The Journal of Symbolic Logic
Volume 00, Number 0, XXX 0000

ON VERY HIGH DEGREES

KENG MENG NG

Abstract. In this paper we show that there is a pair of superhigh r.e. degree that forms a minimal pair. An analysis of the proof shows that a critical ingredient is the growth rates of certain order functions. This leads us to investigate certain high r.e. degrees, which resemble $\emptyset'$ very closely in terms of $\emptyset'$-jump traceability. In particular, we will construct an ultrahigh degree which is cappable.

§1. Introduction.

The motivation for this paper comes from four sources:

1. The study of the relationship between measure and degree.
2. The study of algorithmic randomness, in particular that of Kolmogorov complexity.
3. Questions arising from the study of pseudojump operators.
4. To develop techniques used in the construction of a proper subclass of the uniformly almost everywhere dominating degrees.

We prove the following two results, where the motivation, notation and terminology are discussed below.

Main Theorem 2.1. *There is an r.e. minimal pair of superhigh degree. That is, there are superhigh r.e. sets A and B, such that $\forall D(D \leq_T A \wedge D \leq_T B \Rightarrow D \equiv_T \emptyset)$.*

Main Theorem 3.1. *There is a cappable r.e. ultrahigh set. That is, there is an ultrahigh r.e. set A, and a non-recursive r.e. set B, such that $\forall D(D \leq_T A \wedge D \leq_T B \Rightarrow D \equiv_T \emptyset)$.*

Our first motivation comes from the study of the relationship between measure and degree. A is said to be almost everywhere (a.e.) dominating, if for almost all $X \in 2^\omega$, and all $g \leq_T X$, there is $f \leq_T A$ such that f dominates g. A is said to be uniformly almost everywhere (u.a.e.) dominating, if there is $f \leq_T A$, such that for almost all $X \in 2^\omega$, and all $g \leq_T X$, we have f dominating g. Kurtz [15] showed that $\emptyset'$ was u.a.e. dominating. Motivated by Kurtz, Dobrinen and Simpson [7] were led to study the class of a.e. dominating and u.a.e. dominating reals. They conjectured that the a.e. dominating reals and the u.a.e. dominating reals coincide, and that the degrees of such reals were precisely those above $\mathbf{0}'$. In [5], Cholak, Greenberg and Miller gave a direct construction of an incomplete r.e. set which is u.a.e. dominating, while Barmpalias and Montalbán [1] constructed a u.a.e. dominating degree which is half of a minimal pair. It turns out that the a.e. dominating degrees and the u.a.e.

Received February 14, 2007.

dominating degrees do coincide [13, 14], but such degrees are not always complete. However, it follows from a result of Martin [16] that the u.a.e. dominating degrees resemble $\emptyset'$ in that they are all high. The intuition is that not only are they high, but they resemble $\emptyset'$ very strongly.

Simpson [20] continued the theme of showing that the u.a.e. dominating reals resemble $\emptyset'$, by showing that every u.a.e. dominating degree is superhigh (i.e., $A' \equiv_{tt} \emptyset''$). In a fascinating connection between the theory of algorithmic randomness and effective measure theory, it was recently shown that u.a.e. dominating reals can be defined in terms of relativized Kolmogorov complexity. As we will see, this concerns the class of reals A such that $\emptyset'$ is K-trivial relative to A. Thus we see that the u.a.e. dominating reals arises quite naturally in two different ways.

The class of K-trivial reals was first introduced in [21]. They are defined as those reals A such that the K-complexity[1] of each initial segment of A is as low as it can be. That is, a real A is K-trivial, if

$$\exists c\, \forall n (K(A\restriction_n) \leq K(n) + c).$$

Every initial segment of A contains no more information than just its length. Such reals might seem very similar to the computable ones. Indeed, Chaitin [3] showed that if C was used[2] in place of K, the only C-trivial reals are the recursive ones. Solovay however, was the first to construct a non-recursive K-trivial, showing that the property of K-triviality was different from being recursive.

The K-trivials have aroused great interest in recent years, and are related to various other classes defined independently. In [8], Downey, Hirschfeldt, Nies and Stephan showed that the K-trivial reals are natural solutions to Post's problem in the sense that they are T-incomplete. In [17, 18], it was shown that "all is one" i.e., A is K-trivial iff A is low[3] for K iff A is Martin-Löf low.[4] Subsequently, more characterizations for the K-trivial reals were found, such as the bases of Martin-Löf randomness [10], and the reals low for weak 2-randomness.

How does all these relate to u.a.e. dominating? We say that a Δ^0_2 set A is almost complete (or $\emptyset'$-trivializing), if $\emptyset'$ is K-trivial relative to it, namely

$$\exists c\, \forall n (K^A(\emptyset'\restriction_n) \leq K^A(n) + c).$$

In upcoming work, Binns, Kjös-Hanssen, Miller and Solomon showed that for $A \leq_T \emptyset'$, A has u.a.e. dominating degree iff A is almost complete. Thus we find that the natural class of u.a.e. dominating reals coincide with the almost complete ones. Progress in understanding the relation of u.a.e. dominating reals to the degrees has been slow, and little is known [1, 5, 20]. This paper aims to contribute to the understanding of this class and its relationship with the Turing degrees.

In spite of the results of Cholak, Greenberg and Miller [5], and Barmpalias and Montalbán [1], there is no known construction of an almost complete real that combines with upper cone avoidance.[5] The only result related to cone avoidance

[1]We use $K(\sigma)$ to denote the prefix-free Kolmogorov complexity of the string σ. K^A denotes the relativized K-complexity with oracle A.

[2]C to denote plain complexity.

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

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

[5]That is, given any set B, there is an almost complete real A such that $A \not\geq_T B$.

is due to Nies and Shore [19]. Nies and Shore gave a direct construction of an r.e. almost complete real A, avoiding the upper cone of a K-trivial B.

The connection between K-triviality and u.a.e. domination allows for another construction of a u.a.e. dominating real using the pseudojump technique of Jockusch and Shore. A pseudojump operator V is one that takes each set X to the set $V(X) := X \oplus W^X$ r.e. in and above X (for a fixed r.e. set W). We also say that A completes the pseudojump operator V, if $V(A) \equiv_T \emptyset'$. Jockusch and Shore [12] proved the pseudojump inversion theorem, which states that every pseudojump operator has an r.e. non-recursive completion. Nies [19] showed that in fact every pseudojump operator has a ML-random completion. If we apply Nies' result to the construction of a K-trivial, we get a ML-random, almost complete degree.

There are a number of general questions about pseudojump operations and their ability to combine with various degree-theoretic constraints. In [6], Coles, Downey, Jockusch and LaForte proved that there is a pseudojump operator V non-trivial over the r.e. sets (i.e., for all r.e. sets W, we have $V(W) >_T W$), such that V does not avoid upper cones. They asked the question if V can be strengthened to be non-trivial over all sets, as any natural operator arising from relativizing constructions of r.e. sets must be. They also asked if it is true that every operator has a cappable[6] completion. We might hope that perhaps the construction of a K-trivial might give an operator which cannot avoid cones. In Theorem 3.1, we show that the operator arising from relativizing the construction of a strongly jump traceable (to be defined soon) has a cappable completion. The results of Barmpalias and Montalbán [1], and Cholak, Greenberg and Miller [5] are immediate corollaries to our theorem.

The questions regarding completions of pseudojump operators, and the results of [1] and [5] inspire us to study a proper subclass of the almost complete reals. An order function h is a recursive, non-decreasing and unbounded function. A set A is said to be jump traceable with order h, if there is a recursive g, such that for all x, $|W_{g(x)}| \leq h(x)$, and $J^A(x) \in W_{g(x)}$. (Here, $J^A(x)$ denotes the value of the jump function $\{x\}^A(x)$). This is a variation of the concepts of recursive traceability (introduced by Terwijn and Zambella [22]), and r.e. traceability (studied by Ishmukhametov [11]). The class of jump traceable reals was introduced by Nies [17], initially to study lowness properties. Nies [17] showed that the notions of jump traceability and superlowness (i.e., $A' \equiv_{tt} \emptyset'$, introduced by Bickford and Mills [2]) coincide for r.e. sets, but differed in general.

Nies [17, 18] showed that every K-trivial real is jump traceable (with an order function of growth rate $\sim h(n) = n \log n$). Inspired by these results, Figueira, Nies and Stephan [9] further characterized jump traceability in terms of C-complexity, and went on to define the notion of strong jump traceability: A set is strongly jump traceable, if it is jump traceable via all order functions. Figueira, Nies and Stephan showed the existence of a non-recursive strongly jump traceable r.e. set via a cost function construction similar to the construction of a K-trivial found in Downey, Hirschfeldt, Nies and Stephan [8].

Recently Cholak, Downey and Greenberg [4] showed that the r.e. strongly jump traceables form a *proper* sub-class of the K-trivials. The proper containment was shown by constructing a K-trivial real which cannot be jump-traced by an order

[6]A cappable degree is half of a minimal pair.

$\sim h(n) = \log\log n$. Applying the pseudojump inversion theorem to the construction of a strongly jump traceable, we get an incomplete r.e. set A, such that $\emptyset'$ is strongly jump traceable relative to it. That is, for every A-recursive order h^A, there is a uniformly A-r.e. sequence of traces $\{V_e^A\}_{e\in\mathbb{N}}$, such that for all e, $|V_e^A| \leq h^A(e)$ and $J^{\emptyset'}(e) \in V_e^A$. We call these reals *ultrahigh*. Using the relativized version of the results in [4], this new class of reals is seen to be a proper subclass of the almost complete reals.

The proofs of Main Theorems 2.1 and 3.1 not only involve an analysis of the growth rates of the order functions, but also require a careful scheduling procedure which decides when numbers are allowed to be enumerated and when the lengths of agreement are allowed to rise. We believe this may be of independent technical interest.

In Section 2 we will construct a minimal pair of r.e. superhigh sets. Shore has also proved the same result in unpublished work, where he has a different way to handle the thickness requirements of the construction of a minimal pair of high r.e. sets. In the the proof of Theorem 2.1, we will code *Tot* directly into the jump of the constructed set, such that the trace can be recovered in a *tt* way. We remark that both methods essentially run along the same lines; both methods will require keeping track of whether each minimal pair requirement is currently holding A-restraint, or holding B-restraint, and will involve coding the true path of the construction into the jump of the sets we are constructing. However, the presentation of the proof below was chosen, because it illustrates clearly the scheduling procedure of when lengths of agreements are allowed to rise—this is a crucial ingredient in the proof of Theorem 3.1, where we will combine the minimal pair requirements, with the requirements constructing an ultrahigh set.

This result says that while the questions surrounding minimal pairs of almost complete reals and minimal pairs of ultrahigh reals remain open, their answers will depend crucially upon the growth rates of the order functions. In Section 3 we will construct an r.e. ultrahigh set which is half of a minimal pair.

§2. A minimal pair of superhigh sets.

In this section, we will proof the following theorem.

THEOREM 2.1. *There is an r.e. minimal pair of superhigh degree. That is, there are superhigh r.e. sets A and B, such that $\forall D(D \leq_T A \wedge D \leq_T B \Rightarrow D \equiv_T \emptyset)$.*

2.1. Requirements. We build r.e. sets A and B satisfying the following requirements:

$$\mathscr{N}_e: \text{If } \Phi_e(A) = \Phi_e(B) = h \text{ is total, then } h \text{ is recursive,}$$
$$\mathscr{P}_e^A: e \in Tot \Leftrightarrow A' \vDash \sigma_e \text{ (for some truth table } \sigma_e),$$
$$\mathscr{P}_e^B: e \in Tot \Leftrightarrow B' \vDash \tau_e \text{ (for some truth table } \tau_e).$$

Here, we let Φ_e denote the e^{th} Turing reduction, and $Tot = \{e \in \mathbb{N} \mid q_e \text{ is total}\}$, where q_e is the e^{th} partial recursive function of a single variable. We will ensure that the sequences $\{\sigma_e\}_{e\in\mathbb{N}}$ and $\{\tau_e\}_{e\in\mathbb{N}}$ are recursive.

We adopt the convention of using uppercase Greek letters for functionals, and lowercase Greek letters for their use. The use of any convergent computation at a stage s is assumed to be bounded by s. We append $[s]$ to parameters, functionals or their use (e.g., $\Phi(A;x)[s]$) to describe their values at a stage s.

2.2. Strategy of a single requirement. There are two different types of requirements in this construction. The negative requirements $\mathscr{N}_e$ tries to make A and B a minimal pair by keeping numbers out of A and B. The positive requirements $\mathscr{P}_e^A$ and $\mathscr{P}_e^B$ tries to make $Tot \leq_{tt} A'$ or $Tot \leq_{tt} B'$ by attempting to control the configuration of an initial segment of A' or B'. As e goes in and out of $Tot[s]$, we will need to put numbers into A or B to force changes in A' or B'. The main conflicts we need to consider, are when some $\mathscr{P}_e^A$ wants to make a change in A below the use of a computation that some negative requirements might want to preserve.

We will firstly remind the reader of the strategy used to satisfy a single negative (minimal pair) requirement $\mathscr{N}_e$: We will define a (partial) recursive function h_e that computes the common value of $\Phi_e(A) = \Phi_e(B)$ (if they are equal). Whenever we observe $\Phi_e(A;x)[s]\downarrow = \Phi_e(B;x)[s]\downarrow$, we will set $h_e(x) = \Phi_e(A;x)[s] = \Phi_e(B;x)[s]$, and preserve *either* of $A\upharpoonright_{\varphi_e(A;x)}[s]$ or $B\upharpoonright_{\varphi_e(B;x)}[s]$. This allows us to have a period of time in which numbers are allowed to freely enter, say A (for the sake of the A-positive requirements) while we preserve $B\upharpoonright_{\varphi_e(B;x)}[s]$. When the destroyed A-computation recovers at some stage $s' > s$, we have $\Phi_e(A;x)[s'] = \Phi_e(B;x)[s'] = \Phi_e(B;x)[s] = h_e(x)$ and so the common value at each recovery stage is forced to be the same (so that $h_e(x) = \Phi_e(A;x) = \Phi_e(B;x)$). At stage s' we could now allow numbers to enter B while restraining A to give the numbers a chance to enter B (for the sake of the B-positive requirements).

Before we go any further, we would like to highlight the difference between a requirement making A high, and a requirement making A superhigh. If we were just trying to make A high, we would attempt to define a reduction $Tot = \Gamma^{A'}$. For each e, we have an associated $\gamma(e)$ use targetted at A', which we could control since we are building A. As e enters and leaves Tot (membership of Tot is a Σ_2^0/Π_2^0 fact), we have to put a stream of numbers into A to flip $A'(\gamma(e))$ back and forth. At times a negative requirement might block A, and prevent us from changing $A'(\gamma(e))$. When that happens we have to abandon the current value of $\gamma(e)$, and pick another one. The point is that as long as we limit the amount of negative restraint on A, this $\gamma(e)$ value eventually settles; that is all that really matters. In terms of the thickness requirements, this translates to the fact that we are allowed to miss finitely many numbers in the e^{th} column before we get to a stage where the requirement is never injured.

The reader will remember how this combines with the minimal pair requirements to produce a minimal pair of high r.e. sets—each high coding requirement experiences only a finite amount of A or B restraint. How different are our requirements in this case? We have to now code Tot into A' and B', while putting a recursive bound on the use. In other words, we have to count in advance, for each superhigh coding requirement, the number of times a negative requirement of a stronger priority will block its actions. If we directly adopt the strategy above we would be in trouble, for we cannot count in advance how many times a minimal pair requirement will choose to increase its A or B restraint before it hits an x where $\Phi^A(x) \neq \Phi^B(x)$. Extra care has to be taken to fix this problem—we require a careful scheduling of when a minimal pair requirement is allowed to increase its length of agreement (and hence increase the restraint it imposes). How this is arranged, and the impact it has on the rest of the construction, will be explained later.

Consider a single A-positive requirement, $\mathscr{P}_e^A$. We describe briefly exactly how we intend to carry out the coding. We fix in advance the index of two Turing functionals $\eta_f < \eta_\infty$ which we are enumerating, and the membership $Tot(e)$ will eventually be decided by looking at the configuration $A'(\eta_f)A'(\eta_\infty)$. Suppose at stage s, $Tot(e)[s] = 0$. We would then put η_f into $A'[s]$ with use $u(\eta_f, s)$ by enumerating the axiom $\langle A_s\upharpoonright_{1+u(\eta_f,s)}, \eta_f\rangle$ into Φ_{η_f}. Note that $u(\eta_f, s)$ is chosen larger than s, and therefore its entry into A later will only destroy those $\mathscr{N}$-computations which converge after stage s.

Suppose that after stage s, we never see an increase in $dom(q_e)$. Then, $Tot(e) = 0$ and $A'(\eta_f) = 1$. On the other hand if e enters $Tot[s']$ at a later stage $s' > s$, we could put $u(\eta_f, s)$ into A to take η_f out of A'. We would then put η_∞ into $A'[s']$ with use $u(\eta_\infty, s')$. If e enters and leaves Tot infinitely often, then $Tot(e) = 1$ and $A'(\eta_f)A'(\eta_\infty) = 01$, since once η_∞ is put into A', it is never removed (we are not considering any injury to $\mathscr{P}_e^A$ for the time being).

2.3. Interaction among strategies. We begin by considering a single A-positive requirement $\mathscr{P}$ working below a negative requirement $\mathscr{N}$. Suppose that at stage s, $\mathscr{P}$ puts η_f into A' with the use $u(\eta_f, s) > s$. At the next stage $s' > s$ where $dom(q)$ increases, $\mathscr{P}$ would want to take η_f out of A' and put η_∞ in. However, $u(\eta_f, s)$ might be less than the use of some computation $\Phi(A; x)[s']$ which had converged in the meantime (after stage s). If there had already been an enumeration into B below the use of $\Phi(B; x)$, then we would not be able to take η_f out of A' until the B-computation $\Phi(B; x)$ recovers. Unfortunately, if $\Phi(B; x)$ never recover after stage s', we would not be able to make $A'(\eta_f) = 0$.

The solution to this is the usual—put the requirements on a Π_2-guessing tree (which is possible, since the predicate "$e \in Tot$" is a Π_2^0 fact). There are now two versions of the requirement $\mathscr{P}$: Firstly, $\mathscr{P}^\infty$ which guesses that the hypothesis in the $\mathscr{N}$-requirement is true and hence will only act at those stages where both sides $\Phi(A; x)[s] = \Phi(B; x)[s]$ are convergent. The other version is $\mathscr{P}^f$, which guesses that $\Phi(A) \neq \Phi(B)$ and will only get to act when one of $\Phi(A; x)[s]$ or $\Phi(B; x)[s]$ is allowed to be injured, for some $x \in dom(h)$.

Now, $\mathscr{P}^\infty$ would handle the functionals with indices $\eta_f^\infty < \eta_\infty^\infty$, while the other version $\mathscr{P}^f$ of $\mathscr{P}$ have the functionals $\eta_f^f < \eta_\infty^f$. We set things up so that $\eta_f^f < \eta_\infty^f < \eta_f^\infty < \eta_\infty^\infty$, and use A' on these four values as the truth table. At each $\mathscr{N}$-expansionary stage, where the $\mathscr{N}$-hypothesis has been further verified, $\mathscr{P}^\infty$ would run its (modified) basic strategy:

1. If $dom(q)$ has increased since $\mathscr{P}^\infty$'s last action, restore the configuration $A'(\eta_f^f)A'(\eta_\infty^f)A'(\eta_f^\infty)A'(\eta_\infty^\infty) = 0001$.
2. If $dom(q)$ has not increased since $\mathscr{P}^\infty$'s last action, we force the configuration $A'(\eta_f^f)A'(\eta_\infty^f)A'(\eta_f^\infty) = 001$.

The reader should note that either of the actions taken above would result in an enumeration of a historical use into A (because of the first two bits of the truth table), which might be below the use of Φ^A-computations. This is alright since $\mathscr{N}$ only needs to preserve one of the two sides of the newly converged computations.

At those stages which are not $\mathcal{N}$-expansionary, that is, we are waiting for one of the two computations $\Phi^A(\max dom(h))$ or $\Phi^B(\max dom(h))$ to recover, and at the same time preserving the other one, $\mathcal{P}^f$ would be able to have a chance to run its basic strategy:

1. If $dom(q)$ has increased since $\mathcal{P}^f$'s last action, force the configuration $A'(\eta_f^f)A'(\eta_\infty^f) = 01$.
2. If $dom(q)$ has not increased since $\mathcal{P}^f$'s last action, set $A'(\eta_f^f) = 1$.

In 1. above, $\mathcal{P}^f$ would have to make an enumeration of a historical use $u(\eta_f^f)$ into A, which might be less than the A-restraint that $\mathcal{N}$ is currently putting up. This situation will arise from the following sequence of events: At the last $\mathcal{N}$-expansionary stage s, we had enumerated numbers into B instead of A. Thus even though $\mathcal{P}^\infty$ has had a chance to act at stage s, it did not do so in order to allow numbers into B. This is bad, for now $\mathcal{N}$ will increase its A restraint above $u(\eta_f^f)$. If $\mathcal{N}$ never sees a recovery on the B side, $\mathcal{P}^f$ would be stuck.

Note that if we were not required to make the reduction tt, we could simply let $\mathcal{P}^f$ move on to another index $\eta' > \eta_f^f$ and repeat. Unfortunately this is illegal in our case—we really have to make do with what we are given.

In order to overcome this difficulty, we will further split $\mathcal{P}^f$ into two versions, $\mathcal{P}^{f_A}$ and $\mathcal{P}^{f_B}$. Hence, the requirement $\mathcal{P}$ has now three different versions—$\mathcal{P}^\infty$ as above, and $\mathcal{P}^{f_A}, \mathcal{P}^{f_B}$ which respectively get to act at stages where $\mathcal{N}$ is holding A and B restraint. $\mathcal{P}^{f_A}$ now get to work with the indices η_f^A and η_∞^A, and $\mathcal{P}^{f_B}$ will work with the indices η_f^B and η_∞^B. We will code the totality of q into the configuration $A'(\eta_f^A)A'(\eta_\infty^A)A'(\eta_f^B)A'(\eta_\infty^B)A'(\eta_f^\infty)A'(\eta_\infty^\infty)$, a truth table of size 6. Depending on whether or not $e \in Tot$ when $\mathcal{P}^\infty, \mathcal{P}^{f_B}, \mathcal{P}^{f_A}$ are visited, each requirement above respectively tries to restore the configuration 000001 or $00001w$, $0001w'$ or $001w''$, and $01w'''$ or $1w''''$. For more details on the truth table, we refer the reader to Section 2.6.

To ensure that this strategy works, one will also need to carefully schedule when we allow the length of agreement for $\mathcal{N}$ to rise; more precisely we need to arrange when we extend $dom(h)$, which is the function computing the common value of both sides of the computations measured at $\mathcal{N}$.

Suppose $\mathcal{P}^{f_A}$ had already defined $u(\eta_f^A)$ when it acted at some stage where $\mathcal{N}$ was waiting for the recovery of $\Phi^B(x)$, where $x = \max dom(h)$. Suppose recovery occurs at the next $\mathcal{N}$-expansionary stage t, where we also have $\Phi^A(x+1)[t]\downarrow = \Phi^B(x+1)[t]\downarrow$, and we extend $dom(h)$ to include $x+1$. Although it is the case that $u(\eta_f^A, t) > \varphi(A; x)$, but $\Phi^A(x+1)$ can very well converge with a use larger than $u(\eta_f^A, t)$. This would be a problem if $\mathcal{P}^{f_A}$ wants to act before we have a chance to clear $u(\eta_f^A, t) < \varphi(A; x+1)[t]$.

However, we can see that at stage t, even though the length of agreement between Φ^A and Φ^B has increased, there is really no hurry to define $h(x+1)$ at stage t. The correctness and totality of h only matters if the $\mathcal{N}$-hypothesis is correct, in which case $\mathcal{P}^\infty$ would definitely get a chance to enumerate $u(\eta_f^A, t)$ into A at some time in the future. After $\mathcal{P}^\infty$ places $u(\eta_f^A, t)$ into A (and destroys the $\Phi^A(x+1)[t]$ computation at stage t), we could then wait until the next $\mathcal{N}$-expansionary stage

t'' when $\Phi^A(x+1)[t''] \downarrow = \Phi^B(x+1)[t''] \downarrow$ again, and see if the situation at stage t occurs again at stage t'' (i.e., $u(\eta_f^A, t'') < \varphi(A; x+1)[t'']$). The point is that if the $\mathscr{N}$-hypothesis is true, there must be an $\mathscr{N}$-expansionary stage v such that $\Phi^A(x+1)[v] \downarrow = \Phi^B(x+1)[v] \downarrow$, and the computation $\Phi^A(x+1)[v]$ is *believable*, that is, $u(\eta_f^A, v) \not< \varphi(A; x+1)[v]$. We can then extend the definition of h when $\Phi^A(x+1)[v]$ (and $\Phi^B(x+1)[v]$) become believable. Controlling the definition of h in this manner allows us to ensure that we never accept an agreement in the computations $\Phi^A(x+1) = \Phi^B(x+1)$, until we are certain that any followers below both uses will only get enumerated during $\mathscr{N}$-expansionary stages.

The steps taken by a general X-positive requirement $\mathscr{P}$ below a number of negative requirements is essentially the same. If $\mathscr{P}$ is arranged to be of a lower priority than k many $\mathscr{N}$-requirements, then there will be 3^k many different versions of $\mathscr{P}$; Each version of $\mathscr{P}$ acts at stages where its guess about the states of the $\mathscr{N}$-requirements above are correct.

We remark that the reader should really think of the positive requirements as acting on "boxes". Each of the 3^k many different versions of $\mathscr{P}$ will lie on the same level of the construction tree. Each of these nodes are assigned a different location of the jump $X'(\eta)$, i.e., a box. Thus, the truth table at this level is just the configuration of the row of 3^k boxes. Setting $X'(\eta) = 0$ is known as "emptying the box $X'(\eta)$", achieved by making an enumeration into X, while setting $X'(\eta) = 1$ is known as "filling the box $X'(\eta)$", achieved by enumerating a new axiom into that part of the jump. Thus a box has to be emptied before it can be filled with a new axiom.

When a version of $\mathscr{P}$ acts, it will empty all boxes assigned to the other versions of $\mathscr{P}$ to its right. It will either fill or empty its own box $X'(\eta)$ depending on the situation of *Tot*, thus setting up the truth table to look how it wants (see Section 2.6). This helps to give us a visual image of what is to come—as we will see, the main difficulty in Theorem 3.1, is that the ultrahigh coding requirements require us not only to place a bound on the number of boxes used, but all the positive requirements at the same level have to "share boxes", i.e., each version of the same $\mathscr{P}$ no longer have the luxury of working on its own boxes; two or possibly more of the positive requirements at the same level have to work on the same box.

On a final note, we remark that in the proof, one actually codes the true path into A' and B', because the configuration of each truth table recorded in the jump during the construction not only specifies the membership of *Tot*, but also records which outcomes were played infinitely often during the construction. This is a crucial refinement of the priority tree used to produce a minimal pair of high sets—in addition to guessing the outcomes and doing the coding based on these guesses, we also have to code in the state of the $\mathscr{N}$ nodes (i.e., whether they are holding A or B restraints).

2.4. Construction tree layout. The construction takes place on the full binary tree. Nodes of length $|\alpha| = 4e+1$ are assigned the requirement $\mathscr{P}_e^A$, while nodes of length $|\alpha| = 4e+3$ are assigned the requirement $\mathscr{P}_e^B$. The outcomes are labelled ∞ corresponding to a q_e-expansionary stage, and f corresponding to a non-expansionary stage for q_e. We place ∞ to the left of f.

Nodes α of length $|\alpha| = 2e$ are assigned the requirement $\mathscr{N}_e$. Instead of having two separate finite outcomes f_A and f_B, which are to be played when α is holding A and B restraints respectively, we will identify both outcomes together, call it f. We will however, need to introduce a separate variable *state*(α) (to be defined below) to record whether α is currently holding A or B restraint. Again we arrange the infinite outcome ∞ (stands for infinitely many $\mathscr{N}$-expansionary stages) to the left of the finite outcome f ($\mathscr{N}$ settles on a final restraint; whether the final restraint is on A or B is recorded in *state*(α)). The construction tree grows downwards, i.e., we say that α *is above* β, if $\alpha \prec \beta$.

We say that $\alpha <_{left} \beta$, if there is some $i < \min\{|\alpha|, |\beta|\}$, such that $\alpha\restriction_i = \beta\restriction_i$, $\alpha(i) = \infty$, and $\beta(i) = f$. That is, α is to the left of β on the construction tree.

A node α is said to be a $\mathscr{Q}$-node, if it is assigned the requirement $\mathscr{Q}$. α is a negative node, if α is a $\mathscr{N}_e$-node for some e. The node α is an X-positive node if α is a $\mathscr{P}_e^X$-node for some e. At each stage s during the construction, we will define the approximation to the true path, δ_s of length s. A node α is visited at stage s, if $\delta_s \succ \alpha$.

2.5. Notations. The symbol X is to be be used as a set variable, which will refer to either A or B. Let X^c be B if $X = A$, and vice versa. At each $\mathscr{N}_e$-node α, we will define a partial recursive function h_α. If $\Phi_e^A = \Phi_e^B$ is total, we will ensure that h_α is total and that $h_\alpha = \Phi_e^A = \Phi_e^B$. The function h_α initially starts off as $\emptyset$, and from time to time we will increase the domain of h_α, denoted by $dom(h_\alpha)$. We will not do this at every α-expansionary stage however, and will hold back until the relevant computations become believable.

For each $\mathscr{N}_e$-node α we use the notation $R(\alpha, s)$ to record whether α is holding A or B restraint at non-expansionary stages s; $R(\alpha, s) = X$ indicates that α is holding X-restraint at stage s. At each stage s of the construction, numbers will be enumerated into either A or B but not both. We call s an X-*stage*, if numbers are enumerated into X during construction stage s.

If δ is a node on the construction tree, there may be several negative nodes (say for instance $\tau_0, \dots, \tau_k$ such that $\tau_i^\frown f \prec \delta$, for $i \leq k$). At each stage s, each of the τ_i's might be trying to preserve an A or B computation. To keep a record of this fact, we define for each node δ and stage s, the string

$$state(\delta, s) = X_0 X_1 \dots X_{|\delta|-1},$$

where for all $i < |\delta|$, the value $X_i \in \{A, B, \infty, f\}$ is determined by the following: If $\delta\restriction_i$ is a positive node, let $X_i = \delta(i)$. If $\delta\restriction_i$ is a negative node such that $\delta(i) = \infty$, then let $X_i = \infty$. Otherwise we let $X_i = R(\delta\restriction_i, s)$.

The introduction of the *state*(α) variable require us to define new orderings $<_A$ and $<_B$, used to determine priority amongst the different possible *state*-values. We let $\infty <_A f <_A B <_A A$ and also define $\infty <_B f <_B A <_B B$. We extend $<_A$ and $<_B$ lexicographically to orderings $<_A$ and $<_B$ on $\{A, B, \infty, f\}^{<\infty}$. The orderings are defined this way to be consistent with the style of the ordering $<_{left}$ of the nodes on the construction tree—$\gamma <_A \sigma$ means that γ is lexicographically left of σ, i.e., of a higher $<_A$ priority.

At each odd level n of the construction, we have *boxes* η_∞^γ and η_f^γ for each $\gamma \in \{A, B, f, \infty\}^n$. These are actually Turing functionals which we define during the construction, and we let u_x^γ be the use of the box η_x^γ (henceforth, x is one of ∞ or f). Note that for each fixed n, the axioms for the different boxes η_X^γ form a uniformly r.e. set; hence we are able to compute an index for each box η_X^γ, which we also denote as η_X^γ. There are 2.4^n many boxes at level n, indexed by $\gamma \in \{A, B, f, \infty\}^n$. In addition, we call η_X^γ an *A-box* if $|\gamma| = 4e + 1$ for some e, and call it a *B-box* if $|\gamma| = 4e + 3$ for some e. To *empty* the X-box η_X^γ means that we enumerate the use u_X^γ into X. To *fill* the X-box η_X^γ at stage s with use u, means that we enumerate the axiom $\langle X_s\restriction_{u+1}, 0\rangle$ into η_X^γ, and set the use $u_X^\gamma = u$.

At an A-stage s if α of length n is visited (say α is A-positive), it will work on the boxes $\eta_f^{state(\alpha)}, \eta_\infty^{state(\alpha)}$. By this, we mean that α will fill the $\eta_\infty^{state(\alpha)}$ box and clear the $\eta_f^{state(\alpha)}$ box if it is an α-expansionary stage, and fill the $\eta_f^{state(\alpha)}$ box if it is not an α-expansionary stage. In either case α will also clear all γ-boxes of a lower $<_A$-priority than (or to the right of) the current boxes, i.e., $\gamma >_A state(\alpha)$. The variable $state(\alpha)$ should be viewed as a pointer, which points at the two boxes where α is currently working on. The truth table at level n will eventually be specified by the α on the true path, as well as the final value of $state(\alpha)$.

At level n, $state(\alpha)$ will point at different boxes at various stages of the construction, depending on the state of the negative nodes above it. However, at no time will an $state(\alpha)$ point at a box which another α' at the same level has used before; this means that the boxes at level n "are not shared" amongst the different α. As we will see, in the proof of Theorem 3.1, different α will have to share boxes, and two α at the same level might have to point at the same box.

All variables and parameters retain their assigned values until the next assignation. If the context is clear we omit the stage number from the parameters.

DEFINITION 2.2. For a negative node α, we say that a computation $\Phi^A(n)[s]$ with A-use w is *α-believable* at stage s, if

1. for all A-positive nodes β with $\beta^\frown\infty \preceq \alpha$, the box $\eta_f^{state(\beta)}$ is either empty, or has use $> w$,
2. for all $\gamma >_A state(\alpha)^\frown B$, both A-boxes $\eta_f^\gamma, \eta_\infty^\gamma$ are empty or have use $> w$.

Condition 1. ensures that there are no pending changes below the use w due to incorrectly filled boxes of higher priority, while condition 2. ensures that the A-boxes of lower $<_A$-priority will not be blocked by an increased α-restraint—these boxes can always be cleared when α is visited at an A-stage. A similar definition follows for B-computations, with B and $<_B$ in place of A and $<_A$.

At each $\mathscr{N}_e$-node α, we define the *length of agreement* between the e^{th} reductions of A and B, based on believable computations:

$$l(\alpha, s) = \max\{y < s \mid (\forall x < y) \ \ (\Phi_e^A(x)[s] \downarrow = \Phi_e^B(x)[s] \downarrow \text{ are both } \alpha\text{-believable computations})\}.$$

We say that a stage s is α-expansionary, if α is visited at stage s, and $l(\alpha, s) \geq |\,dom(h_\alpha)[s]|$. That is, we require only that $l(\alpha, s)$ is equal to $dom(h_\alpha)[s]$ to be expansionary, and not strictly greater. This is to ensure that certain boxes can always be emptied, and that correct computations eventually become α-believable.

For a $\mathscr{P}_e^X$-node α, we define $l(\alpha, s)$, the *length of convergence of q_e* as:

$$l(\alpha, s) = \max\{y < s \mid (\forall x < y) \ \ (q_e(x)[s] \downarrow)\}.$$

In this case, we say that s is α-expansionary, if α is visited at stage s and $l(\alpha, s) > l(\alpha, s^-)$ where $s^- < s$ is the largest stage such that α was visited at stage s^-.

2.6. The truth table. We define the truth table σ_e; a similar definition holds for τ_e with B and $4e + 3$ in place of A and $4e + 1$. We first label all the $\{A, B, f, \infty\}$-sequences of length $4e + 1$ by $\gamma_1 <_A \cdots <_A \gamma_n$. The truth table σ_e is then:

$A'(\eta_f^{\gamma_n})$	$A'(\eta_\infty^{\gamma_n})$	$A'(\eta_f^{\gamma_{n-1}})$	$A'(\eta_\infty^{\gamma_{n-1}})$	$\dots$	$A'(\eta_f^{\gamma_1})$	$A'(\eta_\infty^{\gamma_1})$
0	1	?	?	$\dots$	?	?
0	0	0	1	$\dots$	?	?
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
0	0	0	0	$\dots$	0	1

Thus, reading the truth table from left to right, we would see a string of zeroes, followed by a first entry with a 1. Everything that follows the first 1 is unhelpful garbage, and are accessed only when we visit the left of the true path, or when the states along the true path have not yet settled. To decide if $e \in Tot$ we see if this entry with the first 1 belongs to a η_f^γ box, or a η_∞^γ box. If it is the former then $e \in Tot$, otherwise $e \notin Tot$.

2.7. The construction. Each stage s of the construction will either be an A-stage enumerating into A, or a B-stage enumerating into B. When we pick a *fresh* number at a stage s, we mean that we pick a number $> s$ and $>$ any number used or mentioned so far. At stage $s > 0$, do the following in the order given:

1. Inductively define the stage s approximation to the true path, δ_s of length s. Suppose that $\delta_s\restriction_d$ has been defined for $d < s$. If s is $\delta_s\restriction_d$-expansionary, let $\delta_s(d) = \infty$. Otherwise, let $\delta_s(d) = f$.
2. Declare that s is an X-stage, where X is to be decided below. If $s = 0$ or $s = 1$, let $X = A$. Otherwise, look for the longest $\beta \prec \delta_s$ such that β has been visited prior to stage s. Let $s^- < s$ be the most recent stage such that $\delta_{s^-} \succ \beta$. If s^- is an A-stage, let $X = B$, otherwise let $X = A$.
3. Take actions for each expansionary negative node on δ_s. For each negative $\mathcal{N}_e$-node β such that $\beta^\frown\infty \preceq \delta_s$,
   (a) Set $R(\beta, s + 1) = X^c$ (since the X-side of the β-computations will be destroyed at this stage).
   (b) For each $n < l(\beta, s)$ we set $h_\beta(n) \downarrow= \Phi_e^A(n)[s]$, if not already defined.
4. Injure the negative nodes to the right of δ_s.
   (a) For each negative $\beta >_{left} \delta_s$, set $h_\beta = \emptyset$ and set $R(\beta, s + 1) = A$.
5. Clear all X-boxes of lower $<_X$-priority than the current state.
   (a) For each $\gamma >_X state(\delta_s)$, we clear the X-boxes $\eta_f^\gamma, \eta_\infty^\gamma$.
6. Take actions for each X-positive node on δ_s.
   (a) For each X-positive node β such that $\beta^\frown\infty \preceq \delta_s$, clear the box $\eta_f^{state(\beta)}$.
   (b) For each X-positive node $\beta \prec \delta_s$, fill the box $\eta_x^{state(\beta)}$ with fresh use $u > s$, where $x = \delta_s(|\beta|)$.

2.8. Verification. Let $TP = \liminf_s \delta_s$ be the true path of the construction under $<_{left}$, i.e., the leftmost path visited infinitely often. At each stage s, numbers (possibly none) are enumerated into either A or B but not both.

For any node α, the state of the nodes above α does not change unless the construction visits left of α. Hence if α is on the true path, then $state(\alpha)$ eventually settles at some value γ, pointing to the two boxes η^{γ}_{f} and η^{γ}_{∞}. We also point out that during the construction, any box filled at stage s is filled with a use $> s$. Hence any convergent computation $\Phi^{X}(n)[s]\downarrow$ at stage s, can only be destroyed later due to the enumeration of an X-box use, which was filled prior to stage s.

We begin with a preliminary lemma:

Lemma 2.3. *Suppose α is visited infinitely often. Then for each $X \in \{A, B\}$, there are infinitely many X-stages s such that $\delta_s \succ \alpha$.*

Proof. If α is visited at an A-stage s, then the next visit to α would be a B-stage, unless some $\beta \succ \alpha$ causes us to choose $X = A$ at step 2 of the construction. Once all such β are visited at least once after stage s, we would have to choose $X = B$ at step 2. ⊣

Hence any α on the true path will be visited at infinitely many A-stages, as well as at infinitely many B-stages. This gives every positive node on the true path infinitely many chances to act.

Lemma 2.4. *Along the true path of construction, all the negative requirements are satisfied.*

Proof. Let α be an $\mathcal{N}_e$-node on true path, such that $\Phi^{A}_{e} = \Phi^{B}_{e}$ is total. Let s_0 be a stage large enough such that the construction never visits left of α after that. After s_0, h_α is never injured and so it is recursive.

Claim 2.4.1. *In between α-expansionary stages, at least one side of the Φ_e-computations is preserved. That is, if $s_1 \geq s_0$ is an α-expansionary X-stage such that $X^{c}_{s_1}\restriction_m \neq X^{c}_{s_2}\restriction_m$ for some $s_2 > s_1$, where $m =$ the use of $\Phi^{X^c}_{e}\restriction_{dom(h_\alpha)}[s_1]$, then there must be an α-expansionary stage u, such that $s_1 < u < s_2$.*

Proof of claim. Suppose on the contrary there are no such u. Since s is an X-stage, the number $k < m$ has to enter X^c at some stage t, where $s_1 < t < s_2$. By assumption that there is no such u, we have $\delta_t >_{left} \alpha^\frown\infty$. The number k has to be enumerated under step 5(a) or 6(a) of the construction at stage t. Since $R(\alpha, t) = X^c$ it follows that $state(\delta_t, t) >_{X^c} state(\alpha, s_1)^\frown X$. However, due to the α-believability of the $\Phi^{X^c}_{e}$-computations at stage s_1, neither of 5(a) nor 6(a) is possible. ⊣

Claim 2.4.2. *There are infinitely many α-expansionary stages.*

Proof of claim. The idea is that we will show that any correct computation has to eventually become α-believable, using Lemma 2.3. Suppose there are only finitely many expansionary stages, and let s_1 be the last α-expansionary stage. Each time the construction visits left of α, we will set $h_\alpha = \emptyset$, so we certainly have $s_1 \geq s_0$ as well as $h_\alpha = h_{\alpha,s_1} \neq \emptyset$. Suppose that s_1 is an A-stage, a similar argument can be run if s_1 is a B-stage instead.

Let $s_2 > s_1$ be a stage such that $A_{s_2}\restriction_{m'} = A\restriction_{m'}$ where $m' =$ use of $\Phi^{A}_{e}\restriction_{dom(h_\alpha)}$. By Claim 2.4.1, it follows that $B_{s_1}\restriction_m = B\restriction_m$ where $m =$ use of $\Phi^{B}_{e}\restriction_{dom(h_\alpha)}[s_1]$. Since B has not changed below m, and $state(\alpha)$ never changes after stage s_1, it follows that the $\Phi^{B}\restriction_m$ computations must be still believable at stage s_2. Hence, if s_2 is not expansionary, it must be because the computation $\Phi^{A}_{e}(j)[s_2]$ is not α-believable, for some $j \in dom(h_\alpha)$. This means that at stage s_2, one or both of the following is true:

1. There is an A-positive node β such that $\beta^\frown\infty \preceq \alpha$, such that the box $\eta_f^{state(\beta)}$ has use $< m'$.
2. For some $\gamma >_A state(\alpha)^\frown B$, the box η_x^γ has use $< m'$.

For (2), note that at the last α-expansionary A-stage s_1, all such boxes are cleared. Hence the box η_x^γ has to be filled after stage s_1. However note that $R(\alpha)$ is set to B at stage s_1 and can only change if we visit left of α, or have an α-expansionary stage, neither of which is possible after stage s_1. So, it follows that $\gamma >_A state(\alpha)$ because only the nodes $\beta >_{left} \alpha$ can fill such a box. This case, together with (1) is impossible because α will eventually be visited again at an A-stage, and enumerate into A below m', a contradiction.

Thus, the Φ_e^A-computations at stage s_2 must also be α-believable, hence s_2 is α-expansionary. ⊣

Note that Claim 2.4.2 is not enough to show that h_α is total, as $l(\alpha, s)$ need not be strictly increasing at expansionary stages. The fact that h_α is total follows from the fact that any correct computation must eventually become α-believable:

Claim 2.4.3. *h_α is total.*

Proof of claim. Let p be any number, and we want to show that $h_\alpha(p)$ eventually receives a definition after stage s_0. Consider a stage $s_3 > s_0$ such that $A_{s_3}\upharpoonright_{m''}$ and $B_{s_3}\upharpoonright_{m''}$ are correct up to m'', where m'' = larger of the two uses $\Phi_e^A\upharpoonright_{p+1}$ and $\Phi_e^B\upharpoonright_{p+1}$. Now s_3 has to be α-expansionary with $l(\alpha, s_3) > p$, otherwise there is some X-box with pending action below m''. Such a box will be cleared at the next α-expansionary X-stage, causing a change in $X\upharpoonright_{m''}$, a contradiction. ⊣

Thus h_α is total, and computes correctly since one side of $\Phi_e^A = \Phi_e^B$ is always preserved between expansionary stages. ⊣

Lemma 2.5. *Along the true path of construction, all the positive requirements are satisfied.*

Proof. We consider the A-positive requirements, a similar argument follows for the B-positive requirements. Let α be a $\mathscr{P}_e^A$-node on the true path, and σ_e is the truth table in Section 2.6. We want to show that q_e is total iff $A' \models \sigma_e$. Let s_0 be a stage after which we never visit left of α, also let $\gamma_0 = state(\alpha, s_0)$, the final state value.

It is clear that every A-box η_x^γ for every $\gamma >_A \gamma_0$ cannot be permanently defined, because we will clear such a box at each visit to α at an A-stage larger than s_0. Therefore we only need to show that

- q_e is total $\Rightarrow \eta_\infty^{\gamma_0}$ is permanently defined, while $\eta_f^{\gamma_0}$ is never permanently defined.
- q_e is not total $\Rightarrow \eta_f^{\gamma_0}$ is permanently defined.

We begin by assuming that q_e is total, hence there are infinitely many α-expansionary A-stages. At these stages, $\eta_f^{\gamma_0}$ has to be emptied, so it is never permanently defined. Suppose that $\eta_\infty^{\gamma_0}$ is filled at the α-expansionary A-stage $s_1 > s_0$, with use $u_\infty^{\gamma_0}$. We must show that it is never emptied after that; suppose on the contrary that some number $k \leq u_\infty^{\gamma_0}$ is enumerated at the A stage $t > s_1$. Suppose k is the use u_x^γ. The possibilities for γ are:

1. $\gamma \succ \gamma_0{}^\frown\infty$: Since uses are always chosen fresh, it follows that the box η_x^γ must have been filled at some α-expansionary A-stage $s_2 < s_1$, in which the box

$\eta^{\gamma_0}_{\infty}$ must also be filled with use $u^{\gamma_0}_{\infty}[s_2] < u^{\gamma}_{x}[s_2]$, if it is not already occupied. Hence we have to empty the box $\eta^{\gamma_0}_{\infty}$ after s_2, before it can be filled at stage s_1, but this action would also empty the box η^{γ}_{x}, contrary to assumption.

2. $\gamma >_A \gamma_0{}^\frown\infty$: Such a box would be emptied at s_1, and even if they were filled later, the use would have to be larger than $u^{\gamma_0}_{\infty}$.
3. $\gamma <_A \gamma_0$: If this applies, then the construction has to visit left of α at stage t.
4. $\gamma \preceq \gamma_0$: Then, k is the use of some box $\eta^{state(\beta)}_{f}[t]$ for some β with $\beta^\frown\infty \preceq \alpha^\frown\infty$. As above, such a box would have to be emptied at stage s_1.

So, $\eta^{\gamma_0}_{\infty}$ has to remain permanently defined. If q_e is not total, a similar argument applies to show that $\eta^{\gamma_0}_{f}$ will be permanently defined. ⊣

§3. A cappable ultrahigh set.

In this section, we construct an ultrahigh r.e. set A, which is half of a minimal pair:

THEOREM 3.1. *There is a cappable r.e. ultrahigh set. That is, there is an ultrahigh r.e. set A, and a non-recursive r.e. set B, such that $\forall D(D \leq_T A \wedge D \leq_T B \Rightarrow D \equiv_T \emptyset)$.*

3.1. Requirements. We build the r.e. sets A and B with B coinfinite, satisfying the following requirements.

$$\mathscr{N}_e : \text{If } \Phi^A_e = \Phi^B_e = h \text{ is total, then } h \text{ is recursive,}$$
$$\mathscr{P}^A_e : \text{If } \Phi^A_e \text{ is an order, make } \emptyset' \text{ } A\text{-jump traceable via } \Phi^A_e,$$
$$\mathscr{P}^B_e : |W_e| = \infty, \text{ make } B \cap W_e \neq \emptyset.$$

Here, we let Φ_e denote the e^{th} Turing reduction, and $J^{\emptyset'}(k)$ denote the value of the universal $\emptyset'$-partial recursive function $\Phi^{\emptyset'}_k(k)$. If Φ^A_e is an order, the requirement $\mathscr{P}^A_e$ will build an A-u.r.e. sequence $\{V^A_k\}_{k\in\mathbb{N}}$ such that for all k, $|V^A_k| \leq \Phi^A_e(k)$ and $J^{\emptyset'}(k) \in V^A_k$. To do this, $\mathscr{P}^A_e$ will divide the task into infinitely many substrategies, or modules. The the k^{th} module will be responsible for building V^A_k. The requirement $\mathscr{P}^A_e$ will be split into infinitely many subrequirements $\mathscr{P}^A_{e,0}, \mathscr{P}^A_{e,1}, \dots$. Each subrequirement will be responsible for ensuring the success of finitely many of these modules. The modules that each subrequirement $\mathscr{P}^A_{e,i}$ is allocated will change from time to time, during the construction, and depends on the values of $\Phi^A_e[s]$.

3.2. The atomic strategy. The negative requirements $\mathscr{N}_e$ are as before—at each node α assigned a negative requirement $\mathscr{N}_e$, we will build a partial recursive function h_α, that records the common value of $\Phi^A_e = \Phi^B_e$ observed at α-expansionary stages. We ensure that in between α-expansionary stages, at least one side of the Φ_e-computations is preserved.

We first outline the strategy to make A ultrahigh. For simplicity, let us first consider the situation in which we want to trace $J^{\emptyset'}(k)$ with bound $f(k)$ for some recursive order f. That is, we only need $\emptyset'$ to be jump traceable relative to A via a single bound f. We are building an A-uniformly r.e. sequence $\{V^A_k\}_{k\in\mathbb{N}}$ so that for all k, we have $J^{\emptyset'}(k) \in V^A_k$, and $|V^A_k| \leq f(k)$. In actual fact, we are enumerating an A-recursive functional $\Psi^A(k,n)$ such that for every k, the range $\{\Psi^A(k,n) \mid n \in \mathbb{N}\} = V^A_k$. Therefore, if we enumerate a certain number p into V^A_k with use u, at a later stage we are allowed to change our mind. That is,, we can *remove p from V^A_k*, at a cost of putting u into A. If we were instead building a plain u.r.e. sequence (without oracle A), we would be unable to change our mind in this way; any enumerated number has to increase the size of the trace permanently.

The freedom for us to change our mind could be exploited in the following way: Whenever the opponent plays $J^{\emptyset'}(k)[s]\downarrow$, we could trace the value $J^{\emptyset'}(k)[s]$ into V_k^A. When the opponent next changes the value of $J^{\emptyset'}(k)[t]$ and shows us a new value (note that he could in fact do this infinitely often, so that $J^{\emptyset'}(k)\uparrow$), we could remove the earlier trace and replace it with the new one.

Things are not so simple, of course, for if we could always do this, then $|V_k^A| = 1$ for all k, so that A is already Turing complete. We will have to remember that there are negative requirements which might be imposing A-restraint, at the time when our opponent shows us a new value to be traced. Therefore, at times we might be forced to leave a wrong trace in V_k^A, and continue with the strategy using a new location. We return to the "box" intuition used in Theorem 2.1. This time round, we think of each traced $J^{\emptyset'}(k)$ value as occupying a "box" location. Hence each V_k^A is allowed $f(k)$ many locations, or boxes which we may fill with potential candidates for $J^{\emptyset'}(k)$. If the positive strategy fills a box with an incorrect value, and at a later stage, the negative requirements above it decide to drop the A-restraint, we could go back and clear the box by removing the wrong trace. In this way we are able to re-use the box in future.

We will discuss the problems faced, and the solutions, for two typical scenarios in the construction: Firstly we describe how a negative requirement might survive the infinite positive action of some A-positive requirement living above it (of higher priority). Secondly, we will talk about how an A-positive requirement living below (of lower priority than) a negative requirement survives the A-restraint put on it. As in Theorem 2.1, the construction tree grows downwards.

Scenario 1: *$\mathscr{N}$ living below $\mathscr{P}^A$.*

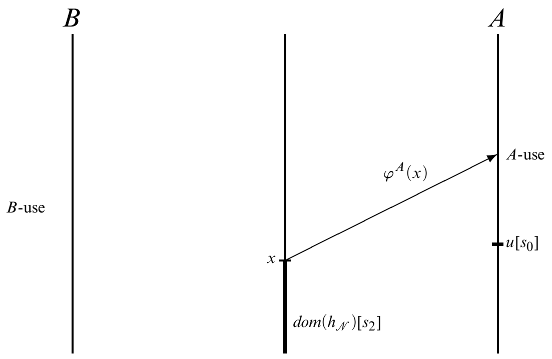
We consider a requirement $\mathscr{P}^A$, devoted to tracing $J^{\emptyset'}(k)$ into V_k^A. Since $\mathscr{P}^A$ might make infinitely many enumerations into A, we have to arrange for it to be equipped with two outcomes: Firstly, we need to have the infinitary outcome ∞, which represents the situation in which the opponent shows us infinitely many different values to be traced. The other outcome is finite f, which says that the opponent eventually makes up his mind on a certain value (i.e., $J^{\emptyset'}(k)\downarrow$). In this case, $\mathscr{P}^A$ would trace the final value and never remove it from the appropriate box.

The purpose of equipping $\mathscr{P}^A$ with two outcomes is so that we can have two versions $\mathscr{N}$ and $\widehat{\mathscr{N}}$ of the same negative requirement below $\mathscr{P}^A$, guessing the outcomes ∞ and f respectively. Thus, $\mathscr{N}$ will never believe in a Φ^A-computation until $\mathscr{P}^A$ clears the relevant boxes with use below the Φ^A-computations. This is alright, because correct Φ^A-computations must eventually become believable. On the other hand $\widehat{\mathscr{N}}$ will always believe in new Φ^A-computations whenever they show up—it acts on the guess that $\mathscr{P}^A$ never needs to clear any of its boxes.

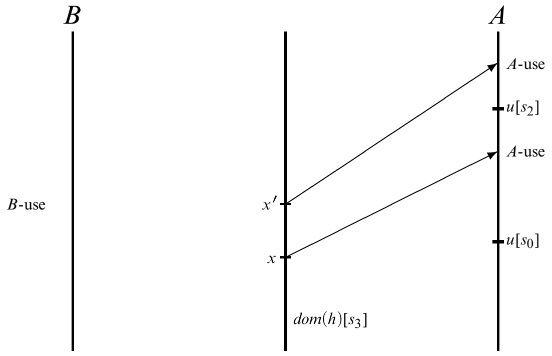
Scenario 2: *$\mathscr{P}^A$ living below $\mathscr{N}$.*

There are two versions of the positive requirement—$\mathscr{P}^A$ which guesses that there are infinitely many $\mathscr{N}$-expansionary stages, and $\widehat{\mathscr{P}^A}$ which guesses otherwise.

At stage s_0, suppose that $\widehat{\mathscr{P}^A}$ gets to act while $\mathscr{N}$ is waiting for the recovery of an A-computation $\Phi^A(x)$ for some $x \in dom(h)$. Suppose further that at stage s_0, $\widehat{\mathscr{P}^A}$ traces the current value $J^{\emptyset'}(k)[s_0]$ into V_k^A with use $u[s_0]$. At some later stage $s_1 > s_0$, we might get recovery of $\Phi^A(x)$ (with an A-use above $u[s_0]$). It might well be the case that at stage s_1, enumeration is made into B and not A (we have to allow for the B-restraint to be dropped infinitely often). When $\widehat{\mathscr{P}^A}$ gets to act later at stage $s_2 > s_1$, it would not be able to clear the box should the opponent challenge it to, since $\mathscr{N}$ has dropped B-restraint and is currently holding A-restraint above $u[s_0]$. The following diagram describes the situation at stage s_2:



What $\widehat{\mathscr{P}^A}$ needs to do now is to trace the new value into a second box, say with use $u[s_2] > \varphi^A(x)[s_2]$. Suppose the situation repeats again, namely, at a later stage the B-computation $\Phi^B(x)$ recover, and we have a length of agreement larger than some new $x' > x$, with $\varphi^A(x') > u[s_2]$. $\mathscr{N}$ then drops B restraint again and when $\widehat{\mathscr{P}^A}$ next gets to act at stage s_3, it finds itself in the following situation:



This would be bad, for $\widehat{\mathscr{P}^A}$ would have to move on and occupy yet another box. The solution to this, as in Theorem 2.1, is to delay extending $dom(h)$ until the Φ^A-computations become *believable*. When $\mathscr{N}$ first saw the length of agreement increase beyond x, it should not have expanded $dom(h)$ to include x', for $u[s_2]$—the use of the second box—was actually below $\varphi^A(x')$. $\mathscr{N}$ could actually wait until the next expansionary A-stage, where both of the boxes can be cleared. The point is, once again, that if $\Phi^A(x') = \Phi^B(x')$, then eventually $\mathscr{N}$ would be able to place x' into $dom(h)$.

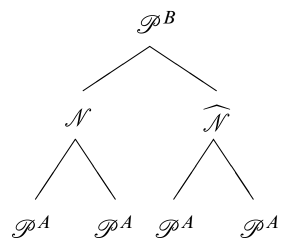
The reader might ask, what exactly is the difference between this (ultrahigh) strategy and the previous (superhigh) strategy, for the same method was used to decide when we allow $dom(h)$ to rise. The above discussion shows that the requirement with the two different versions $\mathscr{P}^A$ and $\widehat{\mathscr{P}^A}$ at that level, requires 2 boxes to run its strategy. Generally if a positive requirement lives on level m of the construction tree with 2^m different versions, it will need 2^m many boxes to handle all the possible combinations of A/B-restraint each negative requirement above it is currently holding. If the tracing order $f(n)$ has an exponential growth rate of $\sim 2^n$, the above strategy would work fine, because we could alternate the positive and negative strategies in the priority order $\mathscr{P}^A < \mathscr{N} < \mathscr{P}^A < \mathscr{N} < \dots$. At the $2k^{th}$ level, the positive requirement will be responsible for tracing $J^{\emptyset'}(k)$, and have have 2^{2k} many different versions. Each of the 2^{2k} many different versions of the positive requirement will be allocated a different box, and each version will work on tracing the jump value into its own box when it is visited in the construction. So we will need altogether something like $2^{2k} \sim f(k)$ many boxes at level $2k$. The reader will note that this is actually the situation in Theorem 2.1—the proof in Theorem 2.1 can be easily seen to be one which also produces a minimal pair A, B, such that $\emptyset'$ is jump traceable relative to both A and B via an order function of exponential growth rate.

The problem now of course, is that f is given to us—it is an arbitrarily slow-growing order. The main difference is that we cannot arrange our requirements as in the case when $f(n) \sim 2^n$. The cost of having an extra negative requirement above a particular $\mathscr{P}^A$ would have the effect of doubling the number of boxes it needs, because of the extra information we have to keep track of. We would have to arrange the requirements in the following order instead:

$$\underbrace{\mathscr{P}^A < \cdots < \mathscr{P}^A}_{\tilde{f}(1)\text{ many}} < \mathscr{P}^B < \mathscr{N}$$
$$< \underbrace{\mathscr{P}^A < \cdots < \mathscr{P}^A}_{\tilde{f}(2)-\tilde{f}(1)\text{ many}} < \mathscr{P}^B < \mathscr{N}$$
$$< \underbrace{\mathscr{P}^A < \cdots < \mathscr{P}^A}_{\tilde{f}(3)-\tilde{f}(2)\text{ many}} < \dots,$$

where $\tilde{f}(n) := \min\{k \mid f(k) > 2^n\}$. We will now discuss how to ensure that this arrangement succeeds—the trick involves allowing negative restraint to *transfer sideways*, and making positive requirements at the same level *share boxes*.

Let us consider the simple case when $\tilde{f}(1) = 0$, hence the first positive requirement appears at level 2 as shown below, with four different versions:



At this level, we will be allowed only two boxes for $\mathscr{P}^A$ (instead of four, as in the strategy we have discussed above). Hence each version of $\mathscr{P}^A$ no longer has the luxury of having its own box—all of its siblings have to share the two boxes available to them. If f grows very slowly, then there might be many levels (corresponding to each k such that $f(k) < 2$) below level 2 which are allowed two boxes. Box-sharing is therefore a necessary feature in this construction (compare it to the previous construction); we need it to resolve the complications introduced by an f which we do not have control of, as the box size is not always allowed to increase as we move down the construction tree.

The observation we need to make is that both versions $\mathscr{N}$ and $\widehat{\mathscr{N}}$ of the same negative requirement are actually trying to do the same thing, namely they are both measuring the same length of agreement. The only difference between the two, is that they would each believe in different computations, i.e., for instance the A-computation $\Phi^A(x)[s]$ converges, and $\widehat{\mathscr{N}}$ believes in it but not $\mathscr{N}$. Also assume that $\mathscr{N}$ and $\widehat{\mathscr{N}}$ are building recursive functions h and $\tilde{h}$ respectively. The plan is to allow B-restraint to transfer sideways from the left ($\mathscr{N}$) to the right ($\widehat{\mathscr{N}}$).

Suppose $\mathscr{N}$ was visited, and the two versions of $\mathscr{P}^A$ below $\mathscr{N}$ fills up the two boxes allocated to level 2. When $\widehat{\mathscr{N}}$ is next visited, the versions of $\mathscr{P}^A$ below $\widehat{\mathscr{N}}$ would not have enough boxes left to use, if $\mathscr{N}$ is holding A-restraint (i.e., waiting for B-recovery). However, $\widehat{\mathscr{N}}$ can wait until all the B-computations in $dom(h)$ have recovered, before it decides to do anything. When this happens, $\widehat{\mathscr{N}}$ can restraint B on the use of these computations. Since $\mathscr{N}$ no longer cares what we put into A (as long as we hold the B-restraint and continue preserving the common value recorded by h), $\mathscr{N}$ could drop its A-restraint, and the versions of $\mathscr{P}^A$ below $\widehat{\mathscr{N}}$ can recycle the boxes used earlier. In general, a negative node α on the construction tree will wait for the lengths of agreement of all the nodes to its left and at the same level to recover, before α plays its expansionary stage. Once that happens, α would then drop A restraint to allow boxes below it to become accessible once again. The downside is of course, we would increase the B-restraint by a huge amount each time. This "transferred" B-restraint has to be obeyed even at α-expansionary B-stages, and will increase only at each visit to the left of α. Since the amount of transferred B-restraint is finite if α is on the true path, the B-positive requirements would be satisfied.

In this theorem we only make B non-recursive. Suppose we wanted to make B ultrahigh, using the method described above. The problem is that now, the B-positive requirements also has a number of pre-determined B-boxes at each level. In the strategy above, we had allowed the restraint to be transferred from the A

side to the B side, so that certain A-boxes can become accessible again. However by doing so we might block some B-boxes and cause them to become unusable. Therefore, the strategy above does not immediately produce an ultrahigh, or even just a superhigh B. Indeed, it is not known if there is a minimal pair of A and B, such that $\emptyset'$ is jump traceable relative to both A and B, via an arbitrary order f.

3.3. The global considerations. In the previous section we had described how to make a minimal pair A, B such that $\emptyset'$ is A-jump traceable via an arbitrary recursive order, and B is non-recursive. In general a positive requirement living on level k will only need to use 2^n many boxes, where n is the number of levels $< k$ of the construction tree devoted to negative requirements.

The full construction requires us to have requirements $\mathscr{P}_e^A$ which wants to do its tracing at order Φ_e^A. We now need to trace $J^{\emptyset'}(k)$ with respect to all A-recursive orders, instead of just one. Hence, the arrangement of the A-modules on the construction tree cannot be pre-determined, and will have to be decided as more and more of Φ_e^A shows up.

Suppose τ is assigned the requirement $\mathscr{P}_e^A$. τ would measure the length of convergence $l(\tau, s)$ of Φ_e^A, that is, the initial segment of $\Phi_e^A[s]$ such that $\Phi_e^A[s]\upharpoonright_{l(\tau,s)}$ looks like it is an order. τ will divide its job amongst infinitely many sub-requirements $\{\mathscr{P}_{e,i}^A\}_{i\in\mathbb{N}}$, which are all spread out at different levels below τ. Let

$$m_i^\tau = |j \in \mathbb{N} \colon \tau < \mathscr{N}_j < \mathscr{P}_{e,i}^A|,$$

which is the number of levels j of the construction tree devoted to a negative requirement, between τ and its i^{th} sub-requirement.

Suppose α is a node on the construction tree assigned the requirement $\mathscr{P}_{e,i}^A$. Instead of tracing $J^{\emptyset'}(k)$ for a single k at level $|\alpha|$, we let α (and all the other nodes on the same level) handle several of these k-modules, for all $k \in L(|\alpha|, s)$, such that

$$L(|\alpha|, s) := \{k < l(\tau, s) \mid 2^{m_i^\tau} < \Phi_e^A(k)[s] \leq 2^{m_{i+1}^\tau}\}.$$

The atomic strategy described above is used for each of these k-modules. However, α will require two outcomes (k, ∞) and (k, f) for each k-module it is looking after, hence requiring altogether $2^{|L(|\alpha|,s)|}$ many outcomes, which we could fix in advance if not for the fact that $L(|\alpha|, s)$ itself might change as the construction proceeds. Thus, the construction tree will have to be infinite branching at the levels assigned to subrequirement nodes.

When τ first detects that the length of convergence is long enough so that $\Phi_e^A(l(\tau, s))[s] > 2^{m_{i+1}^\tau}$, it will make all the nodes at level $|\alpha|$ start their atomic strategies. These nodes will start the process of tracing all the $J^{\emptyset'}(k)$ for $k \in L(|\alpha|, s)$. Note that all the traces will be made with A-use larger than s. In future if $L(|\alpha|, t) \neq L(|\alpha|, s)$, it must be the case that there is a change in A below s. Hence the next time τ sets its i^{th} sub-requirement to work with the new $L(|\alpha|, t)$, all of the earlier traces (which we no longer want) would have been automatically removed; no intervention by τ is needed for this. It might also be possible that $L(|\alpha|)$ changes infinitely often (as in the case when Φ_e^A is not an order), so we will have to arrange a leftmost outcome 0 for α. Each time before α can start work with a new set of τ-modules $L(|\alpha|)$, we will make α play outcome 0 once. This ensures that the true path is well-defined, in the case when α never receives a permanent set of instructions.

3.4. Construction tree layout. The construction takes place on a subtree of $\omega^{<\omega}$. Nodes of length $3e$ are assigned the requirement $\mathscr{N}_e$ with the outcomes ∞ for infinitely many expansionary stages, and f for finitely many expansionary stages, with $\infty <_{left} f$.

Nodes of length $3\langle e, i\rangle + 1$ are assigned the requirement $\mathscr{P}_e^A$ if $i = 0$, and the subrequirement $\mathscr{P}_{e,i-1}^A$ if $i > 0$. The possible outcomes of $\mathscr{P}_e^A$ are $\infty <_{left} f$, which stands respectively for infinite (finite) action taken. The outcomes of $\mathscr{P}_{e,i}^A$ are $0 <_{left} 1 <_{left} 2 <_{left} \dots$. Outcome 0 to the extreme left means no action, i.e., the atomic strategies have not started. The other outcomes each represent the code number of a finite sequence of pairs $(n_0, x_0), (n_1, x_1), \dots, (n_j, x_j)$ where the n_i's are distinct natural numbers, and $x_i \in \{\infty, f\}$, in some effective coding $\langle\cdot\rangle$ with range $\mathbb{N}\backslash\{0\}$. The important point here is to ensure that if σ and η are two such sequences, then $\langle\sigma, (n, \infty), \eta\rangle < \langle\sigma, (n, f), \eta\rangle$.

We say that m specifies the pair (k, x), if m is the code number of a sequence of pairs in which (k, x) appears. An outcome specifying (k, ∞) is played when α is charged with the task of tracing $J^{\emptyset'}(k)$, and a new value has just been shown by the opponent. An outcome specifying (k, f) on the other hand, will be played when α is assigned to trace $J^{\emptyset'}(k)$, but no clearing of boxes are pending.

Finally, nodes of length $3e + 2$ are assigned the requirement $\mathscr{P}_e^B$, with outcome s for success (in meeting the simplicity requirement), placed to the left of w, for waiting.

Let $\alpha <_{left} \beta$ denote that node α is strictly to the left of node β, extended lexicographically from the ordering among the outcomes. We say that α is a *$\mathscr{Q}$-node* if α is assigned the requirement $\mathscr{Q}$. A *negative node* is an $\mathscr{N}_e$-node for some e. We say that α is a *mother node*, if α is a $\mathscr{P}_e^A$-node for some e, and that α is an *A-positive* node if α is a $\mathscr{P}_{e,i}^A$-node for some e, i. A *B-positive* node is a $\mathscr{P}_e^B$-node for some e.

A node τ is said to be the *mother* node of α, if $\tau \prec \alpha$, and τ is a $\mathscr{P}_e^A$-node and α is a $\mathscr{P}_{e,i}^A$ for some e, i. In this case we also call α an *i-daughter* node of τ. If α is an A-positive node, then $\tau(\alpha)$ denotes its unique mother node. α and β are *sibling* nodes, if $\tau(\alpha) = \tau(\beta)$, and $|\alpha| = |\beta|$.

If α is an A-positive node, we let $Left(\alpha)$ denote all the sibling nodes of α, strictly to the left of α. If α is any other type of nodes, $Left(\alpha)$ simply denotes the set of all nodes $\beta <_{left} \alpha$ such that $|\beta| = |\alpha|$. We let $Right(\alpha)$ be defined similarly with the inequality reversed.

3.5. Notations. Suppose that τ is a $\mathscr{P}_e^A$-node. The k^{th} module of τ, which wants to trace $J^{\emptyset'}(k)$, is also called the *(τ, k)-module*. We divide a cofinite segment of $\mathbb{N}$ into infinitely many partitions $\{M_i^\tau\}_{i\in\mathbb{N}}$, where

$$M_i^\tau = \{x\colon 2^{\langle e,i+1\rangle - \langle e,0\rangle} < x \leq 2^{\langle e,i+2\rangle - \langle e,0\rangle}\}.$$

Basically $\langle e, i+1\rangle - \langle e, 0\rangle = m_i^\tau$, i.e., the number of levels j between τ and its i^{th} subrequirement containing negative nodes. An i-daughter node α of τ will be allowed $2^{m_i^\tau}$ many boxes at its level. Hence α will be entrusted with the task of ensuring the success of the (τ, k)-module for every k such that $\Phi_e^A(k) \in M_i^\tau$.

τ's job is to coordinate the actions of all of its daughter nodes, and build the A-u.r.e. sequence $\{V_k^\tau\}_{k\in\mathbb{N}}$ (note that oracle A is suppressed from the notation V_k^τ). In actual fact we are constructing, at the node τ, the Turing functional $\Psi^A(k, n)$

so that for every k, $Range(\Psi^A(k,\cdot)) = V_k^\tau$. We refer to each value $\Psi^A(k,n)$ as a *box*, which we will fill with a number. Fixing an effective coding $\langle\cdot\rangle$ of all finite $\{A,B\}$-sequences, we will call $\Psi^X(k,\langle\sigma\rangle)$ a *σ-box of V_k^τ*. That is, we think of each set V_k^τ as a row of boxes, where each box is labelled by a finite string $\sigma \in \{A,B\}^{<\omega}$. The purpose of this definition, is that during the period of time where α (and all its sibling nodes at the same level) is running the (τ,k)-module, it will be allowed to place numbers in, as well as take numbers out of the σ-boxes of V_k^τ, for every $\sigma \in \{A,B\}^{<\infty}$ of length m_i^τ. Hence at level $|\alpha|$ we have access to $2^{m_i^\tau}$ many boxes.

At a particular stage s, when we say that we *fill a σ-box of V_k^τ with use u*, we mean that we enumerate the axiom $\langle\langle k,\langle\sigma\rangle\rangle, y, A_s\restriction_{u+1}\rangle$ into Ψ. The s^{th} stage use of the computation $\Psi^A(k,n)[s]$ is denoted by $u_{k,n}^\tau[s]$. To *clear*, or to *empty the σ-box of V_k^τ at a stage s*, means that we enumerate $u_{k,\langle\sigma\rangle}^\tau[s]$ into A. As usual we associate finite strings with their code numbers, and write $u_{k,\sigma}^\tau$ in place of $u_{k,\langle\sigma\rangle}^\tau$. During a stage s of the construction in which we enumerate into A, if an A-positive node α is visited and we filled some box for the sake of α, we will say that the box is *filled by α* at stage s. This remains true until the box is next emptied.

Define the *length of convergence* for Φ_e^A at stage s, to be

$$l(e,s) = \max\{y < s \mid (\forall x \le y)\ \ \Phi_e^A(x)[s]\downarrow \ge \Phi_e^A(x-1)[s]\}.$$

Since τ is measuring $l(e,s)$, we also write $l(\tau,s)$ in place of $l(e,s)$. A stage s is *τ-expansionary*, if either $s = 0$, or else τ is visited at stage s of the construction, and $l(\tau,s) > l(\tau,s^-)$ where $s^- < s$ is the largest τ-expansionary stage smaller than s.

α has a list of instructions at stage s, defined as:

$$L(\alpha,s) = \{k < l(\tau,s) \mid \Phi_e^A(k)[s] \in M_i^\tau\},$$

which is a list of τ-modules that α has to run. This list will change as the construction proceeds, and when more of Φ_e^A is revealed. All sibling nodes of α will have the same list of instructions at all times, so they will all run the same modules, until τ says otherwise.

We will say that α *is active* at stage s, if τ allows α to start running the modules in the instruction list $L(\alpha)$. That is, α is active at s, if $\alpha \succ \tau^\frown\infty$ and we have the following:

- (AC.1): $\Phi_e^A(l(\tau))[s] \notin M_i^\tau$.
  [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 $t < s$ where α is visited at both stages t and s, $L(\alpha,t) = L(\alpha,s)$, and for every $k \le 1 + \max L(\alpha,s)$, the computation $\Phi_e^A(k)[s]$ has persisted for at least two visits to α (i.e., $A\restriction_{\varphi_e(A;k)}[t] = A\restriction_{\varphi_e(A;k)}[s]$).
  [To ensure 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 instructions.]

We also say that α is *permanently active* at stage s, if it is active at every visit to α after stage s, i.e., it never gets interrupted in running the modules.

Let β be an $\mathscr{N}_e$-node. At β we define a partial recursive function h_β to record the common value of $\Phi_e^A = \Phi_e^B$. As in the previous proof, we do not increase $dom(h_\beta)$ at all β-expansionary stages, instead we hold back until an appropriate time. We also let $R(\beta, s) = X$ if β is holding X-restraint at stage s. Each stage s of the construction is either an A-stage where numbers are enumerated into A, or a B-stage in which numbers are enumerated into B.

Suppose τ is a mother node, and $\delta \succ \tau$ is any node. Let $\beta_0 \prec \cdots \prec \beta_{k-1} \prec \delta$ be precisely all the negative nodes lying on the path between τ and δ. As in the previous proof, we need to record at each stage s, whether each of these β_i is holding A or B restraint. We are allowed less storage space now—we will only record the outcomes of these negative nodes: let

$$state(\tau, \delta, s) = X_0 X_1 \ldots X_{k-1},$$

where for all $i < k$, the value $X_i \in \{A, B\}$ is determined by the following: if $\beta_i^\frown\infty \preceq \delta$ then let $X_i = B$, otherwise we let $X_i = R(\beta_i, s)$.

If δ is a daughter node of τ, we abbreviate $state(\tau, \delta, s)$ by $state(\delta, s)$. If δ is a $\mathscr{P}_{e,i}^A$-node, then $|state(\delta, s)| = m_i^\tau$. As in the previous construction, $state(\delta, s)$ functions as a pointer telling δ to run all of its modules using the $state(\delta, s)$-box. It might be that several of δ's sibling nodes are also pointing simultaneously at the same box—unlike in the previous proof—and we have to ensure that δ's progress will not be undone by another of its sibling node. It also follows trivially from the definition that if $\delta_1 \preceq \delta_2$, then $state(\tau, \delta_1) \preceq state(\tau, \delta_2)$.

We fix a partial ordering $<_A$ on the set of all finite $\{A, B\}$-sequences, which will be used to determine priority amongst the different *state* values. Let $\rho <_A \sigma$, iff there is some least $i < \min\{|\rho|, |\sigma|\}$, such that $\rho\restriction_i = \sigma\restriction_i$, and $\rho(i) = B \neq \sigma(i)$.

An A-positive node α will start work only if it is active. The variable $state(\alpha)$ points at a particular box in the k^{th} row of boxes (for each k in the instruction list $L(\alpha)$). α will fill the box in the k^{th} row each time it plays an outcome specifying (k, f), and will clear the box in the k^{th} row each time the outcome specifies (k, ∞). As the situation in the negative nodes above α changes, α may move on to other boxes in the same row; however α only has access to $2^{m_i^\tau}$ boxes in each row.

All sibling nodes on the same level will share boxes; all of them will have access to all the boxes available to that level. Because of this, and because of the transfer of restraints, we might actually return to a box of higher $<_A$-priority, as we move from the left to the right of the construction tree. We have to be a bit careful with notations, in particular when considering α-believablity of A-computations. There are now more cases to consider:

Definition 3.2. For a negative node α, we say that a computation $\Phi^A(n)[s]$ with A-use w is *α-believable* at stage s, if

1. for all A-positive nodes $\beta \prec \alpha$, and each k such that $\alpha(|\beta|)$ specifies (k, ∞), the $state(\beta, s)$-box of $V_k^{\tau(\beta)}$ is either empty, or has use $> w$,
2. for all A-positive nodes β which has been visited prior to stage s, such that $\tau(\beta)^\frown\infty \prec \beta^\frown 0 \preceq \alpha$, and all $k \in L(\beta, s)$, the $state(\beta, s)$-box of $V_k^{\tau(\beta)}$ is either empty, or has use $> w$,
3. for all mother nodes $\tau \prec \alpha$, all strings $\gamma >_A state(\tau, \alpha, s)^\frown B$ and all x, the γ-box of V_x^τ is either empty, or has use $> w$,

4. for all A-positive nodes $\beta \prec \alpha$, and all A-positive nodes $\sigma >_{left} \beta$, with $\tau(\sigma) \prec \alpha$, and all strings $\gamma \succeq state(\tau(\sigma), \beta, s)$, and all x, if the γ-box of $V_x^{\tau(\sigma)}$ is currently full and was filled by σ, then the use is $> w$.

We now give an explanation for having each of the conditions 1–4 above. Condition 1 ensures that there are no pending changes due to incorrectly filled boxes from above. For condition 2, we note that if β has true outcome 0, then all the relevant boxes will have to be eventually cleared, so we don't want to believe in an α-computation until they are cleared. Condition 3 ensures that boxes of a lower $<_A$-priority will not be blocked by an increased restraint. Condition 4 is present because at each visit to β on the true path, we want to clear all the boxes filled by some $\sigma >_{left} \beta$ (even though these boxes might be of a higher $<_A$-priority).

Suppose $\alpha <_{left} \alpha'$ are negative nodes at the same level. As discussed previously, the two nodes are measuring the same length of convergence. Thus, our plan was to let α' wait until all the B-computations below $|\,dom(h_\alpha)|$ recover, before we allow α' to play an expansionary stage. However, α and α' may actually believe in different computations. In particular there might be some A-computation $\Phi^A(x)$ for some $x \in dom(h_\alpha)$ which α believed in (and hence placed in $dom(h_\alpha)$), but which α' does not believe in. It would be unreasonable for us to insist that α' wait for the recovery of such computations—instead we allow α' to neglect these computations when considering α'-expansionary stages. Doing so might cause h_α to record the wrong value, but as we will see, things will work out fine—h_α can only be wrong at finitely many values if α is to be on the true path.

Apart from considering lengths of agreement, we define a new parameter called the *length of believability*. The intention for this is to let α' assess the believability of $dom(h_\alpha)$: if α is an $\mathscr{N}_e$-node, we define the length of believability to be

$$l_b(\alpha, s) = \max\{y < s \mid (\forall x < y)\ \ (\Phi_e^A(x)[s]\downarrow, \text{ and is } \alpha\text{-believable})\}.$$

We also define the *length of agreement* measured at α to be:

$$l(\alpha, s) = \max\{y < s \mid (\forall x < y)\ \ (\Phi_e^A(x)[s]\downarrow = \Phi_e^B(x)[s]\downarrow), \text{ and } l_b(\alpha, s) \geq y\}.$$

We say that a stage s is *α-expansionary*, if α is visited at stage s, and both of the following hold:

1. $l(\alpha, s) \geq |\,dom(h_\alpha)[s]|$,
2. For each $\beta \in Left(\alpha)$ such that $R(\beta, s) = A$, and $l_b(\alpha, s) \geq |\,dom(h_\beta)[s]|$, we require that $l(\alpha, s) \geq |\,dom(h_\beta)[s]|$.

3.6. The construction. Each stage s will either be an A-stage with enumerations into A, or a B-stage with enumerations into B. When we pick a *fresh* number at a stage s, we mean that we pick a number $> s$ and $>$ any number used or mentioned so far. At stage $s > 0$, we inductively define the stage s approximation to the true path, δ_s of length s. Suppose that $\alpha = \delta_s\restriction_d$ has been defined for $d < s$. If α is an $\mathscr{N}_e$-node or a $\mathscr{P}_e^A$-node, then we let $\delta_s(d) = \infty$ if s is α-expansionary, and $\delta_s(d) = f$ otherwise. If α is a $\mathscr{P}_e^B$-node, then we check if $B_s \cap W_{e,s} = \emptyset$. If so, we let $\delta_s(d) = w$, else we let $\delta_s(d) = s$.

Finally, suppose that α is a $\mathscr{P}_{e,i}^A$-node. If α is currently not active, we let $\delta_s(d) = 0$. Otherwise, for each $k \in L(\alpha, s)$, we have to let $\delta_s(d)$ specify either (k, ∞) or (k, f).

For each $k \in L(\alpha, s)$, we check if the following holds (we will explain the choices for OUT.1–3 in the paragraph that follows):

- (OUT.1): The current box that α is pointing at in the k^{th} row (i.e., the $state(\alpha, s)$-box of $V_k^{\tau(\alpha)}$) is either unoccupied, or it is occupied by the correct value (i.e., contains $J^{\emptyset'}(k)[s]$).
- (OUT.2): $J^{\emptyset'}(k)[s]\downarrow$, and we require that the computation has persisted for at least two visits to α. We let $t < s$ be the least stage such that the computation $J^{\emptyset'}(k)[s]\downarrow$ applies at stage t (i.e., $J^{\emptyset'}(k)[t]\downarrow$ with $\emptyset'_t\restriction_{j(k)[t]} = \emptyset'_s\restriction_{j(k)[t]}$). Hence, α is visited at some stage u, $t \leq u < s$.
- (OUT.3): We also check that for every $k' \in L(\alpha, s) - \{k\}$, such that $J^{\emptyset'}(k')[t]\downarrow$ and $\emptyset'_t\restriction_{j(k')[t]} \neq \emptyset'_s\restriction_{j(k')[t]}$, there is a stage u such that $t < u < s$, α is visited at stage u, and $\emptyset'_t\restriction_{j(k')[t]} \neq \emptyset'_u\restriction_{j(k')[t]}$.

If all of the above hold, we let $\delta_s(d)$ specify (k, f), otherwise we let $\delta_s(d)$ specify (k, ∞). That is, we let $\delta_s(d) = \langle (k_0, x_0), \dots, (k_p, x_p) \rangle$, where k_i's are precisely all the distinct elements of $L(\alpha, s)$, and the x_i's are specified above.

As promised, we will justify the choices for the conditions OUT.1–3 above—these are implemented purely for technical reasons. OUT.1 ensures that if the current box is occupied by the wrong value, then we immediately force outcome (k, ∞)—this ensures that the box will be cleared at the next possible chance. The combination of OUT.2 and 3 ensures that the true path is consistent with the "truth of outcome". In particular, consider the following scenario: suppose that $\{k_0, k_1\} \subset L(\alpha, s)$. It might be the case that both $J^{\emptyset'}(k_0)\uparrow$ and $J^{\emptyset'}(k_1)\uparrow$, in which case the correct outcome which we expect is $(k_0, \infty), (k_1, \infty)$. Because we combined both modules at the single node α, we have to ensure that this outcome is played infinitely often. This might be a problem if $J^{\emptyset'}(k_0)$ and $J^{\emptyset'}(k_1)$ take turns to show us a new value.

The solution to this is to ensure the following. If it is the case that $J^{\emptyset'}(k)\downarrow$ and has settled by stage t, then we will delay playing the outcome (k, f) until all the other modules $k' \in L(\alpha)$ which has not yet settled on $J^{\emptyset'}(k')$, has had a chance to play (k', ∞).

This completes the definition of δ_s. We now give the actions to be taken at stage s, in the following order:

1. Declare that s is an X-stage, where X is to be decided below. If $s = 0$ or $s = 1$, let $X = A$. Otherwise, look for the longest $\beta \prec \delta_s$ such that β has been visited prior to stage s. Let $s^- < s$ be the most recent stage such that $\delta_{s^-} \succ \beta$. If s^- is an A-stage, let $X = B$, otherwise let $X = A$.
2. Take actions for each expansionary negative node β. For each $\mathcal{N}_e$-node β such that $\beta^\frown\infty \preceq \delta_s$, we do the following:
   (a) Set $R(\beta, s + 1) = X^c$.
   (b) For all $x < l(\beta, s)$, we set $h_\beta(x)\downarrow = \Phi_e^A(x)[s]$, unless it is already defined.
   (c) Take actions for every β' whose restraint is transferred to β; this step of the construction is known as *transferring β'-restraint to β*: for every $\beta' \in Left(\beta)$ such that $R(\beta', s) = A$ and $l_b(\beta, s) \geq |dom(h_{\beta'})[s]|$, we do the following:
      (i) Set $R(\beta', s + 1) = B$, since this transferred B-restraint will be held until the next β'-expansionary stage.

   (ii) Injure every negative node $\xi \succ \beta'^\frown f$, i.e., we set $h_\xi = \emptyset$, since this is a "pseudo" β'-expansionary stage.
3. Injure the negative nodes on the right, as well as the negative nodes on the left which have believed in wrong computations.
   (a) On the right: for each negative $\beta >_{left} \delta_s$ such that $|\beta| < s$, we set $h_\beta = \emptyset$, and set $R(\beta, s+1) = R(\delta_s\lceil_{|\beta|}, s+1)$. This ensures we do not use a box of a stronger $<_A$-priority when we visit right of δ_s later.
   (b) On the left: for each negative $\beta <_{left} \delta_s$, $|\beta| < s$ such that $R(\beta, s) = A$ and $l_b(\delta_s\lceil_{|\beta|}, s) < |\mathit{dom}(h_{\beta,s})|$, we set $h_\beta = \emptyset$. That is, β on the left has previously believed and set up restraint to preserve A-computations that are no longer believable. We injure β even though it is to the left, and we will have to show that this happens only finitely often if β is on the true path.
4. Injure all the mother nodes to the right.
   (a) For each mother node $\tau >_{left} \delta_s$, we set $V^\tau_k = \emptyset$ for all k, and set $u^\tau_{k,n} \uparrow$ for all k, n.
5. If $X = A$, we empty all the boxes which need to be cleared.
   (a) Action for nodes along δ_s: for each A-positive node $\beta \prec \delta_s$, and k such that $\delta_s(|\beta|)$ specifies (k, ∞), we clear the $\mathit{state}(\beta, s)$-box of $V^{\tau(\beta)}_k$, if it is not already empty.
   (b) Clear all boxes of lower $<_A$-priority: for each mother node $\tau \prec \delta_s$, a number x and a string $\gamma >_A \mathit{state}(\tau, \delta_s, s)$, we clear the γ-box of V^τ_x, if not already empty.
   (c) Clear all boxes filled by some $\sigma >_{left} \delta_s$, regardless of the $<_A$-priority: for all A-positive nodes β and σ such that $\beta \prec \delta_s$ and $\sigma >_{left} \beta$ with $\tau(\sigma) \prec \delta_s$, and some string $\gamma \succeq \mathit{state}(\tau(\sigma), \beta, s)$, some number x, such that the γ-box of $V^{\tau(\sigma)}_x$ is currently full and was filled by σ, we clear the box.
6. If $X = A$, we will also fill all the boxes needing to be filled, unless the decision to fill a particular box made at the beginning of the stage, is now deemed unwise due to enumerations made in step 5.
   (a) For each A-positive $\beta \prec \delta_s$, and k such that $\delta_s(|\beta|)$ specifies (k, f), we place $J^{\emptyset'}(k)[s]$ into the $\mathit{state}(\beta, s)$-box of $V^{\tau(\beta)}_k$ with a fresh use, unless the box is already full.
   An exception to the rule is the following: a node $\beta \prec \delta_s$ might want to fill a box of $V^{\tau(\beta)}_k$ (hence β is active when we are defining δ_s above), but the actions in step 5 might have caused an enumeration into A below some Φ^A-use, and β is no longer active after step 5 is done. In this case, β would *not* fill the box.
   That is, we would do the above for each A-positive $\beta \prec \delta_s$, unless in step 5, we had made an enumeration into A below the use of $\Phi^A_r(n)[s]$ (where β is a $\mathscr{P}^A_{r,j}$-node) for some $n \leq 1 + \max L(\beta, s)$.
7. If $X = B$, we will give the B-positive nodes a chance to act.
   (a) For each $\mathscr{P}^B_e$-node β such that $\beta^\frown w \preceq \delta_s$, enumerate into B, the least number x (if there is one) satisfying the following:
      - $x > 2e$ and $x \in W_{e,s}$,

- $x > \max\{t < s \mid \delta_t <_{left} \beta\}$, and
- for every $\mathcal{N}_{e'}$-node σ such that $\sigma <_{left} \beta$ or $\sigma^\frown f \preceq \beta$ for some e', and for each $n \in \cup_{t \leq s} dom(h_\sigma)[t]$, we also require that $x > \varphi_{e'}(B; n)[s]$.

That is, once a negative node $\sigma <_{left} \beta$ has a B-recovery, β will not be able to enumerate below the recovered use. This is to ensure that B-positive requirements obey the transferred B-restraint at all times.

3.7. Verification. We say that a node α plays an outcome x at a stage s, if $\delta_s \succ \alpha^\frown x$. It is easy to see that there is a leftmost path visited infinitely often: if α is A-positive 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 $L(\alpha)$ eventually settles and α will have to play one of the $2^{|L(\alpha)|}$ many possible outcomes. Let TP be the true path of the construction. We list a few facts regarding the construction:

1. Suppose α is visited infinitely often. Then for each $X \in \{A, B\}$, there are infinitely many X-stages s such that $\delta_s \succ \alpha$ (see lemma 2.3).
2. For each negative α, $dom(h_\alpha)$ is changed only at α-expansionary stages (where it is increased), or when it is *reset* (i.e., set to $\emptyset$).
3. For a negative α, there are three actions in the construction which can cause a change in $R(\alpha)$ (and therefore the *state* variable):
   - when we visit left of α,
   - during an α-expansionary stage, where we set $R(\alpha)$ based on whether the current stage is an A or B-stage,
   - when we visit right of α, and transfer α-restraint to the right; in this case we change $R(\alpha)$ from $A \to B$.
4. Because of fact 3, if $\tau \prec \alpha$ are along the true path, then $state(\tau, \alpha)$ eventually settles. We denote the limit value of $state(\tau, \alpha)$ by $State^\tau_\alpha$.

For a node α on the true path, we let $True(\alpha)$ be the least stage s such that α is visited at stage s, the construction never visits left of α after stage s, and for every $\tau \prec \alpha$, $state(\tau, \alpha, s)$ has settled. Note that if $\alpha \preceq \beta$, then $True(\alpha) \leq True(\beta)$.

We begin with a few preliminary lemmas. Firstly, we argue that for any segment $\tau \prec \alpha$ on the true path, once $state(\tau, \alpha)$ has settled, no box of a stronger $<_A$-priority than $state(\tau, \alpha)$ can be accessed:

Lemma 3.3. *Let $\tau \prec \alpha$ be on the true path, where τ is a mother node. Then, for every $t \geq True(\alpha)$, we have*

$$state(\delta_t\restriction_{|\tau|}, \delta_t\restriction_{|\alpha|}, t) \geq_A State^\tau_\alpha .$$

Proof. Let $s_0 = True(\alpha)$. Let ξ be a negative node such that $\tau \prec \xi^\frown f \preceq \alpha$. We claim that if there is a ξ'-expansionary stage $t > s_0$ for some $\xi' \in Right(\xi)$, then $R(\xi)$ will be set to B (if ξ is not already holding B-restraint), and $R(\xi)$ will stay at B forever: otherwise h_ξ will be reset, and for every $\eta \in Left(\xi)$, either h_η will be reset, or $R(\eta)$ will be set to B (and stays that way). This is impossible because the next visit to ξ after t will have to be ξ-expansionary, by definition.

To prove the lemma, we suppose that there is a $t > s_0$ such that $state(\delta_t\restriction_{|\tau|}, \delta_t\restriction_{|\alpha|}, t) <_A State_\alpha^\tau$, and derive a contradiction. We have

$$\begin{aligned} state(\delta_t\restriction_{|\tau|}, \delta_t\restriction_{|\alpha|}, t) &= \sigma B \sigma' \\ State_\alpha^\tau &= \sigma A \sigma'' \end{aligned}$$

for some strings σ, σ' and σ''. The $(|\sigma|+1)^{th}$ digits in $state(\delta_t\restriction_{|\tau|}, \delta_t\restriction_{|\alpha|}, t)$ and $State_\alpha^\tau$ must correspond to negative nodes ξ_0 and ξ_1 respectively with $\xi_0 \geq_{left} \xi_1$. Clearly $\tau \prec \xi_1{}^\frown f \preceq \alpha$, and also that $R(\xi_1) = A$ after stage s_0.

By the claim at the beginning of this proof, stage t cannot be a ξ_0-expansionary stage. It follows that $R(\xi_0, t) = B$. However when α was visited at stage $s_0 < t$, step 3(a) of the construction would have set $R(\xi') = A$ for every $\xi' \in Right(\xi_1)$. In fact, step 3(a) was included in the construction to ensure that the situation discussed here does not arise—we want to prevent a box of a stronger $<_A$-priority from being accessed later. Clearly $R(\xi')$ for all $\xi' \in Right(\xi_1)$ will have to be A after stage s_0, a contradiction. ⊣

Next, we turn our attention to the positive nodes. For the next two lemmas 3.4 and 3.5, we let τ be a $\mathscr{P}_e^A$-node and α be an i-daughter node of τ such that $\tau^\frown\infty \prec \alpha$, with both lying on the true path. Let $L := \lim_{s\to\infty} L(\alpha, s)$, which may or may not exist. It is not hard to see that

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

In the next lemma, we want to show that the true outcome is consistent with the "truth". This becomes a concern because we are combining several modules, each with their separate outcomes, in a single node. 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 that level. This ensures that boxes filled by a positive node on the true path, does not become emptied while the construction was visiting right of α.

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

(i) *Let $L = \{k_0, \dots, k_p\}$. The true outcome of α is $\langle(k_0, x_0), \dots, (k_p, x_p)\rangle$ where for each $i = 0, \dots, p$, $x_i = \infty$ 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 $\delta_t \succ \tau^\frown\infty$ and $\delta_t(|\alpha|)$ specifies (k, ∞).*

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 $s_0 \geq True(\alpha)$ be an A-stage large enough, such that α is visited at s_0, and at every stage $t \geq s_0$,

- $L(\alpha, t) = L$,
- α is active at stage t (if it is visited),
- for each $k \in L_f$, $J^{\emptyset'}(k)[t]\downarrow$ on the correct use, and
- for each $k \in L_f$, the $State_\alpha^\tau$-box of V_k^τ is either unoccupied, or else occupied by $J^{\emptyset'}(k)$. This is possible because if the $State_\alpha^\tau$-box is occupied by a number other than $J^{\emptyset'}(k)$, α will play an outcome specifying (k, ∞) every time it is visited, until the box is cleared of the erroneous value.

At every visit to α after s_0, α will play one of the outcomes from the list $\{\langle (k_0, a_0), \dots, (k_p, a_p)\rangle \mid a_i = \infty$ or f for all $i\}$. To prove that the true outcome of α is $\langle (k_0, x_0), \dots, (k_p, x_p)\rangle$, we will 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 use after stage s_1. Wait until all the changes occur, for all these $k' \in L_\infty$, and we let $s_2 > s_0$ be the *second* (or more) visit to α after the last change. We claim that after stage s_2, any outcomes played by α has to specify (k, f): Pick $k' \in L - \{k\}$, such that $J^{\emptyset'}(k')[s_1] \downarrow$ with $\emptyset'_{s_1}\restriction_{j(k')[s_1]} \neq \emptyset'_{s_2}\restriction_{j(k')[s_1]}$. We want to show that α is visited at some stage u between $s_1 < u < s_2$, where the change has already occured by stage u. If $k' \in L_f$ then $u = s_0$ since any such change has to occur by stage s_0; on the other hand if $k' \in L_\infty$ then such a u must exist, by the choice of s_2.

To prove the second part, we let $s_3 > s_0$ be a stage where α is visited. 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 stage $s_4^k > s_3$ such that α is visited, and $\emptyset'_{s_3}\restriction_{j(k)[s_3]} \neq \emptyset'_{s_4^k}\restriction_{j(k)[s_3]}$. Let $\tilde{k}$ be some member of $\tilde{L}$ with the largest $s_4^{\tilde{k}}$, and let $s_4 = s_4^{\tilde{k}}$. Therefore, if $s_4^- < s_4$ is the previous visit to α, then $\emptyset'_{s_3}\restriction_{j(\tilde{k})[s_3]} = \emptyset'_{s_4^-}\restriction_{j(\tilde{k})[s_3]} \neq \emptyset'_{s_4}\restriction_{j(\tilde{k})[s_3]}$, i.e., the oracle will change below the use for $\tilde{k}$, but only after stage s_4^-.

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. Hence $k \neq \tilde{k}$, and therefore condition (OUT.3) in the construction is not met when deciding the outcome for k. Hence (k, ∞) is played at stage s_4.

(ii) This is proved in a similar way as in (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 in this case we have to wait until every node $\alpha' \in Right(\alpha)$ which has been visited prior to stage s_2, is visited again at least one more time (if ever), say 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 $\delta_t \succ \tau^\frown\infty$, we either have $\delta_t(|\alpha|) = 0$, or else $\delta_t(|\alpha|)$ specifies (k, f). $\dashv$

The next lemma tells us that each time α plays outcome 0—which will be the case every time A changes below the use of some computation in $L(\alpha)$—certain boxes will also be cleared. (i) states that if k is not eventually in $L(\alpha)$, then all the γ-boxes for $|\gamma| = m_i^\tau$, which are incorrect locations for the k^{th} row, will be cleared eventually. (ii) covers the case when $k \in \lim L(\alpha)$, 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 each convergent value $\Phi_e^A(k)[t] \in M_i^\tau$. Each time α plays outcome 0, the box might be full, unlike in (i), because some sibling node of α might fill the box after each change in the use for $\Phi_e^A(k)[t]$. However, if α is to have true outcome 0, then this box cannot remain permanently filled—it has to be cleared before the next visit to $\alpha^\frown 0$.

LEMMA 3.5. *Let τ and α be as above.*

(i) *If α plays outcome* 0 *at a stage s, then for every number $k \notin L(\alpha, s)$ and string γ of length m_i^τ, the γ-box of V_k^τ has to be empty at stage s.*

(ii) *If α is visited at stage s, then for every k and string γ of length m_i^τ, if the γ-box of V_k^τ 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 $|\gamma| = |\mathit{State}_\alpha^\tau|$ be such that the box is full with use $u_{k,\gamma}^\tau[s] \downarrow$. The box must have been filled at some A-stage $t < s$, in which $\delta_t(|\alpha|)$ was active at stage t and $k \in L(\alpha, t)$. The fact that $k \notin L(\alpha, s)$, means that A has to change below the use of the $\Phi_e^A(k)[t]$-computation, i.e., change below t. Furthermore this change cannot happen at stage t itself—otherwise we would have not have filled the box at stage t—hence the change occurs after $u_{k,\gamma}^\tau[s] > t$ was set, a contradiction. In fact, we don't need the assumption that α plays outcome 0, but this will always be true if k goes out of $L(\alpha)$.

(ii) Let α be visited at stage s, and $u_{k,\gamma}^\tau \downarrow$ at the end of stage s. This box is filled at some τ-expansionary stage $t \leq s$, in which $k \in L(\alpha, t)$. Before α can next play outcome 0, there must be a change below some computation in $L(\alpha, t)$, i.e., below t. As in (i), this change has to occur after $u_{k,\gamma}^\tau > t$ was set, a contradiction. ⊣

In the spirit of minimal pair arguments, one would want to show that at each negative node α, at least one side of computations is preserved between α-expansionary stages. In the next lemma we show that if t_0 is an α-expansionary B-stage, then the A-computations will be preserved until one of the three things happen:

1. we come to the next α-expansionary stage,
2. h_α gets reset because some $\alpha' \in \mathit{Right}(\alpha)$ does not believe in the A-computations which α had believed in. This happens only finitely often, as shown in part (ii).
3. the B-side of the computations recover, and the A-restraint is dropped while the construction was at the right of α. α doesn't care if this happens, since the value is now preserved on the B-side.

LEMMA 3.6. *Let α be an $\mathcal{N}_e$-node on the true path.*

(i) *Suppose $t_0 \geq \mathit{True}(\alpha)$ is an α-expansionary X-stage, such that $X_{t_0}^c\restriction_m \neq X_{t_1}^c\restriction_m$ for some $t_1 > t_0$, where $m =$ use of $\Phi_e^{X^c}\restriction_{dom(h_\alpha)}[t_0]$ and $R(\alpha, t_1) = X^c$. Then, there must be a stage u such that $t_0 < u < t_1$, where either u is α-expansionary, or else h_α is reset at stage u.*

(ii) *h_α is reset only finitely often.*

PROOF. (i) Let $s_0 = \mathit{True}(\alpha)$, and t_0, t_1 and m be as in the statement of the lemma. Suppose on the contrary there is no such u. The number $k < m$ has to enter at an X^c-stage t, where $t_0 < t < t_1$. By the assumption that there is no such u, we must have $\delta_t >_{\mathit{left}} \alpha^\frown\infty$.

The case $X = A$ is easy, so we consider $X = B$. We will show that the enumeration of $k < m$ into A at stage t contradicts α-believability of the use m at stage t_0. Since uses are chosen fresh, we may assume that $k = u_{x,\gamma}^\tau$ was set before stage t_0, and that $\tau \prec \alpha$. Since $R(\alpha, t_1) = A$, it follows that $R(\alpha, u) = A$ at every stage $u = t_0 + 1, \dots, t_1$. In fact, the following is true:

CLAIM 3.6.1. *For every $\alpha' \in Right(\alpha)$, and $t_0 < u < t_1$, we must have $R(\alpha', u) = A$, and that u is not α'-expansionary.*

PROOF OF CLAIM. Since t_0 was α-expansionary, step 3(a) of the construction would have ensured that $R(\alpha') = A$ for all $\alpha' \in Right(\alpha)$. This stays that way until some α'-expansionary stage u, in which we will either transfer α-restraint by flipping $R(\alpha)$ over to B, or we will reset h_α, both of which are impossible by assumption. ⊣

By Claim 3.6.1 we have $state(\tau, \delta_t, t) \succeq state(\tau, \delta_t\restriction_{|\alpha|}, t)^\frown A$, and by Lemma 3.3, we in fact have $state(\tau, \delta_t, t) >_A State^\tau_\alpha{}^\frown B$. At stage t, $k = u^\tau_{x,y}$ was enumerated under step 5 of the construction. There are three cases:

1. Step 5(b): if this was the action putting k into A, then $\gamma >_A State^\tau_\alpha{}^\frown B$, contradicting α-believability of use m at t_0.
2. Step 5(a): we either have $\delta_t \succ \alpha^\frown f$, or $\delta_t >_{left} \alpha$. The former can immediately be seen to contradict α-believability, so we assume at stage t, the latter holds. Let β be the node which wants to clear $u^\tau_{x,y}$, where $\gamma = state(\beta, t)$. If $|\beta| > |\alpha|$, then clearly $\gamma >_A State^\tau_\alpha{}^\frown B$, contradicting α-believability just as in 1. above. Suppose that $|\beta| < |\alpha|$, and so it follows that the use m was α-believable at stage t_0, but is not $\delta_t\restriction_{|\alpha|}$-believable at stage t, i.e., $l_b(\delta_t\restriction_{|\alpha|}, t) < |\,dom(h_\alpha)[t]|$. In this case the construction would promptly reset h_α at stage t, contrary to assumption.
3. Step 5(c): let $\beta \prec \delta_t$ be the node which wants to clear a box filled by $\sigma >_{left} \beta$. Clearly $\beta \prec \alpha$ can immediately be seen to contradict α-believability, so suppose $\beta \not\prec \alpha$. If $|\beta| > |\alpha|$ we would also have $\gamma >_A State^\tau_\alpha{}^\frown B$ just as in 1. above, while $|\beta| < |\alpha|$ means that $l_b(\delta_t\restriction_{|\alpha|}, t) < |\,dom(h_\alpha)[t]|$ just as in 2. above.

All three cases are impossible, hence the enumeration of k cannot happen.

(ii) We note that a consequence of Lemma 3.4(ii), is that there is a stage s_1 large enough, such that for every A-positive node $\beta \prec \alpha$, and every k such that $\alpha(|\beta|)$ specifies (k, f), and every $\beta' \in Right(\beta)$ with $\beta' \succ \tau(\beta)^\frown\infty$, we must have β' playing (k, f) each time it is visited after stage s_1 (unless β' is visited for the very first time). That is, only the outcome (k, f) can be played at level $|\beta|$, unless we are visiting the node for the very first time (and thus the node is not active). We let $s_1 \geq s_0$ be large enough so that this holds, and argue that h_α cannot be reset after stage s_1.

The only way for h_α to be reset after stage s_1, would be for α to have believed in some computation which is placed in $dom(h_\alpha)$, but which later some $\alpha' \in Right(\alpha)$ refuses to believe in. We have to go through each case 1–4 of Definition 3.2 and show that they still apply.

Let $s_2 > s_1$ be a stage where h_α is reset. Each time h_α is reset after stage s_1, there must be an α-expansionary stage before h_α can next be reset, so we may as well assume that $s_1 < s_{exp} < s_2$ where s_{exp} is the largest α-expansionary stage less than s_2. Since $R(\alpha, s_2) = A$ (otherwise we would leave h_α alone at s_2), hence s_{exp} has to be a B-stage setting $R(\alpha) = A$. Letting $m =$ use of $\Phi^A_e\restriction_{dom(h_\alpha)}[s_{exp}]$, by part (i) we know that the A-side has to be preserved, i.e., $A_{s_{exp}}\restriction_m = A_{s_2}\restriction_m$, and therefore any witness to non-believability at stage s_2, has to be some $u^{\tau}_{x,\gamma} < m$, which was set before stage s_{exp}. We let $\alpha' = \delta_{s_2}\upharpoonright_{|\alpha|} \in \mathit{Right}(\alpha)$, i.e., α' is the node where $l_b(\alpha', s_2) < |\mathit{dom}(h_\alpha)|$ and refuses to believe in some computation in $\mathit{dom}(h_\alpha)$ with use $< m$. Corresponding to conditions 1–4 of Definition 3.2, we have one of the following:

1. For some A-positive node $\beta' \prec \alpha'$, we have $\gamma = \mathit{state}(\beta', s_2)$ and $\alpha'(|\beta'|)$ specifies (x, ∞): the first thing to note is that $\gamma \geq_A \mathit{State}^{\tau(\beta')}_{\alpha\upharpoonright_{|\beta'|}}$. If they are not equal then we have $\gamma >_A \mathit{State}^{\tau(\beta')}_{\alpha}{}^\frown B$, which is impossible because of the α-believability of use m at stage s_{exp}. Hence, $\gamma = \mathit{State}^{\tau(\beta')}_{\alpha\upharpoonright_{|\beta'|}}$. The question is, what is the true outcome $\alpha(|\beta'|)$? We know $\alpha(|\beta'|)$ cannot specify (k, ∞) because of α-believability of m. On the other hand, the true outcome $\alpha(|\beta'|)$ cannot specify (k, f) by the choice of s_1, otherwise (k, ∞) cannot be played at level $|\beta'|$ anymore. This leaves us with the fact that 0 is the true outcome of $\alpha\upharpoonright_{|\beta'|}$. By α-believability of m again, it must be the case that $k \notin L(\beta', s_{exp})$, in which case by Lemma 3.5(i), the use $u^{\tau}_{x,\gamma}$ has to be set after stage s_{exp}, a contradiction.
2. For some A-positive node β' which has been visited prior to s_2, we have $\beta'{}^\frown 0 \preceq \alpha'$, $x \in L(\beta', s_2)$ and $\gamma = \mathit{state}(\beta', s_2)$: a contradiction is derived in a similar way as in 1. above.
3. For some mother node $\tau \prec \alpha'$, we have $\gamma >_A \mathit{state}(\tau, \alpha', s_2)^\frown B$: this is not possible because $\gamma >_A \mathit{State}^{\tau}_{\alpha}{}^\frown B$, and because of α-believability once again.
4. For some A-positive nodes β' and σ such that $\beta' \prec \alpha'$ and $\sigma >_{\mathit{left}} \beta'$, we have $\tau(\sigma) \prec \alpha$, and $\gamma \succeq \mathit{state}(\tau(\sigma), \beta', s_2)$: we have $\mathit{state}(\tau(\sigma), \beta', s_2) \geq_A \mathit{State}^{\tau(\sigma)}_{\alpha\upharpoonright_{|\beta'|}}$, and hence the inequality can be split into two cases: $=$ and $>_A$, both cases are treated in the same way as above.

Thus all 4 cases are not possible, so the use m has to be α'-believable at stage s_2. This shows that h_α can be reset only finitely often. ⊣

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

Proof. Let α be an $\mathcal{N}_e$-node on the true path such that $\Phi^A_e = \Phi^B_e$ is total. It is easy to see that if α is visited at some stage s after which h_α is never reset, then necessarily $s \geq \mathit{True}(\alpha)$: this is because if some negative β such that $\beta^\frown f \preceq \alpha$ has β-restraint being transferred, then the same step of the construction will also reset h_α. We first claim:

Claim 3.7.1. *There are infinitely many α-expansionary stages.*

Proof of claim. Suppose that on the contrary, there are finitely many α-expansionary stages—let s_{exp} be the last α-expansionary stage. Obviously h_α cannot be reset after s_{exp}, and hence by the observation above we have $s_{exp} \geq \mathit{True}(\alpha)$.

Since α-expansionary stages only occur if all the computations of $\mathit{Left}(\alpha)$ recover, we let $l = \max|\mathit{dom}(h_\beta)[s_{exp}]|$ for $\beta = \alpha$ or $\beta \in \mathit{Left}(\alpha)$. Let $s_2 > s_{exp}$ be a stage large enough, so that both A_{s_2} and B_{s_2} are correct up to $m =$ use of $\Phi^A_e\upharpoonright_l$ and $\Phi^B_e\upharpoonright_l$. The only reason why s_2 is not α-expansionary, is because the use m is not α-believable at stage s_2. Hence, one of the cases 1–4 of Definition 3.2 must not apply.

Cases 1 and 4 give straightforward contradictions to the correctness of $A_{s_2}\restriction_m$. Suppose that case 2 fails, i.e., for some β with $\beta^\frown 0 \preceq \alpha$, and some $k \in L(\beta, s_2)$, we have $u^{\tau(\beta)}_{k,state(\beta,s_2)}[s_2] < m$. In this case even though β has true outcome 0, it might be the case that boxes in the k^{th}-row always get filled by some overly zealous sibling node of β. Despite this, it follows by Lemma 3.5(ii) that the use $u^{\tau(\beta)}_{k,state(\beta,s_2)}[s_2]$ has to be cleared before $\beta^\frown 0$ can next be visited after s_2, another contradiction to the correctness of $A_{s_2}\restriction_m$. Lastly, suppose that case 3 fails, i.e., $u^\tau_{x,\gamma}[s_2] < m$ for some $\tau \prec \alpha$, x and $\gamma >_A State^\tau_\alpha{}^\frown B$. If $R(\alpha, s_2) = B$, then the construction would enumerate $u^\tau_{x,\gamma}[s_2] < m$ into A at s_2. So, $R(\alpha, s_2) = A$ but in that case s_{exp} has to be an B-stage. Since the A-computations have to be preserved after s_{exp}, it follows that $u^\tau_{x,\gamma}[s_2]$ has to be set before s_1, and thus would cause the use m not to be α-believable at stage s_1—a contradiction again. ⊣

By Lemma 3.6(ii), h_α is reset only finitely often, hence h_α is recursive. As in the previous theorem, the fact that there are infinitely many α-expansionary stages does not immediately imply that h_α is total. One has to use Claim 3.7.1 to show that every use which appears non-believable must be cleared at α-expansionary stages. Similar to Claim 2.4.3, one can then show that h_α is total, i.e., every number is eventually put in $dom(h_\alpha)$.

Finally, it is not hard to see that h_α records the correct value. The only thing new in this construction, is that α-restraint might be transferred while the construction was visiting right of α (i.e., $R(\alpha)$ flips from A to B). However, this happens only if the B-computations measured by α recovers, and the B-positive nodes to the right of $\alpha^\frown\infty$ always respects this use the instant it converges. Therefore, all negative requirements are satisfied. ⊣

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

Proof. Let τ be a $\mathscr{P}^A_e$-node on the true path, such that Φ^A_e is an order. Clearly ∞ is the true outcome of τ; we need to show that $\{V^\tau_k\}_{k\in\mathbb{N}}$ traces $J^{\emptyset'}$ correctly. We fix a number k. Let i be the least such that $\Phi^A_e(k) \in M^\tau_i$, and α be the version of $\mathscr{P}^A_{e,i}$ on the true path, with true outcome o. Hence α will eventually be responsible for tracing $J^{\emptyset'}(k)$ in V^τ_k.

We first of all show that $|V^\tau_k| < \Phi^A_e(k)$. A box in the k^{th} row cannot be permanently set by a node β, where $|\beta| \neq |\alpha|$: suppose such a β fills a box in the k^{th} row at stage s, and $k \in L(\beta, s)$. We know k eventually has to leave $L(\beta)$, so it is not hard to see that at the first visit to $TP\restriction_{|\beta|}$ (the true version of β) after k leaves $L(\beta)$, we must have $TP\restriction_{|\beta|}$ playing outcome 0. By Lemma 3.5(i), this means that all boxes on the k^{th} row filled at level $|\beta|$ has to be empty at that visit.

Hence any box of V^τ_k can only be filled permanently by a node at level $|\alpha|$. However such nodes will only fill a γ-box for $|\gamma| = m^\tau_i$. There are only $2^{m^\tau_i} < \Phi^A_e(k)$ many choices for γ.

Next, suppose that $J^{\emptyset'}(k)\downarrow$. We want to show that $J^{\emptyset'}(k) \in V^\tau_k$. We know α will become permanently active and has true outcome specifying (k, f). Let $s_0 \geq True(\alpha^\frown o)$ be large enough for our purpose. At stage s_0, α would put $J^{\emptyset'}(k)$ into the $State^\tau_\alpha$-box of V^τ_k (unless it is occupied, in which case it has to be occupied by the right value). The only thing that would stop us, is that step 5 of the construction had made an enumeration below some computation in $L(\alpha, s_0)$, in which case α would not be active at the next visit and plays outcome 0 left of its true outcome—impossible. At any rate the $State^{\tau}_{\alpha}$-box on the k^{th} row has to contain the correct value at the end of stage s_0, and we want to show that this box is never emptied after s_0.

Suppose on the contrary, that at stage $t > s_0$ of the construction, an enumeration $p \leq u^{\tau}_{k,State^{\tau}_{\alpha}}[s_0]$ was made. Now, p must be the use of some box, which is set at some stage $t' \leq s_0$. We may in fact assume that $True(\alpha^\frown o) < t'$ (by letting s_0 wait long enough), and the use p was set by node β at stage t'. There are now three cases, depending on the position of β; we want to get a contradiction in all three cases:

1. $\beta \preceq \alpha$: then p is the use of the $State^{\tau(\beta)}_{\beta}$-box in some k'-row. Since β is along the true path, the only way for us to clear the use p at stage t, would be through Step 5a. Therefore the true outcome of β has to specify (k', ∞), and thus p actually has to be cleared at stage s_0, before α sets the use $u^{\tau}_{k,State^{\tau}_{\alpha}}$. This is impossible.
2. $\beta >_{left} \alpha$: step 5(c) of the construction was specially included, with the purpose of clearing all such use, which were set by nodes to the right of the true path.
3. $\beta \succ \alpha$: at stage t' when we set the use p, we would do it *after* α sets the use $u^{\tau}_{k,State^{\tau}_{\alpha}}[t']$. Since $u^{\tau}_{k,State^{\tau}_{\alpha}}[t'] < p \leq u^{\tau}_{k,State^{\tau}_{\alpha}}[s_0]$, it follows that the use p has to be cleared before stage s_0, another contradiction.

Hence such a use p cannot exist, so the $State^{\tau}_{\alpha}$-box, once filled by α, remains full. This shows A is ultrahigh. Next, we want to show that the capping companion B is non-recursive. B is certainly coinfinite, and since $dom(h_{\sigma})$ only increases at σ-expansionary stages for an A-positive σ, it follows that the restraint in step 7(a) of the construction is finite for a B-positive node on the true path. $\dashv$

This ends the proof of Theorem 3.1.

§4. Further questions.

Several natural questions present themselves. For instance, in Theorem 3.1, if the B-positive requirements make infinitely many enumerations, such as in making B ultrahigh, almost complete, or even just high, the method of transferring B-restraint no longer works. This is because if $\widehat{\mathcal{N}}$ to the right of $\mathcal{N}$ drops the A-restraint while holding B restraint for $\mathcal{N}$, it might do so while believing incorrectly (infinitely often) in certain B-computations.

1. Is there a minimal pair of r.e. ultrahigh, or a minimal pair of r.e. almost complete?
2. In general, for any arbitrary recursive order f, is there a minimal pair of superhigh r.e. sets, via f?
3. Is there an r.e. set which is non-cuppable and ultrahigh?

REFERENCES

[1] G. Barmpalias and A. Montalbán, *A cappable almost everywhere dominating computably enumerable degree*, ***Electronic Notes in Theoretical Computer Science***, vol. 167 (2007), pp. 17–21.

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

[3] G. Chaitin, *Information-theoretical characterizations of recursive infinite strings*, ***Theoretical Computer Science***, vol. 2 (1976), pp. 45–48.

[4] P. Cholak, R. Downey, and N. Greenberg, *Triviality and jump traceability*, to appear.

[5] P. Cholak, N. Greenberg, and J. Miller, *Uniform almost everywhere domination*, this Journal, vol. 71 (2006), no. 3, pp. 1057–1072.

[6] R. Coles, R. Downey, C. Jockusch, and G. LaForte, *Completing pseudojump operators*, ***Annals of Pure and Applied Logic***, vol. 136 (2005), pp. 297–333.

[7] N. Dobrinen and S. Simpson, *Almost everywhere domination*, this Journal, vol. 69 (2004), pp. 914–922.

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

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

[10] D. Hirschfeldt, A. Nies, and F. Stephan, *Using random sets as oracles*, to appear.

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

[12] C. Jockusch and R. Shore, *Pseudojump operators. I. The r.e. case*, ***Transactions of the American Mathematical Society***, vol. 275 (1983), no. 2, pp. 599–609.

[13] B. Kjös-Hanssen, *Low for random reals and positive-measure domination*, ***Proceedings of the American Mathematical Society***, (2006), to appear.

[14] B. Kjös-Hanssen, J. Miller, and R. Solomon, *Lowness notions, measure and domination*, 2006, to appear.

[15] S. Kurtz, ***Randomness and genericity in the degrees of unsolvability***, Ph.D. thesis, University of Illinois at Urbana-Champaign, 1981.

[16] D. Martin, *Classes of recursively enumerable sets and degrees of unsolvability*, ***Zeitschrift für Mathematische Logik und Grundlagen der Mathematik***, vol. 12 (1966), pp. 295–310.

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

[18] ———, *Lowness properties and randomness*, ***Advances in Mathematics***, vol. 197 (2005), pp. 274–305.

[19] ———, ***Computability and randomness***, Oxford University Press, 2006, to appear.

[20] S. Simpson, *Almost everywhere domination and superhighness*, ***Mathematical Logic Quarterly***, vol. 53 (2007), pp. 462–482.

[21] R. Solovay, *Draft of paper (or series of papers) on Chaitin's work*, unpublished notes, 215 pages, 1975.

[22] S. Terwijn and D. Zambella, *Algorithmic randomness and lowness*, this Journal, vol. 66 (2001), pp. 1199–1205.

SCHOOL OF MATHEMATICS, STATISTICS AND COMPUTER SCIENCE
VICTORIA UNIVERSITY OF WELLINGTON
PO BOX 600, WELLINGTON, NEW ZEALAND
E-mail: Keng.Meng.Ng@mcs.vuw.ac.nz