we could prepare in advance $h(0)$ many agitators $c_1, \cdots, c_{h(0)}$ targeted for $\emptyset'$ (more specifically we are building a c.e. set C which is hardwired into $\emptyset'$), and then compute $\emptyset'(0)$ from the first $\gamma(c_{h(0)})$ many bits of D. Namely, if 0 does enter $\emptyset'$ we want to force $D \upharpoonright \gamma(c_{h(0)})$ to change. We do this by enumerating $c_1, c_2, \cdots, c_{h(0)}$ into $\emptyset'$ one at a time. At each single attack, if $D \upharpoonright \gamma(c_{h(0)})$ changes then we are done; otherwise $A \upharpoonright \gamma(c_{h(0)})$ has to change, and we can get to redefine a new value for $g(0)$ at every attack we make. Since the number of wrong certifications for $g(0)$ is at most $h(0)$, one of the attacks must result in a D change, and we are done.

In the above argument, note that we can obtain the bound h effectively from g. Since g obviously can be made total, the above argument works even when A is only array computable. If A is now totally ω-c.a., we no longer can obtain the bound h effectively from g. To overcome this, we will have to "guess" at the correct bound, and build infinitely many reductions attempting to witness $\emptyset' \leq_{wtt} D$, based on each guess.

Formal proof. We may assume that φ is actually the number of bits of A (or the number of bits of D, if this is larger) which is accessed during the computation. By the recursion theorem, we can build some c.e. set C as part of $\emptyset'$ which we control, and assume that $C = \Phi^{A\oplus D}$. We fix an effective list $\langle f_e, h_e \rangle$ of partial computable functions such that f_e is binary and h_e is unary.

We perform infinitely many constructions, and argue that one of them works. The e^{th} construction will produce (uniformly in e) a total function

$g_e \leq_T A$, and partial computable function δ_e. The aim of the e^{th} construction is to satisfy the single requirement

$\mathcal{R}_e$: If $\lim f_e(\langle e, x\rangle, s) = g_e(x)$ for every x with at most $h_e(\langle e, x\rangle)$ many mind changes, then ensure $\emptyset' \leq_{wtt} D$ with computable use δ_e.

Construction. We now describe the e^{th} construction; for convenience we drop e from the notations. Since $C = \Phi^{A\oplus D}$, we will only act whenever $C = \Phi^{A\oplus D}$ looks correct on a sufficiently long segment. If we take any action in the construction that kills a current computation, we always wait until the relevant computations recover. We also assume that new Φ-computations converge instantly when required.

The parameters that are needed are the following: A list of agitators $c_{k,0}, c_{k,1}, \cdots$ targeted for entry into C, and we single $c_{k,0}$ out as the *leading k-agitator* for each k. We may assume that we fix the leading agitators $c_{k,0} = 2k$ for all k; the rest of the agitators will be picked during the construction from the odd integers. We also build a partial computable function δ, which will be total if the $\mathcal{R}$-hypothesis is correct. At stage $s = 0$, set everything to be undefined. At $s > 0$, we look for the least k such that one of the following holds.

(A1) $\delta(k)[s]\uparrow$ and $h(k)[s]\downarrow$. In this case pick fresh agitators $c_{k,1} < \cdots < c_{k,h(k)}$. Set $\delta(k) = \varphi(c_{k,h(k)})$.

(A2) *Correction needed*: there is some k such that $\delta(k)\downarrow$, k has entered $\emptyset'$ at some stage $t < s$ and $D \upharpoonright \delta(k)[t] = D \upharpoonright \delta(k)[s]$. We begin the sequence of attacks as follow (to force $D \upharpoonright \delta(k)$ to change): the first attack starts by enumerating $c_{k,0}$ into C, and waits for either $A \upharpoonright \varphi(c_{k,0})$ or $D \upharpoonright \varphi(c_{k,0})$ to change. If D changes then the attack is successful, otherwise if A changes, we wait for $f(x)$ to switch to output the current value of $A \upharpoonright \varphi(c_{k,h(k)})$ (which is different, since A has changed). We begin the second attack after $f(x)$ changes, and repeat with $c_{k,1}$. We stop whenever one of the attacks is successful.

If no such k exists, we do nothing.

Verification. We first of all verify that the e^{th} construction works. Let $\tilde{h}(k) := h_e(\langle e, k\rangle)$. Define $g_e(k)$ by the following. Go to the first stage s such that $A \upharpoonright \varphi(c_{k,0})[s] = A \upharpoonright \varphi(c_{k,0})$. If $\tilde{h}(k)[s]$ has not yet converged, then we output $A \upharpoonright \varphi(c_{k,0})$. Otherwise output $A \upharpoonright \varphi(c_{k,\tilde{h}(k)})$. Note that g_e is total computable in A, regardless of what happens in the construction. Suppose the hypothesis in $\mathcal{R}_e$ holds. First of all, we argue that for each k, whenever correction is needed under (A2) for k, one of the attacks must be successful. Suppose all of the $\tilde{h}(k) + 1$ many attacks fail. We must have $g_e(k) = A \upharpoonright \varphi(c_{k,\tilde{h}(k)})$, since the first attack fails. Each time a new attack fails, $f_e(\langle e, k\rangle)$ has to switch to give a new string, since f_e predicts $g_e(k)$ correctly and A is c.e. But $f_e(\langle e, k\rangle)$ will have to switch $\tilde{h}(k)$ many times, a contradiction.

It follows therefore that each k receives attention at most twice, and hence δ_e is total. To show that $\emptyset' \leq_{wtt} D$ with use δ_e, we fix k and let s be the first stage where $D \upharpoonright \delta_e(k)[s] = D \upharpoonright \delta_e(k)$. Clearly $k \in \emptyset' \Leftrightarrow k \in \emptyset'[s]$, because if k enters $\emptyset'$ after s, then correction will be needed for k, in which $D \upharpoonright \delta(k)$ will be forced to change after stage s.

Finally, we verify that the theorem holds. Let $g(\langle e, x\rangle) := g_e(x)$, which is total computable since g_e is generated effectively. There will be some e such that the pair f_e, h_e witnesses that g is ω-c.a., and hence satisfies the hypothesis in $\mathcal{R}_e$. As a final note, we remark that even though each $g_e \leq_{wtt} A$, but the amalgamation g is merely computable in A, as we might well expect. □

References

[1] M. Bickford and C. Mills. Lowness properties of r.e. sets. *Unpublished manuscript*, (1982).

[2] S. B. Cooper. Minimal pairs and high recursively enumerable degrees. *Journal of Symbolic Logic*, 39 (1974) pp. 655–660.

[3] Rod Downey and Noam Greenberg. Pseudo-jump inversion, upper cone avoidance, and strong jump-traceability. *Advances in Mathematics*, 237 (2013) pp. 252–285.

[4] Rod Downey and Noam Greenberg. A transfinite hierarchy of lowness notions in the computably enumerable degrees, unifying classes, and natural definability. *Submitted.*

[5] Rodney Downey and Denis Hirschfeldt, *Algorithmic Randomness and Complexity*, Springer-Verlag, 2010.

[6] Rod Downey, Denis Hirschfeldt, Andre Nies and Frank Stephan. Trivial reals. *Proceedings of the 7th and 8th Asian Logic Conferences*, (2003) pp. 103–131.

[7] Rod Downey, Carl Jockusch Jr. and Michael Stob. Array nonrecursive sets and multiple permitting arguments. *Proceedings of the Conference held at the Mathematisches Forschungsinstitut, Oberwolfach, March 19–25, 1989*, (1990) pp. 141–174.

[8] Rod Downey, Steffen Lempp and Richard Shore. Highness and bounding minimal pairs. *Mathematical Logic Quarterly*, 39(1) (1993) pp. 475–491.

[9] Rod Downey and Michael Stob. Minimal pairs in initial segments of the recursively enumerable degrees. *Israel Journal of Mathematics*, 100 (1997) pp. 7–27.

[10] Andre Nies. Lowness properties and randomness. *Advances in Mathematics*, 197 (2005) pp. 274–305.

[11] Keng Meng Ng. On very high degrees. *Journal of Symbolic Logic*, 73(1) (2008) pp. 309–342.

[12] Keng Meng Ng. Computability and Traceability. *PhD thesis, Victoria University of Wellington.*

[13] R.I. Soare. Recursively enumerable sets and degrees. *Springer-Verlag, Heidelberg* (1987).

School of Mathematics and Statistics, Victoria University of Wellington, PO Box 600, Wellington, New Zealand, , Division of Mathematical Sciences, School of Physical and Mathematical Sciences, Nanyang Technological University, Republic of Singapore