CUPPING AND JUMP CLASSES IN THE COMPUTABLY ENUMERABLE DEGREES

NOAM GREENBERG, KENG MENG NG, AND GUOHUA WU

Abstract. We show that there is a cuppable c.e. degree, all of whose cupping partners are high. In particular, not all cuppable degrees are low_3-cuppable, or indeed low_n cuppable for any n, refuting a conjecture by Li. On the other hand, we show that one cannot improve highness to superhighness. We also show that the low_2-cuppable degrees coincide with the array computable-cuppable degrees, giving a full understanding of the latter class.

1. Introduction

What help must one have in order to fully understand the halting problem? This is an informal way to state the interaction between cupping and jump classes. A c.e. degree $\mathbf{a}$ is *cupped* by a c.e. degree $\mathbf{b}$ if $\mathbf{a} \vee \mathbf{b} = \mathbf{0}'$. We think of $\mathbf{b}$ as adding enough information to what $\mathbf{a}$ already knows, so that together they know everything a c.e. degree can know. We also call $\mathbf{b}$ a *cupping partner* for $\mathbf{a}$. A way of measuring the complexity of a c.e. degree is to ask what kind of degrees can serve as its cupping partners.

All kinds of behaviour are possible. We call a degree *cuppable* if it has an incomplete cupping partner. A corollary of the Sacks splitting theorem [Sac63] is that there is a low, cuppable degree. On the other hand, Yates and Cooper (see for example [Mil81]) showed that there are nonzero *noncuppable* degrees, ones for which no amount of incomplete information can help reach $\mathbf{0}'$. Cupping (and dually, capping) properties of c.e. degrees have been studied extensively (see for example [FS81]), and played a role in proofs of undecidability of the theory of the c.e. degrees as a partial ordering [HS82, NSS98].

A natural question to ask is: *how much* more information does a degree $\mathbf{a}$ need in order to compute $\mathbf{0}'$? We can use the hierarchy of high_n and low_n jump classes in order to give quantitative answers to this question. The most important result related to this question is the characterisation of the *low-cuppable* degrees, those that have a low cupping partner, as precisely those that contain a promptly simple set (Ambos-Spies, Jockusch, Shore and Soare [ASJSS84]). Thus, two notions of strength coincide: on the one hand, having enough computational power so that just a low addition suffices to reach $\mathbf{0}'$; and containing a set with a quick approximation, a dynamic property that resembles the halting problem itself. Another important concept concerning the interaction of joins with jump classes is the almost deep degree of Cholak, Groszek and Slaman [CGS01].

Date: October 16, 2020.

The first author is partially supported by the Marsden Fund of New Zealand. The second author is partially supported by grant MOE2015-T2-2-055. The third author is supported by Singapore Ministry of Education Tier 2 grant MOE2016-T2-1-083 (M4020333); NTU Tier 1 grants RG32/16 (M4011672) and RG111/19 (M4012245).

There are cuppable degrees which are not low-cuppable (this follows from Harrington's cup and cap theorem, see [FS81]). Li, Wu and Zhang [LWZ00] showed that there is a low_2-cuppable degree which is not low-cuppable. They suggested investigating a hierarchy of cuppable degrees, namely the low_n-cuppable degrees for $n \geqslant 1$. It is open if the low_n-cupping hierarchy collapses at some n:

Question 1.1. Is each level of the low_n-cupping hierarchy distinct from the next?

Li [Li] conjectured that every cuppable degree is low_3-cuppable. In this paper we refute Li's conjecture in a strong way:

Theorem 1.2. *There is a cuppable degree, all of whose cupping partners are high.*

It follows that the cuppable degrees are not contained in the union of the hierarchy of the low_n-cuppable degrees. Theorem 1.2 is proved in Section 2.

We continue by considering cuppability with refined jump classes. Bickford and Mills [BM82] suggested a refinement of the high/low hierarchy of jump classes by considering the truth-table degree of the jump, rather than its Turing degree. In particular, this gives the notions of superhigh and superlow degrees. In Section 3 we show that in Theorem 1.2 we cannot replace "high" by "superhigh" (Theorem 3.1). We note that Diamondstone [Dia09] showed that superlow-cupping does not coincide with low-cupping, giving more evidence to the lack of symmetry between superlowness and superhighness.

The next class close to lowness other than the low_2 degrees and the superlow degrees are the array computable (AC) degrees. In the c.e. degrees, array computability (introduced by Downey, Jockusch and Stob [DJS90, DJS96]) is stronger than being low_2, incomparable with lowness, and weaker than superlowness. One could expect, in light of the distinction between superlow, low and low_2-cupping, that AC-cupping would be distinct still and incomprable with low-cupping. However, Downey, Greenberg, Miller and Weber [DGMW08] showed that in fact low cuppability implies AC-cuppability. They did show that low-cuppability and AC-cuppability do not coincide. In Section 5 we improve their result by giving a complete characterisation of AC-cuppability:

Theorem 5.1. AC-cuppability coincides with low_2-cuppability.

A fundamental result about cupping is the *continuity of cupping* by Ambos-Spies, Lachlan and Soare [ASLS93]: no cuppable degree has a minimal cupping partner. That is, if $\mathbf{c}$ cups $\mathbf{a}$ to $\mathbf{0}'$ then there is some $\mathbf{b} < \mathbf{c}$ which also cups $\mathbf{a}$ to $\mathbf{0}'$. (The analogous result for capping was established by Harrington and Soare [HS92]). In Section 4 we complement this result for low-cuppable degrees, by showing that low-cuppability is witnessed below any cupping partner:

Theorem 4.1. If $\mathbf{a}$ is low-cuppable then below any cupping partner of $\mathbf{a}$ there is a low cupping partner of $\mathbf{a}$.

That is, if $\mathbf{a} \vee \mathbf{c} = \mathbf{0}'$ and also, $\mathbf{a}$ is low-cuppable, then there is some low $\mathbf{b} \leqslant \mathbf{c}$ such that $\mathbf{a} \vee \mathbf{b} = \mathbf{0}'$. This result answers a question left open in [DGMW08]: it implies that every low-cuppable degree has a cupping partner which is both low and array computable (Corollary 4.2).

We remark that in Theorem 3.1 we observe a similar phenomenon: we showed, in fact, that for any cuppable A, for any cupping partner C of A, there is a non-superhigh cupping partner $X \leqslant_{\mathrm{T}} C$ of A.

2. Cupping with only high degrees

This section is devoted to the proof of Theorem 1.2.

The requirements. We will enumerate a c.e. set A, whose degree will be as promised by the theorem. To make A cuppable, we enumerate a c.e. set C, to serve as a cupping partner of A. So a global requirement is that $\emptyset' \leqslant_{\mathrm{T}} A \oplus C$. To ensure that C is incomplete, we enumerate an auxiliary c.e. set E (a "private copy" of $\emptyset'$), and for every functional Ψ, we meet the requirement

$\mathcal{P}_\Psi$**:** $E \neq \Psi(C)$.

We need to ensure that every cupping partner of A is high. For this too we enumerate an auxiliary c.e. set P – another private copy of $\emptyset'$ – and for each c.e. set W and each functional Φ, we meet the requirement

$\mathcal{R}_{W,\Phi}$**:** $P = \Phi(A, W) \implies W$ is high.

2.1. **Discussion.** Let us give an informal description of our strategies for meeting the requirements, and the tensions caused by their interaction.

Cupping. We build a reduction of $\emptyset'$ to $A \oplus C$. The uses of this reduction are called *moving markers*: to code the entrance of k into $\emptyset'$ we enumerate the marker $\gamma_s(k)$ into either A or C. Otherwise, markers are allowed to move if they, or smaller markers, are enumerated into either A or C.

Incompleteness requirements. Naively, an incompleteness requirement $\mathcal{P}_\Psi$ will appoint a follower y, wait for $\Psi(C, y) = 0$, enumerate y into E, and freeze C below the use $\psi(y)$. To freeze C, we need to promise that any marker $\gamma_s(k)$ smaller than the use will be enumerated, if necessary, into A rather than C.

Highness requirements. We will give below the formal details of how to ensure that a c.e. set W is high, using oracle traces. We now describe it less precisely. We fix a universal partial Σ^0_2 function g. It has a computable approximation $\langle g_s \rangle$ in the sense that $\langle g_s(n) \rangle$ converges if and only if $n \in \operatorname{dom} g$. To show that W is high, for each n, we declare a W-use $u_s(n)$ for "tracking" (or tracing) $g_s(n)$.

- If $n \in \operatorname{dom} g$, then we need the uses $u_s(n)$ to stabilise.
- If we see a change in $g(n)$ (that is, $g_t(n) \neq g_s(n)$), then we want W to change below $u_s(n)$. We are allowed to fail, but only finitely many times for each n.[1]

How do we effect a change in W? Recall that we are working for requirement $\mathcal{R}_{W,\Phi}$, so let us assume that $\Phi(A, W) = P$. To induce W-changes we appoint an "agitator" q targeted for P. We let $u = u_s(n)$ be the use $\varphi_s(q)$ of the computation $\Phi(A, W, q) = 0$. When we want the change, we enumerate q into P, *and freeze A below u*. Once we see a recovery of agreement between $\Phi(A, W)$ and P, it must be the case that we got the W-change we were after. Note, however, that freezing $A \upharpoonright u$ until we saw that W-change is absolutely crucial. Moreover, in fact we need to freeze A below u *even before we want to effect a W-change*, that is, even before we see a

[1]The reason this makes W high is the following: since g is universal, its domain has degree $\emptyset''$; we need to show that W' computes $\operatorname{dom} g$, i.e., that W can computably approximate $\operatorname{dom} g$. We guess that $n \in \operatorname{dom} g$ if $W_s \upharpoonright u_s(n)$ is W-correct. If $n \notin \operatorname{dom} g$, then except for finitely many errors, we will never see $W_s \upharpoonright u_s(n)$ to be correct: the value of $u_s(n)$ will approach ∞ faster than the settling time of W.

change in $g(n)$. This is because we have to keep $\varphi(q) \leqslant u$; an A-change allows $\varphi(q)$ to rise, and once it does, q is useless for effecting a change in W below u.

Timing A-enumerations and protecting markers. The main tension in the construction is now clear: $\mathcal{R}$ requirements need to restrain A, whereas $\mathcal{P}$ requirements make promises about markers going into A. The restraint imposed by an $\mathcal{R}$ requirement is not permanent: say that $u = \varphi(q)$ and we want a $W \upharpoonright u$-change. We enumerate q into P and wait until we get that change. Once we see this change, the W-tracking of the old value of $g(n)$ is released. Eventually we will want to track the new value with potentially larger use; but we can wait a bit, and for now, drop the restraint and allow small numbers to go into A.

We see that as is often the case, timing is everything. However the naive plan for $\mathcal{P}$ was to promise that some markers $\gamma(k)$ will be enumerated into A rather than C; this enumeration happens when k enters $\emptyset'$, and we have no control over the timing of this event. What we do is anticipate this event, and when $\mathcal{R}$ drops its restraint, immediately enumerate the marker into A. This allows us to clear the markers below the use $\psi(y)$ and appoint new ones, larger than this use. Thus when k does enter $\emptyset'$, we can always enumerate the marker $\gamma(k)$ into C.

This has to be done with certain priority; for coding to work, we need to ensure that each marker is moved only finitely many times. Thus each $\mathcal{P}$ requirement will be allowed to move markers $\gamma(m)$ for m greater than some k, and k increases as the priority of the requirement decreases. What about smaller markers? We have to allow them to injure the realising computation $\Psi(C, y)$. This happens at most k times.

The tree. As expected, we do everything on a tree of strategies. Incompleteness requirements $\mathcal{P}$ need to guess and coordinate the restraint and its simultaneous dropping imposed by stronger $\mathcal{R}$ requirements. Most obviosuly, we cannot be sure that the $\mathcal{R}$ hypothesis $\Phi(A, W) = P$ is correct; for example, we may enumerate an agitator q and wait for a resulting $W \upharpoonright u$-change, but this never happens, because we never get a recovery of $\Phi(A, W, q) = P(q)$. The corresponding A-restraint is then never dropped.

Let τ be a node working for $\mathcal{R}_{W,\Phi}$. It is possible that we get infinitely many τ-expansionary stages – stages at which we see more and more convergence and agreement between $\Phi(A, W)$ and P. However it is still possible that, fixing an input n, the use $u_s(n)$ goes to infinity. This has two possible reasons:

- $g_s(n)$ does not stabilise. As described, each time it changes, we enumerate an agitator q for the sake of tracking $g(n)$ and get the desired W-change. We then need to appoint a new agitator q', for the new observed value of $g(n)$, with possibly larger use $u' = \varphi(q')$. This can happen infinitely often.
- $g_s(n)$ stabilises to $g(n)$, so we have a last agitator q appointed for this n; but the use $\varphi_s(q)$ is unbounded, as $\Phi(A, W, q)\uparrow$.

We remark that in either case, the requirement $\mathcal{R}_{W,\Phi}$ is happy: in the first, because there is no real value $g(n)$ that we need to track, so from its point of view, it is not a problem that the use $u_s(n)$ is unbounded; in the second, because the hypothesis $\Phi(A, W) = P$ does not hold.

However, weaker $\mathcal{P}$ nodes (those who live below $\tau^\frown\infty$) need to guess, for each n, whether the restraint imposed on behalf of $u_s(n)$ is bounded or not, and if not, coordinate stages at which it drops (between various inputs n). Therefore, τ will

have children ρ scattered below $\tau\hat{}\infty$, each looking at one particular input $n = n(\rho)$; nodes below such a child guess the behaviour of $u_s(n)$.

The noncupping construction and delayed enumeration. Using our set-up we can give a very similar description of the construction of a non-cuppable degree (see [Mil81]). In this construction, there is no C (or E); we replace the incompleteness requirements $\mathcal{P}_\Psi$ with non-computability requirements: $A \neq \Psi$; and the conclusion of $\mathcal{R}$ is that W is complete, rather than merely high.

The basic strategy for the non-cuppability requirements $\mathcal{R}$, instead of following $g_s(n)$ and allowing finitely many failures in tracking, is to wait for n to enter $\emptyset'$ and then require a W-change below $u_s(n) = \varphi(q)$, again by enumerating $q = q_s(n)$ into P. We need to do this only once, but we are not allowed to fail.

What has not been explained above yet, and which pertains to the non-cuppability construction as well, is what to do with bigger n. That is: let σ, extending $\tau\hat{}\infty$, be a $\mathcal{P}$-node. In the non-cuppability construction, it appoints a follower k, waits for $\Psi(k)\downarrow = 0$, and then wants to enumerate k into A. The node σ can guess the behaviour of $u_s(n)$ for only finitely many n, say up to $n(\sigma)$. For those n for which it guesses a bound on $u_s(n)$, it can appoint k to be larger than these bounds; for the other n, it waits until the resraint is dropped. But what do we do for $n > n(\sigma)$? Here, since we are not allowed any mistakes, what we need to do is voluntarily enumerate $q_s(n)$ into P and wait for a $W \upharpoonright u_s(n)$-change. The node τ can live with that, since each σ will act at most once, and only finitely many will have $n(\sigma) < n$; so each $q_s(n)$ is "injured" only finitely often.

What this does entail, though, is a delay in enumerating k into A. Recall that after we enumerate $q_s(n)$ into P, we need to wait until the next τ-expansionary stage to see the recovery of $\Phi(A, W, q)$, which guarantees the required W-change — and we need to keep A frozen below $u_s(n)$ until that next τ-expansionary stage. So what will happen is, first, σ announces that it wants to enumerate k into A; it will enumerate $q_s(n)$ into P. Then, we wait for the next τ-expansionary stage; at that stage, τ remembers σ's request and enumrates k into A, without waiting for σ to be accessible again. This last bit is necessary because if we wait for another σ-stage, then as τ cannot wait in the meantime, it will set up a new value $q_t(n)$, and then we will need to enumerate it into P and wait before we put k into A; this process may never terminate.

Now consider what happens when we have two $\mathcal{R}$-nodes $\tau_0 \prec \tau_1 \prec \sigma$. Then the process of delayed enumeration will have one more step: (1) At stage s_2, σ declares it wants k in A; it enumerates $q_{s_2}(\tau_1, n_1)$ into P. (2) At the next τ_1-expansionary stage $s_1 > s_2$, we enumerate $q_{s_1}(\tau_0, n_0)$ into P, and not let any nodes extending τ_1 be accessible yet. (3) At the next τ_0-expansionary stage $s_0 > s_1$, we enumerate k into A. The point is that between stages s_2 and s_1, τ_0 will have been accessible many times and made many W_{τ_0}-definitions; so even if we enumerate $q_{s_2}(\tau_0, n_0)$ into P at stage s_2, this would not have helped us and we would need to do it again at stage s_1.

Finitely many errors, and different versions of $q(n)$. Back to our construction, we observe what goes wrong if we would try to make A non-cuppable rather than only high-cuppable, while still making it cuppable. The key here is the fact that in the non-cuppable construction, each $q_s(n)$ is injured only finitely many times, because each positive requirement $\mathcal{P}$ acts at most once.

In our only-high-cuppable construction, this is no longer the case. Consider the following sequence of events: (1) σ sees a realising computation $\Psi(C, y) = 0$, and declares that it wants to enumerate y into E and a tracker $\gamma_s(k(\sigma))$ into A. It notifies τ such that $\tau^\frown\infty \preccurlyeq \sigma$, and enumerates $q = q_s(\tau, n(\sigma))$ into P. (2) At the next τ-expansionary stage, we see that due to coding $\emptyset'$, the realised computation $\Psi(C, y)$ is no longer valid. The node σ no longer wants y in E. This sequence could happen infinitely often; it would be very bad to keep enumerating and redefining $q(\tau, n)$.

So we cannot prove a contradiction, but we see that there is still a problem with our original construction: it is possible that $n \in \operatorname{dom} g$, so we do not want $q_s(n)$ to change infinitely often. What we do is distribute the agitators and responsibilities among the children ρ of τ. A child ρ not only provides (via its outcomes) guesses to nodes below it, about a particular restraint; it is also responsible for tracing $g_s(n)$ for some $n = n(\rho)$, has an agitator $q_s(\rho)$, and acts relatively independently. If $\sigma \prec \rho$, then ρ will guess whether σ will actually ever enumerate numbers into E or A or not. If it guesses that it will not, then it does its thing, ignoring σ. If it was wrong about this guess, then it will end up to the left of the true path (assuming σ lies on the true path). It is true then that σ's action decouples $u(\rho)$ and $\varphi(q(\rho))$, and so we cannot get rid of the value that ρ traced, even if it is wrong; but we are allowed finitely many mistakes, so this is ok.

The problem of the right. While we do not mind mistakes made by the left, we are worried about the right, since to the right of the true path there are infinitely many children ρ of τ which all work on the same input n. In the scenario above, if σ lies to the right of the true path, we still need to find a way to force a W-change below $u(\rho)$, even though σ's action made the agitator $q(\rho)$ useless for this task.

Well, σ itself has branched off the true path at some node α; but note that as ρ extends σ, it must be that $\alpha \succcurlyeq \tau^\frown\infty$. What we do is give every finite outcome ζ of α its own agitator $p(\zeta)$, which will be activated when ζ lies to the right of an accessible node. Any extension of ζ, such as σ, will guess the restraint imposed for keeping the agitator $p(\zeta)$ useful, and ρ will ensure that $u(\rho) > \varphi(p(\zeta))$; so $p(\zeta)$, rather than the relatively bigger $q(\rho)$, can be used to correct mistakes made by ρ.

The last thing to note is what happens when we have more than one node τ. Again say $\tau_0^\frown\infty \prec \tau_1^\frown\infty \preccurlyeq \alpha$. Below the finite outcomes of α, we need to put nodes ζ for both τ_0 and τ_1. There is only one correct way to do this: longer τ first; see Fig. 1.

Suppose we did it the other way round. So the finite outcome descendant of α is ζ_0, measuring $\varphi_{\tau_0}(p(\zeta_0))$ (with outcomes ∞ and finite r for all $r \in \mathbb{N}$); and each successor of ζ_0 is a ζ_1 node, similarly measuring $\varphi_{\tau_1}(p(\zeta_1))$. For this discussion, let us group all the finite outcomes together into an outcome f, so we imagine that ζ_0 and each ζ_1 have two outcomes, f and ∞, measuring whether $\varphi_{\tau_i}(p(\zeta_i))$ stabilizes or not. Now consider the two nodes $\alpha^\frown\mathrm{f}^\frown\infty$ and $\alpha^\frown\mathrm{f}^\frown\mathrm{f}$. Below each of these two nodes will be ρ_0 nodes, children of τ_0, and sometimes the ones on the left will need to undo the work of the ones on the right. So below $\alpha^\frown\mathrm{f}^\frown\mathrm{f}$ we need yet another ζ_0-node for τ_0. So we are forced to put a ζ_0-node below the finite outcome of a ζ_1-node.

Why does our way work? And what about the infinite outcomes? Well, the point here is that the infinite outcome of a ζ-node indicates that $\Phi_\tau(A, W, p(\zeta))\uparrow$. But this means that the hypothesis of the requirement $\mathcal{R}_{W_\tau, \Phi_\tau}$ is false. Below that outcome, we do not need to work for this requirement. That is, below $\zeta^\frown\infty$, there

are no children of τ. So if we guess for τ_1 first, then τ_0, the only possibly problematic pair of outcomes is $\alpha\hat{}\mathrm{f}\hat{}\infty$ and $\alpha\hat{}\mathrm{f}\hat{}\mathrm{f}$; below both of those we still need to work for τ_1. But below $\alpha\hat{}\mathrm{f}\hat{}\infty$ we have won τ_0. This allows us to restart a new version of τ_1 below $\alpha\hat{}\mathrm{f}\hat{}\infty$. Child nodes of this new version do not interact with children of the old τ_1, in particular, they do not need to undo mistakes by τ_1-children extending $\alpha\hat{}\mathrm{f}\hat{}\mathrm{f}$. Note that when $\tau\hat{}\mathrm{f}\hat{}\infty$ is accessible, the restraint $u(\zeta_0)$ is dropped (we have just seen a voluntary change in W_{τ_0}, so no nodes extending this outcome could do anything which is injurious to ζ_0, or indeed to any τ_0-children extending $\alpha\hat{}\mathrm{f}\hat{}\mathrm{f}$. So we do not need yet another ζ_1-node at $\alpha\hat{}\mathrm{f}\hat{}\mathrm{f}$. Overall we see that we get a mild $\emptyset^{(3)}$-construction (one with no links).

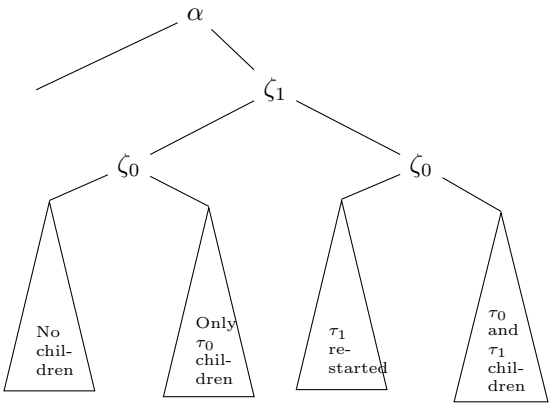

FIGURE 1. Two levels of ζ-nodes. Left is the infinite outcome.

A remark on initialisations. The following is not fundamental, but may help with understanding the formal construction. Rather than outright initialising nodes, we ensure that nodes that ought to be initialised are simply never accessible in the future. This is achieved by adding outcomes to nodes; rather than two outcomes (f and ∞), typically, the outcomes will be $\infty < 0 < 1 < \dots$, with a finite outcome denoting the last stage at which the infinite outcome was accessible. Essentially, this means that each node guesses the last stage at which it is initialised. Lemma 2.2 is the lemma which says that "initialised" nodes are not visited again.

2.2. **Construction.** We now turn to the formal details of the construction.

Markers. To ensure that $\emptyset' \leqslant_{\mathrm{T}} A \oplus C$ we use moving markers. At every stage s we have markers $\gamma_s(k)$ for all $k < \omega$. The marker can move, i.e., $\gamma_{s+1}(k) \neq \gamma_s(k)$, only if some number $x \leqslant \gamma_s(k)$ is enumerated into either A_{s+1} or C_{s+1}. The markers are increasing in k. If $\gamma_{s+1}(k) \neq \gamma_s(k)$ then for all $m \geqslant k$, $\gamma_{s+1}(m)$ is chosen to be large (as usual, this means larger than numbers used in the construction so far).

We fix an enumeration of $\emptyset'$ in which a single number enters $\emptyset'$ at odd stages, and no number enters $\emptyset'$ at even stages. Thus, at odd stages we will react to changes in $\emptyset'$ by coding; at even stages we take care of the other requirements.

Tracing. Recall that a *c.e. trace* is a function T such that for all n, $T(n)$ is a finite set, and $\{(n,x) : x \in T(n)\}$ is c.e. A partial function g is *traced* by T if for all $n \in \operatorname{dom} g$, $g(n) \in T(n)$. A c.e. set X is low if and only if every X-partial computable function has a c.e. trace. It suffices to trace a universal X-partial computable function. Relativising, we see that a Δ^0_2 set W is high if and only there is a W-c.e. trace for a universal Σ^0_2 partial function. We fix such a function g and let $\langle g_s \rangle$ be a partial computable approximation for g. That is, the sequence $\langle g_s \rangle$ is uniformly computable (and total); for $n \in \operatorname{dom} g$, $\lim_s g_s(n) = g(n)$; for $n \notin \operatorname{dom} g$, $\limsup_s g_s(n) = \infty$.

Types of nodes. There are six kinds of nodes (strategies) that we employ.

The incompleteness requirements $\mathcal{P}_\Psi$ employ mother nodes, that we denote by σ, and child nodes, which we denote by π. We denote the functional Ψ by Ψ_σ. The children of σ are all of the immediate extensions of σ on the tree of strategies. When visited, a node σ will pick a *threshhold marker* $k(\sigma)$. The outcomes of σ are $0 < 1 < 2 < \dots$. These outcomes indicate stages by which $\emptyset' \upharpoonright k(\sigma)$ has stabilised.

A $\mathcal{P}$ child node π will appoint a follower $y(\pi)$. For the sake of recording the steps in the delayed enumeration of $y(\pi)$, it will issue a *request token*, which will be placed on some predecessors of π. A $\mathcal{P}$ child node π has two outcomes, $\mathtt{w} < \mathtt{v}$. The outcome $\mathtt{w}$ (wait) indicates that no positive action is taken for π; $\mathtt{v}$ stands for victory: $y(\pi) \in E$.

The highness requirements $\mathcal{R}_{W,\Phi}$ employ four kinds of nodes, related as in Fig. 2. *Matriarch nodes* are denoted by τ. They measure $\limsup_s \operatorname{dom} \Phi(A, W)\,[s]$ (relative to the τ-stages) and its agreement with P. There will be one matriarch node for this requirement on any path. We will write Φ_τ for Φ and W_τ for W. The outcomes are $\infty < 0 < 1 < \dots$.

Matriarch nodes have two kinds of children, *sons* and *daughters*. Sons, denoted ζ, collectively measure $\liminf_s \operatorname{dom} \Phi_\tau(A, W_\tau)\,[s]$. A visited son node ζ will appoint an *initialisation agitator* $p(\zeta)$, and will measure whether $\Phi_\tau(A, W_\tau, p(\zeta))\!\downarrow$, and if so, what is the use of this computation. The outcomes are $\infty < 0 < 1 < \dots$. The outcome ∞ guesses that $\Phi_\tau(A, W_\tau, p(\zeta))\!\uparrow$, so below this outcome, no further action is taken for this requirement.

Daughter nodes, denoted η, coordinate the enumeration of a trace $T^{W_\tau}_\eta$. They represent restarting the requirement after we guessed that a stronger requirement is no longer active. On the true path there will be finitely many daughter nodes, the last one is the one which will enumerate the successful trace. A daughter node only has a single outcome. These nodes don't do much on their own; they are mostly present for book-keeping.

Daughter nodes have children of their own (grandchildren of the matriarch), denoted ρ. Each grandchild ρ is assigned an input $n = n(\rho)$ and is responsible for enumerating $T^{W_\tau}_\eta(n)$. Visited grandchild nodes ρ will appoint an *agitator* $q_s(\rho)$, targeted for P. The value of this agitator may be redefined during the construction, possibly infinitely many times (depending on whether $n(\rho) \in \operatorname{dom} g$ or not). The purpose of the agitator is to generate W_τ-changes to erase enumerations that ρ made into $T^{W_\tau}_\eta(n(\rho))$, when we see a change in the value of $g_s(n(\rho))$.

The outcomes of grandchildren nodes ρ are $\infty < 0 < 1 < \dots$, with ∞ denoting that either $n \notin \operatorname{dom} g$, or that $\Phi_\tau(A, W_\tau, q(\rho))\!\uparrow$.

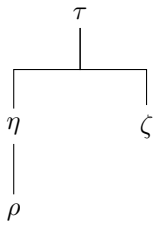

FIGURE 2. A family tree of $\mathcal{R}$ nodes. Nodes reproduce asexually; only female nodes have children.

Assigning requirements. To build the tree of strategies, we start with an ω-list of the requirements in which every $\mathcal{R}$ requirement appears infinitely often. If we already assigned an identity (and thus a requirement) for all the predecessors of a node α then we let $\mathtt{prec}(\alpha)$ be the set of $\mathcal{R}$ matriarch nodes τ such that $\tau\hat{}\infty \preccurlyeq \alpha$. We say that $\tau \in \mathtt{prec}(\alpha)$ is *active at* α if there is no son ζ of τ such that $\zeta\hat{}\infty \preccurlyeq \alpha$.

Identities are assigned to nodes as follows.

- If τ is an $\mathcal{R}$ matriarch node, then $\tau\hat{}\infty$ is a daughter of τ. It will be called the *eldest* daughter of τ.
- As mentioned above, all the immediate extensions of a $\mathcal{P}$ mother node σ are children π of σ.
- Let α be an $\mathcal{R}$ matriarch or an $\mathcal{R}$ grandchild node. If $\mathtt{prec}(\alpha)$ is nonempty and some $\tau \in \mathtt{prec}(\alpha)$ is still active at α, then every finite outcome of α starts a ζ-*block*. Let $\tau_0 \prec \tau_1 \prec \cdots \prec \tau_k$ be the list of those nodes in $\mathtt{prec}(\alpha)$ that are still active at α. Then each finite outcome of α is a son ζ_k of τ_k; every immediate successor of a ζ_k node (including the ∞ outcome) is a son ζ_{k-1} of τ_{k-1}; and so on, until we assign sons ζ_0 of τ_0.

  Then we restart requirements: following each ζ_0-node we add a sequence of daughter nodes η_i for i such that τ_i is still active, but that some τ_j for $j < i$ is no longer active.

  A sketch for $k = 2$ is given in Fig. 3.
- Otherwise, let α be a node on the tree, suppose that every predecessor of α has been assigned an identity, but that α has not yet been assigned an identity. Let k be the number of predecessors of α which are not π, ζ or η nodes. Consider the requirement appearing on the k^{th} position of our ω-list of requirements mentioned above. If this is a $\mathcal{P}$ requirement then we let α be a σ-node working for that requirement.

  If this requirement is an $\mathcal{R}$ requirement, then:

  – If no predecessor of α is a matriarch for this requirement, then we let α be a matriarch for this requirement.

  – Otherwise, let $\tau \prec \alpha$ be the (unique) matriarch for this requirement. If $\tau \in \mathtt{prec}(\alpha)$ and is still active at α, then we let α be a grandchild ρ of τ, and let $n(\rho)$ be the number of grandchildren of τ bewteen α and the longest daughter η of τ which is a predecessor of α. That node η will be ρ's mother.

– Otherwise (a matriarch $\tau \prec \alpha$ was assigned but is no longer active at α, or α extends a finite outcome of τ), we ignore this $\mathcal{R}$ requirement and try the next requirement on our list.

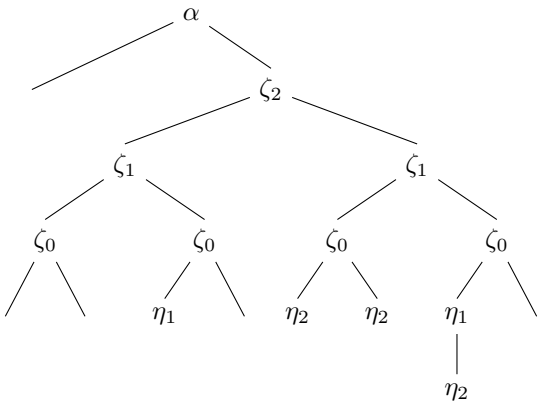

FIGURE 3. A ζ-block with three levels. Left is the infinite outcome.

Lemma 2.1. *Let f be an infinite path on the tree of strategies. Then:*

(1) *Every $\mathcal{P}$ requirement is assigned to a unique mother node σ and a unique child node π on f;*
(2) *On f, every $\mathcal{R}$ requirement is assigned to a unique matriarch τ and finitely many daughter nodes η. Either $\tau\hat{}t \in f$ for some $t \in \mathbb{N}$; or there is a son ζ of τ such that $\zeta\hat{}\infty$ is on f; or for the longest daughter η of τ on f, for every $n < \omega$ there is a unique child ρ of η on f such that $n = n(\rho)$.*

Construction. We start with all sets empty, no paramters defined, and $\gamma_0(k) = k$ for all $k < \omega$. For every node α which is accessible at some stage, we let $s^*(\alpha)$ be the least stage at which it is accessible.

Recall that we fixed an enumeration of $\emptyset'$ which occurs only at odd stages. Suppose that a number m enters $\emptyset'$ at an odd stage s. We enumerate $\gamma_s(m)$ into C_{s+1}. Recall that this means that $\gamma_{s+1}(n)$ is large for all $n \geqslant m$.

Now let $s > 0$ be even. We define the path of nodes accessible at stage s. A convenient terminology is "t is an α-stage" to mean that α is accessible at t. For each node α which we declare accessible, we describe what α does, whether it ends the stage or not, and if not, which outcome of α is chosen (i.e., which immediate successor of α is next accessible). The root node λ is accessible at every even stage.

If a $\mathcal{P}$ mother node σ is accessible: If $s = s^*(\sigma)$ then we define $k(\sigma)$ to be large and end the stage.

If $s > s^*(\sigma)$, let $t \leqslant s$ be the least σ-stage such that $\emptyset'_s \upharpoonright k(\sigma) = \emptyset'_t \upharpoonright k(\sigma)$. Let $\pi = \sigma\hat{}t$ be next accessible.

Now, if $s = s^*(\pi)$ then we appoint a new, large follower $y(\pi)$, and end the stage. Suppose that $s > s^*(\pi)$.

- If $y = y(\pi) \in E_s$ then we let $\pi\hat{}\mathtt{v}$ be next accessible.

Suppose that $y \notin E_s$. Let $t < s$ be the previous π-stage.

• If at stage t, a π-request token was issued, then we let $\pi\hat{}\mathtt{w}$ be next accessible. The same outcome is taken if it is not the case that $\Psi_\sigma(C,y)[s]{\downarrow}= 0$.

Suppose that no request token was issued by π at stage t, and also suppose that $\Psi_\sigma(C,y)[s]{\downarrow}= 0$.

• If $\mathtt{prec}(\pi)$ is nonempty, then we issue a request for π. The π-request token is placed on the eldest daughter $\eta = \tau\hat{}\infty$ of the longest node τ in $\mathtt{prec}(\pi)$. We end the stage.

• If $\mathtt{prec}(\pi)$ is empty, then we enumerate y into E_{s+1} and $\gamma_s(k(\sigma))$ into A_{s+1}. We end the stage.

If an $\mathcal{R}$ matriarch τ is accessible: If $s = s^*(\tau)$ then we let $\tau\hat{}\infty$ be next accessible.

Suppose that $s > s^*(\tau)$. Let t be the last $\tau\hat{}\infty$-stage before stage s. Let $\#(t)$ be the largest number mentioned by the construction at or before stage t. If for all $z < \#(t)$, $\Phi_\tau(A, W_\tau, z){\downarrow}= P(z)\ [s]$, then we let $\tau\hat{}\infty$ be next accessible. Otherwise, we let $\tau\hat{}t$ be next accessible.

If an $\mathcal{R}$ son node ζ is accessible: If $s = s^*(\zeta)$ then we appoint an agitator $p(\zeta)$ (with large value) and end the stage. If s is the next ζ-stage after $s^*(\zeta)$ then we let $\zeta\hat{}\infty$ be next accessible. Otherwise, let $t < s$ be the last $\zeta\hat{}\infty$-stage before stage s. Since every ζ-stage is a $\tau\hat{}\infty$-stage, where τ is ζ's mother, we know that $\Phi_\tau(A, W_\tau, p(\zeta)){\downarrow}\ [t]$. Let $u = \varphi_{\tau,t}(p(\zeta))$ be the use of that computation. If either $A_s \upharpoonright u \neq A_t \upharpoonright u$ or $W_{\tau,s} \upharpoonright u \neq W_{\tau,t} \upharpoonright u$ then we let $\zeta\hat{}\infty$ be next accessible. Otherwise, we let $\zeta\hat{}t$ be next accessible.

If an $\mathcal{R}$ daughter node η is accessible: If η is not the eldest daughter of its mother τ, then η's unique successor is next accessible.

Suppose that $\eta = \tau\hat{}\infty$ is τ's eldest daughter. If there is no request token currently residing at η, then we let η's unique successor be next accessible.

Suppose that there is a π-request token currently residing at η. There will be exactly one. Let σ be π's mother.

• We check to see if this request should be cancelled. Let r be the last stage at which π was accessible (this is the stage at which the request was made). Let $y = y(\pi)$. If C_s disagrees with C_r below the use $\psi_{\sigma,r}(y)$ of the computation $\Psi_\sigma(C,y)\,[r]$ then we cancel the request and remove the token; we let η's unique successor be next accessible.

• Suppose that the request is not cancelled. If $\mathtt{prec}(\tau)$ is nonempty then we let $\bar{\tau}$ be the longest node in $\mathtt{prec}(\tau)$. We transfer the π-request token from η to $\bar{\eta} = \bar{\tau}\hat{}\infty$, and end the stage.

• If the request is not cancelled and $\mathtt{prec}(\tau)$ is empty then we enumerate y into E_{s+1} and $\gamma_s(k(\sigma))$ into A_{s+1}. We end the stage.

If an $\mathcal{R}$ grandchild node ρ is accessible: Let τ be ρ's grandmother and let η be ρ's mother (the longest daughter of τ which is extended by ρ). Let $n = n(\rho)$.

If $s = s^*(\rho)$ then we appoint $q_{s+1}(\rho)$ to be large, and end the stage. If s is the next ρ-stage after stage $s^*(\rho)$ then we let $\rho\hat{}\infty$ be next accessible. Otherwise, let t be the last $\rho\hat{}\infty$-stage prior to stage s. Let $u = \varphi_{\tau,t}(q_t(\rho))$.

- If $q_t(\rho) \neq q_s(\rho)$, or if either $A_s \upharpoonright u \neq A_t \upharpoonright u$ or $W_{\tau,s} \upharpoonright u \neq W_{\tau,t} \upharpoonright u$, then we let $\rho\hat{}\infty$ be next accessible.
- If not, but $g_s(n) \neq g_t(n)$, then we enumerate $q_s(\rho)$ into P_{s+1}, choose $q_{s+1}(\rho)$ to be large, and let $\rho\hat{}\infty$ be next accessible.
- Otherwise, we enumerate $g_s(n)$ into $T_{\eta,s+1}^{W_{\tau,s}}(n)$ with use u, and let $\rho\hat{}t$ be next accessible.

At the end of the stage, before we move to the next stage, for every node ζ (an $\mathcal{R}$ son node) which lies to the right of the last accessible node and such that $p(\zeta)$ is defined and not already in P_s, we enumerate $p(\zeta)$ into P_{s+1}.

2.3. **Verification.** Let us first observe that the construction can be carried out as described. We note that every stage is finite: we will eventually reach a node that was not visited before and that will halt the stage. We also note that indeed, at any stage, at most one request token resides at a given eldest daughter $\eta = \tau\hat{}\infty$ of an $\mathcal{R}$ matriarch node τ. For a node extending η is accessible only if no token resides at η; only such nodes put tokens at η; and if a token is placed at η (by some $\mathcal{P}$ child node or by a different matriarch), the stage is immediately stopped, so at most one token is placed at any stage.

As promised, we observe that "initialised nodes" are not visited again.

Lemma 2.2. *Let $r < s$ be stages. Suppose that a node α is accessible at stage r, and that a node that lies to the left of α is accessible at stage s. Then α is not accessible after stage s.*

Proof. Let β be the longest initial segment of α which is accessible at stage s. So the outcome of β taken at stage s lies to the left of the outcome taken at stage r (the outcome extended by α). This only happens if β is an $\mathcal{R}$ node (a matriarch, a son or a grandchild) and the outcome taken at stage s is ∞. But then any outcome of β chosen after stage s is either ∞ or at least s, whereas the outcome taken at stage r is some β-stage smaller than r. □

We will later need an improvement upon Lemma 2.2.

Lemma 2.3. *Let σ be a $\mathcal{P}$ mother node. Let r be a stage. Suppose that $s^*(\sigma) < r$ but that a node that lies to the left of σ is accessible at stage r. Then at no stage $t \geqslant r$ do we enumerate $y(\pi)$ into E for any child π of σ.*

Proof. Let π be a child of σ and let $y = y(\pi)$. By Lemma 2.2, σ is not visited afer stage r; so any request for enumerating y into E is made at a stage $s < r$. We of course assume that $y \notin E_r$. Then $y \in E$ could only happen if a request is made by π before stage r and at stage r, this request is still uncancelled; at the beginning of stage r, the π-request token resides at $\tau\hat{}\infty$ for some $\tau \in \mathtt{prec}(\sigma)$.

Let α be a node to the left of σ which is accessible at stage r. Since $\tau\hat{}\infty \prec \sigma$, either τ resides to the right of α, or $\tau \in \mathtt{prec}(\alpha)$. In the first case, by Lemma 2.2, τ will not be visited at any stage $t \geqslant r$, and so the π-request token will never move from τ, and then $y \notin E$. Suppose that $\tau\hat{}\infty \preccurlyeq \alpha$; then $\tau\hat{}\infty \prec \alpha$. The stage r is a $\tau\hat{}\infty$-stage, but $\tau\hat{}\infty$ did not end stage r. Hence, the token that resided at $\tau\hat{}\infty$ at the beginning of stage r was cancelled at that stage, so again $y \notin E$. □

The true path is the path of nodes which are visited infinitely often, but no node to their left is visited infinitely often.

Lemma 2.4. *The true path is infinite.*

Proof. The root node lies on the true path. Let α be a node on the true path; we need to show that some successor of α is accessible infinitely often.

Suppose that α is a $\mathcal{P}$ mother node σ. Then σ ends the stage only once, at stage $s^*(\sigma)$ when it appoints $k(\sigma)$. If t is the least σ-stage such that $\emptyset'_t \upharpoonright k(\sigma) = \emptyset' \upharpoonright k(\sigma)$ then $\sigma\hat{}t$ is accessible at all σ-stages $s \geqslant t$.

Suppose that α is a $\mathcal{P}$ child node π. Suppose that t is a π-stage and that π ends stage t. Then a child of π is accessible at the next π-stage. The outcome chosen is always $\mathtt{w}$, or switches to $\mathtt{v}$ once $y(\pi)$ is enumerated into E.

The usual Σ_2/Π_2 behaviour holds when α is an matriarch, a son or a grandchild $\mathcal{R}$ node. Such nodes end the stage at most once.

Suppose that $\alpha = \eta$ is a daughter of an $\mathcal{R}$ matriarch τ. If η is not τ's eldest daughter, then η never ends the stage. Suppose that $\eta = \tau\hat{}\infty$ is τ's eldest daughter. Suppose that at stage t, the node η ends the stage. This is because it had some request token at the beginning of the stage. As explained above, at the next η-stage, there is no request token residing at η, and so η's unique successor is accessible then. □

The global requirement is met.

Lemma 2.5. $\emptyset' \leqslant_{\mathrm{T}} A \oplus C$.

Proof. Since we obey the usual rules for markers, it suffices to show that every marker is changed only finitely many times. Let $k \in \mathbb{N}$. Other than a number $\leqslant k$ entering $\emptyset'$, the only prompt for changing the marker $\gamma_s(k)$ is if $k \geqslant k(\sigma)$ for some $\mathcal{P}$ mother node σ, and $\gamma_s(k(\sigma))$ is enumerated into A_{s+1} because of action for a child π of σ. As observed above, only finitely many children of σ are ever accessible, and for each one, we act by enumeration at most once. Since the value $k(\sigma)$ is chosen large, there are only finitely many nodes σ with $k(\sigma) \leqslant k$. □

Every positive requirement is met.

Lemma 2.6. $\emptyset' \not\leqslant_{\mathrm{T}} C$.

Proof. Let Ψ be a functional; by Lemma 2.1 there is a $\mathcal{P}_\Psi$ mother node σ on the true path. Let π be the child of σ that lies on the true path. Let $y = y(\pi)$. We need to argue two things:

- If $\Psi(C, y)\downarrow = 0$ then $y \in E$.
- If $y \in E$ then $\Psi(C, y)\downarrow = 0$.

For the first, let t be a π-stage sufficiently late so that $\Psi(C, y)\downarrow = 0\,[t]$ by a C-correct computation; and suppose that $y \notin E_t$. If $\pi\hat{}\mathtt{w}$ is accessible at stage t because at the previous stage a π-request token was issued and later cancelled, replace t by the next π-stage. After doing that, we see that a π-request token is issued at stage t. This request token cannot be cancelled. By reverse induction on the nodes $\tau \in \mathtt{prec}(\pi)$, we see that the request token will be passed to $\tau\hat{}\infty$, and at the shortest such τ (or at stage t if $\mathtt{prec}(\pi)$ is empty), y will be enumerated into E.

For the second, suppose that y is enumerated into E at stage t. At that stage, since the request token is not cancelled, we still have $\Psi_\sigma(C, y)\downarrow = 0\,[t]$. We argue that this computation is C-correct; let $u = \psi_{\sigma,t}(y)$ be its use. At stage t, $\gamma_t(k(\sigma))$ is enumerated into A, and so all markers $\gamma_s(m)$ for $s > t$ and $m \geqslant k(\sigma)$ are greater than u. Since π lies on the true path and is accessible before stage t, we know that

$\emptyset'_t \upharpoonright k(\sigma) = \emptyset' \upharpoonright k(\sigma)$. Hence no enumeration into $\emptyset'$ will ever cause a marker $\gamma_s(m)$ for $m < k(\sigma)$ to be enumerated into C at stage $s > t$. It follows that $C_t \upharpoonright u = C \upharpoonright u$ as required.

Hence $E \not\leqslant_{\mathrm{T}} C$; since E is c.e., the lemma follows. □

It remains to show that every cupping partner of A is high – that the $\mathcal{R}$ requirements are met. Fix a c.e. set W and a functional Φ, and suppose that $\Phi(A, W) = P$; such a functional exists if $\emptyset' \leqslant_{\mathrm{T}} A \oplus W$. By Lemma 2.1 there is an $\mathcal{R}_{W,\Phi}$ matriarch τ on the true path (so $W = W_\tau$ and $\Phi = \Phi_\tau$). Since $\Phi(A, W)$ is total and equals P, $\tau^\frown\infty$ lies on the true path. Then by the same lemma, there is a longest daughter η of τ on the true path. We need to show that T^W_η is a trace for the universal Σ^0_2 function g. That is, we need to show:

- For all n, $T^W_\eta(n)$ is finite; and
- For all $n \in \operatorname{dom} g$, $g(n) \in T^W_\eta(n)$.

We quickly dispose of the second.

Lemma 2.7. *For all $n \in \operatorname{dom} g$, $g(n) \in T^W_\eta(n)$.*

Proof. Since $\Phi(A, W)$ is total, for no son ζ of τ can we have $\zeta^\frown\infty$ on the true path. That is, τ is active at every node on the true path extending τ. By Lemma 2.1, let ρ be a child of η on the true path such that $n = n(\rho)$.

We argue that there are only finitely many $\rho^\frown\infty$-stages. Since $\langle g_s(n)\rangle$ eventually stabilises, the value $q_s(\rho)$ will be redefined only finitely many times. Let $q = \lim_s q_s(\rho)$ be the last value chosen.

Since $\Phi(A, W)$ is total, $\Phi(A, W, q)\!\downarrow$; let $u = \varphi(q)$ be the use of this computation. If t is a $\rho^\frown\infty$-stage, q was chosen before stage t and the correct computation $\Phi(A, W, q)$ has appeared by stage t, then there will be no more $\rho^\frown\infty$-stages.

Let t be the last $\rho^\frown\infty$-stage. So $q = q_t(\rho)$ and the computation $\Phi(A, W, q)[t]$ is correct, with use u. At any ρ-stage $s > t$ we enumerate $g(n)$ into $T^W_\eta(n)$ with use u; this enumeration is permanent. □

We now work toward showing that every $T^W_\eta(n)$ is finite. We define a c.e. set $\mathcal{B}$ of nodes which are guaranteed to not be on the true path:

- If σ is a $\mathcal{P}$ mother node, s is a σ-stage, $t < s$ and $\emptyset'_s \upharpoonright k(\sigma) \neq \emptyset'_t \upharpoonright k(\sigma)$, then all nodes $\alpha \succcurlyeq \sigma^\frown t$ are added to $\mathcal{B}_s$.
- If π is a $\mathcal{P}$ child node and $y(\pi)$ is enumerated into E at stage s, then all nodes $\alpha \succcurlyeq \pi^\frown \mathtt{w}$ are added to $\mathcal{B}_s$.
- If $\tilde{\zeta}$ is a son of an $\mathcal{R}$ matriarch $\tilde{\tau} \in \mathtt{prec}(\tau)$ and $\tilde{\zeta}^\frown\infty$ is accessible at stage s, then for all $t < s$, all nodes extending $\tilde{\zeta}^\frown t$ are added to $\mathcal{B}_s$.

Note that indeed no node in $\mathcal{B}_s$ is accessible at or after stage s.

Lemma 2.8. *Suppose that α is either a son ζ of τ or a child ρ of η. Let r be an $\alpha^\frown\bar{r}$-stage (for some $\bar{r} \in \mathbb{N}$).*

- *If $\alpha = \zeta$ is a son, let $q = p(\zeta)$.*
- *If $\alpha = \rho$ is a grandchild, let $q = q_r(\rho)$.*

Let $u = \varphi_r(q)$. Let $s \geqslant r$ be a $\tau^\frown\infty$-stage, and suppose that $W_r \upharpoonright u = W_s \upharpoonright u$. Suppose that $\alpha \notin \mathcal{B}_s$. Then:

1. *$A_r \upharpoonright u = A_s \upharpoonright u$, so $\varphi_s(q) = u$.*
2. *No node that lies to the left of $\alpha^\frown\bar{r}$ is accessible at any stage $t \in [r, s)$.*

Proof. We prove both parts of the lemma by simultaneous induction on the $\tau^\frown\infty$-stages $s \geqslant r$. The lemma holds at $s = r$ vacuously. Let $s > r$ be a $\tau^\frown\infty$-stage. Let $\bar{s} \geqslant r$ be the previous $\tau^\frown\infty$-stage before stage s, and suppose that the lemma held for stage $\bar{s}$. We show it holds at s. Let α be a node as described; in particular, $\alpha \notin \mathcal{B}_s$.

We start with (1). Every number enumerated into A is $\gamma_t(k(\sigma))$ for some $\mathcal{P}$ mother node σ. Let σ be a $\mathcal{P}$ mother node, and suppose that we enumerate $\gamma_t(k(\sigma))$ at $t \in [\bar{s}, s)$ on behalf of a child π of σ; we show that $\gamma_t(k(\sigma)) > u$. The argument depends on the location of σ. Let $\bar{t}$ be the stage at which the request for π (which resulted in this enumeration) was made.

- Suppose that σ lies to the right of $\alpha^\frown\infty$. As the latter is accessible at stage $\bar{r} < r$, by Lemma 2.3, $s^*(\sigma) > \bar{r} > u$, and of course $\gamma_t(k(\sigma)) \geqslant k(\sigma) > s^*(\sigma)$.
- Suppose that $\sigma \prec \alpha$, but that π does not lie to the left of α. In this case we claim that $\alpha \in \mathcal{B}_s$. Either $\pi \prec \alpha$; in this case, since $y(\pi) \notin E_r$, α extends $\pi^\frown\mathtt{w}$; so α is placed into $\mathcal{B}$ at stage t.

  Alternatively, π lies to the right of α. The children of σ are visited from the left to the right; we never return to children on the left. It follows that $\bar{t} > r$, i.e., that π is visited after stage r; at that stage (before stage s), α is placed into $\mathcal{B}$.
- The interesting case is when π lies to the left of $\alpha^\frown\bar{r}$; here we use the delayed enumeration feature of the construction. In this case, the node π is not accessible at any stage in the interval $(\bar{r}, r]$ (consider the cases $\sigma \succcurlyeq \alpha^\frown\infty$ and π lying to the left of α; for the latter, use Lemma 2.2). By induction, π is not accessible at any stage in the interval $[r, \bar{s})$. Hence either $\bar{t} \leqslant \bar{r}$ or $\bar{t} = \bar{s}$.

  Suppose that $\bar{t} = \bar{s}$. The node π is the last accessible node at stage $\bar{t}$. As $\bar{s}$ is a $\tau^\frown\infty$-stage, it must be that $\pi \succ \tau^\frown\infty$, i.e., $\tau \in \mathtt{prec}(\pi)$. At stage $\bar{s}$, the π-token is placed at $\tilde{\tau}^\frown\infty$ for some $\tilde{\tau} \in \mathtt{prec}(\sigma)$, with $\tilde{\tau} \succcurlyeq \tau$. There is no $\tilde{\tau}^\frown\infty$-stage between stages $\bar{s}$ and s, and so $y(\pi)$ is not enumerated into E prior to stage s.

  Now suppose that $\bar{t} \leqslant \bar{r}$. At the beginning of stage r, the π-token lies at $\tilde{\eta} = \tilde{\tau}^\frown\infty$ for some $\tilde{\tau} \in \mathtt{prec}(\sigma)$. Now where is $\tilde{\eta}$?
  - If $\tilde{\eta}$ lies to the left of τ then by Lemma 2.2, as τ is accessible at stage r, and lies on the true path, $\tilde{\eta}$ cannot be accessible after stage r.
  - $\tilde{\eta}$ cannot lie to the right of $\alpha^\frown\bar{r}$, as $\tilde{\eta} \prec \pi$ and π lies to the left of $\alpha^\frown\bar{r}$. Similarly, $\tilde{\eta}$ does not extend $\alpha^\frown\bar{r}$.
  - $\tilde{\eta} \prec \alpha$ is impossible, as then r would be a $\tilde{\eta}$-stage, and $\tilde{\eta}$ would end that stage.
  - The last case is $\tilde{\eta} \succ \tau^\frown\infty$ but $\tilde{\eta}$ lies to the left of $\alpha^\frown\bar{r}$. In this case, by induction, $\tilde{\eta}$ is not accessible between stages r and $\bar{s}$. It may conceivably be accessible at stage $\bar{s}$ (though below we show this is not the case), but even if it is, at the end of stage $\bar{s}$, the token resides at some $\hat{\eta}$ with $\tau^\frown\infty \preccurlyeq \hat{\eta} \preccurlyeq \tilde{\eta}$, and there is no $\hat{\eta}$-stage stage between stages $\bar{s}$ and s.

That concluded the proof of (1) for stage s. We now verify (2) at stage s. Let β be a node that lies to the left of $\alpha^\frown\bar{r}$, and suppose that β is accessible at some stage $t \in [\bar{s}, s)$. As mentioned above, it is impossible that β lies to the left of τ, so $\beta \succcurlyeq \tau^\frown\infty$. Hence $t = \bar{s}$.

Now there are two cases, depending on whether β extends $\alpha^\frown\infty$ or not.

Suppose that $\beta \succcurlyeq \alpha^\frown\infty$ (so we can take $\beta = \alpha^\frown\infty$). By induction, $\bar{r}$ is the last $\alpha^\frown\infty$-stage prior to stage $\bar{s}$. By assumption, $W_{\bar{r}} \upharpoonright u = W_r \upharpoonright u = W_{\bar{s}} \upharpoonright u$, and by induction, $A_{\bar{r}} \upharpoonright u = A_r \upharpoonright u = A_{\bar{s}} \upharpoonright u$. Hence it must be that $\alpha = \rho$ is a grandchild of τ and that $g_{\bar{s}}(n(\rho)) \neq g_{\bar{r}}(n(\rho))$, whence q is enumerated into P at stage $\bar{s}$.

Now by (1) at stage $\bar{s}$, $\varphi_{\bar{s}}(q) = u$, but as $q \in P_s$ and s is a $\tau^\frown\infty$-stage, $\Phi(A, W, q)[s] = 1 \neq 0 = \Phi(A, W, q)[\bar{s}]$. But by (1) at stage s, and by assumption, $W_{\bar{s}} \upharpoonright u = W_s \upharpoonright u$ and $A_{\bar{s}} \upharpoonright u = A_s \upharpoonright u$, making the change in the value of the computation impossible. So this cannot happen.

Suppose that β lies to the left of α.

Claim. There is some son $\zeta \preccurlyeq \alpha$ of τ which lies to the right of β.

Proof. We may of course assume that α is not a son of τ, i.e. is a grandchild.

Let δ be the longest common initial segment of α and β. Then $\delta \succ \tau^\frown\infty$, and must be some $\mathcal{R}$ matriarch, son or grandchild (not necessarily for the same $\mathcal{R}$ requirement as τ's); $\beta \succcurlyeq \delta^\frown\infty$ and $\alpha \succcurlyeq \delta^\frown v$ for some $v \in \mathbb{N}$.

Since τ is active at α, it is also active at δ. If δ is an $\mathcal{R}$ matriarch or grandchild, then $\delta^\frown v$ starts a ζ-block, one level of which will consist of sons of τ; α will extend one of them.

Suppose that δ is a son of an $\mathcal{R}$-matriarch $\tilde{\tau}$; it is a part of some ζ-block, which also includes a son $\zeta \prec \alpha$ of τ. We consider the ordering between τ and $\tilde{\tau}$.

- If $\tau \prec \tilde{\tau}$ then $\zeta \succ \delta$, and so is as required.
- If $\tilde{\tau} \prec \tau$ then α is added to $\mathcal{B}_{\bar{s}}$.
- Suppose that $\tau = \tilde{\tau}$, so $\delta = \zeta$. Then $v < \bar{r}$ is the last $\zeta^\frown\infty$-stage prior to stage r, and by induction, in fact, v is the last $\zeta^\frown\infty$-stage prior to stage $\bar{s}$. Let $\bar{u} = \varphi_v(p(\zeta))$. Since there is no ζ-stage between stage v and r, $A_v \upharpoonright \bar{u} = A_r \upharpoonright \bar{u}$ and $W_v \upharpoonright \bar{u} = W_r \upharpoonright \bar{u}$. It follows that $\bar{u} = \varphi_r(p(\zeta))$.

  Since $\zeta \prec \alpha$, $p(\zeta) < q$, and so $\bar{u} = \varphi_r(p(\zeta)) \leqslant \varphi_r(q) = u$. It follows by (1) at stage $\bar{s}$ then that $A_r \upharpoonright \bar{u} = A_{\bar{s}} \upharpoonright \bar{u}$ and $W_r \upharpoonright \bar{u} = W_{\bar{s}} \upharpoonright \bar{u}$. Altogether we see that $A_v \upharpoonright \bar{u} = A_{\bar{s}} \upharpoonright \bar{u}$ and $W_v \upharpoonright \bar{u} = W_{\bar{s}} \upharpoonright \bar{u}$. But then the instructions show that $\zeta^\frown v$, and not $\zeta^\frown\infty$, is accessible at stage $\bar{s}$. □

Let ζ be as guaranteed by the claim. At stage $\bar{s}$, $p(\zeta)$ is enumerated into P. The argument now repeats: since $p(\zeta) < q$, $\varphi_{\bar{s}}(p(\zeta)) \leqslant \varphi_{\bar{s}}(q) = u$, and so the enumeration of $p(\zeta)$ into P necessitates a change in either A or W below $\varphi_{\bar{s}}(p(\zeta))$, and hence below u, by the next $\tau^\frown\infty$-stage s. By assumption, and part (1) for stage s, which we proved above, this does not happen — a contradiction. This concludes the proof of Lemma 2.8. □

The following lemma concludes the proof of Theorem 1.2.

Lemma 2.9. *For all n, $T^W_\eta(n)$ is finite.*

Proof. Let $n < \omega$.

First, let ρ be the child of η on true path for which $n = n(\rho)$. Note that $\rho \notin \mathcal{B}$ as it lies on the true path. Suppose that at some stage r, ρ enumerates a number $x = g_r(n)$ into $T^W_\eta(n)$ with some use u. Then for some $\bar{r} < r$, $\rho^\frown\bar{r}$ is accesible at stage r; and $u = \varphi_r(q_r(\rho))$. If there is some $\rho^\frown\infty$-stage $t > r$, then by Lemma 2.8, $W_s \upharpoonright u \neq W_r \upharpoonright u$, where s is the next $\tau^\frown\infty$-stage after stage t. Otherwise, $g_t(n) = g_r(n)$ for every ρ-stage $t > r$. It follows that at most one number in $T^W_\eta(n)$ is there because of an enumeration that ρ makes.

Nodes to the left of ρ of course make only finitely many enumerations into $T^W_\eta(n)$, and nodes extending ρ make no enumerations into $T^W_\eta(n)$. So it suffices to show that nodes that lie to the right of the true path do not make any permanent enumerations into $T^W_\eta(n)$.

Let $\tilde{\rho}$ be a child of η with $n(\tilde{\rho}) = n$ that lies to the right of the true path.

Claim. There is a son $\zeta \prec \tilde{\rho}$ of τ which lies to the right of the true path, such that $\zeta \notin \mathcal{B}$.

Proof. Let δ be the longest common initial segment of $\tilde{\rho}$ and the true path. Again, δ is a matriarch, son, or grandchild for some $\mathcal{R}$ requirement; $\delta\hat{}\infty$ lies on the true path, and $\tilde{\rho}$ extends $\delta\hat{}r$ for some $r \in \mathbb{N}$.

Some options are impossible. The node δ is not a son of τ, as the infinite outcome of a son of τ cannot be on the true path ($\Phi(A, W)$ is total). Similarly, δ is not the son of some matriarch $\bar{\tau} \prec \tau$: Otherwise, following $\delta\hat{}\infty$ there will be another daughter of τ on the true path. But $\tilde{\rho}$, and hence δ, extend η, which is the longest daughter of τ on the true path.

Hence, either δ is a matriarch or a grandchild, or is a son of some matriacrh $\bar{\tau} \succ \tau$. In all of these cases, there is some son ζ of τ such that $\delta\hat{}r \preccurlyeq \zeta \prec \tilde{\rho}$ and strictly between δ and ζ there are only sons of matriarchs $\bar{\tau} \succ \tau$. This last fact shows that $\zeta \notin \mathcal{B}$; if $\zeta \in \mathcal{B}$ then $\delta \in \mathcal{B}$, and δ lies on the true path. □

Suppose that at some stage r, $\tilde{\rho}$ makes an enumeration into $T^W_\eta(n)$, with some use u. We want to show that $W \upharpoonright u \neq W_r \upharpoonright u$. Let ζ be a son of τ given by the claim. Note that $\zeta\hat{}\bar{r} \preccurlyeq \tilde{\rho}$ for some $\bar{r} < r$, as no grandchild of τ extends $\zeta\hat{}\infty$. So r is a $\zeta\hat{}\bar{r}$-stage. As we observed above, $\varphi_r(p(\zeta)) \leqslant u$. Let $s > r$ be a $\tau\hat{}\infty$-stage such that some node the the left of ζ is accessible between stages r and s. Then by Lemma 2.8 applied to ζ, $W_s \upharpoonright u \neq W_r \upharpoonright u$. This shows that indeed, no enumeration into $T^W_\eta(n)$ made by a node that lies to the right of the true path is permanent. □

3. Every cuppable degree has a non-superhigh cupping partner

In this section we show that Theorem 1.2 is sharp:

Theorem 3.1. *Every cuppable degree has a cupping partner which is not superhigh. A non-superhigh cupping partner can be found below any given cupping partner.*

In fact, as we soon discuss, we can always find a cupping partner relative to which $\emptyset'$ is not c.e. traceable, equivalently, not array computable (Ishmukhametov [Ish99]). The remainder of this section is devoted to the proof of Theorem 3.1.

3.1. **The Requirements.** Fix a cuppable c.e. set A; let C be a c.e. set which is a cupping partner of A. We shall enumerate a c.e. set $X \leqslant_{\mathrm{T}} C$, a cupping partner of A, which is not superhigh. As in the previous proof, to ensure that $\emptyset' \leqslant_{\mathrm{T}} A \oplus X$ we employ movable markers $\delta_s(n)$; we allow $\delta_{s+1}(n) \neq \delta_s(n)$ only if a number $x \leqslant \delta_s(n)$ enters either A or X at stage s.

Also as above, we enumerate an auxiliary c.e. set P, a private copy of $\emptyset'$. By the fixed point theorem, we may assume that we are given a functional Γ such that $P = \Gamma(A, C)$. Again we assume that the A- and C-use of Γ are always the same. By speeding up the enumerations of A and C, we assume that at every stage s we have agreement between $\Gamma_s(A_s, C_s)$ and P_s which is long enough for our purposes. Thus, if we enumerate a number m we previously prepared into P at stage s (meaning

that $m \in P_{s+1} \setminus P_s$) then we see some $x < \gamma_s(m)$ enter either A or C at that stage as well.

We will ensure that $\emptyset'$ is not c.e. traceable relative to X, which implies that it is not superhigh. We know that a c.e. degree is c.e. traceable if and only if every function in that degree has a c.e. trace T with size bounded by the identity: $|T(n)| < n$ for all n.[2] Relativising, to show that $\emptyset'$ is not c.e. traceable relative to X, we will construct a Δ^0_2 function $f\colon \omega \to \omega$ and ensure that the following requirements are met for all e:

$\mathcal{R}_e$**:** There exists z such that $f(z) \notin T_e^X(z)$.

Here $\langle T_e \rangle$ is an effective list of all oracle trace operators with $|T_e^Y(n)| < n$ for every e, n, and every oracle Y. Obviously we will build a computable approximation $\langle f_s \rangle$ for f.

3.2. **A first attempt.** Let us consider some basic aspects of the construction. Coding $\emptyset'$ requires enumerating numbers into X when no corresponding changes are given by A. On the other hand, the basic idea for meeting $\mathcal{R}_e$ is to freeze X. Namely, we set a value $f_{s_0}(z)$ for some z; we wait for it to show up in $T_e^X(z)$, say at some stage $s_1 > s_0$. Let v be the X-use of this enumeration; we then want to preserve $X \upharpoonright v$, change $f_{s_1}(z)$ to be some new large value, and repeat. If we do this z many times we must win. Further, another requirement is that $f_s(z)$ does reach a limit $f(z)$.

The natural approach to balancing these competing requirements is to try to clear markers $\delta(n)$ beyond the various uses v mentioned above. We hope to achieve this by getting A-changes below these markers. To get these changes, we use *agitators*, which are numbers that we enumerate into P. If $\gamma(a) < \delta(n)$ and we enumerate a into P then we are guaranteed a change below $\delta(n)$ in *either* A *or* C. The former is desirable. To make use of the latter, we need to employ the incompleteness of C. We threaten to compute $\emptyset'$ from C by building a functional Ψ. The naive plan is then:

(i) Prepare z many agitators a with $\gamma(a) \leqslant \psi(z), \delta(z)$; wait for z to enter $\emptyset'$.
(ii) When we see the current version of $f(z)$ enumerated into $T_e^X(z)$ with use v, enumerate one of the agitators a into P, and observe the change we get:
  (a) If the change is in C, we get to correct $\Psi(C, z)$; we give up on z. We enumerate $\delta(z)$ into X to record the change in $\emptyset'$.
  (b) If the change is in A, then we can lift $\delta(z)$ beyond v without enumerating it into X, preserving $X \upharpoonright v$. We then enumerate another agitator into P and repeat.

The idea is that since $\Psi(C)$ cannot equal $\emptyset'$, for some z we will only get A-changes, allowing us to fill up $T_e^X(z)$, so that the last value of $f(z)$ will not show up in that trace. Note that if in the meantime some smaller $z' < z$ enters $\emptyset'$, we can abandon our attack on z and start one for z'; of course a correction of $\Psi(C, z')$ will also yield a correction for $\Psi(C, z)$.

So what goes wrong? Surprisingly, the difficulty is not in the interaction between the requirements. We can let each requirement control infinitely many inputs for f; we can let each requirement build its own threat Ψ_e for computing $\emptyset'$ from C. As long as we set up sufficiently many agitators, the requirements can share them.

[2]We stipulate that $|T(0)| = |T(1)| = 1$.

Actually, the real difficulty is in *spontaneous A-changes*. For the naive plan above to work, we set up z many agitators a, and we need to keep $\gamma(a)$ smaller than $\psi(z)$. But a small number going into A, not necessarily in response to an enumeration into P, may cause some or all of these $\gamma(a)$ to grow beyond $\psi(z)$, without us gaining anything toward meeting $\mathcal{R}_e$.

3.3. **The solution: cascading attempts.** How can we utilise spontaneous A-changes? The idea is the following. We will arrange our agitators by blocks $G(y)$, and ensure that $\delta(y) > \gamma(a)$ for every agitator a in the y^{th} block $G(y)$; see Fig. 4. When trying to attack with y, we relax our attempt to clear the marker $\delta(y)$ beyond the uses v; rather, we only try to clear $\delta(y+1)$. Nonetheless we use the agitators in the block $G(y)$ for this attack, rather than the agitators in $G(y+1)$. If there is a spontaneous A-change somewhere in the block $G(y)$, that would spoil our attack, we see that this change allows us to lift $\delta(y)$ — thereby allowing us to attack with $y-1$ instead.

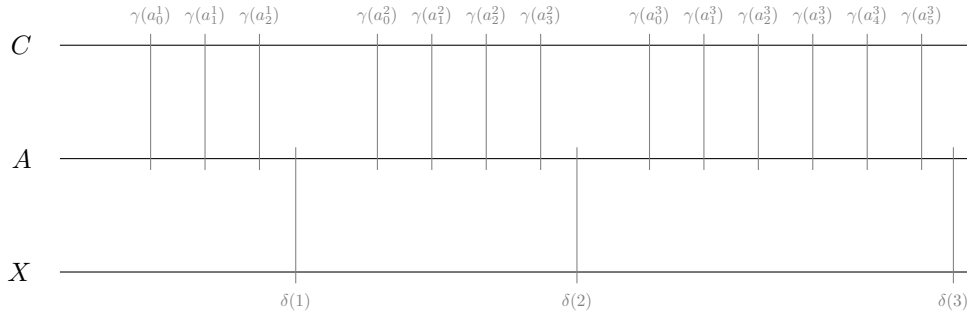

FIGURE 4. $G(1) = \{a_0^1, a_1^1, a_2^1\}$, etc.

The way we interleave the various requirements is as follows. We let $\mathcal{R}_e$ control the f-inputs $\langle e, y\rangle$ for $y > e$. Toward meeting $\mathcal{R}_e$ on input $\langle e, y\rangle$, we will allow the requirement to use some agitators from the block $G(y)$. Ignoring reassignments of agitators, on input y, $\mathcal{R}_e$ will need at most $\langle e, y\rangle$ many agitators (because $|T_e^X(\langle e, y\rangle)| < \langle e, y\rangle$); to ensure that we never run out of agitators, we will set the number of agitators in $G(y)$ to be

$$N_y = 1 + \sum_{e<y} \langle e, y\rangle.$$

To ensure that we do not waste any agitators, and minding that we need to keep $\gamma(a) < \delta(y)$ for $a \in G(y)$, we see that we have to use the agitators backwards, always enumerating the largest possible agitator into P.

Notation. For typographical nicety, we write $f(e, y)$ for $f(\langle e, y\rangle)$ and $T_e^X(e, y)$ for $T_e^X(\langle e, y\rangle)$.

Also, if $f(e, y) \in T_e^X(e, y)\,[s]$ then we write $u_{e,s}(y)$ for the use of enumerating this value in the trace; otherwise we say that $u_{e,s}(y)$ is undefined. We use the convention that the use of enumerating z in a trace is at least z, so $u_{e,s}(y) \geqslant f_s(e, y)$ (if defined).

The guiding principles for the construction are:

(1) Maintaining the general structure indicated in Fig. 4: at every stage,
$$\gamma(\max G(y)) < \delta(y) < \gamma(\min G(y+1)).$$
(2) Making sure that we never run out of agitators: we need to ensure that when we use an agitator in $G(y)$ on behalf of some $e < y$, we protect a new element of $T_e^X(e, y)$.
(3) Making sure that $f_s(e, y)$ reaches a limit; this we do by only changing the current value of $f(e, y)$ if an A-change causes $\delta(y+1)$ to grow beyond $u_e(y)$.
(4) For the sake of computing $\emptyset'$ from $A \oplus X$, we need to ensure that each marker $\delta(y)$ eventually stabilises. In light of the requirement $\delta(y) > \gamma(\max G(y))$, this implies that we need $G(y)$ to stabilise as well.
(5) Keeping our ability to change $\Psi_e(C, y)$ if y enters $\emptyset'$. For this, our aim is to keep $\psi_e(y) \geqslant \delta(y)$ (as this is greater than $\gamma(\max G(y))$). However, the main point is that our renewed strategy requires more, namely, we aim for $\psi_e(y) \geqslant \delta(y+1)$. By induction, if this holds for $y-1$, then we immediately get $\psi_e(y) \geqslant \delta(y)$. Thus, the dangerous situation is when $\delta(y) \leqslant \psi_e(y) < \delta(y+1)$, which is when we use agitators to change things.

3.4. **Construction.** The construction at stage s consists of the following steps.

Step 1 - enumerating agitators: Let a be the smallest of:
- $\min G_s(n)$, where $n \in \emptyset'_{s+1} \setminus \emptyset'_s$; and
- $\max G_s(y)$, for the least y such that for some $e < y$,
$$\delta(y) \leqslant \psi_e(y) < \delta(y+1) \leqslant u_e(y)\ [s]$$
(in particular $\psi_{e,s}(y)\downarrow$ and $u_{e,s}(y)\downarrow$).

Enumerate a into P_{s+1}.

Step 2 - responding to changes: Let x be the smallest element in either $A_{s+1} \setminus A_s$ or $C_{s+1} \setminus C_s$. We know that $x < \gamma_s(a)$. Let c be the smallest agitator (at stage s) such that $x < \gamma_s(c)$; let y such that $c \in G_s(y)$.
(a) Suppose that $x \in C_{s+1}$.
  (i) If $c = a$ and it is not the case that $y \in \emptyset'_{s+1} \setminus \emptyset'_s$, enumerate $\delta_s(y+1)$ into X_{s+1}.
  (ii) Otherwise, enumerate $\delta_s(y)$ into X_{s+1}.
(b) In all cases, reappoint agitators above c and markers as described below.

Step 3 - updating f: For each $e < y < s$ such that
$$\max\{\psi_{e,s}(y), u_{e,s}(y)\} < \delta_{s+1}(y+1)$$
(in particular both $\psi_{e,s}(y)\downarrow$ and $u_{e,s}(y)\downarrow$), redefine $f_{s+1}(e, y)$ to be large.

Step 4 - defining Ψ_e: For each $e < s$,
(a) if $\delta_{s+1}(e+1) > \delta_s(e+1)$, restart Ψ_e.
(b) let $y > e$ be least such that $\Psi_{e,s}(C_{s+1}, y)\uparrow$. If $u_{e,s+1}(y)\downarrow$, define $\Psi_e(C, y) = \emptyset'(y)\ [s+1]$ with use
$$\psi_e(y) = \max\{u_e(y), \delta(y+1), \psi_e(y-1)\}\ [s+1],$$
where we always let $\psi_e(e) = 0$.

Reappointing agitators above c and markers $\delta(n)$, if $c \in G_s(y)$, means: discarding all agitators greater than c; and appointing new large agitators and markers so that:
(i) For all $z > y$, $|G_{s+1}(z)| = N_z$;
(ii) If $\delta_{s+1}(y) > \delta_s(y)$, then $|G_{s+1}(y)| = N_y$;

(iii) For all n, $\gamma(\max G(n)) < \delta(n) < \gamma(\min G(n+1))$ $[s+1]$.

We do this according to the following cases:

(1) If $y \in \emptyset'_{s+1} \setminus \emptyset'_s$, then $G_{s+1}(y)$ consists of all new agitators, and $\delta_{s+1}(y)$ is large.
(2) Otherwise, if $c < a$, we let $\delta_{s+1}(y)$ be large, and add followers to $G_{s+1}(y)$ (but keep those $b \in G_s(y)$ with $b \leqslant c$).
(3) Otherwise, we let $G_{s+1}(y) = G_s(y) \setminus \{a\}$, and let $\delta_{s+1}(y) = \delta_s(y)$.

In all cases we preserve $\delta(y-1)$ and all $G(z)$ for $z < y$; and for $z > y$ we completely redefine $G_{s+1}(z)$ and let $\delta_{s+1}(z)$ be large. In making these definitions, we use our assumption that the enumerations of A and C are sped-up sufficiently so that the agreement between $\Gamma(A, C)\,[s+1]$ and P_{s+1} is sufficiently long to enable us to find sufficiently many new agitators a with $\gamma_{s+1}(a)$ already defined.

Remarks. Before we formally verify that the construction works, we explain some of the details. In case (3) above (when $a = c$), we leave $\delta_{s+1}(y) = \delta_s(y)$ because we do not want to enumerate $\delta_s(y)$ into X_{s+1}. This is important of course only if the change below $\gamma_s(c)$ is in C rather than A. The point is that in this case, to justify using the agitator a, we need to protect $X_s \upharpoonright \delta_s(y+1)$, as this oracle enumerates elements of $T_e^X(e, y)\,[s]$ that account for some agitators used.

Above we mentioned that the dangerous situation that requires enumerating an agitator is when $\delta(y) \leqslant \psi_e(y) < \delta(y+1)$. However in the construction we added the clause $\delta(y+1) \leqslant u_e(y)$. This is important because otherwise we could keep enumerating followers after $\delta(y+1)$ and $\psi_e(y)$ have stabilised. When accounting for the agitators used (Lemma 3.4 below), this requirement shows that the next value in $T_e^X(e, y)$, that will account for the agitator we use, was not an old value that was already used to account for older agitators.

The idea behind restarting Ψ_e when $\delta(e+1)$ changes is being the base case in an inductive argument (see the proof of Lemma 3.9 below). The idea is that if we cannot correct $\Psi_e(C, y)$, it is because we could possibly make progress on meeting $\mathcal{R}_e$ on input $y-1$. If this fails as well, we look at $y-2$, ...; but at the lowest y, namely $y = e+1$, we cannot go lower, so when this attack fails, we must simply restart the whole process, starting with a fresh version of Ψ_e.

3.5. **Verification.**

Lemma 3.2. *For all s, for all $y < s$, $\gamma(\max G(y)) < \delta(y) < \gamma(\min G(y+1))$ $[s]$.*

Lemma 3.3. *$X \leqslant_{\mathrm{T}} C$. In fact, for all s and m, if $X_{s+1} \upharpoonright m \neq X_s \upharpoonright m$ then $C_{s+1} \upharpoonright (m-1) \neq C_s \upharpoonright (m-1)$.*

Proof. By examining the construction. If $\delta_s(k) \in X_{s+1}$ then there is some a in either $G_s(k)$ or $G_s(k-1)$ enumerated into P_{s+1}, and $x < \gamma_s(a) < \delta_s(k)$ enumerated into C_{s+1} (as $\gamma_s(\max G_s(k)) < \delta_s(k)$). □

For $e < y < s$ let $M_{e,s}(y)$ be the set of $n \in T_e^X(e, y)\,[s]$, enumerated with X_s-use $v < \delta_s(y+1)$. Note that $|M_{e,s}(y)| < \langle e, y\rangle$.

Lemma 3.4. *Let $e < y$, and let $s_0 < s_1$ be two stages s at which we enumerate $\max G_s(y)$ into P_{s+1} on behalf of e. Suppose that between stages s_0 and s_1, no new agitators are added to $G(y)$. Then $M_{e,s_0}(y) \subsetneq M_{e,s_1}(y)$.*

Proof. The assumption that we do not add agitators to $G(y)$ means that at no stage $s \in [s_0, s_1]$ do we enumerate $\delta_s(y)$, or any smaller marker, into X_{s+1}. It follows that for all $s \leqslant t$ in the interval $[s_0, s_1]$, $X_s \upharpoonright \delta_s(y+1) = X_t \upharpoonright \delta_t(y+1)$. Therefore $M_{e,s_0}(y) \subseteq M_{e,s_1}(y)$.

At stage s_0, we have $\delta(y) \leqslant \psi_e(y) < \delta(y+1) \leqslant u_e(y)\,[s_0]$, so $f_{s_0}(e,y) \notin M_{e,s_0}(y)$. Let x be the least number in $(A_{s_0+1} \setminus A_{s_0}) \cup (C_{s_0+1} \setminus C_{s_0})$. There are two options.

If $x \notin C_{s_0+1}$, then $\delta_{s_0+1}(y+1) > u_{e,s_0}(y)$, while $X_{s_0+1} = X_{s_0}$. The argument above applied to $s = s_0+1$ and $t = s_1$ shows that $f_{s_0}(e,y) \in M_{e,s_1}(y)$, yielding the desired inequality.

The case $x \in C_{s_0+1}$ is slightly more complicated. In this case we do enumerate $\delta_{s_0}(y+1)$ into X_{s_0+1}, making the enumeration of $f_{s_0}(e,y)$ into $T_e^X(y)[s_0]$ void. However, $x < \gamma(\max G(y)) < \delta(y) \leqslant \psi_e(y)\ [s_0]$, that is, C changes below $\psi_{e,s_0}(y)$ between stages s_0 and s_0+1. On the other hand, $\Psi_e(C,y)\!\downarrow\ [s_1]$; let $t \leqslant s_1$ at which we define this computation. At stage t we have $u_{e,t}(y)\!\downarrow$, i.e., $f(e,y) \in T_e^X(e,y)\,[t]$. Either $f_t(e,y) = f_{s_0}(e,y)$, or $f_t(e,y)$ was redefined after stage s_0, and is therefore large relative to stage s_0; in either case, $f_t(e,y) \notin M_{e,s_0}(y)$.

The computation $\Psi_e(C,y)$ defined at stage t persists up to stage s_1; by Lemma 3.2, as $u_{e,t}(y) \leqslant \psi_{e,t}(y)$, the enumeration of $f_t(e,y)$ into $T_e^X(e,y)[t]$ is preserved until stage s_1 as well, with use $u = u_{e,t}(y)$. At stage s_1 we have $\psi_{e,t}(y) = \psi_{e,s_1}(y) < u_{e,s_1}(y)$, which means that $f_{s_1}(e,y) \neq f_t(e,y)$. At some stage $r \in [t, s_1)$ we redefine $f_{r+1}(e,y)$ to be large, and this happens because $\psi_{e,r}(y) < \delta_{r+1}(y+1)$. Since $\psi_{e,r}(y) = \psi_{e,t}(y)$, we have $u < \delta_{r+1}(y+1) \leqslant \delta_{s_1}(y+1)$, thus $f_t(e,y) \in M_{e,s_1}(y)$, again giving the inequality. □

Corollary 3.5. *For all s and all $y < s$, $G_s(y)$ is nonempty.*

Proof. Fix y. For $s > y$ let $t_s(y)$ be the greatest $t \leqslant s$ such that $|G_t(y)| = N_y$. Note that for all $r \in [t_s(y), s)$, either $G_r(y) = G_{r+1}(y)$ or $G_{r+1}(y) = G_r(y) \setminus \{\max G_r(y)\}$, and this enumeration is made on behalf of some $e < y$ (and in particular, it is not the case that $y \in \emptyset'_{r+1} \setminus \emptyset'_r$). For $e < y$ let $k_s(e,y)$ be the number of stages $r \in [t_s(y), s)$ at which $\max G_r(y)$ was enumerated into P_{r+1} on behalf of e. Then

$$|G_s(y)| = N_y - \sum_{e<y} k_s(e,y).$$

Lemma 3.4 implies that $k_s(e,y) \leqslant \langle e,y\rangle$ for all $e < y$, and $N_y = \sum_{e<y} \langle e,y\rangle + 1$. □

Lemma 3.6. *For every y, $G_s(y)$ stabilizes, that is, for some s, for all $t > s$, $G_t(y) = G_s(y)$.*

So after stage s, no agitator from $G_t(y)$ is every enumerated into P, no agitator is cancelled, and no new agitators are appointed to $G_t(y)$. We call the stabilised set $G(y)$.

Proof. By induciton on y. Suppose that after stage s_0, no changes happen to $G(z)$ for all $z < y$. For every $a \in \bigcup_{z<y} G(z)$, both A and C eventually stabilise below $\gamma(a)$, after which $\gamma_s(a)$ is stable. Suppose that after stage s_1, for no $a \in \bigcup_{z<y} G(z)$ do we have an A- or a C-change below $\gamma_s(a) = \gamma(a)$. Also suppose that $\emptyset'_{s_1}(y) = \emptyset'(y)$.

Now for all $s \geqslant s_1$ write $G_s(y) = \{a_{0,s}, a_{1,s}, \ldots, a_{m_s,s}\}$, in increasing order. By induction on k we show that either $a_k = \lim_s a_{k,s}$ exists (and so is never cancelled, or enumerated into P), or $m_s < k$ for almost all s. Let $k \in [0, N_y)$, and suppose

that for all $k' < k$, $a_{k'} = \lim_s a_{k',s}$ exists; say these have stabilised by some stage s, and further A and C have stablised below $\gamma_s(a_{k'}) = \gamma(a_{k'})$ for all $k' < k$. There are several possibilities. Either $a_{k,s}$ is undefined ($m_s = k-1$). In that case no new followers will ever be appointed, so $G_t(y) = G_s(y)$ for all $t > s$. Suppose that $a_{k,s}$ is defined. It can never be cancelled. Either there is some stage $r \geqslant s$ at which $a_{k,s} = a_{k,r}$ is enumerated into P_{r+1}; in that case $G_{r+1}(y) = \{a_0, \ldots, a_{k-1}\}$, and for all $t > r$, $G_t(y) = G_{r+1}(y)$. Otherwise, for all $t > s$, $a_{k,t} = a_{k,s}$. □

Lemma 3.7. *For all y, $\delta_s(y)$ stabilises; $\emptyset' \leqslant_{\mathrm{T}} A \oplus X$.*

Proof. That $\delta_s(y)$ stabilises follows immediately from Lemma 3.6, since after $G(y)$ stabilises, we never increase $\delta_s(y)$. Also, we have ensured that $\delta_s(y)$ can only change after changes to either A or X below $\delta_s(y)+1$, showing that the final value $\delta(y)$ is computable from $A \oplus X$. Finally, if y enters $\emptyset'$ at stage s, then we ensure that either $\delta_s(y) \in X_{s+1}$, or there is an A-change below $\delta_s(y)$. □

Lemma 3.8. *$f = \lim_s f_s$ exists.*

Proof. Fix a pair (e, y) with $e < y$; we show that $f_s(e, y)$ stablises. Suppose that $\delta_s(y+1)$ is stable after stage s_0 (Lemma 3.7). Suppose that at stage $s > s_0$ we redefine $f_{s+1}(e, y)$. Then for all $t > s$, if $u_{e,t}(y)$ is defined then (by convention) it is at least $f_t(e, y)$ which is greater than $\delta(y+1)$, meaning that we will not redefine f after stage s. □

Lemma 3.9. *Every requirement $\mathcal{R}_e$ is satisfied.*

Proof. By Lemma 3.6, the functional Ψ_e is reset only finitely many times; we refer here to the last version. We assume, for a contradiction, that for all $y > e$, $f(e, y) \in T_e^X(e, y)$; we will show that for all $y > e$, $\Psi_e(C, y)\downarrow = \emptyset'(y)$, contradicting the incompleteness of C.

First we show that for all $y > e$, $\Psi_e(C, y)\downarrow$. Since $f(e, y) \in T_e^X(e, y)$, the use $u_e(y)$ eventually stabilises; also $\delta(y+1)$ eventually stabilises, which means that eventually $\psi_e(y)$ is constant at all stages at which we attempt to define $\psi_e(y)$. By induction, eventually, $\Psi_e(C, y')\downarrow$ (and is C-correct) for all $y' \in (e, y)$; so eventually, whenever $\Psi_e(C, y)\uparrow$ we will make an attempt to redefine it; and we will eventually successfully define a C-correct computation.

Toward showing that $\Psi_e(C, y) = \emptyset'(y)$, we prove that for all $y > e$, $\psi_e(y) \geqslant \delta(y+1)$. We prove this by induction on y. For $y > e$, let t_y be the stage at which the C-correct computation $\Psi_e(C, y)$ is first defined.

First, we tackle $y = e+1$: here we note that t_{e+1} is later than the last stage at which Ψ_e is restarted, which is the last stage at which we increase $\delta(e+1)$. At stage t_{e+1} we define $\psi_e(y) \geqslant \delta_{t_y}(e+1)$, which equals $\delta(e+1)$.

Now, let $y > e+1$ and suppose that for all $z \in (e, y)$, $\psi_e(z) \geqslant \delta(z+1)$. At stage t_y we define $\psi_e(y) \geqslant \psi_{e,t_y}(y-1)$; since C does not change below $\psi_e(y)$ after stage t_y, it follows that $\psi_{e,t_y}(y-1) = \psi_e(y-1)$. And so $\psi_e(y) \geqslant \psi_e(y-1) \geqslant \delta(y)$.

Suppose, for a contradiction, that $\psi_e(y) < \delta(y+1)$. If $s > t_y$ is late enough, and $u_{e,s}(y) < \delta_s(y+1)$, then we redefine $f_{s+1}(e, y)$ to be large, whence $u_e(y) > \delta_s(y+1)$. It follows that $u_e(y) \geqslant \delta(y+1)$. To sum up, we have

$$\delta(y) \leqslant \psi_e(y) < \delta(y+1) \leqslant u_e(y);$$

so at any late enough stage, we will enumerate $\max G_s(y)$ into P_{s+1}, which contradicts the stabilisation of $G(y)$. Hence $\psi_e(y) \geqslant \delta(y+1)$ as desired.

This completes the proof: suppose, for a contradiction, that y enters $\emptyset'$ at stage $r > t_y$. At stage r we redefine $\delta_{r+1}(y)$ (and so $\delta_{r+1}(y+1)$) to be large, larger than $\psi_e(y)$ — impossible.[3] □

4. Witnesses for low-cuppability

In this section we prove:

Theorem 4.1. *Suppose* $\mathbf{a}$ *is low-cuppable and* $\mathbf{c}$ *is a cupping partner of* $\mathbf{a}$*. Then there is a low* $\mathbf{b} \leqslant \mathbf{c}$ *which is also a cupping partner of* $\mathbf{a}$*.*

Actually, the constructed B is wtt-reducible to C. As mentioned above, this answers a question from [DGMW08]:

Corollary 4.2. *If* $\mathbf{a}$ *is low-cuppable, then it has a cupping partner which is both low and array computable.*

Proof. Let $\mathbf{a}$ be low-cuppable. By [DGMW08], there is an array computable degree $\mathbf{c}$ which joins $\mathbf{a}$ to $\mathbf{0}'$. Apply Theorem 4.1 to $\mathbf{a}$ and $\mathbf{c}$. □

4.1. **Requirements and notations.** As mentioned in the introduction, the low-cuppable c.e. degrees coincide with the promptly simple degrees [ASJSS84]. Let A be a given promptly simple set, and let C be a c.e. cupping partner of A. We will construct a c.e. set B, and an auxiliary c.e. set P. We will ensure that B is low, that $B \leqslant_{\mathrm{T}} C$, and that $\emptyset' \leqslant_{\mathrm{T}} A \oplus B$. By the recursion theorem, we are given a functional Φ such that $P = \Phi(A, C)$.

To ensure the lowness of B, we will try to preserve the computation $J^B(e)$ each time it converges, where $J^B(e)$ is a universal partial B-computable function. The uses of the functionals Φ and J are denoted by φ and j respectively.

4.2. **Coding and Permitting.** To ensure that $\emptyset' \leqslant_{\mathrm{T}} A \oplus B$ we use movable markers $\gamma(n)$ as in the constructions above. To define $\gamma(e)$, we first choose an agitator q_e, and then obtain a use $\varphi(q_e)$ for a computation $\Phi(A, C, q_e) = 0$. We define $\gamma(e) \geqslant \varphi(q_e)$. When either A or C changes below $\varphi(q_e)$ we will want to move $\gamma(e)$; when A changes we can do this with no penalty, but when C changes, we will enumerate $\gamma(e)$ into B. We then update $\gamma(e)$ to be greater than the new use $\varphi(q_e)$.

When e enters $\emptyset'$, we need a change in either A or B below $\gamma(e) + 1$. In this case we enumerate q_e into P, and thus force a change in either A or C below $\varphi(q_e)$. Again, if the change is in C rather than in A, we need to enumerate $\gamma(e)$ into B.

As these are the only enumerations into B, as in the previous construction, we see that every enumeration of some $\gamma(e)$ into B is accompanied by a change in C below $\varphi(q_e) \leqslant \gamma(e)$, yielding $B \leqslant_{\mathrm{T}} C$, in fact, with identity use.

Note that we do not require the prompt simplicity of A in the coding process.

4.3. **Lowness of B.** We now consider how to ensure that B is low. We need to ensure that for each e, if $J^B(e)\!\downarrow$ infinitely often, then $J^B(e)\!\downarrow$.

$\mathcal{N}_e$: $\exists^\infty s\, (J^B(e)\!\downarrow [s]) \;\Rightarrow\; J^B(e)\!\downarrow$.

[3] Note that in the end we in fact get $\psi_e(y) < \delta(y)$ rather than just $\psi_e(y) < \delta(y+1)$ for the contradiction; but to prove that $\psi_e(y) \geqslant \delta(y)$ inductively, we actually need to prove $\psi_e(y) \geqslant \delta(y+1)$.

To this end, whenever $J^B(e)[s]$ converges (with use $j_s(e)$), we will try to lift $\gamma(e)$ above $j_s(e)$ by forcing an A-change below $\gamma_s(e)$. We will use the prompt simplicity of A here, to ensure that a wanted A-change will appear eventually.

We will enumerate an auxiliary array of c.e. sets U_e, and by a slowdown lemma and the recursion theorem, we may assume that we have a computable function p such that for the original given enumeration of A, if we put numbers $x_0 < x_1 < \cdots$ into U_e at stages $s_0 < s_1 < \cdots$ respectively, then A has to promptly permit one of them; namely there is some i such that $A_{s_i} \upharpoonright x_i \neq A_{p(s_i)} \upharpoonright x_i$.

When $J^B(e)\downarrow$ $[s]$, in order to lift $\gamma_s(e)$ above the use $j_s(e)$, we enumerate $\gamma_s(e)$ into U_e, and hope for the desired A-change, which we can check by observing $A_{p(s)}$. If the required change has not happened, we try again by enumerating q_e into P, forcing a change in either A or C below $\varphi(q_e) \leqslant \gamma(e)$. If we fail again (the change was in C, not A), then we enumerate $\gamma(e)$ into B, killing the convergence of $J^B(e)$. We then need to pick a new version of q_e. The prompt simplicity of A will ensure that after finitely many attempts, we will succeed in lifting $\gamma(e)$ beyond $j_s(e)$, helping us protect the convergence of $J^B(e)$.

Note that we cannot make B superlow because we do not have control over when we receive A-permission to lift $\gamma(e)$.

The speed-up. In the previous construction, we have sped-up the enumerations of A and C to get long agreement between $\Phi(A, C)$ and P at every stage. In this construction though, the function p witnessing prompt simplicity with respect to the array $\langle U_e \rangle$ is with respect to the original given enumeration $\langle A_s \rangle$ of A. We therefore incorporate p into a speed-up. Let us be precise. Suppose that $\langle \hat{A}_s \rangle$, $\langle \hat{C}_s \rangle$ are given enumerations of A and C (and $\hat{\Phi}_s$ is a given enumeration of the functional Φ), and that p witnesses prompt simplicity of A with respect to the enumeration $\langle \hat{A}_s \rangle$ and an array $\langle \hat{U}_{e,s} \rangle$ that we define during the construction: for every e, if $\hat{U}_e$ is infinite, then there are infinitely many s and x such that x enters $\hat{U}_e$ at stage s and $\hat{A}_{p(s)} \upharpoonright x \neq \hat{A}_s \upharpoonright x$. We also define, during the construction, an enumeration $\langle \hat{P}_s \rangle$ of the set P we are building.

We then define an increasing sequence $t(0) < t(1) < \dots$ of stages as follows:

- $t(0) = 0$;
- $t(s+1)$ is the least $t > p(t(s))$ such that the agreement between $\hat{\Phi}_t(\hat{A}_t, \hat{C}_t)$ and $\hat{P}_t$ is well beyond any number previously seen in the construction.

Then, we define $A_s = \hat{A}_{t(s)}$, $C_s = \hat{C}_{t(s)}$, $P_s = \hat{P}_{t(s)}$, $\Phi_s = \hat{\Phi}_{t(s)}$ and $U_{e,s} = \hat{U}_{e,t(s)}$. Then $\langle A_s \rangle$, $\langle C_s \rangle$ and $\langle P_s \rangle$ are computable enumerations of A, C and P; for every s, the agreement between $\Phi_s(A_s, C_s)$ and P_s is large relative to what happened so far in the construction; and for all e, if U_e is infinite, then there are infinitely many s such that for some x, enumerated into U_e at s, we have $A_{s+1} \upharpoonright x \neq A_s \upharpoonright x$. This is because the construction really uses the hatted versions and happens on the stages $t(s)$ (so elements enter $\hat{U}_e$ only at stages $t(s)$); but we rephrase it using the un-hatted notation, to keep things tidy, if less precise.

4.4. **The construction.** We are ready to give a full construction. By our speed-up, we may assume that at the beginning of each stage s, $\Phi(A, C)[s]$ and P_s agree on all the agitators $q_{e,s}$ for $e < s$, and that $\gamma_s(e) = \varphi_s(q_{e,s})$ for all $e < s$.

Let e be least such that either:

(1) $e \in \emptyset'_{s+1} \setminus \emptyset'_s$; or

(2) $J^B(e)\downarrow$ $[s]$ with use $j_s(e) > \gamma_s(e)$.

Our actions for e are, depending on the case:

(1) Enumerate $q_{e,s}$ into P_{s+1}.
(2) Enumerate $\gamma_s(e)$ into $U_{e,s}$. Check if $A_{s+1} \upharpoonright \gamma_s(e) \neq A_s \upharpoonright \gamma_s(e)$. If not, enumerate $q_{e,s}$ into P_{s+1}.

If we enumerated $q_{e,s}$ into P_{s+1}, then we choose $q_{e,s+1}$ to be large (and so $q_{e',s}$ to be large for all $e' > e$ as well). Next, let d be the least such that either $A_{s+1} \upharpoonright \gamma_s(d) \neq A_s \upharpoonright \gamma_s(d)$ or $C_{s+1} \upharpoonright \gamma_s(d) \neq C_s \upharpoonright \gamma_s(d)$ (or $d = s$ if there is no such d). Note that if we enumerated $q_{e,s}$ into P_{s+1} then $d \leqslant e$. If $A_{s+1} \upharpoonright \gamma_s(d) = A_s \upharpoonright \gamma_s(d)$ then we enumerate $\gamma_s(d)$ into B_{s+1}. In either case we redefine $\gamma_{s+1}(d') = \varphi_{s+1}(q_{d',s+1})$ for all $d' \geqslant d$.

4.5. **Verification.** The verification is straightforward. Note that if $\gamma_{s+1}(e) \neq \gamma_s(e)$ then either A or B changes below $\gamma_s(e) + 1$; that such a change occurs if e enters $\emptyset'$ at stage s; and that $B \leqslant_{\mathrm{T}} C$ as promised.

By induction on e we show that:

(i) U_e is finite; and
(ii) $\mathcal{N}_e$ is satisfied.

Note that (i) implies that $\langle q_{e,s} \rangle$ stabilises, as its value changes only when we enumerate numbers into U_e, or once when e enters $\emptyset'$. In turn, because $\Phi(A, C)$ is total, this implies that $\langle \gamma_s(e) \rangle$ stabilises as well; it will follow that $\emptyset' \leqslant_{\mathrm{T}} A \oplus B$. For a proof, suppose that these all hold for all $d < e$, and say that $q_{d,s}$ and $\gamma_s(d)$ have all stabilised by some stage s_0; also let s_0 be sufficiently late so that $\emptyset'_{s_0} \upharpoonright e + 1$ is correct.

If U_e is infinite, then there is some stage $s > s_0$ and some x enumerated into U_e at stage s such that $A_{s+1} \upharpoonright x \neq A_s \upharpoonright x$. Since $x = \varphi_s(q_{e,s})$, we have $B_{s+1} = B_s$ and we redefine $\gamma_s(e)$ to be larger than $j_s(e)$. However, by assumption, no markers $\gamma_s(d)$ for $d < e$ enter B after stage s_0; so in this case the computation $J^B(e)\,[s]$ is B-correct, and no further enumerations into U_e are ever made. Thus U_e is finite. This, in turn, implies that $\mathcal{N}_e$ is satisfied: from some stage on, either $J^B(e)\uparrow$ $[s]$, or $J^B(e)\downarrow$ with use $j_s(e) \leqslant \gamma_s(e)$; as $\langle \gamma_s(e) \rangle$ stabilises, if this happens infinitely often, then $J^B(e)\downarrow$.

5. low_2-cuppable degrees are AC-cuppable

In this section we show that the classes of Low_2-cuppable degrees and AC-cuppable degrees coincide.

Theorem 5.1. *Suppose $\emptyset' \leqslant_{\mathrm{T}} A \oplus C$ where C is low_2. Then there is an array-computable set B such that $\emptyset' \leqslant_{\mathrm{T}} A \oplus B$.*

5.1. **The method of exploiting low_2-ness.** The usual way of utilizing the low_2-ness of C is to use it to enable us to successfully guess whether any reduction $\Gamma(C)$ is total. This is due to the fact that the totality of $\Gamma(C)$ can be described by a Δ^0_3-procedure for each Turing functional Γ. One can set up the construction so that the answers to such questions are naturally represented on the construction tree, and such that the leftmost outcome gives the true answer about the totality of each $\Gamma(C)$ being tested in the construction. One can then use this information along the true path of the construction to reduce certain injuries, in a similar way

to how Robinson's guessing technique converts infinitary actions into finite activity for a given low c.e. set.

Recall that another characterization of a low_2 c.e. set C is the existence of some $f \leqslant_{\mathrm{T}} \emptyset'$ which dominates every C-computable function. Using this alternate formulation of low_2, we can exploit a given low_2 set C in a game-theoretic way. A construction involving C can be viewed as a game between ourselves and the opponent. The opponent controls the function f and the low_2 set C and provides computable approximations $\langle f_s \rangle$ and $\langle C_s \rangle$ for both. At the same time we can build a reduction $\Delta(C)$ (and even allow a different Δ to be built at each requirement and Δ can be specifically tailored for that requirement). We will ensure that we make $\Delta(C)$ total unless we can have an easy win at the requirement. Each axiom $\langle \sigma, x, y \rangle$ we enumerate into Δ will represent a *challenge* to the opponent, and can be viewed as a request for the assurance that $\sigma \prec C$. If C is low (instead of merely low_2) the opponent has to respond promptly to each of our challenges. Specifically, the opponent must either demonstrate that $\sigma \not\prec C$ by changing C_s below $|\sigma|$, or he has to provide the certification that $\sigma \prec C$. Our strategy can utilize this by freezing the construction until one of the two outcomes occurs. Since C might be noncomputable, the opponent might first certify that $\sigma \prec C$, and then later change C below $|\sigma|$. However, as C is low, this can only happen finitely often at each requirement.

Now however C is merely low_2 and we must set up a similar game to play with the opponent. We can still issue challenges to the opponent in the form of axioms $\langle \sigma, x, y \rangle$ as discussed above. The opponent must now respond with one of the following:

(i) demonstrate that $\sigma \not\prec C$ by changing $C_s \upharpoonright |\sigma|$,
(ii) certify that $\sigma \prec C$ by changing the approximation of $f_s(x)$ to become larger than y, or
(iii) do nothing at all.

The third outcome is now available to the opponent, and unlike in the low case, we cannot freeze the construction and wait for (i) or (ii). We call a challenge *pending* if neither (i) nor (ii) happens after we issue a challenge. We must proceed with the construction and make sure that all of our future actions do not undo or destroy the strategy associated with a pending challenge. In particular we must continue making $\Delta(C)$ total. Only then can we be assured that only finitely many challenges remain pending forever. Even on the cofinite set of challenges where the opponent does respond, there will potentially be a huge delay between the stage where we issue the challenge and the stage where he responds with (i) or (ii). In the next section we will discuss how to set up the construction correctly so that we can benefit from the low_2-ness of C.

5.2. **The setup and the modular approach.** Recall that by Ishmukhametov [Ish99], the array computable c.e. sets are exactly the same as the c.e. traceable c.e. sets. Thus, we want to build a c.e. traceable cupping partner B for A and satisfy the requirements

$$\mathcal{R}_\Psi \quad : \quad \text{if } \Psi(B) \text{ is total then it has a c.e. trace } T \text{ with } |T(x)| \leqslant x \text{ for all } x.$$

The choice of the bound $|T(x)| \leqslant x$ is unimportant; any fixed non-decreasing, unbounded computable function (an *order function* would do). The reduction of $\emptyset'$ to $A \oplus B$ is built by movable markers. We maintain a set of markers $\varphi(0), \varphi(1), \cdots$

which should be viewed as the use of a reduction $\Phi(A,B) = \emptyset'$. In the actual construction, we will need two seperate sets of markers $\{\varphi_A(x)\}$ and $\{\varphi_B(x)\}$ for the A and B-use; however for simplicity we will assume that they are the same for the time being (this is not a real restriction since we can always take the larger of the two). Coding always takes place in B; i.e. whenever some x enters $\emptyset'$, we will enumerate $\varphi(x)$ into B. This obviously conflicts with the strategy $\mathcal{R}_\Psi$, which want to restrain B each time after we had committed some current value $\Psi(B,x)\,[s]$ into the trace $T(x)$. Each $\mathcal{R}_\Psi$ obviously cannot directly restrain the global requirement from performing coding, so it will have to ensure that B does not change too often by disengaging dangerous φ-markers from below the use of $\Psi(B)$-computations before committing any value into the c.e. trace $T(x)$. The meaning of "disengaging a φ_B-marker $\varphi_B(x)$ from below the use of a computation $\Psi(B;z)$" is to move the marker $\varphi_B(x)$ to be larger than the use $\psi(z)$ of the computation $\Psi(B;z)$ without damaging the computation. Since $\varphi_B(x)$ can only be moved if A changes below $\varphi_A(x)$ or B changes below $\varphi_B(x)$ (see section 4.2), we cannot do this at will as we do not control the set A. Therefore in order to disengage a φ_B-marker, we will need to try and stimulate an A-change below the corresponding φ_A-marker.

The construction takes place on a tree of strategies. The design of the priority tree and the strategy for the requirements will have to take into account of pending challenges. Each level is devoted to one requirement $\mathcal{R}_\Psi$. For a node α we write Ψ_α for the functional Ψ. Each node α divides its main strategy into infinitely many substrategies, which are run separately by individual *α-modules* $M_0^\alpha, M_1^\alpha, \ldots$. We divide $\mathbb{N}$ into infinitely many partitions, $\mathtt{zone}_0^\alpha, \mathtt{zone}_1^\alpha, \ldots$, and the module M_k^α monitors the computations $\Psi_\alpha(B,x)$ for each $x \in \mathtt{zone}_k^\alpha$.

At the beginning of the construction, we set $\mathtt{zone}_x^\alpha = \{x\}$ for all x. The partitions will be rearranged as the construction proceeds. If x is currently in $\mathtt{zone}_k^\alpha$, the module will monitor the computation $\Psi_\alpha(B,x)[s]$ and wait for it to converge. The module will be allowed to trace the value $\Psi_\alpha(B,x)[s]$ into $T(x)$ only if it manages to disengage $\varphi(k)$ from below the use of $\Psi_\alpha(B,x)[s]$ (recall that this first requires an A-change).

If the computation $\Psi_\alpha(B,x)[s]$ is later on injured (either due to higher priority requirements acting or the global coding actions), we will lower the tolerance of x and transfer control of x to a higher priority α-module. In particular, we will now set $\mathtt{zone}_{j-1}^\alpha[s+1] = \mathtt{zone}_j^\alpha[s]$ for every $j \geqslant k$. Thus we say that x previously had *tolerance number* k, and after the injury to $\Psi_\alpha(B,x)[s]$ it will have a reduced tolerance number of $k-1$ and will now be handled by the module M_{k-1}^α. When x reaches a tolerance number of 0 it will only be traced if M_0^α manages to disengage $\varphi(0)$, so no injury can possibly happen and $\Psi_\alpha(B,x)$ will be preserved forever.

This allows us to limit the injury to each $\Psi_\alpha(B,x)[s]$ and help ensure that $T(x)$ will have size at most x. There are two potential problems here and we will describe how they are resolved later. Firstly we have to ensure that each $\varphi(k)$ is moved only finitely often. Secondly, we have to ensure that for almost all x, the module eventually responsible for x can manage to disengage φ and trace the correct value $\Psi(B,x)$.

Atomic strategy of a single M_k^α. In the rest of the proof the super-script α is dropped from our notation whenever the context is clear. We first describe how a single module M_k^α acts to disengage $\varphi(k)$ from below the use $\psi(x)$ of every $x \in \mathtt{zone}_k^\alpha$. The goal of M_k^α is to force an $A \upharpoonright \varphi(k)$-change whenever $\Psi(B,x)\!\downarrow$ so

that it can allow the marker $\varphi(k)$ to be lifted while at the same time preserving the $\Psi(B, x)$-computation.

As in previous constructions, we enumerate a private version P of $\emptyset'$. We are given a functional Γ such that $\Gamma(A, C) = P$. Numbers that potentially go into P are called *agitators*. The purpose of agitators is to enumerate them into P at appropriate times in order to stimulate changes in A that are required for modules to achieve the disengagement of φ-markers.

The agitator for M_k^α is denoted by a_k^α. The module waits for $\Gamma(A, C, a_k^\alpha)\downarrow [s]$. We say that M_k^α is *set correctly* if $\gamma(a_k^\alpha) < \varphi(k)$; in order for the agitator to be able to help in disengaging $\varphi(k)$, it must first be set correctly. If M_k^α is not set correctly then we simply enumerate $\varphi(k)$ into B and move the $\varphi(k)$ marker above $\gamma(a_k^\alpha)$. Since C and Γ are not under our control, we may have to do this repeatedly to keep M_k^α in the state of being set correctly. When M_k^α is set correctly and we see $\Psi(B, x)\downarrow$, we enumerate a challenge $\Delta(C, p)$ with C-use $\sigma = C_s \upharpoonright \gamma(a_k^\alpha)$. As discussed above (in Section 5.1), the opponent has to (i) demonstrate that $\sigma \not\prec C$, (ii) certify that $\sigma \prec C$, or (iii) do nothing for this challenge.

If (i) happens then we enumerate $\varphi(k)$ into B to kill the current $\Psi(B, x)$-computation, set M_k^α correctly and wait for it to converge again. We repeat with a new challenge.

If (ii) happens we enumerate a_k^α into P and wait for an A or a C-change below $\gamma(a_k^\alpha)$. If an A-change is given, then we can lift the marker $\varphi(k)$ and successfully disengage $\varphi(k)$. We can then trace the current value of $\Psi(B, x)$ in $T(x)$. If a C-change is given then the opponent would lose one change available to him, and since we have not yet committed the current value of $\Psi(B, x)$ into the trace $T(x)$, we can simply set M_k^α correctly again and repeat with a new challenge.

If neither (i) nor (ii) happens then the module M_k^α cannot proceed further, since it requires the disengagement of $\varphi(k)$ before it can trace any $\Psi(B; x)$ computation. In this case, M_k^α is pending and does no further work until it sees (i) or (ii) happen for the pending challenge.

This describes the actions of a single module M_k^α in isolation. We make two straightforward but important observations. First, note that as each γ-use is finite, (i) cannot happen infinitely often on the same agitator. Second, note that each module may wait in the pending state for an arbitrary number of stages. However, there are only finitely many k such that M_k^α is forever pending; otherwise $\Delta(C)$ will be total and escapes f.

Now we will discuss how two different modules interact. We first discuss in Section 5.3 how modules belonging to different nodes interact. In Section 5.4 we discuss how to coordinate different modules belonging to the same node.

5.3. **Coordination between different nodes.** Since each node α may potentially make infinitely many enumerations into B (though only finitely often for each module), we have to equip α with two outcomes, ∞ to the left of f. As usual a node $\varepsilon \succcurlyeq \alpha\hat{}f$ is not allowed to injure α, so at each stage when ε is visited, only the modules M_j^ε for $j >$ the previous α-expansionary stage are allowed to act. By the usual convention on picking followers, these modules will only act with numbers that are far larger than anything seen by α so far.

However, a node $\beta \succcurlyeq \alpha\hat{}\infty$ will wait for enough α-modules to be successful. In particular, a module M_k^β will wait for all of $M_0^\alpha, \ldots, M_k^\alpha$ to successfully trace every computation they are looking after before β begins to act for M_k^β. By coordinating

the actions of α and β in this way, we ensure that α never injures any β-module, although it might be the case for instance, that M_k^β acts and injures $M_{k+1}^\alpha, M_{k+2}^\alpha, \dots$ which already have traced computations. This means that a module M_k^α can only be injured by finitely many modules belonging to a node extending $\alpha^\frown\infty$ (unless of course α itself is initialized).

As per usual we arrange for nodes of each length k to agree never to move $\varphi(k)$. In this way, we can ensure that each $\varphi(k)$ is only moved finitely often (since there are only finitely many nodes on the construction tree that can move it). Also this ensures that for instance, injury to M_k^β will eventually cease, since the only modules which can injure it are of the form M_j^σ for some $j < k$ and $\sigma \succcurlyeq \beta^\frown\infty$. Hence every module belonging to a node along the true path is injured only finitely often.

5.4. **Coordinating different modules belonging to the same node.** Due to the fact that modules can be pending forever, a node α cannot run the actions of its modules sequentially and start M_{k+1}^α only after M_k^α has completed all of its actions. Rather, α will need to go through its modules in two phases. The first phase is done sequentially, where we go through $M_0^\alpha, M_1^\alpha, \dots$ in order and extend $\Delta_\alpha(C)$ at each module; this step only requires $\Psi(B)$ to be total and therefore can be done by the modules of α in order.

After a module M_k^α has completed its first phase, it is pending and is waiting for the opponent to respond to the challenge. We cannot go through all the modules without the opponent responding to cofinitely many of the challenges; if infinitely many modules are pending forever we will make sure that $\Delta_\alpha(C)$ is total which will then contradict the fact that $\Delta_\alpha(C)$ is dominated by f. On the other hand if the opponent eventually responds to the challenge on some pending M_k^α, we will start the second phase of M_k^α and try to force a change in $A \upharpoonright \varphi(k)$.

The opponent is of course not obliged to respond to our challenges in a sequential fashion, and only has to make sure that at the end of time, he responds to all but finitely many of α's challenges. This introduces additional injury on top of the global coding actions. We illustrate this with the following example.

Suppose that M_k^α has completed its first phase and is now pending. A module M_j^α for $j > k$ might successfully trace the computations that it is looking after while M_k^α is still pending. Suppose that later on we attend to M_k^α and enumerate $\varphi(k)$ into B because we received certification on an incorrect C-segment. This will injure all modules of lower priority than M_k^α. Consequently, all the computations previously in $\bigcup_{j>k} \mathtt{zone}_j^\alpha$ will have to lower their tolerance. In particular, $\mathtt{zone}_k^\alpha$ will receive new values of x.

We need to argue that $\mathtt{zone}_k^\alpha$ does not receive infinitely many new values in this way: Notice that M_k^α has not yet been successful, and we will wait for $\Psi(B, x)$ to converge for all the new $x \in \mathtt{zone}_k^\alpha$, before issuing a new challenge to the opponent. The above cannot happen infinitely often because otherwise infinitely many incorrect certifications are issued contradicting that $f \leqslant_T \emptyset'$.

We have just seen that changing B will cause $\mathtt{zone}_k^\alpha$ to increase; the above scenario doesn't pose a big problem since we can limit the increase in $\mathtt{zone}_k^\alpha$ by counting the number of incorrect certifications received at the module M_k^α. However, a more serious issue occurs if the B-change is made by another requirement in the construction. For instance, $\Psi_\alpha(B, x)$ (for some $x \in \mathtt{zone}_k^\alpha$) could converge with a very large use $\psi_\alpha(x)$, and after a challenge is issued by M_k^α, we might change B below $\psi_\alpha(x)$ when M_k^α is still pending. Now if $A \upharpoonright \gamma(a_k^\alpha)$ changes before $\Psi_\alpha(B, x)$

next converges, we would be unable to benefit from the A-change. The opponent could then certify that C was correct (on the old $\gamma(a_k^\alpha)$-use), and never change C below that. This is bad for M_k^α, for it would have lost all progress it had previously made on the test $\Delta(C,p)$, and p cannot be used in future challenges. We will be forced to pick a new value of p for M_k^α to use in future iterations of its strategy, but since $\Psi_\alpha(B,x)$ might converge with larger and larger use, M_k^α might end up losing every p it picks if we are not careful. Now of course locally, the requirement at α is satisfied as $\Psi_\alpha(B)$ is not total, however, the module M_k^α will end up moving the marker $\varphi(k)$ infinitely often, which we absolutely must avoid doing.

To deal with the above problem, we will do the following. We ensure that every time some computation in $\mathtt{zone}_k^\alpha$ is injured while M_k^α is pending, there must also a sufficiently small C-change so that $\Delta_\alpha(C,p)$ is made undefined and M_k^α can reuse the same test $\Delta_\alpha(C,p)$. To be more specific, we wait for $\Psi_\alpha(B,x)$ to converge for every $x \in \mathtt{zone}_k^\alpha$. We then challenge the opponent (via $\Delta_\alpha(C,p)$) to certify that $C \upharpoonright \gamma(a_p^\alpha)$ is correct, where p is largest such that $\varphi(p) < \psi_\alpha(x)$ for any $x \in \mathtt{zone}_k^\alpha$. Any action which can injure the computations in $\mathtt{zone}_k^\alpha$ will have to be either due to the global actions (which happens at most once for each $\varphi(p)$), or due to the actions of some M_j^β where $\beta \succ \alpha$ and $j \leqslant p$. Therefore, the only reason why a module M_j^β enumerates $\varphi(j)$ into B and destroys the computation, must be due to one of the following two reasons:

(i) M_j^β is no longer set correctly, and it enumerates $\varphi(j)$ into B in an attempt to set itself correctly again. In this case, there must have been some change in C which now makes $\Delta_\alpha(C,p)$ undefined, because of M_k^α-believability.

(ii) the opponent responded to some challenge put forward by M_j^β, where we would have enumerated a_j^β in response, and observed a resulting $C \upharpoonright \gamma(a_j^\beta)$-change. In this case $\Delta_\alpha(C,p)$ is now undefined as well.

In either of the two cases above, M_k^α can now reuse the test $\Delta_\alpha(C,p)$ on the same p, so that previous progress on this test location is not wasted. Even if this happens infinitely often, M_k^α will only use finitely many agitators and hence only move $\varphi(k)$ finitely often. In this way, the actions of a larger module M_j^α may affect M_k^α, but no real injury is inflicted on M_k^α. Any $\Psi_\alpha(B)$-computation which is true, will eventually become believable. For each k, we need to arrange for two outcomes - $k\infty$ to the left of kf.

Since α does not know what the nodes extending it are currently doing, it is the responsibility of $\beta_1 \succcurlyeq \alpha^\frown kf$ to ensure that it never damages a convergent $\Psi_\alpha(B,x)$-computation which is pending the opponent's response for every $x \in \mathtt{zone}_k^\alpha$. This outcome kf of α also encodes the fact that every challenge associated with M_k^α will eventually cease to be pending; by the assumption that f dominates $\Delta_\alpha(C)$, one of the outcomes of α must be true. Now since β below α has to act above the use of convergent $\Psi_\alpha(B,x)$-computation, and because of the possibility of $\psi_\alpha(x)$ going to ∞ for some $x \in \mathtt{zone}_k^\alpha$, we will also need to have the infinitary outcome $k\infty$. A node $\beta_0 \succcurlyeq \alpha^\frown k\infty$ is only visited if $\Delta_\alpha(C,p)$ is undefined, so M_k^α does not care what β_0 does during the $\alpha^\frown k\infty$-stages.

5.5. **Requirements and conventions.** As discussed above, we are given A and C, enumerate P, and are given Γ such that $\Gamma(A,C) = P$. We enumerate B. We are given a computable approximation $\langle f_s \rangle$ to a Δ_2^0 function f which dominates all C-computable functions.

The usual convention regarding stage numbers and notations applies. The use of functionals Γ and Ψ are denoted respectively by γ and ψ. We assume that for all x and s, if $\Psi(B,x)\downarrow\ [s]$ then $x < \psi_s(x) < s$, and similarly for Γ; we also assume that the uses are monotone. During the actual construction, there may be several actions taken one after another in a single stage s. It is sometimes convenient to break down a single stage into substages where a single action is taken at each substage. In this construction we will not bother with distinguishing between a stage and its substages; when we refer to a stage s we actually mean the instance within the stage s (or the substage of s) where the action is taken.

As indicated above, we introduce a minor difference with respect to the markers; we will have separate markers $\varphi_A(k)$ and $\varphi_B(k)$. If either A changes below $\varphi_A(k)$ or B changes below $\varphi_B(k)+1$ then we are allowed to lift either or both uses.

5.6. **The construction tree.** The construction takes place on an infinite branching tree. Nodes at level e are devoted to the requirement $\mathcal{R}_\Psi$ for the e^{th} functional $\Psi = \Psi_e$ in some effective list of all functionals. There are two groups of outcomes, and we will alternate between the two groups in the ordering: $1\infty <_L 1f <_L 2\infty <_L 2f <_L 3\infty <_L 3f <_L \dots$. The first group of outcomes $1\infty, 2\infty, \dots$ are called *infinite outcomes*, in order to distinguish these from the second group $1f, 2f, \dots$, which we will call the *finite outcomes*. The choice of these names have little to do with the frequency of actions and when the respective outcomes are played; in the actual construction the finite outcomes do have infinitary actions, however they are tagged as "finite" simply because certain associated Ψ_e^B-computations converge in the limit.

The ordering between outcomes extends lexicographically to an ordering of nodes. $\alpha <_L \beta$ means that α is strictly to the left of β. We write $\alpha \prec_\infty \beta$ if $\alpha\hat{}n\infty \preccurlyeq \beta$ for some n, and $\alpha \prec_f \beta$ if a similar situation holds with f instead of ∞. We write $\alpha \preccurlyeq_\infty \beta$ if $\alpha \prec_\infty \beta$ or $\alpha = \beta$, and similarly write $\alpha \preccurlyeq_f \beta$.

We say that α *is an* $\mathcal{R}_\Psi$*-node*, if α is assigned the requirement $\mathcal{R}_\Psi$, and write Ψ_α for Ψ. Each node α measures the totality of $\Psi_\alpha(B)$. As described previously, $\alpha\hat{}k\infty$ will be visited if M_k^α has its current test reset, i.e. $\Delta_\alpha(C,p)$ undefined, so that M_k^α does not care what happens in the region $[\alpha\hat{}k\infty]$ (defined as the set of all nodes $\beta \succcurlyeq \alpha\hat{}k\infty$). Whenever α plays this outcome, we initialise all nodes $\beta \succcurlyeq \alpha\hat{}o$ for any outcome o to the right of $k\infty$. Whenever M_k^α progresses in its atomic strategy, we will visit the outcome kf, and initialise all nodes extending an outcome $o >_L kf$. Any node $\beta \succcurlyeq \alpha\hat{}kf$ will coordinate its actions with α as described previously. That is, M_j^β will wait until all the modules $M_k^\alpha, M_{k+1}^\alpha, \dots, M_j^\alpha$ are successful, before M_j^β is allowed to act (this prevents α-modules from injuring β-modules). To ensure the true path of the construction exists, whenever a module M_k^α becomes successful, we will visit the outcome $k'f$ where k' is the least place $\leqslant k$ where a run of successful modules starts. For instance, using s to denote success, and w otherwise, we could have initially:

Module	M_0^α	M_1^α	M_2^α	M_3^α	M_4^α	M_5^α	M_6^α	M_7^α	M_8^α	$\cdots$
State	w	w	w	w	w	w	w	w	w	$\cdots$

Outcome played $= 0f$. Then we have:

Module	M_0^α	M_1^α	M_2^α	M_3^α	M_4^α	M_5^α	M_6^α	M_7^α	M_8^α	$\cdots$
State	w	w	s	w	w	w	w	w	w	$\cdots$

Outcome played $= 2f$.

Module	M_0^α	M_1^α	M_2^α	M_3^α	M_4^α	M_5^α	M_6^α	M_7^α	M_8^α	$\cdots$
State	w	w	s	w	w	s	w	w	w	$\cdots$

Outcome played $= 5f$.

Module	M_0^α	M_1^α	M_2^α	M_3^α	M_4^α	M_5^α	M_6^α	M_7^α	M_8^α	$\cdots$
State	w	w	s	w	w	s	w	w	s	$\cdots$

Outcome played $= 8f$.

Module	M_0^α	M_1^α	M_2^α	M_3^α	M_4^α	M_5^α	M_6^α	M_7^α	M_8^α	$\cdots$
State	w	w	s	w	w	s	s	w	s	$\cdots$

Outcome played $= 5f$.

Module	M_0^α	M_1^α	M_2^α	M_3^α	M_4^α	M_5^α	M_6^α	M_7^α	M_8^α	$\cdots$
State	w	w	s	w	s	s	s	w	s	$\cdots$

Outcome played $= 4f$.

Module	M_0^α	M_1^α	M_2^α	M_3^α	M_4^α	M_5^α	M_6^α	M_7^α	M_8^α	$\cdots$
State	w	w	s	s	s	s	s	w	s	$\cdots$

Outcome played $= 2f$.

Hence if $\Psi_\alpha(B)$ is total, then $\Delta_\alpha(C)$ is total and the opponent has to respond at all but finitely many of the modules. In between, infinitary outcomes may be played from time to time, as different Ψ_α-computations are injured, but the above will describe the situation in the limit. In the above example we will visit α-outcome $2f$ infinitely often, and so a node $\beta \succcurlyeq \alpha\hat{}2f$ can wait patiently to start any β-module. On the other hand if some infinite outcome $k\infty$ is played infinitely often then it must be that some $x \in \mathtt{zone}_k^\alpha$ is divergent and $\Delta_\alpha(C)$ is not total, so that the opponent doesn't have to respond at all. However this is an automatic win for us at α.

5.7. **Notation for the formal construction.** Each α builds a c.e. trace T^α, and ensures that for all x, $|T^\alpha(x)| \leqslant x$. For each α and x, we have a parameter t_x^α which keeps track of an upper bound for $x - |T_s^\alpha(x)|$. At the beginning (each time α is initialised) we start with $t_x^\alpha = x$. Every time a value $\Psi_\alpha(B, x)[s]$ is traced in $T^\alpha(x)$ and later $B \upharpoonright \psi_\alpha(x)$ is changed, we will decrease t_x^α by 1. When t_x^α reaches 1 (if ever), then any current value traced must be preserved forever. This ensures that $|T^\alpha(x)| \leqslant x$. During the construction we will sometimes say that we *lower* t_x^α. This simply means that we decrease the value of t_x^α by 1. When we lower t_x^α, we also lower t_y^α for every $y > x$ at the same time. The inverse of the parameter t_x^α is denoted by the parameter $\mathtt{zone}_k^\alpha$, which is the set of all numbers x such that $t_x^\alpha = k$. Each node α will be building a Turing functional Δ_α, which is used as a test. Following conventions, the use of currently applicable computations $\Delta_\alpha(C, x)$ is denoted by $\delta_\alpha(x)$.

Every x in the same zone has the same goals; they all want to "disengage" the same φ_B-markers from below their use $\psi_\alpha(x)$. That is, every x in $\mathtt{zone}_k^\alpha$ wants to ensure that $\psi_\alpha(x) < \varphi_B(k)$. The k^{th} module M_k^α will act on behalf of all the $x \in \mathtt{zone}_k^\alpha$, and will elect a number p_k^α called a *follower*. The idea is that M_k^α will be enumerating computations for $\Delta_\alpha(C, p_k^\alpha)$. It also appoints a number a_k^α called an *agitator*, and as the name suggests, will be used to force an $A \oplus C$-change.

We will introduce what we call global agitators, denoted by g_k. That is, g_k will be used to help in the coding of $\emptyset'(k)$. The primary aim of the global agitator is to ensure that if k enters $\emptyset'$ and we need to code, we will only change $B \upharpoonright \varphi_B(k)$ if we

have an accompanying $C \upharpoonright \gamma(g_k)$-change. Since we only need to use up g_k if coding occurs, we may choose and fix the values for the global agitators in advance.

Each module M_k^α is in a particular state at any point in time. It can either be `unstarted`, `ready`, `waiting` or `successful`. Roughly speaking, being in the state `unstarted` means that the module has just been initialised and needs to pick a new follower and agitator. The state `ready` represents the fact that we are waiting for an appropriate chance to define $\Delta_\alpha(C, p_k^\alpha)$. When the module passes to state `waiting`, we have defined $\Delta_\alpha(C, p_k^\alpha)$ and are now waiting for the opponent to respond with $f(p_k^\alpha) > \Delta_\alpha(C, p_k^\alpha)$. Lastly the state `successful` represents the fact that we have managed to disengage all dangerous markers, and have increased the trace size by 1 for all $x \in \mathtt{zone}_k^\alpha$. The state of a module will retain its assigned value until we assign a new state to it. There is one exception to this: if a module M_k^α has state `waiting` and a C-change occurs so that $\Delta_\alpha(C, p_k^\alpha)$ becomes undefined, then we set the module to have state `ready` (this is assumed to be a background task which is done implicitly, we do not mention this step in the construction). The reason why we want to implement this is to be consistent with the fact that

$$\begin{aligned} M_k^\alpha \text{ has state } \mathtt{ready} &\Rightarrow \Delta_\alpha(C, p_k^\alpha) \text{ is undefined,} \\ M_k^\alpha \text{ has state } \mathtt{waiting} &\Rightarrow \Delta_\alpha(C, p_k^\alpha) \text{ is defined.} \end{aligned}$$

We say that the module M_k^α is *set correctly*, if its state is not `unstarted`, and $\gamma(a_k^\alpha) \leqslant \varphi_A(k)$, with everything mentioned here defined. That is, a module is said to be set correctly if its agitator has its use in the correct place. Similarly, the global agitator g_k is said to be *set correctly*, if $\gamma(g_k) \leqslant \varphi_A(k)$. We define l_k^α to be the module ending the longest run of consecutive α-modules starting with M_k^α, that are in the state `successful`. That is, $l_k^\alpha =$ least number $j \geqslant k$ such that M_j^α is not in the state `successful`.

Definition 5.2 (Active modules)**.** With each node α we associate a collection of α-modules which we call *active modules*. At stage s, a module M_k^α is said to be active, if the following holds:

(1) $k > |\alpha|$,
(2) for every $\beta \prec \alpha$ and n such that $\beta^\frown nf \preccurlyeq \alpha$, we have $n < k < l_n^\beta$,
(3) for every $\beta \prec \alpha$ and n such that $\beta^\frown n\infty \preccurlyeq \alpha$, we have $n < k$ and for every $n \leqslant k' \leqslant k$, we have $M_{k'}^\beta$ is either `successful` or is set correctly,
(4) g_n is set correctly for all $n \leqslant k$.

When we visit α during the construction, only the α-modules which are currently active get a chance to act. The idea is that once a module M_k^β enters the state `successful`, then it never moves $\varphi_B(k)$ anymore, and so when $\alpha \succ_f \beta$ is visited we only allow α-modules which are currently active to act. This is to prevent injury to the α-modules by β-modules.

Definition 5.3 (Believable computation)**.** We define what we mean by a believable computation. A computation $\Psi(B, x)$ which converges at a stage s with use $u = \psi(x)$, is said to be M_k^α*-believable*, if for every $z \geqslant k$ such that $\varphi_B(z)\downarrow < u$, we have M_z^α is currently active, and either `successful` or is set correctly.

When we *initialise a module* M_k^α, we set p_k^α and a_k^α to be undefined, and declare the state of M_k^α as `unstarted`. When we *initalize a node* α, we first initialise all of its modules. Then, we set $T^\alpha(x) = \emptyset$ and $t_x^\alpha = x$ for all x, and restart Δ_α.

5.8. **The ordering amongst modules.** The driving force behind the construction is the action of the individual modules (instead of the actions of nodes). During the construction when a node α is visited, we will take actions for some α-module. This action will affect and injure β-modules, possibly for all nodes β comparable with α. Due to these interactions between the modules of different nodes, we will introduce the following two concepts.

We first define a local priority ordering amongst the modules (of different nodes). Note that a module M_k^α only enumerates current $\varphi_B(k)$-marker values into B (under its individual strategy). If M_k^α is an α-module, we define the set of modules with a *lower local priority*, to be all the modules M_i^β for some $\beta \prec_f \alpha$, and $k \leqslant i$. We also include all the modules M_i^α for $i > k$ in this list (i.e. larger α-modules are also of a lower local priority).

We say that a computation $\Psi_\beta(B, x)[s]$ is *a current traced computation* at s, if $\Psi_\beta(B, x)\!\downarrow \in T^\beta(x)\,[s]$, and there is no change in B below $\psi_\beta(x)$ since the time the value was traced in $T^\beta(x)$. Furthermore we also require that t_x^β was not lowered since the time the value was traced. That is, current traced computations are the computations that we should not allow to be injured easily. Another minor technical point to note is the following. We will think of the trace $T^\beta(x)$ as tracing the use of $\Psi_\beta(B, x)[s]$, instead of the value of the output (so when we enumerate $\Psi_\beta(B, x)[s]$ into T_x^β we assume that we enumerate the code of the use instead). If the computation $\Psi_\beta(B, x)$ is traced at two different times, we will have two different strings in the trace $T^\beta(x)$. At stage s, we will also say that a computation $\Psi_\alpha(B, x)$ has *persisted for two visits to* α, if $\Psi_\alpha(B, x)[s^-]\!\downarrow$ and there has been no change in B below the ψ-use between s^- and s, where s^- is the previous visit to α. We say that a computation has persisted between two stages $s < t$ if the above holds with the obvious modifications. The point of making this definition is to identify the following situation: a module M_i^β may have a currently traced computation $\Psi_\beta(B, x)$, but it may be initialised after it had made the trace of $\Psi_\beta(B, x)$ into $T^\beta(x)$. Namely, it is possible for a module to have a current traced computation, but it is not `successful`.

Next, for an α-module M_k^α, we define the *injury set of* M_k^α to be the set of modules M_i^β where $\beta \succ \alpha$, $i > k$, and such that for some $x \in \mathtt{zone}_i^\beta$, we have $\Psi_\beta(B, x)$ is a current traced computation (in this case, we also say that M_i^β has a current traced computation). Note that if M_i^β is `successful`, we will always consider it to be in the injury set as well. That is, M_i^β is a module in which some $\Psi_\beta(B, x)$-computation which has already been traced, would be injured if M_k^α decides to act and enumerate $\varphi_B(k)$ into B. Note that the modules of a lower priority depends only on the layout of the tree and do not change with time, while the injury set varies with time.

5.9. **The construction.** At stage $s = 0$, initialise all nodes and do nothing. At stage $s > 0$ we define which nodes (up to length s) are accessible at that stage; as usual, the root (the empty node) is always accessible.

Suppose then that a node α was declared to be accessible at stage s. We state the actions to be taken by α at stage s, and if $|\alpha| < s$, its successor which is next accessible. If such exists, pick the smallest $k < s$ such that k is greater than the last stage at which α was initialised, and M_k^α is currently active and requires attention. We say that M_k^α *requires attention*, if one of the following holds:

(A1) M_k^α is `unstarted`.

Otherwise, $\varphi_A(k)$ and $\varphi_B(k)$ are defined, $\Gamma(A, C, a_k^\alpha)\downarrow = P(a_k^\alpha)$, and as well as one of the following:

(A2) M_k^α is `ready`, and let s^- be the previous visit to α. There exists some largest α-stage $t \leqslant s^-$ such that after α has acted at t, we have M_k^α not `ready`. We require that for some $x \in \mathtt{zone}_k^\alpha$, $\Psi_\alpha(B, x)$ has persisted between t and s^-, but not between s^- and s. In short, some relevant computation has recently been destroyed (and possibly by α itself at s^-).

(A3) M_k^α is `ready`, but is not set correctly.

(A4) M_k^α is `ready`, and for every $x \in \bigcup_{j \leqslant k} \mathtt{zone}_j^\alpha$, $\Psi_\alpha(B, x)[s]\downarrow$ and is M_k^α-believable.

(A5) M_k^α is `waiting`, and $\gamma(a_k^\alpha) > \delta_\alpha(p_k^\alpha)[s]$.

(A6) M_k^α is `waiting`, and $\Delta_\alpha(C, p_k^\alpha)[s] < f(p_k^\alpha)[s]$.

Step 1. We will act for the α-module M_k^α. At each visit to α, at most one α-module receives attention. Choose the first item in the list above that applies, and take the corresponding action:

- (A1) applies: pick a fresh follower p_k^α and a fresh agitator a_k^α. Declare M_k^α to be in `ready` state.
- (A2) applies: do nothing. This is included to ensure that infinite outcomes of α gets a chance to act.
- (A3) applies: enumerate $\varphi_B(k)$ (if defined) into B.
- (A4) applies: let p be largest such that $\varphi_B(p)\downarrow < \psi_\alpha(x)$ for some $x \in \bigcup_{j \leqslant k} \mathtt{zone}_j^\alpha$ (we always take $p \geqslant k$), and let

  $$z = \max\{g_0, \cdots, g_p\} \cup \{a_q^\beta \,:\, \beta \preccurlyeq_\infty \alpha \ \&\ q \leqslant p\}.$$

  Note that all the parameters involved must be defined and set correctly, because of M_k^α-believability. Let

  $$m = \max\{\delta_\alpha(y)\downarrow\ [s] \,:\, y \leqslant p_k^\alpha\}.$$

  Define $\Delta_\alpha(C, y)\downarrow =$ a fresh number, with use $C_s \upharpoonright \gamma(z) + m$, for every $y \leqslant p_k^\alpha$ such that $\Delta_\alpha(C, y)\uparrow [s]$. Declare M_k^α as `waiting`.
- (A5) applies: declare M_k^α as `successful`. For every $x \in \mathtt{zone}_k^\alpha$, we enumerate the value of $\Psi_\alpha(B, x)[s]$ into the trace $T^\alpha(x)$.
- (A6) applies: enumerate a_k^α into P and pick a fresh agitator. Wait for either $C \upharpoonright \gamma(a_k^\alpha)$ or $A \upharpoonright \gamma(a_k^\alpha)$ to change (one of the two must change). If A changes do nothing, else if C changes then we enumerate $\varphi_B(k)$ (if defined) into B.

If an enumeration was made into B in Step 1, we say that *the module M_k^α was injurious*. We separate this case from the rest because these are the "bad actions" which will injure the other modules on the tree.

Step 2. We now determine the effect that our actions in step 1 have (if any) on the other modules in the construction. If M_k^α was not injurious, we do nothing. Otherwise we do the following in the specified order.

- for every $x \in \mathtt{zone}_j^\beta$ such that M_j^β is of lower local priority (than M_k^α) and $j > k$, we lower t_x^β.
- we initialise all modules of a lower local priority.

- for every $\beta \succ \alpha$, do the following. Let $i > k$ be least (if any) such that M_i^β is in the injury set of M_k^α. Note that we naturally consider the injury set before step 1 is taken, so that M_i^β is the smallest β-module which has a current traced computation being injured by the action in step 1. We then initialise all modules M_j^β for all $j \geqslant i-1$, and we lower t_x^β for every $x \in \bigcup_{j \geqslant i} \mathtt{zone}_j^\beta$.

Step 3. Now decide which outcome of α is next accessible. Suppose that M_k^α has just received attention (if no module received attention let $k = s$). If (A5) was the action taken, let the next outcome be jf, where $s^- < j \leqslant k$ is largest such that M_j^α is active and M_{j-1}^α is not `successful`. Otherwise if (A5) was not the action taken, we search for the least j with $s^- < j \leqslant k$ such that M_j^α is `ready` and active, and for some $x \in \mathtt{zone}_j^\alpha$, $\Psi_\alpha(B, x)$ has not persisted for at least two visits to α. If j exists, we let the next outcome be $j\infty$; otherwise we let the next outcome be kf.

Global actions. At the end of stage s, we initialise all nodes β that lie to the right of the last accessible node, and take the *global actions* for coding: pick the smallest $e < s$ such that either

(i) g_e is not set correctly, or
(ii) e enters $\emptyset'$ at stage s.

If (i), holds enumerate $\varphi_B(e)$ into B. On the other hand if (ii) holds, then we enumerate g_e into $\emptyset'$, and wait for $A \upharpoonright \gamma(g_e)$ or $C \upharpoonright \gamma(g_e)$ to change. If $C \upharpoonright \gamma(g_e)$ changes first then we enumerate $\varphi_B(e)$ into B. In addition, if we had enumerated $\varphi_B(e)$ into B in the previous step, we will also perform the following: for every node α and every $k \geqslant e$, we initialise the module M_k^α. We also lower t_x^α for every $x \in \mathtt{zone}_k^\alpha$ where $k > e$.

Next, we define new φ-markers. Wait for $\Gamma(A, C, z)\downarrow = P(z)$ for every $z \leqslant$ the largest number used so far. For every $e < s$ for which we have observed a change in A below $\varphi_A(e)$ or a change in B below $\varphi_B(e)$, or for which the markers have not yet been defined, we define new markers: $\varphi_B(e)$ is defined to be large, whereas $\varphi_A(e)$ is defined to be

$$\max\{\gamma(a_e^\alpha) : \alpha \text{ is a node such that } a_e^\alpha\downarrow\} \cup \{\gamma(g_e)\}.$$

Clearly there is a danger that the A-use $\varphi_A(e)$ might get driven to infinity, but we will later show that this cannot be the case.

5.10. **Verification.** It is easy to verify the following facts.

Fact 5.1. Suppose M_j^α is a module which is active at a stage s. Then for any $k < j$, either M_k^α is also active at s, or else M_k^α will never be active after s.

Fact 5.2. Observe that we never wait forever at some step of the construction. When a module M_k^α acts, it can only enumerate the current marker value of $\varphi_B(k)$ into B. If M_x^α is initialised, then all larger α-modules are also initialised by the same action. Furthermore if M_x^α is initialised then at the same time either the node α is initialised, or the markers $\varphi(x)$ becomes undefined (due to changes in either the A or B side). For any α and k, $\mathtt{zone}_k^\alpha$ is never empty. If M_j^α and M_i^α are both active at some stage s, and $j < k < i$, then M_k^α is also active at s. If $p_j^\alpha\downarrow$ and $p_k^\alpha\downarrow$ for $j < k$, then we have $p_j^\alpha < p_k^\alpha$.

It is also not too difficult to verify the following fact. The nontrivial case to consider is when M^α_{k-1} takes an injurious action, and initialises M^α_k. In this situation, M^α_{k-1} itself is not initialised, however its state will be `ready`:

Fact 5.3. If an action is taken to lower t^α_x for some $x \in \mathtt{zone}^\alpha_k$, then M^α_k will also be initialised and M^α_{k-1} will become either `unstarted` or `ready`.

Lemma 5.4. *Suppose M^α_k receives attention at s, and $\varphi(k)[s]\downarrow$ and $s^- < s$ is the stage of the most recent visit to the left of α. Then, every module M^α_i which is active at s, for $s^- < i < k$ must either be* `successful` *or be set correctly at s.*

Proof. M^α_i cannot be `unstarted` (otherwise M^α_i would have received attention at s). If M^α_i is `ready` then it has to be set correctly (otherwise again, M^α_i would have received attention). This leaves the case M^α_i is `waiting`. Since $\varphi(i)$ is defined at s, by considering $\Delta_\alpha(C, p^\alpha_i)$, it is not difficult to see that at s we must have M^α_i set correctly as well. □

Lemma 5.5. *Suppose M^α_k is* `waiting` *at stage s. Then for every $x \in \bigcup_{j\leqslant k} \mathtt{zone}^\alpha_j$, we have $\Psi_\alpha(B, x)[s]\downarrow$. Furthermore $B \upharpoonright \psi_\alpha(x)[s]$ cannot change before M^α_k has a change of state.*

Proof. Let $s_0 \leqslant s$ be the largest when (A4) holds to change M^α_k from `ready` to `waiting`. By Fact 5.3, $\mathtt{zone}^\alpha_j[s_0] = \mathtt{zone}^\alpha_j[s]$ for every $j \leqslant k$, it suffices to prove the lemma with s_0 in place of s. Fix an $x \in \mathtt{zone}^\alpha_k[s_0]$, clearly $\Psi_\alpha(B, x)[s_0]\downarrow$. Suppose that $\varphi_B(z)[s_0] < \psi_\alpha(x)[s_0]$ is enumerated into B at some least stage $t > s_0$, and some z. Suppose to the contrary that β has had no state change between s_0 and t. Suppose firstly that the enumeration was made by the global actions. We have g_z is set correctly at s_0; this follows from the fact that M^α_k is active at s_0 (if $z < k$), and M^α_k-believability if $z \geqslant k$. In either case we have $\delta_\alpha(p^\alpha_k)$ set to be larger than $\gamma(g_z)[s_0]$ at s_0, and consequently neither $\gamma(g_z)$ nor $\varphi(z)$ could have changed between s_0 and t. Hence, the global actions at t would destroy the $\Delta_\alpha(C, p^\alpha_k)$-computation set at s_0, resulting in a change of state at t.

Next, we want to show that at stage s_0 the outcome kf is played when α is visited: if not then there is some $j < k$ and $j > s_0^-$ such that M^α_j is `ready` and active. For $x \in \bigcup_{y\leqslant j} \mathtt{zone}^\alpha_y$, we claim that the $\Psi_\alpha(B, x)[s_0]$-computation (besides being M^α_k-believable) is also M^α_j-believable at s_0. This follows by Lemma 5.4 because every module M^α_i for $j \leqslant i < k$ must be active (since M^α_j and M^α_k are). Hence M^α_j would have received attention instead of M^α_k at s_0, a contradiction.

Suppose now that the enumeration of $\varphi_B(z)[s_0]$ was made by the module M^β_z for some node β, at stage t. Clearly $\beta <_L \alpha$, or $\beta >_L \alpha$ is trivial. Suppose $\beta \succcurlyeq \alpha\hat{}o$ for some outcome o. Again $o >_L kf$ is trivial, and from the fact that M^α_k is `waiting` between s and t, it follows that $o = kf$ is impossible. Suppose that $o = k'f$ for some $k' < k$. Since M^β_z is active at t it follows that at t we have $z < l^\alpha_{k'} \leqslant k$, which means that after M^β_z acts at t, we would initialise M^α_k (M^α_k being of lower local priority). If on the other hand we have $o = k'\infty$ for some $k' \leqslant k$, then stage t is strictly after s_0. We may assume $k' < k$ otherwise M^α_k is `ready` at t. At t there is some $x' \in \mathtt{zone}^\alpha_{k'}[t]$ such that $\Psi_\alpha(B, x')$ has not persisted for two visits to α. Since M^α_k had no change in state between s_0 and t, it follows that $x' \in \mathtt{zone}^\alpha_{k'}[s_0] = \mathtt{zone}^\alpha_{k'}[t]$, which means that $x' < x$. Since $\Psi_\alpha(B, x')$ had not persisted between s_0 and t, this contradicts the minimality of t.

We are now left with the case $\beta \preccurlyeq \alpha$. If $\beta^\frown nf \preccurlyeq \alpha$ then at s_0 we have $z < l^\beta_n$ (again due to either M^α_k being active or M^α_k-believability). If $z < n$ then at t some outcome to the left of nf will be played when β acts, resulting in an initialisation to α. Suppose therefore, that $n \leqslant z < l^\beta_n$, and hence M^β_z is `successful` at s_0. The only way for M^β_z to get out of `successful`, is for M^β_z to be initialised before t, and so by Fact 5.2, either β itself is initialised or $\varphi_B(z)$ is lifted between s_0 and t, another contradiction.

Finally we have the case $\beta^\frown n\infty \preccurlyeq \alpha$ or $\beta = \alpha$. We can conclude (again due to either M^α_k being active or M^α_k-believability) that M^β_z is either `successful` or is set correctly at s_0 (for $\beta = \alpha$ and $z < k$, use Fact 5.1 and Lemma 5.4). A similar argument as the one used above can be applied to show that M^β_z cannot be `successful`. Hence M^β_z has to be set correctly at s_0, and furthermore α at s_0 will set the use $\delta_\alpha(p^\alpha_k)[s_0]$ at least as big as $\gamma(a^\beta_z)[s_0]$. Between s_0 to t there can be no change in A below $\gamma(a^\beta_z)[s_0]$, because M^β_z was observed to be set correctly at s_0. There is also no change in C below $\gamma(a^\beta_z)[s_0]$, since M^α_k was assumed to have no state change. Hence at t when β gets to act, it must still be that M^β_z is set correctly, and that (A6) applies, causing us to enumerate a^β_z and get a $C \upharpoonright \gamma(a^\beta_z)$-change, which would in turn cause $\Delta_\alpha(C, p^\alpha_k)$ to become undefined after the action at stage t. □

Lemma 5.6. *At any time, if the module M^α_k is active, and $k >$ stage number of the previous visit to the left of α, the following are true:*

(i) M^α_k is not `unstarted` $\Leftrightarrow$ *p^α_k and a^α_k are both defined.*
(ii) M^α_k has state `ready` $\Rightarrow$ *$\Delta_\alpha(C, p^\alpha_k)$ is undefined.*
(iii) M^α_k has state `waiting` $\Rightarrow$ *$\Delta_\alpha(C, p^\alpha_k)$ is defined.*

Proof. (i) and (iii) are obvious.

(ii): suppose on the contrary that s is a stage such that M^α_k is `ready` and is active, but $\Delta_\alpha(C, p^\alpha_k)$ has an axiom that applies. Let $s^- < s$ be the latest action that caused M^α_k to become `ready`. Note that this must be either M^α_k receiving attention under (A1), or due to some C-change occuring. In any case it must be that $\Delta_\alpha(C, p^\alpha_k)[s^-]$ is undefined. Hence at some time t where $s^- < t < s$ we have some module M^α_j where $j > k$ taking action under (A4) and enumerating the axiom for $\Delta_\alpha(C, p^\alpha_k)[s]$. Since M^α_j is active at t, it follows by Fact 5.1 that M^α_k is also active at t, and in fact by Lemma 5.4, every M^α_i has to be active, and has to be either `successful` or be set correctly at t for all $k \leqslant i < j$. Hence at stage t when α was visited, it is not hard to see that we have $t^- < k < t$, and also that M^α_k requires attention under (A4), because every relevant computation converges and is M^α_k-believable at t. Hence it is impossible for M^α_j to act at t, a contradiction. □

Lemma 5.7. *Suppose that M^α_k has just received attention under (A5) at s. Then for every $x \in \bigcup_{j \leqslant k}$ `zone`$^\alpha_j[s]$, we have $\Psi_\alpha(B, x)[s]{\downarrow}$. Furthermore if $\varphi_B(k)[s]{\downarrow}$, then $\varphi_B(k)[s] > \psi_\alpha(x)[s]$.*

Proof. Let $s_0 < s$ be the largest stage where (A4) applies to M^α_k and causes the state changes from `ready` to `waiting`. It follows again by Fact 5.3 that `zone`$^\alpha_j[s_0] =$ `zone`$^\alpha_j[s]$ for all $j \leqslant k$, and by Lemma 5.5, we have $\Psi_\alpha(B, x)[s_0]{\downarrow}$, and this computation is not injured between s_0 and s. Furthermore at s_0 we had set $\Delta_\alpha(C, p^\alpha_k){\downarrow}$ with use $u = \delta_\alpha(p^\alpha_k)$, where we clearly have M^α_k set correctly with $\gamma(a^\alpha_k) < u$. It also follows that the values of $C \upharpoonright u$ and p^α_k did not change between

s_0 and s, so consequently $A \upharpoonright \gamma(a_k^\alpha)[s_0]$ has to change between s_0 and s in order for (A5) to hold at s. Because M_k^α was set correctly at s_0, it follows that $\varphi_B(k)[s]$ is picked fresh and so is larger than $\psi_\alpha(x)[s_0]$ (this is why it is important that $B \upharpoonright \psi_\alpha(x)[s_0]$ did not change between s_0 and s, so that we can benefit from the A-change in between). □

Lemma 5.8. *Suppose $\varphi_B(x)[s]$ is enumerated into B at stage s (by any action), and for some α and $k > x$, M_k^α is either* `successful` *or has a current traced computation at s. Then the same action will either*

- *initialise the node α, or*
- *initialise M_k^α, and lower t_x^α for every $x \in \mathtt{zone}_k^\alpha$*

Proof. We proceed by induction on the sequence of actions (or substages) in the construction. If $\varphi_B(x)$ was enumerated by the global actions, then it is clear. So, we assume some M_x^β took an injurious action. We must have β and α comparable (otherwise it is trivial), and if $\beta \succcurlyeq_f \alpha$ then it is easy. If $\beta \prec \alpha$ then one can verify that M_k^α would be in the injury set of M_x^β when β acted. The only case left is $\alpha\hat{}n\infty \preccurlyeq \beta$ for some n. We show this case is not possible.

At stage s we have M_k^α is either `successful` or has a current traced computation. Clearly we have $k > x > n$. When α was visited earlier in the stage s, we had some $y \in \mathtt{zone}_n^\alpha[s]$ where $\Psi_\alpha(B, y)$ has not persisted for two visits to α. If M_k^α has a current traced computation at s, then M_k^α would have received attention under (A5) and enumerated $\Psi_\alpha(B, z)[s_0]$ into T_z^α at some stage $s_0 \leqslant s$. Furthermore between s_0 and s, t_z^α is not lowered. On the other hand if M_k^α is successful at s, then it also receives attention under (A5) at some stage $s_0 \leqslant s$. We claim that in either case, we have $t_y^\alpha[s_0] \leqslant k$.

In the latter case, this follows because of Fact 5.3 and the fact that $k > n$. In the former case, we cannot apply Fact 5.3 directly because M_k^α might be initialised between s_0 and s. However if $z < y$ then $k = t_z^\alpha[s] \leqslant t_y^\alpha[s] = n$ is a contradiction, and on the other hand if $z \geqslant y$ then $k = t_z^\alpha[s_0] \geqslant t_y^\alpha[s_0] > k$.

In that case it follows by Fact 5.3 and by Lemma 5.5 that $\Psi_\alpha(B, y)[s_0]{\downarrow}$ and at s_0 it had already persisted for two visits to α. Therefore the visit to α at s_0 and the visit to β at s cannot take place in the same stage. Therefore $B \upharpoonright \psi_\alpha(y)[s_0]$ has to change at some action strictly between s_0 and s. By Lemma 5.7 this has to be $\varphi_B(x')$ for some $x' < k$. Apply induction hypothesis to get a final contradiction, and hence we conclude that the case $\alpha \prec_\infty \beta$ is also not possible. □

Lemma 5.9. *For any β and e, if M_e^β is initialised finitely often, then it only picks finitely many agitators a_e^β.*

Proof. Suppose t_0 is a stage after which M_e^β is never initialised. Then $p = \lim p_e^\beta$ must exist. Since after t_0, we only pick a new agitator if (A6) applies, it follows that we may assume that there are infinitely many stages $t_1 < t_2 < \cdots$ larger than t_0 such that M_e^β receives attention under (A6) at all of these stages. When M_e^β next receives attention after each t_n, we must have M_e^β `ready` (otherwise it is not hard to see by Lemma 5.6 that (A5) will apply and break the cycle), which means that $\Delta_\beta^C(p)$ receives a fresh axiom between each t_n and t_{n+1}. This is contrary to the fact that $f(p)$ reaches a limit. □

Lemma 5.10. *For each e, the following are true:*

(i) $\varphi(e)$ is moved finitely often,

(ii) there is a stage after which no module M_k^α for any α and $k \leqslant e$ is injurious.

Proof. We prove (i) and (ii) simultaneously by induction on e. Let $s_0 > e$ be large enough such that (i) and (ii) holds for all $k < e$, and also the global actions for coding have stopped acting for $\varphi(0), \cdots, \varphi(e)$. That is, the global requirements no longer enumerate $\varphi(0), \cdots, \varphi(e)$. Note that we are not assuming that (i) and (ii) of the global actions hold for e finitely often; instead we only require that the global actions enumerate $\varphi_B(e)$ into B finitely often. Let α be the leftmost node such that $|\alpha| = e - 1$ and α is visited on or after stage e (assume s_0 large enough such that this happens before s_0). Thus the only modules M_e^β which can receive attention after stage s_0, will belong to the nodes $\beta \preccurlyeq \alpha$, because modules to the right of α are never active after stage s_0. We proceed in a series of steps:

Claim 1: if $\beta_0 \prec_f \beta_1 \preccurlyeq \alpha$, then no β_0-module can take an action which initialises $M_e^{\beta_1}$ after stage s_0. The only β_0-module which can do that after s_0 is $M_e^{\beta_0}$, and so if $\beta_0{}^\frown nf \preccurlyeq \beta_1$, then we must have $n \leqslant e$ (otherwise it is trivial). Let $t_1 > s_0$ be a stage where $M_e^{\beta_0}$ acts and initialises $M_e^{\beta_1}$. Note that we only need to show that there are finitely many of such stages t_1 (since we can choose s_0 large enough for the rest of the lemma). Clearly $M_{e+1}^{\beta_1}$ is in the injury set of $M_e^{\beta_0}$ at t_1, and consequently we have $M_{e+1}^{\beta_1}$ is either `successful` at t_1, or else it has a current traced computation at t_1. In any case we can let $t_0 < t_1$ be the stage where $M_{e+1}^{\beta_1}$ receives attention under (A5), and sets things up for t_1. Note that at t_1, we will also have $M_{e+1}^{\beta_1}$ initialised and every $x \in \mathtt{zone}_{e+1}^{\beta_1}$ gets lowered, but the latter does not happen between t_0 and t_1. Therefore, if there are infinitely many such stages t_1, we may assume that t_0 is large (enough for our purpose) as well. At t_0, we have $M_e^{\beta_0}$ is `successful` (since $M_{e+1}^{\beta_1}$ is active), and hence between t_0 and t_1, the module $M_e^{\beta_0}$ has to be initialised. Since t_0 is large, the module $M_e^{\beta_0}$ has to be initialised by the actions of a third module M_e^σ for some σ, which took an injurious action. By Lemma 5.8 this is impossible, by the choice of t_0. This contradiction shows that $M_e^{\beta_0}$ has to be `successful` at t_1 still, and so cannot be injurious towards $M_e^{\beta_1}$.

Note that a β-module can receive attention infinitely often, in which case Ψ_β^C is not total. However, a module cannot be infinitely injurious; this is important because we want each module to have a finite effect on the rest of the construction:

Claim 2: for any β, if M_e^β is initialised finitely often, then it can be injurious only finitely often. This follows directly from Lemma 5.9.

Claim 3: for any β such that $\beta \preccurlyeq_\infty \alpha$, we have M_e^β is initialised finitely often. Start with the minimal such node $\beta \preccurlyeq \alpha$, and work downwards inductively by applying Claims 1 and 2.

Claim 4: if $\beta \prec_f \alpha$, then M_e^β is also initialised finitely often. This time round, start with the maximal such node $\beta \preccurlyeq \alpha$, and work upwards inductively by applying Claims 1, 2 and 3.

The claims above prove (ii) for e. We now show (i). Assume $s_1 > s_0$ is such that no module M_k^σ for any σ and $k \leqslant e$ is injurious anymore. Note that if $\sigma >_L \alpha$, then a_e^σ is undefined after s_0, and never receives a definition again. If $\sigma <_L \alpha$ then a_e^σ never receives a new value after s_0. Let $\sigma \preccurlyeq \alpha$, and by Lemma 5.9, we have $\lim a_e^\sigma$ exists. Finally, it is not hard to put the various facts together and conclude that (i) holds for e as well. □

An immediate corollary to Lemma 5.10 is that $\emptyset' \leqslant_{\mathrm{T}} A \oplus B$. We define the true path of the construction to be the leftmost path visited infinitely often. Before we can show that the true path exists, we start with a preparatory lemma:

Lemma 5.11. *Suppose α is visited at s and has just finished acting and has decided to visit the outcome $k_0\infty$. Then, it is impossible for $M^\alpha_{k_1}$ to be* `waiting` *if $k_1 > k_0$.*

Proof. Let $s^- \leqslant s$ be the stage where $M^\alpha_{k_1}$ received attention under (A4) to give its current `waiting` state. Since outcome $k_0\infty$ was played at s, it follows by Fact 5.1 that $M^\alpha_{k_0}$ would be able to receive attention at s^- if it required to do so. Hence at s^-, $M^\alpha_{k_0}$ cannot be `unstarted` or `successful`, and by Lemma 5.6(ii) we have $\Delta_\alpha(C, p^\alpha_{k_0})[s^-]\downarrow$. Hence when $\delta_\alpha(p^\alpha_{k_1})$ was set at s^-, it must be larger than $\delta_\alpha(p^\alpha_{k_0})[s^-]$. Since $M^\alpha_{k_0}$ is `ready` at s it follows that C must change below $\delta_\alpha(p^\alpha_{k_0}) < \delta_\alpha(p^\alpha_{k_1})$ between s^- and s, a contradiction. □

Note that in the above lemma, we do not actually need α to have outcome $k_0\infty$ at s. All we really require is that $k_0 > s^-$ (the previous visit to α) and $M^\alpha_{k_0}$ is active and `ready` at s.

Lemma 5.12. *The true path of construction exists.*

Proof. Suppose we have defined the true path up till α. Hence α is visited infinitely often, and nodes to the left of α are accessible at only finitely many stages. We want to show that some outcome of α is visited infinitely often. Suppose each outcome is visited finitely often; we want to derive a contradiction to the fact that C is low_2. Since α is initialised only finitely often, it follows by Lemma 5.10 that each α-module is initialised finitely often. We first show the following:

Claim 1: for almost all k, M^α_k eventually becomes active and is active at every α-stage after that. Fix k large enough. We may assume by induction hypothesis, that the statement of the claim holds for any $\beta \prec \alpha$. Let s_0 be large enough such that all parameters for M^β_j for $\beta \prec \alpha$ and $j \leqslant k$ have settled. We can do this because each β-module is initialised finitely often, and by Lemma 5.9, we know that a^β_j is never refreshed again. We want to show that (1) to (4) of Definition 5.2 holds for M^α_k forever after s_0. (1) and (4) are obvious, so we consider the other two. First consider some $\beta\hat{}nf \preccurlyeq \alpha$.

If M^β_n receives attention infinitely often, then it is not hard to see that eventually we must have M^β_n only receiving attention under (A4). Note that no β-module smaller than M^β_n can receive attention, so only M^β_n can be responsible for enumerating $\Delta^C_\beta(y)$-axioms for $y \leqslant p^\beta_n$. Therefore there must be some $x \in \mathtt{zone}^\beta_n$ (which would have settled) such that $\Psi_\beta(B, x)$ is divergent (because otherwise the variables z and m in (A4) would eventually settle). Consequently we must visit β at some stage t in which $\Psi_\beta(B, x)[t]$ has not persisted for two β-stages, and by Lemma 5.5 M^β_n would have to be `ready` at t. It follows that (A2) will apply to M^β_n at t, a contradiction (to the fact that nf is the true outcome of β) follows by applying the induction hypothesis to β. Therefore, M^β_n only receives attention finitely often, and it follows that eventually, at each time $\beta\hat{}nf$ is visited, some module M^β_j for $j \geqslant n$ would have to receive a state change from `waiting` to `successful`, and subsequently stays `successful` forever (assuming of course, that $j \leqslant k$). If we wait long enough then $l^\beta_n > k$ at every visit to α.

Now consider some $\beta\hat{}n\infty \preccurlyeq \alpha$. We want to show that eventually, every module $M^\beta_n, \ldots, M^\beta_k$ is either `successful` or is set correctly at every α-stage. Any β-module

M_j^β for $j \leqslant k$ which receives attention at an α-stage, can only have done so if (A2) applies (use Lemma 5.11 for this, and the fact that a_j^β has settled). If every module M_j^β for $n \leqslant j \leqslant k$ receives attention at finitely many α-stages, then by Lemma 5.4, we clearly have what we want. Therefore, we may assume that j is least such that M_j^β receives attention at infinitely many α-stages for some $n \leqslant j \leqslant k$. Take a large enough α-stage t where M_j^β receives attention.

Let $t^- < t$ be the previous visit to β, and $t' \leqslant t^-$ be the largest β-stage where we finished β's action with M_j^β in state `waiting`. Furthermore, t' and t^- satisfy the conditions described in (A2) of the construction. Hence at stage $t^* \leqslant t'$ when we bestowed the state `waiting` upon M_j^β, we had set $\delta_\beta(p_j^\beta) = \gamma(z) + m$ where z, m are as defined in the construction. We may assume $t^* > s_0$. Since no module smaller than M_j^β can be responsible for setting Δ_β-axioms anymore, it follows by an argument similar as above, that m will reach a limit. Hence at stage t^*, there must be some $x \in \mathtt{zone}_j^\beta[t^*]$ such that $\varphi_B(k+1)[t^*]{\downarrow} < \psi_\beta(x)[t^*]$. Since the computation $\Psi_\beta(B, x)[t^*]$ must be M_j^β-believable at t^*, we can also deduce that every module $M_n^\beta, \cdots, M_k^\beta$ is either `successful` or is set correctly. This concludes the proof of Claim 1.

By Claim 1 it follows that $p_n = \lim p_n^\alpha$ is defined for almost all n. Note that each α-module is initialised only finitely often, and receives attention finitely often (by assumption). If Ψ_α^B is not total then one can verify using Lemma 5.11 that $\alpha\hat{}n\infty$ will be visited infinitely often for some n, a contradiction. On the other hand suppose that $\Psi_\alpha(B)$, is total. If n is large enough, what is the final state of the module M_n^α? Clearly it cannot be `unstarted`, and cannot be `ready` since $\Psi_\alpha(B)$ is total. If it is `waiting` for infinitely many n, then $\Delta_\alpha(C, p_n)$ will be defined forever at infinitely many n. Consequently Δ_α^C is defined on almost all inputs, which implies (by low$_2$-ness) that $f(p_n) > \Delta_\alpha(C, p_n)$ for almost all n. This in turn implies that for all large enough n, M_n^β has to become `successful` and stay in that state forever. Hence some finitary outcome of β will be visited infinitely often (depending on where the consecutive run of `successful` modules start), contrary to assumption. Therefore, there is a leftmost outcome of α visited infinitely often. □

Lemma 5.13. *Along the true path of construction, the requirements succeed.*

Proof. Let α be on the true path of construction, and suppose that $\Psi_\alpha(B)$ is total. We show that the version of T^α built after the final initialisation of α at s_0 traces $\Psi_\alpha(B, x)$ correctly for almost all x.

Claim 1: for any $s \geqslant s_0$ and every x, we have $x+1-|T^\alpha(x)| \geqslant t_x^\alpha$. Fix two stages $s_2 > s_1 > s_0$ such that two different elements are enumerated into $T^\alpha(x)$ at s_1 and s_2. We will be done if we can show that t_x^α is decreased between s_1 and s_2. At s_1 it must be the case that M_m^α receives attention under (A5), enumerating $\Psi_\alpha(B, x)[s_1]$ into $T^\alpha(x)$ where $m = t_x^\alpha[s_1]$. By Lemma 5.7, the change in $B \upharpoonright \psi_\alpha(x)[s_1]$ has to be due to some $\varphi_B(z)$ entering B for some $z < m$. By Lemma 5.8, t_x^α is certainly lowered, which proves the claim.

From Claim 1 it follows that $|T^\alpha(x)| \leqslant x + 1$, because $\mathtt{zone}_0^\alpha$ is never lowered. It remains to show that almost every $\Psi_\alpha(B, x)$ is traced. Firstly, we argue that the true outcome of α cannot be infinitary. Suppose not, and the true outcome of α is $n\infty$ for some n. Let s_1 be large enough, so that no module M_j^α is initialised, and $\varphi(j)$ has settled, for every $j \leqslant n + 1$. Thus $\mathtt{zone}_j^\alpha$ has settled for all $j \leqslant$

n, and we also assume that $\Psi_\alpha(B,x)[s_1]\downarrow$ on the correct use for every x in the final $\bigcup_{j\leqslant n}\mathtt{zone}^\alpha_j$. By Claim 1 of Lemma 5.12 (applied to $\alpha\hat{}n\infty$) we may as well assume that all the $\Psi^B_e(x)$-computations above are M^α_n-believable. Observe that M^α_n cannot remain `ready` forever after s_1, because otherwise (A4) will eventually apply for M^α_n, and M^α_n would have received attention at the next stage where $\alpha\hat{}n\infty$ is visited. So, M^α_n eventually becomes `waiting` and enumerates some computation $\Delta_\alpha(C,p^\alpha_n)$, with a use that we can assume does not change anymore. This is a contradiction because M^α_n has to get back to state `ready` in time for the next $\alpha\hat{}n\infty$ visit.

Thus we let the true outcome of α be nf for some n. Hence $l^\alpha_n\to\infty$, because $M^{\alpha\hat{}nf}_j$ eventually becomes active for all large enough $j>n$, following from Claim 1 of Lemma 5.12. Again wait for a stage s_1 large enough so that all relevant activity in $M^\alpha_0,\cdots,M^\alpha_{n+1}$ has ceased. Let $x_0=\min\bigcup_{j\geqslant n+1}\mathtt{zone}^\alpha_j[s_1]$. We claim that $q=\Psi_\alpha(B,x)$ is traced in $T^\alpha(x)$ for all $x>x_0$. Note that $t^\alpha_x\geqslant n+1$ after s_1, so let $m\geqslant n+1$ be the final value attained by t^α_x. Let s be the time where an action was taken to make $t^\alpha_x=m$ (note that s may be smaller than s_1). If the action was an initialisation to α, then M^α_m is initialised as well. Otherwise the action was to lower t^α_x from $m+1$ to m. By Fact 5.3 we know that M^β_m will have to become `successful` after stage s, which means that at some point $s'>s$, we have (A5) applies for M^β_m and $\Psi_\alpha(B,x)[s']$ enumerated into $T^\alpha(x)$. By Lemmas 5.7 and 5.8, we have $\Psi_\alpha(B,x)[s']=p$. □

References

[ASJSS84] Klaus Ambos-Spies, Carl G. Jockusch, Jr., Richard A. Shore, and Robert I. Soare. An algebraic decomposition of the recursively enumerable degrees and the coincidence of several degree classes with the promptly simple degrees. *Trans. Amer. Math. Soc.*, 281(1):109–128, 1984.

[ASLS93] Klaus Ambos-Spies, Alistair H. Lachlan, and Robert I. Soare. The continuity of cupping to $\mathbf{0}'$. *Ann. Pure Appl. Logic*, 64(3):195–209, 1993.

[BM82] Mark Bickford and Chris F. Mills. Lowness properties of r.e. sets. Manuscript, UW Madison, 1982.

[CGS01] Peter Cholak, Marcia Groszek, and Theodore Slaman. An almost deep degree. *J. Symbolic Logic*, 66(2):881–901, 2001.

[DGMW08] Rod Downey, Noam Greenberg, Joseph S. Miller, and Rebecca Weber. Prompt simplicity, array computability and cupping. In *Computational prospects of infinity. Part II. Presented talks*, volume 15 of *Lect. Notes Ser. Inst. Math. Sci. Natl. Univ. Singap.*, pages 59–77. World Sci. Publ., Hackensack, NJ, 2008.

[Dia09] David Diamondstone. Promptness does not imply superlow cuppability. *J. Symbolic Logic*, 74(4):1264–1272, 2009.

[DJS90] Rodney G. Downey, Carl G. Jockusch, Jr., and Michael Stob. Array nonrecursive sets and multiple permitting arguments. In *Recursion theory week (Oberwolfach, 1989)*, volume 1432 of *Lecture Notes in Math.*, pages 141–173. Springer, Berlin, 1990.

[DJS96] Rod Downey, Carl G. Jockusch, and Michael Stob. Array nonrecursive degrees and genericity. In *Computability, enumerability, unsolvability*, volume 224 of *London Math. Soc. Lecture Note Ser.*, pages 93–104. Cambridge Univ. Press, Cambridge, 1996.

[FS81] Peter A. Fejer and Robert I. Soare. The plus-cupping theorem for the recursively enumerable degrees. In *Logic Year 1979–80 (Proc. Seminars and Conf. Math. Logic, Univ. Connecticut, Storrs, Conn., 1979/80)*, volume 859 of *Lecture Notes in Math.*, pages 49–62. Springer, Berlin, 1981.

[HS82] Leo A. Harrington and S. Shelah. The undecidability of the recursively enumerable degrees. *Bull. Amer. Math. Soc*, 6:79–80, 1982.

[HS92] Leo Harrington and Robert I. Soare. Games in recursion theory and continuity properties of capping degrees. In *Set theory of the continuum (Berkeley, CA, 1989)*, volume 26 of *Math. Sci. Res. Inst. Publ.*, pages 39–62. Springer, New York, 1992.

[Ish99] Shamil Ishmukhametov. Weak recursive degrees and a problem of Spector. In *Recursion theory and complexity (Kazan, 1997)*, volume 2 of *de Gruyter Ser. Log. Appl.*, pages 81–87. de Gruyter, Berlin, 1999.

[Li] Angsheng Li. A hierarchy characterisation of cuppable degrees. University of Leeds, Dept. of Pure Math., 2001, Preprint series, No. 1, 21pp.

[LWZ00] Angsheng Li, Guohua Wu, and Zaiyue Zhang. A hierarchy for cuppable degrees. *Illinois J. Math.*, 44(3):619–632, 2000.

[Mil81] David P. Miller. High recursively enumerable degrees and the anticupping property. In *Logic Year 1979–80 (Proc. Seminars and Conf. Math. Logic, Univ. Connecticut, Storrs, Conn., 1979/80)*, volume 859 of *Lecture Notes in Math.*, pages 230–245. Springer, Berlin, 1981.

[NSS98] André Nies, Richard A. Shore, and Theodore A. Slaman. Interpretability and definability in the recursively enumerable degrees. *Proc. London Math. Soc. (3)*, 77(2):241–291, 1998.

[Sac63] Gerald E. Sacks. On the degrees less than $\mathbf{0}'$. *Ann. of Math. (2)*, 77:211–231, 1963.

School of Mathematics and Statistics, Victoria University of Wellington, Wellington, New Zealand

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

Division of Mathematical Sciences, School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore 637371, Singapore

E-mail address: kmng@ntu.edu.sg

Division of Mathematical Sciences, School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore 637371, Singapore

E-mail address: guohua@ntu.edu.sg