Computably enumerable Turing degrees and the meet property

COMPUTABLY ENUMERABLE TURING DEGREES AND THE MEET PROPERTY

BENEDICT DURRANT, ANDY LEWIS-PYE, KENG MENG NG, AND JAMES RILEY

Abstract. Working in the Turing degree structure, we show that those degrees which contain computably enumerable sets all satisfy the meet property, i.e. if $\boldsymbol{a}$ is c.e. and $\boldsymbol{b} < \boldsymbol{a}$, then there exists non-zero $\boldsymbol{m} < \boldsymbol{a}$ with $\boldsymbol{b} \wedge \boldsymbol{m} = \boldsymbol{0}$. In fact, more than this is true: $\boldsymbol{m}$ may always be chosen to be a minimal degree. This settles a conjecture of Cooper and Epstein from the 80s.

1. Introduction

The Turing degrees, introduced in [9] and implicit in [16], provide a way of comparing the computational difficulty of decision problems. Of particular interest are those degrees containing the characteristic functions of computably enumerable (c.e.) sets, i.e. those sets of natural numbers for which there exists an algorithm enumerating precisely the elements of the set. Another natural class of interest are those degrees containing Δ^0_2 sets, i.e. those sets whose characteristic functions are computable relative to Turing's halting problem. Although these structures have been very extensively studied (for a good introduction see, for example, [1, 15]) many of even the most basic structural questions seem to require sophisticated techniques and often remain unanswered. In this paper we establish the following simple fact: given any non-zero c.e. degree $\boldsymbol{a}$ and any (necessarily Δ^0_2) degree $\boldsymbol{b} < \boldsymbol{a}$, there is a minimal degree $\boldsymbol{m} < \boldsymbol{a}$ such that $\boldsymbol{m} \not\leq \boldsymbol{b}$. This settles a conjecture from [5] and also refutes the conjecture made in [2, 3] (and dating back to 1986) that there exist c.e. degrees which fail to satisfy the meet property.

Our result can be seen as belonging to a long line of research concerning the *complementation, meet* and *join* properties:

(1) A degree $\boldsymbol{a}$ satisfies the complementation property, if for all non-zero $\boldsymbol{b} < \boldsymbol{a}$, there exists $\boldsymbol{c} < \boldsymbol{a}$ with $\boldsymbol{b} \vee \boldsymbol{c} = \boldsymbol{a}$ and $\boldsymbol{b} \wedge \boldsymbol{c} = \boldsymbol{0}$.
(2) A degree $\boldsymbol{a}$ satisfies the join property, if for all non-zero $\boldsymbol{b} < \boldsymbol{a}$, there exists $\boldsymbol{c} < \boldsymbol{a}$ with $\boldsymbol{b} \vee \boldsymbol{c} = \boldsymbol{a}$.
(3) A degree $\boldsymbol{a}$ satisfies the meet property, if for all $\boldsymbol{b} < \boldsymbol{a}$, there exists non-zero $\boldsymbol{c} < \boldsymbol{a}$ with $\boldsymbol{b} \wedge \boldsymbol{c} = \boldsymbol{0}$.

For a full description of known results relating to these properties, we refer the reader to [10], or for a slightly shorter version we refer to [11]. We mention here only a few of those results which are pertinent. Lewis-Pye, Slaman and Seetapun [12] established that $\boldsymbol{0'}$ satisfies complementation in a very strong way, namely that the complementing degree $\boldsymbol{c}$ can always be chosen to be minimal (i.e. $\boldsymbol{c} > \boldsymbol{0}$ and there does not exist $\boldsymbol{d}$ with $\boldsymbol{0} < \boldsymbol{d} < \boldsymbol{c}$). On the other hand, Cooper [4] and independently Slaman and Steel [14] had already shown that there are c.e. degrees which do not satisfy the join property. In 1979 Epstein [6] conjectured that in the lower cone below any c.e. degree $\boldsymbol{a}$, any non-zero c.e. degree $\boldsymbol{b} < \boldsymbol{a}$ has a minimal complement, and proved this for the case that $\boldsymbol{a}$ is high [7]. The full conjecture was refuted by Cooper and Epstein in [5]. In that same paper, however, it was shown that in the case that $\boldsymbol{a}$ is low, one can always find a minimal degree $\boldsymbol{m} < \boldsymbol{a}$ for which $\boldsymbol{b} \wedge \boldsymbol{m} = \boldsymbol{0}$, thereby establishing a weak form of the meet property for $\boldsymbol{a}$ which

Lewis-Pye was supported by a Royal Society University Research Fellowship.

applies only to c.e. predecessors $\boldsymbol{b}$. It was conjectured in that paper that one cannot drop either of the assumptions that $\boldsymbol{a}$ is low or that $\boldsymbol{b}$ is c.e.. Contradicting an earlier claim by Li and Yang [13], Ishmukhametov [8] was able to refute part of this conjecture, showing that one can drop the assumption of lowness. The result of this paper establishes that the assumption that $\boldsymbol{b}$ be c.e. is also unnecessary.

2. Every c.e. degree has the meet property

Theorem 2.1. *Given any non-zero c.e. degree $\boldsymbol{a}$ and any degree $\boldsymbol{b} < \boldsymbol{a}$, there is a minimal degree $\boldsymbol{m} < \boldsymbol{a}$ such that $\boldsymbol{m} \not\leq \boldsymbol{b}$.*

Corollary 2.2. *Every c.e. degree has the meet property.*

2.1. **Requirements and notation.** We let p_i be the ith prime number. We let Φ_e denote the eth Turing functional according to some fixed effective listing. Then for any $C \subseteq \mathbb{N}$ and any $e, n \in \mathbb{N}$, $\Phi_e^C(n)$ denotes the output of the eth Turing functional with oracle input C on argument n (which may or may not be defined). For a finite binary string σ we adopt the convention that $\Phi_e^\sigma(n) \downarrow$ (i.e. is defined) only if the corresponding computation converges in at most $|\sigma|$ many steps, $|\sigma| > n$ and $\Phi_e^\sigma(n') \downarrow$ for all $n' \in \mathbb{N}$ with $n' < n$. For $C \subseteq \mathbb{N}$ and $n \in \mathbb{N}$, $C \upharpoonright n$ denotes the initial segment of (the characteristic function of) C of length n. We suppose that we are given the c.e. set A with computable enumeration $\{A_s\}_{s\in\mathbb{N}}$. We also suppose we are given $B <_T A$ and a Turing functional Γ such that $B = \Gamma^A$. Speeding up the enumeration of A as necessary, we let $\{B_s\}_{s\in\mathbb{N}}$ be a computable approximation of B, with B_s a finite binary string of length s such that $B_s \subseteq \Gamma^{A_s}$. Although B_s is a finite binary string, we consider A_s to be an infinite string (which is nevertheless the characteristic function of a finite set).

So we are given $A \in \boldsymbol{a}$ and $B \in \boldsymbol{b}$ such that $\boldsymbol{b} < \boldsymbol{a}$, and we must construct M of minimal degree $\boldsymbol{m} < \boldsymbol{a}$ which is not below $\boldsymbol{b}$. We build M by specifying a computable approximation $\{\mu_s\}_{s\in\mathbb{N}}$, where each μ_s is a finite binary string, and then we let $M = \lim_s \mu_s$. We need to meet the requirements $\mathcal{P}_0, \mathcal{P}_1 \cdots$, where $\mathcal{P}_e$ ensures that if Φ_e^B is total then $\Phi_e^B \neq M$. To ensure the minimality of $\boldsymbol{m}$, we satisfy requirements $\mathcal{M}_0, \mathcal{M}_1, \cdots$. So $\mathcal{M}_e$ is the e^{th} minimality requirement, and ensures that if Φ_e^M is total then either it is computable or else it computes M.

2.1.1. *The construction tree.* In standard fashion, the construction is organised with the help of a *construction tree* T. This tree is labelled as follows. Each node of length $2e$ (for $e \in \mathbb{N}$) is assigned the requirement $\mathcal{M}_e$, with two outcomes $\infty <_L f$. The outcome ∞ indicates that 'splits' (as defined later) are found above infinitely many initial segments of M, while the outcome f indicates that the latter condition does not hold. A node of length $2e+1$ is assigned the requirement $\mathcal{P}_e$, with infinitely many outcomes labelled $0 <_L 1 <_L 2 <_L \cdots <_L f$. Hence the set of outcomes for $\mathcal{P}_e$ has order type $\omega + 1$ with a single distinguished rightmost outcome f. Outcome f indicates either that there is some argument m for which $\Phi_e^{B_s}(m) \downarrow$ for only finitely many s, or else that the requirement is satisfied directly through diagonalisation (and so requires only finite action). Each of the other outcomes can be thought of as a guess as to the least m for which there are infinitely many stages at which $\Phi_e^{B_s}(m) \downarrow$ with different uses (and thus the observed uses are unbounded).

For nodes of the construction tree α and β, we write $\alpha <_L \beta$, to denote that α is strictly to the left of β. Then we consider the nodes to be ordered lexicographically, so that α has *higher priority* than β if either $\alpha <_L \beta$, or $\alpha \subset \beta$. At each stage s we define TP_s of length s, which indicates the nodes on T which are *visited* by the construction at stage s.

Typically we use letters α, β and γ to refer to nodes of the construction tree T. We use σ and τ for potential initial segments of A and B respectively. We use η and μ for potential initial segments of M. The variable ρ we use to range over binary strings more generally. If $\rho \neq \emptyset$ then we let $\rho^\dagger$ be the binary string of the same length as ρ which differs only on the last bit, and we let $\rho^- = \rho \cap \rho^\dagger$,

i.e. the initial segment of ρ of length $|\rho| - 1$. If α on T is assigned the requirement $\mathcal{M}_e$ or $\mathcal{P}_e$, then we shall often write Φ_α to denote Φ_e.

2.2. **Outline of the proof.**

2.2.1. *Ensuring $M \leq_T A$.* We shall enumerate axioms for a functional Ψ such that $\Psi^A = M$. These axioms will be enumerated at the end of each stage s, for arguments $n < s$. The *use* for argument $n < s$ at stage s is chosen as follows. Let s_0 be the maximum of n and $s_1 - 1$ for any stage $s_1 \leq s$ at which a number $\leq n$ has been enumerated into A. Let σ be the shortest initial segment of A_s such that $B_s \upharpoonright (s_0 + 1) \subseteq \Gamma^\sigma$ (such a σ exists according to our conventions regarding B_s). Then at the end of stage s, we define $\Psi^\sigma(n) = \mu_s(n)$.

At any point during stage s we say that η is *permissible* if it is compatible with Ψ^{A_s}.

2.2.2. *Satisfying $\mathcal{M}_e$.* The approach taken in order to satisfy each requirement $\mathcal{M}_e$ is entirely standard. For a good exposition of how to build a minimal degree below an arbitrary non-zero c.e. degree, see []. We shall henceforth assume that the reader has an understanding of these techniques, which rely heavily on the use of various kinds of *trees*. A tree is a partial computable function $S : 2^{<\mathbb{N}} \mapsto 2^{<\mathbb{N}}$ satisfying the usual properties: (i) $dom(S)$ is downwards closed under initial segment, (ii) $S(\rho * 0) \downarrow$ iff $S(\rho * 1) \downarrow$, (iii) S preserves $\subset$, i.e., $\rho \subset \rho'$ iff $S(\rho) \subset S(\rho')$. We say that $\eta \in S$, or η is on S, if there is some ρ such that $S(\rho) = \eta$, and that η is of *level* k on S if $|\rho| = k$. If $|\rho| > 0$, we say that $S(\rho)$ is a *successor* of $S(\rho^-)$ on S. A string η is an *S-leaf* if η is a maximal string on S (i.e. is on S and has no successors on S). S is a subtree of T, denoted $S \subseteq T$, if every string on S is on T.

So to meet requirement $\mathcal{M}_e$ we employ the usual idea of 'splitting trees'. Each node α on the construction tree and of even length, maintains a splitting tree S_α. We say that η_0 and η_1 are *α-splitting*, or that they α-split, if $\Phi_\alpha^{\eta_0}$ and $\Phi_\alpha^{\eta_1}$ are incompatible. Further, we say that they are an *α-splitting above η* if it also holds that η_0 and η_1 extend η. The tree S_α will be α-splitting, which means that whenever η_0 and η_1 are on S_α and are incompatible, they α-split. If $\alpha = \emptyset$, then we let S_{α^*} be the identity tree, and otherwise we define α^* to be the longest $\beta \subset \alpha$ of even length on T such that $\beta * f \not\subseteq \alpha$ (although this is not always defined, we assume that Φ_0 is the identity functional and always play outcome ∞ for the node $\emptyset$, which has the consequence that α^* will be defined for any α which is actually visited). Then S_α is constructed as a subtree of S_{α^*}.

During the construction, when μ_s extends η which is an S_α-leaf, we search for η_0 and η_1 on S_{α^*} which both extend η, and which α-split. All such pairs found are stored in a list, and as soon as one becomes permissible (i.e. both η_0 and η_1 are permissible) we enumerate these strings into S_α (so if $\eta = S(\rho)$ then we define $S(\rho * 0)$ and $S(\rho * 1)$ to be η_0 and η_1). Ultimately, if α is on the true path (i.e. is visited at infinitely many stages and only initialised finitely often), standard arguments show that if S_α is infinite then $M \leq_T \Phi_\alpha^M$, and that otherwise Φ_α^M is either partial or else is computable.

2.2.3. *Satisfying $\mathcal{P}_e$.* Suppose that α is assigned the requirement $\mathcal{P}_e$. To meet $\mathcal{P}_e$ we ensure that if Φ_e^B is total, then there exists some m with $\Phi_e^B(m) \neq M(m)$. Very roughly, the idea works as follows. Suppose that we monitor $\Phi_e^B \upharpoonright n$ for a fixed n. If we find at some stage s that $\Phi_e^B \upharpoonright n$ agrees with $\mu_s \upharpoonright n$, then we know that a suitable A-change (i.e. a sufficiently small number being subsequently enumerated into A) would mean that we could change our approximation to M below n, and thus successfully diagonalise should it be the case that $B_s \subset B$. While waiting for such an A-change we can map the initial segment of B involved in this computation to the initial segment of A we are waiting to change, and begin working for larger n. Since $A \not\leq_T B$, ultimately we must either get some A-permission which allows us to diagonalise, or else there must be some n for which $M \upharpoonright n$ is not an initial segment of Φ_e^B.

Now let us see in more detail the way in which such a strategy is implemented in the context of a construction in which we must coordinate with minimality requirements. The node α will build a Turing functional Ψ_α. It also maintains infinitely many modules $M_\alpha^0, M_\alpha^1, \cdots$. The module M_α^i is responsible for enumerating axioms for $\Psi_\alpha(i)$, where we threaten to make $\Psi_\alpha^B(i) = A(i)$. Letting S_{α^*} be defined as in Section 2.2.2, α will work in the tree S_{α^*}. For convenience we build $\Psi_\alpha(i)$ to be a c.e. set of strings. We ensure that after i enters A (if ever) no more strings are enumerated into $\Psi_\alpha(i)$. Then to compute $\Psi_\alpha^X(i)$, one runs the enumeration of $\Psi_\alpha(i)$ until either τ is found such that $\tau \subset X$, or else i enters A. In the former case we output 0 otherwise we output 1.

While $i \notin A$, module M_α^i waits until it sees $\eta \subseteq \Phi_e^\tau$ for some $\tau \subseteq B_s$ and $\eta \subseteq \mu_s$ such that $\eta = S_{\alpha^*}(\rho)$ for some ρ which is specific to this module. Then it enumerates τ into $\Psi_\alpha(i)$ as well as the *demand* $(\tau, i, \eta_0, \eta_1)$, where $\eta_0 = S_{\alpha^*}(\rho^-)$ and $\eta_1 = S_{\alpha^*}(\rho^\dagger)$. This demand should be read "if $\tau \subset B$ and $i \in A$, then $\eta_0 \subset M \Rightarrow \eta_1 \subset M$."

When a demand is acted upon and so plays a role in the definition of μ_s we shall say that it is *implemented* at stage s (this will be specified precisely during the construction).

The reader might wonder why we do not issue demands of a simpler form such as "if $\tau \subset B$ and $i \in A$, then $\eta_1 \subset M$", or "if $\tau \subset B$ and $i \in A$, then $\eta \not\subset M$." The answer is that the question of A-permission for issued demands is a slightly delicate matter, requiring strategies to the left of TP_s to have influence at stage s. The problem with the first of these two alternatives is that if $i \in A$ then we shall have permission to change our mind as to whether $\eta_1 \subseteq M$ as our information changes as to whether $\tau \subseteq B$, *so long as* μ_0 is an initial segment of our approximations to M. The second alternative leads to more subtle problems concerning the interactions between $\mathcal{P}$ requirements.

2.3. **Formal construction.**

2.3.1. *Initialisation.* First of all, let us specify precisely what it means for a node α on the construction tree to be initialised, and when exactly this takes place.

If α is assigned a requirement $\mathcal{M}_e$ then α being initialised means that we make $S_\alpha(\rho) \uparrow$ for all ρ. We also discard any splittings found, i.e. α's list of splittings becomes empty. If α is assigned a requirement $\mathcal{P}_e$ then α being initialised means that we discard all axioms enumerated for Ψ_α, together with all demands issued by modules maintained by α. We also discard all 'recorded computations' for α – what 'recorded computations' are will be specified during the construction. Whichever type of requirement is assigned to α we also make z_α undefined (so z_α is just a number which is chosen to be large every time α is first visited after being initialised, and which we shall use to make sure that α doesn't interfere with nodes of higher priority).

The conditions which cause α to be initialised are independent of the type of requirement assigned. The node α is initialised at stage s as soon as any of the following conditions are satisfied:

(a) $s = 0$.
(b) A node strictly to the left of α is visited.
(c) β enumerates strings into S_β at stage s, and either $\beta * f \subseteq \alpha$ or $\beta <_L \alpha$.
(d) A demand issued by a module M_β^j, such that either $\beta <_L \alpha$ or $\beta * i \subseteq \alpha$ for $j < i$, is *implemented* at stage s but that demand was not implemented at stage $s-1$, or vice versa, the demand was implemented at stage $s-1$ but is not implemented at stage s.

At any point, a module is *active* if it has been visited subsequent to its last initialisation. A tree S_α is active if α is.

So if $s = 0$ then all nodes are initialised. At stages $s > 0$, the instructions consist of four phases (the third phase being that at which we visit nodes on the construction tree).

2.3.2. *Phase 1. Tree enumeration.* For each α which is assigned a minimality requirement and is active, in order of priority, consider α's list of splittings. If there exists a first which is permissible, then enumerate this splitting into S_α and empty α's list of splittings.

2.3.3. *Phase 2. Defining μ_s.* We perform the following iteration, which terminates after a finite number of steps. At each step we redefine the value μ_s^*, which is initially the empty string, and then ultimately μ_s takes the final value μ_s^*. The iteration simply defines a path through the nested splitting trees, taking account of issued demands when they exist, and taking the left path otherwise. As we proceed, we also enumerate pairs of the form (μ, β) in order to keep track of the priority with which we have implemented demands.

Step 0. Define $\mu_s^* = \emptyset$.
Step $k > 0$. Check to see whether there exists a demand issued by some M_β^i of the form $(\tau, i, \eta_0, \eta_1)$ such that $\tau \subseteq B_s$, $i \in A_s$, $\eta_0 \subseteq \mu_s^*$, and such that we have not already enumerated any pair (μ, γ) during the iteration at stage s with $\mu \supset \eta_0$ and γ of higher priority than β. If so, choose that for which η_0 is shortest, declare that this demand is *implemented* at stage s, redefine $\mu_s^* = \eta_1$, enumerate the pair (η_1, β) and go to the next step. Otherwise check to see whether there exists α such that $\mu_s^* \in S_\alpha$ but is not an S_α-leaf. If not then define $\mu_s = \mu_s^*$ and terminate the iteration, and otherwise let α be the lowest priority of all such nodes. Let μ be the left successor of μ_s^* on S_α then redefine μ_s^* to be μ and go to the next step.

Note that implemented demands may subsequently be *injured* by another of higher priority, i.e. for the implemented demand $(\tau, i, \eta_0, \eta_1)$ it may not be the case that $\eta_1 \subseteq \mu_s$.

2.3.4. *Phase 3. Visiting phase.* In this phase we define TP_s, the nodes visited at stage s. Let $\alpha = TP_s \upharpoonright i$ be defined. We describe the actions taken by α and decide the outcome played. If $|\alpha| \geq s$ then α performs no actions at this stage, and we terminate phase 3 of stage s. Otherwise we choose z_α to be a large odd number if it not already defined. For the remaining instructions there are then two cases.

(i) α is assigned $\mathcal{M}_e$. If $S_\alpha(\emptyset) \uparrow$ we set $S_\alpha(\emptyset) = S_{\alpha^*}(\rho)$ where $|\rho| = z_\alpha$ and $S_{\alpha^*}(\rho) \subseteq \mu_s$, or if no such ρ exists then we leave $S_\alpha(\emptyset) \uparrow$. Otherwise if $\eta \subset \mu_s$ for some S_α-leaf η, search for Φ_e-splittings above η of length $\leq s$ consisting of strings of odd level on S_{α^*}, and enumerate any found into α's list of splittings.

If strings have been enumerated into S_α since the last stage at which α was visited or if $\alpha = \emptyset$, then α has outcome ∞. Otherwise it has outcome f.

(ii) α is assigned $\mathcal{P}_e$. We determine the least $i < s$ such that M_α^i *requires attention*. This is true if there exists $\mu \subseteq \mu_s$ such that $\mu = S_{\alpha^*}(\rho)$ for ρ of length $p_{z_\alpha}^i$, and $\mu \subseteq \Phi_e^\tau$ for some shortest $\tau \subseteq B_s$, but M_α^i has not yet 'recorded the computation Φ_e^τ' (see next paragraph).

If no M_α^i requires attention then α performs no action and has outcome f. Otherwise, let i be the least such that M_α^i requires attention, and proceed as follows. Declare Φ_e^τ to be a recorded computation. If $i \notin A$ then issue the demand $(\tau, i, \eta_0, \eta_1)$, where ρ is as above, $\eta_0 = S_{\alpha^*}(\rho^-)$ and $\eta_1 = S_{\alpha^*}(\rho^\dagger)$, and also enumerate τ into $\Psi_\alpha(i)$. At stage s, α then has outcome i.

2.3.5. *Phase 4. Defining Ψ.* For each $n < s$ such that $\mu_s(n) \downarrow$ proceed as follows. Let s_0 be the maximum of n and $s_1 - 1$ for any stage $s_1 \leq s$ at which a number $\leq n$ has been enumerated into A. Let σ be the shortest initial segment of A_s such that $B_s \upharpoonright (s_0 + 1) \subseteq \Gamma^\sigma$. Define $\Psi^\sigma(n) = \mu_s(n)$.

2.4. **Verification.** First of all we must verify that the instructions are well defined. The only point of contention is during phase 2 of stage s when the instructions for step $k > 0$ require us to select the relevant demand $(\tau, i, \eta_0, \eta_1)$ for which η_0 is shortest. We must verify that there exists a

unique such demand. Once we have done this it will be clear that the instructions for each stage (in particular those for phase 2) are finite, since:

(*a*) At stage s, if the demand $(\tau, i, \eta_0, \eta_1)$ is implemented at step k (of the iteration during phase 2) and $(\tau', j, \eta_2, \eta_3)$ is implemented at step $k' > k$, then η_2 properly extends η_0.
(*b*) At each stage only finitely many demands are issued and only finitely many strings are enumerated into trees.

So we wish to ensure:

(†) At any point of the construction, if the two demands $(\tau, i, \eta_0, \eta_1)$ and $(\tau', j, \eta_2, \eta_3)$ have both been issued (and not discarded by initialisation), then $\eta_0 = \eta_2$ implies $i = j$ and that both demands were issued by the same module M^i_α.

Now since z_α is chosen to be large whenever a node is visited for the first time subsequent to initialisation, it follows that when $\mu \in S_\alpha \cap S_{\alpha'}$ for distinct nodes α and α', we must have $\alpha * \infty \subset \alpha'$ or $\alpha' * \infty \subset \alpha$. When a string belongs to two valid trees, in other words, it must be the case that one of these trees is built purposely as a subtree of the other. The following three facts then combine to give (†), as required:

(1) For α of even length, strings in S_α are of odd level in S_{α^*}.
(2) If M^i_α issues a demand $(\tau, i, \eta_0, \eta_1)$ then η_0 is of even level in S_{α^*}.
(3) If α_1 and α_2 are of odd length, are both valid and $S_{\alpha_1^*} = S_{\alpha_2^*}$, then from the fact that $z_{\alpha_1} \neq z_{\alpha_2}$ it follows that for any two demands $(\tau, i, \eta_0, \eta_1)$ and $(\tau', j, \eta_2, \eta_3)$ issued by modules $M^i_{\alpha_1}$ and $M^j_{\alpha_2}$ respectively, we have $\eta_0 \neq \eta_2$.

So far, we have concluded that the construction is well defined, and that the instructions at each stage are finite.

Lemma 2.3. *At every stage s, μ_s is permissible. M is total and $\Psi^A = M$.*

Proof. From the incomputability of A it immediately follows that $S_\emptyset$ is infinite. For every length ℓ there must therefore exist s with $|\mu_s| > \ell$. Since the use of Ψ on argument n is clearly bounded, the second statement of the lemma follows from the first.

The proof is by induction on s. Suppose that μ_s is incompatible with μ_{s-1}, and consider the iteration that takes place during phase 2 of stages s and $s-1$. At each step of the iteration a certain instruction is carried out: either a) a demand is implemented, or b) we find α of lowest priority such that $\mu_s^* \in S_\alpha$ but is not an S_α-leaf and redefine μ_s^* to be the left successor of its previous value on S_α, or c) we terminate the iteration. There must therefore be a least step k at which the two iterations at stages s and $s-1$ diverge, i.e. at which they carry out different instructions. There are three possibilities to consider:

(1) Step k at stage $s-1$ implements a demand $(\tau, i, \eta_0, \eta_1)$ and during step k at stage s it is not the case that any demand $(\tau', j, \eta_2, \eta_3)$ is implemented with $\eta_2 \subset \eta_0$.

In this case, i was enumerated into A at a stage $> |\tau|$, and since η_0 is of length $> i$, any $\sigma \subset A_{s-1}$ such that $\eta_0 \subseteq \Psi^\sigma$ at the end of stage $s-1$, is sufficiently long that $\tau \subseteq \Gamma^\sigma$. Since the demand $(\tau, i, \eta_0, \eta_1)$ is not implemented at stage s, $\tau \not\subseteq B_s$, and so any extension of η_0 is permissible.

(2) Step k at stage s implements a demand $(\tau, i, \eta_0, \eta_1)$ and during step k at stage $s-1$ it is not the case that any demand $(\tau', j, \eta_2, \eta_3)$ is implemented with $\eta_2 \subseteq \eta_0$.

In this case, the fact that the demand $(\tau, i, \eta_0, \eta_1)$ was not implemented at stage $s-1$ leaves two possibilities. It could be that i was enumerated into A at stage s, in which case any extension of η_0 is permissible. Otherwise, it must be that $\tau \not\subseteq B_{s-1}$. Now we can argue much as in case (1). We have that i was enumerated into A at a stage $s' > |\tau|$, and since η_0 is of length $> i$, any $\sigma \subset A_{s-1}$ such that $\eta_0 \subseteq \Psi^\sigma$ at the end of stage $s-1$, is sufficiently

long that $\tau' \subseteq \Gamma^\sigma$, where τ' is the initial segment of B_{s-1} of length s'. Again, we conclude that any extension of η_0 is permissible.

(3) The previous cases do not hold, and at step k of stage s case b) holds i.e. we find α of lowest priority such that $\mu_s^* \in S_\alpha$ but is not an S_α-leaf and redefine μ_s^* to be the left successor of its previous value on S_α.

In this case $\mu := \mu_s^*$ (before its redefinition at step k) was a leaf of S_α prior to stage s. The two successors of μ in S_α were enumerated into this tree at stage s and are both permissible. Let μ' be the longest string which is an initial segment of both successors of μ in S_α and also of μ_{s-1}. There are now further possibilities to consider. If $\mu' \subset \mu_s$ then μ_s is permissible. Otherwise there must be a demand $(\tau, i, \eta_0, \eta_1)$ such that $\eta_0 \subset \mu'$, and which is implemented at step $k+1$ of stage s. If this demand was also implemented at step $k+1$ of stage $s-1$ then the two processes have not really diverged in a meaningful way. We can say that, in this case, the two iterations did not *strongly* diverge at step k, since the same demand was implemented anyway at the next step, and choose instead the least step at which the two iterations strongly diverge. Given that a demand is implemented at step $k+1$ of stage s, which was not implemented at step $k+1$ of stage $s-1$, we now have two cases to consider, which are identical to cases (1) and (2), but with 'k' replaced by '$k+1$'.

□

Lemma 2.4. *For all n, there exists a leftmost node of length n which is visited infinitely often, α_n say. This node satisfies the following:*

(1) *α_n is initialised only finitely many times.*
(2) *If α_n is of length $2e+1$ then it ensures $\mathcal{P}_e$ is satisfied. Either α_n has outcome f at all sufficiently large stages at which it is visited, or else there exists some least m such that α_n has outcome m at infinitely many stages.*
(3) *If α_n is of length $2e$ and has outcome ∞ at infinitely many stages then S_α is infinite and $M \leq_T \Phi_e^M$. Otherwise S_α is finite and Φ_e^M is partial or computable.*

Proof. The proof is by induction on n. Our assumptions concerning Φ_0 mean that the result for $n = 0$ is clear. So suppose $n > 0$ and that the result holds for all $n' < n$. Then (2) of the induction hypothesis implies that α_n exists, i.e. there exists a leftmost node of length n which is visited at infinitely many stages, and also that there are only finitely many stages at which nodes strictly to the left of α_n are visited. Also, (3) of the induction hypothesis implies that for β of even length with $\beta * f \subset \alpha_n$, S_β is finite. Those $\beta <_L \alpha_n$ are only visited finitely many times, and so can only enumerate finitely many splittings into their lists. We conclude that α_n satisfies any of the conditions for initialisation (a),(b) or (c), at only finitely many stages. We are left to deal with (d). Those modules M_β^j, such that either $\beta <_L \alpha$ or $\beta * i \subseteq \alpha$ for $j < i$, can only enumerate finitely many demands. Consider one such demand $(\tau, j, \eta_0, \eta_1)$, issued by $M_{\beta_0}^j$ say. If $\tau \not\subset B$, $j \notin A$ or $\eta_0 \not\subset M$ then at all sufficiently late stages this demand is not implemented (if implemented at stage s then $\eta_0 \subset \mu_s$). On the other hand, for any stage s at which $\tau \subseteq B_s$, $j \in A_s$ and $\eta_0 \subseteq \mu_s$, the only way in which the demand could fail to be implemented (we are not concerned with injury) would be the implementation of a demand of higher priority $(\tau', k, \eta_2, \eta_3)$, such that $\eta_2 \subset \eta_0$ and $\eta_3 \supset \eta_0$. When two distinct trees S_α and $S_{\alpha'}$ are not nested (i.e. when it is not the case that $\alpha * \infty \subset \alpha'$ or $\alpha' * \infty \subset \alpha$), initialisation means that all of the strings in one of the trees are of strictly greater length than all strings in the other. Let β_1 be the node which issued the demand $(\tau', k, \eta_2, \eta_3)$. Since $\eta_2 \subset \eta_0$ and $\eta_3 \supset \eta_0$, it must be that $S_{\beta_0^*}$ and $S_{\beta_1^*}$ are nested. Since β_1 is of higher priority, $S_{\beta_0^*}$ must either be equal to $S_{\beta_1^*}$ or else built as a subtree of it. This contradicts the condition $\eta_2 \subset \eta_0$ and $\eta_3 \supset \eta_0$, given that η_3 is a successor η_2 in $S_{\beta_1^*}$. So for each member of this finite set

of demands there is either a stage after which they are always implemented, or else a stage after which they are never implemented. Thus α_n is initialised only finitely many times.

Now suppose that α_n is of length $2e+1$. We wish to show that any demand $(\tau, i, \eta_0, \eta_1)$ issued by α_n subsequent to its final initialisation is *met*, i.e. if $\tau \subset B$, $i \in A$ and $\eta_0 \subset M$, then $\eta_1 \subset M$ (under these conditions there is a stage, in other words, after which the demand is always implemented and not injured). The argument above, that α_n only satisfies (d) of the conditions for initialisation at finitely many stages, suffices to show that at any stage at which $\tau \subseteq B_s$, $i \in A_s$ and $\eta_0 \subset M$, the demand is implemented. In order for the demand to be injured we would then have to implement another demand $(\tau', j, \eta_2, \eta_3)$ of higher priority, at a later step of the iteration for phase 2 of that stage, for which $\eta_0 \subset \eta_2 \subset \eta_1$. Initialisation means that β which issued this demand, cannot satisfy $\beta <_L \alpha_n$ (since α_n chooses z_{α_n} large). In fact $\beta * k \subset \alpha_n$ for some $k \in \mathbb{N}$ with $k \leq j$. The finite length of η_1 also means that there are only finitely many possible values for j, and in order for any such demand to be implemented, it must be the case that the demand is issued at a stage prior to one at which j is enumerated into A. Thus there can only be issued finitely many demands $(\tau', j, \eta_2, \eta_3)$ of the correct form to cause injury to the demand $(\tau, i, \eta_0, \eta_1)$. The fact that the injuring demand is issued by M^j_β and $\beta * k \subset \alpha_n$ for some $k \in \mathbb{N}$ with $k \leq j$, means that there there is a stage s such that for all $s' \geq s$, $\tau' \not\subset B_s$ and the potentially injuring demand is not implemented.

Now if α_n has outcome f at all sufficiently large stages at which it is visited, or else there exists some least m such that α_n has outcome m at infinitely many stages, then it is clear that $\mathcal{P}_e$ is satisfied. So suppose this does not hold. Then $\Phi_e^B = M$. For each $i \notin A$, there exists $\tau \subset B$ enumerated into $\Psi_{\alpha_n}(i)$. If $i \in A$, then for any $\tau \subset B$ enumerated into $\Psi_{\alpha_n}(i)$, there is a demand issued $(\tau, i, \eta_0, \eta_1)$, such that $\eta_0 \subset \Phi_e^\tau$ and η_1 is incomparable with Φ_e^τ. Since $\Phi_e^B = M$, we have $\eta_0 \subset M$, and there is a stage after which this demand is always implemented and not injured, giving the required contradiction.

Finally, suppose that α_n is assigned the requirement $\mathcal{M}_e$. Our task is to show that, subsequent to the last initialisation of α_n, once S_α is non-empty, μ_s extends a leaf of S_α at every stage at which α_n is visited. Once we have achieved this, entirely standard arguments suffice to give (3) for the induction step. Let s_0 be the first stage at which α_n is visited subsequent to its last initialisation. Let $s_1 > s_0$ be the stage at which we define $S_{\alpha_n}(\emptyset)$. Then at every subsequent stage $s \geq s_1$ at which α_n is visited, $S_{\alpha_n}(\emptyset) \subseteq \mu_s$ and the implemented demands are precisely those which were implemented at stage s_1, together with possibly extra demands issued by nodes properly extending α_n on the construction tree, which are of the form $(\tau, 1, \eta_0, \eta_1)$ for η_0 and η_1 in S_{α_n}. □

References

[1] S.B. Cooper, *Computability Theory*, Chapman & Hall, 2004.

[2] S.B. Cooper, Some negative results on minimal degrees below $\mathbf{0}'$, *Recursive Function Theory Newsletter*, 34, item 353.

[3] S.B. Cooper, Local degree theory, *Handbook of Computability Theory*, Elsevier 1999.

[4] S.B. Cooper, The strong anticupping property for recursively enumerable degrees, *Journal of Symbolic Logic*, vol. 54, 527-539, 1989.

[5] S.B. Cooper, R. Epstein, Complementing below recursively enumerable degrees, *Annals of Pure and Applied Logic*, 34, 7–27, 1987.

[6] R. Epstein, Degrees of unsolvability, structure and theory, *Lecture Notes in Mathematics*, 759, Springer-Verlag, Berlin, 1979.

[7] R. Epstein, Initial segments of degrees below $\mathbf{0}'$, Memoirs of the American Mathematical Society, no 241, 1981.

[8] S. Ishmukhametov, On a problem of Cooper and Epstein, *Journal of Symbolic Logic*, 68, no 1, 52–64, 2003.

[9] S.C. Kleene, E.L. Post, The uppersemilattice of degrees of recursive unsolvability, *Annals of Mathematics*, 59, 379–407, 1054.

[10] A.E.M. Lewis, The search for natural definability in the Turing degrees, submitted.

[11] A.E.M. Lewis, Properties of the jump classes, *Journal of Logic and Computation*, 22 (4): 845-855, 2012.
[12] A.E.M. Lewis, Minimal complements for degrees below 0', *Journal of Symbolic Logic*, 69 (4), 937-966, 2004.
[13] A. Li, D. Yang, Bounding minimal degrees by computably enumerable degrees, *Journal of Symbolic Logic*, 63, 1319–1347, 1998.
[14] T.A. Slaman, J.R. Steel, Complementation in the Turing degrees, *Journal of Symbolic Logic*, vol. 54 no. 1, 160–176, 1989.
[15] R.I. Soare, *Recursively enumerable sets and degrees*, Springer, New York, (1987).
[16] A.M. Turing, Systems of logic based on ordinals, *Proceedings of the London Mathematical Society*, 45, 161–228, 1939.

School of Mathematics, University of Leeds, Leeds, UK

Department of Mathematics, London School of Economics, London, UK
URL: `aemlewis.co.uk`
E-mail address: `andy@aemlewis.co.uk`

School of Physical & Mathematical Sciences, Nanyang Technological University, 21 Nanyang Link, Singapore
E-mail address: `kmng@ntu.edu.sg`

School of Mathematics, University of Leeds, Leeds, UK