# A transfinite hierarchy of lowness notions in the computably enumerable degrees, unifying classes, and natural definability

Rod Downey

Noam Greenberg

Author address:

School of Mathematics, Statistics and Operations Research, Victoria University, P.O. Box 600, Wellington, New Zealand
*E-mail address*: Rod.Downey@vuw.ac.nz

School of Mathematics, Statistics and Operations Research, Victoria University, P.O. Box 600, Wellington, New Zealand
*E-mail address*: greenberg@msor.vuw.ac.nz

November 26, 2015

# Contents

CHAPTER I

# Introduction

The conceptual basis of this monograph is centered in a long-term programme in computability theory, seeking to understand the relationship between dynamic properties of sets and functions and their algorithmic complexity. In this paper, along with the companion papers [17] and [15], we introduce a new hierarchy of computably enumerable (c.e.) Turing degrees based on the complexity of approximations of functions in these degrees. Since all such functions are $\Delta^0_2$, by the Limit Lemma they have approximations $g = \lim_s g_s$ with $\langle g_s \rangle$ uniformly computable. The idea is to classify the degrees according to the complexity of a bound on the "mind change" function $\#\{s : g_{s+1}(x) \neq g_s(x)\}$. We will use a classification of $\Delta^0_2$ functions defined by Ershov in [28, 29, 30].

The reader might well ask why we need yet another hierarchy in computability theory. Below we discuss three aspects of this work.

(i) New natural definability results in the c.e. degrees. These definability results are in the $\mathrm{low}_2$ degrees and hence are not covered by the current metatheorems of Nies, Shore and Slaman [52]. Moreover they are amongst the very few natural definability results in the theory of the c.e. Turing degrees.

(ii) A new methodology for classifying and unifying the combinatorics of a number of constructions from the literature.

(iii) The introduction of a number of construction techniques which are injury-free and highly non-uniform. These would seem to have wider applications.

**Unifying constructions.** It is not common, in computability theory, to find a class of degrees which captures the underlying dynamics of a number of apparently similar constructions. A good example is the class of high degrees, which arise from dense simple, maximal, hyperhypersimple, and other similar kinds of c.e. set constructions (Martin [48]). Another example would be the class of the promptly simple degrees (Ambos-Spies, Jockusch, Shore and Soare [2]). A more recent example of current interest is the class of $K$-trivial degrees (see for example [22, 50, 51]), which have several characterisations arising from lowness constructions.

The example most relevant to the current work is the class of array computable degrees, defined by Downey, Jockusch and Stob [23, 24]. A c.e. Turing degree $\mathbf{a}$ is array computable if every function $g \in \mathbf{a}$ has a computable approximation $\langle g_s \rangle$ such that for all $n$ there are at most $n$ many stages $s$ such that $g_{s+1}(n) \neq g_s(n)$ (in other words, the mind-change function is bounded by the identity function). The array computable degrees capture the combinatorics of a wide class of constructions. To wit, we observe that a c.e. degree is array noncomputable if and only if...

(1) it is the degree of a perfect thin $\Pi^0_1$ class (Cholak, Coles, Downey and Herrmann [7]);
(2) it bounds a disjoint pair of c.e. sets which have no separator computing $\varnothing'$ (Downey, Jockusch, Stob [23]);
(3) it contains a c.e. set with maximal Kolmogorov complexity (Kummer [41]);
(4) it does not have a strong minimal cover in the Turing degrees (Ishmukhametov [37]);
(5) it has effective packing dimension 1 (Downey and Greenberg [16]);
(6) it contains two left-c.e. reals with no common upper bound in the cl-degrees of left-c.e. reals (Barmpalias, Downey and Greenberg [5]);
(7) it contains a set which is not reducible to the halting problem with tiny use (Franklin, Greenberg, Stephan and Wu [32]).

The dynamics captured by classes of degrees are often phrased in terms of permitting. The relative strength of each class is reflected, roughly, in the amount of permitting that can be expected of its members, and sometimes its timing. In simple permitting (given by any noncomputable c.e. degree) each requirement ought to be satisfied by a single permission. Prompt permission is similar, except that permission has to be given essentially immediately when asked. High permitting allows a requirement to ask for infinitely many permissions, and all but finitely many requests are granted — this is "co-finite permitting". Array noncomputable permitting, originally called "multiple permitting", is an intermediate version, in which for each attempt at meeting a requirement, a number of required permissions is stated in advance. The connection with the complexity of approximations of functions in the degree is direct: mind-changes essentially correspond to instances of permission; the computable bound on the number of mind-changes is the same bound on the number of permissions required to meet a requirement. The remarkable fact is that in many cases it is shown that the level of permitting is not only sufficient but also necessary for the construction to succeed.

In 2005, J. Miller (unpublished) defined a non-uniform version of the class of array computable degrees. We call a function *$\omega$-computably approximable* ($\omega$-c.a.) if it has a computable approximation whose mind-change function is bounded by some computable function. This is equivalent to the function being weak truth-table reducible to $\varnothing'$. The notion is widely use in computability, with applications in algorithmic randomness as well (for example in [33, 36, 35, 31]).

Definition 0.1. A c.e. degree is *totally $\omega$-c.a.* if every function in it is $\omega$-c.a.

Array computability is in some sense a uniform version of this notion: it requires the same bound for all functions in the degree. The associated permission notion is stronger than array noncomputable permitting in that the number of permissions required for the each attempt to meet a requirement is stated during the construction, not necessarily in advance (and thus can, for example, take into consideration the stage number at which the attempt is started).

In [17], the authors, with R. Weber, showed that the class of totally $\omega$-c.a. degrees indeed captures the dynamics of some constructions. In this work we extend this by showing another equivalence, characterising the dynamics of an existing construction. In [25], Downey and LaForte constructed a noncomputable left-c.e. real $\alpha$, all of whose c.e. presentations are computable.

Theorem 0.2.

(1) *If a c.e. degree* $\mathbf{d}$ *is not totally* $\omega$*-c.a. then there is a left-c.e. real* $\alpha \leqslant_{\mathrm{T}} \mathbf{d}$ *and a c.e. set* $B <_{\mathrm{T}} \alpha$ *such that every presentation of* $\alpha$ *is* $B$*-computable.*
(2) *If a left-c.e. real* $\alpha$ *has a totally* $\omega$*-c.a. degree then there is a presentation of* $\alpha$ *which is Turing equivalent to* $\alpha$*.*

For the definitions and more details see Chapter V.

After our results were announced, Barmpalias, Downey and Greenberg [5] obtained yet another constructions whose dynamics are captured by this class. Their results concern the interaction of Turing and weak truth-table reducibility. They showed that a c.e. degree is totally $\omega$-c.a. if and only if every set in that degree is weak truth-table reducible to a ranked set (equivalently, to a hyperimmune set, or to a proper initial segment of a computable, scattered linear ordering.) In further work, Brodhead, Downey and Ng [6] showed that the totally $\omega$-c.a. degrees capture a finite form of randomness.

**Natural definability and lattice embeddings.** Shore [56] articulated the difference between naturally defined classes of degrees, definitions which are structural in nature; and classes defined by external means, usually by coding models of arithmetic. Natural definitions in degree theory are few. In contrast, there has been significant success in obtaining general, abstract definability results in the c.e. degrees, culminating in the work of Nies, Shore and Slaman.

Theorem 0.3 (Nies, Shore, Slaman [52]). *Any relation on the c.e. degrees which is invariant under the double jump is definable in the c.e. degrees if and only if it is definable in first order arithmetic.*

The proof of Theorem 0.3 involves interpreting the standard model of arithmetic in the structure of the c.e. degrees without parameters, and obtaining a definable map from degrees to indices (in the model) which preserves the double jump. The result gives a definition of a large collection of classes of degrees (for example all jump classes $\mathrm{High}_n$ and $\mathrm{Low}_n$, the latter for $n \geqslant 2$).

Theorem 0.3 has two shortcomings. One is the reliance on the invariance of the relation under the double jump. It follows that no collection of c.e. degrees that contains some, but not all, $\mathrm{low}_2$ degrees, can be defined using the theorem; these are the kinds of collections that we investigate here.

Another issue, as mentioned, is that the definitions provided by the theorem are not natural. Examples of natural definitions are:

- A c.e. degree is promptly simple if and only if it is not cappable (Ambos-Spies, Jockusch, Shore, and Soare [2]).
- A c.e. degree is contiguous if and only if it is locally distributive (Downey and Lempp [26]) if and only if it is not the tope of the pentagon (the non-modular, 5 element lattice $N_5$) (Ambos-Spies and Fejer [1]).
- A c.e. truth table degree is $\mathrm{low}_2$ if and only if it has no minimal cover in the c.e. truth table degrees (Downey and Shore [18]).

Natural definitions are closely related to embeddings of finite lattices into the c.e. degrees; see for example [44, 45, 47]. Central to lattice embeddings is the notion of a critical triple (see [14, 62]): incomparable elements $\mathbf{a}_0, \mathbf{a}_1, \mathbf{b}$ of an upper semi-lattice such that $\mathbf{a}_0 \vee \mathbf{b} = \mathbf{a}_1 \vee \mathbf{b}$ but $\mathbf{a}_0 \wedge \mathbf{a}_1 \leqslant \mathbf{b}$ (in the sense that any element below both $\mathbf{a}_0$ and $\mathbf{a}_1$ is also below $\mathbf{b}$). The authors, with R. Weber, showed:

THEOREM 0.4 ([17]). *A c.e. degree bounds a critical triple in the c.e. Turing degrees if and only if it is not totally $\omega$-c.a.*

All distributive lattices can be embedded into the c.e. degrees. The basic non-distributive lattices are the pentagon, and the 1-3-1 lattice (Figure 1), also known as $M_5$. The three middle elements of the 1-3-1 lattice form a critical triple.

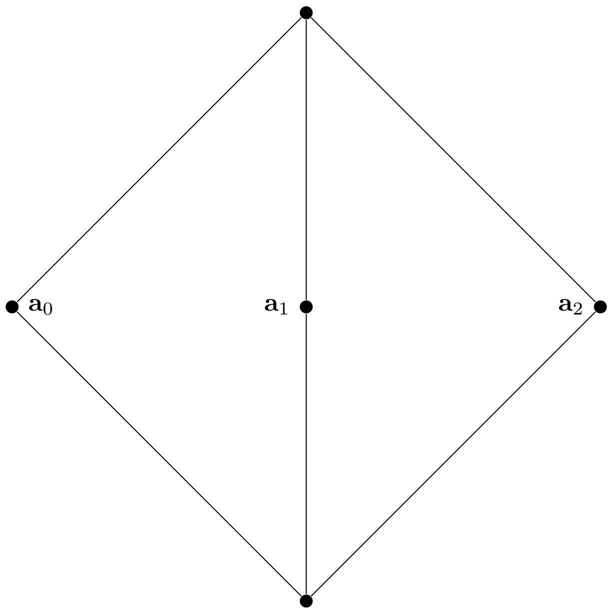


FIGURE 1. The 1-3-1 lattice

For a while it appeared that the critical triple precisely captures the need for "continuous tracing" which is used in an embedding of the 1-3-1 lattice into the c.e. degrees (Lachlan [42]). The first nonembeddability result was by Lachlan and Soare ([43], suggested by Lerman) who demonstrated that an "infimum into a 1-3-1" could not be embedded in the c.e. degrees (namely the lattice $S_8$, which consists of a diamond above a 1-3-1, cannot be embedded into the c.e. degrees). The necessity of continuous tracing was further demonstrated by Downey [14] and Weinstein [62] who showed that there are initial segments of the c.e. degrees where no lattice with a (weak) critical triple can be embedded. It was also noted in [14] that the embedding of critical triples seemed to be tied up with multiple permitting in a way that was similar to non-low$_2$-ness. Indeed this intuition was to some extent verified by Downey and Shore [19], who showed that every non-low$_2$ c.e. degree bounds a copy of the 1-3-1 lattice in the c.e. degrees.

The notion of non-low$_2$-ness seemed too strong to capture the class of degrees which bound a copy of the 1-3-1 lattice, but it was felt that something like that should suffice. On the other hand, Walk [61] constructed an array noncomputable c.e. degree bounding no weak critical triple, and hence it was already known that array noncomputability was not enough for such embeddings. Theorem 0.4 completely determines what it takes to embed a critical triple into the c.e. degrees, and gives a natural definition of the class of totally $\omega$-c.a. degrees. We remark that a definition of array computability is still not known. We also remark that in this work (Chapter IV) we show that there are maximal totally $\omega$-c.a. degrees. These maximal degrees form a naturally definable antichain in the c.e. degrees. The only other known naturally definable antichain consists of the maximal contiguous degrees (Cholak, Downey and Walk [9]).

It turns out though that embedding the 1-3-1 lattice requires more strength than embedding a critical triple. Informally, a critical triple embedding can be done on a tree. The basic idea is the following. We are trying to meet a requirement toward showing that the middle degree $\mathbf{b}$ is not above say $\mathbf{a}_0$. While we wait for a follower to be realised, an "entourage" of traces (alternating between $\mathbf{a}_0$ and $\mathbf{a}_1$) keeps getting extended. When the follower is realised, a final trace targeted for $\mathbf{b}$ is appended at the end of the entourage. Henceforth, at every stage at which the strategy is accessible (all resraints are dropped simultaneously) we can enumerate the two last traces in the entourage into $\mathbf{a}_i$ and $\mathbf{b}$ (and appoint a new $\mathbf{b}$-trace at the end). The main point is that stronger negative requirement do not allow enumerations into both $\mathbf{a}_0$ and $\mathbf{a}_1$ at the same stage; but they have no problems with enumerations into both some $\mathbf{a}_i$ and $\mathbf{b}$. We see that when the follower is realised we know how many times the strategy needs to act until it reaches the follower at the head of the entourage and meets the positive requirement. This is precisely the level of non-total $\omega$-c.a. permitting (whereas as we mentioned above, array noncomputable is insufficient because we need to wait until the follower is realised in order to state how many permissions we need; there is no uniform bound that would work for all requirements).

The dynamics of the 1-3-1 embedding are more complicated. There we build $\mathbf{a}_0$, $\mathbf{a}_1$ and $\mathbf{a}_2$; restraining a positive requirement are several negative requirements, and for each pair $\mathbf{a}_i$ and $\mathbf{a}_j$ there will be a negative requirement which does not allow enumerations into both sets at the same stage. To overcome this serious restriction, we work with a pinball machine. The idea is to recreate the tracing phenomenon at each gate, locally retargeting the rest of the entourage to a pair $\mathbf{a}_i$ and $\mathbf{a}_j$ which is not restrained by that gate. When the gate opens this last segment of the entourage drops to the next unoccupied gate below. This new gate may very well restrain the pair $\mathbf{a}_i$ and $\mathbf{a}_j$. One by one we peel the end of this segment of the entourage, retarget, build a new end for the entourage, and repeat. Even with two gates below a positive requirement, we see that even when we see the follower realised, we cannot tell how many times the requirement needs attention: we know how many times we will need to drop an entourage segment from the top gate; each time that happens, and not earlier, we will find out how many times the segment now at the lower gate will need to drop below (and get enumerated into sets).

What is needed, roughly, to pass two gates, is permission at the level of $\omega^2$ approximations. Below we define and study the hierarchy of totally $\alpha$-c.a. degrees for many computable ordinals $\alpha$. This is based on Ershov's generalisation of $\omega$-computable approximability using other ordinals (but we need to restrict ourselves to particularly nice computable copies of these ordinals). However to state our main result here we can give an equivalent, inductive definition of the first $\omega$ layers:

DEFINITION 0.5. Let $n < \omega$. A function is $\omega^{n+1}$-*computable approximable* if it has a computable approximation whose associated mind-change function is bounded by an $\omega^n$-c.a. function.

A definition analogous to being weak truth-table reducible to $\varnothing'$ is also available and discussed in Chapter II below. Equipped with this hierarchy, we define:

DEFINITION 0.6. A c.e. degree is *totally* $<\omega^\omega$*-c.a.* if every function in it is $\omega^n$-c.a. for some $n$.

The location of this level of the hierarchy (with respect to levels such as totally $\omega^n$-c.a. degrees) is discussed in Chapter III. But it is definitely the case that there are degrees which are totally $<\omega^\omega$-c.a. but not totally $\omega$-c.a. Our main result is:

THEOREM 0.7. *A c.e. degree bounds a copy of the 1-3-1 lattice if and only if it is not totally* $<\omega^\omega$*-c.a.*

Thus our hierarchy pinpoints and separates between the complexity of embedding a critical triple and embedding the 1-3-1 lattice. And conversely, Theorem 0.7 gives a natural definition of another level of our hierarchy.

We remark that just like the totally $\omega$-c.a. degrees, this new class of totally $<\omega^\omega$-c.a. degrees serves to unify constructions which all share similar dynamics. In Chapter VI we show that an $m$-topped degree cannot be totally $<\omega^\omega$-c.a. (a sketch appeared in [15]). This is sharp as $m$-topped degrees appear in the next level of our hierarchy, the class of totally $\omega^\omega$-c.a. degrees [15]. (This cannot give a complete classification since $m$-topped degrees cannot be low, but each level of our hierarchy contained both low and non-low degrees.) More recently Day [11] related this level of the hierarchy to computing indifferent sets for 1-genericity. We are convinced that other constructions are related, for example the construction of a set of c.e. Turing degree computable from a set of minimal weak truth-table degree (Downey, Ng and Solomon, in preparation). However the complexity of this level of permitting is also reflected in the complexity of the arguments involved, so it may be difficult to flesh out the details.

We also remark that the negative direction of Theorem 0.7 (as well as the $m$-topped result) uses an "anti-permitting" technique which we believe is quite novel. It is an injury-free construction, however it is highly non-uniform. We believe that this technique may be of use elsewhere as well.

**Promptness.** One can ask, regarding the embedding of the 1-3-1 lattice, what it would take to get an embedding whose bottom degree is $\mathbf{0}$ (as is obtained in Lachlan's original construction). We discuss this in Chapter VIII, where we introduce prompt versions of all levels in our hierarchy. This generalises the already familiar notion of prompt permitting, which is the prompt version of simple permitting. Prompt array noncomputable permission, for example, allows us to construct a pair of separating classes whose elements form minimal pairs (Theorem 2.1); whereas traditional (non-prompt) array noncomputable permission only gives Turing incomparability [23]. Similalry, a degree which is promtply not totally $<\omega^\omega$-c.a. bounds a copy of the 1-3-1 lattice with bottom $\mathbf{0}$.

This however cannot be reversed: every high degree bounds a copy of the 1-3-1 lattice with bottom $\mathbf{0}$, and there are high degrees which are not promtply simple (let alone promptly non totally $<\omega^\omega$-c.a.) Informally what this says is that there are at least two ways to get such an embedding: either by quickly getting the precise number of permissions required; or by getting many permissions (cofinitely), in which case we can wait for the permissions and don't need them promptly.

It would be interesting to find a common generalisation.

**An application to admissible computability.** Combined with results of the second author, our work has an application to admissible computability. This is a generalisation of traditional computability to ordinals beyond $\omega$. In [34] it is shown that for any admissible ordinal $\alpha$, the $\alpha$-c.e. degrees are not elementarily equivalent to the c.e. degrees. This was done in cases, depending on the proximity of $\alpha$ to $\omega$. In one case the separation between the theories is not natural but relies on coding models of arithmetic. However one result is:

THEOREM 0.8 ([34]). *Let $\alpha > \omega$ be an admissible ordinal, and let* **a** *be an incomplete $\alpha$-c.e. degree. The following are equivalent:*

(1) **a** *computes a cofinal $\omega$-sequence in $\alpha$.*
(2) **a** *bounds a copy of the 1-3-1 lattice.*
(3) **a** *bounds a critical triple.*

Again, it is the analysis of continuous tracing that underlies this result. The basic idea is the following. Consider again the embedding of a critical triple: as time goes by, a longer and longer entourage is build for a follower. When the follower is realised, the entourage is peeled back (from the end to the beginning), one member at a time. Trying to do this when time goes beyond $\omega$ presents a completely new problem: after $\omega$ many stages, we will have an entourage of order-type $\omega$, that is, one without a last element. We cannot then peel it back, each step removing only the last element. It turns out that this blockage is fundamental. The only case it might be possible for a degree **a** to bound a copy of the 1-3-1 lattice is if it itself can see that $\alpha$ is far from being a regular cardinal — if it can essentially re-order time and space to order-type $\omega$, so that the construction can be (at least after the fact) seen to have taken $\omega$ steps, avoiding infinite entourages. In one direction, effective closed and unbounded sets are used to show that this is necessary. In the other direction, a fine-structural result of Shore's [55] says that an incomplete degree of computable cofinality $\omega$ must be high, and can compute a bijection between $\alpha$ and $\omega$. Working below such a degree, we can translate back to $\omega$-computability, and use non-low$_2$ permitting to embed the 1-3-1 lattice (for a technical reason, we cannot quite use high permitting).

To sum, what this says is that once we go beyond $\omega$, the fine distinctions between totally $\omega$-c.a. degrees and totally $< \omega^\omega$-c.a. degrees completely disappear. Combined with the current work, this gives us a single, natural sentence which separates the elemntary theory of the c.e. degrees from the theory of the $\alpha$-c.a. degrees for any admissible $\alpha > \omega$.

THEOREM 0.9. *Let $\alpha \geqslant \omega$ be admissible. The following are equivalent:*

(1) *There is an incomplete $\alpha$-c.e. degree which bounds a critical triple but not the 1-3-1 lattice.*
(2) $\alpha = \omega$.

## 1. Notation and general definitions

**1.1. Computable approximations and enumerations.** A *computable approximation* for a function $f\colon \omega \to \omega$ is a uniformly computable sequence $\langle f_s \rangle_{s<\omega}$ of functions such that for all $x$, for almost all $s$, $f_s(x) = f(x)$. In other words, $f = \lim_s f_s$ when we equip $\omega$ with the discrete topology. Shoenfield's limit lemma [54] states that a function $f$ is $\Delta^0_2$-definable if and only it is computable from the halting set $\varnothing'$ if and only if it has a computable approximation. If $A$ is a *set* (a subset of $\omega$, identified with an element of Cantor space) then a computable approximation of $A$ is a sequence of sets.

A *computable enumeration* of a c.e. set $A$ is a computable, $\subseteq$-increasing sequence of finite sets $\langle A_s \rangle$ such that $A = \bigcup_s A_s$. We can also think of a computable enumeration as a computable approximation of $A$, again by taking characteristic functions. We say that a number $x$ is *enumerated into* $A_s$ if $x \in A_s \backslash A_{s-1}$.

**1.2. Turing functionals.** A (Turing) *functional* is a c.e. set of triples $\langle \sigma, x, y \rangle$ consisting of a finite sequence $\sigma$ of natural numbers and a pair of natural numbers $x$ and $y$. We consider such triples as *axioms*, and sometimes write them as $\sigma \mapsto (x, y)$. If $f\colon \omega \to \omega$ and $\Phi$ is a functional, then we define the multi-valued function (i.e., relation) $\Phi(f) \subseteq \omega \times \omega$ by letting $\Phi(f, x) = y$ if there is some finite $\sigma \prec f$ such that the axiom $\sigma \mapsto (x, y)$ is in $\Phi$. We write $\Phi(f, x)\downarrow$ for $x \in \operatorname{dom} \Phi(f)$ and $\Phi(f, x)\uparrow$ for $x \notin \operatorname{dom} \Phi(f)$.

In general we allow functionals, especially the ones that we build, to be *inconsistent*. That is, we allow them to contain contradictory axioms: a pair of axioms $\sigma \mapsto (x, y)$ and $\tau \mapsto (z, w)$ such that $\sigma$ and $\tau$ are comparable (that means that $\sigma \preccurlyeq \tau$ or $\tau \preccurlyeq \sigma$), $x = z$ but $y \neq w$. A functional $\Phi$ is called *consistent relative to an oracle* $f$ if $\Phi(f)$ is a partial function, i.e., is not multi-valued. A functional is consistent if and only if it is consistent relative to every oracle.

The following are equivalent for $f, g\colon \omega \to \omega$:

(1) there is a consistent functional $\Phi$ such that $\Phi(f) = g$;
(2) there is a functional $\Phi$, consistent relative to $f$, such that $\Phi(f) = g$;
(3) $g \leqslant_{\mathrm{T}} f$.

If $\langle \Phi_s \rangle$ is a computable enumeration of a functional $\Phi$, then each $\Phi_s$ is also a functional. If $\langle f_s \rangle$ is a computable approximation of a function $f\colon \omega \to \omega$, then the finite multi-valued function $\Phi_s(f_s)$ can be effectively obtained from $s$. If for all $s$, $\Phi_s$ is consistent relative to $f_s$, then $\Phi$ is consistent relative to $f$. Note that if further, $\Phi(f)$ is a total function, then we can extend $\langle \Phi_s(f_s) \rangle$ to a computable approximation of $\Phi(f)$, since $\langle \operatorname{dom} \Phi_s(f_s) \rangle$ is uniformly computable. When the notation $\Phi_s(f_s)$ becomes unwieldy, we sometimes write $\Phi(f)[s]$, and in general may use Lachlan's square bracket notation.

Suppose that $\Phi$ is a functional which is consistent relative to an oracle $f$. If $x \in \operatorname{dom} \Phi(f)$, we also refer to $\Phi(f, x) = y$ as a "computation". Let $\sigma$ be the shortest initial segment of $f$ for which $\sigma \mapsto (x, y)$ is an axiom in $\Phi$. Often in fact there will be a unique such initial segment. The string $\sigma$ determines the *use* of the computation, denoted by $\varphi(f, x)$ (and when $f$ is clear from the context, by $\varphi(x)$). We will use two conflicting notions:

- If either $f$ or $\Phi$ are given, then the use is the length of $\sigma$.
- If both $f$ and $\Phi$ are built by us then we let the use be $|\sigma| - 1$, the "greatest number queried during the computation". In this case $f$ is usually a

c.e. set $A$. The idea is that we may want to void the computation by enumerating the use $\varphi(x)$ into $A$.

If $\langle \Phi_s \rangle$ is a computable enumeration of a Turing functional $\Phi$, and $\langle f_s \rangle$ is a computable approximation of a function $f$ (and again we assume that for all $s$, $\Phi_s$ is consistent relative to $f_s$), $s < \omega$ and $x \in \operatorname{dom} \Phi_s(f_s)$, then we say that the computation $\Phi_s(f_s, x)$ is *destroyed* (or *injured*) at stage $s+1$ if $\sigma \nprec f_{s+1}$, where $\sigma$ as above is the shortest axiom applying to $f$ giving the computation at stage $s$. That is, if $f_{s+1} \restriction_u \neq f_s \restriction_u$ where $u = \varphi_s(f_s, x)$ is the use of the computation, in the case in which either $f$ or $\Phi$ are given; if both are built by us, then the computation is destroyed if $f_s \restriction_{u+1} \neq f_{s+1} \restriction_{u+1}$, and as described above, this will often happen because we enumerate $u$ into $f_{s+1}$.

In contrast, we say that a computation $\Phi_s(f_s, x) = y$ is *$f$-correct* if $\sigma \prec f$. The fundamental fact about Turing computations, used without mention throughout computability theory, is that $x \in \operatorname{dom} \Phi(f)$ if and only if there is a stage $s$ (equivalently, for almost all stages $s$) such that $x \in \operatorname{dom} \Phi_s(f_s)$ by an $f$-correct computation. When working with c.e. sets we often use the fact that correct computations never go away: if $\langle A_s \rangle$ is a computable enumeration of a c.e. set $A$, and $\Phi_s(A_s, x)$ is an $A$-correct computation, then for all $t \geqslant s$, $x \in \operatorname{dom} \Phi_t(A_t)$ by the same computation.

The following lemma is used when we build functionals which apply to c.e. sets that we enumerate.

LEMMA 1.1. *Let $\langle \Phi_s \rangle$ be a computable enumeration of a functional $\Phi$, and let $\langle A_s \rangle$ be a computable enumeration of a c.e. set $A$. Suppose that for all $s$,*

(1) *if an axiom $\sigma \mapsto (x, y)$ is enumerated into $\Phi_s$, then $\sigma \prec A_s$;*
(2) *for each $x$, at most one axiom $\sigma \mapsto (x, y)$ is enumerated into $\Phi_s$.*

*Let $s < \omega$, and suppose that $\Phi_s$ is consistent for $A_s$. Suppose that for all $x < \omega$,*

(3) *If an axiom $\sigma \mapsto (x, y)$ is enumerated into $\Phi_{s+1}$, and $x \in \operatorname{dom} \Phi_s(A_s)$, then some number $u \leqslant \varphi_s(A_s, x)$ is enumerated into $A_{s+1}$.*

*Then $\Phi_{s+1}$ is consistent for $A_{s+1}$.*

Hence if conditions (1)–(3) hold at every stage $s$, then $\Phi$ is consistent for $A$. Note that usually $\Phi$ will not be consistent for all oracles: we could void a computation $\Phi_s(A_s, x)$ by enumerating $u = \varphi_s(A_s, x)$ into $A_{s+1}$, and then define a new computation $\Phi_{s+1}(A_{s+1}, x)$ with smaller use, so $\Phi_{s+1}$ may be inconsistent for $A_s$.

CONVENTION 1.2. We often assume that for a given consistent functional $\Phi$, for any oracle $f$, $\operatorname{dom} \Phi(f)$ is an initial segment of $\omega$. That is, we require that if $\sigma \mapsto (x, y)$ is in $\Phi$, then for all $x' < x$ there is some $\sigma' \preccurlyeq \sigma$ and some $y'$ such that $\sigma' \mapsto (x', y')$ is also in $\Phi$. We simply prevent $\sigma \mapsto (x, y)$ from entering $\Phi$ until we see the other necessary axioms.

In this situation we also assume that if $\langle \Phi_s \rangle$ is a computable enumeration of a Turing functional $\Phi$, then for all $s$ and $f$, $\operatorname{dom} \Phi_s(f)$ is an initial segment of $\omega$.

The point is that if we are only interested in *total* functions computable from an oracle $f$, then we can restrict ourselves to functionals of the type described.

We let $\langle \Phi_e \rangle$ be some enumeration of all *consistent* functionals; associated with which we are given uniformly computable enumerations $\langle \Phi_{e,s} \rangle$ of $\Phi_e$.

CONVENTION 1.3. We sometimes identify natural numbers with the von-Neumann ordinals isomorphic to them; that is, we identify the natural number

$n$ with the set $\{0, 1, 2, \ldots, n-1\}$. In particular, if for some functional $\Phi$ and oracle $f$, $\operatorname{dom}\Phi(f)$ is an initial segment of $\omega$ (per Convention 1.2), then we write $x < \operatorname{dom}\Phi(f)$ for $x \in \operatorname{dom}\Phi(f)$, and $x \leqslant \operatorname{dom}\Phi(f)$ for $\{0, 1, \ldots, x-1\} \subseteq \operatorname{dom}\Phi(f)$.

Functionals which take more than one oracle are treated in a similar fashion. For example, when taking two oracles, axioms will be of the form $(\sigma, \tau) \mapsto (x, y)$. Usually, for a pair of oracles $f, g$ in which we are interested, for each $x$ there will be at most one pair of strings $\sigma \prec f$ and $\tau \prec g$ such that $(\sigma, \tau) \mapsto (x, y)$ is in the functional $\Phi$ we are building or examining. These determine the $f$-*use* and the $g$-*use* of the computation $\Phi(f, g, x)$, according to the notational convention discussed above. When $\Phi$ is not built by us we often assume that the $f$-use and the $g$-use are the same, and that common value is referred to simply as the use $\varphi(f, g, x)$ of the computation.

**1.3. Priority arguments and tree constructions.** In our constructions we keep the *convention of small numbers*.

CONVENTION 1.4. At stage $s$ of a construction, all numbers played by the "opponent" are bounded by $s$. These are the values of functions that are not defined by us during the construction.

On the other hand, the constructions would often call on us to define new values for functions that are *large*. This means that the new values are picked to be numbers that are larger than any other number previously used or observed in the construction, including the stage number.

*Tree constructions*, namely priority constructions done with the aid of a *tree of strategies*, are now standard; a reference is Chapter XIV of [57]. Elements of the tree are called *strategies*, *agents* or *nodes*; these are finite sequences of symbols. To describe the tree of strategies, we give two pieces of information:

(a) An association of requirements for nodes; we say that a node *works for* the requirement associated with it. Often, but not always, all nodes of a given level of the tree work for the same requirement.
(b) For nodes working for some requirement, the list of *outcomes* of these nodes.

The tree is then defined recursively. The empty node is always on the tree of strategies; if a node $\sigma$ has already been determined to lie on the tree of strategies, and a requirement $R$ has been associated with it, then the immediate successors of $\sigma$ on the tree are the nodes of the form $\sigma\hat{\ }o$, where $o$ is a possible outcome for nodes working for $R$.

The collection of possible outcomes of any node will be linearly ordered; we say that an outcome $o$ is *stronger* than an outcome $o'$ if $o < o'$. This ordering induces a linear ordering of the tree of strategies, by taking a lexicographic amalgamation of the orderings of outcomes: $\sigma < \tau$ if $\sigma \prec \tau$, or if there are $\eta$, $o$ and $o'$ such that $\sigma \succcurlyeq \eta\hat{\ }o$, $\tau \succcurlyeq \eta\hat{\ }o'$, and $o < o'$. We say that a node $\sigma$ is *stronger* than a node $\tau$ if $\sigma < \tau$, and that a node $\sigma$ *lies to the left* of a node $\tau$ if $\sigma < \tau$ but $\sigma \not\preccurlyeq \tau$. We sometimes write $\sigma <_L \tau$; this has nothing to do with the constructible hierarchy.

At any stage $s$, the construction describes the (finite) collection $\delta_s$ of nodes that are *accessible* at stage $s$. This will always be linearly ordered by extension of nodes, and usually also be an initial segment of the tree of strategies; the exception is when the construction equips the tree with *links* which are travelled, skipping some nodes between two accessible nodes. Usually, the empty node $\langle\rangle$ is accessible.

We then say that a node $\sigma$ lies on the *true path* $\delta_\omega$ if there are infinitely many stages $s$ of the construction such that $\sigma \in \delta_s$ (that is, such that $\sigma$ is accessible at stage $s$), but the same is not true for any node $\tau$ that lies to the left of $\sigma$. The true path $\delta_\omega$ will be linearly ordered by node extension $\preccurlyeq$. In practice, if every $\delta_s$ is an initial segment of the tree, then so is $\delta_\omega$. We will need to prove that the true path is infinite, and contains, for every requirement $R$, a node working for $R$; the latter part will be immediate if $\delta_\omega$ is an infinite initial segment of the tree, that is, an infinite *path* of strategies.

Most constructions will employ a notion of *initialisation* of nodes on the tree of strategies. This would usually mean that when a node is initialised, all parameters associated with the node (such as followers) are removed (or *cancelled*), and new ones will have to be defined, either immediately, or more often, at the next stage at which the node is accessible. When a stage ends, every node which lies to the right of an accessible node (a node in $\delta_s$) is initialised. Often, but not always, nodes extending the longest node in $\delta_s$ are also initialised at the end of stage $s$. We ensure that whenever a node $\sigma$ is initialised, and $\tau$ is a node weaker than $\sigma$, then $\tau$ is also initialised at the same time.

We say that the construction is *fair* to a node $\sigma$ if $\sigma$ is initialised only finitely many times (i.e., at only finitely many stages of the construction). The main *fairness lemma* for each construction will state that the construction is fair to every node on the true path $\delta_\omega$. If $\sigma$ is a node on the true path and the construction is not fair to $\sigma$ then there will be some node $\tau \prec \sigma$ on the true path which initialises $\sigma$ at infinitely many stages. This is because initialisation has to respect the priority ordering; no node weaker than $\sigma$ can initialise $\sigma$.

Other standard conventions of priority constructions are employed without mention. For example, we use "stickiness" or "persistence" of parameters: if, for example, a requirement $R$ or strategy $\sigma$ has a "follower" (a witness with which to meet the requirement) at some stage $s$, and the requirement or node is not tampered with (e.g., initialised) at stage $s+1$, say, then that follower is still considered to be a follower for the requirement or strategy at stage $s+1$.

CHAPTER II

# $\alpha$-c.a. functions

Ershov [29] extended the hierarchy of differences of c.e. sets into the transfinite, based on Kleene's notations for computable ordinals. Unfortunately, the levels of this hierarchy depend heavily on the choice of notation. To get around this problem, based on ideas from [10], we focus on lower levels of the hierarchy, using canonical well-orderings. We then, extending [3], relate these lower, canonical levels, to iterations of a jump in the weak truth-table degrees.

## 1. $\mathcal{R}$-c.a. functions

Let $\mathcal{R} = (R, <_{\mathcal{R}})$ be a computable well-ordering of a computable set $R$. An *$\mathcal{R}$-computable approximation* of a function $f$ is a computable approximation $\langle f_s \rangle$ of $f$, equipped with a uniformly computable sequence $\langle o_s \rangle_{s<\omega}$ of functions from $\omega$ to $R$ such that for all $x$ and $s$:

- $o_{s+1}(x) \leqslant_{\mathcal{R}} o_s(x)$; and
- if $f_{s+1}(x) \neq f_s(x)$, then $o_{s+1}(x) <_{\mathcal{R}} o_s(x)$.

The sequence $\langle o_s \rangle_{s<\omega}$, together with the well-foundedness of $\mathcal{R}$, witnesses the fact that the approximation $\langle f_s \rangle$ indeed reaches a limit.

Definition 1.1. A function $f\colon \omega \to \omega$ is *$\mathcal{R}$-computably approximable* (or *$\mathcal{R}$-c.a.*) if it has an $\mathcal{R}$-computable approximation.

The following equivalent formulation is sometimes taken as a definition:

Proposition 1.2. *A function $f\colon \omega \to \omega$ is $\mathcal{R}$-c.a. if and only if there is a partial computable function $\psi$ such that for all $x$, $f(x) = \psi(x, z)$ for the $\mathcal{R}$-least $z$ such that $(x, z) \in \operatorname{dom} \psi$.*

(In particular, the totality of $f$ implies that for all $x < \omega$ there is some $z \in R$ such that $(x, z) \in \operatorname{dom} \psi$.)

Proof. Let $\langle f_s, o_s \rangle$ be an $\mathcal{R}$-computable approximation of $f$. For $x < \omega$ and $z \in R$, let $\psi(x, z) = f_s(x)$ for any $s < \omega$ such that $o_s(x) = z$; if there is no such $s$, we let $\psi(x, z)\uparrow$. The fact that $f_s(x)$-changes have to be accompanied by an $o_s(x)$-change implies that $\psi$ is well-defined. Then $\psi$ witnesses that $f$ is $\mathcal{R}$-c.a.

Suppose that $\psi$ is a partial computable function as in the proposition. Define a unifomly computable sequence $\langle o_s \rangle$ as follows. Let $A = \operatorname{dom} \psi$. Since $A$ is c.e., let $\langle A_s \rangle$ be some effective enumeration of $A$. Since $f$ is total, for all $x < \omega$ there is some $t_x < \omega$ such that $(x, z) \in A_{t_x}$ for some $z \in R$. For any $x < \omega$ and $s < \omega$ we let $o_s(x)$ be the $\mathcal{R}$-least $z$ such that $(x, z) \in A_{\max\{s, t_x\}}$.

Since $A_t \subseteq A_s$ whenever $t \leqslant s$, we have $o_{s+1}(x) \leqslant o_s(x)$ for all $x$ and $s$. Let $f_s(x) = \psi(x, o_s(x))$. Then $\langle f_s, o_s \rangle$ is an $\mathcal{R}$-computable approximation of $f$. ☐

**1.1. $\mathcal{R}$-c.e. sets.** For sets, Ershov refined the hierarchy of $\mathcal{R}$-c.a. functions to levels resembling the arithmetic hierarchy. For $z \in R$, let $R\restriction_z = \{w \in R : w <_{\mathcal{R}} z\}$, which is a computable $\mathcal{R}$-initial segment of $R$; and let $\mathcal{R}\restriction_z$ be the restriction of $<_{\mathcal{R}}$ to $R\restriction_z$. Recall that an ordinal is *even* if it is of the form $\alpha + 2n$ for some limit ordinal $\alpha$ (or $\alpha = 0$), where $n < \omega$; and *odd* otherwise. We say that $\mathcal{R}$ is *even* if the order-type $\operatorname{otp}(\mathcal{R})$ is even, and *odd* otherwise; and we say that an element $z \in R$ is *$\mathcal{R}$-even* if $\mathcal{R}\restriction_z$ is even, and *$\mathcal{R}$-odd* otherwise. If $\mathcal{R}$ is even, we write $\texttt{parity}(\mathcal{R}) = 0$; otherwise we write $\texttt{parity}(\mathcal{R}) = 1$. Similarly, we write $\texttt{parity}_{\mathcal{R}}(z) = \texttt{parity}(\mathcal{R}\restriction_z)$.

Definition 1.3. Suppose that the collection of $\mathcal{R}$-even elements of $R$ is computable. A set $A \subseteq \omega$ is *$\mathcal{R}$-c.e.* if there is a uniformly c.e. sequence $\langle A_z\rangle_{z\in R}$ such that:

- If $z <_{\mathcal{R}} w$ then $A_z \subseteq A_w$; and
- for all $x < \omega$, $x \in A$ if and only if $x \in \bigcup_{z\in R} A_z$, and for the $\mathcal{R}$-least $z$ such that $x \in A_z$ we have $\texttt{parity}_{\mathcal{R}}(z) \neq \texttt{parity}(\mathcal{R})$.

We let $\Sigma^{-1}_{\mathcal{R}}$ denote the collection of all $\mathcal{R}$-c.e. sets.

The definition should be understood dynamically. Indexed by some *late* element $z$ of $\mathcal{R}$ we see a number $x$ enter the "playground" $\bigcup_w A_w$. We then move backwards in $\mathcal{R}$, so to speak, and at each step we change our mind about whether $x$ is in the target set or not. Thus, this notion extends the finite difference hierarchy. For $n \geqslant 1$, let $n$ also denote a computable linear ordering which has exactly $n$ elements. Then a set is 1-c.e. if it is c.e., is 2-c.e. if it is the (set theoretic) difference of two c.e. sets (also known as d.c.e.), and in general, is $n+1$-c.e. if it is of the form $A\backslash B$, where $A$ is c.e. and $B$ is $n$-c.e.

Ershov lets $\Pi^{-1}_{\mathcal{R}}$ be the collection of complements of $\mathcal{R}$-c.e. sets, and lets $\Delta^{-1}_{\mathcal{R}} = \Sigma^{-1}_{\mathcal{R}} \cap \Pi^{-1}_{\mathcal{R}}$ be the collection of sets which are both $\mathcal{R}$-c.e. and co-$\mathcal{R}$-c.e.

Proposition 1.4. *Suppose again that the parity function* $\texttt{parity}_{\mathcal{R}}$ *is computable. Then every set in $\Delta^{-1}_{\mathcal{R}}$ is $\mathcal{R}$-c.a. If further the order-type of $\mathcal{R}$ is a limit ordinal, then $\Delta^{-1}_{\mathcal{R}}$ coincides with the collection of $\mathcal{R}$-c.a. sets.*

Proof. Suppose that $A \in \Delta^{-1}_{\mathcal{R}}$. Suppose, for simplicity of notation, that $\mathcal{R}$ is even; the odd case is identical. Let $\langle A_z\rangle_{z\in R}$ witness that $A \in \Sigma^{-1}_{\mathcal{R}}$, and $\langle B_z\rangle_{z\in R}$ witness that $A \in \Pi^{-1}_{\mathcal{R}}$. Define a partial computable function $\psi$ as follows. Let $x < \omega$ and $z \in R$. If $x \notin A_z \cup B_z$, we let $\psi(x,z)\uparrow$. Otherwise, $x$ shows up first in either $A_z$ or $B_z$.

- If $x$ shows up first in $A_z$, then we let $\psi(x,z) = \texttt{parity}_{\mathcal{R}}(z)$.
- If $x$ shows up first in $B_z$, then we let $\psi(x,z) = 1 - \texttt{parity}_{\mathcal{R}}(z)$.

Fix $x < \omega$. Then $x \in \bigcup_{z\in R}(A_z \cup B_z)$ because $A \subseteq \bigcup_z A_z$ and $\omega\backslash A \subseteq \bigcup_z B_z$. Hence there is some $z \in R$ such that $(x,z) \in \operatorname{dom}\psi$. Let $z$ be the $\mathcal{R}$-least element of $R$ such that $(x,z) \in \operatorname{dom}\psi$. If $x \in A_z$, then $z$ is the $\mathcal{R}$-least such that $x \in A_z$; so $x \in A$ if and only if $\texttt{parity}_{\mathcal{R}}(z) \neq \texttt{parity}(\mathcal{R}) = 0$. So if $x$ shows up first in $A_z$, then we let $\psi(x,z) = 1$ if and only if $\texttt{parity}_{\mathcal{R}}(z) = 1$ if and only if $A(x) = 1$. If $x \in B_z$, then $z$ is the $\mathcal{R}$-least such that $x \in B_z$, and so $x \notin A$ if and only if $\texttt{parity}_{\mathcal{R}}(z) = 1$; so if $x$ shows up first in $B_z$, then we let $\psi(x,z) = 0$ if and only if $\texttt{parity}_{\mathcal{R}}(z) = 1$ if and only if $A(x) = 0$. Overall, we see that for all $x$, $A(x) = \psi(x,z)$ for the $\mathcal{R}$-least $z$ such that $(x,z) \in \operatorname{dom}\psi$. By Proposition 1.2, $A$ is $\mathcal{R}$-c.a.

For the other direction, it is sufficient to show that every $\mathcal{R}$-c.a. set is in $\Sigma^{-1}_{\mathcal{R}}$; the result would follow from the fact that the complement of an $\mathcal{R}$-c.a. set is also $\mathcal{R}$-c.a. Let $A$ be an $\mathcal{R}$-c.a. set; by Proposition 1.2, let $\psi$ be a partial computable function such that for all $x$, $A(x) = \psi(x, z)$ for the $\mathcal{R}$-least $z$ such that $(x, z) \in \operatorname{dom}\psi$. We assume now that $\mathcal{R}$ has no greatest element. In particular, $\mathcal{R}$ is even.

We define the sequence $\langle A_z\rangle_{z\in R}$ which will show that $A \in \Sigma^{-1}_{\mathcal{R}}$. Let $(x, z) \in \operatorname{dom}\psi$.

- If $\psi(x, z) = \mathtt{parity}_{\mathcal{R}}(z)$ then we let $x \in A_w$ for all $w \geqslant_{\mathcal{R}} z$.
- If $\psi(x, z) \neq \mathtt{parity}_{\mathcal{R}}(z)$ then we let $x \in A_w$ for all $w >_{\mathcal{R}} z$.

It is clear that if $z <_{\mathcal{R}} w$ then $A_z \subseteq A_w$. Let $x < \omega$. We know that there is some $z \in R$ such that $(x, z) \in \operatorname{dom}\psi$. Since $\mathcal{R}$ has no greatest element, no matter what the parity of $z$ is, we enumerate $x$ into some $A_w$; so $x \in \bigcup_w A_w$. Let $w$ be the $\mathcal{R}$-least element of $R$ such that $x \in A_w$. We want to show that $x \in A$ if and only if $w$ is odd in $\mathcal{R}$, in other words, that $A(x) = \mathtt{parity}_{\mathcal{R}}(w)$.

Let $z$ be the $\mathcal{R}$-least element of $R$ such that $(x, z) \in \operatorname{dom}\psi$. Either $\psi(x, z) = \mathtt{parity}_{\mathcal{R}}(z)$, in which case $z = w$; or $\psi(x, z) \neq \mathtt{parity}_{\mathcal{R}}(z)$, in which case $w$ is the $\mathcal{R}$-successor of $z$. In the first case,

$$A(x) = \psi(x, z) = \mathtt{parity}_{\mathcal{R}}(z) = \mathtt{parity}_{\mathcal{R}}(w)$$

as required. In the second case,

$$A(x) = \psi(x, z) = 1 - \mathtt{parity}_{\mathcal{R}}(z) = \mathtt{parity}_{\mathcal{R}}(w),$$

again as required. □

Ash and Knight [4] refer to the sets in $\Delta^{-1}_{\mathcal{R}}$ as "$\mathcal{R}$-computable". However, in common yet misleading terminology, many authors refer to $\mathcal{R}$-c.a. sets as "$\mathcal{R}$-c.e." We prefer to be careful and not confuse the two notions.

**1.2. Listing $\mathcal{R}$-c.a. functions.** For any computable well-ordering $\mathcal{R}$, we can effectively list all $\mathcal{R}$-c.a. functions. To do this we need to consider a nice class of $(\mathcal{R} + 1)$-computable approximations. We of course let $\mathcal{R} + 1$ denote a computable well-ordering extending $\mathcal{R}$ by one element at the end.

DEFINITION 1.5. Let $\mathcal{R}$ be a computable well-ordering. An $(\mathcal{R}+1)$-computable approximation $\langle f_s, o_s\rangle$ is *tidy* if:

- For all $n$, $f_0(n) = 0$; and
- For all $n$ and $s$, if $o_s(n + 1) \in R$ then $o_s(n) \in R$.

The idea is that we have a "partial" $\mathcal{R}$-computable approximation, in that $\langle f_s\rangle$ is total but we may wait a while to declare the elements of $R$ that we use; while we wait we let $o_s(n)$ be the new element beyond $\mathcal{R}$. And further, at every stage we will have declared our "true ordinals" (elements of $R$) for an initial segment of inputs.

LEMMA 1.6. *If $f$ has a tidy $(\mathcal{R}+1)$-computable approximation then $f$ is $\mathcal{R}$-c.a.*

PROOF. Let $\langle g_s, m_s\rangle_{s<\omega}$ be a tidy $(\mathcal{R} + 1)$-computable approximation of $f$. There are two cases. In the first, for all $x$ there is some $s$ such that $m_s(x) \in R$. We say that the approximation is *eventually* $\mathcal{R}$-computable. We then modify the approximation $\langle g_s, m_s\rangle$ by waiting until we see this happen. Formally, for each $x$ we let $t(x)$ be the least $t$ such that $m_t(x) \in R$; we then let, for all $x$ and $s$, $o_s(x) = m_{\max\{s,t(x)\}}(x)$ and $f_s(x) = g_{\max\{s,t(x)\}}(x)$; $\langle f_s, o_s\rangle$ is an $\mathcal{R}$-computable approximation of $f$.

In the second case, for all but finitely many $x$, $o_s(x)$ is constant and equals the extra element of $\mathcal{R}+1$. In that case $f(x) = 0$ for all such $x$, so $f$ is computable. $\square$

It is clear from the proof of Lemma 1.6 that passing from a tidy $(\mathcal{R}+1)$-computable approximation for a function $f$ to an $\mathcal{R}$-computable approximation for $f$ cannot be done uniformly. Indeed a diagonalisation argument shows that there cannot be an effective list of $\mathcal{R}$-computable approximations listing all $\mathcal{R}$-c.a. functions. However we can make a list of tidy $(\mathcal{R}+1)$-computable approximations that yields all $\mathcal{R}$-c.a. functions.

Proposition 1.7. *There is a computable list $\langle\langle f_s^e, o_s^e\rangle_{s<\omega}\rangle_{e<\omega}$ of tidy $(\mathcal{R}+1)$-computable approximations such that letting $f^e = \lim_s f_s^e$, the sequence $\langle f^e\rangle_{e<\omega}$ lists the $\mathcal{R}$-c.a. functions.*

Proof. There is an effective list of all pairs $\langle h_s, m_s\rangle$ of uniformly computable sequences of partial functions. We show how to convert any such pair, uniformly, to a tidy $\mathcal{R}+1$-computable approximation $\langle f_s, o_s\rangle$, such that if $\langle h_s, m_s\rangle$ is an $\mathcal{R}$-computable approximation, then $\lim h_s = \lim f_s$.

Fix such $\langle h_s\rangle$ and $\langle m_s\rangle$. The idea is to define $\langle f_s\rangle$ by copying $\langle h_s\rangle$ with delays, until we see evidence that a change is allowed. Let $\infty$ denote the extra element of $\mathcal{R}+1$. Let $x < \omega$. We start with $f_0(x) = 0$ and $o_0(x) = \infty$. Let $s > 0$. To define $f_s(x)$ and $o_s(x)$, we enumerate the graphs of $\langle h_s\rangle$ and $\langle m_s\rangle$ for $s$ many steps. We let $t_s(x)$ be the greatest $t \leqslant s$ such that for all $r \leqslant t$ and all $y \leqslant x$,

- at stage $s$ we see that $h_r(y)\downarrow$ and $m_r(y)\downarrow$;
- $m_r(y) \in R$, and if $r > 0$, $m_r(y) \leqslant_{\mathcal{R}} m_{r-1}(y)$;
- if $r > 0$ and $h_r(y) \neq h_{r-1}(y)$ then $m_r(y) <_{\mathcal{R}} m_{r-1}(y)$.

If there is no such $t$, then we leave $t_s(x)$ undefined. If $t_s(x)$ is defined then we let $f_s(x) = h_{t_s(x)}(x)$ and $o_s(x) = m_{t_s(x)}(x)$. If $t_s(x)$ is not defined then we let $f_s(x) = 0$ and $o_s(x) = \infty$. $\square$

Note that restricting our approximations to sets, we also get a listing of all $\mathcal{R}$-c.a. sets.

Corollary 1.8. *The collection of $\mathcal{R}$-c.a. functions is uniformly computable from $\mathbf{0}'$. That is, there is a uniformly $\mathbf{0}'$-computable sequence $\langle f^e\rangle_{e<\omega}$ of all $\mathcal{R}$-c.a. functions.*

Remark 1.9. The reader may wonder why, in the case that $\operatorname{otp}(\mathcal{R})$ is a successor ordinal, we cannot list all $\mathcal{R}$-c.a. functions, each with an $\mathcal{R}$-computable approximation. After all, now we do not need to guess which ordinal to start with, we always start with $\max \mathcal{R}$. However we still need to guess what the initial value of our approximation is; we allowed $f_0$ to be any computable function. If we require that $f_0$ is the constant function 0 then we know the initial value but when attempting to diagonalise are restricted to keep our initial value 0 as well, and so may never be allowed to diagonalise. Note that a function has an $(\mathcal{R}+1)$-computable approximation $\langle f_s, o_s\rangle$ with $f_0 = 0$ if and only if it is $\mathcal{R}$-c.a.

### 1.3. Effective embeddings and isomorphisms.

Proposition 1.10. *Let $\mathcal{R}$ and $\mathcal{S}$ be computable well-orderings. If there is a computable embedding of $\mathcal{R}$ into $\mathcal{S}$, then every $\mathcal{R}$-c.a. function is $\mathcal{S}$-c.a.*

PROOF. Let $j\colon R \to S$ be an embedding of $\mathcal{R}$ into $\mathcal{S}$. Let $\langle f_s, o_s\rangle$ be an $\mathcal{R}$-computable approximation. Then $\langle f_s, j \circ o_s\rangle$ is an $\mathcal{S}$-computable approximation. $\square$

COROLLARY 1.11. *Let $\mathcal{R}$ and $\mathcal{S}$ be computable well-orderings. If there is a computable isomorphism between $\mathcal{R}$ and $\mathcal{S}$, then a function is $\mathcal{R}$-c.a. if and only if it is $\mathcal{S}$-c.a.*

**1.4. Bounds on mind-change functions.** Let $\langle f_s\rangle_{s<\omega}$ be a computable approximation of a function $f$. The associated *mind-change function* is

$$m_{\langle f_s\rangle}(x) = \#\left\{s : f_{s+1}(x) \neq f_s(x)\right\}.$$

For any function $g\colon \omega \to \omega$, we say that the approximation $\langle f_s\rangle$ is a *$g$-bounded* approximation if for all $x$, $m_{\langle f_s\rangle}(x) \leqslant g(x)$, that is, if $g$ majorizes $m_{\langle f_s\rangle}$.

Recall that if $\mathcal{A} = (A, <_{\mathcal{A}})$ and $\mathcal{B} = (B, <_{\mathcal{B}})$ are linear orderings, then the product linear ordering $\mathcal{A} \cdot \mathcal{B}$ is the right-lexicographic ordering on $A \times B$. Its order-type is obtained by replacing every point in $\mathcal{B}$ by a copy of $\mathcal{A}$.

PROPOSITION 1.12. *Let $\mathcal{R}$ be a computable well-ordering. A function is $\omega \cdot \mathcal{R}$-c.a. if and only it has a computable approximation which is $g$-bounded for some $\mathcal{R}$-c.a. function $g$.*

PROOF. Let $\langle f_s\rangle$ be a computable approximation of a function $f$.

Suppose that $\langle f_s, o_s\rangle$ is an $\omega \cdot \mathcal{R}$-computable approximation. For any $x$ and $s$, let $o_s(x) = (n_s(x), l_s(x)) \in \omega \times R$. For any $x$ and $s$, we let $t_s(x)$ be the least stage $t \leqslant s$ such that $l_s(x) = l_t(x)$. We then let

$$g_s(x) = n_{t_s(x)}(x) + \#\left\{r < t_s(x) : f_{r+1}(x) \neq f_r(x)\right\}.$$

Then $\langle g_s, l_s\rangle$ is an $\mathcal{R}$-computable approximation of a bound on $m_{\langle f_s\rangle}$.

Suppose that we are given an $\mathcal{R}$-computable approximation $\langle g_s, l_s\rangle$ for a bound $g$ on $m_{\langle f_s\rangle}$. We may assume that for all $x$ and $s$,

$$g_s(x) \geqslant \#\left\{r < s : f_{r+1}(x) \neq f_r(x)\right\},$$

since otherwise we can just wait until $g_t(x)$ changes at some $t > s$. We can therefore let

$$n_s(x) = g_s(x) - \#\left\{r < s : f_{r+1}(x) \neq f_r(x)\right\},$$

and $o_s(x) = (n_s(x), l_s(x))$. If $t < s$ and $l_s(x) = l_t(x)$ then $g_s(x) = g_t(x)$ which shows that if $f_{s+1}(x) \neq f_s(x)$ then $o_{s+1}(x) <_{\omega\cdot\mathcal{R}} o_s(x)$, so $\langle f_s, o_s\rangle$ is an $\omega \cdot \mathcal{R}$-computable approximation. $\square$

Since the computable functions are characterised as those functions which are $\mathcal{R}$-c.a. for $\mathcal{R}$ of order-type 1, and since for any such $\mathcal{R}$, $\omega \cdot \mathcal{R}$ is computably isomorphic to $\omega$, we see that Proposition 1.12 generalises the well-known fact that a function is $\omega$-c.a. if and only if it has a computable approximation whose mind-change function is bounded by a computable function.

## 2. Canonical well-orderings and strong notations

Ershov proved the following:

THEOREM 2.1. *Every $\Delta^0_2$ function is $\mathcal{R}$-c.a. for some computable well-ordering $\mathcal{R}$ of order-type $\omega$.*

Proof. Let $f$ be a $\Delta^0_2$ function. By Shoenfield's limit lemma, let $\langle f_s \rangle$ be a computable approximation for $f$. Let

$$R = \{(x,s) \in \omega \times \omega \,:\, s = 0 \text{ or } f_s(x) \neq f_{s-1}(x)\}\,.$$

For $(x,s)$ and $(y,t) \in R$, let $(x,s) <_{\mathcal{R}} (y,t)$ if $x < y$ or if $x = y$ and $s > t$. For any $x < \omega$ let $R_x$ be the collection of pairs $(x,s)$ in $R$; so $R$ is the disjoint union of the $R_x$'s, each $R_x$ is finite (as $\langle f_s(x) \rangle$ reaches a limit), and the ordering $\mathcal{R} = (R, <_{\mathcal{R}})$ orders $R_0 < R_1 < R_2 < \cdots$ So $\operatorname{otp}(\mathcal{R}) = \omega$.

For $x, s < \omega$, let $t(x,s)$ be the least $t \leqslant s$ such that for all $u \in [t,s]$, $f_u(x) = f_s(x)$. For all $x$ and $s$, $(x, t(x,s)) \in R_x$, and so we can let $o_s(x) = (x, t(x,s))$. It is clear that $\langle f_s, o_s \rangle$ is an $\mathcal{R}$-computable approximation for $f$. □

Ershov's theorem is displeasing as we try to define a hierarchy of complexity inside the $\Delta^0_2$ functions. Its meaning is that calibrating the complexity of a function $f$ by the length of a computable well-ordering $\mathcal{R}$ such that $f$ is $\mathcal{R}$-c.a. is not very informative: the hierarchy collapses at level $\omega$. The reason for this collapse is not that all $\Delta^0_2$ functions have simple approximations, but that the complexity of these approximations can be coded into the isomorphism between $\mathcal{R}$ and $\omega$. In other words, if $\mathcal{R}$ is complicated then $\mathcal{R}$-c.a. functions may be complicated as well, even if $\mathcal{R}$ is short. In terms of the algebraic complexity of $\mathcal{R}$ itself, we notice that key functions associated with $\mathcal{R}$, such as the predecessor and successor function, may be far from computable.

One possible solution is to restrict the computable well-orderings to those given by notations on some $\Pi^1_1$ path through Kleene's $\mathcal{O}$. This is less than satisfying on two accounts. The first is that even though the path may be cofinal in $\mathcal{O}$ (so have notations for every computable ordinal), this does not exhaust all $\Delta^0_2$ functions [30]. The other is that there is no canonical way to choose a path through Kleene's $\mathcal{O}$, and so any such choice is arbitrary, and different choices give different hierarchies of functions.

Another way forward is to give up any claim to exhausting all $\Delta^0_2$ functions, but restrict our attention to a particularly well-behaved class of computable well-orderings. We will require that all orderings in the class that have the same length are computably isomorphic, so Corollary 1.11 will ensure that we will have a good notion of $\alpha$-c.a. functions for some class of computable ordinals $\alpha$. The criterion for canonicity of these orderings is the computability of all reasonable associated functions, such as the predecessor, succesor and so on. It turns out that up to $\varepsilon_0$, the function which encapsulates all the required information is Cantor's normal form.

**2.1. Cantor normal form.** Recall that every ordinal $\alpha$ has a unique expression as the sum

$$\omega^{\alpha_1} n_1 + \omega^{\alpha_2} n_2 + \cdots + \omega^{\alpha_k} n_k$$

where $n_i < \omega$ are nonzero and $\alpha_1 > \alpha_2 > \cdots > \alpha_k$ are ordinals. Recall also that

$$\varepsilon_0 = \sup\left\{\omega, \omega^\omega, \omega^{\omega^\omega}, \omega^{\omega^{\omega^\omega}}, \dots\right\}$$

is the least ordinal $\gamma$ such that $\omega^\gamma = \gamma$, so for all $\alpha < \varepsilon_0$, every ordinal appearing in the Cantor normal form of $\alpha$ is strictly smaller than $\alpha$.

Let $\mathcal{R} = (R, <_{\mathcal{R}})$ be a computable well-ordering, and let $|\cdot|\colon R \to \operatorname{otp}(\mathcal{R})$ be the unique isomorphism between $\mathcal{R}$ and its order-type. The pullback to $\mathcal{R}$ of the

Cantor normal form function is the function $\mathtt{nf}_{\mathcal{R}}$ whose domain is $R$ and is defined by letting

$$\mathtt{nf}_{\mathcal{R}}(z) = \langle (z_1, n_1), (z_2, n_2), \ldots, (z_k, n_k) \rangle$$

where $n_i < \omega$ are nonzero, $z_i \in R$, $z_1 >_{\mathcal{R}} z_2 >_{\mathcal{R}} \cdots >_{\mathcal{R}} z_k$, and

$$|z| = \omega^{|z_1|} n_1 + \omega^{|z_2|} n_2 + \cdots + \omega^{|z_k|} n_k.$$

Definition 2.2. A computable well-ordering $\mathcal{R}$ is *canonical* if its associated Cantor normal form function $\mathtt{nf}_{\mathcal{R}}$ is also computable.

Note that if the relations of ordinal addition and exponentiation by $\omega$ in $\mathcal{R}$ are computable, then $\mathcal{R}$ is canonical.

Proposition 2.3. *Let $\mathcal{R}$ and $\mathcal{S}$ be canonical computable well-orderings, with* $\operatorname{otp}(\mathcal{R}) \leqslant \operatorname{otp}(\mathcal{S}) \leqslant \varepsilon_0$. *Then the unique embedding of $\mathcal{R}$ as an initial segment of $\mathcal{S}$ is computable.*

Proof. Given $z \in R$, recursively construct a tree $T_{\mathcal{R}}(z)$ by placing $z$ at the root of $T_{\mathcal{R}}(z)$, and if $w$ is placed in $T_{\mathcal{R}}(z)$ and $|w| \neq 0$ (that is, $w$ is not the least element of $\mathcal{R}$), then we let the children of $w$ on $T_{\mathcal{R}}(z)$ be the elements of $\mathcal{R}$ which appear as first coordinates in $\mathtt{nf}_{\mathcal{R}}(w)$. The chains of $T_{\mathcal{R}}(z)$ are descending sequences in $\mathcal{R}$, and so all are finite. Also, every node in $T_{\mathcal{R}}(z)$ has but finitely many children on $T_{\mathcal{R}}(z)$, and so by König's Lemma, $T_{\mathcal{R}}(z)$ is finite. Since $\mathtt{nf}_{\mathcal{R}}$ is computable, the map $z \mapsto T_{\mathcal{R}}(z)$ is computable. We similarly define $T_{\mathcal{S}}(w)$ for $w \in S$.

Let $j\colon R \to S$ be the embedding of $\mathcal{R}$ into $\mathcal{S}$ as an initial segment. Then $j(z) = w$ if and only if there is an isomorphism $i$ between $T_{\mathcal{R}}(z)$ and $T_{\mathcal{S}}(w)$ which preserves Cantor normal form, namely for all $x \in T_{\mathcal{R}}(z)$, if

$$\mathtt{nf}_{\mathcal{R}}(x) = \langle (x_1, n_1), (x_2, n_2), \ldots, (x_k, n_k) \rangle$$

then

$$\mathtt{nf}_{\mathcal{S}}(i(x)) = \langle (i(x_1), n_1), (i(x_2), n_2), \ldots, (i(x_k), n_k) \rangle.$$

Hence $j$ is computable. □

Beyond $\varepsilon_0$, we need to strengthen canonicity to obtain an extension of Proposition 2.3. We do not develop this further here, as $\varepsilon_0$ is well beyond the ordinals that come up in the constructions we examine.

**2.2. Existence of canonical well-orderings.** For a computable well-ordering $\mathcal{R}$, The computable well-ordering $\omega^{\mathcal{R}}$, whose order-type is $\omega^{\operatorname{otp}(\mathcal{R})}$, is defined using Cantor normal form. The field of $\omega^{\mathcal{R}}$ is the collection of all sequences of pairs $\langle (z_1, n_1), (z_2, n_2), \ldots, (z_k, n_k) \rangle$ from $R \times (\omega \backslash \{0\})$ such that $z_1 >_{\mathcal{R}} z_2 >_{\mathcal{R}} \cdots >_{\mathcal{R}} z_k$. We let

$$\langle (z_1, n_1), (z_2, n_2), \ldots, (z_k, n_k) \rangle <_{\omega^{\mathcal{R}}} \langle (w_1, m_1), (w_2, m_2), \ldots, (w_l, m_l) \rangle$$

if $k < l$ and for all $i \leqslant k$, $(z_i, n_i) = (w_i, m_i)$; or if for the least $i \leqslant k$ such that $(z_i, n_i) \neq (w_i, m_i)$ we have $w_i <_{\mathcal{R}} z_i$ or $w_i = z_i$ and $n_i < m_i$ (that is, if $(n_i, z_i) <_{\omega \cdot \mathcal{R}} (m_i, w_i)$).

Lemma 2.4. *Let $\mathcal{R}$ be a canonical computable well-ordering. Then the embedding of $\mathcal{R}$ into $\omega^{\mathcal{R}}$ as an initial segment is computable.*

Proof. In fact, this embedding is exactly $\mathtt{nf}_{\mathcal{R}}$. □

LEMMA 2.5. *If $\mathcal{R}$ is a canonical computable well-ordering, then so is $\omega^{\mathcal{R}}$.*

Indeed, a computable index for $\mathtt{nf}_{\omega^{\mathcal{R}}}$ can be effectively obtained from a computable index for $\mathtt{nf}_{\mathcal{R}}$.

PROOF. Let $j = \mathtt{nf}_{\mathcal{R}}$ be the canonical embedding of $\mathcal{R}$ into $\omega^{\mathcal{R}}$. For any $\langle (z_1, n_1), (z_2, n_2), \dots, (z_k, n_k)\rangle$ in the field of $\omega^{\mathcal{R}}$, we have

$$\mathtt{nf}_{\omega^{\mathcal{R}}}\left(\langle (z_1, n_1), (z_2, n_2), \dots, (z_k, n_k)\rangle\right) = \langle (j(z_1), n_1), (j(z_2), n_2), \dots, (j(z_k), n_k)\rangle.$$

□

LEMMA 2.6. *Let $\langle \mathcal{R}_n\rangle$ be a sequence of uniformly computable, uniformly canonical well-orderings (that is, the functions $\mathtt{nf}_{\mathcal{R}_n}$ are uniformly computable). Suppose that for all $n$, $\operatorname{otp}(\mathcal{R}_n) \leqslant \operatorname{otp}(\mathcal{R}_{n+1})$; let $i_n\colon R_n \to R_{n+1}$ be the embedding of $\mathcal{R}_n$ into $\mathcal{R}_{n+1}$ as an initial segment, and suppose that the sequence $\langle i_n\rangle$ is uniformly computable.*

*Then the direct limit of the system $\langle \mathcal{R}_n, i_n\rangle_{n<\omega}$ has a canonical copy.*

PROOF. For $m \leqslant n$, let $i^n_m = i_{n-1} \circ i_{n-2} \circ \dots \circ i_m$ be the initial segment embedding of $\mathcal{R}_m$ into $\mathcal{R}_n$ (and $i^n_n = \mathrm{id}_{R_n}$).

Let

$$\Gamma = \bigcup_n R_n \times \{n\}.$$

For $(w, m), (z, n) \in \Gamma$ where $m \leqslant n$, we let $(w, m) \sim (z, n)$ if $i^n_m(w) = z$. Then $\sim$ is an equivalence relation on $\Gamma$, and the universe of the direct limit of $\langle \mathcal{R}_n, i_n\rangle$ is $\Gamma/\sim$, the collection of $\sim$-equivalence classes. To get a computable copy, we pick out representatives to be the ones that appear earliest in an effective enumeration $\langle \Gamma_s\rangle$ of $\Gamma$, using the fact that $\langle \sim\restriction_{\Gamma_s}\rangle$ is uniformly computable. We let $R$ be this computable set of representatives. The ordering $<_{\mathcal{R}}$ is defined by letting, for $(w, m), (z, n) \in R$ such that $m \leqslant n$, $(w, m) <_{\mathcal{R}} (z, n)$ if $i^n_m(w) <_{\mathcal{R}_n} z$. Certainly $\mathcal{R} = (R, <_{\mathcal{R}})$ is computable, and isomorphic to the direct limit of the system $\langle \mathcal{R}_n, i_n\rangle$. Note also that the representation function $c\colon \Gamma \to R$ defined by requiring that $c(z, n) \sim (z, n)$ is also computable.

Let $(z, n) \in R$, and let $\mathtt{nf}_{\mathcal{R}_n}(z) = \langle (z_1, m_1), \dots, (z_k, m_k)\rangle$. Then

$$\mathtt{nf}_{\mathcal{R}}(z, n) = \langle (c(z_1, n), m_1), (c(z_2, n), m_2), \dots, (c(z_k, n), m_k)\rangle$$

and so $\mathtt{nf}_{\mathcal{R}}$ is computable. □

COROLLARY 2.7. *There is a canonical computable well-ordering of order-type $\varepsilon_0$.*

PROOF. Let $\mathcal{R}_0 = (\omega, <)$ and $\mathcal{R}_{n+1} = \omega^{\mathcal{R}_n}$, and apply Lemmas 2.4, 2.5 and 2.6. □

If $\mathcal{R}$ is a canonical computable well-ordering, then for all $z \in R$, the restriction of $\mathcal{R}$ to the initial segment of $\mathcal{R}$ defined by $z$ is also a canonical computable well-ordering. Hence the collection of ordinals $\alpha$ for which there is a canonical computable well-ordering of length $\alpha$ forms an initial segment of the ordinals. Corollary 2.7 implies the following:

PROPOSITION 2.8. *For every $\alpha \leqslant \varepsilon_0$, there is a canonical computable well-ordering of order-type $\alpha$.*

In view of Propositions 2.3 and 2.8, we identify ordinals $\alpha \leqslant \varepsilon_0$ with canonical well-orderings of order-type $\alpha$.

DEFINITION 2.9. Let $\alpha \leqslant \varepsilon_0$. A function $f$ is *$\alpha$-c.a.* if it is $\mathcal{R}$-c.a. for some (all) canonical well-ordering $\mathcal{R}$ of order-type $\alpha$.

This notion is well-defined by Corollary 1.11 and Propositions 2.3 and 2.8. By Propositions 1.10 and 2.3, if $\alpha < \beta \leqslant \varepsilon_0$, every $\alpha$-c.a. function is $\beta$-c.a.

We go further and fix a canonical well-ordering $\mathcal{R}_{\varepsilon_0}$ of order-type $\varepsilon_0$. We identify $\alpha < \varepsilon_0$ with the element $z \in R_{\varepsilon_0}$ such that $|z|_{\mathcal{R}_{\varepsilon_0}} = \alpha$. As from $z$ we can effectively obtain the initial segment $\mathcal{R}_{\varepsilon_0}\restriction_z$ of $\mathcal{R}_{\varepsilon_0}$ determined by $z$, we say that effectively from $\alpha < \varepsilon_0$ we can get a canonical well-ordering $\mathcal{R}_\alpha$ of order-type $\alpha$. The identification of $\alpha$ with both $\mathcal{R}_\alpha$ and with $\mathcal{R}_\alpha$'s least upper bound in $\mathcal{R}_{\varepsilon_0}$ is true to von Neumann's definition of ordinals: an ordinal here is identified with the collection of its predecessors.

Note that the listing of tidy $\mathcal{R}+1$-computable approximations provided by the proof of Proposition 1.7 is uniform in an index for $\mathcal{R}$. Hence, uniformly in $\alpha < \varepsilon_0$, we can fix an effective list $\langle f_s^{e,\alpha}, o_s^{e,\alpha}\rangle$ of tidy $(\alpha+1)$-computable approximations, where, letting $f^{e,\alpha} = \lim_s f_s^{e,\alpha}$, the sequence $\langle f^{e,\alpha}\rangle_{e<\omega}$ is a listing of all $\alpha$-c.a. functions.

Proposition 1.12 allows us to define some levels of the hierarchy of $\alpha$-c.a. functions:

PROPOSITION 2.10.

(1) *Let $n < \omega$. A function is $\omega^{n+1}$-c.a. if and only if it has a computable approximation which is bounded by an $\omega^n$-c.a. function.*
(2) *Let $\alpha \geqslant \omega$, $\alpha \leqslant \varepsilon_0$. A function is $\omega^\alpha$-c.a. if and only if it has a computable approximation which is bounded by an $\omega^\alpha$-c.a. function.*

**2.3. On ordinal notations.** One of the main uses of Kleene's system of ordinal notations [40] is to define effective transfinite iterations of the Turing jump, giving rise to the hyperarithmetic hierarchy. Roughly speaking, a notation for an ordinal corresponds to a computable well-ordering on which the successor function is computable, and which associates with every limit element a computable cofinal sequence. Formally, for a notation $o \in \mathcal{O}$, the set of predecessors $I(o)$ of $o$ according to $<_\mathcal{O}$ is c.e., uniformly in $o$, but not necessarily computable; however, the uniformity allows us to pull back $<_\mathcal{O}\restriction_{I(o)}$ by an effective enumeration of $I(o)$ to give a computable well-ordering $\mathcal{R}_o$ (with computable domain) isomorphic to $<_\mathcal{O}\restriction_{I(o)}$.

Spector's theorem [58] is in some sense a version of Proposition 2.3: if $a, b \in \mathcal{O}$ and $|a|_\mathcal{O} \leqslant |b|_\mathcal{O}$ then $H_a$, the iteration of the jump along $\mathcal{R}_a$, is Turing reducible to $H_b$, the iteration of the jump along $\mathcal{R}_b$. This suffices to give a precise definition of an increasing sequence of degrees $\mathbf{0}^{(\alpha)}$ for all computable ordinals $\alpha$.

For the purposes of defining $\alpha$-c.a. functions and later, totally $\alpha$-c.a. degrees, general notations are not sufficient, as the well-orderings $\mathcal{R}_o$ are not necessarily canonical. For example, Ershov [29], and later, Epstein, Haas and Kramer [27], define a function to be $\alpha$-c.a. if it is $\mathcal{R}_o$-c.a. for any notation $o \in \mathcal{O}$ for $\alpha$. Under this definition, every $\Delta^0_2$ function is $\omega^2$-c.a., and as we shall see below, every $\Delta^0_2$, low$_2$ degree is totally $\omega^2$-c.a. For the small ordinals we are interested in, there is a natural choice for a system of notations: we say that a notation $o \in \mathcal{O}$ is a *strong notation* if $\mathcal{R}_o$ is canonical. This method was also chosen by Coles, Downey and LaForte [10] in unpublished work looking at hierarchies based on truth table reductions below $\mathbf{0}'$, and by Diamondstone, Hirschfeldt and Nies (unpublished) for variations on Demuth randomness. Note that every notation for an ordinal below $\omega^2$ is strong, but as we shall see, there are notations for $\omega^2$ which are not strong.

Let us say that a computable well-ordering $\mathcal{R}$ of successor order-type is *notation-like* if:

- the successor function on $\mathcal{R}$ is computable; and
- the collection $L(\mathcal{R})$ of limit points of $\mathcal{R}$ is computable.

LEMMA 2.11. *Let $\mathcal{R}$ be notation-like. Then there is an effective map giving, for every $z \in L(\mathcal{R})$, an index for a computable $<_{\mathcal{R}}$-increasing sequence (of order-type $\omega$) cofinal in $\mathcal{R}\restriction_z$.*

PROOF. For each $n$ consider in turn the $\mathcal{R}$-greatest element of $R\restriction_z \cap \{0, \dots, n\}$. $\square$

The reason that we only consider successor order-types is that if $\operatorname{otp}(\mathcal{R})$ is a limit then we would need to add the requirement that there is a computable increasing sequence cofinal in $\mathcal{R}$. It is not fundamnetal.

LEMMA 2.12. *A computable well-ordering of successor order-type is computably isomorphic to $\mathcal{R}_o$ for some $o \in \mathcal{O}$ if and only if $\mathcal{R}$ is notation-like.*

PROOF. Of course, $\mathcal{R}$ is computably isomorphic to $\mathcal{R}_o$ if and only if the order-preserving bijection between $\mathcal{R}$ and $(I(o), <_{\mathcal{O}}\restriction_{I(o)})$ is computable.

If $j\colon R \to I(o)$ is order-preserving, then for all $z \in R$ except for the top element of $\mathcal{R}$, the successor of $z$ in $\mathcal{R}$ is $w$ where $j(w) = 2^{j(z)}$; collection $L(\mathcal{R})$ of limit points of $\mathcal{R}$ is the collection of $z \in R$ such that $j(z) = 3 \cdot 5^e$ for some $e$. This shows that if $\mathcal{R}$ is isomorphic to $\mathcal{R}_o$ for some $o \in \mathcal{O}$ then $\mathcal{R}$ is notation-like.

Suppose now that $\mathcal{R}$ is notation-like. By Lemma 2.11, let $f$ be a computable function such that for $z \in L(\mathcal{R})$, $\varphi_{f(z)}$ is an $<_{\mathcal{R}}$-increasing and cofinal sequence in $\mathcal{R}\restriction_z$.

By effective transfinite recursion (as in [53]) we define a computable injection $j\colon R \to \mathcal{O}$ by letting:

(1) $j(z) = 1$, where $z$ is the $\mathcal{R}$-least element of $R$;
(2) If $z$ is the successor of $w$ in $\mathcal{R}$, then we let $j(z) = 2^{j(w)}$;
(3) If $z \in L(\mathcal{R})$ then $j(z) = 3 \cdot 5^e$ where $\varphi_e = j \circ \varphi_{f(z)}$.

Specifically, we define a partial computable function $F\colon \omega \times R \to \omega$ as follows:

- If $z$ is the $\mathcal{R}$-least element of $R$, then for all $e$ we let $F(e, z) = 1$.
- If $z$ is the $\mathcal{R}$-successor of $w$, then we let $F(e, z) = 2^{\varphi_e(w)}$.
- Let $g$ be a computable function such that for all $a$ and $b$, $\varphi_{g(a,b)} = \varphi_a \circ \varphi_b$. If $z \in L(\mathcal{R})$, then we let $F(e, z) = 3 \cdot 5^{g(e, f(z))}$.

By the recursion theorem, there is an index $e$ such that for all $z \in R$, $F(e, z) = \varphi_e(z)$. Then $j = \varphi_e\restriction_R$ satisfies the conditions (1)–(3) above. The main point is that $R \subseteq \operatorname{dom} \varphi_e$: otherwise, since $\mathcal{R}$ is well-founded, there is an $\mathcal{R}$-least $z \in R$ for which $\varphi_e(z)\uparrow$, which by definition of $F$, must be an $\mathcal{R}$-successor element of some $w \in R$; but then $\varphi_e(w)\downarrow$ implies that $F(e, z)\downarrow$ for a contradiction.

Now the fact that $R \subseteq \operatorname{dom} \varphi_e$ implies that for all $z \in L(\mathcal{R})$, $\varphi_e(z) = 3 \cdot 5^d$ where $\varphi_d$ is indeed an increasing and cofinal sequence in $I(\varphi_e(z))$, so by transfinite induction on the elements of $\mathcal{R}$ we can show that $j$ is an order-preserving bijection between $\mathcal{R}$ and $I(o)$, where $o = 2^{j(z)}$ for $z$ being the $\mathcal{R}$-maximal element of $R$. $\square$

LEMMA 2.13. *Every canonical well-ordering of successor order-type $\beta < \varepsilon_0$ is notation-like.*

PROOF. Let $\alpha \leqslant \varepsilon_0$, and let $\alpha = \omega^{\alpha_1}n_1 + \cdots + \omega^{\alpha_k}n_k$ be the Cantor normal form of $\alpha$. Then $\alpha$ is a limit ordinal if and only if $\alpha_k \neq 0$. If $\alpha$ is a limit, then the successor of $\alpha$ is the ordinal $\beta$ whose Cantor normal form is $\omega^{\alpha_1}n_1 + \cdots + \omega^{\alpha_k}n_k + \omega^0 1$; otherwise, it is the ordinal $\beta$ whose Cantor normal form is $\omega^{\alpha_1}n_1 + \cdots + \omega^{\alpha_k}(n_k + 1)$. $\square$

COROLLARY 2.14 (Coles, Downey, LaForte). *For every $\alpha < \varepsilon_0$, there is a strong notation $o \in \mathcal{O}$ for $\alpha$.*

PROOF. This is immediate for successor $\alpha$; for limit $\alpha$, if $o$ is a strong notation for $\alpha + 1$, then $\log_2 o$ is a strong notation for $\alpha$. $\square$

We show that some notations are not strong.

LEMMA 2.15. *Let $\mathcal{R}$ be a computable well-ordering of order-type $\omega$. Then $\omega \cdot \mathcal{R} + 1$ is notation-like.*

PROOF. The successor of $(n, z) \in \omega \times R$ in $\omega \cdot \mathcal{R}$ is $(n+1, z)$. Let $z_0$ be the $\mathcal{R}$-least element of $R$. Then the collection of limit points of $\omega \cdot \mathcal{R}$ is $(\omega \backslash \{0\}) \times \{z_0\}$. $\square$

Let $\mathcal{R}$ be a computable well-ordering of order-type $\omega$. Certainly $z \mapsto (0, z)$ is a computable embedding of $\mathcal{R}$ into $\omega \cdot \mathcal{R}$. By Proposition 1.10, every $\mathcal{R}$-c.a. function is $\omega \cdot \mathcal{R}$-c.a. By Lemmas 2.12 and 2.15, every $\mathcal{R}$-c.a. function is $\mathcal{R}_o$-c.a. for some notation $o \in \mathcal{O}$ for $\omega^2$. Ershov's Theorem 2.1 now implies:

COROLLARY 2.16 (Ershov). *For every $\Delta^0_2$ function $f$ there is a notation $o \in \mathcal{O}$ for $\omega^2$ such that $f$ is $\mathcal{R}_o$-c.a.*

Most $\Delta^0_2$ function are not $\omega^2$-c.a., and so there are many notations for $\omega^2$ which are not strong.

## 3. Weak truth-table jumps and $\omega^\alpha$-c.a. sets and functions

Coles, Downey and LaForte [10], and independently Anderson and Csima [3], examined the analogue of the Turing jump in the weak truth-table degrees. Anderson and Csima went on to tie levels of sets in the Ershov hierarchy to finite iterations of this bounded jump, generalising the well-known fact that a set is $\omega$-c.a. if and only if it is weak-truth table reducible to $\varnothing'$. If $\varnothing^{\langle n \rangle}$ is the result of iterating the bounded jump operation $n$ times, starting with $\varnothing$ (we give a precise definition below), then Anderson and Csima showed that a set $A \in 2^\omega$ is $\omega^n$-c.a. if and only if it is weak truth-table reducible to $\varnothing^{\langle n \rangle}$.

Coles, Downey and LaForte defined strong notations in order to define an analogue of the $H$-sets in the $\Delta^0_2$ weak truth-table degrees, namely to find a way to define transfinite iterations of the bounded jump operator which are invariant in the weak truth-table degrees. We carry out their programme for ordinals below $\varepsilon_0$, and extend Anderson's and Csima's result to all such ordinals (Theorem

3.11(1)). We further discuss what happens when we pass from sets to functions (Theorem 3.11(2)).

**3.1. bounded $g$-c.e. sets and the bounded jump.** Recall that a function $f$ is *weak truth-table* reducible to a function $g$ if there is a Turing functional $\Phi$ such that $\Phi(g) = f$, and the use function of this reduction is bounded by a computable function. We can extend this to partial functions: for any function $g\colon \omega \to \omega$, we say that a partial function $\psi\colon \omega \to \omega$ is *bounded $g$-computable* if there is a Turing functional $\Phi$ and a partial computable function $\varphi$ such that for all $x$ and $y$, $\psi(x) = y$ if and only if $\varphi(x)\downarrow$ and $\Phi(g\restriction_{\varphi(x)}, x) = y$; and $x \notin \operatorname{dom}\psi$ if $\varphi(x)\uparrow$ or if there is no such $y$. A total function $f$ is bounded $g$-computable if and only if $f \leqslant_{\text{wtt}} g$. Note that in this section, we abandon the convention that for a Turing functional $\Phi$ and an oracle $X$, $\operatorname{dom}\Phi(X)$ is an initial segment of $\omega$.

A *weak truth-table functional* is a pair $(\Phi, \varphi)$ consisting of a Turing functional and a partial computable function. If $(\Phi, \varphi)$ is a weak truth-table functional, $x < \omega$ and $g\colon \omega \to \omega$, then we write $\hat{\Phi}(g, x) = y$ if $\varphi(x)\downarrow$ and $\Phi(g\restriction_{\varphi(x)}, x) = y$. We write $\hat{\Phi}(g, x)\downarrow$ if $\hat{\Phi}(g, x) = y$ for some $y$. The notation $\hat{\Phi}$ assumes that the partial function $\varphi$ is clear from context.

We say that a set $A \in 2^\omega$ is *bounded $g$-c.e.* if it is the domain of a partial bounded $g$-computable function.

We can enumerate all partial bounded $X$-computable functions, and all bounded $X$-c.e. sets, by giving an effective enumeration $\langle \Phi_e, \varphi_e\rangle_{e<\omega}$ of all weak truth-table functionals. We fix such an enumeration which is moreover *acceptable*: if $\langle \Psi_e, \psi_e\rangle_{e<\omega}$ is any effective list of weak truth-table functionals, then there is an (injective) computable function $g$ such that for all $e$, $(\Phi_{g(e)}, \varphi_{g(e)}) = (\Psi_e, \psi_e)$. For all $g\colon \omega \to \omega$, $\left\langle \hat{\Phi}_e(g) \right\rangle_{e<\omega}$ is a $g$-effective list of all partial bounded $g$-computable functions, and letting $\hat{W}^g_e = \operatorname{dom}\hat{\Phi}_e(g)$, $\left\langle \hat{W}^g_e \right\rangle_{e<\omega}$ is a list of all bounded $g$-c.e. sets.

Some of the basic properties of partial computable functions and c.e. sets do not carry over to the bounded realm. The following proposition is meant as a cautionary tale.

PROPOSITION 3.1. *Let $g\colon \omega \to \omega$.*

(1) *Every nonempty bounded $g$-c.e. set is the range of some function $f \leqslant_{\text{wtt}} g$; but there is a function $f \leqslant_{\text{wtt}} \varnothing'$ whose range is not bounded $\varnothing'$-c.e.*
(2) *The graph of any partial bounded $g$-computable function is bounded $g$-c.e.; but there is a (total) function $f$ which is not bounded $\varnothing'$-computable, but whose graph is bounded $\varnothing'$-c.e.*
(3) *If $A \leqslant_{\text{wtt}} g$, then $A$ is bounded $g$-c.e. (and so is its complement). However, there is a c.e. set $C$ and a set $A$ such that both $A$ and its complement $\omega\backslash A$ are bounded $C$-c.e., but $A \nleqslant_{\text{wtt}} C$. For the set $C$ we* cannot *choose $\varnothing'$: if both $A$ and $\omega\backslash A$ are bounded $\varnothing'$-c.e., then $A \leqslant_{\text{wtt}} \varnothing'$.*

Note, however, that with a computable oracle the distinctions disappear: a partial function is bounded $\varnothing$-computable if and only if it is partial computable, and a set is c.e. if and only if it is bounded $\varnothing$-c.e.

SKETCH OF PROOF. For (1), we note that for any $g$, if $A$ is $g$-c.e. and nonempty, then there is some $f \leqslant_{\mathrm{wtt}} g$ such that $A = \operatorname{range} g$. In fact, the use function for reducing $f$ to $g$ can grow as slowly as we like; we simply wait with enumerating some $x \in A$ into the range of $f$ until the input of $f$ is large enough for $A$ to see that $x$ is in $A$. Hence, every nonempty $\Sigma^0_2$ set is the range of some $\omega$-c.a. function. On the other hand, below we see that every bounded $\varnothing'$-c.e. set is $\Delta^0_2$ (in fact, the Anderson-Csima result implies that a set is bounded $\varnothing'$-c.e. if and only if it is $\omega^2$-c.a.) The result follows from the fact that there are $\Sigma^0_2$ sets that are not $\Delta^0_2$.

For (2), note that if $f$ is a $\Delta^0_2$ function which has an increasing approximation, that is , a computable approximation $\langle f_s \rangle$ such that for all $x$ and $s$, $f_s(x) \leqslant f_{s+1}(x)$, then the graph of $f$ is d.c.e., and so $\omega$-c.a., and so weak truth-table reducible to $\varnothing'$, and so certainly bounded $\varnothing'$-c.e. For any $\alpha \leqslant \varepsilon_0$ it is easy to define an increasing approximation for a function $f$ which is not $\alpha$-c.a. by diagonalising against all partial $\alpha$-computable approximations (Proposition 1.7), always *increasing* the value of $f$ if we want to change it. If we choose $\alpha = \omega$, then we get a function which is not $\omega$-c.a., and so not weak truth-table reducible to $\varnothing'$, and so, since it is total, not bounded $\varnothing'$-computable.

We sketch the proofs of (3). First, we enumerate a c.e. set $C$ and define a set $A$ such that both $A$ and $\omega\backslash A$ are bounded $C$-c.e. For $e < \omega$, the requirement $R_e$ seeks a witness $x$ such that $A(x) \neq \hat{\Phi}_e(C, x)$ if the latter converges. After picking a new witness $x$, we state that $x \notin A$ with fresh $C$-use $\psi_{\mathtt{no}}(x)$, and freeze $C\restriction_{\psi_{\mathtt{no}}(x)}$. If later $\hat{\Phi}_e(C_s, x)\downarrow = 0$ (i.e. "$\mathtt{no}$"), then we enumerate $\psi_{\mathtt{no}}(x) - 1$ into $C$ and declare that $x \in A$ with $A$ use $\psi_{\mathtt{yes}}(x) > \varphi_e(x), \psi_{\mathtt{no}}(x)$. Of course this enumeration into $C$ may free the opponent to change their mind and later still let $\hat{\Phi}_e(C_s, x) = 1$ (i.e. "$\mathtt{yes}$"). In that case we enumerate $\psi_{\mathtt{yes}}(x) - 1$ into $C$ but freeze $C$ below that number, and declare that $x \notin A$ with the old use $\psi_{\mathtt{no}}(x)$. The point is that $\psi_{\mathtt{yes}}(x) > \varphi_e(x)$, so us freezing $C$ means that the opponent cannot change their mind again and is stuck with declaring that $x \in A$, whereas we leave $x \notin A$ for ever after that. Each time a requirement acts, all weaker requirements are initialised and are forced to pick new witnesses; so this is a finite injury construction.

The difference between $C$ and $\varnothing'$, is that unlike an arbitrary c.e. set $C$, the opponent in the previous construction, that is us in the current construction, controls a portion of $\varnothing'$. That is, we enumerate an auxiliary c.e. set $E$, and by the recursion theorem we know an index $e$ such that $E = W_e$ which is the $e^{\text{th}}$ column of $\varnothing'$. Suppose that we are given that $A = \operatorname{dom} \hat{\Phi}_1(\varnothing')$ and $\omega\backslash A = \operatorname{dom} \hat{\Phi}_0(\varnothing')$. To reduce $A$ to $\varnothing'$, given $x$, we wait for some $i$ and $s$ such that $\hat{\Phi}_i(\varnothing'_s, x)\downarrow$, i.e., $\varphi_i(x)\downarrow$ at stage $s$ and $\Phi_i(\varnothing'_s, x)$ converges with use below $\varphi_i(x)$. We then set $\psi(x)$ to be some number large enough so that the agent which is responsible for computing $A(x)$ can control $\varphi_i(x)$ many elements of $\varnothing'$ (via $E$) with no interference from other agents which have already staked their claims for portions of $E$. We show that this control is sufficient to compute $A(x)$ from $\varnothing'\restriction_{\psi(x)}$. As long as $\hat{\Phi}_i(\varnothing'_s, x)\downarrow$, we keep stating that $A(x) = i$, with use $\varnothing'_s\restriction_{\psi(x)}$. If $\varnothing'$ changes below $\varphi_i(x)$, and we then see that $\hat{\Phi}_{1-i}(\varnothing'_s, x)\downarrow$, then we declare that $A(x) = 1 - i$ with use the new version of $\varnothing'\restriction_{\psi(x)}$ (as $\psi(x) > \varphi_i(x)$). If the computation $\hat{\Phi}_{1-i}(\varnothing', x)$ fizzles, we wait to see if we next get a new computation $\hat{\Phi}_i(\varnothing'_s, x)\downarrow$. If not, then we will later get a new $\hat{\Phi}_{1-i}(\varnothing', x)\downarrow$ computation, and we didn't need to do anything. Otherwise, we enumerate one of our agitators into $E$ so that we can redefine $A(x) = i$ with the new version of $\varnothing'\restriction_{\psi(x)}$. The point is that no matter how large $\varphi_{1-i}(x)$ (it may be much larger than $\psi(x)$), every enumeration into $E$ on behalf of computing $A(x)$ is tied to a failed $\hat{\Phi}_i(\varnothing'_s, x)$ computation, and so to some historic version of $\varnothing'\restriction_{\varphi_i(x)}$. Thus we never run out of agitators and we can keep up with the changes in $A$ and record them into $\varnothing'\restriction_{\psi(x)}$ correctly. $\square$

With the dangers of bounded oracle computations in mind, we turn to define the bounded jump and a universal "jump function". For an oracle $g\colon \omega \to \omega$, we let

$$g^\dagger = \bigoplus_{e<\omega} \hat{W}_e^g = \left\{(e,x) \,:\, x \in \hat{W}_e^g\right\}.$$

In analogy with the jump function $J$, we define a function $I^g$ as follows:

$$I^g(e,x) = \begin{cases} 0, & \text{if } x \notin \hat{W}_e^g; \\ \hat{\Phi}_e(g,x)+1, & \text{otherwise.} \end{cases}$$

Elementary properties of these jump operations are analogous to those of the Turing jump.

LEMMA 3.2. *Let $g\colon \omega \to \omega$.*

(1) *$g^\dagger$ is 1-complete for the class of bounded $g$-c.e. sets.*
(2) *$g^\dagger$ is computably isomorphic to the set $\left\{e \,:\, e \in \hat{W}_e^g\right\}$.*

PROOF. (1) – the fact that $g^\dagger$ is bounded $g$-c.e. – follows from the fact that the enumeration $\langle \Phi_e, \varphi_e\rangle_{e<\omega}$ is effective: $\langle \Phi_e\rangle$ is uniformly c.e., and $\langle \varphi_e\rangle$ are uniformly partial computable.

Let $g^* = \left\{e \,:\, e \in \hat{W}_e^g\right\}$. Since $g^*$ is bounded $g$-c.e., to show (2) it is sufficient to show that $g^*$ is also 1-complete for the class of bounded $g$-c.e. sets. Let $(\Phi, \varphi)$ be a weak truth-table functional. To reduce $\hat{\Phi}(g)$ to $g^*$, given any $x < \omega$ we define a partial computable function $\psi_x$ such that for all $w$, $\psi_x(w)\downarrow$ if and only if $\varphi(x)\downarrow$, in which case, $\psi_x(w) = \varphi(x)$ for all $w$; and also define a Turing functional $\Psi_x$ such that if $\varphi(x)\uparrow$, then $\Psi_x(h,w)\uparrow$ for all $w < \omega$ and all oracles $h$, and if $\varphi(x)\downarrow$, then $\Psi_x(h,w) = \Phi(h,w)$ for all oracles $h$ and all $w < \omega$, with the same use. Since the numbering $\langle \Phi_e, \varphi_e\rangle$ is acceptable, there is an injective computable function $f$ such that for all $x < \omega$, $(\Psi_x, \psi_x) = (\Phi_{f(x)}, \varphi_{f(x)})$. Then $f$ witnesses that $\operatorname{dom} \hat{\Phi}(g) \leqslant_1 g^*$. $\square$

For functions $f, g\colon \omega \to \omega$, we say that $f \leqslant_{\mathrm{m}} g$ if there is a computable function $h$ such that $f = g \circ h$. Note that this definition extends the familiar one for sets. If $f \leqslant_{\mathrm{m}} g$ then $f \leqslant_{\mathrm{wtt}} g$.

LEMMA 3.3. *Let $g\colon \omega \to \omega$. A set $A$ is bounded $g$-c.e. if and only if $A \leqslant_{\mathrm{m}} g^\dagger$.*

PROOF. Let $A \leqslant_{\mathrm{m}} g^\dagger$; so there is a computable function $h$ such that $A = h^{-1} g^\dagger$. Let $x < \omega$; let $(e,y) = h(x)$. Then we let $\psi(x) = \varphi_e(y)$ and $\Psi(h,x) = \Phi_e(h,y)$ for every oracle $h$, with the same use. Then $A = \operatorname{dom} \hat{\Psi}(g)$. $\square$

LEMMA 3.4. *For all $g\colon \omega \to \omega$,*

(1) *$g^\dagger \leqslant_{\mathrm{wtt}} I^g$.*
(2) *$I^g$ is many-one equivalent to the "diagonal function" $e \mapsto I^g(e,e)$.*

PROOF. For (1), we have $(e,x) \in g^\dagger$ if and only if $I^g(e,x) \neq 0$. For (2), the reduction $f$ of the proof of Lemma 3.2 satisfies $I^g(e,x) = I^g(f(e,x), f(e,x))$ for all $e$ and $x$. $\square$

LEMMA 3.5. *Let $g\colon \omega \to \omega$.*

(1) $g <_{\mathrm{wtt}} I^g$.
(2) *For any* set $A$, $A <_{\mathrm{wtt}} A^\dagger$.

PROOF. Let $\psi = \mathrm{id}$ be the identity function, and let $\Psi$ be a Turing functional which maps any sequence $\sigma$ to itself. So for all $g\colon \omega \to \omega$, $\hat{\Psi}(g) = g$. Hence there is some $e$ such that $g(x) = I^g(e,x)$, so $g \leqslant_{\mathrm{wtt}} I^g$.

Every set $A$ is bounded $A$-c.e., and so by Lemma 3.3, $A \leqslant_{\mathrm{m}} A^\dagger$. It follows that $A \leqslant_{\mathrm{wtt}} A^\dagger$.

The proof of (1) and (2) will be complete with the aid of Lemma 3.4(1), once we show that for any function $g$, $g^\dagger \nleqslant_{\mathrm{wtt}} g$. This is Cantor's argument, as the set

$$\left\{e \,:\, e \notin \hat{W}_e^g\right\}$$

is weak truth-table reducible to $g^\dagger$, and is not bounded $g$-c.e., so cannot be weak truth-table reducible to $g$ (Proposition 3.1(3)). $\square$

LEMMA 3.6. *Let $f, g\colon \omega \to \omega$.*

(1) *$f \leqslant_{\mathrm{wtt}} g$ if and only if $I^f \leqslant_{\mathrm{m}} I^g$.*
(2) *If $f \leqslant_{\mathrm{wtt}} g$ then $f^\dagger \leqslant_{\mathrm{m}} g^\dagger$. The converse fails, even when restricting to sets rather than functions.*

*If $f, g\colon \omega \to \omega$ and $f \leqslant_{\mathrm{wtt}} g$, then from an index $e$ such that $\hat{\Phi}_e(g) = f$ we can effectively obtain indices $c$ and $d$ such that $\hat{\Phi}_c(I^g) = I^f$ and $\hat{\Phi}_d(g^\dagger) = f^\dagger$.*

It follows that the operations $g \mapsto I^g$ and $g \mapsto g^\dagger$ induce well-defined, strictly increasing functions on the partial ordering of the weak truth-table degrees.

PROOF. It is easy to show that if $f \leqslant_{\mathrm{wtt}} g$, then $I^f \leqslant_{\mathrm{m}} I^g$ and $f^\dagger \leqslant_{\mathrm{m}} g^\dagger$. One simply composes the reduction of $f$ to $g$ with any weak truth-table functional; this composition is uniform in an index for a reduction of $f$ to $g$.

Let $f, g\colon \omega \to \omega$, and suppose that $h$ is computable and $I^f = I^g \circ f$. Fox $e$ such that $f = \hat{\Phi}_e(f)$. Let $x < \omega$, and let $(d,y) = h(e,x)$. Since

$$I^g(d,y) = I^f(e,x) = f(x) + 1 > 0,$$

we have $\hat{\Phi}_d(g,y){\downarrow} = f(x)$, which shows that $f \leqslant_{\mathrm{wtt}} g$.

The failure of the converse to (2) is exhibited by an argument similar to the one proving the first part of Proposition 3.1(3), and so we only sketch it. We enumerate a c.e. set $B$ and approximate a d.c.e. set $B$ such that $A \nleqslant_{\mathrm{wtt}} B$ but $A^\dagger \leqslant_{\mathrm{m}} B^\dagger$. Instances $R_{e,x}$ of a global requirement for coding $A^\dagger$ into $B^\dagger$ define the value at $(e,x)$ of a partial computable function $\psi$ and enumerate axioms with use $B_s \restriction_{\psi(e,x)}$ into a functional $\Psi$; we then can find a computable function $h$ such that for all $e$ and $x$, $(h(e,x), h(e,x)) \in B^\dagger$ if and only if $\hat{\Psi}(B,e,x){\downarrow}$; we need to ensure that this happens if and only if $(e,x) \in A^\dagger$. Requirements $P_i$ diagonalise $A(z)$ against $\hat{\Phi}_i(B,z)$ for some appointed follower $z$. The priorities of the $R_{e,x}$ requirements are interspersed between the $P_i$ requirements. In a typical scenario, $R_{e,x}$ observes that $\hat{\Phi}_e(A_s,x){\downarrow}$ for the first time; it sets $\psi(e,x)$ to be some large number, and lets $\hat{\Psi}(B_s,e,x){\downarrow}$. The size of $\psi(e,x)$ allows the requirement $R_{e,x}$ to enumerate a number into $B$ once for each follower $z < \varphi_e(x)$ for a requirement $P_i$ stronger than $R_{e,x}$. Of course followers for weaker requirements are cancelled and new ones are chosen to be larger than $\varphi_e(x)$. Whenever a strong $P_i$ enumerates its follower $z$ into $A$, the opponent may change whether $\hat{\Psi}_e(A_s, x)$ converges or not. If the change is from convergence to divergence, then we need to enumerate some number below $\psi(e,x)$ into $B$. This, in turn, may cause $\hat{\Phi}_i(B, z)$ to change, making $P_i$ want to extract $z$ from $A$. It does so, this time freezing $B\upharpoonright_{\varphi_i(z)}$. The fact that $\varphi_i(z)$ may be larger than $\psi(e,x)$ does not disturb us: the extraction of $z$ from $A$ gives our opponent an opportunity to make $\hat{\Phi}_e(A, x)$ converge again, but we can then make $\hat{\Psi}(B, e, x)$ converge without changing $B\upharpoonright_{\psi(e,x)}$, simply by enumerating a new axiom into $\Psi$. If later an even stronger requirement $P_j$ acts, the process repeats, injuring $P_i$, but any new $P_i$ follower will be greater than $\varphi_e(x)$, and so never disturb $R_{e,x}$ again. Hence we can fix $\psi(e,x)$ based on the priority of $R_{e,x}$ whenever we see $\varphi_e(x)$ converge. □

LEMMA 3.7. *For all $\Delta^0_2$ functions $g$, $I^g$ is also $\Delta^0_2$.*

And so $g^\dagger$ is also $\Delta^0_2$.

PROOF. Let $\langle g_s\rangle$ be a computable approximation for $g$. For $e, x, s < \omega$, let $h_s(e,x) = 0$ if $\varphi_{e,s}(x)\uparrow$, or if $\Phi_{e,s}(g_s\upharpoonright_{\varphi_e(x)})\uparrow$. Otherwise let $h_s(e,x) = \Phi_{e,s}(g_s\upharpoonright_{\varphi_e(x)})$. Then $\langle h_s\rangle$ is a computable approximation of $I^g$. The point, of course, is that if $\varphi_e(x)\downarrow$, then $g_s\upharpoonright_{\varphi_e(x)}$ eventually stabilizes. □

Since bounded $\varnothing$-c.e. sets are simply c.e. sets, $\varnothing^\dagger$ and $\varnothing'$ are computably isomorphic. Both sets are weak truth-table equivalent to $I^\varnothing$, since if we know that $\hat{\Phi}_e(\varnothing, x)\downarrow$, then finding the value $\hat{\Phi}_e(\varnothing, x)$ can be done effectively. Hence, for any $\Delta^0_2$ function $g$ we have $g^\dagger \equiv_{\mathrm{T}} I^g \equiv_{\mathrm{T}} \varnothing'$.

**3.2. Transfinite iterations of the bounded jump.** Let $g\colon \omega \to \omega$. We define, for a computable well-ordering $\mathcal{R} = (R, <_{\mathcal{R}})$, the iteration of the bounded jump set and function along $\mathcal{R}$, by induction on the order-type of $\mathcal{R}$.

- If $\mathcal{R}$ is empty, then we let $g^{\langle\mathcal{R}\rangle} = I^g_{\mathcal{R}} = g$.

Suppose that $\mathcal{R}$ is nonempty, and that by recursion, for all $z \in R$, both $g^{\langle\mathcal{R}\upharpoonright_z\rangle}$ and $I^g_{\mathcal{R}\upharpoonright_z}$ have already been defined.

- If the order-type of $\mathcal{R}$ is a successor ordinal, let $z$ be the $\mathcal{R}$-greatest element of $R$; we then let $g^{\langle\mathcal{R}\rangle} = \left(g^{\langle\mathcal{R}\upharpoonright_z\rangle}\right)^\dagger$ and $I^g_{\mathcal{R}} = I^{I^g_{\mathcal{R}\upharpoonright_z}}$.
- If the order-type of $\mathcal{R}$ is a limit ordinal, we let $g^{\langle\mathcal{R}\rangle} = \bigoplus_{z\in R} g^{\langle\mathcal{R}\upharpoonright_z\rangle}$ and $I^g_{\mathcal{R}} = \bigoplus_{z\in R} I^g_{\mathcal{R}\upharpoonright_z}$. By this we mean that for all $z$ and $x$, $(z,x) \in g^{\langle\mathcal{R}\rangle}$ if and only if $z \in R$ and $x \in g^{\langle\mathcal{R}\upharpoonright_z\rangle}$ (so we consider $g^{\langle\mathcal{R}\rangle}$ as an element of $2^\omega$); and if $z \in R$, then $I^g_{\mathcal{R}}(z,x) = I^g_{\mathcal{R}\upharpoonright_z}(x)$, whereas if $z \notin R$ then $I^g_{\mathcal{R}}(z,x) = 0$; so we can consider $I^g_{\mathcal{R}}$ as a function from $\omega$ to $\omega$.

PROPOSITION 3.8. *Let $\mathcal{R}$ and $\mathcal{S}$ be computable well-orderings. Suppose that* $\mathrm{otp}(\mathcal{R}) \leqslant \mathrm{otp}(\mathcal{S})$. *Also suppose that the embedding of $\mathcal{R}$ as an initial segment of $\mathcal{S}$ is computable. Suppose further that $\mathcal{R}$ is notation-like. Then for all $g\colon \omega \to \omega$, $g^{\langle\mathcal{R}\rangle} \leqslant_{\mathrm{wtt}} g^{\langle\mathcal{S}\rangle}$ and $I^g_{\mathcal{R}} \leqslant_{\mathrm{wtt}} I^g_{\mathcal{S}}$.*

PROOF. Let $j\colon R \to S$ be the initial segment embedding of $\mathcal{R}$ into $\mathcal{S}$. We show that there are computable functions $f$ and $h$ such that for all $z \in R$, for all

$g\colon \omega \to \omega$

$$g^{\langle \mathcal{R}\restriction_z \rangle} = \hat{\Phi}_{f(z)}(g^{\langle \mathcal{S}\restriction_{j(z)} \rangle})$$

and

$$I^g_{\mathcal{R}\restriction_z} = \hat{\Phi}_{h(z)}(I^g_{\mathcal{S}\restriction_{j(z)}}).$$

Replacing $\mathcal{R}$ and $\mathcal{S}$ by one element extensions $\mathcal{R}+1$ and $\mathcal{S}+1$ then yields the desired conclusion.

The definitions of $f$ and $h$ is done by effective transfinite recursion along $\mathcal{R}$. Directly, we define:

(1) If $z$ is the $\mathcal{R}$-least element of $R$, then we let $f(z) = h(z) = e$ where $\hat{\Phi}_e(g) = g$ for all $g\colon \omega \to \omega$.

(2) If $z$ is the $\mathcal{R}$-successor of $w$, then by Lemma 3.6, from $f(w)$ we can effectively find a number $f(z)$ such that for all $g$,

$$\hat{\Phi}_{f(z)}\left(g^{\left\langle \mathcal{S}\restriction_{j(z)} \right\rangle}\right) = \hat{\Phi}_{f(z)}\left(\left(g^{\left\langle \mathcal{S}\restriction_{j(w)} \right\rangle}\right)^{\dagger}\right) = \left(g^{\langle \mathcal{R}\restriction_w \rangle}\right)^{\dagger} = g^{\langle \mathcal{R}\restriction_z \rangle},$$

and from $h(w)$ we can effectively find a number $h(z)$ such that for all $g$,

$$\hat{\Phi}_{h(z)}\left(I^g_{\mathcal{S}\restriction_{j(z)}}\right) = \hat{\Phi}_{h(z)}\left(I^{I^g_{\mathcal{S}\restriction_{j(w)}}}\right) = I^{I^g_{\mathcal{R}\restriction_w}} = I^g_{\mathcal{R}\restriction_z}.$$

(3) If $z$ is a limit point of $\mathcal{R}$, then from $g^{\left\langle \mathcal{S}\restriction_{j(z)} \right\rangle}$ and $I^g_{\mathcal{S}\restriction_{j(z)}}$ we can obtain, uniformly in $g$ and in $w <_{\mathcal{R}} z$, $g^{\left\langle \mathcal{S}\restriction_{j(w)} \right\rangle}$ and $I^g_{\mathcal{S}\restriction_{j(w)}}$, respectively, in a weak truth-table fashion. Thus from $f\restriction_{R\restriction_z}$ and $h\restriction_{R\restriction_z}$ we can compute indices $f(z)$ and $h(z)$ such that for all $w <_{\mathcal{R}} z$, for all $x < \omega$, for all $g$,

$$\begin{aligned}\hat{\Phi}_{f(z)}\left(g^{\left\langle \mathcal{S}\restriction_{j(z)} \right\rangle}, (w,x)\right) &= \hat{\Phi}_{f(w)}\left(g^{\left\langle \mathcal{S}\restriction_{j(w)} \right\rangle}, x\right) \\ &= g^{\langle \mathcal{R}\restriction_w \rangle}(x) = g^{\langle \mathcal{R}\restriction_z \rangle}(w,x)\end{aligned}$$

and

$$\hat{\Phi}_{h(z)}\left(I^g_{\mathcal{S}\restriction_{j(z)}}, (w,x)\right) = \hat{\Phi}_{h(w)}\left(I^g_{\mathcal{S}\restriction_{j(w)}}, x\right) = I^g_{\mathcal{R}\restriction_w}(x) = I^g_{\mathcal{R}\restriction_z}(w,x),$$

and so $\hat{\Phi}_{f(z)}\left(g^{\left\langle \mathcal{S}\restriction_{j(z)} \right\rangle}\right) = g^{\langle \mathcal{R}\restriction_z \rangle}$ and $\hat{\Phi}_{h(z)}\left(I^g_{\mathcal{S}\restriction_{j(z)}}\right) = I^g_{\mathcal{R}\restriction_z}$ as required.

The details of the effective transfinite recursion, using the recursion theorem, are as in the proof of Lemma 2.12. ☐

It follows that if $\mathcal{R}$ and $\mathcal{S}$ are computably isomorphic, then for all $g$, $g^{\langle \mathcal{R} \rangle} \equiv_{\mathrm{wtt}} g^{\langle \mathcal{S} \rangle}$ and $I^g_{\mathcal{R}} \equiv_{\mathrm{wtt}} I^g_{\mathcal{S}}$. Hence, using canonical well-orderings, for $\alpha \leqslant \varepsilon_0$, we can unambiguously define $g^{\langle \alpha \rangle}$ and $I^g_\alpha$ for all $g$ – these are unique up to weak truth-table degree, and in fact many-one degree if $\alpha > 0$, and induce well-defined operations on the weak truth-table degrees. If $\alpha < \beta$ then $g^{\langle \alpha \rangle} <_{\mathrm{wtt}} g^{\langle \beta \rangle}$ and $I^g_\alpha <_{\mathrm{wtt}} I^g_\beta$.

Proposition 3.9. *Let $g\colon \omega \to \omega$ and let $\alpha \leqslant \varepsilon_0$. Then $g^{\langle \alpha \rangle} \leqslant_{\mathrm{wtt}} I^g_\alpha$.*

Proof. By effective transfinite recursion on $\varepsilon_0 + 1$ we build a computable function $R$ such that for all $\alpha \leqslant \varepsilon_0$ and all $g$, $\hat{\Phi}_{R(\alpha)}(I^g_\alpha) = g^{\langle \alpha \rangle}$. This is done by cases:

(1) Since $I^g_\alpha = g^{\langle 0 \rangle} = g$, we let $R(0)$ be a number such that for all $g$, $\hat{\Phi}_{R(0)}(g) = g$.

(2) The proofs of Lemma 3.4(1) and Lemma 3.6(2) show that there is a computable function $S$ such that for all $\alpha < \varepsilon_0$ and all $a < \omega$, if $\hat{\Phi}_a(g) = f$ then $\hat{\Phi}_{S(\alpha,a)}(I^g) = f^\dagger$. We then let, for all $\alpha < \varepsilon_0$, $R(\alpha+1) = S(\alpha, R(\alpha))$.
(3) For limit $\alpha$, we string together the reductions for $\beta < \alpha$. The construction for part (3) of the proof of Proposition 3.8 shows that there is a computable function $T$ such that for all limit ordinals $\alpha \leqslant \varepsilon_0$ and all $a < \omega$, for all sequences $\langle g_\beta \rangle_{\beta<\alpha}$ of functions,

$$\hat{\Phi}_{T(\alpha,a)}\left(\bigoplus_{\beta<\alpha} g_\beta\right) = \bigoplus_{\beta<\alpha} \hat{\Phi}_a(g_\beta).$$

We then let $R(\alpha) = T(\alpha, a)$, where $a$ is an index such that $\varphi_a \restriction_\alpha = R \restriction_\alpha$.

Again to make things concrete, we show how to perform this recursion: we define a function $F$. For all $a < \omega$, we let $F(0,a) = R(0)$, $F(\alpha+1, a) = S(\alpha, a)$ and for limit $\alpha$, $F(\alpha, a) = T(\alpha, a)$. By the recursion theorem, there is an index $a$ such that $F(-,a) = \varphi_(a)$. Since $F(a, 0)$ is defined for all $a$, and since $S$ and $T$ are total, we have $\varepsilon_0 + 1 \subseteq \operatorname{dom}\varphi_a$. The function $R = \varphi_a\restriction_{\varepsilon_0+1}$ is as required. ☐

Proposition 3.10. *For any $\Delta^0_2$ function $g$ and any $\alpha \leqslant \varepsilon_0$, $I^g_\alpha$ is $\Delta^0_2$.*

Proof. Fix a $\Delta^0_2$ function $g$. By effective transfinite recursion we build a computable function $R$ such that for all $\alpha \leqslant \varepsilon_0$, $\varphi_{R(\alpha)} = \langle g^\alpha_s \rangle$ is a computable approximation of $I^g_\alpha$.

(1) We let $R(0)$ be an index for a computable approximation of $g$.
(2) The proof of Lemma 3.7 shows that there is a computable function $S$ such that for all $a < \omega$, if $\varphi_a$ is a computable approximation of a function $h$, then $\varphi_{S(a)}$ is a computable approximation of $I^h$. For any $\alpha < \varepsilon_0$, we let $R(\alpha+1) = S(R(\alpha))$.
(3) An argument similar to previous ones shows that there is a computable function $T$ such that for all $a < \omega$ and all limit $\alpha \leqslant \varepsilon_0$, if for all $\beta < \alpha$, $\varphi_{\varphi_a(\beta)}$ is a computable approximation of a function $h_\beta$, then $\varphi_{T(\alpha,a)}$ is a computable approximation of $\bigoplus_{\beta<\alpha} h_\beta$. Then we let, for limit ordinals $\alpha$, $R(\alpha) = T(\alpha, a)$, where $a$ is an index for $R\restriction_\alpha$. ☐

The following theorem, a refinement of Proposition 3.10, is the goal of this section:

Theorem 3.11. *Let $\alpha \leqslant \varepsilon_0$.*

(1) *A set $A$ is $\omega^\alpha$-c.a. if and only if $A \leqslant_{\mathrm{wtt}} \varnothing^{\langle\alpha\rangle}$.*
(2) *A function $g$ is $\omega^\alpha$-c.a. if and only if $g \leqslant_{\mathrm{wtt}} I^{\varnothing}_\alpha$.*

This theorem generalises the fact that a function or a set is $\omega$-c.a. if and only if it is weak truth-table reducible to $\varnothing'$. Anderson and Csima [3] proved part (1) of the theorem for $\alpha < \omega$.

We note that for $\alpha \geqslant 2$, we really do need to use the function jump $I^g$ rather than the set jump $g^\dagger$:

Proposition 3.12. *There is an $(\omega+1)$-c.a. function which is not weak truth-table reducible to any set.*

Proof. We define an $(\omega+1)$-computable approximation $\langle f_s, o_s \rangle$ for a function $f$. For each $e < \omega$, we want to ensure that for any set $A$, $f(e) \neq \hat{\Phi}_e(A, e)$. Let $e < \omega$.

We let $V_e$ be the collection of all values $\hat{\Phi}_e(\sigma, e)$, as $\sigma$ ranges over all binary strings of length $\varphi_e(e)$. For any set $A$, if $\hat{\Phi}_e(A, e)\downarrow$, then $\hat{\Phi}_e(A, e) \in V_e$. The sequence $\langle V_e \rangle$ is c.e., uniformly in $e$; if $\varphi_e(e)\uparrow$, then $|V_e| = 0$, and otherwise $|V_e| \leqslant 2^{\varphi_e(e)}$. Let $\langle V_{e,s} \rangle$ be a uniformly computable enumeration of the sets $V_e$.

For all $s < \omega$, if $\varphi_{e,s}(e)\uparrow$, then we let $f_s(e) = 0$ and $o_s(e) = \omega$. Otherwise, we let $f_s(e)$ be the least element of $\omega \backslash V_{e,s}$, and let $o_s(e) = 2^{\varphi_e(e)} - |V_{e,s}|$. Then $f(e) \notin V_e$, which gives the desired diagonalisation. □

**3.3. Commutative addition and powers of $\omega$.** We focus on ordinal powers of $\omega$ because these consist precisely of the ordinals which are closed under addition.

PROPOSITION 3.13. *An ordinal $\alpha > 0$ is closed under addition if and only if $\alpha = \omega^\beta$ for some $\beta$.*

PROOF. Let $\beta$ be any ordinal, and let $\gamma, \delta < \omega^\beta$. Let $\gamma = \omega^{\gamma_1} n_1 + \dots \omega^{\gamma_k} n_k$ and $\delta = \omega^{\delta_1} m_1 + \cdots + \omega^{\delta_l} m_l$ be the Cantor normal forms of $\gamma$ and $\delta$. Since $\omega^{\gamma_1} \leqslant \gamma < \omega^\beta$, we have $\gamma_1 < \beta$; similarly, $\delta_1 < \beta$. Hence

$$\gamma + \delta \leqslant \omega^{\gamma_1}(n_1 + 1) + \omega^{\delta_1}(m_1 + 1) \leqslant \omega^{\max\{\gamma_1, \delta_1\}}(n_1 + m_1) < \omega^\beta,$$

so $\omega^\beta$ is closed under addition.

Let $\alpha$ be an ordinal which is not a power of $\omega$. Let $\alpha = \omega^{\alpha_1} n_1 + \cdots + \omega_{\alpha_k} n_k$ be the Cantor normal form of $\alpha$. Since $\alpha \neq \omega^{\alpha_1}$, we have $\omega^{\alpha_1} < \alpha < \omega^{\alpha_1}(n_1 + 1)$. This shows that $\alpha$ is not closed under addition. □

While addition of ordinals is a natural and useful operation, it has a few shortcomings, in particular its lack of commutativity. Less well-used is the operation of "commutative addition" (as termed for instance in [4]), based on Cantor normal form.

Let $\alpha_1 > \alpha_2 > \cdots > \alpha_k$. Let $\beta = \omega^{\alpha_1} n_1 + \omega^{\alpha_2} n_2 + \cdots + \omega^{\alpha_k} n_k$, and $\gamma = \omega^{\alpha_1} m_1 + \omega^{\alpha_2} m_2 + \cdots + \omega^{\alpha_k} m_k$, where of course $n_i, m_i < \omega$, but we allow some $n_i, m_i = 0$. We let

$$\beta \oplus \gamma = \omega^{\alpha_1}(n_1 + m_1) + \omega^{\alpha_2}(n_2 + m_2) + \cdots + \omega^{\alpha_k}(n_k + m_k).$$

Cantor normal form allows us to define $\beta \oplus \gamma$ for all ordinals $\beta$ and $\gamma$: we extend their Cantor normal form to a presentation as above with a common sequence of decreasing exponents by adding zero coefficients; for any sequence of exponents, this presentation is unique.

Moreover, canonicity of our fixed computable well-orderings implies that the operation $\oplus$ for pairs of ordinals below $\varepsilon_0$ is computable.

LEMMA 3.14. *Let $\alpha$, $\beta$ and $\gamma$ be ordinals.*

(1) $\beta \oplus \gamma = \gamma \oplus \beta$.
(2) $\alpha \oplus (\beta \oplus \gamma) = (\alpha \oplus \beta) \oplus \gamma$.

PROOF. Quite straightforward, based on the commutativity and associativity of addition of natural numbers. For associativity, the point is that if $\alpha_1 > \alpha_2 > \dots \alpha_k$ mentions all exponents of $\omega$ in the Cantor normal forms of $\alpha$, $\beta$ and $\gamma$, and

$$\alpha = \omega^{\alpha_1} n_1 + \cdots + \omega^{\alpha_k} n_k,$$
$$\beta = \omega^{\alpha_1} m_1 + \cdots + \omega^{\alpha_k} m_k,$$
$$\gamma = \omega^{\alpha_1} l_1 + \cdots + \omega^{\alpha_k} l_k,$$

then

$$(\alpha \oplus \beta) \oplus \gamma = \alpha \oplus (\beta \oplus \gamma) = \omega^{\alpha_1}(n_1 + m_1 + l_1) + \cdots + \omega^{\alpha_k}(n_k + m_k + l_k). \qquad \square$$

The associativity and commutativity of $\oplus$ allows us to unambiguously define $\bigoplus A$ for finite multisets of ordinals $A$.

Lemma 3.15. *Any power of $\omega$ is closed under $\oplus$.*

Proof. Let $\beta = \omega^{\alpha_1} n_1 + \cdots + \omega^{\alpha_k} n_k$ and $\gamma = \omega^{\alpha_1} m_1 + \cdots + \omega^{\alpha_k} m_k$ be smaller than $\omega^\delta$. Then for all $i \leqslant k$, $\omega^{\alpha_i} n_i, \omega^{\alpha_i} m_i < \omega^\delta$. Since $\omega^\delta$ is closed under addition, it follows that $\beta \oplus \gamma < \omega^\delta$. $\square$

Lemma 3.16. *Let $\beta_1, \beta_2, \dots, \beta_n$ and $\gamma_1, \gamma_2, \dots, \gamma_n$ be two $n$-tuples of ordinals. Suppose that for all $i \leqslant n$, $\beta_i \leqslant \gamma_i$. Then $\bigoplus_{i \leqslant n} \beta_i \leqslant \bigoplus_{i \leqslant n} \gamma_i$, and $\bigoplus_{i \leqslant n} \beta_i < \bigoplus_{i \leqslant n} \gamma_i$ if and only if there is some $i \leqslant n$ such that $\beta_i < \gamma_i$.*

Proof. Again, this is known and quite straightforward, but we give details for completeness of our presentation. Let $\alpha_1 > \alpha_2 > \cdots > \alpha_k$ be the exponents of $\omega$ appearing in the Cantor normal form of any of the $\beta_i$'s and $\gamma_i$'s; let $\beta_i = \sum_{j \leqslant k} \omega^{\alpha_j} n_{i,j}$ and $\gamma_i = \sum_{j \leqslant k} \omega^{\alpha_j} m_{i,j}$. So $\bigoplus_{i \leqslant n} \beta_i = \sum_{j \leqslant k} \left(\omega^{\alpha_j} \sum_{i \leqslant n} n_{i,j}\right)$, and $\bigoplus_{i \leqslant n} \gamma_i = \sum_{j \leqslant k} \left(\omega^{\alpha_j} \sum_{i \leqslant n} m_{i,j}\right)$.

If $\bigoplus_{i \leqslant n} \beta_i = \bigoplus_{i \leqslant n} \gamma_i$ then by the uniqueness of Cantor normal form, for all $j \leqslant k$, $\sum_{i \leqslant n} n_{i,j} = \sum_{i \leqslant n} m_{i,j}$. By induction on $j \leqslant k$, we show that for all $i$, $n_{i,j} = m_{i,j}$; it would follow that for all $i$, $\beta_i = \gamma_i$. Fix $j$, and suppose that for all $j' < j$, for all $i \leqslant n$, $n_{i,j'} = m_{i,j'}$. Since $\beta_i \leqslant \gamma_i$, the induction assumption implies that $n_{i,j} \leqslant m_{i,j}$. Now $\sum_{i \leqslant n} n_{i,j} = \sum_{i \leqslant n} m_{i,j}$ implies that for all $i \leqslant n$, $n_{i,j} = m_{i,j}$.

Suppose that $\bigoplus_{i \leqslant n} \beta_i \neq \bigoplus_{i \leqslant n} \gamma_i$. Let $j$ be the least index such that $\sum_{i \leqslant n} n_{i,j} \neq \sum_{i \leqslant n} m_{i,j}$. An induction as in the previous paragraph shows that for all $j' < j$, for all $i \leqslant n$, $n_{i,j'} = m_{i,j'}$. This information, together with the fact that $\beta_i \leqslant \gamma_i$ for all $i$, shows that for all $i$, $n_{i,j} \leqslant m_{i,j}$, and so that $\sum_{i \leqslant n} n_{i,j} \leqslant \sum_{i \leqslant n} m_{i,j}$. Since $\sum_{i \leqslant n} n_{i,j} \neq \sum_{i \leqslant n} m_{i,j}$, we must have $\sum_{i \leqslant n} n_{i,j} < \sum_{i \leqslant n} m_{i,j}$. The choice of $j$ now shows that $\bigoplus_{i \leqslant n} \beta_i < \bigoplus_{i \leqslant n} \gamma_i$ $\square$

The operation of commutative addition allows us to show that if $\alpha \leqslant \varepsilon_0$ is closed under addition, then the $\alpha$-c.a. functions induce an initial segment of the weak truth-table degrees.

Proposition 3.17. *Let $\alpha \leqslant \varepsilon_0$. If $f \colon \omega \to \omega$ is $\omega^\alpha$-c.a. and $g \leqslant_{\mathrm{wtt}} f$, then $g$ is $\omega^\alpha$-c.a.*

Proof. Let $\langle f_s, o_s \rangle_{s<\omega}$ be an $\omega^\alpha$-computable approximation of $f$, and let $(\Phi, \varphi)$ be a weak Truth-table functional such that $\hat{\Phi}(f) = g$. For any $x, s < \omega$, we recursively define a strictly increasing sequence $\langle t_s(x) \rangle_{s<\omega}$ of stages such that for all $s$, $\hat{\Phi}(f_{t_s(x)}, x)\!\downarrow$. Let $g_s(x) = \hat{\Phi}(f_{t_s(x)}, x)$, and let $m_s(x) = \bigoplus_{y < \varphi(x)} o_{t_s(x)}(y)$. Then $\langle g_s, m_s \rangle$ is an $\omega^\alpha$-computable approximation of $g$: by Lemma 3.15, for all $x$ and $s$, $m_s(x) < \omega^\alpha$, and by Lemma 3.16, for all $x$ and $s$, $m_{s+1}(x) \leqslant m_s(x)$ and if $g_{s+1}(x) \neq g_s(x)$ then $m_{s+1}(x) < m_s(x)$, because $f_{t_{s+1}(x)} \restriction_{\varphi(x)} \neq f_{t_s(x)} \restriction_{\varphi(x)}$. $\square$

**3.4. The complexity of the iterated bounded jump.** We wish to establish the following:

Proposition 3.18. *For all $\alpha \leqslant \varepsilon_0$, $I_\alpha^\varnothing$ is $\omega^\alpha$-c.a.*

As a result, by Proposition 3.9 and Proposition 3.17, $\varnothing^{\langle\alpha\rangle}$ is also $\omega^\alpha$-c.a.

Proposition 3.18 is proved by effective transfinite recursion, which means that the $\omega^\alpha$-computable approximation for $I_\alpha(\varnothing)$ will be given uniformly in $\alpha$. That is, by effective transfinite recursion on $\varepsilon_0$, we will show that there is a computable function $R\colon \varepsilon_0 \to \omega$ such that for all $\alpha < \varepsilon_0$, $\left\langle f_s^{R(\alpha),\omega^\alpha}, o_s^{R(\alpha),\omega^\alpha}\right\rangle$ is a $\omega^\alpha$-computable approximation of $I_\alpha^\varnothing$; recall that $\langle\langle f_s^{e,\alpha}, o_s^{e,\alpha}\rangle_{s<\omega}\rangle_{e<\omega}$ is an effective list, uniform in $\alpha$, of all tidy $(\alpha+1)$-computable approximations (Proposition 1.7).

The following two lemmas correspond to two of the three cases in the definition of $R$.

Lemma 3.19. *Let $\alpha < \varepsilon_0$. If $g$ is an $\omega^\alpha$-c.a. function, then $I^g$ is an $(\omega^\alpha+1)$-c.a. function. From $\alpha$, and an index of an $\omega^\alpha$-computable approximation of a function $g$, we can effectively obtain an index of an $(\omega^\alpha+1)$-computable approximation of $I^g$.*

Proof. Let $\langle g_s, o_s\rangle$ be an $\omega^\alpha$-computable approximation of $g$. For $e, x, s < \omega$, if $\hat\Phi_e(g_s, x)$ converges in $s$ many steps, we let $h_s(e,x) = 1 + \hat\Phi_e(g_s,x)$; otherwise, we let $h_s(e,x) = 0$. Then $\langle h_s\rangle$ is a computable approximation of $I^g$. We may assume that for all $e$ and $x$, $h_0(e,x) = 0$.

Fix $e, x < \omega$. For all $s < \omega$, let $r_s(e,x)$ be the least $r \leqslant s$ such that for all $t \in [r,s]$, $h_t(e,x) = h_s(x)$. We define a function $m_s(e,x)$:

- If $r_s(e,x) = 0$, let $m_s(e,x) = \omega^\alpha$.
- If $r_s(e,x) > 0$ then we know that $\varphi_e(x)\downarrow$. There are two sub-cases:
  - If $h_s(e,x) > 0$, then we let
  $$m_s(e,x) = \bigoplus_{y<\varphi_e(x)} \left(o_s(y) \oplus o_s(y)\right).$$
  - If $h_s(e,x) = 0$, then we let
  $$m_s(e,x) = \bigoplus_{y<\varphi_e(x)} \left(o_s(y) \oplus o_{r_s(e,x)-1}(y)\right).$$

By Lemma 3.15, for all $e, x$ and $s$, $m_s(e,x) \leqslant \omega^\alpha$. We show that $\langle h_s, m_s\rangle$ is an $(\omega^\alpha+1)$-computable approximation. Fix $e, x, s < \omega$.

If $h_{s+1}(e,x) = h_s(e,x)$, then $r_s(e,x) = r_{s+1}(e,x)$. In the three cases for defining $m_s(e,x)$ and $m_{s+1}(e,x)$, Lemma 3.16, and the fact that $o_{s+1}(y) \leqslant o_s(y)$ for all $y$, implies that $m_{s+1}(e,x) \leqslant m_s(e,x)$.

Now suppose that $h_{s+1}(e,x) \neq h_s(e,x)$; we want to show that $m_{s+1}(e,x) < m_s(e,x)$. Note that $r_{s+1}(e,x) = s+1$; let $r = r_s(e,x)$. There are four cases.

(1) If $r = 0$, then $m_s(e,x) = \omega^\alpha$ and $m_{s+1}(e,x) < \omega^\alpha$.

(2) Suppose that $r > 0$ and that $h_{s+1}(e,x) = 0$. Then $h_s(e,x) > 0$. This means that $\hat\Phi_e(g_s,x)$ converges in $s$ steps, but that $\hat\Phi_e(g_{s+1},x)$ does not converge in $s+1$ steps; so necessarily $g_{s+1}\restriction_{\varphi_e(x)} \neq g_s\restriction_{\varphi_e(x)}$. So there is some $y < \varphi_e(x)$ such that $o_{s+1}(y) < o_s(y)$. We have
$$m_s(e,x) = \bigoplus_{y<\varphi_e(x)} \left(o_s(y) \oplus o_s(y)\right),$$
and
$$m_{s+1}(e,x) = \bigoplus_{y<\varphi_e(x)} \left(o_{s+1}(y) \oplus o_s(y)\right).$$

The desired inequality then follows from Lemma 3.16.

(3) Suppose that $r > 0$, that $h_{s+1}(e,x) > 0$, and that $h_s(e,x) > 0$. Then $h_{s+1}(e,x) \neq h_s(e,x)$ implies that $g_{s+1}\restriction_{\varphi_e(x)} \neq g_s\restriction_{\varphi_e(x)}$, so again, there is some $y < \varphi_e(x)$ such that $o_{s+1}(y) < o_s(y)$. We have

$$m_s(e,x) = \bigoplus_{y<\varphi_e(x)} (o_s(y) \oplus o_s(y)),$$

and

$$m_{s+1}(e,x) = \bigoplus_{y<\varphi_e(x)} (o_{s+1}(y) \oplus o_{s+1}(y)),$$

so again $m_{s+1}(e,x) < m_s(e,x)$.

(4) The last case is that $r > 0$, $h_{s+1}(e,x) > 0$ and $h_s(e,x) = 0$. Now the point is that $h_{r-1}(e,x) > 0$, so the argument in case (2) show that there is some $y < \varphi_e(x)$ such that $o_r(y) < o_{r-1(y)}$, whence $o_{s+1}(y) < o_{r-1}(y)$. We have

$$m_{s+1}(e,x) = \bigoplus_{y<\varphi_e(x)} (o_{s+1}(y) \oplus o_{s+1}(y))$$

and

$$m_s(e,x) = \bigoplus_{y<\varphi_e(x)} (o_s(y) \oplus o_{r-1}(y)),$$

so we get the required inequality in this case too. $\square$

Lemma 3.20. *There is a computable function $T$ such that for any limit ordinal $\alpha \leqslant \varepsilon_0$ and $a < \omega$, if for some $g$, for all $\beta < \alpha$, $\left\langle f_s^{\varphi_a(\beta),\omega^\beta}, o_s^{\varphi_a(\beta),\omega^\beta}\right\rangle_{s<\omega}$ is a total $\omega^\beta$-computable approximation of $I^g_\beta$, then $\left\langle f_s^{\omega^\alpha,T(\alpha,a)}, o_s^{\omega^\beta,T(\alpha,a)}\right\rangle_{s<\omega}$ is an $\omega^\alpha$-computable approximation of $I^g_\alpha$.*

Proof. Given $\alpha$ and $a$, define $\langle h_s, m_s\rangle$ by letting $h_s(\beta,x) = f_s^{\varphi_a(\beta),\omega^\beta}(x)$ and $m_s(\beta,x) = o_s^{\varphi_a(\beta),\omega^\beta}(x)$, if $\varphi_a(\beta)\downarrow$ and $o_s^{\varphi_a(\beta),\omega^\beta}(x)\downarrow$, otherwise we let $h_s(\beta,x)$ and $m_s(\beta,x)$ diverge. For $z \neq \beta$ for any $\beta < \alpha$, we of course let $h_s(z,x) = m_s(z,x) = 0$. By the acceptability of the list of tidy $(\omega^\alpha+1)$-computable approximations, we can define $T(\alpha,a)$ such that if $\varphi_a$ is total, and for all $\beta<\alpha$, $\left\langle o_s^{\varphi_a(\beta),\omega^\beta}\right\rangle$ is total, then $\left\langle o_s^{\omega^\alpha,T(\alpha,a)}\right\rangle$ is total, and $\lim_s f_s^{\omega^\alpha,T(\alpha,a)} = \lim_s h_s$. $\square$

Proposition 3.18 now follows by effective transfinite recursion.

**3.5. Reducing $\omega^\alpha$-c.a. sets and functions to iterations of the wtt-jump.** Let $\alpha < \varepsilon_0$. An *instance of an $\omega^\alpha$-computable approximation* is a pair $(f,o)$ of computable functions $f\colon \omega \to \omega$ and $o\colon \omega \to \omega^\alpha$ such that for all $s < \omega$, $o(s+1) \leqslant o(s)$ and if $f(s+1) \neq f(s)$ then $o(s+1) < o(s)$.

As is done in the proof of Proposition 1.7, we can list, uniformly in $\alpha$, tidy instances of $(\omega^\alpha+1)$-computable approximations. In other words, there is an effective list $\langle f^\alpha_e, o^\alpha_e\rangle$ of pairs of computable functions with the following properties:

(1) For every $\alpha < \varepsilon_0$ and every $e < \omega$, $(f^\alpha_e, o^\alpha_e)$ is an instance of an $(\omega^\alpha+1)$-computable approximation with $f^\alpha_e(0) = 0$ and $o^\alpha_e(0) = \omega^\alpha$;

(2) The listing is *acceptable*: there is a (total) computable function $c(\alpha, d, e)$ such that for all $\alpha \leqslant \varepsilon_0$ and $d, e < \omega$, if $(\varphi_d, \varphi_e)$ is an instance of an $\omega^\alpha$-computable approximation, then $\lim_s \varphi_d(s) = \lim_s f^\alpha_{c(\alpha,d,e)}$.

Again the idea is to convert any pair $(g, m)$ of partial computable functions into a pair functions $(f, o)$ as in (1). We enumerate the graphs of $m$ and $g$ until we see that both $m(0)$ and $g(0)$ converge. As long as we don't see convergence, both $f$ and $o$ are constant; otherwise we slowly copy the values that we see.

Given the lists $\langle f^\alpha_e, o^\alpha_e\rangle$, we define the following for $\alpha \leqslant \varepsilon_0$:

(1) $C_\alpha = \{e : \exists s\ (o^\alpha_e(s) < \omega^\alpha)\}$. The sets $C_\alpha$ are c.e., uniformly in $\alpha$.
(2) Partial functions $F_\alpha \colon C_\alpha \to \omega$ by letting, for $e \in C_\alpha$, $F_\alpha(e) = \lim_s f^\alpha_e(s)$.

LEMMA 3.21. *For every $\alpha \leqslant \varepsilon_0$, $F_\alpha$ is partial bounded $I^\varnothing_\alpha$-computable.*

Lemma 3.21 is proved by effective transfinite recursion on $\varepsilon_0 + 1$, so again it has to be uniform: we construct a computable function $R$ such that for all $\alpha \leqslant \varepsilon_0$,

$$\hat{\Phi}_{R(\alpha)}\left(I^\varnothing_\alpha\right) = F_\alpha.$$

The following three lemmas explain how to define $R(\alpha)$ for the three kinds of ordinals $\alpha$: $\alpha = 0$, successor $\alpha$, and limit $\alpha$.

LEMMA 3.22. *$F_0$ is a partial computable function, and so is partial bounded $\varnothing$-computable.*

PROOF. For each $e \in C_0$, $F_0(e) = f^0_e(n)$ for $n$ such that $o^0_e(n) = 0$. □

LEMMA 3.23. *There is a computable function $S$ such that for all $\alpha < \varepsilon_0$ and all $a < \omega$, for any function $g \colon \omega \to \omega$, if $\hat{\Phi}_a(g) = F_\alpha$, then $\hat{\Phi}_{S(\alpha,a)}(I^g) = F_{\alpha+1}$.*

PROOF. We show how to define, effectively from $\alpha$ and $a$, a weak truth-table functional $(\Psi, \psi)$ such that for any function $g$, if $\hat{\Phi}_a(g) = F_\alpha$ then $\hat{\Psi}(I^g) = F_{\alpha+1}$. The acceptability of the enumeration of weak truth-table functionals then allows us to effectively find an index $S(\alpha, a)$ such that $(\Psi, \psi) = (\Phi_{S(\alpha,a)}, \varphi_{\alpha,a})$.

Let $e < \omega$. If $e \notin C_{\alpha+1}$, then we leave $\psi(e)\uparrow$, and for any oracle $g$, $\Psi(g, e)\uparrow$.

Suppose that $e \in C_{\alpha+1}$. For abbreviation, let $(f, o) = (f^{\alpha+1}_e, o^{\alpha+1}_e)$. The idea now is to break up the instance $(f, o)$ into a finite sequence of instances, each within a copy of $\omega^\alpha$ sitting inside $\omega^{\alpha+1}$. Let $s^*$ witness that $e \in C_{\alpha+1}$: $o(s^*) < \omega^{\alpha+1}$. Since $\omega^{\alpha+1} = \omega^\alpha \cdot \omega$, we can write, for $s \geqslant s^*$, $o(s) = \omega^\alpha n(s) + \beta(s)$ for unique $n(s) < \omega$ and $\beta(s) < \alpha$.

Let $M = n(s^*)$. For $m \leqslant M$ we define an instance $(f^m, o^m)$ of an $\omega^\alpha$-computable approximation by copying $\beta(s)$ on stages on which $n(s) = m$. Namely let $J_m = \{s \geqslant s^* : n(s) = m\}$. Then $J_M < J_{M-1} < \cdots < J_k$ is a partition of $[s^*, \omega)$ for some $k \leqslant M$; let us assume that $J_m$ for $m \geqslant k$ is nonempty (i.e. the approximation $(f, o)$ does not skip over the $m^{\text{th}}$ copy of $\omega^\alpha$); this is easily arranged. For $m \geqslant k$ we define $f^m(s) = f(s)$ and $o^m(s) = \beta(s)$ for $s \in J_m$, and extend in aconstant way otherwise (i.e., for $s < J_m$, $f^m(s) = f(\min J_m)$ and $o^m(s) = \beta(\min J_m)$; and if $m > k$ and $s > J_m$, we define similarly but with $\max J_m$ replacing $\min J_m$). For $m < k$ we leave $f^m(s)$ and $o^m(s)$ undefined for all $s$. The point is of course that $\lim_s f^k(s) = F_{\alpha+1}(e)$.

By the acceptability of the list $\langle f^\alpha_d, o^\alpha_d\rangle$, we can effectively get numbers $d_m$ for $m \leqslant M$ such that for all $m \leqslant M$,

- $d_m \in C_\alpha$ iff $m \geqslant k$; and

- If $m \geqslant k$, then $F_\alpha(d_m) = \lim_s f^m(s)$.

Now the procedure $\Psi$ queries the oracle on each pair $(a, d_m)$. The use is bounded by $\max\{(a, d_m) : m < M\}$; this is revealed to us once we see that $e \in C_{\alpha+1}$, so this use is partial computable (uniformly in $a$, $\alpha$ and $e$). If indeed $\hat{\Phi}_a(g) = F_\alpha$ then $I^g(a, d_m) = 0$ iff $m < k$ and $I^g(a, d_k) = 1 + F_{\alpha+1}(e)$, so this is what $\Psi$ outputs. ☐

LEMMA 3.24. *There is a recursive function $T$ such that for any limit ordinal $\alpha$, $a < \omega$ and sequence $\langle g_\beta \rangle_{\beta<\alpha}$ of functions, if for all $\beta < \alpha$, $\hat{\Phi}_{\varphi_a(\beta)}(g_\beta) = F_\beta$, then $\hat{\Phi}_{T(\alpha,a)}\left(\bigoplus_{\beta<\alpha} g_\beta\right) = F_\alpha$.*

PROOF. Of course now the point is that $\omega^\alpha$ is the limit of the ordinals $\omega^\beta$ for $\beta < \alpha$. So given $e \in C_\alpha$ we can effectively find some $\beta < \alpha$ and some $s^*$ such that $o_e^\alpha(s^*) < \omega^\beta$, and so can translate $(f_e^\alpha, o_e^\alpha)$ to an instance of an $\omega^\beta$-computable approximation; so we can find some $d \in C_\beta$ such that $\lim_s f_e^\alpha(s) = \lim_s f_d^\beta(s)$. We can then find some number $\psi(e)$, effectively computed from $e$, such that from $\left(\bigoplus_{\beta<\alpha} g_\beta\right)\restriction_{\psi(e)}$ we can compute $g_\beta\restriction_{\varphi_a(d)}$, and so using $\Phi_a$ can output

$$\hat{\Phi}_a(g_\beta, d) = F_\beta(d) = F_\alpha(e)$$

as required. Again, all this can be coded by a functional $\Psi$, and by acceptability we can effectively find an index $T(\alpha, a)$ such that $(\Phi_{T(\alpha,a)}, \varphi_{T(\alpha,a)}) = (\Psi, \psi)$. ☐

Now effective transfinite recursion on $\varepsilon_0 + 1$, using Lemmas 3.22, 3.23, and 3.24, builds a computable function $R$ such that for all $\alpha \leqslant \varepsilon_0$, $\hat{\Phi}_{R(\alpha)}(I_\alpha^\varnothing) = F_\alpha$, and so proves Lemma 3.21.

PROOF OF PART (2) OF THEOREM 3.11. Let $\alpha \leqslant \varepsilon_0$. Proposition 3.18 states that $I_\alpha^\varnothing$ is $\omega^\alpha$-c.a. By Proposition 3.17, every function $g \leqslant_{\text{wtt}} I_\alpha^\varnothing$ is also $\omega^\alpha$-c.a.

For the converse, let $g$ be an $\omega^\alpha$-c.a. function; let $\langle g_s, m_s \rangle$ be an $\omega^\alpha$-computable approximation for $g$. For every $x < \omega$, the sequence $\langle g_s(x), m_s(x) \rangle_{s<\omega}$ is an instance of an $\omega^\alpha$-computable approximation, and so by acceptability of the numbering of the partial instances of such approximations, there is a computable function $h$ such that for all $x$, $h(x) \in C_\alpha$ and $g(x) = F_\alpha(h(x))$. By Lemma 3.21, there is a weak-truth table functional $(\Phi, \varphi)$ such that $F_\alpha = \hat{\Phi}(I_\alpha^\varnothing)$. Let $\psi(x) = \varphi(h(x))$, and for any oracle $f$, let $\Psi(f, x) = \Phi(f, h(x))$ with the same use. Then $\psi$ is total (as range $h \subseteq C_\alpha$), and $\hat{\Psi}(I_\alpha^\varnothing) = g$, so $g \leqslant_{\text{wtt}} I_\alpha^\varnothing$. ☐

PROOF OF PART (1) OF THEOREM 3.11. The proof of the backward direction is identical to the corresponding proof of part (2), because as mentioned after the statement of Proposition 3.18, Proposition 3.9 implies that $\varnothing^{\langle\alpha\rangle}$ is $\omega^\alpha$-c.a.

For $\alpha \leqslant \varepsilon_0$, define $D_\alpha \colon C_\alpha \to \{0, 1\}$ by letting $D_\alpha(e) = F_\alpha(e) \mod 2$. If $A$ is an $\omega^\alpha$-c.a. set, then there is a computable function $h \colon \omega \to C_\alpha$ such that for all $x$, $A(x) = D_\alpha(h(x))$. Hence to show that every $\omega^\alpha$-c.a. set is weak truth-table reducible to $\varnothing^{\langle\alpha\rangle}$, we show that $D_\alpha$ is a partial bounded $\varnothing^{\langle\alpha\rangle}$-computable function.

The proof follows the line of argument for Lemma 3.21. A computable function $R$ such that for all $\alpha$, $\hat{\Phi}_{R(\alpha)}\left(\varnothing^{\langle\alpha\rangle}\right) = D_\alpha$ is constructed by effective transfinite recursion on $\varepsilon_0 + 1$, once analogues of Lemmas 3.22, 3.23, and 3.24 are proved:

(1) $D_0$ is a partial computable function;

(2) There is a computable function $S$ such that for all $\alpha < \varepsilon_0$ and all $a < \omega$, for any set $A \in 2^\omega$, if $\hat{\Phi}_a(A) = D_\alpha$, then $\hat{\Phi}_{S(\alpha,a)}(A^\dagger) = D_{\alpha+1}$.

(3) There is a recursive function $T$ such that for any limit ordinal $\alpha$, $a < \omega$ and sequence $\langle A_\beta \rangle_{\beta<\alpha}$ of sets, if for all $\beta < \alpha$, $\hat{\Phi}_{\varphi_a(\beta)}(A_\beta) = D_\beta$, then $\hat{\Phi}_{T(\alpha,a)}\left(\bigoplus_{\beta<\alpha} A_\beta\right) = D_\alpha$.

(1) follows immediately from the definition of $D_0$ and Lemma 3.22. For (3) we can simply take the function given by the proof of Lemma 3.24. The only new ingredient is in the proof of (2). Again, given $\alpha$, $a$ and $e \in C_{\alpha+1}$, we let $(f, o) = (f_e^{\alpha+1}, o_e^{\alpha+1})$, and define the functions $\beta$ and $n$, the number $M$, the pairs $(f^m, o^m)$ for $m \leqslant M$ and the numbers $k$ and $d_m$ exactly as was done in the proof of Lemma 3.23. So $m \geqslant k$ if and only if $d_m \in C_\alpha$, and $D_{\alpha+1}(e) = D_\alpha(d_k)$.

The difficulty of course is that $A^\dagger$ does not tell us the value of $\hat{\Phi}_a(A, d_m) = D_\alpha(d_m)$, only whether $\hat{\Phi}_a(A, d_m)$ converges or not. But since the value is either 0 or 1, we can convert it to convergence or divergence of an auxiliary functional. That is, we can effectively calculate and index $b$ and numbers $c_m$ for $m \leqslant M$ such that for any oracle $X$, $\hat{\Phi}_b(X, c_m)\downarrow$ if and only if $d_m \in C_\alpha$ and $\hat{\Phi}_a(X, d_m)\downarrow= 1$; for the use we can let $\varphi_b(c_m) = \varphi_a(d_m)$. We then let

$$\psi(e) = 1 + \max\left\{(a, d_m), (b, c_m) \,:\, m \leqslant M\right\},$$

which is again partial computable; and for any oracle $Y \in 2^\omega$, we calculate, for $e \in C_{\alpha+1}$, $\hat{\Psi}(Y, e)$ by first finding the least $m$ such that $(a, d_m) \in Y$ (we diverge if there is none), and then output $Y(b, c_m)$. If $\hat{\Phi}_a(A) = D_\alpha$ and $e \in C_{\alpha+1}$ then the least $m$ such that $(a, d_m) \in A^\dagger$ is $k$, and

$$\Psi(A^\dagger, e) = A^\dagger(b, c_k) = \hat{\Phi}_a(A, d_k) = D_\alpha(d_k) = D_{\alpha+1}(e)$$

as required. $\square$

CHAPTER III

# The hierarchy of totally $\alpha$-c.a. degrees

The following is the central definition of this work. For $\alpha = \omega$, this definition was originally made by J.S. Miller (unpublished), and first investigated in detail in [17].

Definition. Let $\alpha \leqslant \varepsilon_0$. A Turing degree $\mathbf{d}$ is *totally $\alpha$-c.a.* if every function $f \in \mathbf{d}$ is $\alpha$-c.a.

## 1. Totally $\mathcal{R}$-c.a. degrees

Basic properties of totally $\alpha$-c.a. degrees are shared among totally $\mathcal{R}$-c.a. degrees, even when $\mathcal{R}$ is not canonical. For any computable well-ordering $\mathcal{R}$, we say that a Turing degree $\mathbf{d}$ is *totally $\mathcal{R}$-c.a.* if every function $f \in \mathbf{d}$ is $\mathcal{R}$-c.a.

We note the following:

Lemma 1.1. *Let $\mathcal{R}$ be a computable well-ordering. A degree $\mathbf{d}$ is totally $\mathcal{R}$-c.a. if and only if every $f \leqslant_{\mathrm{T}} \mathbf{d}$ is $\mathcal{R}$-c.a.*

Proof. Suppose that $\mathbf{d}$ is a totally $\mathcal{R}$-c.a. degree. Let $g \in \mathbf{d}$ be any function. Let $f \leqslant_{\mathrm{T}} \mathbf{d}$. Then $f \oplus g \in \mathbf{d}$, so $f \oplus g$ has an $\mathcal{R}$-computable approximation, from which we can get an $\mathcal{R}$-computable approximation for $f$. □

**1.1. Totally $\mathcal{R}$-c.a. degrees and low$_2$ degrees.** The following theorem shows that total $\mathcal{R}$-c.a.-ness is indeed a notion of lowness.

Theorem 1.2. *For any computable well-ordering $\mathcal{R}$, every totally $\mathcal{R}$-c.a. degree is* low$_2$.

Proof. Let $\mathcal{R}$ be a computable well-ordering. By Corollary 1.8, there is a $\mathbf{0}'$-computable sequence $\langle f^e \rangle_{e<\omega}$ consisting of all $\mathcal{R}$-c.a. functions. Using this sequence, it is easy to construct a $\mathbf{0}'$-computable function $f$ which dominates every $\mathcal{R}$-c.a. function. Hence if $\mathbf{d}$ is a totally $\mathcal{R}$-c.a. degree, then $f$ dominates all functions in $\mathbf{d}$. By a classic result of Martin's [48], $\mathbf{d}$ is low$_2$. □

Ershov's Theorem 2.1 can be extended to low$_2$ degrees.

Proposition 1.3. *Every $\Delta^0_2$,* low$_2$ *degree is totally $\mathcal{R}$-c.a. for some computable well-ordering $\mathcal{R}$ of order-type $\omega$.*

Proof. Let $\mathbf{d}$ be a $\Delta^0_2$, low$_2$ degree. The proof of Theorem 2.1 can be adapted once we give a uniform $\mathbf{0}'$-enumeration of all the functions reducible to $\mathbf{d}$.

Let $D \in \mathbf{d}$, and let $\langle D_s \rangle$ be a computable approximation of $D$.

Since $\mathbf{d}$ is low$_2$, the collection of $e$ such that $\Phi_e(D)$ is total is $\Sigma^0_3$; say $\Phi_e(D)$ is total iff $\exists x \forall y \exists z\, R(e,x,y,z)$ where $R$ is computable. For $e, x$ and $s < \omega$, let $y_s(e,x)$ be the greatest $y$ such that for all $y' \leqslant y$ there is some $z < s$ such that $R(e,x,y',z)$ holds. Now for all such $e,x$ and $s$, for $n < \omega$, let $f^{e,x}_s(n) = \Phi_e(D,n)[y_s(e,x)]$ if $n < \operatorname{dom} \Phi_e(D)[y_s(e,x)]$, and $f^{e,x}_s(n) = 0$ for other $n$ (recall that we write $\Phi_e(D)[s]$ for $\Phi_{e,s}(D_s)$).

If $x$ witnesses that $\Phi_e(D)$ is total, then $\lim_s f^{e,x}_s = \Phi_e(D)$; if not, then the sequence $\langle f^{e,x}_s \rangle_{s<\omega}$ is eventually constant. Hence, renumbering, we get a uniformly computable sequence $\left\langle \left\langle f^d_s \right\rangle_{s<\omega} \right\rangle_{d<\omega}$ of computable approximations, with the collection of limits $\{ f^d : d < \omega \}$ (where $f^d = \lim_s f^d_s$) consisting precisely of all the functions computable from $\mathbf{d}$.

Now we let $\mathcal{R}$ be the interspersed union of the well-orderings built in the proof of Theorem 2.1 for the approximations $\langle f^d_s \rangle$. We let

$$R = \left\{ (d,x,s) \in \omega \times \omega \times \omega \,:\, s = 0 \text{ or } f^d_s(x) \neq f^d_{s-1}(x) \right\},$$

and for $(d,x,s),(e,y,t) \in R$, let $(d,x,s) <_{\mathcal{R}} (e,y,t)$ if $\langle d,x \rangle < \langle e,y \rangle$ or if $(d,x) = (e,y)$ and $t < s$. The argument of the proof of Theorem 2.1 shows that $\mathcal{R} = (R, <_{\mathcal{R}})$ has order-type $\omega$ and that for every $d < \omega$, $\left\langle f^d_s \right\rangle_{s<\omega}$ can be extended to an $\mathcal{R}$-computable approximation. Hence $\mathbf{d}$ is totally $\mathcal{R}$-c.a. $\square$

The argument for Corollary 2.16 now shows:

Corollary 1.4. *Every $\Delta^0_2$, $\mathrm{low}_2$ degree is totally $\mathcal{R}_o$-c.a. for some notation $o \in \mathcal{O}$ for $\omega^2$.*

**1.2. C.e. degrees.** In this work we focus on totally $\alpha$-c.a. c.e. degrees, namely those totally $\alpha$-c.a. Turing degrees which contain a c.e. set.

The following result shows that the for c.e. degrees, the class of sets captures everything expressed by functions as far as approximations are concerned. Technically, this is the first application in this monograph of the *permitting* method, calibrated at the level of total $\mathcal{R}$-c.a.-ness.

Proposition 1.5. *Let $\mathcal{R}$ be a computable well-ordering. A c.e. degree $\mathbf{d}$ is totally $\mathcal{R}$-c.a. if and only if every set $Z \leqslant_{\mathrm{T}} \mathbf{d}$ is $\mathcal{R}$-c.a.*

The argument of Lemma 1.1 shows now that a c.e. degree is totally $\mathcal{R}$-c.a. if and only if every set $Z \in \mathbf{d}$ is $\mathcal{R}$-c.a.

Proof. Let $\mathbf{d}$ be a c.e. degree, and suppose that some $g \leqslant_{\mathrm{T}} \mathbf{d}$ is not $\mathcal{R}$-c.a. Since $\mathbf{d}$ is c.e., there is some computable approximation $\langle g_s \rangle$ of $g$ such that $\mathbf{d}$ computes the modulus of this approximation.

We construct $Z$ by giving a computable approximation $\langle Z_s \rangle$ for $Z$. Let $\langle \langle Z^e_s, o^e_s \rangle_{s<\omega} \rangle_{e<\omega}$ be an effective enumeration of tidy $(\mathcal{R}+1)$-computable approximations such that letting $Z^e = \lim_s Z^e_s$, the sequence $\langle Z^e \rangle$ enumerates the $\mathcal{R}$-c.a. sets. Further, as is clear from the construction in Proposition II.1.7, every $\mathcal{R}$-c.a. set appears as $Z^e$ for some $e$ such that the approximation $\langle Z^e_s, o^e_s \rangle$ is eventually $\mathcal{R}$-computable: for all $n$ there is some $s$ such that $o^e_s(n) \in R$.

To defeat the threat that $Z = Z^e$, we pick potential witnesses $x$ for this $e^{\text{th}}$ requirement, and try to ensure that $Z(x) \neq Z^e(x)$. Naturally, we examine the sequence $\langle Z^e_s(x) \rangle_{s<\omega}$, and if there is equality between $Z_s(x)$ and $Z^e_{s+1}(x)$, we will want to change the value of $Z(x)$. To keep $Z$ being computable from $D$, each such change must be permitted by $g$. We prompt $g$ to give us such a permission by making a threat of our own, of giving an $\mathcal{R}$-computable approximation for $g$.

Since permission will only be granted eventually, we need to appoint infinitely many followers for each requirement; to avoid unnecessary interaction between requirements, these all have to be distinct. Recall that $\langle \omega^{[e]} \rangle_{e<\omega}$ is a partition of $\omega$ into uniformly computable sets (which we often refer to as "columns").

We start by defining $Z_0 = \varnothing$. At stage $s$ we *wish to flip* $x \in \omega^{[e]}$ if $o^e_s(x) \in R$ and $Z^e_s(x) = Z_{s+1}(x)$. We are *allowed to flip* $x$ at stage $s$ if $g_{s+1} \upharpoonright_x \neq g_s \upharpoonright_x$. If we both wish to flip and are allowed to flip some $x$, then we flip it: we set $Z_{s+1}(x) = 1 - Z_s(x)$. Otherwise, we set $Z_{s+1}(x) = Z_s(x)$. This defines the sequence $\langle Z_s \rangle$.

Let $x < \omega$. If $g_s \upharpoonright_x = g_t \upharpoonright_x$ for all $s \geqslant t$ then $Z_s(x) = Z_t(x)$ for all $s \geqslant t$. Hence $\langle Z_s \rangle$ is a computable approximation of a set $Z$. In fact, since $\mathbf{d}$ computes the moduus for $\langle g_s \rangle$, $Z \leqslant_{\mathrm{T}} \mathbf{d}$.

To show that $Z$ is not $\mathcal{R}$-c.a. we show that if the approximation $\langle Z^e_s, o^e_s \rangle$ is eventually $\mathcal{R}$-computable then $Z \neq Z^e$. Fix such $e$ and suppose for a contradiction that $Z^e = Z$. We define a sequence $\langle h_s, m_s \rangle$ by recursion. For $y < \omega$ let $x$ be the least element of $\omega^{[e]}$ greater than $y$. For all $s$ let $m_s(y) = o^e_s(x)$. Start with $h_s(y) = 0$ for all $s < y$. Now if $m_s(y) = m_{s-1}(y)$ then let $h_s(y) = h_{s-1}(y)$; otherwise let $h_s(y) = g_s(y)$. Then $\langle h_s, m_s \rangle$ is an eventually $\mathcal{R}$-computable approximation for $h = \lim_s h_s$ (which is therefore $h$-c.a.); we show that $h = g$.

Suppose not. Again let $y < \omega$ and let $x$ be the least element of $\omega^{[e]}$ greater than $y$. Let $t$ be the stage at which the sequence $\langle m_s(y) \rangle$ stabilizes. So $h(y) = h_t(y) = g_t(y)$ (by minimality of $t$) and for all $s \geqslant t$, $Z^e_s(x) = Z^e_t(x) = Z^e(x)$. Suppose that $g(y) \neq g_t(y)$. Let $s$ be the least stage $s > t$ at which we see that $g_{s+1} \upharpoonright_x \neq g_s \upharpoonright_x$. We are permitted to flip $Z(x)$ at stage $s$, so $Z_{s+1}(x) \neq Z^e_{s+1}(x)$ (either because we flipped it at stage $s$, or we did not need to). By induction, at no later stage will we want to flip $x$, so $Z(x) \neq Z^e_{s+1}(x) = Z(x)$, contradicting the assumption that $Z = Z^e$. □

The fact that $\mathbf{d}$ is a c.e. degree is heavily used in the proof of Proposition 1.5. Barmpalias (unpublished) constructed a degree $\mathbf{d}$ such that every set $Z \in \mathbf{d}$ is $\omega$-c.a. (in fact, $\mathbf{d}$ is superlow), but some function $f \in \mathbf{d}$ is not $\omega$-c.a.

## 2. The first hierarchy theorem: totally $\omega^\alpha$-c.a. degrees

Let $\gamma < \alpha \leqslant \varepsilon_0$. Since every $\gamma$-c.a. function is also $\alpha$-c.a. (see Section II.2), every totally $\gamma$-c.a. degree is also totally $\alpha$-c.a. The question is when does this hierarchy collapse.

Theorem 2.1. *Let $\alpha \leqslant \varepsilon_0$. There is a totally $\alpha$-c.a. degree which is not totally $\gamma$-c.a. for any $\gamma < \alpha$ if and only if $\alpha$ is a power of $\omega$. If $\alpha$ is a power of $\omega$, then in fact there is a c.e. degree which is totally $\alpha$-c.a. but not totally $\gamma$-c.a. for any $\gamma < \alpha$.*

The first $\omega \cdot 2$ many levels of the hierarchy of totally $\alpha$-c.a. degrees are depicted in Figure 1.

For the forward direction of the first hierarchy theorem, we prove the following lemma. It is proved in generality greater than is currently necessary, but which will be useful later.

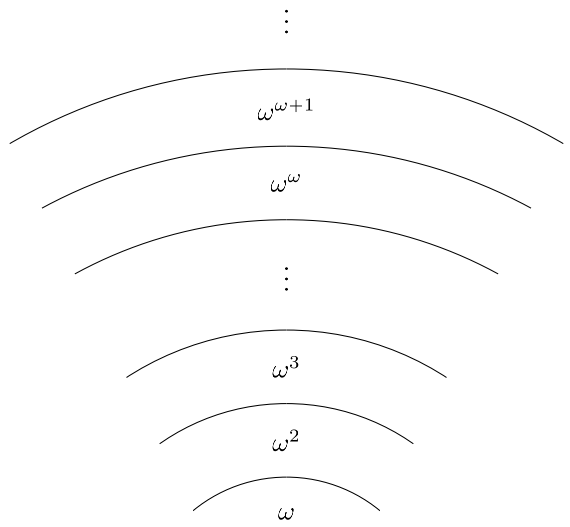


FIGURE 1. The first hierarchy theorem. "$\omega^\alpha$" denotes the collection of totally $\omega^\alpha$-c.a. degrees.

LEMMA 2.2. *Let $\gamma < \varepsilon_0$, and let $\mathbf{d}$ be a Turing degree such that every $g \in \mathbf{d}$ is $\gamma m$-c.a. for some $m < \omega$. Then $\mathbf{d}$ is totally $\gamma$-c.a.*

PROOF. Let $f \in \mathbf{d}$. Define $g(x) = f \restriction_x$; then $g \in \mathbf{d}$. By assumption, there is some $m < \omega$ such that $g$ is $\gamma m$-c.a. Let $\langle g_s, o_s \rangle$ be a $\gamma m$-computable approximation for $g$. By speeding up this approximation, we may assume that for all $x$ and $s$, $g_s(x)$ is a string of length $x$.

For every $x$ and $s$ there is some unique $k < m$ such that $o_s(x) \in [\gamma \cdot k, \gamma \cdot (k+1))$; we denote this $k$ by $k_s(x)$. We have $o_s(x) = \gamma \cdot k_s(x) + \beta_s(x)$ for some $\beta_s(x) < \gamma$. For every $x$ and $s$, $k_{s+1}(x) \leqslant k_s(x)$, and so $k_\omega(x) = \lim_s k_s(x)$ is well-defined. We let $k^* = \liminf_x k_\omega(x)$.

We can now give a $\gamma$-computable approximation $\langle f_s, m_s \rangle$ for $f$. Fix $x^*$ such that for all $x \geqslant x^*$, $k_\omega(x) \geqslant k^*$; so for all $s$ and all $x \geqslant x^*$, $k_s(x) \geqslant k^*$. For any $y < \omega$ we can effectively find some $x = h(y) > y$ such that $k_\omega(x) = k^*$, by insisting that $x \geqslant x^*$ and waiting until we see some stage $s$ such that $k_s(x) = k^*$. We let $t(y)$ be some stage $t$ such that $k_t(h(y)) = k^*$. Fix $y$, and let $x = h(y)$; we then let

$$m_s(y) = \beta_{\max\{s,t(y)\}}(x)$$

and

$$f_s(y) = \left(g_{\max\{s,t(y)\}}(x)\right)(y).$$

If $f_{s+1}(y) \neq f_s(y)$ then $s \geqslant t(y)$ and $g_{s+1}(x)) \neq g_s(x)$, and so $o_{s+1}(x) < o_s(x)$. Since $s \geqslant t(y)$, we have $o_s(x) = \gamma \cdot k^* + \beta_s(x)$ and $o_{s+1}(x) = \gamma \cdot k^* + \beta_{s+1}(x)$, and so $m_{s+1}(y) = \beta_{s+1}(x) < \beta_s(x) = m_s(y)$. Hence $\langle f_s, m_s \rangle$ is indeed a $\gamma$-computable approximation. If $g_s(x) = g(x)$ then $f_s(y) = (g(x))(y) = f(y)$, so $\lim_s f_s = f$. $\square$

The forward direction of Theorem 2.1 follows: if $\alpha$ is not a power of $\omega$, then there is some $\gamma < \alpha$ and some $m$ such that $\alpha \leqslant \gamma m$, and so every totally $\alpha$-c.a. degree is totally $\gamma m$-c.a., and so by Lemma 2.2 is actually totally $\gamma$-c.a.

The rest of this section is devoted to the proof of the backward direction of Theorem 2.1: given some $\alpha \leqslant \varepsilon_0$ which is a power of $\omega$, the construction of a c.e. degree which is totally $\alpha$-c.a. but not totally $\gamma$-c.a. for any $\gamma < \alpha$. Fix such $\alpha$. The key property of $\alpha$, which makes the construction work, is that $\alpha$ is closed under addition (Proposition II.3.13). We define a computable enumeration $\langle D_s \rangle$ of a c.e. set $D$, and ensure that $\deg_{\mathrm{T}}(D)$ is totally $\alpha$-c.a. but not totally $\gamma$-c.a. for any $\gamma < \alpha$.

To witness the properness, we enumerate a Turing functional $\Lambda$ and ensure that $\Lambda(D)$ is not $\gamma$-c.e. for any $\gamma < \alpha$. We fix, for each $\gamma < \alpha$, an enumeration $\langle \langle f_s^{e,\gamma}, o_s^{e,\gamma} \rangle_{s<\omega} \rangle_{e<\omega}$ of tidy $(\gamma+1)$-computable approximations whose limits $f^{e,\gamma} = \lim_s f_s^{e,\gamma}$ consist of all $\gamma$-c.a. functions (Proposition II.1.7). To show that $\Lambda(D)$ is not $\gamma$-c.a. for any $\gamma < \alpha$, it is sufficient to meet, for all $\gamma < \alpha$ and $e < \omega$, the requirement

**$P^{e,\gamma}$:** There is some $p$ such that $\Lambda(D,p) \neq f^{e,\gamma}(p)$.

Of course, we also need to ensure that $\Lambda(D)$ is total. To show that $\deg_{\mathrm{T}}(D)$ is totally $\alpha$-c.a., we need to meet, for all $e < \omega$, the requirement

**$Q_e$:** If $\Phi_e(D)$ is total, then it is $\alpha$-c.a.

*Discussion.* Perhaps surprisingly, the simplest construction one would hope work, does work. We give full details because several other constructions we present later are elaborations on this one.

First, independently consider the strategies for meeting each requirement. To meet $P^{e,\gamma}$, we pick a witness $p$ (also called a *follower*), and whenever we observe that $f_s^{e,\gamma}(p) = \Lambda_s(D_s,p)$, we change the value of $\Lambda(D,p)$ by enumerating the use $\lambda_s(p) = \lambda_s(D_s,p)$ into $D_{s+1}$. Recall our convention that since both $D$ and $\Gamma$ are defined by us, the use $\lambda(p)$ is the largest number actually queried during the computation. If this is performed without interruption, success is guaranteed, because our opponent can change the value of $f^{e,\gamma}(p)$ only finitely many times.

To meet $Q_e$, the only thing we can do is observe, for each input $x$, the value of $\Phi_{e,s}(D_s,x)$, and at various stages $s$ declare that we believe that $\Phi_e(D,x) = \Phi_{e,s}(D_s,x)$. If $\Phi_e(D)$ is total then we will eventually be right; we need to ensure, informally speaking, that the "number of times" we change our mind about the value of $\Phi_e(D,x)$ is bounded by $\alpha$. (Of course, technically we mean that we need to define a decreasing sequence of ordinals below $\alpha$ which is associated with the mind-changes. However, it is useful to think of $\alpha$ as bounding the number of mind-changes, in an analogy with the situation $\alpha = \omega$.) There is one possible action $Q_e$ can take, and that is to impose restraint: if we freeze $D$ below the use $\varphi_{d,s}(D_s,x)$, then our guess is correct.

The conflict between different requirements is now clear: when a requirement $P^{e,\gamma}$ enumerates $\lambda_s(p)$ into $D$, this may destroy a computation $\Phi_{d,s}(D,x)$ for some $d \leqslant e$ say, which $Q_d$ earlier declared it believed. The requirement $Q_d$ can tolerate some injury; after all, it is not trying to make $\Phi_d(D)$ computable. It needs to limit the "amount of injury" to be below $\alpha$. This is possible because once a follower $p$ is chosen, we can tell "how many times" the requirement $P^{e,\gamma}$ will act: the bound is $o_0^{e,\gamma}(p)$. Before starting to make guesses about $\Phi_d(D,x)$, the requirement $Q_d$ will observe which requirements will bother it and take their bounds $o_0^{e,\gamma}(p)$ into consideration. The fact that $\alpha$ is closed under addition means it can deal with injury from more than one other requirement.

This plan will not succeed if we allow requirements $P^{e,\gamma}$ to "gang up" on $Q_d$. Suppose that at some stage $s$, $Q_d$ starts making guesses about $\Phi_d(D,x)$, and declares an ordinal $\beta < \alpha$ bounding the "number of times" it will change its mind about this value. This bound $\beta$ is calculated on the basis of which followers $p$ for requirements $P^{e,\gamma}$ it is observing at stage $s$. It would be bad if we allow a different requirement $P^{e,\gamma}$ (say for some $e > s$) also destroy $\Phi_d(D,x)$: the bound on the action of such a requirement cannot be comprehended by $Q_d$ at stage $s$. Such requirements need to be restrained by $Q_d$: the numbers $\lambda_t(p)$ which they enumerate into $D$ must be greater than the use $\varphi_{d,s}(x)$.

On the face of it, this can be arranged using only finite injury: when $Q_d$ observes a new $\Phi_d(D,x)$ computation, it initialises all requirements $P^{e,\gamma}$ which are not allowed to injure this computation. The use $\lambda_t(p)$ for followers picked by these requirements later will be greater than $\varphi_{d,s}(x)$ as required. The reason that the injury will be finite is that it is guaranteed that the finitely many requirements which do have the right to injure $\Phi_d(D,x)$ only act at finitely many stages. Thus, it would seem, we would eventually either see a final computation $\Phi_d(D,x)$ and injury to weaker $P^{e,\gamma}$ on behalf of this computation will cease; or the computation $\Phi_d(D,x)$ never recovers, in which case also, eventually initialisation of weaker requirements will stop.

However, a complication arises from the combined influence of several negative requirements on some positive requirement. To see this, we first note that the permission to injure a computation that some $Q_d$ is monitoring is *follower-based* rather than *requirement-based*. Say that a positive requirement $P^{e,\gamma}$ picks a follower $p$. Then we see a computation $\Phi_d(D,x)$. Since $p$ is already chosen, $Q_d$ can observe $o_0^{e,\gamma}(p)$ and allow $P^{e,\gamma}$ to injure the computation. However, if for some reason later, $P^{e,\gamma}$ abandons the follower $p$ and replaces it by a new follower $p'$, the requirement $Q_d$ can no longer tolerate any action by $P^{e,\gamma}$: the ordinal $o_s^{e,\gamma}(p')$ may be much larger than $o_0^{e,\gamma}(p)$, and could not have been observed by $Q_d$ at the stage it first started copying $\Phi_d(D,x)$. In a sense, the requirement $P^{e,\gamma}$ is demoted (it loses priority) relative to the pair $(d,x)$.

Now consider such a positive requirement $P = P^{e,\gamma}$ and two negative requirements $Q_c$ and $Q_d$. Suppose that, by an action of a positive requirement stronger than $P$, $P$ is no longer allowed to destroy $\Phi_c(D,0)$, but that currently, $\Phi_c(D,0)\uparrow$. Meanwhile, $P$ has a follower $p_0$, and we observe $\Phi_d(D,0)$ for the first time. The follower $p_0$ is allowed to injure that computation, and that computation is indeed destroyed (by $P$ or by some weaker positive requirement). Then, we see that $\Phi_c(D,0)\downarrow$ with large use; this forces $P$ to cancel $p_0$ and appoint a new follower $p_1$. In turn, this means that $\Phi_d(D,0)$ no longer tolerates $P$-action. While $\Phi_d(D,0)\uparrow$, we see that $\Phi_c(D,1)\downarrow$, and it observes $p_1$; some action destroys the computation. We then see that $\Phi_d(D,0)\downarrow$, and $p_1$ is abandoned and replaced by a new follower $p_2$, and so $\Phi_c(D,1)$ can no longer tolerate $P$. The see-saw between $Q_c$ and $Q_d$ eventually causes infinitely much injury to $P$. Note that one negative requirement is not sufficient for this argument, as we assume that $\operatorname{dom}\Phi_d(D)$ is an initial segment of $\omega$.

The source of this problem is $P$'s haste in appointing a replacement follower. If it waited until $\Phi_c(D,0)$ converged before it appointed $p_0$, no injury would be necessary. For this to be possible, $P$ needs to guess whether $\Phi_c(D,0)$ will indeed converge in the future; if not, it will not wait. This necessitates the use of a tree of strategies in the construction.

*The tree of strategies.* As mentioned above (Section I.1), to define the tree, we specify recursively the association of nodes to requirements, and specify the outcomes of nodes working for particular requirements. To specify the priority ordering of nodes, we specify the ordering between outcomes of any node.

We order all the requirements, $Q_e$ and $P^{e,\gamma}$, in order-type $\omega$; all nodes of length $k$ work for the $k^{\text{th}}$ requirement on the list. The outcomes of a node working for $Q_e$ are $\infty$ and $\texttt{fin}$, with $\infty < \texttt{fin}$; a node working for $P^{e,\gamma}$ has only one outcome.

*Construction.* At stage $s$, we let the collection of accessible nodes $\delta_s$ be an initial segment of the tree of strategies.

Let $\sigma$ be a node which is accessible at stage $s$. We describe the action that $\sigma$ takes, and if it does not end the stage, then we specify which immediate successor of $\sigma$ is also accessible at stage $s$. Both of these depend, of course, on the requirement for which $\sigma$ works.

Suppose first that $\sigma$ works for $Q_e$. Then $\sigma$ takes no action beyond determining which successor is accessible. If $s$ is the least stage at which $\sigma$ is accessible, we let $\sigma\hat{\ }\infty \in \delta_s$. If not, let $t$ be the last stage before stage $s$ at which $\sigma\hat{\ }\infty$ was accessible. If $t < \operatorname{dom}\Phi_{e,s}(D_s)$ (again recall that we assume that $\operatorname{dom}\Phi_{e,s}(D_s)$ is an initial segment of $\omega$ (Convention I.1.2), and that we use von-Neumann natural number notation, Convention I.1.3), let $\sigma\hat{\ }\infty \in \delta_s$. Otherwise, we let $\sigma\hat{\ }\texttt{fin} \in \delta_s$.

Now suppose that $\sigma$ works for $P^{e,\gamma}$. As $\sigma$ has but one outcome, the determination of the next element of $\delta_s$ is immediate, unless $\sigma$ acts and ends the stage, in which case $\sigma$ is the last element of $\delta_s$. We let $\sigma$ act as follows:

(1) If $\sigma$ has no follower, then $\sigma$ appoints a new, large follower $p$ for itself.
(2) If $\sigma$ has a follower $p$, and $\Lambda_s(D_s,p) = f^{e,\gamma}_s(p)$, then $\sigma$ enumerates $\lambda_s(p)$ into $D_{s+1}$. We will later verify that $\lambda_s(p) \notin D_s$.

In either case, we set $\Lambda_{s+1}(D_{s+1},p) = s+1$ with large use. Technically, this means that we pick a large number $u$, and enumerate the axiom $D_{s+1}\restriction_u \mapsto (p,s)$ into $\Lambda_{s+1}$. The point of the value $s+1$ is that $\Lambda_{s+1}(D_{s+1},p) \neq f^{e,\gamma}_{s+1}(p)$, since by convention, for all $t$, $f^{e,\gamma}_t(p) < t$.

Also, in either case, we end the stage. If neither case (1) nor case (2) hold, then $\sigma$ does not act, and the unique immediate successor of $\sigma$ on the tree of strategies is accessible at stage $s$.

If $\sigma$ ended the stage, then all nodes that are weaker than $\sigma$ are *initialised*. For positive requirements $P^{e,\gamma}$, being initialised means that their followers are cancelled, and so at the next time they are visited, they have no follower and need to appoint a new one.

At the end of the stage, for each $p < s$ which is not at that moment a follower for some node on the tree, if $\Lambda_s(D_s,p)\uparrow$ then we set $\Lambda_{s+1}(D_{s+1},p) = 0$ with use $-1$. That is, we enumerate the axiom $\langle\rangle \mapsto (p,0)$ into $\Lambda_{s+1}$.

*Verification.* The following lemma will be familiar to experts in effective constructions, indeed, it is usually taken for granted and not mentioned explicitly. We give a careful and detailed presentation here, but will subsequently only sketch such proofs. For the following lemma, we first note that if $p$ is a follower for some node $\sigma$

at the beginning of stages $t < s$, and $p \in \operatorname{dom} \Lambda_t(D_t)$ and $p \in \operatorname{dom} \Lambda_s(D_s)$ (as we shall soon verify), then $p < \lambda_t(p) \leqslant \lambda_s(p)$, since $\lambda_r(p)$ is always chosen to be large.

LEMMA 2.3. *The functional $\Lambda$ is consistent for $D$. Further, at every stage $s$:*

(a) *$\Lambda_s$ is consistent for $D_s$.*

*Let $\sigma$ be a node which works for a positive requirement, and suppose that at the beginning of stage $s$, $\sigma$ has a follower $p$.*

(b) *$\Lambda_s(D_s, p)\downarrow$ and $\lambda_s(p) \notin D_s$.*

(c) *If $p'$ is, at the beginning of stage $s$, a follower for a node $\sigma'$ weaker than $\sigma$, then $\lambda_s(p) < p'$. And so $\lambda_s(p) < \lambda_s(p')$.*

(d) *Let $t < s$, and suppose that $p$ was a follower for $\sigma$ at the beginning of stage $t$. So $\sigma$ was not initialised at any stage $r \in [t, s)$. Then $D_t \restriction_{\lambda_t(p)} = D_s \restriction_{\lambda_t(p)}$. If, further, $\sigma$ does not act at any stage $r \in [t, s)$, then $D_t \restriction_{\lambda_t(p)+1} = D_s \restriction_{\lambda_t(p)+1}$ (this implies that $\lambda_s(p) = \lambda_t(p)$).*

PROOF. We prove (a), (b), (c) and (d) simultaneously by induction on $s$. Assume the lemma holds for $s - 1$; we consider the action taken at stage $s - 1$.

For (a) at stage $s$, we invoke Lemma I.1.1. Condition (1) of that lemma certainly holds at every stage of the construction. Condition (2) also holds: at stage $s-1$, at most one node $\sigma$ enumerates a new axiom into $\Lambda_s$ which pertains to its follower $p$; at the end of the stage we may enumerate further axioms, but only for numbers which are no longer followers, and so for numbers other than $p$. For condition (3), suppose that a new axiom pertaining to some number $p$ is added to $\Lambda_s$ during stage $s - 1$, but that $p \in \operatorname{dom} \Lambda_{s-1}(D_{s-1})$; we need to show that $\lambda_{s-1}(p)$ is enumerated into $D_s$. The assumption on $p$ implies that $p$ is not chosen as a new follower at stage $s - 1$. At the end of the stage we add axioms only for numbers $p \notin \operatorname{dom} \Lambda_{s-1}(D_{s-1})$; so it must be that $p$ is a follower for some node $\sigma$ at the beginning of stage $s-1$. Thus, $\sigma$ acts at stage $s-1$ and enumerates $\lambda_{s-1}(p)$ into $D_s$ (we use (b) at stage $s-1$); this shows condition (3) of Lemma I.1.1 holds. This shows that (a) holds at stage $s$ as well.

We next prove (d). Let $t$, $\sigma$ and $p$ be as described. Suppose that a number $y$ enters $D_{r+1}$ for some $r \in [t, s)$. Then $y = \lambda_r(p')$ for some follower $p'$ for some node $\sigma'$. Since $\sigma$ is not initialised at stage $r$, either $\sigma'$ is weaker than $\sigma$, in which case by (c) at stage $r$ we have $y > \lambda_r(p) \geqslant \lambda_t(p)$; or $\sigma' = \sigma$, in which case of course $y = \lambda_r(p) \geqslant \lambda_t(p)$. In either case, $D_t \restriction_{\lambda_t(p)} = D_s \restriction_{\lambda_t(p)}$. If $\sigma$ does not act at any stage $r \in [t, s)$ then we always have $y > \lambda_t(p)$ and so $D_t \restriction_{\lambda_t(p)+1} = D_s \restriction_{\lambda_t(p)+1}$.

To show (b) at stage $s$, let $p$ be a follower for a node $\sigma$ at the beginning of stage $s$. If $\sigma$ acts at stage $s - 1$, then at that stage we define $\Lambda_s(D_s, p)\downarrow$ with large use $\lambda_s(p)$; since it is large, we have $\lambda_s(p) \notin D_s$. Otherwise, $p$ is a follower for $\sigma$ at the beginning of stage $s - 1$, and $p$ is not cancelled at that stage. By (b) at stage $s - 1$, $\Lambda_{s-1}(D_{s-1}, p)\downarrow$. By (d) at stage $s$, with $t = s - 1$, we have $D_s \restriction_{\lambda_{s-1}(p)+1} = D_{s-1} \restriction_{\lambda_{s-1}(p)+1}$. This implies that the axiom making $\Lambda_{s-1}(D_{s-1}, p)\downarrow$ applies at stage $s$ as well, and in fact $\lambda_s(p) = \lambda_{s-1}(p)$. By (b) at stage $s - 1$ we have $\lambda_{s-1}(p) \notin D_{s-1}$, and the agreement between $D_{s-1}$ and $D_s$ just observed shows that $\lambda_s(p) = \lambda_{s-1}(p) \notin D_s$ as well.

For (c), let $p'$ and $\sigma'$ be as described. Let $t \leqslant s - 1$ be the stage at which $p'$ was chosen as a follower for $\sigma'$. The fact that the follower $p'$ is kept from stage $t+1$ up to stage $s$ shows that $\sigma'$ was not initialised at any stage $r \in [t, s)$. Since $\sigma$ is stronger than $\sigma'$, this shows that $\sigma$ was not initialised and did not act at any such stage. Thus, $p$ must have been appointed by $\sigma$ at a stage prior to stage $t$, and so $p$ is a follower for $\sigma$ at the beginning of stage $t$. At stage $t$, $p'$ is chosen to be large, and so $p' > \lambda_t(p)$ (the latter exists by (b) at stage $t$). By (d) (at stage $s$, applied to stage $t$), we see that $D_s \upharpoonright_{\lambda_t(p)+1} = D_t \upharpoonright_{\lambda_t(p)+1}$, whence $\lambda_s(p) = \lambda_t(p)$. □

We start by working toward showing that the construction is fair.

LEMMA 2.4. *Let $\sigma$ be a node which works for requirement $P^{e,\gamma}$. Let $s < t$ be stages, and suppose that $\sigma$ acts at both stages $s$ and $t$, and is not initialised at any stage $r \in (s,t)$. Let $p$ be the follower for $\sigma$ at the end of stage $s$. Then $o_t^{e,\gamma}(p) < o_s^{e,\gamma}(p)$.*

PROOF. The follower $p$ is not cancelled at any stage $r \in (s,t]$. In particular, $\sigma$'s action at stage $t$ is not appointing a new follower, and so this action is prompted by the equality $f_t^{e,\gamma}(p) = \Lambda_t(D_t,p)$.

We observe that $\Lambda_t(D_t,p) > s$. This follows from the fact that at stage $s$, we set $\Lambda_{s+1}(D_{s+1},p) = s+1$, and that at no later stage do we decrease the value of $\Lambda_r(D_r,p)$.

Now we have $f_t^{e,\gamma}(p) = \Lambda_t(D_t,p) > s$ and by convention, $f_s^{e,\gamma}(p) < s$. So $f_s^{e,\gamma}(p) \neq f_t^{e,\gamma}(p)$. Since $\langle f_s^{e,\gamma}, o_s^{e,\gamma}\rangle_{s<\omega}$ is a $(\gamma+1)$-computable approximation, and $f_r^{e,\gamma}(p)$ is not constant on $r \in [s,t]$, we must have $o_t^{e,\gamma}(p) < o_s^{e,\gamma}(p)$. □

Since for all $s$, $\delta_s$ is an initial segment of the tree of strategies, the true path $\delta_\omega$ is an initial segment of the tree. Since every node on the tree of strategies has but finitely many outcomes, the only thing that could stop the true path from being infinite is that some node on the true path acts and ends the stage at almost every stage it is accessible.

LEMMA 2.5. *Suppose that $\sigma$ is a node on the true path working for some positive requirement $P^{e,\gamma}$, and that the construction is fair to $\sigma$. Then $\sigma$ acts only finitely many times.*

PROOF. Let $s_0$ be the last stage at which $\sigma$ is initialised. Let $s_1$ be the least stage beyond $s_0$ at which $\sigma$ is accessible. At stage $s_1$, $\sigma$ appoints a follower $p$. Since $s_1 > s_0$, this follower is never cancelled.

The fact that $\sigma$ acts only finitely many times beyond stage $s_1$ now follows from Lemma 2.4. Since $\langle f_s^{e,\gamma}, o_s^{e,\gamma}\rangle$ is a $(\gamma+1)$-computable approximation, there is some stage $t \geqslant s_1$ after which $o_u^{e,\gamma}(p)$ is constant. Then $\sigma$ can act at most once after stage $t$. □

By induction on the length of nodes, we see that the construction is fair to every node on the true path, and so that no node can be the last node on the true path.

COROLLARY 2.6. *The true path $\delta_\omega$ is infinite, and the construction is fair to every node on the true path.*

Next, we show that the positive requirements are met, and so that $\Lambda(D)$ witnesses that $\deg_{\mathrm{T}}(D)$ is not totally $\gamma$-c.a. for any $\gamma < \alpha$.

LEMMA 2.7. *$\Lambda(D)$ is total.*

PROOF. Let $p < \omega$. Suppose that there is some stage $s_0 > p$ at which $p$ is not a follower for any node. After stage $s_0$, we enumerate an axiom into $\Lambda$ regarding $p$ at most once, because such an axiom has use $-1$ and so defines a computation that cannot be destroyed. So overall, only finitely many axioms in $\Lambda$ are made for $p$. Thus, if $p \notin \operatorname{dom}\Lambda(D)$, then at almost every stage $s$ we have $p \notin \operatorname{dom}\Lambda_s(D_s)$. But then at some such stage $s > s_0$ we would define $\Lambda_{s+1}(D_{s+1}, p)\downarrow$ with use $-1$, which would imply that $p \in \operatorname{dom}\Lambda(D)$ after all – contradiction.

Now suppose that $p$ is picked as a follower for some node $\sigma$, and that $p$ is never cancelled. The construction is fair to $\sigma$, and so either $\sigma$ lies to the left of the true path, or lies on the true path. In either case, $\sigma$ acts at most finitely many times (Lemma 2.5). Let $s-1$ be the last stage at which $\sigma$ acts. Lemma 2.3(d) now shows that $D\restriction_{\lambda_s(p)+1} = D_s\restriction_{\lambda_s(p)+1}$ and so $p \in \operatorname{dom}\Lambda(D)$. ☐

LEMMA 2.8. *Every positive requirement is met.*

PROOF. Let $P^{e,\gamma}$ be a positive requirement. Let $\sigma$ be a node on the true path which works for $P^{e,\gamma}$. As in the proof of Lemma 2.7 there is a last stage $s-1$ at which $\sigma$ acts, and at that stage we define a $D$-correct computation $\Lambda_s(D_s, p)$. If $\Lambda(D, p) = f^{e,\gamma}(p)$, then for almost all stages $t > s$ we would have $\Lambda_t(D_t, p) = f_t^{e,\gamma}(p)$. There is such a stage $t > s$ at which $\sigma$ is accessible. At such a stage, $\sigma$ would act – contradiction. ☐

We now need to show that $\deg_{\mathrm{T}}(D)$ is totally $\alpha$-c.a., that is, that every requirement $Q_e$ is met. Fix $e < \omega$, and suppose that $\Phi_e(D)$ is total; we give $\Phi_e(D)$ an $\alpha$-computable approximation.

Since the true path $\delta_\omega$ is infinite, there is some node $\tau \in \delta_\omega$ that works for the requirement $Q_e$. Let $s^*$ be the last stage at which the node $\tau$ is initialised (this is the same as the last stage at which the node $\tau\hat{}\infty$ is initialised). We let

$$S = \{s > s^* : \tau\hat{}\infty \in \delta_s\}.$$

Since $\Phi_e(D)$ is total, $S$ is infinite (so $\tau\hat{}\infty$ is on the true path) – a greatest stage in $S$ would yield a contradiction. Let $s_0, s_1, \ldots$ be the increasing enumeration of the (computable) set $S$.

For $x < \omega$, we let $i(x)$ be the least index $i$ such that $x < \operatorname{dom}\Phi_e(D)[s_i]$. For $j \geqslant i(x)$, we let $a_j(x)$ be the collection of nodes $\sigma \succcurlyeq \tau\hat{}\infty$ which at the beginning of stage $s_j$ have a follower $p = p(\sigma, x)$ which was chosen before stage $s_{i(x)}$. Note that for all $j \geqslant i(x)$, $a_{j+1}(x) \subseteq a_j(x)$. The next lemma says that only nodes in $a_j(x)$ can injure the computation $\Phi_e(D, x)[s_j]$.

LEMMA 2.9. *Let $j \geqslant i(x)$. Suppose that $\Phi_e(D, x)[s_{j+1}] \neq \Phi_e(D, x)[s_j]$. Then the weakest node in $a_{j+1}(x)$ acts at stage $s_j$.*

PROOF. There is some stage $r \in [s_j, s_{j+1})$ at which some node $\sigma$ enumerates a number smaller than $\varphi_{e,s_j}(x)$ into $D_{r+1}$, destroying the computation $\Phi_e(D, x)[s_j]$. Recall that since $\Phi_e$ is not enumerated by us, our convention is that $\varphi_e(x)$ is not the largest number queried but one greater, the length of the string in the axiom defining the computation. We show that $r = s_j$ and that $\sigma$ is the weakest node in $a_{j+1}(x)$.

Let $p$ be $\sigma$'s follower at stage $r$. Let $t$ be the stage at which $p$ was appointed. We have $\lambda_r(p) < \varphi_{e,s_j}(x)$, and so $t < s_j$.

Certainly, $\sigma$ cannot be stronger than $\tau\hat{}\infty$, since $r > s^*$. On the other hand, $\tau\hat{}\infty$ is accessible at stage $s_j$, and $\sigma$ is not initialised at stage $s_j$ (this would cancel

$p$), whence $\sigma$ must extend $\tau\hat{\ }\infty$. From this we already conclude that $r = s_j$, as $\sigma$ is not accessible at any stage in the interval $(s_j, s_{j+1})$.

Since $\sigma$ acts at stage $s_j$, all nodes weaker than $\sigma$ are initialised at stage $s_j$, and so no node weaker than $\sigma$ can have, at stage $s_{j+1}$, a follower chosen prior to stage $s_{i(x)}$. I.e., no node weaker than $\sigma$ can be an element of $a_{j+1}(x)$. To finish the proof of the lemma, it remains to show that $\sigma \in a_{j+1}(x)$, i.e., to show that $t < s_{i(x)}$, and that $\sigma$ is not initialised at some stage $r \in [s_j, s_{j+1})$. The latter is immediate: at stage $s_j$, $\sigma$ acts and so is not initialised; and at stage $r \in (s_j, s_{j+1})$, $\tau\hat{\ }\infty$ is not accessible, and so the fact that $\tau\hat{\ }\infty$ is not initialised at stage $r$ implies that neither is $\sigma$.

Suppose, for a contradiction, that $t \geqslant s_{i(x)}$. Since $\sigma$ extends $\tau\hat{\ }\infty$, we see that $t \in S$ and so that $x < \operatorname{dom} \Phi_e(D)[t]$. This is the crucial point for the entire construction: in this case every time we define $\lambda(p)$ we observe $\Phi_e(D, x)$, and so the former is larger than the use of the latter.

Let $u = \varphi_{e,t}(x)$. At stage $t$ we pick $\lambda_t(p) > u$. Since $\sigma$ is not initialised at any stage $r \in [t, s_j)$, Lemma 2.3(d) shows that $D_t \restriction_u = D_{s_j} \restriction_u$, which in turn implies that $\varphi_{e,s_j}(x) = u$. This contradicts $\lambda_{s_j}(p) < \varphi_{e,s_j}(x)$. □

Fix $x < \omega$. For $j \geqslant i(x)$ and $\sigma \in a_j(x)$ we let $t_j(\sigma)$ be the greatest stage $t < s_j$ at which $\sigma$ acts. Such a stage $t$ exists, because $\sigma$ acts when it appoints the follower $p(\sigma, x)$. We note for later that $\sigma$ is not initialised between stage $t_j(\sigma)$ and stage $s_j$. In fact, $t_j(\sigma) = s_i$ for some $i < j$, but this is not material.

For $j \geqslant i(x)$ and $\sigma \in a_j(x)$ we let $\beta_j(\sigma) = o^{i,\gamma}_{t_j(\sigma)}(p(\sigma, x))$, where $\sigma$ works for the requirement $P^{i,\gamma}$. We order the set $a_j(x)$ by descending priority to obtain a sequence, and let

$$m_j(x) = \sum_{\sigma \in a_j(x)} \beta_j(\sigma),$$

with the addition performed along the order of $a_j(x)$: if $a_j(x) = \langle \sigma_1, \sigma_2, \ldots, \sigma_k \rangle$ then $m_j(x) = \beta_j(\sigma_1) + \beta_j(\sigma_2) + \cdots + \beta_j(\sigma_k)$. We let $g_j(x) = \Phi_e(D, x)[s_j]$. Certainly $\lim_{j\to\infty} g_j(x) = \Phi_e(D, x)$.

Lemma 2.10. *Let $j \geqslant i(x)$. Then $m_{j+1}(x) \leqslant m_j(x) < \alpha$, and if $g_{j+1}(x) \neq g_j(x)$ then $m_{j+1}(x) < m_j(x)$.*

We then let $m_j(x) = m_{i(x)}(x)$ and $g_j(x) = g_{i(x)}(x)$ for all $j < i(x)$, and see that $\langle g_j, m_j \rangle$ is an $\alpha$-computable approximation for $\Phi_e(D)$.

Proof. First note that for each $j \geqslant i(x)$, for each $\sigma \in a_j(x)$, if $\sigma$ works for $P^{i,\gamma}$ then $\beta_j(\sigma) \leqslant \gamma < \alpha$; as $\alpha$ is closed under addition (here is where we use the assumption), $m_j(x) < \alpha$ for all $j$.

Next, we observe that thought of as sequences, $a_{j+1}(x)$ is an *initial segment* of $a_j(x)$. This is because if $\sigma \in a_j(x) \backslash a_{j+1}(x)$, then $\sigma$ is initialised at some stage $r \in [s_j, s_{j+1})$; at that stage $r$, every node weaker than $\sigma$ is also initialised and extracted from $a_{j+1}(x)$.

Now for each $\sigma \in a_{j+1}(x)$, $t_{j+1}(\sigma) \geqslant t_j(\sigma)$ and so $\beta_{j+1}(\sigma) \leqslant \beta_j(\sigma)$. Altogether, we see that $m_{j+1}(x) \leqslant m_j(x)$.

Suppose that $g_{j+1}(x) \neq g_j(x)$. Let $\sigma$ be the weakest node in $a_{j+1}(x)$. We know (Lemma 2.9) that $\sigma$ acts at stage $s_j$. Thus, $t_j(\sigma) < s_j = t_{j+1}(\sigma)$. Since $\sigma$ acts at both stage $t_j(\sigma)$ and stage $t_{j+1}(\sigma)$, and is not initialised between these stages, Lemma 2.4 says that $\beta_{j+1}(\sigma) < \beta_j(\sigma)$. Together with $\beta_{j+1}(\tau) \leqslant \beta_j(\tau)$ for all other $\tau \in a_{j+1}(x)$, and since $\beta_{j+1}(\sigma)$ is the last summand in $m_{j+1}(x)$, we see that $m_{j+1}(x) < m_j(x)$. □

### 3. A refinement of the hierarchy: uniformly totally $\omega^\alpha$-c.a. degrees

Downey, Jockusch and Stob [24] have shown that the following are equivalent for a c.e. degree $\mathbf{d}$:

1. (1) $\mathbf{d}$ is array computable;
2. (2) for every increasing computable function $h$, every function $f \in \mathbf{d}$ has an $h$-bounded computable approximation;
3. (3) there is some increasing computable function $h$ such that every function $f \in \mathbf{d}$ has an $h$-bounded computable approximation.

By Proposition II.1.12, every c.e., array computable degree is totally $\omega$-c.a. Note that the computable enumerablility of $\mathbf{d}$ is necessary here, as there are uncountably many array computable degrees.

The converse does not hold: there is a c.e. degree which is totally $\omega$-c.a. but not array computable. An indirect argument for the existence of such a degree is given by a conjunction of work by Walk [61] and Downey, Greenberg and Weber [17]. Walk constructed a c.e. degree which is not array computable, but does not bound a critical triple. Downey, Greenberg and Weber showed that such a degree must be totally $\omega$-c.a.

Theorem 3.5 gives a direct construction of a c.e. degree which is totally $\omega$-c.a. and not array computable, by finding a generalisation of the notion of array computability to all levels of the hierarchy of totally $\omega^\alpha$-c.a. degrees. We call this generalisation the *uniform version* of total $\omega^\alpha$-computable approximability. The key idea is the observation, mentioned above, that for ordinals $\alpha > \omega$, the first value $o_0(x)$ of an $\alpha$-computable approximation $\langle f_s, o_s\rangle$ is the correct measure of "how many times" the approximation $\langle f_s(x)\rangle_{s<\omega}$ changes, rather than the natural number $m_{\langle f_s\rangle}(x)$, the value of the mind-change function.

Definition 3.1. Let $\alpha \leqslant \varepsilon_0$.

An *$\alpha$-order function* is a non-decreasing computable function $h\colon \omega \to \alpha$ whose range is unbounded in $\alpha$.

Let $h$ be an $\alpha$-order function. An $\alpha$-computable approximation $\langle f_s, o_s\rangle$ is an *$h$-computable approximation* if for all $x$, $o_0(x) < h(x)$. In the language of Section II.3, for each $x$ the seqience $\langle f_s(x), o_s(x)\rangle$ is an instance of an $h(x)$-computable approximation.

We say that a function $f\colon \omega \to \omega$ is *$h$-computably approximable* (or *$h$-c.a.*) if there is an $h$-computable approximation $\langle f_s, o_s\rangle$ such that $\lim_s f_s = f$.

Note that for all $\alpha \leqslant \varepsilon_0$, $\alpha$-order functions exist; in fact, there is a computable, strictly increasing and unbounded function from $\omega$ to $\alpha$ (see Lemma II.2.11). This shows that a function is $\alpha$-c.a. if and only if it is $h$-c.a. for some $\alpha$-order function $h$.

The following uses an argument used by Terwijn and Zambella [60] in the context of computable traceability, and earlier by Downey, Jockusch and Stob [24].

Lemma 3.2. *The following are equivalent for a Turing degree $\mathbf{d}$ and $\alpha \leqslant \varepsilon_0$.*

1. (1) *There is some $\alpha$-order function $h$ such that every $f \in \mathbf{d}$ is $h$-c.a.*
2. (2) *For every $\alpha$-order function $h$, every $f \in \mathbf{d}$ is $h$-c.a.*

Proof. Let $h$ and $\bar{h}$ be $\alpha$-order functions. We show that for all $f\colon \omega \to \omega$ there is some $g \equiv_{\mathrm{T}} f$ such that if $g$ is $\bar{h}$-c.a., then $f$ is $h$-c.a.

The function $g$ is obtained by "stretching" $f$ along the composition of the "discrete inverse" of $h$ with $\bar{h}$. Namely, we (computably) partition $\omega$ into an increasing sequence of finite intervals $I^* < I_0 < I_1 < I_2 < \dots$ so that for all $n$, for all $x \in I_n$, $h(x) \geqslant \bar{h}(n)$. Some intervals $I_n$ are allowed to be empty (this is used when $\bar{h}$ is not injective). We simply let $I^*$ be the set of $x$ such that $h(x) < \bar{h}(0)$; and if

$$\bar{h}(n-1) < \bar{h}(n) = \cdots = \bar{h}(m) < \bar{h}(m+1)$$

(possibly $m = n$) then we let $I_n$ be the set of $x$ such that $\bar{h}(n) \leqslant h(x) < \bar{h}(m+1)$; this is finite since $h$ is unbounded in $\alpha$. For $k$ between $n$ and $m+1$ we let $I_k$ be empty.

We then define $g(n) = f\restriction_{I_n}$. Let $\langle g_s, o_s\rangle$ be an $\bar{h}$-computable approximation for $g$. By speeding up this approximation we may assume that for all $s$ and $n$, $g_s(n)$ is a function from $I_n$ to $\omega$. We can then define $f_s(x) = (g_s(n))(x)$ for $x \in I_n$ (and let $m_s(x) = o_s(n)$); for $x \in I^*$ we let $f_s(x) = f(x)$ and $m_s(x) = 0$. Then $\langle f_s, m_s\rangle$ is an $h$-computable approximation for $f$. □

Definition 3.3. A Turing degree $\mathbf{d}$ is *uniformly totally $\alpha$-c.a.* if for some (all) $\alpha$-order function(s) $h$, every $f \in \mathbf{d}$ is $h$-c.a.

The Downey, Jockusch and Stob characterisation shows that a c.e. degree is array computable if and only if it is uniformly totally $\omega$-c.a.

Lemma 3.4. *A Turing degree $\mathbf{d}$ is uniformly totally $\alpha$-c.a. if and only if for some (all) $\alpha$-order function $h$, every $f \leqslant_T \mathbf{d}$ is $h$-c.a.*

Proof. Suppose that $\mathbf{d}$ is uniformly totally $\alpha$-c.a., and let $h$ be an $\alpha$-order function. Let $f \leqslant_T \mathbf{d}$ and let $g \in \mathbf{d}$; so $f \oplus g \in \mathbf{d}$. Then $f \oplus g$ is $h \oplus h$-c.a.; it follows that $f$ is $h$-c.a. □

The argument of Proposition 1.5 shows that a c.e. degree is uniformly totally $\alpha$-c.a. if and only if for some (all) $\alpha$-order function $h$, every set in $\mathbf{d}$ is $h$-c.a.

We turn to investigate the distribution of uniformly totally $\alpha$-c.a. degrees in the hierarchy of totally $\alpha$-c.a. degrees. An immediate fact, using the constant function with value $\alpha$, is that for all $\alpha < \varepsilon_0$, every totally $\alpha$-c.a. degree is uniformly totally $(\alpha+1)$-c.a.

It follows from the easy direction of Theorem 2.1 that if $\beta \in (\omega^\alpha, \omega^{\alpha+1})$ (that is, if $\beta$ is not a power of $\omega$), then every uniformly totally $\beta$-c.a. degree is totally $\omega^\alpha$-c.a. Hence, if $\beta$ is not a power of $\omega$, then there is an ordinal $\alpha$ which is a power of $\omega$ such that the collection of uniformly totally $\beta$-c.a. degrees is the same as the collection of totally $\alpha$-c.a. degrees.

Thus, the only ordinals $\alpha$ for which the class of uniformly totally $\alpha$-c.a. degrees does not necessarily coincide with the class of totally $\beta$-c.a. degrees for some ordinal $\beta$ are the powers of $\omega$. Theorem 3.5 shows that for ordinals $\alpha \leqslant \varepsilon_0$ which are powers of $\omega$, the uniformly totally $\alpha$-c.a. degrees indeed form a distinct level of the hierarchy.

Theorem 3.5. *Let $\alpha \leqslant \varepsilon_0$ be a power of $\omega$.*

(1) *There is a uniformly totally $\alpha$-c.a. c.e. degree which is not totally $\gamma$-c.a. for any $\gamma < \alpha$.*
(2) *There is a totally $\alpha$-c.a. c.e. degree which is not uniformly totally $\alpha$-c.a.*

The first $\omega\cdot 2$ many levels of the hierarchy of totally and uniformly totally $\alpha$-c.a. degrees are depicted in Figure 2.

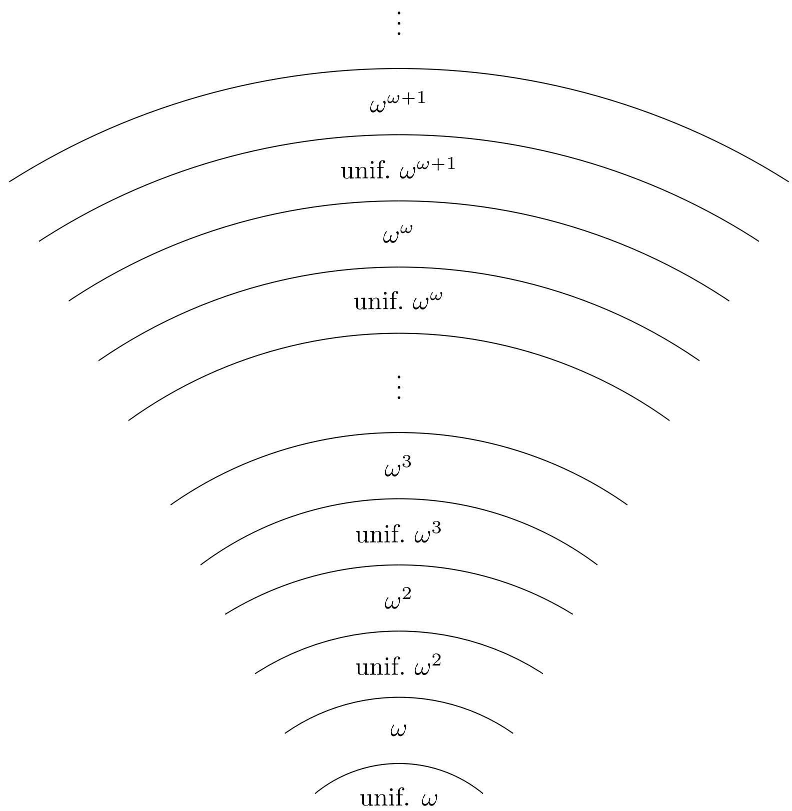


FIGURE 2. The first refinement of the hierarchy. "$\omega^\alpha$" denotes the collection of totally $\omega^\alpha$-c.a. degrees. "unif. $\omega^\alpha$" denotes the class of uniformly totally $\omega^\alpha$-c.a. degrees.

**3.1. Proof of Theorem 3.5(1).** We show that the first part of Theorem 3.5 is actually already proved using the construction used for proving Theorem 2.1. Given $\alpha \leqslant \varepsilon_0$ which is a power of $\omega$, that construction produces a c.e. set $D$ whose Turing degree is totally $\alpha$-c.a., but such that there is some $f \leqslant_{\mathrm{T}} D$ that is not $\gamma$-c.a. for any $\gamma < \alpha$. We show that $\deg_{\mathrm{T}}(D)$ is actually uniformly totally $\alpha$-c.a. The reason for this is the long delay between expansionary stages that was already incorporated into the construction.

For concreteness, let $P^{e_0,\gamma_0}, P^{e_1,\gamma_1}, \dots$ effectively enumerate all the positive requirements $P^{e,\gamma}$, and suppose that for all $k < \omega$, all nodes of length $2k$ work for the requirement $P^{e_k,\gamma_k}$. In particular, all nodes of even length work for some positive requirement.

LEMMA 3.6. *For all stages $s$, for all $\sigma \in \delta_s$, $|\sigma| \leqslant 2s$.*

PROOF. By induction on $s$. If this holds for all stages $t < s$, and if at stage $s$, some node $\sigma$ of length $2s$ is accessible, then since it works for a positive requirement, and was not accessible at any stage before $s$, at stage $s$, the nodes $\sigma$ acts by appointing a follower, and ends the stage. ☐

For all $n < \omega$, let

$$h(n) = \left(\max_{k \leqslant n} \gamma_k\right) \cdot 2^{2n}.$$

Since every ordinal below $\alpha$ appears as some $\gamma_k$, the function $h$ is an $\alpha$-order function. The combinatorial point is that if $\sigma_1, \dots, \sigma_l$ is a sequence of *distinct* nodes on the tree, each of length at most $2n$, with $\sigma_i$ working for $P^{e_{k_i}, \gamma_{k_i}}$ (so $k_i \leqslant n$), then as the tree of strategies is (at most) binary branching, we have $l \leqslant 2^{2n}$, and so

$$\sum_{i \leqslant l} \gamma_{k_i} \leqslant h(n).$$

We show that every $f \leqslant_{\mathrm{T}} D$ is $(h+1)$-c.a. To this end, fix some $e < \omega$ such that $\Phi_e(D)$ is total, and let $\tau$ be a node on the true path which works for requirement $Q_e$. Recall the construction, during the proof of Theorem 3.5, of an $\alpha$-computable approximation $\langle g_j, m_j \rangle$ for $\Phi_e(D)$. We let $s^*$ be the last stage at which $\tau$ was initialised, and

$$S = \{s > s^* \,:\, \tau\hat{\ }\infty \in \delta_s\} = \{s_0, s_1, \dots\}.$$

For all $x < \omega$, $i(x)$ was the least index $i$ such that $x < \operatorname{dom} \Phi_e(D)[s_i]$. For $j \geqslant i(x)$ we observed the set $a_j(x)$ of nodes $\sigma \succcurlyeq \tau\hat{\ }\infty$ that have followers at the beginning of stage $s_{i(x)}$, and are not initialised between stages $s_{i(x)}$ and $s_j$; we focus on $a(x) = a_{i(x)}(x)$. The ordinal $m_0(x) = m_{i(x)}(x)$ was defined to be the sum of ordinals of the form $o_t^{i,\gamma}(p)$, where $t < s_{i(x)}$ is some stage, and $p$ is a follower at stage $s_{i(x)}$ for $\sigma \in a(x)$, working for $P^{i,\gamma}$. Certainly $o_t^{i,\gamma}(p) \leqslant \gamma$. And so, if $2n$ is a bound on the lengths of nodes in $a(x)$, then $m_0(x) \leqslant h(n)$. The proof will be complete when we show that for almost all $x$, $2x$ is a bound on the lengths of nodes in $a(x)$, and so $m_0(x) \leqslant h(x)$; so a modification of the approximation $\langle g_j, m_j \rangle$ on finitely many inputs yields an $h$-computable approximation for $\Phi_e(D)$.

LEMMA 3.7. *For all $x \geqslant \operatorname{dom} \Phi_e(D)[s_1]$, for all $\sigma \in a(x)$, $|\sigma| \leqslant 2x$.*

PROOF. Let $x \geqslant \operatorname{dom} \Phi_e(D)[s_1]$. So $i(x) \geqslant 2$; for brevity, we let $u_0 = s_{i(x)-2}$ and $u_1 = s_{i(x)-1}$. By the instructions for $\tau$, $u_0 < \operatorname{dom} \Phi_e(D)[u_1]$; by minimality of $i(x)$, $\operatorname{dom} \Phi_e(D)[u_1] \leqslant x$; so $x > u_0$. By Lemma 3.6, all nodes accessible at any stage $t \leqslant u_0$ have length at most $2u_0$.

Let $\sigma \in \delta_{u_1}$ be a node working for some positive requirement $P^{i,\gamma}$ which has not been accessible at any stage $s \leqslant u_0$ (if there is such a node). Since $\tau$ and all of its predecessors are accessible at stage $u_0$, we have $\sigma \succcurlyeq \tau\hat{\ }\infty$. But since $u_1$ is the immediate successor of $u_0$ in $S$, $\sigma$ was not accessible at any stage $s \in (u_0, u_1)$; so $u_1$ is the least stage at which $\sigma$ is accessible, and so $\sigma$ ends the stage $u_1$. It follows that for such $\sigma$ we must have $|\sigma| \leqslant 2u_0 + 2$.

In total, if $\sigma \succcurlyeq \tau\hat{\ }\infty$ is accessible at any stage $s \leqslant u_1$, then $|\sigma| \leqslant 2(u_0+1) \leqslant 2x$.

Let $\sigma \in a(x)$. The node $\sigma$ extends $\tau\hat{\ }\infty$, and was accessible at some stage $t \in S$, smaller than $s_{i(x)}$; so $t \leqslant u_1$. Hence $|\sigma| \leqslant 2x$ as required. ☐

**3.2. Proof of Theorem 3.5(2).** A minor modification of the construction for Theorem 2.1 gives the proof of the second part of Theorem 3.5. Again, we are given an ordinal $\alpha$ which is a power of $\omega$, so is closed under addition; and we enumerate a c.e. set $D$ whose Turing degree will be totally $\alpha$-c.a. but not uniformly so. By Lemma 3.4, it is sufficient to fix an $\alpha$-order function $h$ and enumerate a functional $\Lambda$ such that $\Lambda(D)$ is total and is not $h$-c.a. What makes this construction work is that we can enumerate tidy $(h+1)$-computable approximations. The definition is the expected modification of Definition II.1.5. A simpler version of the proof of Proposition II.1.7 yields:

LEMMA 3.8. *Let $\alpha \leqslant \varepsilon_0$ and let $h$ be an $\alpha$-order function. Then there is an effective enumeration $\langle f^e_s, o^e_s\rangle$ of tidy $(h+1)$-computable approximations such that letting $f^e = \lim f^e_s$, the sequence $\langle f^e\rangle_{e<\omega}$ contains all $h$-c.a. functions.*

Fixing $h$, we get an enumeration of $(h+1)$-computable approximations $\langle\langle f^e_s, o^e_s\rangle_{s<\omega}\rangle_{e<\omega}$ as in Lemma 3.8, and repeat the construction for Theorem 2.1 where the positive requirements are now:

$P^e$**:** There is some $p$ such that $\Lambda(D,p) \neq f^e(p)$.

The rest of the construction is identical, as are the verifications, and so we omit them. The critical reader would ask, though: as was shown in the previous subsection, the construction for Theorem 2.1 actually produces a uniformly totally $\alpha$-c.a. degree. Why cannot we replicate the argument now to get a contradiction?

We recall the argument proving the first part of Theorem 3.5. Let $e < \omega$ such that $\Phi_e(D)$ is total. A uniform bound for $m_0(x)$, where $\langle g_j, m_j\rangle_{j<\omega}$ is the $\alpha$-computable approximation for $\Phi_e(D)$, was given by seeing that for almost all $x$, the nodes in $a(x)$ all had length at most $2x$, a fact which is preserved in the current, modified construction. In the previous construction, this was sufficient to give the bound, since for any follower $p$ for some node $\sigma \in a(x)$, working for some $P^{i,\gamma}$, we had $o^{i,\gamma}_0(x) \leqslant \gamma$. In the current construction, of course, we just have $o^i_0(p) \leqslant h(p)+1$, so the size of $p$ plays a role.

Cannot we use the argument showing that $\sigma \in a(x)$ has length at most $2x$ to also bound the size of followers for such $\sigma$? After all, these followers are chosen at some $\tau$-expansionary stage $t$ smaller than $s_{i(x)}$, and, roughly speaking, a follower chosen at stage $t$ has size "close to $t$". As in the proof of Lemma 3.7, let $u_0 < u_1$ be the immediate predecessors of $s_{i(x)}$ in $S$. Then $u_0$ is bounded by $x$, but $u_1$ may be much larger than $x$; and one element $\sigma$ of $a(x)$ may pick its follower at stage $u_1$. So even though the *length* of that $\sigma$ is bounded by $2x$, the *size of its follower* cannot be computably bounded in $x$, and it is this single late-choosing element of $a(x)$ which prevents us from giving an approximation $\Phi_e(D,x)$ with some ordinal bound which depends on $x$ but not on $\sigma$ and $p$ (and so not on $\tau$).

## 4. Another refinement of the hierarchy: totally $<\omega^\alpha$-c.a. degrees

The hierarchy of totally $\alpha$-c.a. degree is not, a priori, the finest one could devise. For a limit ordinal $\alpha$, one could conceive of a totally $\alpha$-c.a. degree $\mathbf{d}$ such that every $f \in \mathbf{d}$ is $\gamma$-c.a. for some $\gamma < \alpha$, but such that $\mathbf{d}$ is not totally $\gamma$-c.a. for any $\gamma < \alpha$.

DEFINITION 4.1. Let $\alpha \leqslant \varepsilon_0$. A Turing degree $\mathbf{d}$ is *totally $<\alpha$-c.a.* if every $f \in \mathbf{d}$ is $\gamma$-c.a. for some $\gamma < \alpha$.

As is the case with totally $\alpha$-c.a. degrees and with uniformly totally $\alpha$-c.a. degrees, a Turing degree $\mathbf{d}$ is totally $< \alpha$-c.a. if and only if every $f \leqslant_{\mathrm{T}} \mathbf{d}$ is $\gamma$-c.a. for some $\gamma < \alpha$.

As was indicated in the introduction, the class of totally $< \omega^\omega$-c.a. degrees is the main class investigated in this work.

As we did for uniformly totally $\alpha$-c.a. degrees, we now examine how the classes of totally $< \alpha$-c.a. degrees fit in the hierarchy of totally $\alpha$-c.a. degrees. Of course, if $\gamma < \alpha$, then every totally $\gamma$-c.a. degree is totally $< \alpha$-c.a., and every totally $< \alpha$-c.a. degree is totally $\alpha$-c.a. In fact, slightly more holds: for any ordinal $\alpha$, every totally $< \alpha$-c.a. degree is uniformly totally $\alpha$-c.a., because for any $\alpha$-order function $h$ and all $\gamma < \alpha$, $h(x) \geqslant \gamma$ for almost all $x$, so any $\gamma$-computable approximation can easily be converted into an $h$-computable approximation.

Lemma 2.2 shows that if $\beta \in (\omega^\alpha, \omega^{\alpha+1}]$, then every totally $< \beta$-c.a. degree is totally $\omega^\alpha$-c.a.; in particular, note that this holds even if $\beta = \omega^{\alpha+1}$. Hence, if $\beta$ is not a *limit of powers of* $\omega$, then there is some $\alpha < \beta$, a power of $\omega$, such that the class of totally $< \beta$-c.a. degrees coincides with the class of totally $\alpha$-c.a. degrees.

Also note that the construction proving Theorem 2.1 and Theorem 3.5(1) produces a degree that is uniformly totally $\alpha$-c.a. but not totally $< \alpha$-c.a.; to show that the degree constructed was not totally $\gamma$-c.a. for any $\gamma < \alpha$, we constructed a single function $\Lambda(D)$ which was not $\gamma$-c.a. for any $\gamma < \alpha$.

The following theorem then completely determines the new levels of our hierarchy, the first $\omega \cdot 2$ levels of which are depicted in Figure 3.

THEOREM 4.2. *If $\alpha \leqslant \varepsilon_0$ is a limit of powers of $\omega$, then there is a c.e. degree which is totally $< \alpha$-c.a. but not totally $\gamma$-c.a. for any $\gamma < \alpha$.*

The rest of this section is devoted to the proof of Theorem 4.2. We are given an ordinal $\alpha \leqslant \varepsilon_0$, a limit of powers of $\omega$, and give a computable enumeration $\langle D_s \rangle$ of a c.e. set $D$ such that $\deg_{\mathrm{T}}(D)$ is totally $< \alpha$-c.a. but not totally $\gamma$-c.a. for any $\gamma < \alpha$.

For every $\gamma < \alpha$ and $e < \omega$ we must meet the requirements

$P^\gamma$**:** There is a function $f \leqslant_{\mathrm{T}} D$ which is not $\gamma$-c.a.

and

$Q_e$**:** If $\Phi_e(D)$ is total, then $\Phi_e(D)$ is $\gamma$-c.a. for some $\gamma < \alpha$.

*Discussion.* The first thing to notice is that we cannot, uniformly in $\gamma$, compute from $D$ a function $f$ which is not $\gamma$-c.a.; for we could string these functions together to get a single function which is not $\gamma$-c.a. for any $\gamma < \alpha$, and so fail to make $\deg_{\mathrm{T}}(D)$ totally $< \alpha$-c.a.

It is also fairly easy to see how the construction necessitates this non-uniformity. For suppose we tried to copy the construction proving Theorem 2.1. A node $\tau$, working for $Q_e$, is now trying to make $\Phi_e(D)$ a $\gamma$-c.a. function for some $\gamma < \alpha$. But extending $\tau\hat{\ }\infty$ are nodes $\sigma$, working for $P^\beta$ for ordinals $\beta$ which are unbounded in $\alpha$; their action would cause changes to $\tau$'s approximation of $\Phi_e(D)$, and so force $\tau$ to have its $\gamma$ larger than all of these $\beta$'s, i.e., to be at least $\alpha$.

The solution concerns that basic staple of both comedy and computability theory, namely timing. Remember that in a situation as above, a node $\sigma$ extending $\tau\hat{\ }\infty$ can injure a computation $\Phi_e(D, x)[s]$ only if the follower $p$ for $\sigma$ at stage $s$ was appointed before the $\tau$-expansionary stage $t = s_{i(x)}$ at which we first observed and

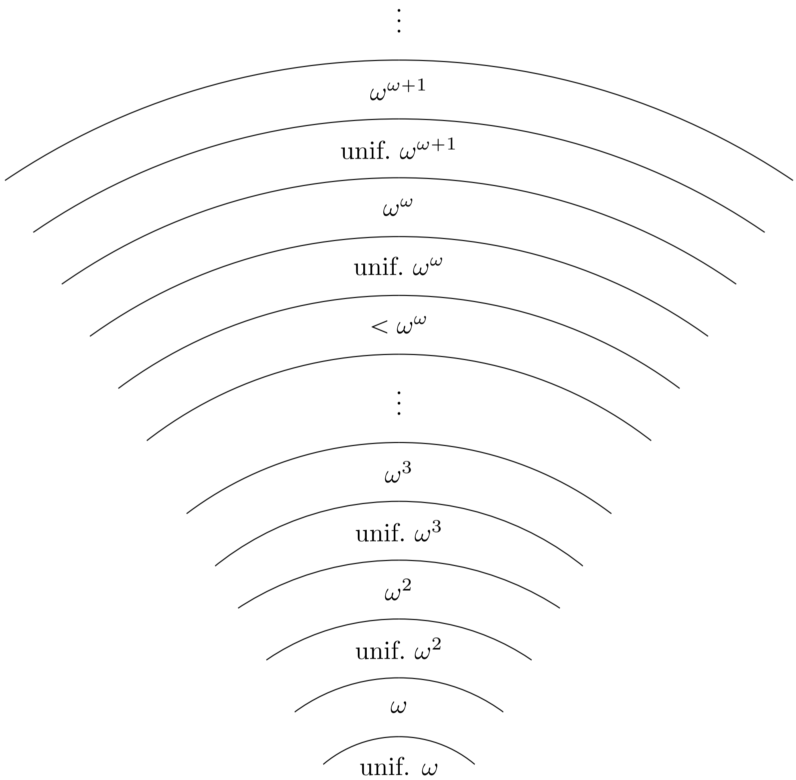


FIGURE 3. The second refinement of the hierarchy. "$\omega^\alpha$" denotes the collection of totally $\omega^\alpha$-c.a. degrees. "$< \omega^\alpha$" denotes the collection of totally $< \omega^\alpha$-c.a. degrees. "unif. $\omega^\alpha$" denotes the class of uniformly totally $\omega^\alpha$-c.a. degrees.

certified a computation $\Phi_e(D, x)[t]$. On the other hand, regardless of when $p$ was appointed, upon enumerating $\lambda_s(p)$ into $D_{s+1}$, we need to *immediately* appoint a new use $\lambda_{s+1}(p)$, without waiting for a new $\Phi_e(D, x)[u]$ computation to recover; this, because we need to make $\Lambda(D)$ total. Even though $\sigma$ guesses that $\Phi_e(D)$ is total, it is participating in the construction of the global functional $\Lambda$, and is responsible for making $p \in \operatorname{dom} \Lambda(D)$, even if its guess is incorrect. Inevitably, the new marker $\lambda_{s+1}(p)$ will be smaller than the use $\varphi_{e,r}(x)$ at the next $\tau$-expansionary stage, and so further action with $p$ will injure $\Phi_e(D, x)$ again.

In the previous construction this was fine, because $\sigma$ provided $\tau$ with a bound $o_0^{i,\gamma}(p)$ on the "number of times" is will act for $p$, and the sum of these bounds was smaller than $\alpha$. As mentioned above, this is insufficient when we want to show that the function $\Phi_e(D)$ is $\gamma$-c.a. for some $\gamma < \alpha$. Once we determined $\gamma$, what we need to do is break the cycle of repeated injury by the same follower $p$, when the bound for the follower is greater than $\gamma$. This is possible if we delay defining $\Lambda(D, p)$ until we see the computation $\Phi_e(D, x)$ recover. To do this, we distribute in a tree of strategies nodes $\eta$, working for $P^\beta$, which are responsible for a local version $\Lambda_\eta(D)$

of $\Lambda(D)$. Only nodes extending $\eta$ contribute to the definition of $\Lambda_\eta(D)$, and the function $\Lambda_\eta(D)$ is required to be total only if $\eta$ lies on the true path. If such $\eta$ extends $\tau\hat{\ }\infty$, then indeed definitions of $\Lambda_\eta(D,p)$ can wait until $\Phi_e(D,x)$ recovers at the next $\tau$-expansionary stage. We see how this gives the non-uniformity in defining the function witnessing $P^\beta$: we need the true path to find it.

How do we find $\gamma$? The approach of waiting to define $\Lambda_\eta$ cannot be employed if $\tau$ extends $\eta$. If $\sigma$ is a "child" node of such $\eta$ with $\tau\hat{\ }\infty \preccurlyeq \sigma$, then we are back at the situation of the original construction: repeated action for a follower $p$ for $\sigma$ will keep injuring a computation $\Phi_e(D,x)$. Again, $\sigma$ provides a bound for its action, and that bound is itself bounded by $\beta$, where $\eta$ works for $P^\beta$. And $\beta < \alpha$. Since there are only finitely many "mother" nodes $\eta \prec \tau$, the bound $\gamma$ will be any ordinal, closed under addition, which bounds the ordinals $\beta$ for these nodes $\eta$. That such an ordinal $\gamma < \alpha$ can be found follows from the fact that $\alpha$ is a limit of ordinals closed under addition.

*The tree of strategies.* Let $\gamma < \alpha$. In order to meet the requirement $P^\gamma$, for each $e < \omega$, we need to meet the subrequirements $P^{e,\gamma}$ which diagonalise against $f^{e,\gamma}$. We arrange all of the requirements and subrequirements – $Q_e$, $P^\gamma$ and $P^{e,\gamma}$ – effectively, in a list of order-type $\omega$, but ensuring that for each $\gamma$ and $e$, $P^\gamma$ appears before $P^{e,\gamma}$. We let all strategies on the tree of length $k$ work for the $k^{\text{th}}$ requirement on the list.

Nodes working for requirements $P^\gamma$ and $P^{e,\gamma}$ have only one outcome. Nodes working for $Q_e$ have two outcomes, $\infty$ and $\mathtt{fin}$, the former stronger than the latter.

Nodes $\eta$ working for $P^\gamma$ enumerate a functional $\Lambda_\eta$. For any node $\sigma$ working for $P^{\gamma,e}$ there is a unique node $\eta \prec \sigma$ working for $P^\gamma$. We denote this node, the "mother" of $\sigma$, by $\eta(\sigma)$.

*Construction.* At stage $s$, we let $\delta_s$, the collection of nodes accessible at stage $s$, be an initial segment of the tree of strategies.

Suppose that a node $\tau$ that works for requirement $Q_e$ is accessible at stage $s$. If $s$ is the least stage at which $\tau$ is accessible, then we let $\tau\hat{\ }\infty \in \delta_s$. Otherwise, we let $t$ be the last stage before $s$ at which $\tau\hat{\ }\infty$ was accessible. If $t < \operatorname{dom}\Phi_{e,s}(D_s)$, then we let $\tau\hat{\ }\infty \in \delta_s$. Otherwise, we let $\tau\hat{\ }\mathtt{fin} \in \delta_s$.

Suppose that a node $\eta$ that works for requirement $P^\gamma$ is accessible at stage $s$. If there is some $p$ which is a follower for some child $\sigma \succ \eta$ of $\eta$ (an extension of $\eta$ working for some subrequirement $P^{e,\gamma}$) such that $p \notin \operatorname{dom}\Lambda_{\eta,s}(D_s)$, then we define $\Lambda_{\eta,s+1}(D_s,p) = s+1$ with large use, and end the stage (in this case, we do not initialise all nodes weaker than $\eta$; but as usual, we do initialise all nodes which lie to the right of $\eta$).

Otherwise, for all $p < s$ which is not in $\operatorname{dom}\Lambda_{\eta,s}(D_s)$, we define $\Lambda_{\eta,s+1}(D_s,p) = 0$ with use $-1$; the unique immediate successor of $\eta$ on the tree of strategies is accessible next.

Suppose that a node $\sigma$ that works for a subrequirement $P^{e,\gamma}$ is accessible at stage $s$.

(1) If $\sigma$ has no follower, then $\sigma$ appoints a new, large follower for itself.
(2) If $\sigma$ has a follower $p$, and $\Lambda_{\eta(\sigma),s}(D_s,p)\downarrow = f^{e,\gamma}_s(p)$, then we enumerate $\lambda_{\eta(\sigma),s}(p)$ into $D_{s+1}$.

Note that in either case, we do not define a new computation $\Lambda_{\eta(\sigma),s+1}(D_{s+1}, p)$. In either case, we end the stage and initialise all nodes weaker than $\sigma$. If $\sigma$ does not act, then the unique immediate successor of $\sigma$ on the tree of strategies is accessible at stage $s$.

*Verification.* Let $\eta$ be a node that works for $P^\gamma$. At stage $s$, we only define a new $\Lambda_{\eta,s+1}(D_s, p)$ computation if $p \notin \operatorname{dom} \Lambda_s(D_s)$. Lemma I.1.1 ensures that each $\Lambda_{\eta,s}$ is consistent for $D_s$, and so that each $\Lambda_\eta$ is consistent for $D$.

LEMMA 4.3. *Let $s$ be a stage, and let $\sigma$ be a node working for $P^{e,\gamma}$ which has a follower $p$ at the beginning of stage $s$.*

(1) *If $p \notin \operatorname{dom} \Lambda_{\eta(\sigma),s}(D_s)$, then at the last stage $t < s$ at which $\eta(\sigma)$ was accessible, so was $\sigma$, and $\sigma$ acted at stage $t$.*
(2) *If $\sigma'$ is a node weaker than $\sigma$, working for $P^{e',\gamma'}$, and has a follower $p'$ at the beginning of stage $s$, then $p < p'$. If in addition $p \in \operatorname{dom} \Lambda_{\eta(\sigma),s}(D_s)$ then $\lambda_{\eta(\sigma),s}(p) < p'$. Consequently, if also $p' \in \operatorname{dom} \Lambda_{\eta(\sigma'),s}(D_s)$ then $\lambda_{\eta(\sigma),s}(p) < \lambda_{\eta(\sigma'),s}(p')$.*

PROOF. Both parts of the lemma are proved simultaneously, by induction on $s$. Assume both parts hold at all stages before stage $s$. Let $\eta = \eta(\sigma)$.

For (1), let $t < s$ be the last stage before $s$ at which $\eta(\sigma)$ was accessible, and suppose that $\sigma$ does not act at stage $t$. Then $p$ is already a follower for $\sigma$ at the beginning of stage $t$, and so $\sigma$ was not initialised at any stage $r \in [t, s)$. If $p \notin \operatorname{dom} \Lambda_{\eta,t}(D_t)$, then at stage $t$, $\eta$ defines a new computation $\Lambda_{\eta,t+1}(D_t, p)$, and ends the stage. This means that $D_{t+1} = D_t$, and so $p \in \operatorname{dom} \Lambda_{\eta,t+1}(D_{t+1})$ with $\lambda_{\eta,t}(p) = \lambda_{\eta,t+1}(p)$. By (2) at all stages $r \in [t+1, s)$, this computation cannot be injured at stage $r$ without initialising $\sigma$, so $p \in \operatorname{dom} \Lambda_{\eta,s}(D_s)$. If, on the other hand, $p \in \operatorname{dom} \Lambda_{\eta,t}(D_t)$, then by (2) at all stages $r \in [t, s)$, this computation cannot be injured without initialising $\sigma$.

For (2), let $\sigma'$ and $p'$ be as described. That $p < p'$ follows as usual from the fact that the stage at which $p'$ was chosen is later than the stage at which $p$ was chosen.

For the second part, let $t < s$ be the stage at which the computation $\Lambda_{\eta,s}(D_s, p)$ was defined. To show that $\lambda_{\eta,s}(p) < p'$, we show that the follower $p'$ was chosen after stage $t$. We know that $\Lambda_\eta(D, p)[t]\uparrow$. Let $u$ be the last stage prior to stage $t$ at which $\eta$ was accessible. By (1) at stage $t$, $\sigma$ acted at stage $u$, and so $\sigma'$ was initialised at stage $u$. Since $\eta \prec \sigma$, $\eta$ is stronger than $\sigma'$. If $\sigma'$ lies to the right of $\eta$, then it is initialised at stage $t$, and so $p'$ is chosen after stage $t$. Otherwise, $\sigma' \succ \eta$, and so $\sigma'$ is not accessible at any stage $r \in (u, t)$ and also not accessible at stage $t$ (as $\eta$ ends the stage). Thus, again, $p'$ was chosen after stage $t$. $\square$

As a corollary we can conclude that for $\sigma$ and $p$ as above, if $p \in \operatorname{dom} \Lambda_{\eta(\sigma),s}(D_s)$, then $\lambda_{\eta(\sigma),s}(p) \notin D_s$. An analogue of Lemma 2.3(d) also holds, with a similar argument.

LEMMA 4.4. *Let $t > s$ be stages and let $\sigma$ be a node which works for some positive subrequirement. Suppose that $p$ is a follower for $\sigma$ at the beginning of stage $s$. Suppose that $\sigma$ is not initialised at any stage $r \in [s, t)$.*

(1) *$D_s \restriction_p = D_t \restriction_p$.*
(2) *If in addition $p \in \operatorname{dom} \Lambda_{\eta(\sigma),s}(D_s)$, then $D_s \restriction_{\lambda_{\eta(\sigma),s}(p)} = D_t \restriction_{\lambda_{\eta(\sigma),s}(p)}$.*

(3) *If, further, $\sigma$ does not act at any stage $r \in [s,t)$, then $p \in \operatorname{dom}\Lambda_{\eta(\sigma),t}(D_t)$ and $\lambda_{\eta(\sigma),s}(p) = \lambda_{\eta(\sigma),t}(p)$.*

LEMMA 4.5. *Suppose that $\sigma$ works for $P^{e,\gamma}$, and that $p$ is a follower which is appointed for $\sigma$ at some stage and is never cancelled. Suppose that $\sigma$ does not act infinitely often. Suppose also that $\eta(\sigma)$ is accessible infinitely often. Then $p \in \operatorname{dom}\Lambda_{\eta(\sigma)}(D)$.*

PROOF. Let $s$ be the last stage at which $\sigma$ acts; since $p$ is not cancelled after stage $s$, $\sigma$ is not initialised after stage $s$. Let $t$ be the least stage after stage $s$ at which $\eta = \eta(\sigma)$ is accessible. If $p \in \operatorname{dom}\Lambda_{\eta,s}(D_s)$ then the action of $\sigma$ at stage $s$ removes $p$ from $\operatorname{dom}\Lambda_{\eta,s+1}(D_{s+1})$; in any case, $p \notin \operatorname{dom}\Lambda_{\eta,t}(D_t)$. At stage $t$, $\eta$ defines a new computation $\Lambda_{\eta,t+1}(D_{t+1},p)$. By Lemma 4.4, this computation is $D$-correct. $\square$

Lemma 2.4 holds for the current construction as well: if $\sigma$, working for $P^{e,\gamma}$, acts at stages $s<t$ and has the same follower $p$ at the end of stage $s$ and the end of stage $t$, then $o^{e,\gamma}_t(p) < o^{e,\gamma}_s(p)$. The proof is similar; the computation $\Lambda_{\eta(\sigma),t}(D_t,p)$ must have been defined by $\eta(\sigma)$ at a stage $u>s$, and so its value is $u+1$ which is bigger than $s$, so $f^{e,\gamma}_s(p) < s < u+1 = f^{e,\gamma}_t(p)$. Now an argument, identical to the argument proving Lemma 2.5, shows that if $\sigma$, working for $P^{e,\gamma}$ is on the true path, and the construction is fair to $\sigma$, then $\sigma$ eventually appoints a follower $p$ which is never cancelled, eventually stops acting, and $\Lambda_{\eta(\sigma)}(D,p) \neq f^{e,\gamma}(p)$. It follows that the true path is infinite, that the construction is fair to every node on the true path, and that if $\eta$ on the true path works for $P^\gamma$, then $\Lambda_\eta(D)$ is total, and is not $\gamma$-c.a. Since the true path has a node in every level, each $P^\gamma$ is met, so $\deg_{\mathrm{T}}(D)$ is not totally $\gamma$-c.a. for any $\gamma<\alpha$.

To conclude the proof of Theorem 4.2, we need to show that for all $e<\omega$ such that $\Phi_e(D)$ is total, $\Phi_e(D)$ is $\gamma$-c.a. for some $\gamma<\alpha$. Fix such $e$, and let $\tau$ be the node on the true path that works for requirement $Q_e$. At first, we proceed as in the proof of Theorem 2.1. Let $s^*$ be the last stage at which $\tau$ is initialised, and let

$$S = \{s > s^* : \tau\hat{\ }\infty \in \delta_s\} = \{s_0, s_1, \dots\}$$

as again, $S$ is infinite. For $x<\omega$ we define $i(x)$ as before, to be the least $i$ such that $x < \operatorname{dom}\Phi_e(D)[s_i]$. And again, for $j \geqslant i(x)$ we let $a_j(x)$ be the set of nodes $\sigma \geqslant \tau\hat{\ }\infty$ which at the beginning of stage $s_j$ have a follower $p = p(\sigma,x)$ which was appointed before stage $s_{i(x)}$. Lemma 2.9 holds for the current construction, with the same proof, except that now we use $p>u$ rather than $\lambda_{\eta(\sigma),t}(p) > u$; so we use part (1) of Lemma 4.4 instead of part (2).

We now find an ordinal bound below $\alpha$ for the complexity of $\Phi_e(D)$. Fix $x<\omega$. For $\sigma \in a(x)$, since $\sigma \geqslant \tau\hat{\ }\infty$, $\eta(\sigma)$ is comparable with $\tau\hat{\ }\infty$.

LEMMA 4.6. *Let $\sigma \in a(x)$, and suppose that $\eta(\sigma) \geqslant \tau\hat{\ }\infty$. Then there is at most one $j \geqslant i(x)$ such that $\sigma$ acts at stage $s_j$ and injures the computation $\Phi_e(D,x)[s_j]$.*

PROOF. Let $s_j$ be a stage at which $\sigma$ acts, where $j \geqslant i(x)$. We show by induction that for all $i>j$ in $S$, if $p$ is a follower for $\sigma$ at the beginning of stage $s_i$, and $p \in \operatorname{dom}\Lambda_{\eta(\sigma)}(D)[s_i]$, then $\lambda_{\eta(\sigma),s_i}(p) > \varphi_{e,s_i}(x)$, so $\sigma$ cannot injure $\Phi_e(D,x)[s_i]$ at stage $s_i$. Let $\eta = \eta(\sigma)$.

The base step is vacuous, and this is the main point of the proof. At stage $s_j$, $\sigma$'s action extracts its follower $p$ from $\operatorname{dom}\Lambda_{\eta(\sigma)}(D)$. The assumption $\eta \geqslant \tau\hat{\ }\infty$ means that $\eta$ is not accessible at any stage $r \in (s_j, s_{j+1})$, and so $p \notin \operatorname{dom} \Lambda_\eta(D)[s_{j+1}]$. Note that $p$ is still the follower for $\sigma$ at the beginning of stage $s_{j+1}$.

Let $i > j+1$ and suppose the inductive claim holds for all $i' \in (j, i)$. Let $p$ be a follower for $\sigma$ at the beginning of stage $s_i$, and suppose that $p \in \operatorname{dom} \Lambda_\eta(D)[s_i]$. The proof follows the idea for Lemma 2.9. Let $t$ be the stage at which the computation $\Lambda_\eta(D, p)[s_i]$ was defined. Since $\eta \geqslant \tau\hat{\ }\infty$, $t \in S$; and $t \geqslant s_{j+1}$. Thus $\lambda_{\eta,t}(p)$ is chosen to be larger than $u = \varphi_{e,t}(x)$. Lemma 4.4(2) now shows that $D_{s_i}\restriction_u = D_t\restriction_u$ and so $\varphi_{e,s_i}(x) = u < \lambda_{\eta,s_i}(p)$ as required. □

Since $\alpha$ is a limit of ordinals which are closed under addition, and $\tau$ has only finitely many predecessors on the tree of strategies, find some ordinal $\delta < \alpha$, closed under addition, such that for all $\eta \prec \tau$ which work for some $P^\gamma$ we have $\gamma < \delta$. We give a $\delta$-computable approximation for $\Phi_e(D)$, along the lines of the proof of Theorem 2.1.

Again fixing $x$, for $j \geqslant i(x)$ and $\sigma \in a_j(x)$ we again let $t_j(\sigma)$ be the greatest stage $t < s_j$ at which $\sigma$ acts. The main part is defining the ordinal $\beta_j(\sigma)$:

- If $\eta(\sigma) \prec \tau$, then we let $\beta_j(\sigma) = o^{i,\gamma}_{t_j(\sigma)}(p(\sigma, x))$, where $\sigma$ works for $P^{i,\gamma}$.
- If $\eta(\sigma) \geqslant \tau\hat{\ }\infty$, then we let $\beta_j(\sigma) = 0$ if there is some $i \in [i(x), j)$ for which $\sigma$ acts at stage $s_i$ and destroys the computation $\Phi_e(D, x)[s_i]$. If there is no such $i$, then we let $\beta_j(\sigma) = 1$.

We then mimic the rest of the proof of Theorem 2.1, ordering $a_j(x)$ by descending priority, and defining $m_j(x) = \sum_{\sigma \in a_j(x)} \beta_j(x)$. The proof of Theorem 4.2 is complete once we show that Lemma 2.10 holds for the current construction (with $\delta$ replacing $\alpha$). The proof of this lemma is identical to the previous proof, except for one case: showing that $\beta_{j+1}(\sigma) < \beta_j(\sigma)$ if $g_{j+1}(x) \neq g_j(x)$, where $\sigma$ is the weakest node in $a_{j+1}(x)$, in the case that $\eta(\sigma) \geqslant \tau\hat{\ }\infty$. But in this case we appeal to Lemma 4.6.

## 5. Domination properties

In [24], Donwey, Jockusch and Stob extend the notion of array computability from the c.e. degrees to all the Turing degrees. This they do by using domination properties of degrees. Such properties have been used early on, to characterise classes such as the hyperimmune-free degrees, the high degrees and the non-low$_2$ degrees. More recently [39], a combination of domination and measure characterisations have yielded a characterisation of LR-hardness.

Recent work has indicated that the generalisations of array computability defined in this chapter can also be extended to the non-c.e. degrees by considering domination. We give the results for completeness.

Recall that if $\mathcal{C}$ is a class of functions from $\omega$ to $\omega$, then a Turing degree $\mathbf{d}$ is *$\mathcal{C}$-dominated* if every function $g \in \mathbf{d}$ (equivalently $g \leqslant_T \mathbf{d}$) is dominated by some function $f \in \mathcal{C}$. For example, the hyperimmune-free degrees are the degrees which are $\mathcal{C}$-dominated, where $\mathcal{C}$ is the collection of all computable functions.

Definition 5.1. A Turing degree is *$\alpha$-c.a. dominated* if it is $\mathcal{C}$-dominated, where $\mathcal{C}$ is the class of all $\alpha$-c.a. functions. I.e., if every $\mathbf{d}$-computable function is dominated by some $\alpha$-c.a. function.

Theorem 5.2 (Diamondstone,Greenberg,Turetsky [12]). *Let $\alpha \leqslant \epsilon_0$. A c.e. degree is totally $\alpha$-c.a. if and only it is $\alpha$-c.a. dominated.*

Proof. Let $\mathbf{d}$ be a c.e. degree and let $D \in \mathbf{d}$ be a c.e. set.

In the non-trivial direction, suppose that $\mathbf{d}$ is $\alpha$-c.a. dominated. Let $g \in \mathbf{d}$, $g = \Gamma(D)$ for a functional $\Gamma$. Since $D$ is c.e. it can compute the modulus $m$ for the approximation $\langle g_s \rangle$ for $g$ given by $g_s = \Gamma_s(D_s)$; here the modulus $m$ is defined by $m(k) = s$ if $s$ is the least stage such that for all $t \geqslant s$, $g_t \upharpoonright_{k+1} = g \upharpoonright_{k+1}$.

Let $h$ be an $\omega$-c.a. function which majorises $m$, and let $\langle h_t, o_t \rangle_{t<\omega}$ be an $\alpha$-computable approximation for $h$. Letting $\tilde{g}_t(k) = g_{h_t(k)}(k)$ we get that $\langle \tilde{g}_t, o_t \rangle$ is an $\alpha$-computable approximation for $g$. Essentially, this argument repeats the proof of Proposition II.3.17, after noticing that $g$ is weak-truth-table reducible to any function dominating the modulus $m$. □

The same argument yields an analogous result for the special limit classes.

Theorem 5.3. *Let $\alpha \leqslant \epsilon_0$ be a limit of powers of $\omega$. A c.e. degree $\mathbf{d}$ is totally $< \alpha$-c.a. if and only if it is $< \alpha$-c.a. dominated, i.e., if for every $\mathbf{d}$-computable function $g$ there is some $\gamma < \alpha$ and some $\gamma$-c.a. function which dominates $g$.*

For the uniform version, for a class of functions $\mathcal{C}$, say that a Turing degree $\mathbf{d}$ is *uniformly $\mathcal{C}$-dominated* if there is some function $f \in \mathcal{C}$ which dominates every function in $\mathbf{d}$. In other words, if $\mathbf{d}$ is $\{f\}$-dominated for some $f \in \mathcal{C}$. For example, a $\Delta^0_2$ degree is low$_2$ if and only if it is uniformly $\mathcal{C}$-dominated, where $\mathcal{C}$ is the class of $\Delta^0_2$ functions. A Turing degree is *uniformly $\alpha$-c.a. dominated* if, as expected, it is uniformly $\mathcal{C}$-dominated, where $\mathcal{C}$ is the collection of all $\alpha$-c.a. functions.

The following is a generalisation of the aforementioned result by Downey, Jockusch and Stob: a c.e. degree is array computable if and only if it is uniformly $\omega$-c.a. dominated.

Theorem 5.4 (with McInerney). *Let $\alpha \leqslant \epsilon_0$ be a power of $\omega$. A c.e. degree $\mathbf{d}$ is uniformly totally $\alpha$-c.a. if and only if it is uniformly $\alpha$-c.a. dominated: some $\alpha$-c.a. function dominates all functions in $\mathbf{d}$.*

Proof. In one direction the argument is similar to the argument for Theorem 5.2, but noticing the uniformity. Assuming that $\mathbf{d}$ is uniformly $\alpha$-c.a. dominated, let $g$ be an $\alpha$-c.a. function which dominates every function in $\mathbf{d}$; fix an $\alpha$-c.a. order function $h$ such that $g$ is $h$-c.a. Let $f \in \mathbf{d}$, and let $\mu$ be the modulus function for $f$, by an approximation given by a c.e. set in $\mathbf{d}$, so $\mu \leqslant_{\mathrm{T}} \mathbf{d}$. Then $g$ dominates $\mu$, and the argument above shows that $f$ is $h$-c.a.

In the other direction, we show that slightly stronger fact, that for any $\alpha$-order-function $h$, there is an $\alpha$-c.a. function which dominates every $h$-c.a. function. Fix an $\alpha$-order-function $h$. Let $\langle f^e \rangle$ be an effective listing of all $h$-c.a. functions, each with a tidy $(h+1)$-computable approximation $\langle f^e_s, o^e_s \rangle$ (Lemma 3.8). Let $\tilde{f}(n) = \max_{e \leqslant n} f^e(n)$. Certainly $\tilde{f}$ dominates every $h$-c.a. function. For $n < \omega$ and $s < \omega$, let $\tilde{f}_s(n) = \max_{e \leqslant n} f^e_s(n)$ and let $\tilde{o}_s(n) = \bigoplus_{e \leqslant n} o^e_s(n)$; see the discussion of commutative addition of ordinals in Subsection II.3.3. Lemmas II.3.15 and II.3.16 show this is an $\alpha$-computable approximation for $\tilde{f}$. □

Downey, Jockusch and Stob also showed that one can pick a single $\omega$-c.a. function dominating all array computable degrees. This holds for the higher uniform levels as well.

Lemma 5.5. *Let $\alpha \leqslant \epsilon_0$. There is an $\alpha$-c.a. function $f$ such that for every $\alpha$-c.a. function $g$ there is a computable function $\varphi$ such that $g \leqslant \varphi \circ f$. I.e., for all $n$, $g(n) \leqslant f(\varphi(n))$.*

Proof. Let $\langle f_s^e, o_s^e\rangle = \langle f_s^{e,\alpha}, o_s^{e,\alpha}\rangle$ be a computable list of tidy $(\alpha+1)$-computable approximations (given by Proposition II.1.7) approximating all $\alpha$-c.a. functions. For $e, n$ and $s < \omega$, let $f(e,n,s) = f^e(n)$ if $o_s^e(n) < \alpha$; otherwise, let $f(e,n,s) = 0$. An $\alpha$-computable approximation $\langle f_t, o_t\rangle$ for $f$ is easily devised by first checking $o_s^e(n)$; if this is $\alpha$ then our approximation is constant, otherwise we follow the approximation $f_t^e(n)$ for $t \geqslant s$.

If $g$ is $\alpha$-c.a. then there is some $e$ such that $g = f^e$ and the approximation $\langle f_s^e, o_s^e\rangle$ is eventually $\alpha$-computable (for all $n$ there is some $s$ such that $o_s^e(n) < \alpha$). Fixing such $e$, for $n < \omega$ let $s(n)$ be the least $s$ such that $o_s^e(n) < \alpha$; then, let $\varphi(n) = (e, n, s(n))$. ☐

Proposition 5.6. *Let $\alpha \leqslant \epsilon_0$ be a power of $\omega$. There is an $\alpha$-c.a. function $f$ such that any Turing degree $\mathbf{d}$ is uniformly $\alpha$-c.a. dominated if and only if it is $\{f\}$-dominated.*

Proof. Let $f$ be the function given by Lemma 5.5. Since $\alpha$ is closed under addition, we may assume that $f$ is (strictly) increasing.

Let $\mathbf{d}$ be a uniformly $\alpha$-c.a. dominated degree; let $g$ be an $\alpha$-c.a. function dominating $\mathbf{d}$. Let $\varphi$ be a computable function such that for all $n$, $g(n) \leqslant f(\varphi(n))$. We may assume that $\varphi$ is (strictly) increasing.

Let $h \in \mathbf{d}$; define $\tilde{h}(n) = \max_{k<\varphi(n+1)} h(k)$. Then $\tilde{h} \leqslant_{\mathrm{T}} \mathbf{d}$ and so $g$ dominates $\tilde{h}$. Suppose that $g(n) \geqslant \tilde{h}(n)$ for all $n \geqslant n^*$. Let $k > \varphi(n^*)$; find $n \geqslant n^*$ such that $\varphi(n) \leqslant k < \varphi(n+1)$. Then $h(k) \leqslant \tilde{h}(n) \leqslant g(n) \leqslant f(\varphi(n)) \leqslant f(k)$. ☐

In fact, for $\alpha = \omega$, Downey, Jockusch and Stob showed that for the "uniformly uniformly dominating" function $f$ one can take $m_{\varnothing'}$, the modulus function for $\varnothing'$. A similar argument can be made for $\alpha = \omega^\beta > \omega$ by replacing $\varnothing'$ with $I_\beta^\varnothing$ of Section II.3, the iterated function wtt-jump.

CHAPTER IV

# Maximal totally $\alpha$-c.a. degrees

For a collection $\mathcal{F}$ of c.e. degrees, we say that a degree $\mathbf{a} \in \mathcal{F}$ is *maximal* in $\mathcal{F}$ if it is maximal as an element of the partial ordering induced on $\mathcal{F}$ by the ordering on the Turing degrees. In other words, if there is no degree $\mathbf{b} > \mathbf{a}$ in $\mathcal{F}$.

Classes of c.e. degrees which contain maximal elements are rare; they are mostly prevented by density considerations. For example, no jump classes contain maximal elements, and there are no maximal cappable degrees. A notable exception is the example of the *contiguous* degrees – those degrees all of whose c.e. elements have the same weak truth-table degree. Cholak, Downey and Walk [9] showed that there are maximal contiguous degrees. Since the contiguous degrees are definable in the c.e. degrees (Downey and Lempp [26]), the maximal contiguous degrees form a definable antichain of c.e. degrees.

The relevance of contiguous degrees to the study in hand is that contiguous degrees are all array computable, that is, uniformly totally $\omega$-c.a. Like the contiguous degrees, the maximality phenomenon occurs in various level of the hierarchy discussed in Chapter III.

## 1. Existence of maximal totally $\omega^\alpha$-c.a. degrees

THEOREM 1.1. *If $\alpha \leqslant \varepsilon_0$ is a power of $\omega$, then there is a maximal totally $\alpha$-c.a. c.e. degree.*

To prove Theorem 1.1, fix an ordinal $\alpha \leqslant \varepsilon_0$ which is a power of $\omega$; we enumerate a c.e. set $D$ whose Turing degree will be maximal totally $\alpha$-c.a. To ensure that $\deg_{\mathrm{T}}(D)$ is totally $\alpha$-c.a., we meet, for each $e < \omega$, the requirements

$Q_e$**:** If $\Phi_e(D)$ is total, then $\Phi_e(D)$ is $\alpha$-c.a.

To ensure maximality, for each $e < \omega$, we want to ensure that either $W_e \leqslant_{\mathrm{T}} D$, or that there is some $f \leqslant_{\mathrm{T}} D \oplus W_e$ which is not $\alpha$-c.a. We enumerate a Turing functional $\Lambda_e$, with the aim of showing that either $W_e \leqslant_{\mathrm{T}} D$ or $\Lambda_e(D, W_e)$ is not $\alpha$-c.a. By Proposition II.1.7 let $\left\langle\left\langle f^i_s, o^i_s\right\rangle_{s<\omega}\right\rangle_{i<\omega}$ be an effective list of tidy $(\alpha+1)$-computable approximations such that letting $f^i = \lim_s f^i_s$, the sequence $\left\langle f^i\right\rangle$ lists the $\alpha$-c.a. functions; and as above, every$\alpha$-c.a. function appears as $f^i$ for some $i$ such that the approximation $\left\langle f^i_s, o^i_s\right\rangle$ is eventually $\alpha$-computable. For $e, i < \omega$, we try to meet the requirement

$P^i_e$**:** If $\left\langle f^i_s, o^i_s\right\rangle$ is eventually $\alpha$-computable then either $W_e \leqslant_{\mathrm{T}} D$ or $\Lambda_e(D, W_e) \neq f^i$.

Globally we need to ensure that for all $e$, $\Lambda_e(D, W_e)$ is total.

*Discussion.* The construction is not difficult. To meet a requirement $Q_e$ we use the mechanism proving the theorems in Chapter III: a node $\tau$, working for $Q_e$, measures an approximation to the question "is $\Phi_e(D)$ total?"; in the case of an affirmative answer, initialisation of weaker nodes that guess incorrectly allows $\tau$ to devise an $\alpha$-computable approximation for $\Phi_e(D)$.

A node $\sigma$ working for a requirement $P^i_d$ would like to appoint a follower $p$ and follow the strategy of nodes working for positive requirements in the constructions of Chapter III: whenever $f^i_s(p) = \Lambda_{d,s}(D_s, W_{e,s}, p)$, to enumerate $\lambda_{d,s}(p)$ into $D_{s+1}$. This action may interfere with the work done by a node $\tau$ for some requirement $Q_e$ such that $\tau\hat{}\infty \preccurlyeq \sigma$. However, unlike previous constructions, when $\sigma$ picks $p$ we do not know yet the ordinal bound on the "number of times" $\sigma$ may need to act for $p$; the functions $o^i_s$ are in some sense partial, since they allow the value $\alpha$, which for us is useless.

We isolate three principles which guide the interaction between $\tau$ and $\sigma$ extending $\tau\hat{}\infty$. These have been followed in previous constructions as well, but sometimes more easily since the approximations were "total". Let $p$ be a follower for $\sigma$, a node working for $P^i_d$.

(a) Suppose that $\tau$ first certifies a computation $\Phi_e(D,x)[s]$ at stage $s$ (in previous notation, $s = s_{i(x)}$). If $o^i_s(p) < \alpha$, then $\tau$ can incorporate this ordinal to the bound on its mind-changes for $\Phi_e(D,x)$. It can thus allow every future action for $p$ to injure $\Phi_e(D,x)$.

(b) If the use $\lambda_{d,t}(p)$ is chosen at a stage $t$ at which we see $\Phi_e(D,x)$ converge, then the next action for $p$ will not injure $\Phi_e(D,x)[t]$.

(c) Since $\Lambda_d$ is global, $\sigma$ needs to define $\lambda_d(p)$ immediately when it appoints $p$, that is, before it sees $o^i(p) < \alpha$.

We remark that we could have made the definition of each $\Lambda_d$ local, tied to a "mother node" $\eta$ as in the proof of Theorem III.4.2. However, in this construction this is not necessary and would not give any benefit. The effect of the finitely many mother nodes $\eta \prec \tau$ would be the same as the effect of having every $\Lambda_d$ be global, i.e. the root of the tree is the mother node for every $\Lambda_d$.

The principles outlined leave one potentially problematic sequence of events. First $\sigma$ appoints $p$ and defines $\lambda_d(p)$; then $\tau$ certifies $\Phi_e(D,x)$; and only later do we see $o^i(p) < \alpha$. In this case, the use is too small, so action for $p$ would injure the certified computation; but $\tau$ did not know how many times $\sigma$ will act for $p$ when it certified the computation. Note that $\tau$ could not wait for this later event, since we may never see $o^i(p) < \alpha$. Of course, this is where we use the additional computational power of $W_d$. Before we see $o^i(p) < \alpha$, $\sigma$ does not need to act for $p$. Once we see $o^i(p) < \alpha$, if $W_d \not\leqslant_{\mathrm{T}} D$, then $W_d$ will permit $\sigma$ to lift the use $\lambda_d(p)$ beyond the use of a computation $\Phi_e(D,x)$, in fact beyond the use of a $D$-correct such computation. Only then is $p$ *cleared* by $\tau$ and $\sigma$ can attack with impunity. We cannot expect that every follower we appoint is permitted, and so $\sigma$ will need to appoint a sequence of followers $p_0, p_1, \dots$; one of them will be permitted.

We note two issues. One is that while $W_d$ will permit some follower appointed by $\sigma$, the stage at which it gives this permission is not necessarily a stage at which $\sigma$ is accessible, and this permission cannot "remain open" until $\sigma$ is next visited: $\sigma$ may never be visited again, and we need to define $\lambda_d(p)$ to keep $\Lambda_d(D, W_d)$ total. So we act on permissions immediately, even if $\sigma$ is not accessible; this does no harm to the rest of the construction.

The other issue is that of totality. For each follower $p$, we note which computations $\Phi_e(D, x)$ it is not allowed to injure, and seek permission from $W_d$ at a stage at which $\Phi_e(D, x)\downarrow$ for all such computations. We are guaranteed eventual permission only if these are $D$-correct computations. How do we know that such a stage will occur? Of course $\sigma$, since it extends $\tau\hat{}\infty$, *guesses* that $\Phi_e(D)$ is total. But there are constructions in which $\tau\hat{}\infty$ lies on the true path but the measured function $\Phi_e(D)$ is in fact not total. This is avoided in this construction because we make $D$ totally $\alpha$-c.a. and so low$_2$.

*The tree of strategies.* As usual, to define the tree, we specify recursively the association of nodes to requirements, and specify the outcomes of nodes working for particular requirements. To specify the priority ordering of nodes, we specify the ordering between outcomes of any node.

We order all of the requirements $Q_d$ and $P^i_e$ in order-type $\omega$; all nodes of length $k$ work for the $k^{\text{th}}$ requirement on the list. The outcomes of a node working for $Q_e$ are $\infty$ and $\texttt{fin}$, with $\infty < \texttt{fin}$. A node working for $P^i_e$ has only one outcome.

*Clearing followers.* A follower $p$ for a node $\sigma$ working for $P^i_e$ can be in one of three states.

(1) When $p$ is first appointed, it is *unready.*
(2) At a later stage (at which $\sigma$ is accessible) we may see that $o^i_s(p) < \alpha$; then $p$ becomes *ready*: we have determined which computations $\Phi_d(D, x)$ it is allowed to injure.
(3) At a later stage yet, $W_e$ may give permission to lift the use $\lambda_e(p)$ and begin an attack with $p$. We say that $p$ is *in the clear.*

Let $\texttt{prec}(\sigma)$ be the collection of nodes $\tau$ such that $\tau$ works for a requirement $Q_d$ and $\tau\hat{}\infty \leqslant \sigma$. This is the collection of nodes that may need to restrain $\sigma$'s action to protect computations they are monitoring. For each follower $p$ for $\sigma$, if $p$ becomes ready (by observing that $o^i(p) < \alpha$) then we define, for each $\tau \in \texttt{prec}(\sigma)$, a value $m^\tau(p)$, which serves as a watermark. If $\tau$ works for $Q_d$, then action by $\sigma$ for $p$ is allowed to injure computations $\Phi_{d,s}(D_s, x)$ for $x \geqslant m^\tau(p)$, but not for smaller values of $x$.

*Construction.* At each stage we will do one of two things. Normally we will build the path of accessible nodes and act accordingly. But at some stages we will observe $W_e$ permissions that will allow us to clear a follower for some $\sigma$. In that case no node is accessible at that stage and no other action is taken by any node. In both cases, though, after the main action, we *maintain* functionals (work toward making them total).

Option A. At stage $s$ we first ask: is there some node $\sigma$ working for a positive requirement $P^i_e$ which currently has a ready follower $p$ such that:

- $p \notin \operatorname{dom} \Lambda_{e,s}(D_s, W_{e,s+1})$; and
- for all $\tau \in \texttt{prec}(\sigma)$, working for $Q_d$, we have $m^\tau(p) \leqslant \operatorname{dom} \Phi_{d,s}(D_s)$.

If so, then we let $\sigma$ be the strongest such node. We pick such a follower $p$ for $\sigma$, and declare it to be *in the clear.* We cancel all other followers for $\sigma$. We let $D_{s+1} = D_s$. We define $\Lambda_{e,s+1}(D_{s+1}, W_{e,s+1}, p) = s + 1$ with large use (the $D$-use and the $W_e$-use will always be equal). We initialise all nodes weaker than $\sigma$. For any pair $(d, q) \leqslant s$ distinct from $(e, p)$ we maintain $\lambda_d(q)$ as follows, and then end the stage.

Maintaining $\lambda_d(q)$: If $q \notin \operatorname{dom} \Lambda_{d,s}(D_{s+1}, W_{d,s+1})$, then we define a new computation $\Lambda_{d,s+1}(D_{s+1}, W_{d,s+1}, q) = s + 1$ with use $\lambda_{d,s+1}(q)$ determined by cases:

- If $q$ is currently a follower for a node $\sigma'$ working for $P_d^j$ for some $j$ (in particular, $q$ was not just cancelled), then we set $\lambda_{d,s+1}(q) = \lambda_{d,s}(q)$.
- Otherwise, $\lambda_{d,s+1}(q) = -1$.

The instructions will ensure that in the first case, $\lambda_{d,s}(q)$ is indeed defined, that is, $q \in \operatorname{dom} \Lambda_{d,s}(D_s, W_{d,s})$. The point of the first clause is to keep $\Lambda_d(D, W_d)$ total when we have $W_d$-changes which are not beneficial, i.e. occur when the follower $q$ is unready or $\operatorname{dom} \Phi_c(D)[s] < m^\tau(q)$ for some $\tau \in \mathtt{prec}(\sigma')$.

Option B. If option A was not taken, then we let, by recursion, the collection of accessible nodes $\delta_s$ be an initial segment of the tree of strategies. So the root of the tree is accessible at stage $s$.

Suppose that a node $\tau$ that works for requirement $Q_e$ is accessible at stage $s$. If $s$ is the least stage at which $\tau$ is accessible then we let $\tau\hat{\ }\infty \in \delta_s$. Otherwise we let $t$ be the last stage before $s$ at which $\tau\hat{\ }\infty$ was accessible. If $t < \operatorname{dom} \Phi_{e,s}(D_s)$ then we let $\tau\hat{\ }\infty \in \delta_s$. Otherwise we let $\tau\hat{\ }\mathtt{fin} \in \delta_s$.

Suppose that a node $\sigma$, working for requirement $P_e^i$, is accessible at stage $s$. There are two cases: either $\sigma$ has a unique follower which is in the clear; or no follower for $\sigma$ is in the clear. In the latter case, $\sigma$ possibly has a number of ready followers, and possibly one unready follower.

**1**. Suppose that $\sigma$ has follower $p$ in the clear.

If $\Lambda_{e,s}(D_s, W_{e,s}, p) = f_s^i(p)$ then we enumerate $\lambda_{e,s}(p)$ into $D_{s+1}$ and redefine $\Lambda_{e,s+1}(D_{s+1}, W_{e,s+1}, p) = s+1$ with large use. We initialise all nodes weaker than $\sigma$ and halt the stage.

If $\Lambda_{e,s}(D_s, W_{e,s}, p) \neq f_s^i(p)$ then the unique immediate successor on the tree of strategies is next accessible.

**2**. Suppose that $\sigma$ has no follower in the clear. There are two things we may do.

(a) If $\sigma$ has a currently unready follower $p$ and $o_s^i(p) < \alpha$, then we declare $p$ to be *ready*. For each $\tau \in \mathtt{prec}(\sigma)$, working for $Q_d$, we define $m^\tau(p) = \operatorname{dom} \Phi_{d,s}(D_s)$.

(b) If either the action in part (a) has just been performed, or $\sigma$ currently has no followers, then currently all followers for $\sigma$ are ready. We then appoint a new, large follower $p'$ for $\sigma$ (which is *unready*) and define $\Lambda_{e,s+1}(D_{s+1}, W_{e,s+1}, p') = s + 1$ with large use.

If neither (a) nor (b) are performed then $\sigma$ already has one unready follower $p$ with $o_s^i(p) = \alpha$, and we do nothing.

If $|\sigma| < s$, then the unique immediate successor on the tree of strategies is next accessible; otherwise we halt the stage. In case 2, we do not initialise weaker nodes even if we appoint a new follower. This is because if $W_e \leqslant_T D$, it is possible that infinitely many followers will be appointed.

At the end of the stage, we maintain $\lambda_d(q)$ for pairs $(d, q) \leqslant s$ (other than pairs for which $\Lambda_d(D, W_d, q)[s + 1]$ has just been defined) as above.

*Verification.* For a while, we follow the verifications for Theorem III.2.1. We have an analogue of Lemma III.2.3. In the verification, we say that a node $\sigma$ *acts* at a stage $s$ if either it is accessible at stage $s$ and enumerates a number into $D_{s+1}$ on behalf of a follower in the clear; or if stage $s$ option A is taken and a follower for $\sigma$ is cleared.

As indicated in the construction, if a follower $p$ for $\sigma$ is cleared at some stage $s$, then all other followers for $\sigma$ are cancelled at that stage. Until possibly a later stage at which $\sigma$ is initialised, $p$ remains $\sigma$'s unique follower.

Lemma 1.2. *Let $s$ be a stage.*

(a) *Every functional $\Lambda_{e,s}$ is consistent for the pair $D_s, W_{e,s}$.*

*Suppose that at the beginning of stage $s$, $p$ is a follower for a node $\sigma$ which works for $P^i_e$.*

(b) $\Lambda_{e,s}(D_s, p)\downarrow$ *and* $\lambda_{e,s}(p) \notin D_s$.

(c) *Suppose that $p'$ is a follower for a node $\sigma'$, weaker than $\sigma$, working for $P^{i'}_{e'}$. Then $\lambda_{e,s}(p) \neq \lambda_{e',s}(p')$. If $p$ is in the clear at the beginning of stage $s$, then $\lambda_{e,s}(p) < p'$. As usual $p' < \lambda_{e',s}(p')$.*

*Let $t < s$, and suppose that $p$ was already a follower for $\sigma$ at the beginning of stage $t$.*

(d) *If $p$ was in the clear at stage $t$, then $D_t \restriction_{\lambda_{e,t}(p)} = D_s \restriction_{\lambda_{e,t}(p)}$; if, in addition, $\sigma$ did not act at any stage $r \in [t, s)$, then $D_t \restriction_{\lambda_{e,t}(p)+1} = D_s \restriction_{\lambda_{e,t}(p)+1}$.*

(e) *If $p$ is not in the clear at the beginning of stage $s$ then $\lambda_{e,t}(p) = \lambda_{e,s}(p)$.*

Proof. Similar to the proof of Lemma III.2.3. We note the differences. For (b), that $\Lambda_{e,s}(D_s, p)\downarrow$ is here immediate, from the maintenance round we do at the end of every stage. To show that $\lambda_{e,s}(p) \notin D_s$, the new case is if at stage $s-1$, when performing maintenance, we saw that $\Lambda_{e,s-1}(D_s, W_{e,s}, p)\uparrow$, and defined a new computation with $\lambda_{e,s}(p) = \lambda_{e,s-1}(p)$. However, by induction, $y = \lambda_{e,s-1}(p) \notin D_{s-1}$. The node $\sigma$ does not act at stage $s-1$, and the first part of (c) (at stage $s-1$) shows that no other node can enumerate $y$ into $D_s$.

For (c), we note that as usual, new uses $\lambda_{e,s}(p)$ are chosen to be large, and so distinct from existing uses. The second part follows from the fact that at the stage at which $p$ is cleared, $\sigma'$ is initialised. The proof of (d) is identical to the previous proof. (e) is new, and follows immediately by induction, since $\sigma$ never acts for $p$ before $p$ is cleared, and once the use $\lambda_{e,t}(p)$ is picked (at the stage at which $p$ is appointed), the use is never lifted (see maintenance step). □

The proof of Lemma III.2.4 gives its analogue, recalling, though, that we say that $\sigma$ acts for $p$ at stage $s$ only if $p$ is cleared at stage $s$, or if $\sigma$ enumerates $\lambda_{e,s}(p)$ into $D_{s+1}$ (when $p$ is already in the clear); not when $p$ is appointed or is declared ready.

Lemma 1.3. *Let $\sigma$ be a node that works for requirement $P^i_e$. Let $p$ be a follower for $\sigma$ at stages $s < t$, and suppose that at both stages, $\sigma$ acts for $p$. Then $o^i_t(p) < o^i_s(p)$.*

It follows that for each $p$, $\sigma$ enumerates $\lambda_e(p)$ into $D$ at only finitely many stages. If the construction is fair to $\sigma$, then it follows that $\sigma$ halts the stage at most finitely many times after it is last initialised: at most once when a follower $p$ becomes cleared, and then finitely many times when it enumerates $\lambda_{e,s}(p)$ into $D$.

LEMMA 1.4. *The true path $\delta_\omega$ is infinite, and the construction is fair to every node on the true path.*

PROOF. The point is that there are infinitely many stages at which we do not take option A and stop the stage: there are infinitely many stages at which $\delta_s$ is nonempty. Suppose for a contradiction that there is a last stage $s^*$ at which we take option B. There are only finitely many nodes $\sigma$ which have followers at the end of stage $s^*$. But for each such node $\sigma$ there is at most one stage $s > s^*$ at which we act for $\sigma$. At that stage, a follower for $\sigma$ is cleared. Either this follower is never cancelled and $\sigma$ does not act again. Or $\sigma$ is initialised at some later stage but never has the chance to appoint new followers. This is a contradiction. □

LEMMA 1.5. *For all $e$, $\Lambda_e(D, W_e)$ is total.*

PROOF. The difference from the proof of Lemma III.2.7 is that $W_e$-changes may make $\Lambda_e$-computations diverge. The maintenance step, and in particular keeping the use fixed unless a follower becomes cleared, addresses this issue. Formally, the convergence of $\Lambda_e(D, W_e, p)$ for a permanent follower $p$ for $\sigma$ follows from Lemma 1.2(e) if $p$ is never cleared, and from Lemma 1.3 if it is. □

The argument of Lemma III.2.5 now shows that if a node $\sigma$ on the true path, working for requirement $P^i_e$, has a follower which is eventually cleared but never cancelled, then $\Lambda_e(D, W_e) \neq f^i$.

As mentioned above, perhaps surprisingly, in order to show that each finitary requirement $P^i_e$ is met, we need to investigate the infinitary requirements first. The verification for the finitary requirements will use the fact that $\deg_{\mathrm{T}}(D)$ is low$_2$.

Fix a node $\tau$, working for requirement $Q_e$, such that $\tau\hat{}\infty$ lies on the true path. By Lemma 1.4, let $s^*$ be the last stage at which $\tau$ is initialised. Let $S = \{s_0, s_1, \dots\}$ be the collection of stages $s > s^*$ at which $\tau\hat{}\infty$ is accessible. For $x < \omega$, let $i(x)$ be the least $i$ such that $x < \operatorname{dom}\Phi_e(D)[s_i]$. For $x < \omega$, we let $a(x)$ be the collection of pairs $(\sigma, p)$ such that $\sigma \geqslant \tau\hat{}\infty$ (in other words $\tau \in \texttt{prec}(\sigma)$), and $p$ is a follower for $\sigma$ which became ready at some stage *prior* to stage $s_{i(x)}$, but is not cancelled by stage $s_{i(x)}$. For $j \geqslant i(x)$ we let $a_j(x)$ be the collection of pairs $(\sigma, p) \in a(x)$ such that $\sigma$ is not initialised at any stage $r \in [s_{i(x)}, s_j)$, and $p$ is still a follower for $\sigma$ at the beginning of stage $s_j$.

The set $a(x)$ plays the same role as it did in the proof of Theorem III.2.1: only action by $\sigma$ for some $p$ such that $(\sigma, p) \in a_j(x)$ can injure a computation $\Phi_e(D, x)$ at stage $s_j$. This will show that $\Phi_e(D)$ is $\alpha$-c.a., as $a(x)$ is finite, effectively obtained from $x$, and at stage $s_{i(x)}$, we already know an ordinal bound $o^k_t(p)$ on the "number of times" $\sigma$ can attack with $p$. Note that for each $\sigma$ there is at most one $p$ such that $(\sigma, p) \in a(x)$ and $\sigma$ will attack with $p$ at a later stage $s_j$. However, the identity of this $p$ – the one follower for $\sigma$ that will be cleared, if there is one – is not yet known at stage $s_{i(x)}$.

LEMMA 1.6. *Let $\sigma \geqslant \tau\hat{}\infty$, working for $P^i_d$, and let $p$ be a follower for $\sigma$ which is already in the clear at the beginning of stage $s \geqslant s_{i(x)}$. Suppose that $(\sigma, p) \notin a(x)$. Then:*

(1) *$m^\tau(p) > x$.*

(2) *Let $t$ be the stage at which $p$ is cleared. Then $x \in \operatorname{dom}\Phi_e(D)[t]$ and $D_t \restriction_{\varphi_{e,t}(x)} = D_s \restriction_{\varphi_{e,t}(x)}$. It follows of course that $x \in \operatorname{dom}\Phi_e(D)[s]$ and that $\varphi_{e,s}(x) = \varphi_{e,t}(x)$.*

(3) $\lambda_{d,s}(p) > \varphi_{e,s}(x)$.

PROOF. For (1), let $w$ be the stage at which $p$ is declared ready. If $w < s_{i(x)}$ then $(\sigma, p) \in a(x)$, so $w \geqslant s_{i(x)}$ (and it follows that $t > s_{i(x)}$). At stage $w$, $\sigma$ is accessible, and so $w = s_j$ for some $j \geqslant i(x)$, whence $x < \operatorname{dom}\Phi_{e,w}(D_w) = m^\tau(p)$.

At stage $t$ we have $\operatorname{dom}\Phi_e(D)[t] \geqslant m^\tau(p)$ – this is one of the conditions for $p$ to be cleared. Hence $x < \operatorname{dom}\Phi_e(D)[t]$, so $\varphi_{e,t}(x)$ is indeed defined. Let $u = \varphi_{e,t}(x)$. At stage $t$, we define $\lambda_{d,t+1}(p)$ to be large, and so larger than $u$.

At stage $t$ no node is accessible, so $D_{t+1} = D_t$. Lemma 1.2(d) applied to $t+1 \leqslant s$ says that $D_s \restriction_{\lambda_{d,t+1}(p)} = D_{t+1} \restriction_{\lambda_{d,t+1}(p)}$, and (2) follows.

As $\lambda_{d,r}(p)$ is non-decreasing with $r$, it follows that $\lambda_{d,s}(p) > u = \varphi_{e,s}(x)$. ☐

We are now ready to prove an analogue of Lemma III.2.9.

LEMMA 1.7. *Let $j \geqslant i(x)$. Let $u = \varphi_{e,s_j}(x)$. Suppose that $D_{s_{j+1}} \restriction_u \neq D_{s_j} \restriction_u$. Then there is some $(\sigma, p) \in a_j(x)$ such that $\sigma$ acts for $p$ at stage $s_j$ and enumerates $\lambda_{d,s_j}(p) < u$ into $D_{s_j+1}$.*

PROOF. The argument follows the proof of Lemma III.2.9. Suppose that at stage $s \in [s_j, s_{j+1})$, a node $\sigma$ acts for some follower $p$ and enumerates $\lambda_{d,s}(p) < \varphi_{e,s_j}(x)$ into $D_{s+1}$. The argument that $\sigma$ extends $\tau\hat{\ }\infty$, and so $s = s_j$, is the same as above. Note that $p$ is already in the clear at the beginning of stage $s_j$. Lemma 1.6(3) shows that $(\sigma, p) \in a(x)$, and so $(\sigma, p) \in a_j(x)$. ☐

The next lemma shows that $D$ is low$_2$.

LEMMA 1.8. *Let $\tau$ be a node on the true path that works for requirement $Q_e$. Then $\tau\hat{\ }\infty$ lies on the true path if and only if $\Phi_e(D)$ is total.*

PROOF. The non-trivial direction is left-to-right. Let $x < \omega$. To show that $x \in \operatorname{dom}\Phi_e(D)$, we observe that there are only finitely many $j \geqslant i(x)$ such that $D_{s_{j+1}} \restriction_{\varphi_{e,s_j}(x)} \neq D_{s_j} \restriction_{\varphi_{e,s_j}(x)}$. This follows from the fact that $a(x)$ is finite, and that for each $(\sigma, p) \in a(x)$, $\sigma$ acts for $p$ at most finitely many times. ☐

We can now show that the positive requirements are met.

LEMMA 1.9. *For all $e$ and $i$, the requirement $P^i_e$ is met.*

PROOF. Let $\sigma$ be a node on the true path, working for $P^i_e$. We observed above that if there is a follower $p$ for $\sigma$ which is at some point cleared and is never cancelled, then $P^i_e$ is met. Let $r^*$ be the last stage at which $\sigma$ is initialised, and suppose that no follower for $\sigma$ is cleared after stage $r^*$. If $\langle f^i_s, o^i_s \rangle$ is not eventually $\alpha$-computable, then $P^i_e$ is met vacuously, so we assume that it is. Then every follower that $\sigma$ appoints after stage $r^*$ eventually becomes ready (of course, using the fact that $\sigma$ is accessible during infinitely many stages). Then $\sigma$ appoints infinitely many followers. We show that $W_e \leqslant_T D$.

Let $p$ be a follower for $\sigma$, appointed after stage $r^*$; let $s_0$ be the stage at which $p$ is appointed, and let $u = \lambda_{e,s_0}(p)$. As $u > s_0$, the numbers $u$ are unbounded, as $p$ ranges over the followers for $\sigma$. To compute $W_e \restriction_u$ from $D$, we first go to the stage $t$ at which $p$ becomes ready. At that stage we observe the numbers $m^\tau(p)$ for

$\tau \in \mathtt{prec}(\sigma)$. For all $\tau \in \mathtt{prec}(\sigma)$, $\tau\hat{\ }\infty$ lies on the true path. By Lemma 1.8, there is a stage $s$ at which for all $\tau \in \mathtt{prec}(\sigma)$, for all $x < m^\tau(p)$, $x \in \operatorname{dom}\Phi_e(D)[s]$ by a $D$-correct computation. Certainly $D$ can find such a stage $s$; and $W_{e,s}\restriction_u = W_e\restriction_u$, for otherwise $p$ would be cleared at some stage $s' > s$. □

We now rejoin the proof of Theorem III.2.1, using Lemma 1.7 to show that for every $e$ such that $\Phi_e(D)$ is total, the node $\tau$ on the true path working for $Q_e$ is successful in devising an $\alpha$-computable approximation for $\Phi_e(D)$. Fix such $e$ and $\tau$; we again use the stages $s_i$, the indices $i(x)$ and the sets $a_j(x)$ discussed above. Fix $x < \omega$. We note, and this is the main point, that for all $(\sigma, p) \in a(x)$, if $\sigma$ works for $P^i_d$ then $o^i_{s_{i(x)}}(p) < \alpha$.

Let $j \geqslant i(x)$ and let $\sigma$ be a node, working for $P^i_d$, which appears in $a_j(x)$ (i.e., $(\sigma, p) \in a_j(x)$ for some $p$). If no follower for $\sigma$ is cleared by the beginning of stage $s_j$, we let

$$\beta_j(\sigma) = \max\left\{o^i_{s_j}(p) \,:\, (\sigma, p) \in a(x)\right\}.$$

Otherwise, let $p$ be the unique follower for $\sigma$ at stage $s_j$; $(\sigma, p) \in a_j(x)$. We let $t_j(\sigma)$ be the greatest stage $t < s_j$ at which $\sigma$ acts (for $p$); such a stage exists, since $p$ becomes cleared at some stage $t < s_j$. We then let $\beta_j(\sigma) = o^i_{t_j(\sigma)}(p)$. Finally, we order the nodes appearing in $a_j(x)$ in descending priority as $\sigma_0, \sigma_1, \dots, \sigma_{k(j)}$, and let $m_j(x) = \sum_{k \leqslant k(j)} \beta_j(\sigma_k)$. We note that if $\sigma_k$ acts at stage $s_j$ then $k(j+1) \leqslant k$. Lemma III.2.10 holds for the current construction, with much the same proof. This completes the proof of Theorem 1.1.

**1.1. Maximal uniformly totally $\omega^\alpha$-c.a. degrees.** Not only are there maximal uniformly totally $\omega^\alpha$-c.a. degrees, but there are such degrees which are also maximal totally $\omega^\alpha$-c.a.

THEOREM 1.10. *If $\alpha$ is a power of $\omega$, then there is a uniformly totally $\alpha$-c.a. degree which is maximal totally $\alpha$-c.a.*

PROOF. To prove Theorem 1.10, we run the construction for Theorem 1.1 with but one modification: a follower $p$ for a node $\sigma$ working for $P^i_e$ becomes ready at a stage $t_1$ if $\sigma$ is accessible at stage $t_1$, and at the previous stage $t_0 < t_1$ at which $\sigma$ was accessible we saw that $o^i_{t_0}(p) < \alpha$. That is, we only let $p$ become ready at the *second* stage at which $\sigma$ is accessible and at which we see $o^i_t(p) < \alpha$. It is easily verified that this delay in declaring a follower to be ready does not affect the success of the construction, so the degree $\deg_{\mathrm{T}}(D)$ produced under this new definition of readiness is also maximal totally $\alpha$-c.a.; we show though that the degree produced is also uniformly totally $\alpha$-c.a.

We follow the argument for proving part (1) of Theorem III.3.5. By design of the current construction, a node $\sigma$ accessible at stage $s$ has length at most $s$. We fix some $\tau$, working for $Q_e$, such that $\tau\hat{\ }\infty$ lies on the true path. Now we examine the proof of Lemma III.3.7. For $x \geqslant \operatorname{dom}\Phi_e(D)[s_1]$, again let $u_0 < u_1 < s_{i(x)}$ be successive stages at which $\tau\hat{\ }\infty$ is accessible. Let $(\sigma, p) \in a(x)$, with $\sigma$ working for $P^i_d$. Then $o^i_{u_0}(p) < \alpha$, and $u_0 < x$. Since $|\sigma| \leqslant u_0$, we may assume that $i < x$. It follows that $m_{i(x)}(x)$ is an ordinal which can be observed at stage $x$ of the construction, and this is independent of $\tau$. This gives an $\alpha$-order function $h$ such that every $f \leqslant_{\mathrm{T}} D$ is $h$-c.a. □

In Section III.3.2 we explained why we could not combine the proofs of the two parts of Theorem III.4.2 and obtain a contradiction (a degree which both is and is not uniformly totally $\alpha$-c.a.). The explanation focussed on the last stage $u_1 = s_{i(x)-1}$, the last stage in $S$ before stage $s_{i(x)}$. A follower $p$ appointed at stage $u_1$ would have bound $o^i_{u_1}(p)$ which can be arbitrarily large with relation to $p$, but will be able to destroy computations $\Phi_e(D,x)[s_j]$ for $j \geqslant i(x)$. In the previous chapter there is no way around this; we have to allow such a $p$ to destroy the computations, or $\sigma$ will not be able to meet its requirement. In the current situation, using $W_d$ to lift the use $\lambda_e(p)$ when $p$ is cleared allows us to choose which followers to restrain, and this enables the proof of Theorem 1.10.

For the case $\alpha = \omega$, Theorem 1.10 says that there is an array computable c.e. degree which is maximal totally $\omega$-c.a. In fact, we suspect that combining the methods of this chapter together with the construction of a contiguous degree, one can show that there is a contiguous degree which is maximal totally $\omega$-c.a. Since every contiguous degree is array computable, such a degree is also maximal contiguous.

The following theorem, for $\alpha = \omega$, shows that not all maximal totally $\omega$-c.a. degrees are maximal contiguous degrees.

Theorem 1.11. *If $\alpha$ is a power of $\omega$, then there is a maximal totally $\alpha$-c.a. degree which is not uniformly totally $\alpha$-c.a.*

Sketch of proof. We combine the construction for Theorem 1.1 with the technique proving Theorem III.3.5(2). To the construction for Theorem 1.1 we add the enumeration of a functional $\Gamma$, with the aim of making $\Gamma(D)$ witness that $\deg_{\mathrm{T}}(D)$ is not uniformly totally $\alpha$-c.a. Again we fix an $\alpha$-order function $h$, and enumerate $h$-c.a. functions $\langle g_i \rangle$ along with tidy $(h+1)$-computable approximations for these functions. We add a third kind of requirement, $R^i$, namely that $\Gamma(D) \neq g_i$. The action for these requirements is identical to that of the previous chapter. There is no interaction (other than mutual initialisations) between nodes working for $R^i$ and nodes working for $P^j_d$; and the interaction between nodes working for $R^i$ and nodes working for $Q_e$ is as in the previous chapter. That is, when showing that $D$ is low$_2$, and then devising an $\alpha$-computable approximation for $\Phi_e(D)$ if it is total, the sets $a(x)$ may contain pairs $(\sigma, p)$ where $\sigma$ works for either a requirement $R^i$ or for a requirement $P^j_d$. In either case, the ordinal bound on the number of times $\sigma$ will act for $p$ can be observed at stage $s_{i(x)}$, and if $(\sigma, p)$ is not in $a(x)$, then action by $\sigma$ for $p$ cannot injure a computation $\Phi_{e,s}(D_s, x)$ observed at a $\tau$-expansionary stage. $\square$

## 2. Limits on further maximality

One might wish for even stronger maximality properties than those provided by Theorem 1.1. Could there be, for example, a totally $\omega$-c.a. degree which is maximal totally $\omega^2$-c.a. degree? In general, can a degree in one level of our crudest hierarchy be maximal for a higher level? The following theorem says it cannot.

Theorem 2.1. *Let $\beta < \epsilon_0$. Every totally $\omega^\beta$-c.a. c.e. degree is bounded by a strictly greater totally $\omega^{\beta+1}$-c.a. c.e. degree.*

To prove Theorem 2.1, fix an ordinal $\beta < \epsilon_0$, and let $\alpha = \omega^\beta$. Let $V$ be a c.e. set whose Turing degree is totally $\alpha$-c.a. We enumerate a set $D$ such that $\deg_{\mathrm{T}}(V \oplus D)$ is strictly greater than $\deg_{\mathrm{T}}(V)$ and is totally $\alpha \cdot \omega = \omega^{\beta+1}$-c.a. The requirements to meet are:

$P_e$**:** $\Psi_e(V) \neq D$;

and

$Q_e$**:** If $\Phi_e(V, D)$ is total then it is $\alpha \cdot \omega$-c.a.

*Discussion.* The main idea for meeting the requirement $P_e$ is as follows. We track $\Phi_e(V, D, x)$ for some $x$. Changes to such a computation can come from two sources: a $V$-change or a $D$-change. To keep track of the $V$-changes – the ones we do not control ourselves – we build what we call a "shadow functional" $\hat{\Phi}_e$, with intended oracle $V$ alone. We pick an input $c$ and define $\hat{\Phi}_e(V, c)$ with the same use as that of $\Phi_e(D, V, x)$ (recall that we assume that the $V$-use and the $D$-use are identical). The input $c$ is called the *tracker* for $x$. We ensure that if $\Phi_e(D, V)$ is total, then $\hat{\Phi}_e(V)$ is total as well. Since $\deg_{\mathrm{T}}(V)$ is totally $\alpha$-c.a., $\hat{\Phi}_e(V)$ will equal $f^i$ for some $i$, where $\langle f^i \rangle$ lists $\alpha$-c.a. functions. We guess the correct index $i$; this will be done using the fact that $V$ is low$_2$. This is a $\Delta^0_3$-guessing process, which is very similar to a $\Pi^0_2/\Sigma^0_2$ process, except that infinitely many outcomes are required. The correct guess will observe $o^i(c)$ and bound the $V$-changes in $\Phi_e(D, V, x)$.

We have to think though what happens when we cause a $D$-change (for the sake of meeting some $P_d$). The computation $\Phi_e(D, V, x)$ is gone, but it is possible that the $V$-part of the computation was correct. In this case $\hat{\Phi}_e(V, c)$ is a correct computation, and we cannot use the tracker $c$ to shadow new $\Phi_e(D, V, x)$ computations. We need to replace $c$ by a new tracker and repeat the process. This is how we get $\alpha \cdot \omega$: when we first certify $\Phi_e(D, V, x)$, we put a bound on the number of $D$-changes that we allow to destroy such a computation; say it is $n$. We appoint a tracker $c_0$ and observe $\beta_0 = o^i_0(c_0)$. We then declare that $\Phi_e(D, V, x)$ will not change more than $\alpha \cdot n + \beta_0$ many "times". While we only see $V$-changes, the associated ordinal is still $\alpha \cdot n + o^i_s(c_0)$. Once we cause a $D$-change that destroys a $\Phi_e(D, V, x)$ computation, we appoint a new tracker $c_1$, observe $\beta_1 = o^i_s(c_1)$, decrease our ordinal to $\alpha \cdot (n-1) + \beta_1$, and repeat the process.

We could be tempted to improve the bound. If we know in advance (i.e. when $\Phi_e(D, V, x)$ is first certified) a bound $n$ on the number of $D$-injuries to the computation, we could immediately appoint $n$ trackers $c_0, \ldots, c_{n-1}$ and start our approximation knowing $\beta_k = o^i_0(c_k)$ for all of these trackers. Then the bound would be $\beta_{n-1} + \beta_{n-1} + \cdots + \beta_0$ which in fact is smaller than $\alpha$. We would prove that there is no maximal totally $\alpha$-c.a. degree. The fallacy is easy to see: we do not know whether we will actually see $n$-many $D$-injuries to the computation; $n$ is just a bound. While we are using the tracker $c_0$ we cannot define computations $\hat{\Phi}_e(V, c_k)$ for the other trackers ($k > 0$); we need to keep them open, because the use of these computations is the use of $\Phi_e(x)$-computations we have not yet observed. This would make $\hat{\Phi}_e(V)$ partial even if $\Phi_e(D, V)$ is total, and so void the whole plan.

We now discuss how to meet $P_e$, bearing in mind the severe restriction imposed by the negative requirements: such requirements need to know in advance (relative to the input $x$) the number of times (in this instance without quotation marks) a $D$-change could ruin a computation $\Phi_d(D, V, x)$.

We pick a follower $p$ and wait for $\Psi_e(V, p)$ to converge, with the intention of ensuring that $\Psi_e(V, p) \neq D(p)$. Of course the difficulty is that we do not know, when presented with such a computation, whether the presented computation is $V$-correct. If $V$ were low we could apply R. Robinson's guessing technique. However $V$ need not be low. But it is low$_2$, and again we use this to guess the answer to the question "is $\Psi_e(V)$ total?".

Independent of the restrictions imposed by the negative requirements, ensuring that $D \nleqslant_{\mathrm{T}} V$ would now be easy. Define a $D$-computable function $\Lambda(D)$. Each outcome of $P_e$ which believes that $\Psi_e(V)$ is total appoints a follower $p$. If such an outcome is believed and we currently see that $\Psi_e(V, p) = \Lambda(D, p)$ then we diagonalise. If such an outcome lies on the true path then its guess is correct: $\Psi_e(V)$ is indeed total, and so the outcome would act only finitely many times.

Such action causes conflict with stronger negative requirements. To keep $\Lambda(D)$ total, a new value for $\lambda(D, p)$ needs to be picked immediately when an outcome of $P_e$ acts. This means that such an outcome will repeatedly injure a computation $\Phi_d(D, V, x)$. We could try to use the fact that $\deg_{\mathrm{T}}(V)$ is totally $\alpha$-c.a., rather than the weaker fact that it is low$_2$. We guess that $\Psi_e(V) = f^i$ for some $\alpha$-c.a. function $f^i$ on our list; the agent following $\Phi_d(D, V, x)$ will observe how many "times" the $P_e$-child will act, and incorporate it into its bound. The bound though is $\alpha$ rather than $\omega$. In this way we could try to make $D \oplus V$ totally $\alpha^2$-c.a., but not totally $\alpha \cdot \omega$-c.a. Of course for $\alpha = \omega$ this is sufficient.

To overcome this difficulty we modify the action of $P_e$ as follows. The problem was that even though we have certification that $\Psi_e(V)$ is total, many single computations we see will be incorrect. To respect the main restriction, after a failed attack with a follower we abandon that follower altogether. To ensure that this does not go on indefinitely we build a shadow functional $\hat{\Psi}_e$, with intended oracle $V$. We need to ensure that if $\Psi_e(V)$ is total then so is $\hat{\Psi}_e(V)$. Each agent that guesses totality appoints an *anchor* $q$ which will serve many followers $p$. We ensure that the uses of $\Psi_e(V, p)$ and $\hat{\Psi}_e(V, q)$ are the same. If the agent is correct then the fact that $\hat{\Psi}_e(V, q)$ stabilises ensures that only finitely many followers are ever appointed by that agent.

We need to discuss in greater detail how a node $\tau$ working for $Q_e$ can tolerate the action of a node $\sigma$ working for $P_d$. Assuming that the nodes $\sigma$ guesses that $\limsup_s \operatorname{dom} \Phi_e(V, D)[s] = \infty$, it also needs to guess whether $\liminf_s \operatorname{dom} \Phi_e(V, D)[s] = \infty$, that is, if $\Phi_e(V, D)$ is total or not. If $\sigma$ guesses that $\Phi_e(V, D)$ is total then for each $x$ we allow an enumeration of a follower for $\sigma$ to injure $\Phi_e(V, D, x)$ at most once. As in the construction of a maximal totally $\alpha$-c.a. degree, we set a "watermark" $m_s(\sigma)$, differentiating between large inputs whose computations $\sigma$ is allowed to injure, and smaller inputs which need to be protected. Each time $\sigma$ attacks, the watermark is updated. It is possible that due to a $V$-change, a follower $p$ is smaller than the use $\varphi_{e,s}(x)$ for some protected input $x < m_s(\sigma)$. In this case the $V$-change makes $\hat{\Psi}_d(V, q)\uparrow$, and we can discard the follower and choose a new, large one. Note that when this is done we do not need to update $m_s(\sigma)$: the node $\tau$ only cares about the number of followers that will injure a computation $\Phi_e(D, V, x)$, not about the identity of the follower that will inflict the injury.

What at first appears to be a trickier situation is when $\sigma$ guesses that $\Phi_e(V,D)$ is partial. We still need to protect computations $\Phi_e(V,D,x)$ for small $x$, because we don't know that $\sigma$'s guess is correct. This means cancelling a follower $p$ for $\sigma$ when we see a $V$-change that causes $\varphi_e(x)$ to increase. But if $\Phi_e(V,D,x)\uparrow$ then this can happen infinitely often. However, $\sigma$ can guess the exact place at which $\Phi_e(V,D)$ becomes partial, that is, the value of $\liminf_s \operatorname{dom} \Phi_e(V,D)[s]$. Say that value is $y$. Inputs $x<y$ will eventually settle and stop causing the cancellation of $\sigma$'s follower. When we guess the value $y$ we delay the definition of $\hat{\Phi}_\tau(V,c)$ where $c$ is the current tracker for $x$. Action by $\sigma$ at such a stage will not cause problems for stronger "totality outcomes" of $\tau$: if $\hat{\Phi}_\tau(V,c)\uparrow\ [s]$ then enumeration of a number into $D$ at stage $s$ does not mean that we need to abandon the tracker. On the other hand if $\Phi_e(V,D)$ is total then such $y$ will be guessed only finitely often and so $\hat{\Phi}_\tau(V,c)$ will eventually be defined and we can ensure that $\hat{\Phi}_\tau(V)$ is total as well, which is necessary for $\tau$'s strategy to work.

*The tree of strategies and $\Delta^0_3$ guessing.* We define the tree of strategies and assign strategies to nodes on the tree by recursion. The definition of the tree could be compacted a little but we believe that a more expansive description may aid the clarity of presentation.

We start with the empty node, to which we assign the requirement $Q_0$. Suppose that $\tau$ is a node on the tree which was assigned the requirement $Q_e$. The node will have a number of children on the tree which help $\tau$ meet its goal. The outcomes of $\tau$ are $\infty < \mathtt{fin}$. These outcomes measure $\limsup_s \operatorname{dom} \Phi_e(V,D)[s]$. The node $\tau\hat{\ }\mathtt{fin}$ is assigned to the requirement $P_e$.

The outcomes of $\tau\hat{\ }\infty$ on the tree are $\infty_n$ and $\mathtt{fin}_n$ for $n<\omega$ (ordered by $\infty_0 < \mathtt{fin}_0 < \infty_1 < \mathtt{fin}_1 < \infty_2 < \cdots$). These outcomes participate in the $\Delta^0_3$ guessing process of whether $\hat{\Phi}_\tau(V)$ is total or not. The nodes $\tau\hat{\ }\infty\hat{\ }\mathtt{fin}_n$ are assigned to the requirement $P_e$. The outcomes of nodes of the form $\tau\hat{\ }\infty\hat{\ }\infty_n$ are all $i<\omega$ (ordered naturally). A node $\tau\hat{\ }\infty\hat{\ }\infty_n$ guesses that $\hat{\Phi}_\tau(V)$ is total. If it is right then $\hat{\Phi}_\tau(V)$ must equal $f^i$ for some $i$, where $\langle f^i\rangle$ as usual is a list of the $\alpha$-c.a. functions equipped uniformly with tidy $(\alpha+1)$-computable approximations $\langle f^i_s, o^i_s\rangle$; this is guessed by the node $\tau\hat{\ }\infty\hat{\ }\infty_n\hat{\ }i$. We assign each such node the requirement $P_e$.

Suppose that a node $\pi$ is assigned the requirement $P_e$. The node $\pi$ has infinitely many outcomes $\infty_n$ and $\mathtt{fin}_n$, ordered as above. Again this is for guessing the totality of $\hat{\Psi}_\pi(V)$, a shadow functional enumerated by the node $\pi$. The children of $\pi$ — its immediate successors on the tree — combine forces to help $\pi$ meets its requirement. They each have a single immediate extension on the tree, which is assigned to the requirement $Q_{e+1}$.

As discussed, nodes $\tau$ working for $Q_e$ define a shadow functional $\hat{\Phi}_\tau$ and nodes $\pi$ working for $P_e$ define a shadow functional $\hat{\Psi}_\pi$. Since $V$ is $\mathrm{low}_2$, the set of indices of functionals $\Theta$ such that $\Theta(V)$ is total is $\Sigma^0_3$. Membership in a $\Pi^0_2$ set can be translated to the question whether a given non-decreasing computable sequence is bounded or not. By the recursion theorem we know the indices of the functionals enumerated by the nodes $\tau$ and $\pi$ on the tree. Thus we obtain for each such node $\mu$ a computable list $\ell_s(\mu,n)$ of sequences, nondecreasing in $s$, such that the functional enumerated by $\mu$ is total if and only if for some $n$, the sequence $\langle \ell_s(\mu,n)\rangle$ is unbounded.

As mentioned above, a node $\tau$ working for $Q_e$ appoints trackers $\mathtt{tr}_s(\tau, x)$ for inputs $x < \omega$. If $\sigma$ is a child of a node $\pi$ working for $P_e$ which believes that $\hat{\Psi}_\pi(V)$ is total (i.e. $\sigma = \pi\hat{\ }\infty_n$ for some $n < \omega$) then $\sigma$ may appoint both an anchor $\mathtt{ac}_s(\sigma)$ and a follower $\mathtt{fl}_s(\sigma)$. All followers, anchors and trackers are cancelled when the node which appointed them is initialised.

Suppose that $\pi$ is a node which works for $P_e$. We let $\mathtt{prec}(\pi)$ be the set of nodes $\tau$ working for some $Q_d$ such that $\tau\hat{\ }\infty \prec \pi$. We split this set into two parts: $\mathtt{prec}_\infty(\pi)$ is the set of nodes $\tau \in \mathtt{prec}(\pi)$ such that $\tau\hat{\ }\infty\hat{\ }\infty_n \prec \pi$ for some $n$; $\mathtt{prec}_{\mathtt{fin}}(\pi)$ is the set of nodes $\tau \in \mathtt{prec}(\pi)$ such that $\tau\hat{\ }\infty\hat{\ }\mathtt{fin}_n \preccurlyeq \pi$ for some $n$. If $\sigma$ is a child of $\pi$ which believes that $\hat{\Psi}_\pi(V)$ is total then during the construction we may define markers $m_s(\sigma)$. Let $\tau \in \mathtt{prec}(\pi)$ and let $x < \omega$. We say that $\sigma$ *respects* the input $x$ (for $\tau$) at stage $s$ if:

- $\tau \in \mathtt{prec}_\infty(\pi)$ and $x < m_s(\sigma)$; or
- $\tau \in \mathtt{prec}_{\mathtt{fin}}(\pi)$ and $x < y$, where $\tau\hat{\ }\infty\hat{\ }\mathtt{fin}_y \preccurlyeq \pi$.

*Construction.* Let $s$ be a stage. We let, by recursion, the collection of accessible nodes $\delta_s$ be an initial segment of the tree of strategies.

Suppose that a node $\tau$, working for requirement $Q_e$, is accessible at stage $s$. Let $t < s$ be the last stage prior to stage $s$ at which $\tau\hat{\ }\infty$ was accessible, $t = 0$ if there is no such stage. If $\operatorname{dom}\Phi_{e,s}(V_s, D_s) \leqslant t$ then we let $\tau\hat{\ }\mathtt{fin}$ be next accessible; otherwise we let $\tau\hat{\ }\infty$ be next accessible.

Suppose that $\tau\hat{\ }\infty$ is accessible at stage $s$. For each $n < s$ let $t_n$ be the last stage prior to stage $s$ at which $\tau\hat{\ }\infty\hat{\ }\infty_n$ was accessible, $t_n = 0$ if there was no such stage. Also, let $y$ be the least such that either $\Phi_{e,t}(V_t, D_t, y)\uparrow$ or the computation $\Phi_{e,t}(V_t, D_t, y)$ was destroyed since stage $t$, that is, either $D_t \upharpoonright_u \neq D_s \upharpoonright_u$ or $V_t \upharpoonright_u \neq V_s \upharpoonright_u$, where $u = \varphi_{e,t}(y)$. Note that $y \leqslant t$. If there is some $n \leqslant y$ such that $\ell_s(\tau, n) \geqslant t_n$ then we let $\tau\hat{\ }\infty\hat{\ }\infty_n$ be next accessible for the least such $n$. Otherwise we let $\tau\hat{\ }\infty\hat{\ }\mathtt{fin}_y$ be next accessible.

Before we proceed we maintain the functional $\hat{\Phi}_\tau$. Let $x < \omega$ such that $c = \mathtt{tr}_s(\tau, x)$ is already defined. If either

- $\tau\hat{\ }\infty\hat{\ }\mathtt{fin}_y$ is next accessible, and $x < y$; or
- $\tau\hat{\ }\infty\hat{\ }\infty_n$ is next accessible, and $x < t$

and $\hat{\Phi}_{\tau,s}(V_s, D_s, c)\uparrow$ then we define $\hat{\Phi}_{\tau,s+1}(V_s, D_s, c) = s$ with use $\varphi_{e,s}(V_s, D_s, x)$. Also, if $c < s$ is not currently a tracker for any input for $\tau$ and $\hat{\Phi}_{\tau,s}(V_s, D_s, c)\uparrow$ then we define $\hat{\Phi}_{\tau,s+1}(V_s, D_s, c) = 0$ with use $0$ (recall that since $V$ is not built by us, the use of $\hat{\Phi}$ is not the largest number queried; it is the length of the string appearing in an axiom applying to the oracle). Finally for every $x < s$ for which $\mathtt{tr}_s(\tau, x)$ is undefined, we define a new, large tracker $\mathtt{tr}_{s+1}(\tau, x)$.

Suppose that $\tau\hat{\ }\infty\hat{\ }\infty_n$ is accessible (for some $n$). For each $i < s$ let $r_i$ be the last stage at which $\tau\hat{\ }\infty\hat{\ }\infty_n\hat{\ }i$ was last accessible, $r_i = 0$ if there was no such stage. We let $\tau\hat{\ }\infty\hat{\ }\infty_n\hat{\ }i$ be next accessible for the least $i \leqslant s$ such that for all $x < r_i$, $c = \mathtt{tr}_s(\tau, x)$ is defined, $o^i_s(c) < \alpha$ and $\hat{\Phi}_{e,s}(V_s, c) = f^i_s(c)$. Note that $r_s = 0$ and so such $i$ does exist.

Suppose that a node $\pi$, working for $P_e$, is accessible at stage $s$. If $|\pi| \geqslant s$ then we end the stage. Suppose that $|\pi| < s$. We first maintain the shadow functional $\hat{\Psi}_\pi$. For every $q < s$ which is not currently an anchor for any child of $\pi$, if $\hat{\Psi}_{\pi,s}(V_s, q)\uparrow$

then we define $\hat{\Psi}_{\pi,s+1}(V_s, q) = 0$ with use 0. Now let $q = \mathtt{ac}_s(\sigma)$ be an anchor for a child $\sigma$ of $\pi$, and suppose that $\hat{\Psi}_{\pi,s}(V_s, q)\uparrow$. Let $p = \mathtt{fl}_s(\sigma)$ be the current follower of $\sigma$.

- If either $p \in D_s$, or for some $\tau \in \mathtt{prec}(\pi)$ working for $Q_d$ and some $x$ which $\sigma$ currently respects (for $\tau$) we have $p < \varphi_{d,s}(x)$, then we cancel $p$ and appoint a new, large follower $\mathtt{fl}_{s+1}(\sigma)$. We leave $\hat{\Psi}_{\pi,s+1}(V_s, q)$ undefined. In the first case we have already attacked with $p$, but now the computation against which we diagonalised has disappeared. In the second case, as described earlier, we need to protect the computation $\Phi_d(D, V, x)$ from the action of $\sigma$.
- Otherwise, if $p \in \operatorname{dom} \Psi_{e,s}(V_s)$ then we define $\hat{\Psi}_{\pi,s+1}(V_s, q) = s$ with use $\psi_{e,s}(p)$. If $p \notin \operatorname{dom} \Psi_{e,s}(V_s)$ then we leave $\hat{\Psi}_{\pi,s+1}(V_s, q)$ undefined.

For $n < s$ let $t_n$ be the last stage at which $\pi\hat{\ }\infty_n$ was accessible, $t_n = 0$ if there is no such stage. Also let $y = \operatorname{dom} \hat{\Psi}_{\pi,s}(V_s)$. If there is some $n \leqslant y$ such that $\ell_s(\pi, n) \geqslant t_n$ then we let $\pi\hat{\ }\infty_n$ be next accessible for the least such $n$. Otherwise we let $\pi\hat{\ }\mathtt{fin}_y$ be next accessible.

Suppose that $\sigma = \pi\hat{\ }\infty_n$ is accessible.

- If $\sigma$ has no anchor then we appoint a new large anchor $q = \mathtt{ac}_{s+1}(\sigma)$ and a new, large follower $p = \mathtt{fl}_{s+1}(\sigma)$. We let $m_{s+1}(\sigma) = s$.
- If $p = \mathtt{fl}_s(\sigma)$ is defined, $p \notin D_s$, $\Psi_{e,s}(V_s, p) = 0$, and $\mathtt{ac}_s(\sigma) \in \operatorname{dom} \hat{\Psi}_{\pi,s}(V_s)$ then we enumerate $p$ into $D_{s+1}$. Redefine $m_{s+1}(\sigma) = s$. For all $\tau \in \mathtt{prec}_\infty(\pi)$ and all inputs $x$ which $\sigma$ does not currently respect (for $\tau$) that is, $x \geqslant m_s(\sigma)$, cancel the tracker $\mathtt{tr}(\tau, x)$.

If either of these happen, we stop the stage and initialise all nodes weaker than $\sigma$. If the stage was not ended, then the unique child of $\sigma$ is next accessible.

*Verification.* First we note that for the functionals $\Xi$ we define, $\hat{\Psi}_\pi$ and $\hat{\Phi}_\tau$, we only define a new axiom $\Xi_{s+1}(V_s, x)$ if $x \notin \operatorname{dom} \Xi_s(V_s)$. This shows that these functionals are consistent for $V$, indeed at every stage.

We will need to show that these shadow functionals behave properly. The $\Psi$-functionals are easy.

Lemma 2.2. *Let $\pi$ be a node working for a requirement $P_e$. Let $\sigma$ be a child of $\pi$. Let $s$ be a stage and suppose that $q = \mathtt{ac}_s(\sigma)$ and $p = \mathtt{fl}_s(\sigma)$ are defined. If $\hat{\Psi}_\pi(V, q)\downarrow [s]$ then $\Psi_e(V, p)\downarrow [s]$ and $\hat{\psi}_{\pi,s}(q) = \psi_{e,s}(p)$.*

Proof. Suppose that $\hat{\Psi}_\pi(V, q)\downarrow [s]$; let $u = \hat{\psi}_{\pi,s}(q)$. Let $t < s$ be the stage at which we defined this computation. So $V_t \upharpoonright_u = V_s \upharpoonright_u$. At stage $t$ we have $\Psi_e(V, p)\downarrow [s]$ with use $u$. Hence this computation persists until stage $s$. We may assume that while $\Psi_e(V, p)\downarrow$, no new computations (with different use) are enumerated into $\Psi_e$. Thus $\psi_{e,s}(p) = u$. □

Lemma 2.3. *Suppose that a node $\pi$ working for $P_e$ is accessible infinitely often and is initialised only finitely often. There is a child $\sigma$ of $\pi$ which is accessible infinitely often. Let $\sigma$ be the strongest such child. Then:*

1. *$\sigma$ ends the stage only finitely many times.*
2. *$\sigma$ believes that $\hat{\Psi}_\pi(V)$ is total if and only if $\hat{\Psi}_\pi(V)$ is indeed total.*
3. *If $\hat{\Psi}_\pi(V)$ is total then the requirement $P_e$ is met.*

Proof. Suppose that $\hat\Psi_\pi(V)$ is not total. Then for every $n$, $\pi\hat{}\infty_n$ is accessible only finitely often (otherwise $\lim_s \ell_s(\pi, n) = \infty$ and this implies that $\hat\Psi_\pi(V)$ is total). On the other hand, because $\hat\Psi_\pi$ is defined only at stages at which $\pi$ is accessible, we know that $y = \liminf_s \operatorname{dom} \hat\Psi_\pi(V)[s]$ is finite, and $y = \operatorname{dom} \hat\Psi_\pi(V)[s]$ at infinitely many stages $s$ at which $\pi$ is accessible. Hence $\pi\hat{}\mathtt{fin}_y$ is accessible infinitely often, and is the strongest child of $\pi$ which is accessible infinitely often. This node never ends the stage.

Suppose that $\hat\Psi_\pi(V)$ is total. There is some $n$ such that $\lim_s \ell_s(\pi, n) = \infty$; let $n$ be the least such. For almost every stage $s$, $\operatorname{dom} \hat\Psi_\pi(V)[s] > n$. Hence $\pi\hat{}\infty_n$ is accessible infinitely often, and is the strongest such child of $\pi$.

At the first stage at which $\sigma = \pi\hat{}\infty_n$ is accessible after that last stage at which it is initialised we define an anchor $q = \mathtt{ac}(\sigma)$; this anchor is never cancelled. Let $t$ be the stage at which the $V$-correct computation $\hat\Psi_\pi(V, q)$ is defined (note that $\sigma$ need not be accessible at that stage). The follower $p = \mathtt{fl}_t(\sigma)$ is never cancelled. After stage $t$, the node $\sigma$ ends the stage at most once, when $p$ is enumerated into $D$.

We claim that $\Psi_e(V, p) \neq D(p)$. We have $p \notin D_t$ (for otherwise $p$ would be cancelled at stage $t$). By Lemma 2.2, $\Psi_e(V, p)\downarrow [t]$ is a $V$-correct computation. If $\Psi_e(V, p) = 0$ then at the next stage $s > t$ at which $\sigma$ is accessible, $p$ is enumerated into $D$. If $\Psi_e(V, p) = 1$ then at no stage do we enumerate $p$ into $D$. □

Lemma 2.4. *Let $\pi$ be a node which works for requirement $P_d$. Let $\tau \in \mathtt{prec}(\pi)$. Let $\sigma$ be a child of $\pi$ which guesses that $\hat\Psi_\pi(V)$ is total. Let $s$ be a stage at which $\pi$ is accessible, and let $x$ be an input for $\tau$ which $\sigma$ respects at stage $s$. Suppose that $p = \mathtt{fl}_s(\sigma)$ and $q = \mathtt{fl}_s(\sigma)$ are defined. Then $\Phi_e(V, D, x)\downarrow [s]$ and either (i) $p \in D_s$; or (ii) $\hat\Psi_\pi(V, q)\uparrow [s]$; or (iii) $\varphi_{e,s}(x) \leqslant p$.*

Proof. Suppose that $\hat\Psi_\pi(V, q)\downarrow [s]$ and that $p \notin D_s$. Let $t < s$ be the stage at which the computation $\hat\Psi_\pi(V, q)[s]$ is defined. When the anchor is chosen it is large, and it is not large at stage $t$; hence $q = \mathtt{ac}_t(\sigma)$. The follower $\mathtt{fl}_t(\sigma)$ is not enumerated into $D$ at stage $t$ since $\hat\Psi_\pi(V, q)\uparrow [t]$. The follower is not cancelled at stage $t$; otherwise $\hat\Psi_\pi(V, q)$ is not defined at stage $t$. The follower is not cancelled at any stage in the interval $(r, s)$ since $\hat\Psi_\pi(V, q)\downarrow$ at these stages. Hence $p = \mathtt{fl}_t(\sigma)$. Since $p \notin D_s$, $m_s(\sigma) < t$.

If $\tau \in \mathtt{prec}_\infty(\pi)$ then $x < m_s(\sigma)$. If $\tau \in \mathtt{prec}_{\mathtt{fin}}(\pi)$ then $x < y$ where $\tau\hat{}\infty\hat{}\mathtt{fin}_y \leqslant \pi$. Since $\pi$ is accessible at stage $m_s(\sigma)$ we have $y < m_s(\sigma)$ so again $x < m_s(\sigma)$. Hence $\Phi_e(V, D, x)\downarrow [r]$ at every stage $r > m_s(\sigma)$ at which $\tau\hat{}\infty$ is accessible. In particular this holds for $r = t$. Since $m_t(\sigma) = m_s(\sigma)$, $x$ is respected by $\sigma$ at stage $t$. If $p < \varphi_{e,t}(x)$ then since $\hat\Psi_\pi(V, q)\uparrow [t]$, $p$ would be cancelled at stage $t$. Hence $p \geqslant \varphi_{e,t}(x)$.

For brevity let $u = \varphi_{e,t}(x)$. We may assume that $\psi_{d,t}(p) \geqslant p$, and $\hat\psi_{\pi,t}(q) = \psi_{d,t}(p)$. The fact that the computation $\hat\Psi_\pi(V, q)[t]$ survives until stage $s$ implies that $V_t\restriction_u = V_s\restriction_u$. The lemma would be proved once we show that $D_t\restriction_u = D_s\restriction_u$; this would imply that the computation $\Phi_e(V, D, x)[t]$ survives until stage $s$ and so $u = \varphi_{e,s}(x) \leqslant p$ as required.

Suppose for a contradiction that at some stage $r \in [t, s)$ a number $p' < u$ is enumerated into $D_{r+1}$; let $r$ be the least such stage. So the computation $\Phi_e(V, D, x)[t]$ survives until stage $r$; $\varphi_{e,r}(x) = u$. The number $p'$ is the follower $\mathtt{fl}_r(\sigma')$ for some node $\sigma'$, a child of a node $\pi'$ working for $P_{d'}$. The node $\pi'$ must extend $\pi$: it must be weaker than $\sigma$, since it does not initialise $\sigma$ at stage $r$; and it is not initialised at stage $t$, because the follower $p'$ is large when it is chosen, and so $p'$ is chosen prior to stage $t$. The node $\sigma'$ is initialised at stage $m_s(\sigma)$. Hence $m_r(\sigma') > m_s(\sigma)$. This shows that $x$ is respected (for $\tau$) by $\sigma'$ at stage $r$ (if $\tau \in \mathtt{prec}_{\mathtt{fin}}(\pi)$ then we use the fact that both $\pi$ and $\pi'$ extend the same child of $\tau\hat{}\infty$). Applying the lemma at stage $r$, since $p' \notin D_r$ and $\hat{\Psi}_{\pi'}(V, \mathtt{ac}_r(\sigma'))\downarrow [r]$ (otherwise $p'$ is not enumerated into $D_{r+1}$), it must be that $p' \geqslant \varphi_{e,r}(x) = u$, a contradiction. $\square$

Lemma 2.5. *Let $\tau$ be a node which works for requirement $Q_e$. Let $s$ be a stage; let $x$ be an input such that $c = \mathtt{tr}_s(\tau, x)$ is defined. Suppose that $\hat{\Phi}_\tau(V, c)\downarrow [s]$. Let $u = \hat{\varphi}_{\tau,s}(c)$. Then:*

(1) *$\Phi_e(V, D, x)\downarrow [s]$ and $u = \varphi_{e,s}(x)$.*
(2) *If $D_s \restriction_u \neq D_{s+1} \restriction_u$ then the tracker $c$ is cancelled at stage $s$.*

Proof. Both parts of the lemma are proved by simultaneous induction on the stage $s$. Suppose the lemma has been verified for all stages prior to stage $s$. Assume the hypotheses of the lemma hold at stage $s$. Let $t < s$ be the stage at which the computation $\hat{\Phi}_\tau(V, c)[s]$ was defined. So $V_t \restriction_u = V_s \restriction_u$. At stage $t$ we have $\Phi_e(V, D, x)\downarrow [t]$ with use $\varphi_{e,t}(x) = u$. Because trackers are chosen large, $c = \mathtt{tr}_t(\tau, x)$.

The conditions of the lemma hold at every stage in the interval $[t, s)$. Since the tracker $c$ is not cancelled at any stage in that interval, by induction on these stages (using (2)) we see that $D_s \restriction_u = D_t \restriction_u$. This shows that the computation $\Phi_e(V, D, x)[t]$ is preserved up to stage $s$, and so establishes (1) at stage $s$.

Suppose that a number $p < u$ is enumerated into $D$ at stage $s$. Then $p = \mathtt{fl}_s(\sigma)$ for some node $\sigma$, a child of a node $\pi$. The follower $p$ must be chosen prior to stage $t$. If $\sigma$ is stronger than $\tau$ then $\tau$ is initialised at stage $s$, whence $c$ is cancelled at stage $s$. Assuming otherwise, it must be the case that $\sigma > \tau\hat{}\infty$, as $\sigma$ is not initialised at stage $t$.

Lemma 2.4 ensures that $\sigma$ does not respect $x$ (for $\tau$) at stage $s$. Suppose that $\tau\hat{}\infty\hat{}\mathtt{fin}_y \leqslant \pi$ for some $y$. Let $r$ be the last stage prior to stage $s$ at which $\tau\hat{}\infty$ was accessible. Then $r \geqslant t$. It follows that $\Phi_e(V, D, x)\downarrow [r]$ and the computation is preserved until stage $s$. Hence $y > x$. But then $\sigma$ respects $x$. So $\tau \in \mathtt{prec}_\infty(\pi)$. Then $\sigma$ is instructed to cancel $c$ at stage $s$; (2) holds. $\square$

Lemma 2.6. *Let $\tau$ be a node which works for $Q_e$. Suppose that $\tau$ is initialised only finitely often, and that $\tau\hat{}\infty$ is accessible infinitely often.*

(1) *For every $x$ we eventually appoint a tracker $\mathtt{tr}(\tau, x)$ which is never cancelled.*
(2) *There is an outcome $o \in \{\infty_n, \mathtt{fin}_n\}$ such that $\tau\hat{}\infty\hat{}o$ is accessible infinitely often.*

*Let $\rho = \tau\hat{}\infty\hat{}o$ be the strongest child of $\tau\hat{}\infty$ which is accessible infinitely often.*

(3) *If $\Phi_e(V, D)$ is total then so is $\hat{\Phi}_\tau(V)$, and $o = \infty_n$ for some $n$. Further, for some $i$, $\tau\hat{}\infty\hat{}\infty_n\hat{}i$ is accessible infinitely often.*
(4) *Otherwise $o = \mathtt{fin}_y$ where $y = \operatorname{dom} \Phi_e(D, V)$.*

Proof. Let $x < \omega$. At any stage $t > x$ at which $\tau\hat{}\infty$ is accessible, if $\mathtt{tr}_t(\tau, x)$ is undefined then we appoint a new tracker $\mathtt{tr}_{t+1}(\tau, x)$. Suppose that a tracker $\mathtt{tr}_s(\tau, x)$ is defined and is cancelled at stage $s$. The stage $s$ is ended by a child $\sigma$

of a node $\pi$ working for some $P_d$; $\tau \in \mathtt{prec}_\infty(\pi)$ and the node $\sigma$ enumerates its follower $p = \mathtt{fl}_s(\sigma)$ into $D_{s+1}$. We have $x \geqslant m_s(\sigma)$. The marker $m_s(\sigma)$ is chosen at stage $m_s(\sigma)$, at which $\sigma$ is accessible. Thus there are only finitely many nodes $\sigma$ which can ever cancel the tracker $\mathtt{tr}_s(x,\tau)$. Each such node does so at most once, since when it does, it updates $m_{s+1}(\sigma) = s > x$. This gives (1).

Suppose that $\Phi_e(V,D)$ is not total; let $y = \operatorname{dom}\Phi_e(V,D)$. Let $c$ be the eventual tracker for $y$, which is never cancelled. Then $y \notin \operatorname{dom}\hat{\Phi}_\tau(V)$. This is ensured by part (1) of Lemma 2.5; If $\hat{\Phi}_\tau(V,c)\downarrow$ with use $u$ then at a late stage at which both $V$ and $D$ are correct up to $u$ we would get a $V,D$-correct computation of $\Phi_e(y)$. Since $\hat{\Phi}_\tau(V)$ is partial, no totality outcome $\infty_n$ is guessed infinitely often. Since $\Phi_e(V,D,x)$ is eventually fixed for all $x < y$, eventually, no outcome stronger than $\mathtt{fin}_y$ is ever guessed; but $\mathtt{fin}_y$ is guessed infinitely often. This gives (4).

Suppose that $\Phi_e(V,D)$ is total. For every $y$, $\mathtt{fin}_y$ is guessed only finitely many times. We show that $\hat{\Phi}_\tau(V)$ is total. This will imply that some $\infty_n$ is guessed infinitely often. Let $c < \omega$. As usual, if $c$ is never chosen as a follower or is chosen and later cancelled, then $\hat{\Phi}_\tau(V,c)\downarrow$. Suppose that $c$ is chosen as a tracker for $x$ at stage $r$, and is never cancelled. Eventually no $\mathtt{fin}_y$ for $y \leqslant x$ is ever guessed; so eventually, at every stage $s$ at which $\tau\hat{\ }\infty$ is accessible, if $\hat{\Phi}_\tau(V,c)\uparrow [s]$ then a new computation $\hat{\Phi}_{\tau,s+1}(V_s,c)$ is defined. The use is $\varphi_{e,s}(x)$. This use stabilizes, and eventually $V$ stabilizes below that use, and so eventually a $V$-correct computation must be made.

Since $\deg_{\mathrm{T}}(V)$ is totally $\alpha$-c.a., there is some $i < \omega$ such that $\hat{\Phi}_\tau(V) = f^i$ and $\langle f^i_s, o^i_s\rangle$ is eventually $\alpha$-computable. Since every input eventually receives a permanent tracker, the outcome $i$ is guessed infinitely often for the least such $i$. □

Lemmas 2.3 and 2.6 together show that the true path is infinite and that the construction is fair to every node on the true path.

LEMMA 2.7. *Every positive requirement $P_e$ is met.*

PROOF. Let $\pi$ be the node on the true path which works for $P_e$. Suppose that $\Psi_e(V)$ is total. We show that $\hat{\Psi}_\pi(V)$ is total (and then appeal to Lemma 2.3).

Let $q < \omega$. To show that $\hat{\Psi}_\pi(V,q)\downarrow$ we may, as usual, assume that $q$ is chosen as an anchor of a child $\sigma$ of $\pi$ at some stage $r$, and is never cancelled. We show that followers for $\sigma$ are cancelled only finitely many times. This suffices: if $p$ is a follower for $\sigma$ which is never cancelled, then eventually we see the $V$-correct computation $\Psi_e(V,x)$. At any stage $s$ at which $\pi$ is accessible, if $\hat{\Psi}_\pi(V,q)\uparrow [s]$ then a new computation is defined with use $\psi_{e,s}(p)$, which eventually stabilizes.

The node $\sigma$ believes that $\hat{\Psi}_\pi(V)$ is total. Hence if $\sigma$ is accessible infinitely often then $\hat{\Psi}_\pi(V)$ is total and we are done. We assume that $\sigma$ is accessible only finitely many times. The marker $m_s(\sigma)$ is updated only when $\sigma$ is accessible, so reaches a final value $m(\sigma)$ at stage $t \geqslant r$.

Suppose that the follower $p = \mathtt{fl}_s(\sigma)$ is cancelled at a stage after stage $t$. This is done on behalf of a node $\tau \in \mathtt{prec}(\pi)$ (working for some $Q_d$) and an input $x$. There are two cases. If $\tau \in \mathtt{prec}_\infty(\pi)$ then a totality outcome for $\tau\hat{\ }\infty$ lies on the true path. This implies that $\Phi_d(V,D)$ is total. Also, $x < m(\sigma)$. If the follower for $\sigma$ is cancelled after the correct computation $\Phi_d(V,D,x)$ appears then a new follower is chosen to be large, and so is greater than $\psi_{d,s}(x)$ for all later $s$. This implies that this $\tau$ can cause only finitely many cancellations of $\mathtt{fl}_s(\sigma)$.

The other case is $\tau \in \texttt{prec}_{\texttt{fin}}(\pi)$; say $\tau\hat{\ }\infty\hat{\ }\texttt{fin}_y \preccurlyeq \pi$; so $x < y$. By Lemma 2.6, $y = \operatorname{dom} \Phi_d(V, D)$, so again $\Phi_d(V, D, x)$ eventually converges by a correct computation. The argument is now the same as in the first case. ☐

To finish the verification we show that every requirement $Q_e$ is met. Let $\tau$ be the node on the true path which works for $Q_e$, and suppose that $\Phi_e(V, D)$ is total. Then $\tau\hat{\ }\infty$ lies on the true path; and Lemma 2.6 says that for some $n$ and $i$, $\rho = \tau\hat{\ }\infty\hat{\ }\infty_n\hat{\ }i$ lies on the true path. Then $\langle f^i_s, o^i_s\rangle$ is eventually $\alpha$-computable and $\hat{\Phi}_\tau(V) = f^i$. As in previous proofs let $s^*$ be the last stage at which $\rho$ is initialised, and let $s_0 < s_1 < s_2 < \cdots$ be the stages after stage $s^*$ at which $\rho$ is accessible.

Fix $x < \omega$. We let $j(x)$ be the least $j$ such that $x < s_{j-1}$. For all $j \geqslant j(x)$, $\Phi_e(V, D, x)\downarrow [s_j]$, $c_j = c_j(x) = \texttt{tr}_{s_j}(\tau, x)$ is defined, $o^i_{s_j}(c_j) < \alpha$ and $\hat{\Phi}_\tau(V, c_j)\downarrow = f^i(c_j)\ [s_j]$. For $j \geqslant j(x)$ let $a_j = a_j(x)$ be the set of nodes $\sigma$, children of nodes $\pi$ working for some $P_d$ such that $\rho \preccurlyeq \pi$, such that $m_{s_j}(\sigma) \leqslant x$. Since $m_s(\sigma)$ is non-decreasing, if $j < j'$ then $a_{j'} \subseteq a_j$.

The following lemma is an analogue of Lemmas III.2.9 and 1.7.

LEMMA 2.8. *Let $x < \omega$ and $j \geqslant j(x)$. Let $u = \varphi_{e,s_j}(x)$.*

(1) *If $a_{j+1} = a_j$ then $c_{j+1} = c_j$;*
(2) *If $D_{s_{j+1}}\restriction_u \neq D_{s_j}\restriction_u$ then $c_{j+1} \neq c_j$;*
(3) *If $D_{s_{j+1}}\restriction_u = D_{s_j}\restriction_u$ but $V_{s_{j+1}}\restriction_u \neq V_{s_j}\restriction_u$ then $o^i_{s_{j+1}}(c_j) < o^i_{s_j}(c_j)$.*

PROOF. The instructions ensure that only a node $\sigma$ (with parent $\pi$) such that $\tau \in \texttt{prec}_\infty(\pi)$ and $m_s(\sigma) \leqslant x$ can cancel $\texttt{tr}_s(\tau, x)$. Say that a node $\pi$ with $\tau \in \texttt{prec}_\infty(\pi)$ is accessible at a stage $r \in (s_j, s_{j+1})$; then $\pi$ is initialised at stage $s_j$ and so $m_s(\sigma) > s_j > x$. So if $c_j$ is cancelled by stage $s_{j+1}$, then it is cancelled by a node $\sigma \in a_j$. But then we define $m_{s_j+1}(\sigma) = s_j > x$ and so $\sigma \notin a_{j+1}$. This gives (1).

The same argument shows that if $D_{s_{j+1}}\restriction_u \neq D_{s_j}\restriction_u$ then $D_{s_j+1}\restriction_u \neq D_{s_j}\restriction_u$. (2) is given by Lemma 2.5(2).

Suppose that $a_{j+1} = a_j$ but $V_{s_{j+1}}\restriction_u \neq V_{s_j}\restriction_u$. Let $s \geqslant s_j$ be the least stage such that $V_{s+1}\restriction_u \neq V_s\restriction_u$.

By Lemma 2.5(1), $u = \hat{\varphi}_{\tau,s_j}(c_j)$, and so $\hat{\Phi}_\tau(V, c_j)\uparrow [s+1]$. When we redefine a value for $\hat{\Phi}_\tau(V, c_j)$, it is the stage number, and so $\hat{\Phi}_\tau(V, c_j)[s_{j+1}] > s_j$. In particular $\hat{\Phi}_\tau(V, c_j)[s_{j+1}] \neq \hat{\Phi}_\tau(V, c_j)[s_j]$. But then $f^i_{s_{j+1}}(c_j) \neq f^i_{s_j}(c_j)$, and (3) follows. ☐

Now let for all $j \geqslant j(x)$

$$\gamma_j = \gamma_j(x) = \alpha \cdot |a_j| + o^i_{s_j}(c_j).$$

Since $\alpha$ is closed under addition, for all $n$ and all $\beta < \alpha$ we have $\alpha\cdot n+\beta < \alpha\cdot(n+1)$. Thus if $a_{j+1} \neq a_j$ then (as $a_{j+1} \subsetneq a_j$) $\gamma_{j+1} < \gamma_j$. Suppose that $a_{j+1} = a_j$. Then $c_{j+1} = c_j$ and so $o^i_{s_{j+1}}(c_{j+1}) = o^i_{s_{j+1}}(c_j) = o^i_{s_j}(c_j)$; so $\gamma_{j+1} \leqslant \gamma_j$. Suppose further that $\Phi_e(V, D, x)[s_j] \neq \Phi_e(V, D)[s_{j+1}]$. Since $c_{j+1} = c_j$, $D_{s_{j+1}}\restriction_u = D_{s_j}\restriction_u$. Then Lemma 2.8(3) ensures that $\gamma_{j+1} < \gamma_j$. This concludes the verification.

**2.1. Uniformity again.** Inspecting the construction we see that $|a_{j(x)}(x)| < x$. This is because $m_{s_{j(x)}}(\sigma)$ is distinct for distinct $\sigma \in a_{j(x)}(x)$ (when $m_s(\sigma)$ is set the stage ends). This shows that in fact $\deg_{\mathrm{T}}(V \oplus D)$ is uniformly totally $\alpha\cdot\omega$-c.a., as every $\Phi_e(V, D)$ is $h$-c.a. for $h(n) = \alpha \cdot (n + 1)$.

**2.2. Maximal $<\alpha$-c.a. degrees.** Suppose that $\alpha$ is a limit of powers of $\omega$, and that $\deg_{\mathrm{T}}(V)$ is totally $<\alpha$-c.a. We can modify the construction above by letting the sequence $\langle f^i \rangle$ range over all functions which are $\beta$-c.a. for some $\beta < \alpha$. Examining the proof above, we see that the ordinal bound on the number of changes of $\Phi_e(V, D, x)$ is given by a finite multiple of $o^i(c)$ for a variety of $c$ but for fixed $i$. Thus, if $f^i$ is $\beta$-c.a., then $\Phi_e(V, D)$ is $\beta \cdot \omega$-c.a. We thus obtain:

THEOREM 2.9. *If $\alpha \leqslant \epsilon_0$ is a limit of powers of $\omega$, then no c.e. degree is maximal totally $<\alpha$-c.a.*

CHAPTER V

# Presentations of left-c.e. reals

In this chapter we prove Theorem I.0.2:

(1) If a c.e. degree $\mathbf{d}$ is not totally $\omega$-c.a. then there is a left-c.e. real $\alpha \leqslant_{\mathrm{T}} \mathbf{d}$ and a c.e. set $B <_{\mathrm{T}} \alpha$ such that every presentation of $\alpha$ is $B$-computable.
(2) If a left-c.e. real $\alpha$ has a totally $\omega$-c.a. degree then there is a presentation of $\alpha$ which is Turing equivalent to $\alpha$.

A motivation for the consideration of presentations of left-c.e. reals comes from algorithmic randomness, where prefix-free c.e. sets of finite binary strings appear in the definition of prefix-free Kolmogorov complexity and help characterise Martin-Löf randomness. We recall some definitions.

DEFINITION 0.10. A real number $\alpha \in \mathbb{R}$ is *left-c.e.* if its left cut, the set of rational numbers $q < \alpha$, is c.e.

A set of strings $C \subset 2^{<\omega}$ is *prefix-free* if no two distinct strings in $C$ are comparable. The *measure* of such a set $C$, denoted by $\lambda(C)$, is the Lebesgue measure of the open subset of $2^\omega$ generated by $C$. It equals $\sum_{\sigma \in C} 2^{-|\sigma|}$.

A *presentation* of a left-c.e. real $\alpha \in [0,1]$ is a c.e. prefix-free set $C \subset 2^{<\omega}$ whose measure is $\alpha$.

A real is left-c.e. if and only if it is the limit of an increasing, computable sequence of rational numbers. If $A$ is a c.e. set then $0.A$ (the real whose binary expansion is $A$) is a left-c.e. real. Unlike c.e. sets, left-c.e. reals may be random. A well-known left-c.e. random real is Chaitin's $\Omega$, which is defined to be the measure of the domain of a universal prefix-free machine.

Since we are discussing presentations, from now all numbers we deal with are in the unit interval $[0,1]$. Implied in the definition of presentations is that the measure of a c.e. prefix-free set of strings is left-c.e. Every left-c.e. real $\alpha \in [0,1]$ has presentations. Indeed with padding it is seen that every left-c.e. real has a computable presentation. Every presentation of a left-c.e. real $\alpha$ is computable from $\alpha$. But bizarre things can happen. In [25], Downey and LaForte constructed a noncomputable left-c.e. real $\alpha$, all of whose c.e. presentations are computable. On the other hand they showed that any left-c.e. real with promptly simple degree has a noncomputable presentation. Stephan and Wu [59] showed that the same holds for all noncomputable $K$-trivial left-c.e. reals. Theorem I.0.2(2) extends their result. See also [20, 63].

*Computing with real numbers.* In this chapter we view real numbers as elements both of the computable metric space $[0,1]$ and as infinite binary sequences in $2^{\mathbb{N}^+}$ (where $\mathbb{N}^+ = \{1,2,3,\dots\}$) by using their binary expansion: $\alpha \in [0,1]$ equals $\sum_{k\in\mathbb{N}^+} 2^{-k}\alpha(k)$. Thus reals in the unit interval are essentially elements of Cantor space. As the former, a real number $\alpha \in [0,1]$ is given (in the language of

Weihrauch, *represented*) by a nested sequence $\langle I_n\rangle$ of closed intervals with dyadic rational endpoints (with the length of $I_n$ being $2^{-n}$) such that $\{\alpha\} = \bigcap_n I_n$.

All numbers in $[0,1]$ have a Turing degree (rather than merely a continuous degree, see [49]) which is the same as the Turing degree of their binary expansion. However passing from a real to its expansion is nonuniform. For this reason there are elements of the computable metric space $[0,1]^\omega$ which do not have Turing degree. The source of the problem are dyadic rational numbers, which have more than one binary expansion and have more than one presentation $\langle I_n\rangle$. We will be working with noncomputable numbers $\alpha$, and so in particular with numbers that are not dyadic rationals, so this problem is avoided.

We remark though that during constructions we may work with dyadic rational numbers $q \in [0,1)$ and their binary expansion. By that we mean the expansion which ends with a string of zeros.

### 1. Presentations of c.e. reals and non-total $\omega$-c.a. permitting

In this section we prove part (1) of I.0.2.

**1.1. A simplified construction.** Before adding permitting we construct a left-c.e. real $\alpha$ and a c.e. set $B$ such that $B <_{\mathrm{T}} \alpha$ but every presentation of $\alpha$ is $B$-computable. As mentioned above, this has been done in [25] with $B = \varnothing$. We present the construction proving the weaker statement because it is simpler than the original one. The simplification is compatible with non-total $\omega$-c.a. permitting. The original construction is in some sense compatible with non-total $< \omega^\omega$-c.a. permitting. We discuss this later, in Subsection 1.3.

Let $\langle \Psi_e\rangle$ be an enumeration of functionals which output reals in the interval $[0,1]$. So for each $k$ (and oracle $X$), $\Psi_e(X,k)$ (if it converges) is a closed interval $I$ (with dyadic rational endpoints) of length $2^{-k}$; if $\Psi_e(X)$ is total then $\langle \Psi_e(X,k)\rangle$ is a representation of $\Psi_e(X)$. We need to meet the requirements:

$P_e$**:** $\Psi_e(B) \neq \alpha$.

Let $\langle C_e\rangle$ be an enumeration of all prefix-free c.e. sets of binary strings. We need to meet the requirements

$N_e$**:** If $\lambda(C_e) = \alpha$ then $C_e \leqslant_{\mathrm{T}} B$.

Globally we also need to ensure that $B \leqslant_{\mathrm{T}} \alpha$.

*Discussion.* Consider first a requirement $N_e$. It monitors the quantities $\alpha_s$ and $\lambda(C_{e,s})$. We note that we may assume that for all $s < \omega$, $\lambda(C_{e,s}) < \alpha_s$. For we will ensure that $\alpha$ is not a dyadic rational. When we see that enumerating a string $\sigma$ into $C_{e,s}$ will make $\lambda(C_{e,s}) \geqslant \alpha_s$ we prevent the enumeration and wait until $\alpha_t$ grows beyond $\lambda(C_{e,s-1} \cup \{\sigma\})$, and only then enumerate $\sigma$ into $C_e$. If $\lambda(C_e) = \alpha$ then such a stage will occur.

At some stage $s$ we discover that $\alpha_s - \lambda(C_{e,s}) < 2^{-t}$ for some $t < \omega$. At that stage, the requirement would like to certify that strings of length smaller than $t$ would never enter $C_e$ again. It does so with some $B$-use $\eta_e(t)$. To ensure that strings of length smaller than $t$ cannot enter $C_e$, the requirement imposes restraint on weaker actors: until further notice we require that $\alpha_u - \alpha_s \leqslant 2^{-t}$. The restraint will be lifted at a later stage at which we again see that $\alpha_u - \lambda(C_{e,u}) < 2^{-t}$ (an "$e$-expansionary" stage). Alternatively, a weaker requirement that really wants to violate the restraint may enumerate the use $\eta_e(t)$ into $B$. We need to ensure that this does not happen to almost all $t$.

Now consider $P_e$. Originally, some restraint is imposed on it: don't increase $\alpha$ by more than $\epsilon$ say. It chooses a "follower" $k$, large relative to $-\log\epsilon$ and waits until it sees $\Psi_e(B,k)$ converge and give us a closed interval $I$ of length $2^{-k}$ which must contain $\Psi_e(B)$ (if total) — provided that $B$ does not change below some use $b = \psi_{e,s}(k)$. While $\alpha_s$ lies far away from $I$ there is nothing to do, but once we see that $\alpha_s \in I$ the requirement wants to add a quantity of more than $2^{-k}$ to $\alpha$ to ensure that $\alpha$ lies to the right of $I$. To ensure its action is useful the requirement imposes $B$-restraint $b$. Actually we need to worry not only when we see that $\alpha_s \in I$, but when the distance between $\alpha_s$ to $I$ is smaller than some small fixed bound; if the non-computability requirements are not met then it is actually possible that $\alpha_s$ converges to the left endpoint of $I$.

Of course time passes between the stage at which $k$ was determined and the stage $r$ at which we see the convergence of $\Psi_e(B,k)$. In the meantime, a requirement $N_d$ stronger than $P_e$ imposes stricter and stricter restraints on $\alpha$. If $t < r$ then the marker $\eta_\tau(t)$ is likely smaller than the use $b$ of the computation $\Psi_e(B,k)[r]$, so enumerating the marker into $B$ by $P_e$ is self-defeating. However, for $t \geqslant r$ the markers are appointed later than $r$ and so are larger than $b$, so $P_e$ has no compunction about enumerating them into $B$. With this in mind, the requirement $P_e$ goes through a cycle of length $2^{r-k}+1$ using a "drip-feed" strategy. It increases $\alpha$ by $2^{-r}$, and then waits for the next $N_d$-expansionary stage $s > r$. At that stage it enumerates markers $\eta_d(t)$ into $B$ for $t \geqslant r$; the fact that $s$ is a new expansionary stage means that $P_e$ is now allowed to add a new quantity of $2^{-r}$ to $\alpha$. If this repeats $2^{r-k}+1$ many times we will have succeeded to add the required amount (more than $2^{-k}$) to $\alpha$ and drive $\alpha$ to the right of the interval $I$ and so meet $P_e$. This strategy has been likened to a cautious investor, slowly realising gains by repeatedly selling small amounts of stock, ensuring that the market does not notice their actions: they only sell a further amount once the stock price recovered to the original value.

Of course to be successful, the requirement $P_e$ needs to know if sufficiently many expansionary stages will occur. It guesses the answer to this question, and so as usual the construction is performed on a tree of strategies.

*The tree of strategies.* The requirements $P_e$ and $N_e$ are ordered in order-type $\omega$; the $k^{\text{th}}$ level of the tree is devoted to meeting the $k^{\text{th}}$ requirement. If $\sigma$ is a node which works for $P_e$, then $\sigma$ has only one outcome. If $\tau$ is a node which works for $N_e$, then the outcomes of $\tau$ are $\infty < \texttt{fin}$.

A node $\sigma$ working for $P_e$ may define first a follower $k_{\sigma,s}$ and then an interval $I_{\sigma,s}$ which it would like $\alpha$ to avoid. It also defines $r_{\sigma,s}$, the amount by which it is allowed to increase $\alpha$ at a single step. When $\sigma$ is initialised, the follower $k_\sigma$, the interval $I_\sigma$ and restraint bound $r_\sigma$ are cancelled. They will be cancelled only when $\sigma$ is initialised.

Nodes $\tau$ working for $N_e$ define markers $\eta_\tau(t)$. We note that it is not necessarily the case that the set of $t$ for which $\eta_\tau(t)$ is defined is an initial segment of $\omega$. In fact $\eta_\tau(t)$ may be defined at most once (at a stage greater than $t$), and $t$ will be a stage at which $\tau$ is accessible. For this reason $\eta_\tau(t)$ is not indexed by the stage number $s$.

*Construction.* At stage $s$ we define the path of accessible nodes $\delta_s$ to be an initial segment of the tree of strategies, and at the end of the stage define $\alpha_{s+1}$.

We start with $\alpha_0 = 0$.

Suppose that a node $\tau$, working for $N_e$, is accessible at stage $s$. Let $t < s$ be the previous stage at which $\tau\hat{}\infty$ was accessible; $t = 0$ if there was no such stage. If $\alpha_s - \lambda(C_{e,s}) < 2^{-t}$ we let $\tau\hat{}\infty$ be next accessible and choose $\eta_\tau(t)$ to be large. Otherwise we let $\tau\hat{}\mathtt{fin}$ be next accessible.

Suppose that a node $\sigma$, working for $P_e$, is accessible at stage $s$. The node may either let its only immediate successor on the tree of strategies be next accessible or decide to end the stage. In the latter case all nodes weaker than $\sigma$ are initialised.

First, suppose that a follower $k_{\sigma,s}$ is not defined. Define $k_{\sigma,s+1}$ to be large; let $\alpha_{s+1} = \alpha_s$ and end the stage.

Next, suppose that $k_{\sigma,s}$ is defined but an interval $I_{\sigma,s}$ is not defined. If $\Psi_e(B, k_\sigma)\downarrow [s] = I$ (recall that $I$ is a dyadic rational interval of length $2^{-k_{\sigma,s}}$) then we let $I_{\sigma,s+1} = I$ and $r_{\sigma,s+1} = s$. Let $\alpha_{s+1} = \alpha_s$ and end the stage.

If $\Psi_e(B, k_\sigma)\uparrow [s]$ then $\sigma$ does not end the stage (and as we said, the unique immediate successor of $\sigma$ is next accessible).

Suppose that $I_{\sigma,s}$ is defined. If $d(\alpha_s, I_{\sigma,s}) < 2^{-r_{\sigma,s}}$ then for all $\tau$ working for some $N_{e'}$ such that $\tau\hat{}\infty \leqslant \sigma$, for all $t \geqslant r_{\sigma,s}$ such that $\eta_\tau(t)$ is defined, enumerate $\eta_\tau(t)$ into $B_{s+1}$. Let $\alpha_{s+1} = \alpha_s + 2^{-r_{\sigma,s}}$ and end the stage.

If the distance $d(\alpha_s, I_{\sigma,s})$ is at least $2^{-r_{\sigma,s}}$ we do not end the stage.

*Verification.* The global requirement is satisfied:

LEMMA 1.1. $B \leqslant_{\mathrm{T}} \alpha$.

PROOF. Suppose that $x$ enters $B$ at stage $s$. Then $x = \eta_\tau(t)$ for some $t$ and $\tau$, and $\alpha_{s+1} = \alpha_s + 2^{-r}$ where $t \geqslant r$. Since $\eta_\tau(t) > t$, we see that once $\alpha - \alpha_s < 2^{-r}$, no numbers below $r$ can enter $B$. ☐

We observe that the construction is fair and that the true path $\delta_\omega$ is infinite. This follows by induction on the length of nodes, using the following lemma.

LEMMA 1.2. *Suppose that a node $\sigma$, working for a positive requirement, is accessible infinitely often and is initialised only finitely often. Then $\sigma$ ends the stage only finitely often.*

PROOF. Let $t$ be the last stage at which $\sigma$ is initialised. At the next stage after $t$ at which $\sigma$ is accessible we appoint a new follower $k_\sigma$ which is never cancelled. If there is no later stage at which an interval $I_\sigma$ is defined then $\sigma$ never stops the stage again.

Otherwise, an interval $I_\sigma$ is defined at some stage $r_\sigma$; the interval (and the bound $r_\sigma$) are never cancelled again. If $\sigma$ is accessible at stage $s > r_\sigma$ then $\sigma$ ends stage $s$ only if $d(\alpha_s, I_\sigma) < 2^{-r_\sigma}$, in which case it adds $2^{-r_\sigma}$ to $\alpha_s$. Since the length of the interval $I_\sigma$ is $2^{-k_\sigma}$, this happens at most $2^{r_\sigma - k_\sigma} + 2$ many times. ☐

To bound the value of $\alpha$, for a positive node $\sigma$ (one working for some $P_e$) and a stage $t$ let

$$\beta(\sigma, t) = \sum (\alpha_{s+1} - \alpha_s) \quad [\![ s \geqslant t \quad \& \quad \sigma \text{ ends stage } s ]\!].$$

So $\alpha - \alpha_t$ is the sum of $\beta(\sigma, t)$ for all positive nodes $\sigma$.

LEMMA 1.3. *Suppose that a positive node $\sigma$ is initialised at stage $t$. Then $\beta(\sigma, t) < 2^{-(3t+1)}$.*

PROOF. Suppose that $\sigma$ is initialised at stage $t$, that $u > t$ and $\sigma$ is not initialised at any stage in the interval $(t, u]$. Let $k_\sigma$ be the value of the follower for $\sigma$ in the interval $[t, u]$ (if appointed). Since $k_\sigma$ is chosen large relative to $t$ we assume that $k_\sigma > 3t + 3$; and $r_\sigma > k_\sigma$. The proof of Lemma 1.2 shows that the sum

$$\sum(\alpha_{s+1} - \alpha_s) \quad [\![ s \in [t, u] \quad \& \quad \sigma \text{ ends stage } s ]\!]$$

is bounded by $2^{-k_\sigma} + 2 \cdot 2^{-r_\sigma}$ which is bounded by $2^{-(3t+2)}$. We now sum over all the stages $t' \geqslant t$ at which $\sigma$ is initialised. ☐

We conclude that $\alpha = \lim_s \alpha_s$ exists and lies in the unit interval.

LEMMA 1.4. $\alpha < 1$.

PROOF. Every node of length $s$ is initialised at every stage $s' \leqslant s$. Thus for such a node $\sigma$ we have $\beta(\sigma, 0) = \beta(\sigma, s) < 2^{-(3s+1)}$. There are at most $2^s$ many nodes of length $s$ as the tree of strategies is binary branching. Hence level $s$ contributes at most $2^{-(s+1)}$ to $\alpha$. Some levels consists of negative nodes and so contribute nothing to $\alpha$. ☐

We turn to showing that all requirements are met.

LEMMA 1.5. *Each positive requirement $P_e$ is met.*

PROOF. Let $\sigma$ be a node on the true path which works for $P_e$. Let $k_\sigma$ be the value of the last follower chosen by $\sigma$, the one which is never cancelled. We suppose that $\Psi_e(B)$ is total; so $I_\sigma$ is eventually defined at a stage $r_\sigma > k_\sigma$. Since $\sigma$ acts only finitely often, for almost all stages $s$, $d(\alpha_s, I_\sigma) \geqslant 2^{-r_\sigma}$. Hence $d(\alpha, I_\sigma) \geqslant 2^{-r_\sigma}$ and so $\alpha \notin I_\sigma$.

It remains to show that $\Psi_e(B) \in I_\sigma$, which would follow once we show that the computation $\Psi_e(B, k_\sigma)[r_\sigma]$ is $B$-correct. Let $u = \psi_e(B, k_\sigma)[r_\sigma]$ be the use of this computation.

Suppose that a number $x$ enters $B$ at stage $s \geqslant r_\sigma$, enumerated by a node $\rho$. We show that $x > u$. The number $x$ equals $\eta_\tau(t)$ for some $\tau\hat{}\infty \leqslant \rho$ and some $t$. We know that $x = \eta_\tau(t) > t \geqslant r_{\rho,s}$. The node $\rho$ cannot be stronger than $\sigma$, for otherwise $\sigma$ is initialised at stage $s \geqslant r_\sigma$, contradicting the permanence of $k_\sigma$ and $I_\sigma$. Hence $r_{\rho,s} \geqslant r_\sigma$: this is clear if $\rho = \sigma$; otherwise, $\rho$ is initialised at stage $r_\sigma$, $s > r_\sigma$ and $r_{\rho,s}$ must be greater than $r_\sigma$. Finally the use $u = \psi_e(B, k)[r_\sigma]$ is bounded by $r_\sigma$. ☐

Toward showing that negative requirements are met, let $\tau$ be a node, working for $N_e$, and suppose that $\tau\hat{}\infty$ lies on the true path. Let $t^*$ be the last stage at which $\tau$ is initialised. We let $S$ be the set of stages $t > t^*$ at which $\tau\hat{}\infty$ is accessible. For $t \in S$ let $t^+$ be the next stage in $S$.

The markers defined by $\tau$ are $\eta_\tau(t)$ for $t \in S$. The marker $\eta_\tau(t)$ is defined at stage $t^+$.

LEMMA 1.6. *Let $u < t$ be two stages in $S$. Assume that $\eta_\tau(u) \notin B_{t+1}$. Then $\alpha_{t^+} - \alpha_t \leqslant 2^{-u}$. It follows that no strings of length less than $u$ lie in $C_{e,t^+} \backslash C_{e,t}$.*

PROOF. We consider various contributions. All nodes that lie to the right of $\tau\hat{\ }\infty$ are initialised at stage $t$. The calculation in the proof of Lemma 1.4 shows that $\alpha_{t^+} - \alpha_{t+1} \leqslant 2^{-t} \leqslant 2^{-(u+1)}$.

Next consider nodes $\sigma \geqslant \tau\hat{\ }\infty$. In the interval of stages $[t, t^+)$, such nodes are only accessible at stage $t$. At stage $t$ at most one such node $\sigma$ increases $\alpha$; the amount of increase $\alpha_{t+1} - \alpha_t$ equals $2^{-r_{\sigma,t}}$. Since $\eta_\tau(u)$ is not enumerated into $B$ at stage $t$ we have $r_{\sigma,t} > u$, and so $\alpha_{t+1} - \alpha_t \leqslant 2^{-(u+1)}$.

As discussed above, the last sentence follows: $\alpha_t - \lambda(C_{e,t}) < 2^{-t}$ and $\lambda(C_{e,t^+}) \leqslant \alpha_{t^+}$ and so $\lambda(C_{e,t^+}) - \lambda(C_{e,t}) < 2^{-u} + 2^{-t} < 2^{-(u-1)}$. □

The verification ends with:

LEMMA 1.7. *Each negative requirement $N_e$ is met.*

PROOF. We assume that $\lambda(C_e) = \alpha$; we need to show that $C_e \leqslant_{\mathrm{T}} B$. Let $\tau$ on the true path working for $N_e$. The assumption implies that $\tau\hat{\ }\infty$ lies on the true path.

We claim that infinitely many markers $\eta_\tau(u)$ are not enumerated into $B$. Let $w > t^*$ be a stage. Let $\sigma$ be the strongest extension of $\tau\hat{\ }\infty$ which acts (ends the stage) after stage $w$. Since infinitely many nodes on the true path act, $\sigma$ cannot lie to the right of the true path. It follows that $\sigma$ acts only finitely often. Let $t$ be the last stage at which $\sigma$ acts. The marker $\eta_\tau(t)$ is appointed at stage $t^+$. Let $\rho \geqslant \tau\hat{\ }\infty$ be a node which enumerates a marker $\eta_\tau(v)$ into $B$ at some stage $s \geqslant t^+$. The node $\rho$ is initialised at stage $t$; after stage $t$ it is first accessible not before stage $t^+$, and so $v \geqslant r_{\rho,s} \geqslant t^{++}$. Hence $\eta_\tau(t)$ (and in fact $\eta_\tau(t^+)$ as well) are never enumerated into $B$.

Now Lemma 1.6 shows that the following algorithm with oracle $B$ correctly computes $C_e$: Given $k < \omega$, find a stage $t > k$ in $S$ such that $\eta_\tau(t) \notin B$. Announce that $C_e\restriction_{2^{<t}} = C_{e,t^+}\restriction_{2^{<t}}$. □

**1.2. Non totally $\omega$-c.a. permitting.** We now add non-totally $\omega$-c.a. permitting to prove part (1) of Theorem I.0.2: if $\mathbf{d}$ is not totally $\omega$-c.a. then there is a left-c.e. real $\alpha \leqslant_{\mathrm{T}} \mathbf{d}$ and a c.e. set $B <_{\mathrm{T}} \alpha$ such that every presentation of $\alpha$ is $B$-computable.

Fix some function $g \in \mathbf{d}$ which is not $\omega$-c.a. Since $\mathbf{d}$ is c.e., we can replace $g$ by its modulus (see the proof of Theorem III.5.2). So we have a computable approximation $\langle g_s \rangle$ of $g$ such that:

- if $s < t$ then $g_s(n) \leqslant g_t(n)$ for all $n$;
- if $g_{s+1}(n) \neq g_s(n)$ then $g_{s+1}(m) \neq g_s(m)$ for all $m > n$.

At first approximation, the idea for reducing $\alpha$ to $g$ (and hence to $\mathbf{d}$) is to declare that if $g_s(k) = g(k)$ then $\alpha - \alpha_s \leqslant 2^{-k}$. Using the notation of the construction above, when a node $\sigma$ is visited and wants to increase $\alpha$ we must first wait for a change in $g(k_\sigma)$. The number of permissions needed to meet $\sigma$'s requirement is bounded by $2^{r_\sigma}$. We note that it is the follower $k_\sigma$ that needs to be permitted, even though at each step we increase $\alpha$ by $2^{-r_\sigma}$, not $2^{-k_\sigma}$. It is the eventual increase in $\alpha$ which counts, because the promise is that if $k$ is not permitted then $\alpha - \alpha_s \leqslant 2^{-k}$.

Of course it is possible that the number of permissions will be insufficient. While waiting for permissions the node $\sigma$ must appoint more followers $k$, with the expectation that at least one of them will receive the necessary number of permissions. If the follower $k$ does not receive enough permissions then we can approximate $g(k)$ with fewer than $2^{r_\sigma}$ many mind-changes. If no follower receives enough permissions then infinitely many of them will be appointed. This will give an $\omega$-computable approximation of $g$.

The remaining issues are the timing of permissions and necessary cancellation of followers. The follower $k$ could be permitted at a stage $s$ at which $\sigma$ is not accessible. We cannot "leave the permission open" and wait to increase $\alpha$ at the next stage at which $\sigma$ is accessible, since we do not know whether such a stage will occur. We need to act on permissions immediately.

When a follower $k$ receives a permission we increase $\alpha$ by the associated amount $2^{-r_\sigma(k)}$ (determined by the stage $r_\sigma(k)$ at which we see the computation $\Psi_e(B,k)$ converge) and we need to enumerate markers $\eta_\tau(t)$ for $t \geqslant r_\sigma(k)$ into $B$. This means that the computations $\Psi_e(B,k')$ for followers $k' > k$ for the same node are destroyed. We cannot keep these followers: overall we want action for some follower $k$ to not increase $\alpha$ by more than $2^{-k+1}$ say. So the larger followers $k'$ are cancelled, and later, larger followers may be appointed.

But this creates a problem when arguing that eventually some follower will be permitted. Suppose that a follower $k$ is eventually cancelled. When approximating $g(k)$ we do not know in advance that it will be cancelled, so we promise that our guesses for $g(k)$ will not change more than $2^{r_\sigma(k)}$ many times. We observe many changes, and then $k$ is cancelled. Henceforth changes in $g(k)$ do not seem to help us to meet $\sigma$'s requirement, which means that there is no mechanism which will bound these changes. We need to ensure that every change in $g(k)$ is useful.

The solution (as in [17]) is to allow stronger followers take over the responsibility for approximating greater portions of $g$. When a follower $k$ is permitted, larger followers $k' > k$ are cancelled. We declare that from now on, what would have been permissions for $k'$ must count as permissions for $k$. Technically we define moveable markers $a_{k,s}$, and we declare that $k$ is permitted if $g(a_k)$ changes (rather than $g(k)$). When $k$ is permitted then we raise $a_{k,s}$ to be greater than the previous values of $a_{k'}$ for the followers $k'$ which were cancelled.

*Construction.* The tree of strategies is the same as in the construction above. Positive nodes $\sigma$ appoint followers. All followers are cancelled when $\sigma$ is initialised or when smaller followers for $\sigma$ receive attention; otherwise they are retained. For all followers $k$ of $\sigma$ (except possibly for the largest one) we also define associated intervals $I_\sigma(k)$ (of length $2^{-k}$) and bounds $r_\sigma(k)$ as above. Any number can be chosen at most once as a follower for any requirement.

Negative nodes $\tau$ define markers $\eta_\tau(t)$ as in the previous construction. Globally we define location markers $a_{k,s}$ for all $k < s$, useful for reducing $\alpha$ to $g$.

We start with setting $\alpha_0 = 0$. At stage $s$ we either act on permissions or define the path of accessible nodes $\delta_s$ and act for nodes on that path.

We say that a node $\sigma$ is *already met* by stage $s$ if at stage $s$ there is some follower $k$ for $\sigma$ such that $I_\sigma(k)$ is defined and $\alpha_s$ lies strictly to the right of $I_\sigma(k)$.

Option A: Acting on Permissions. We say that a follower $k$ (for a positive node $\sigma$) *requires attention* at stage $s$ if:

- The node $\sigma$ is not already met at stage $s$;

- The interval $I_\sigma(k)$ is defined;
- $d(\alpha_s, I_\sigma(k)) < 2^{-r_\sigma(k)}$;
- the follower $k$ did not receive attention since the last stage at which $\sigma$ was accessible; and
- $g_{s+1}(a_{k,s}) \neq g_s(a_{k,s})$.

If no follower requires attention then we take option B. Otherwise let $k$ be the strongest follower which requires attention: the node $\sigma$ is the strongest, any of whose followers requires attention at stage $s$; and $k$ is the strongest (smallest) follower for $\sigma$ that requires attention at stage $s$. We say that the follower $k$ *receives attention.*

We execute the following instructions. Let $\alpha_{s+1} = \alpha_s + 2^{-r_\sigma(k)}$. For all negative nodes $\tau$ such that $\tau\hat{\ }\infty \leqslant \sigma$, for all $t \geqslant r_\sigma(k)$ such that $\eta_\tau(t)$ is defined, enumerate $\eta_\tau(t)$ into $B_{s+1}$. Initialise all nodes weaker than $\sigma$; cancel all followers for $\sigma$ greater than $k$ and their associated intervals. Redefine $a_{m,s+1}$ to be large for all $m \geqslant k$, and define a new marker $a_{s,s+1}$ to be large as well. End the stage.

Option B: Building the path of accessible nodes.

If option A was not taken then we define the path $\delta_s$ of accessible nodes. Since no permissions were used, we set $\alpha_{s+1} = \alpha_s$ and $a_{m,s+1} = a_{m,s}$ for all $m < s$; we define $a_{s,s+1}$ to be large.

Suppose that a node $\tau$ working for $N_e$ is accessible at stage $s$. Let $t < s$ be the previous stage at which $\tau\hat{\ }\infty$ was accessible; $t = 0$ if there was no such stage. If $\alpha_s - \lambda(C_{e,s}) < 2^{-t}$ we let $\tau\hat{\ }\infty$ be next accessible and choose $\eta_\tau(t)$ to be large. Otherwise we let $\tau\hat{\ }\texttt{fin}$ be next accessible.

Suppose that a node $\sigma$ working for $P_e$ is accessible at stage $s$. The node may either let its only immediate successor on the tree of strategies be next accessible or decide to end the stage. In the latter case all nodes weaker than $\sigma$ are initialised. If the node $\sigma$ is already met by stage $s$ then $\sigma$ takes no action and does not end the stage.

Suppose that $\sigma$ is not already met. If $\sigma$ has no followers then a new, large one is appointed, and the stage is ended. Otherwise, let $k$ be the largest follower for $\sigma$.

If $I_\sigma(k)$ is defined and $d(\alpha_s, I_\sigma(k)) < 2^{-r_\sigma(k)}$ then appoint a new, large follower for $\sigma$ and end the stage. If $d(\alpha_s, I_\sigma(k)) \geqslant 2^{-r_\sigma(k)}$ then the stage is not ended.

Suppose that $I_\sigma(k)$ is not defined. If $\Psi_e(B,k)\downarrow [s]$ then set $I_\sigma(k) = \Psi_e(B,k)[s]$ and $r_\sigma(k) = s$; end the stage. If $\Psi_e(B,k)\uparrow [s]$ then no action is taken and the stage is not ended.

*Verification.* Suppose that a positive node $\sigma$ is initialised only finitely many times. Every follower for $\sigma$ is either eventually cancelled, or receives attention only finitely many times. As above the point is that the follower $k$ cannot receive attention more than $2^{r_\sigma(k)-k} + 1$ many times, as each time $\alpha$ is increased by $2^{-r_\sigma(k)}$. Indeed if a follower $k$ receives attention the full number of times then the requirement is declared met and no follower for $\sigma$ receives attention, at least until a later stage at which $\sigma$ is cancelled.

Since new followers are always chosen large we see that as promised, each $k$ is chosen at most once to be a follower (for any node). A location marker $a_{m,s}$ is moved only when some follower $k \leqslant m$ receives attention. We conclude that the location markers $a_{m,s}$ reach limits $a_m$. Thus, for all $m < \omega$ there is some stage $s$ at which $g_s(a_{m,s}) = g(a_{m,s})$. The following lemma then shows that $\alpha$ is computable from $g$, and so from $\mathbf{d}$.

LEMMA 1.8. *Suppose that $g_s(a_{m,s}) = g(a_{m,s})$. Then $\alpha - \alpha_s \leqslant 2^{-(m-1)}$.*

PROOF. Note that $a_{m,t+1} \neq a_{m,t}$ only if $g_{t+1}(a_{m,t}) \neq g_t(a_{m,t})$. Hence $a_{m,s} = a_m$ is the final value of this marker. Let $\beta(k)$ be the sum of $\alpha_{t+1} - \alpha_t$, as $t$ ranges over the stages at which the follower $k$ receives attention. As discussed above, $\beta(k)$ is bounded by $2^{-k} + 2^{-r_\sigma(k)} \leqslant 2 \cdot 2^{-k}$ (where $\sigma$ is the node for which $k$ is a follower), since $r_\sigma(k) > k$. Since no follower of size less than or equal to $m$ receives attention after stage $s$ we know that

$$\alpha - \alpha_s \leqslant \sum_{k>m} \beta(k) \leqslant 2 \cdot 2^{-m}. \qquad \square$$

The proof that $B \leqslant_{\mathrm{T}} \alpha$ is identical to the one given earlier. The proof that $\alpha < 1$ requires minor modifications but is essentially the same. If $\sigma$ is a positive node which is initialised at stage $t$ then the total contribution to $\alpha - \alpha_t$ due to stages at which followers for $\sigma$ receive attention is bounded by $2\sum 2^{-k}$ where the sum ranges over follower $k$ for $\sigma$ appointed after stage $t$. Since all of these followers are chosen to be large we may assume that this sum is bounded by $2^{-3t-1}$ as above.

The following lemma ensures that the true path is infinite and that the construction is fair to nodes on the true path. First note that there are infinitely many stages at which option B is taken: if $s$ is the last stage at which option $B$ is taken, then only finitely many followers are ever appointed and each one receives attention at most once after stage $s$.

LEMMA 1.9. *Suppose that $\sigma$ is a node which works for requirement $P_e$, is only initialised finitely many times and is accessible infinitely often. Then the unique immediate successor of $\sigma$ on the tree of strategies is initialised only finitely often and so is accessible infinitely often. Further, the requirement $P_e$ is met.*

PROOF. Let $t^*$ be the last stage at which $\sigma$ is initialised.

Let $s^* > t^*$ and let $k$ be a follower for $\sigma$ at stage $s^*$ which is never cancelled. No follower stronger than $k$ receives attention after stage $s^*$.

If the interval $I_\sigma(k)$ is never defined then no larger followers for $\sigma$ are ever appointed and $\sigma$ never later ends a stage at which it is accessible. Since all followers receive attention only finitely many times we see that the successor of $\sigma$ is initialised only finitely many times. Further, in this case $\Psi_e(B, k)\uparrow$ and so the requirement $P_e$ is met.

Suppose then that at some stage $r_\sigma$ the interval $I_\sigma(k)$ is defined. The argument in the previous construction shows that the computation $\Psi_e(B, k)[r_\sigma(k)]$ is $B$-correct and so if total, $\Psi_e(B) \in I_\sigma(k)$.

If at all stages $s \geqslant r_\sigma(k)$, $\alpha_s$ lies to the left of $I_\sigma(k)$ and $d(\alpha_s, I_\sigma(k)) \geqslant 2^{-r_\sigma(k)}$ then no follower greater than $k$ is ever appointed for $\sigma$, so again the successor of $\sigma$ is on the true path and the construction is fair to that successor. As before, in this case $d(\alpha, I_\sigma(k)) \geqslant 2^{-r_\sigma(k)}$ so $\alpha \neq \Psi_e(B)$.

Similarly, if at some stage $s \geqslant r_\sigma(k)$ we see that $\alpha_s$ lies strictly to the right of $I_\sigma(k)$ then $\sigma$ is declared met and no action is taken for $\sigma$ after stage $s$. Since $\alpha \geqslant \alpha_s$ again we see that $\alpha \notin I_\sigma(k)$ and so $P_e$ is met.

Further, in this last case we do not need to assume in advance that $k$ is never cancelled: once we see $\alpha_s$ lying to the right of $I_\sigma(k)$, all action for $\sigma$ ceases and no follower is cancelled.

We claim that there is some follower $k$ for $\sigma$ which is never cancelled and for which one of the cases described above holds. Assume, for a contradiction that this is not the case. We show that $g$ is $\omega$-c.a.

The assumption means that:

- For every follower $k$ for $\sigma$ appointed after stage $s$, either $k$ is cancelled or $I_\sigma(k)$ is eventually defined and for all but finitely many stages $s \geqslant r_\sigma(k)$.
- The node $\sigma$ is never declared met after stage $t^*$.

For a follower $k$ of $\sigma$, if there is such a stage, we let $s_\sigma(k)$ be the least stage $s \geqslant r_\sigma(k)$ such that $d(\alpha_s, I_\sigma(k)) < 2^{-r_\sigma(k)}$ and $\sigma$ is accessible at stage $s$. As observed above, if $k$ is a follower for $\sigma$ at a stage $s$ and is not the largest follower for $\sigma$ at that stage, then $s > s_\sigma(k)$.

Let $x < \omega$. Let $S(x)$ be the set of stages $s > t^*$ satisfying:

- $\sigma$ is accessible at stage $s$; and
- there is some follower $k$ of $\sigma$ at stage $s$ such that $s \geqslant s_\sigma(k)$ and $x \leqslant a_{k,s}$.

For $s \in S(x)$ let $k_s(x)$ be the smallest follower for $\sigma$ witnessing that $s \in S(x)$. We first claim that if $s \in S(x)$, $t > s$ and $\sigma$ is accessible at stage $t$ then $t \in S(x)$ and $k_t(x) \leqslant k_s(x)$. Let $k = k_s(x)$. If $k$ is still a follower for $\sigma$ at stage $t$ then $k$ witnesses that $t \in S(x)$, because $a_{k,s} \leqslant a_{k,t}$. Otherwise a follower stronger than $k$ receives attention at a stage between stages $s$ and $t$. Let $m$ be the strongest such follower. Then $m$ is still a follower for $\sigma$ at stage $t$. If $m$ receives attention at stage $u \in (s,t)$ then we define $a_{m,u+1}$ to be large, in particular greater than $x$, and so $x < a_{m,t}$ and $m$ witnesses that $t \in S(x)$.

Suppose that $s < t$ are successive stages in $S(x)$ and that $g_t(x \neq g_s(x)$. Let $k = k_s(x)$. The fact that $x \leqslant a_{k,s}$ implies that $g_t(a_{k,s}) \neq g_s(a_{k,s})$. Let $m$ be the smallest follower for $\sigma$ such that $g_t(a_{m,s}) \neq g_s(a_{m,s})$; so $m \leqslant k$. Let $u$ be the least stage $u \in (s,t)$ at which $g_{u+1}(a_{m,s}) \neq g_u(a_{m,s})$. Then $m$ is not cancelled by stage $u$, and as it did not receive attention at stages between $s$ and $u$, it requires attention at stage $u$, and receives it.

Above we calculated for any follower $k$ for which $I_\sigma(k)$ is ever appointed a bound $h(k) = 2^{r_\sigma(k)-k} + 1$ for the number of times $k$ receives attention. It follows that the number of stages $s \in S(x)$ such that $g_t(x) \neq g_s(x)$ (where $t$ is the next stage in $S(x)$) is bounded by $\sum h(m)$, where $m$ is a follower for $\sigma$ at stage $s = \min S(x)$ and $s_\sigma(m) \leqslant s$. From this we can construct an $\omega$-computable approximation for $g$. $\square$

It remains to show that every negative requirement is met. Let $e < \omega$ and let $\tau$ on the true path work for $N_e$; in the interesting case $\tau\hat{}\infty$ also lies on the true path. The proof of Lemma 1.7, that infinitely many markers $\eta_\tau(t)$ are not enumerated into $B$ goes through as above: say $w$ is a late stage; let $\sigma$ be the strongest node which ever acts (ends the stage) or a follower of whose receives attention after stage $w$. Then $\sigma$ extends $\tau\hat{}\infty$ and does not lie to the right of the true path. Either $\sigma$ lies to the left of the true path, in which case $\sigma$ appoints only finitely many followers; each one receives attention infinitely often. If $\sigma$ lies on the true path then Lemma 1.9 shows that $\sigma$ acts only finitely often. Hence there is a last stage $t$ at which $\sigma$ is accessible and ends the stage, or a follower for $\sigma$ receives attention. Any node $\rho$

which acts after stage $t$ is initialised at stage $t$. If $t'$ is the least stage $t' \geqslant t$ at which $\tau\hat{}\infty$ is accessible then $\eta_\tau(t')$ is not enumerated into $B$.

Thus we need to prove an analogue of Lemma 1.6. Again let $u < t$ be two late stages at which $\tau\hat{}\infty$ is accessible and suppose that $\eta_\tau(u) \notin B_{t^+}$, where again $t^+$ is the next stage after $t$ at which $\tau\hat{}\infty$ is accessible. As above, the total contribution to $\alpha_{t^+} - \alpha_t$ made by nodes that lie to the right of $\tau\hat{}\infty$ is bounded by $2^{-t}$, as all such nodes are initialised at stage $t$. It is no longer true however that nodes extending $\tau\hat{}\infty$ do not act at stages strictly between $t$ and $t^+$, nor that only one such node acts between these stages. Nonetheless, every follower $k$ for a node $\sigma \succcurlyeq \tau\hat{}\infty$ receives attention at most once between stages $t$ and $t^+$, and so the total increase in $\alpha$ attributed to such nodes is bounded by $\sum 2^{-r_\sigma(k)}$ where $\sigma \succcurlyeq \tau\hat{}\infty$, $k$ is a follower for $\sigma$ at stage $t$ and $r_\sigma(k) > u$ (again as $\eta_\tau(u) \notin B_{t^+}$). Since the numbers $r_\sigma(k)$ are distinct for distinct followers $k$ we see that this sum is bounded by $2^{-u}$. We conclude that $\alpha_{t^+} - \alpha_t$ is bounded by $2^{-(u-1)}$ and so that strings of length smaller than $u - 1$ do not enter $C_e$ between stages $t$ and $t^+$, completing the proof.

**1.3. The complexity of the original construction.** As mentioned above, the original construction in [25] constructs a noncomputable left-c.e. real $\alpha$, all of whose presentations are computable. That is, $B = \varnothing$. This construction is more complicated than the one presented above. Since we are not allowed to enumerate markers into $B$, promises that a node $\tau$ makes at an expansionary stage are binding to all. Considering one such node $\tau$ and one positive node $\sigma$ extending $\tau\hat{}\infty$, ensuring that $\sigma$ acts only finitely many times requires $\tau$ to delay making stricter restraints. Suppose that an interval $I_\sigma$ is defined at some stage $r_\sigma$. Ignoring subtleties we assume that at that stage the node $\tau$ declares that from now on, any increase in $\alpha$ between two successive $\tau$-expansionary stages must be bounded by $2^{-r_\sigma}$.

$\alpha - \alpha_{r_\sigma} \leqslant 2^{-r_\sigma}$, and that the node $\sigma$ wants to drive $\alpha$ to be to the right of $I_\sigma$. The node $\sigma$ issues a request from $\tau$: until $\sigma$'s mission is accomplished, $\tau$ should refrain from imposing stronger bounds on the increase of $\alpha$ between $\tau$-expansionary stages. In turn, since $\tau$ does not know if $\sigma$ will be accessible sufficiently many times to complete its task, it cannot abide by $\sigma$'s request indefinitely. Hence $\tau$ takes upon itself to act on $\sigma$'s behalf: at the next few $\tau$-expansionary stages, the stage ends when $\tau$ is accessible and an amount of $2^{-r_\sigma}$ is added to $\alpha$. This happens finitely many times, until $\alpha_s$ lies to the right of $I_\sigma$; after that, $\sigma$ never acts again and $\tau$ is free to make stricter promises about increases of $\alpha$.

So far the number of actions required is similar to the previous construction, but the story gets more complicated when more than one node $\tau$ is considered. Suppose now that $\tau_1$ and $\tau_2$ are two negative nodes with $\tau_1\hat{}\infty \preccurlyeq \tau_2$ and $\tau_2\hat{}\infty \preccurlyeq \sigma$. At stage $r_\sigma$ both negative nodes promise that between $\tau_i$-expansionary stages, $\alpha$ increases by no more than $2^{-r_\sigma}$. So we cannot increase $\alpha$ by the desired $2^{-r_\sigma}$ until the next $\tau_2$-expansionary stage. Now $\tau_1$ is in a bind. It cannot act on its own to help $\sigma$, it seems; but it does not know if there are infinitely many $\tau_2$-expansionary stages, so it cannot wait for one while not making its own promises about $\alpha$ stricter.

The solution is to follow a nested loop. Suppose that $t \geqslant r_\sigma$ is $\tau_2$-expansionary. Unlike $\tau_1$, the node $\tau_2$ can afford to wait until $\sigma$ is done, and so keeps the bound between $\tau_2$-expansionary stages to be $2^{-r_\sigma}$. Until the next $\tau_2$-expansionary stage the entire construction is restricted to the interval $[\alpha_s, \alpha_s + 2^{-r_\sigma})$. At stage $s$ the node $\tau_1$ announces a strict bound, roughly $2^{-t}$. At subsequent $\tau_1$-expansionary stages we increase $\alpha$ on $\sigma$'s behalf, say up to $\alpha_s + 2^{-r_\sigma}/2$. This means that at the next $2^{t-r_\sigma-1}$ many $\tau_1$-expansionary stages, the path of accessible nodes ends at $\tau_1$. After this action, the construction continues without special action on $\sigma$'s behalf but with sufficient initialisations to the right of $\sigma$ so that the promise that $\alpha < \alpha_s + 2^{-r_\sigma}$ is honoured. At the next $\tau_2$-expansionary stage we repeat the cycle again: a new, stricter bound $2^{-t'}$ is announced by $\tau_1$; for the next $2^{t'-r_\sigma-1}$ many $\tau_1$-expansionary stages we act on behalf of $\sigma$, and then again wait for a new $\tau_2$-expansionary stages. After no more than $2^{r_\sigma}$ many such iterations we meet $\sigma$'s requirement.

Consider now how this argument would translate to a permitting argument. We know in advance that to meet $\sigma$ on the follower $k$ we will need sometime like $2^{r_\sigma(k)}$ many $\tau_2$-expansionary stages. If $t$ is one of these stages, we will need roughly $2^t$ many permissions for $\tau_1$ to act on $\sigma$'s behalf. We will not know the next value until we actually observe the next $\tau_2$-expansionary stage. So the number of total permissions required is given by an $\omega^2$-c.a. function: the number of times we change our mind about how many permissions we need for follower $k$ is bounded by the computable number $2^{r_\sigma(k)}$. If we know that $\mathbf{d}$ is not totally $\omega^2$-c.a. then we can meet $\sigma$'s requirement. If there are three negative nodes $\tau_1, \tau_2$ and $\tau_3$ below $\sigma$, then we have three layers of nesting of loops, and so the number of permissions is now given by an $\omega^3$-c.a. function, and so on. Overall we see that this kind of permission is related to non-total $< \omega^\omega$-c.a. permission.

## 2. Total $\omega$-c.a. anti-permitting

We prove part (2) of Theorem I.0.2. Let $\alpha$ be a left-c.e. real such that $\deg_{\mathrm{T}}(\alpha)$ is totally $\omega$-c.a. We enumerate a presentation $C$ of $\alpha$ which is Turing equivalent to $\alpha$.

The technique we use is the so-called "anti-permitting" technique described in [**17**, **5**]. In some sense it is a mirror image of the previous construction. As discussed earlier in this chapter, we view $\alpha$ as an infinite binary sequence via binary expansion. This is unique as we may assume that $\alpha$ is noncomputable. In fact we will later make significant use of the assumption that $\alpha$ is noncomputable; it will help us lift array computable anti-permitting to total-$\omega$-c.a. anti-permitting.

**2.1. Basic algorithm and plan.** Before we describe the construction we discuss one of the algorithms that will be used in the construction and the high-level plan for the construction.

*Building presentations.* We want to enumerate a presentation $C$ of $\alpha$. We follow a proof by J. Miller of the Kraft-Chaitin theorem of algorithmic randomness theory (see [**21**]). We fix an increasing approximation $\langle \alpha_s \rangle$ of $\alpha$, where each $\alpha_s \in [0, 1)$ is a dyadic rational number. We will not require that $\lambda(C_s) = \alpha_s$ for all stages $s$. We will only add strings to $C_s$ to bring its measure up to $\alpha_s$ at stages $s$ at which we receive some "certification" that various initial segments of $\alpha_s$ are correct. This process of certification is the heart of the construction. Ignoring the mechanics of certification for the moment, let $s$ be a stage at which we want to add strings to $C_{s-1}$ to ensure that $\lambda(C_s) = \alpha_s$. The instruction will be:

Adding strings to $C$.

> Let $\beta = \alpha_s - \lambda(C_{s-1})$. For each $k$ such that $\beta(k) = 1$, add a single string of length $k$ to $C_s$.

(Recall that we consider $\beta$ as a string via its binary expansion.) Since $\beta = \sum \beta(k)2^{-k}$, it is clear that $\lambda(C_s) = \alpha_s$. The pertinent point is:

> if $\beta \geqslant 2^{-k}$ then a string of length at most $k$ enters $C_s$.

We need to argue though that the instruction can be carried out while keeping $C_s$ prefix-free. This is done by using an auxiliary sequence of strings. At each stage $t$ we will have reserved strings $\tau_{k,t}$ for each $k$ such that $\lambda(C_t)(k) = 0$, with $|\tau_{k,t}| = k$, such that $C_t \cup \{\tau_{k,t} : \lambda(C_t)(k) = 0\}$ is prefix-free. We work with each length at a time, so we may assume that $\beta = 2^{-k}$, i.e., we want to add a single string of length $k$ to $C_{s-1}$. Since $\lambda(C_{s-1}) < \alpha_s < 1$ there is some $m \leqslant k$ such that $\lambda(C_{s-1})(m) = 0$. Let $m$ be the greatest such. So the change in $\alpha_s$ compared to $\alpha_{s-1}$ is that the $m^{\text{th}}$ bit changes from 0 to 1, and the $n^{\text{th}}$ bit, for all $n \in (m, k]$ (if $k > m$) changes from 1 to 0. So $\tau_{n,s} = \tau_{n,s-1}$ for $n \notin [m, k]$. We then add $\tau_{m,s-1}\hat{\ }0^{k-m}$ to $C_s$ and for $n \in (m, k]$ we let $\tau_{n,s} = \tau_{m,s-1}\hat{\ }0^{n-m-1}1$. See Figure 1.

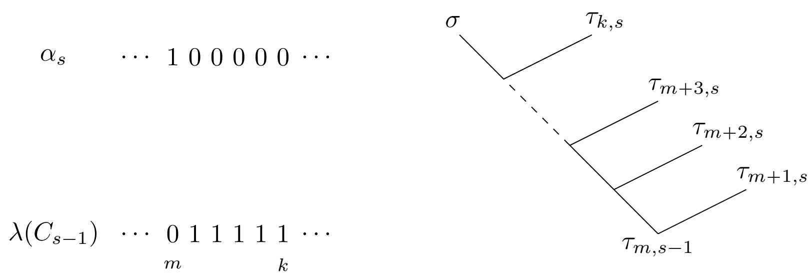


FIGURE 1. $C_s = C_{s-1} \cup \{\sigma\}$.

Henceforth the details of the auxiliary strings are assumed, and we only invoke the algorithm above.

*Layers.* Suppose that we enumerate a presentation $C$ as described above. Why is it the case that $C$ might not compute $\alpha$? We have arranged for that in the previous construction: we gradually add small amounts to $\alpha$. If we update $C$ each time, this means that only long strings enter $C$. However the cumulative effect on $\alpha$ may be big, which is a change that $C$ does not comprehend.

In terms of binary expansions, the problematic case is when $\alpha_s$ contains a long block of 1's. Suppose that $\alpha_s \upharpoonright_{(m,k]}$ is a string of ones. Then adding $2^{-k}$ to $\alpha_s$ results in adding a string of length $k$ to $C$ but changes the bit $\alpha(m)$.

We can try to prevent this by setting up *layers* which contain sufficiently many zeros, and appropriately set uses for computing $\alpha$ from $C$. We set markers $\delta_0 < \delta_1 < \delta_2 < \cdots$ such that the block $\alpha \upharpoonright_{[\delta_n, \delta_{n+1})}$ contains many zeros (and the idea is that the markers may increase with time, but hopefully settle down eventually). We let $\delta_{n+1}$ be the use for reducing $\alpha \upharpoonright_{\delta_n}$ to $C$. See Figure 2. Here since $C$ is a set of strings, by use $u$ we mean querying the oracle on strings of length less than $u$.

Now the point is that if between stages $s$ and $s+1$, $\alpha$ changes on the interval $[\delta_{n-1}, \delta_n)$, then since the interval $\alpha_s \upharpoonright_{[\delta_n, \delta_{n+1})}$ contains zeros, the increase $\alpha_{s+1} - \alpha_s$

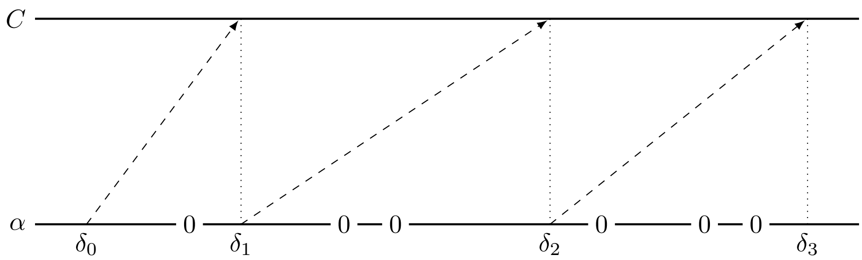


FIGURE 2. Layers. The dashed lines represent the reduction of $\alpha$ to $C$.

is greater than $2^{-\delta_{n+1}}$; and so if we update $C$ then some string of length smaller than $\delta_{n+1}$ will enter $C$ and allow us to fix the reduction of $\alpha \restriction_{\delta_n}$ to $C$.

After this increase, we may have $\alpha_{s+1} \restriction_{[\delta_n, \delta_{n+1})}$ be all ones, but we can increase $\delta_{n+1}$ so that the new interval contains many zeros. However, it is possible that no string of length smaller than $\delta_n$ entered $C_{s+1}$; so we cannot increase $\delta_n$, as this is the use of computing $\alpha \restriction_{\delta_{n-1}}$. Which is a problem, since we lost a zero on the interval $[\delta_{n-1}, \delta_n)$. Note though that we lose at most one zero, or the increase is beyond $2^{-\delta_n}$ and strings of length smaller than $\delta_n$ will in fact enter $C_{s+1}$. *So if we can ensure that the number of times this happens is at most the number of zeros we originally set up in the interval* $[\delta_{n-1}, \delta_n)$, the construction will succeed. This is precisely what the certification process gives us.

*Certification.* The certification process relies on the computational weakness of $\deg_{\mathrm{T}}(\alpha)$. We enumerate a Turing functional $\Gamma$ with intended oracle $\alpha$, and ensure that $\Gamma(\alpha)$ is total. We know that the function $\Gamma(\alpha)$ is $\omega$-c.a. Suppose that $\langle g_s, o_s \rangle$ is an $\omega$-computable approximation for $\Gamma(\alpha)$. When a computation $\Gamma_s(\alpha_s, n)$ is destroyed, we redefine it with a new value. It follows that there are fewer than $o_0(n)$ many stages $s$ at which $\Gamma_s(\alpha_s, n) = g_s(n)$ and the computation $\Gamma_s(\alpha_s, n)$ is $\alpha$-incorrect.

The plan for setting up the layers is then as follows. Given $\delta_{n-1}$, calculate $o_0(n)$ and let $\delta_n$ be sufficiently large so that the current version of $\alpha$ contains at least $o_0(n)$ many zeros in the interval $[\delta_{n-1}, \delta_n)$. Define $\Gamma(\alpha, n)$ with use $\delta_n$. Recall that since the oracle $\alpha$ is given our convention is that by use $u$ we mean that $\alpha \restriction_u$ computes $\Gamma(n)$, not $\alpha \restriction_{u+1}$.

We can then carry out our original plan. Suppose that for a while, everything is stable, but that at some stage $t$ we see an increase in $\alpha_{t+1}$, say a quantity $q \in (2^{-\delta_{n+1}}, 2^{-\delta_n}]$. As discussed above, this may change the bits of $\alpha$ on the interval $[\delta_{n-1}, \delta_n)$. This means that now $\Gamma(\alpha, n)\uparrow$. We define a new value for the computation (say $t$) with the same use. Before we act, we wait for certification: for a later stage $s$ at which we see that $g_s(n)$ equals that new value $t$. Only once we've seen this certification do we add strings to $C_{s+1}$ (of lengths between $\delta_n$ and $\delta_{n+1}$). Compared to $\alpha_t \restriction_{[\delta_{n-1}, \delta_n)}$, the interval $\alpha_s \restriction_{[\delta_{n-1}, \delta_n)}$ contains one zero fewer. But this is compensated by the change in $g$, which ensures that $o_s(n) < o_t(n)$. Note though that while waiting, further increases can occur. If the amount increases beyond $2^{-\delta_n}$ then we can abandon $\delta_n$ and repeat the work on the interval $[\delta_{n-2}, \delta_{n-1})$.

*Uniformity, and simple permitting.* All is well, except that even if we ensure that $\Gamma(\alpha)$ is total, we cannot effectively find an $\omega$-computable approximation for $\Gamma(\alpha)$. We need to guess one. Let $\langle g^e \rangle$ be an enumeration of the $\omega$-c.a. functions, equipped with tidy $(\omega+1)$-computable approximations $\langle g^e_s, o^e_s \rangle$ (Proposition II.1.7). We perform countably many constructions which are almost independent of each other. The $e^{\text{th}}$ construction guesses that $\Gamma(\alpha) = g^e$, and based on this guess enumerates a prefix-free set $C^e$ and a reduction of $\alpha$ to $C^e$. If the guess is correct then the construction will succeed.

Since they enumerate distinct sets and reductions, there is very little interaction between the different constructions. However they do combine forces in defining $\Gamma(\alpha)$. To keep things simple, the $e^{\text{th}}$ construction defines $\Gamma(\alpha, n)$ for inputs $n \in \omega^{[e]}$ (the $e^{\text{th}}$ column of $\omega$). The catch is that even if the guess that $\Gamma(\alpha) = g^e$ is incorrect, an eventual $\alpha$-correct definition of $\Gamma(\alpha, n)$ must be made by the $e^{\text{th}}$ construction, for all $n \in \omega^{[e]}$.

Even while waiting for an agreement between $\Gamma(\alpha, n)$ and $g^e(n)$, the $e^{\text{th}}$ construction can keep defining new values of markers $\delta_m$ for $m > n$ in $\omega^{[e]}$, and with them computations $\Gamma(\alpha, m)$. If $\deg_{\mathrm{T}}(\alpha)$ were array computable this would not be a problem. Recall that we need to ensure that the block ending with $\delta_m$ must contain at most $o^e_s(m)$ many zeros (where $s$ is the stage at which we make the definition). If we know that $\Gamma(\alpha)$ is say id-c.a., then we can work with a list of tidy $(\mathrm{id}+1)$-computable approximations, and so $o^e_0(m) = m$ for all $e$ and $m$, and we can find how large $\delta_m$ must be. However under the weaker assumption that $\deg_{\mathrm{T}}(\alpha)$ is totally $\omega$-c.a., we need to work with what are essentially (if not formally) partial approximations. So the conflict is that we need to define $\delta_m$ even if $o^e_s(m) = \omega$ for all $s$; so we cannot wait for a value $o^e_s(m) < \omega$ to show up. But if we define $\delta_m$ before seeing $o^e_s(m) < \omega$ then we will not have enough zeros and will not be able to carry out the construction outlined above, *even if the* $e^{\text{th}}$ *guess is correct.*

The solution (as in [17]) is to make use of the fact that $\alpha$ is noncomputable. We actually use simple permitting. This is perhaps paradoxical in an anti-permitting argument. But of course the point is that noncomputable (simple) permitting is weaker than non-total $\omega$-c.a. permitting, and so the former can co-exist with the negation of the latter.

What we do is go ahead and define a computation $\Gamma(\alpha, m)$ without waiting for $o^e_s(m)$ to give us a natural number. But we wait with the definition of the reduction of $\alpha$ to $C^e$ (which is fine, as it is local to the $e^{\text{th}}$ construction). Once we see the value $o^e_s(m)$ we wait for a voluntary change in $\alpha$ below the use $\gamma(m)$. Simple permitting will ensure that for infinitely many $m$ we will see such changes (provided of course that the approximation is eventually $\omega$-computable). If we see such a change then we can now define a new large value for $\gamma(m)$, bounding sufficiently many zeros, and declare it to be one of our markers $\delta_m$. Note again that to move $\delta_m$ we need not only an $\alpha$-change below $\delta_m$, but also a change in $C^e$ on strings of length below $\delta_m$, *if the reduction of* $\alpha\restriction_{\delta_{m-1}}$ *to* $C^e$ *has already been defined.* This is why it is important to keep this reduction undefined until we see $o^e_s(m) < \omega$.

This discussion contained all the ideas needed for the proof, and so we turn to giving the formal details.

**2.2. Total $\omega$-c.a. anti-permitting: the details.** As discussed, we are given a noncomputable left-c.e. real $\alpha \in [0, 1)$ with an increasing approximation $\langle \alpha_s \rangle$. We use a list $\langle g^e \rangle$ of all $\omega$-c.a. functions, with tidy $(\omega+1)$-computable approximations $\langle g^e_s, o^e_s \rangle$.

We enumerate a Turing functional $\Gamma$, with intended oracle $\alpha$, viewed as an element of Cantor space.

For every $e < \omega$ we perform the $e^{\text{th}}$ construction. These constructions are independent of each other. Fix some $e < \omega$. In the $e^{\text{th}}$ construction we enumerate a prefix-free c.e. set $C^e$ and define $\Gamma(\alpha, m)$ for all $m \in \omega^{[e]}$. Also, we define an increasing sequence of numbers $k^e(0) < k^e(1) < \dots$ (the list may eventually be finite or infinite). All of the numbers $k^e(n)$ are elements of $\omega^{[e]}$. These will be the numbers that are permitted (simply) and so they will be the ones that will be used as inputs for defining the layers. We renumber our markers by letting $\delta^e_n = \gamma(k^e(n))$.

*The beginning of stage $s$.* By the beginning of a stage $s$ we will have already:

(1) Enumerated the set $C^e_s$;
(2) Defined the sequence $k^e(0), k^e(1), \dots, k^e(v)$ for some $v = v^e_s$, such that each $k^e_n \in \omega^{[e]} \cap s$. For brevity we let $b^e_s = k^e(v^e_s)$ be the last element of this sequence.
(3) Defined computations $\Gamma_s(\alpha_s, m)$ for all $m \in \omega^{[e]} \cap s$, with uses $\gamma_s(m)$. For $n \leqslant v^e_s$ we let $\delta^e_{n,s} = \gamma_s(k^e(n))$.

The uses $\gamma_s(m)$ are not quite monotone:

- if $k^e(n-1) < m < k^e(n)$ for some $n \leqslant v^e_s$ then $\gamma_s(m) = 0$. That is, the computation $\Gamma_s(\alpha_s, m)$ does not look at the oracle and so is never destroyed. These inputs $m$ were discarded when we got permission to use $k^e(n)$ to define the next layer ending with $\delta^e_{n,s}$.
- Otherwise, the uses are monotone:
  - $\delta^e_{n,s} = \gamma_s(k^e(n)) < \gamma_s(k^e(n+1)) = \delta^e_{n+1,s}$ for all $n < v^e_s$;
  - If $n \leqslant v^e_s$ and $m > b^e_s$ then $\delta^e_{n,s} < \gamma_s(m)$;
  - If $b^e_s < m < m'$ then $\gamma_s(m) < \gamma_s(m')$.

See Figure 3.

*The $e^{\text{th}}$ construction.* The construction begins at stage $s = \min \omega^{[e]}$. At that stage we define $k^e(0) = s$ and $C^e_{s+1} = \varnothing$. We define a new computation $\Gamma_{s+1}(\alpha_{s+1}, s) = 0$ with use 1. So $\delta^e_{0,s+1} = 1$. Recall our convention that $\alpha = 0.\alpha(1)\alpha(2)\cdots$. This means that $\alpha \upharpoonright_k$ is the bit-sequence $\alpha(1)\alpha(2)\cdots\alpha(k-1)$. If we define a computation with use 1 this means that the oracle is not consulted and so this computation is never destroyed.

Now suppose that $s > \min \omega^{[e]}$. We give the instructions for the $e^{\text{th}}$ construction at stage $s$.

STEP 1: REDEFINING DESTROYED COMPUTATIONS $\Gamma(\alpha, m)$.

We may see that some of the computations $\Gamma_s(\alpha_s, m)$ are destroyed by the change from $\alpha_s$ to $\alpha_{s+1}$. If none of these computations are destroyed then we skip to step 2 below.

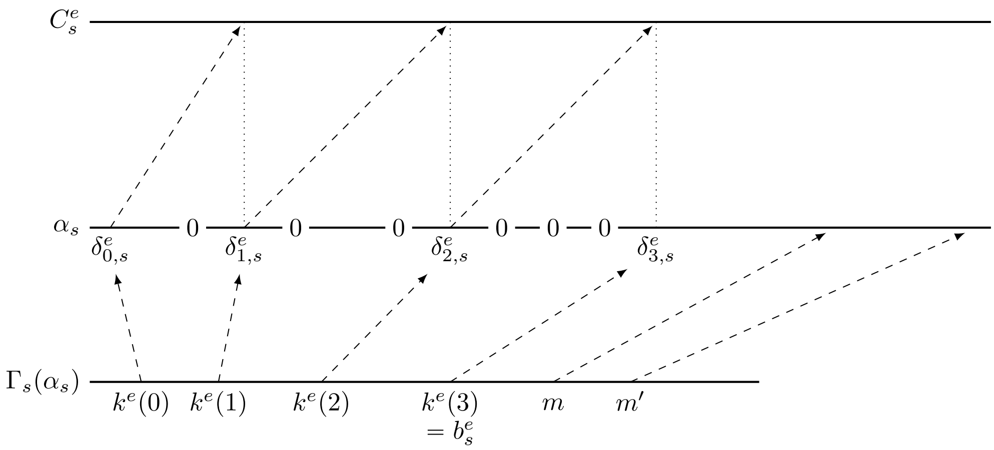


FIGURE 3. The $e^{\text{th}}$ construction at stage $s$. In this example $v^e_s = 3$, and $\Gamma_s(\alpha_s)$ is also defined on $m' > m > b^e_s = k^e(3)$, the two next elements in $\omega^{[e]}$.

Otherwise we need to define new computations $\Gamma_{s+1}(\alpha_{s+1}, m)$ for $m$ for which the computations were destroyed. In all but one case the value of the new computations will be $s+1$, and so to define these computations we only need to specify their use $\gamma_{s+1}(m)$.

Let $p$ be the smallest element of $\omega^{[e]}$ such that $p < s$ and $\Gamma_s(\alpha_{s+1}, p)\uparrow$. There are three cases.

FIRST CASE. USELESS CHANGE: $p > b^e_s$ BUT $o^e_s(p) = \omega$.

For all $m \in [p, s) \cap \omega^{[e]}$ set $\gamma_{s+1}(m) = \gamma_s(m)$. We don't increase the uses, to ensure that they do not go to infinity.

SECOND CASE. MAKING USE OF SIMPLE PERMISSION: $p > b^e_s$ AND $o^e_s(p) < \omega$.

In this case we add $p$ as the new last element of the list of useful inputs. That is, we define $k^e(n) = p$ where $n = v^e_s + 1 = v^e_{s+1}$; so $p = b^e_{s+1}$.

- For $m \in (b^e_s, b^e_{s+1}) \cap \omega^{[e]}$ define $\Gamma_{s+1}(\alpha_{s+1}, m) = \Gamma_s(\alpha_s, m)$ with use 1. We use the previous value to keep the functional consistent.
- Set $\gamma_{s+1}(p)$ (which of course equals $\delta^e_{n,s+1}$) to be the least $u > \delta^e_{n,s} + 1$ such that the block $\alpha_{s+1} \upharpoonright_{[\delta^e_{n-1,s}, u)}$ contains at least $o^e_s(p) + 2$ many zeros.
- For $m \in (p, s) \cap \omega^{[e]}$ set $\gamma_{s+1}(m)$ to be large.

THIRD AND MAIN CASE: $p = k^e(q)$ FOR SOME $q \leqslant v^e_s$.

Let $n \in [q, v^e_s]$. Let $\beta = \alpha_{s+1} - \lambda(C^e_s)$.

- If $\beta \leqslant 2^{-\delta^e_{n,s}}$ then set $\delta^e_{n,s+1} = \delta^e_{n,s}$ (in other words, set $\gamma_{s+1}(k^e(n)) = \gamma_s(k^e(n))$).

- If $\beta > 2^{-\delta^e_{n,s}}$ then set $\delta^e_{n,s+1}$ to be the least possible value greater than $\delta^e_{n-1,s} + 1$ so that the block $\alpha \restriction_{[\delta^e_{n-1},\delta^e_n)} [s+1]$ contains at least $o^e_s(n) + 2$ many zeros.

As in the second case, for all $m \in (b^e_s, s) \cap \omega^{[e]}$ set $\gamma_{s+1}(m)$ to be large.

This defines the computations $\Gamma_{s+1}(\alpha_{s+1}, m)$ for all $m < s$ in $\omega^{[e]}$ and concludes the first step.

Step 2: updating $C^e$.
Let $\beta = \alpha_{s+1} - \lambda(C^e_s)$. Suppose that $\beta > 0$ and that for all $n \leqslant v^e_s$, $n > 0$ such that $\beta \leqslant 2^{-\delta^e_{n-1,s}}$ we have $\Gamma(\alpha, k^e(n)) = g^e(n)\,[s+1]$. Then enumerate strings into $C^e_{s+1}$ following the algorithm above to ensure that $\lambda(C^e_{s+1}) = \alpha_{s+1}$.

Step 3: a new computation.
At the very end of the stage, if $s \in \omega^{[e]}$ we define a new computation $\Gamma_{s+1}(\alpha_{s+1}, s)$ with new, large use.

This concludes the instructions for stage $s > \min \omega^{[e]}$.

*Verification.* Each functional $\Gamma_s$ is consistent for $\alpha_s$. This uses the fact that $\langle \alpha_s \rangle$ is an increasing approximation, and that at every stage we define a new computation $\Gamma_{s+1}(\alpha_{s+1}, m)$ only if $\Gamma_s(\alpha_{s+1}, m)\uparrow$, or otherwise we let $\Gamma_{s+1}(\alpha_{s+1}, m) = \Gamma_s(\alpha_s, m)$.

Lemma 2.1. *$\Gamma(\alpha)$ is total.*

Proof. Let $m < \omega$; let $e$ be such that $m \in \omega^{[e]}$. Let $v^e = \sup_s v^e_s$.

If there is some $n < v^e$ such that $k^e(n) < m < k^e(n+1)$ then at the stage at which $k^e(n+1)$ is defined we define a new computation $\Gamma(m)$ with use 1. Recall that this means that the oracle is not consulted. So certainly $\Gamma(\alpha, m)\downarrow$.

Suppose that $m = k^e(n)$ for some $n \leqslant v^e$ or that $v^e < \omega$ and $m > b^e = k^e(v^e)$. By induction on such $m$ we show that $\Gamma(\alpha, m)\downarrow$. For every $s > m$ the computation $\Gamma_s(\alpha_s, m)$ converges. To show that $\Gamma(\alpha, m)\downarrow$ it is sufficient to show that the sequence $\langle \gamma_s(m) \rangle$ is bounded. For if it is bounded by some value $u$ and $\alpha_s \restriction_u = \alpha \restriction_u$, then $\Gamma_s(\alpha_s, m)$ is an $\alpha$-correct computation.

First suppose that $m = k^e(n)$ for some $n$. If $n = 0$ then $\gamma_{m+1}(m) = 0$ which implies that the computation $\Gamma_{m+1}(\alpha_{m+1}, m)$ is $\alpha$-correct and so is never destroyed. Suppose that $n > 0$. By induction we assume that $\delta^e_{n-1,s}$ reaches a limit $\delta^e_{n-1}$. Let $r > m$ be a stage sufficiently late so that $n \leqslant v^e_r$ and $\delta^e_{n-1,s} = \delta^e_{n-1}$ for all $s \geqslant r$. We note that the fact that $n \leqslant v^e_r$ implies that $o^e_r(m)\downarrow$. Let $u$ be the least number greater than $\delta^e_{n-1}$ such that the block $\alpha \restriction_{[\delta^e_{n-1}, u)}$ contains at least $o^e_r(m)$ many zeros; such a number exists since $\alpha$ is not a dyadic rational. By increasing $r$ we may assume that $\alpha_r \restriction_u = \alpha \restriction_u$ (and so $\alpha_s \restriction_u = \alpha \restriction_u$ for all $s \geqslant r$). If $s \geqslant r$ is a stage at which $\gamma_{s+1}(m)$ is redefined then we choose $\gamma_{s+1}(m) \leqslant u$.

Now suppose that $v^e < \omega$ and $m > b^e$. Let $m'$ be $m$'s predecessor in $\omega^{[e]}$. By induction find a stage $r > m$ sufficiently late so that the computation $\Gamma_r(\alpha_r, m')$ is $\alpha$-correct. At every stage $s \geqslant r$ at which we redefine $\gamma_{s+1}(m)$ we let $\gamma_{s+1}(m) = \gamma_s(m)$. □

Since we assume that $\deg_{\mathrm{T}}(\alpha)$ is totally $\omega$-c.a. there is some $e$ such that $\Gamma(\alpha) = g^e$ and the approximation $\langle g^e_s, o^e_s\rangle$ is eventually $\omega$-computable. We fix such $e$. From now we only concern ourselves with the $e^{\text{th}}$ construction. For clarity of notation we omit the superscript $e$ from all the associated objects (we write $g_s$ for $g^e_s$, $C$ for $C^e$, $\delta_{n,s}$ for $\delta^e_{n,s}$ and so on).

LEMMA 2.2. $\lim_s v_s = \omega$.

PROOF. Assume for a contradiction that $v = \lim_s v_s$ is finite. Let $r$ be a stage sufficiently late so that by stage $r$, $\delta_{v,s}$ has reached a limit $\delta_v$ and $\alpha_r \upharpoonright_{\delta_v} = \alpha \upharpoonright_{\delta_v}$. The assumption for contradiction means that at all stages $s > r$, for all $m \in \omega^{[e]} \cap s$ such that $o_s(m) < \omega$, the computation $\Gamma_s(\alpha_s, m)$ is $\alpha$-correct. This implies that $\alpha$ is computable. Given $u < \omega$, to compute $\alpha \upharpoonright_u$ we pick $m > u, r$ in $\omega^{[e]}$ and wait for a stage $s > m$ at which $o_s(m) < \omega$; so $\alpha_s \upharpoonright_{\gamma_s(m)} = \alpha \upharpoonright_{\gamma_s(m)}$. But $\gamma_s(m) > m > u$. ☐

We can show that $C$ is a presentation of $\alpha$.

LEMMA 2.3. $\lambda(C) = \alpha$.

PROOF. Suppose not. Let $n$ be sufficiently large so that $2^{-n} < \alpha - \lambda(C)$. But if $s$ is a very late stage then all markers $\delta_{m,s}$ for all $m \leqslant n$ have stabilised to their final values and are all certified: $g_s(k(m)) = \Gamma(\alpha, k(m))$ for all $m \leqslant n$. Also assume that $\alpha - \alpha_s < 2^{-n}$ and $\alpha_s - \lambda(C_s) > 2^{-n}$. Then at stage $s$ we would increase $C$ to have measure $\alpha_s$, which is a contradiction. ☐

The next lemma (really an observation) is trivial but useful. Both parts rely on the fact that for all $\beta \in [0, 1)$ and $k \geqslant 1$, $\beta \upharpoonright_k$ (as a number in binary) is the integral part of $2^{k-1}\beta$.

LEMMA 2.4. *Let $t < s$ and $k \geqslant 1$.*

1. *If $\alpha_t - \alpha_s \geqslant 2^{-(k-1)}$ then $\alpha_t \upharpoonright_k \neq \alpha_s \upharpoonright_k$.*
2. *If $\alpha_s - \alpha_k \leqslant 2^{-k}$ and further $\alpha_t(k) = 0$ then $\alpha_t \upharpoonright_k = \alpha_s \upharpoonright_k$.*

The following is the main combinatorial lemma.

LEMMA 2.5. *Let $s$ be a stage, and let $n > 0$, $n \leqslant v_s$. The block $\alpha \upharpoonright_{[\delta_{n-1}, \delta_n)} [s]$ contains a zero.*

PROOF. Fix $n$. For brevity let $m = k(n)$. Suppose that $s$ is a stage and $n \leqslant v_s$.

As above, say that the marker $\delta_{n,s}$ is *certified* at stage $s$ if $\Gamma_s(\alpha_s, k(n)) = g_s(k(n))$. Let $S_{\mathtt{cert}}$ be the set of such stages. This set contains a final segment of $\omega$.

We say that the marker $\delta_{n,s}$ is *redefined* if $\Gamma_s(\alpha_{s-1}, m)\uparrow$ and either

- $v_{s-1} = n - 1$, i.e. $\delta_{n,s}$ is the very first value of this marker; or
- $\beta = \alpha_s - \lambda(C_{s-1}) > 2^{-\delta_{n,s-1}}$.

Let $S_{\mathtt{redef}}$ be the set of such stages $s$.

Let $S = S_{\mathtt{cert}} \cup S_{\mathtt{redef}}$. We show by induction on the stages $s$ for which $n \leqslant v_s$ that:

(a) If $s \in S$ then the block $\alpha \upharpoonright_{[\delta_{n-1}, \delta_n)} [s]$ contains at least $o_s(m) + 2$ many zeros.
(b) If $s \notin S$ then the block $\alpha \upharpoonright_{[\delta_{n-1}, \delta_n)} [s]$ contains at least $o_s(m) + 1$ many zeros.

In either case the number is positive, and so the lemma follows.

The induction starts with $s = \min S_{\texttt{redef}}$. The instructions ensure that (a) holds at every stage $s \in S_{\texttt{redef}}$.

Let $t \in S$ and suppose that $(a)$ has already been verified for stage $s$. Let $r$ be the next stage in $S$ after stage $t$. We verify that $(a)$ holds at stage $r$ and that $(b)$ holds at all stages $s \in (t, r)$.

The marker $\delta_{n,s}$ is constant for $s \in [t, r)$; we denote this fixed value by $\delta_n$ (note that this is not necessarily the final value of this marker). Similarly define $\delta_{n-1}$.

Now for brevity let:

- $A$ be the set of stages $u \in (t, r)$ such that $C_u \neq C_{u-1}$.
- If $r \in S_{\texttt{redef}}$ let $B$ be the set of stages $s \in (t, r)$ such that $\alpha_s \restriction_{\delta_n} \neq \alpha_{s-1} \restriction_{\delta_n}$; if $r \notin S_{\texttt{redef}}$ let $B$ be the set of such stages in the interval $(t, r]$.

We make two observations.

(1) Let $u \in A$. Then $\alpha_u - \lambda(C_{u-1})$ is strictly greater than $2^{-\delta_{n-1}}$. This is because $u \notin S_{\texttt{cert}}$.
(2) Let $s \in B$. Then $\alpha_s - \lambda(C_{s-1}) \leqslant 2^{-\delta_n}$. For otherwise $s \in S_{\texttt{redef}}$.

In particular, $A$ and $B$ are disjoint.

Suppose that $B$ is empty. Then $\alpha_{r-1} \restriction_{\delta_n} = \alpha_t \restriction_{\delta_n}$; and if $r \notin S_{\texttt{redef}}$ then $\alpha_r \restriction_{\delta_n} = \alpha_t \restriction_{\delta_n}$. Since $o_s(m) \leqslant o_t(m)$ for all $s \in (t, r]$, we see that $(b)$ holds for all $s \in (t, r)$. If $r \in S_{\texttt{redef}}$ then we already know that (a) holds at $r$. If $r \notin S_{\texttt{redef}}$ then the latter equality ensures that (b) holds at stage $r$.

We assume therefore that $B$ is nonempty.

Suppose that $A$ is nonempty. We claim that $A < B$. That is, there are no $s \in B$ and $u \in A$ with $s < u$. For a contradiction, suppose there are. By choosing a maximal $s$ and then minimal $u$ we can find $s \in B$, $u \in A$ such that $s < u$ but the interval $(s, u)$ is disjoint from both $A$ and $B$. Since $A \cap [s, u)$ is empty we see that $C_{s-1} = C_{u-1}$. Let $q = \lambda(C_{s-1})$; then $\alpha_s - q \leqslant 2^{-\delta_n}$ and $\alpha_u - q > 2^{-\delta_{n-1}}$. Since $\delta_{n-1} > \delta_n + 1$, this means that $\alpha_u - \alpha_s > 2 \cdot 2^{-\delta_n}$. By Lemma 2.4, $\alpha_u \restriction_{\delta_n} \neq \alpha_s \restriction_{\delta_n}$. This contradicts the assumption that $B \cap (s, u]$ is empty.

Thus, we let $t' = \max A$ if $A$ is nonempty, and $t' = t$ otherwise. Then $\alpha_{t'} \restriction_{\delta_n} = \alpha_t \restriction_{\delta_n}$.

Let $r' = \max B$. Then $\alpha_{r'} \restriction_{\delta_n} = \alpha_{r-1} \restriction_{\delta_n}$; and if $r \notin S_{\texttt{redef}}$ then $\alpha_{r'} \restriction_{\delta_n} = \alpha_r \restriction_{\delta_n}$. Also we note that $C_{r'} = C_{t'}$ and so $\alpha_{r'} - \lambda(C_{t'}) \leqslant 2^{-\delta_n}$.

Let $k$ be the rightmost zero in the block $\alpha \restriction_{[\delta_{n-1}, \delta_n)} [t]$ – the greatest $k < \delta_n$ such that $\alpha_t(k) = 0$. Such $k$ exists by induction.

Since $\alpha_{r'} - \alpha_{t'} \leqslant 2^{-\delta_n}$ and $\alpha_{t'}(k) = \alpha_t(k) = 0$, Lemma 2.4 says that $\alpha_{r'} \restriction_k = \alpha_{t'} \restriction_k$. Overall, we see that $\alpha_{r-1} \restriction_k = \alpha_t \restriction_k$; and if $r \notin S_{\texttt{redef}}$ then $\alpha_r \restriction_k = \alpha_t \restriction_k$.

The block $\alpha_t \restriction_{[\delta_{n-1}, k)}$ contains at least $o_t(m) + 1$ many zeros. Since $o_s(m) \leqslant o_t(m)$ for all $s > t$, we see that (b) holds for all stages $s \in (t, r)$.

Now consider $r$. We may assume that $r \notin S_{\texttt{redef}}$. Then the argument above shows that the block $\alpha_r \restriction_{[\delta_{n-1}, \delta_n]}$ contains at least $o_t(m) + 1$ many zeros. Further, $\delta_{n,r} = \delta_n$ and $\delta_{n-1,r} = \delta_{n-1}$.

We assumed that $B \neq \varnothing$. Indeed, a new computation $\Gamma_{r'}(\alpha_{r'}, m)$ is defined and $\Gamma_r(\alpha_r, m) = \Gamma_{r'}(\alpha_{r'}, m) = r'$. Since $r \in S$ it must be that $r \in S_{\texttt{cert}}$.

Thus $g_r(m) = r' > t > g_t(m)$. It follows that $o_r(m) < o_t(m)$, and so $o_r(m) + 2 \leqslant o_t(m) + 1$. This establishes (a) for stage $r$. $\square$

Finally we show that $C$ computes $\alpha$.

LEMMA 2.6. *Let $s$ be a stage and let $n < v_s$. Suppose that for all strings $\sigma$ of length at most $\delta_{n+1,s}$, $\sigma \in C$ if and only if $\sigma \in C_s$. Then $\alpha \restriction_{\delta_{n,s}} = \alpha_s \restriction_{\delta_{n,s}}$.*

PROOF. Let $s$ be a stage as described. The assumption means that for all $u > s$, if $C_u \neq C_{u-1}$ then $\lambda(C_u) - \lambda(C_{u-1}) < 2^{-\delta_{n+1,s}}$.

For brevity let $\delta_n = \delta_{n,s}$ and $\delta_{n+1} = \delta_{n+1,s}$. By induction on $t \geqslant s$ we show that $\delta_{n,t} = \delta_n$, $\delta_{n+1,t} = \delta_{n+1}$ and $\alpha_t \restriction_{\delta_n} = \alpha_s \restriction_{\delta_n}$. Suppose this is known for $t - 1 \geqslant s$.

We claim that $\beta = \alpha_t - \lambda(C_{t-1}) \leqslant 2^{-\delta_{n+1}}$. Suppose otherwise; let $u$ be the least stage $u \geqslant t$ such that $C_u \neq C_{u-1}$. Then $\lambda(C_{u-1}) = \lambda(C_{t-1})$ and $\alpha_u \geqslant \alpha_t$ and so $\lambda(C_u) - \lambda(C_{u-1}) \geqslant \beta$, contradicting our assumption on $C_s$.

The instructions (third case) now show that at stage $t-1$ we set $\delta_{n,t} = \delta_{n,t-1}$ and $\delta_{n+1,t} = \delta_{n+1,t-1}$.

Further, $\alpha_t \restriction_{\delta_n} = \alpha_s \restriction_{\delta_n}$. Since $\delta_{n+1,t-1} = \delta_{n+1}$ and $\delta_{n,t-1} = \delta_n$, Lemma 2.5 implies that the block $\alpha_{t-1} \restriction_{[\delta_n, \delta_{n+1})}$ contains a zero. If $\alpha_t \restriction_{\delta_n} \neq \alpha_{t-1} \restriction_{\delta_n}$ then by Lemma 2.4, $\alpha_t - \alpha_{t-1} > 2^{-\delta_{n+1}}$, and of course $\alpha_{t-1} \geqslant \lambda(C_{t-1})$. $\square$

CHAPTER VI

# $m$-topped degrees

Downey and Jockusch [13] showed that there are incomplete *$m$-topped* c.e. degrees: c.e. Turing degrees $\mathbf{d}$ which contained a many-one degree, greatest among all of the *c.e.* many-one degrees inside $\mathbf{d}$. These are all low$_2$, and in fact every low$_2$ c.e. degree is bounded by an $m$-topped degree [18].

In [15] we investigated the dynamics required for the Downey-Jockush construction. We showed that the cascading effect that happened in the construction led to an $\omega^\omega$-type behaviour. Specifically, we showed that there is an $m$-topped degree which is totally $\omega^\omega$-c.a. We also hinted at a proof that this is the best possible: no $m$-topped degree is totally $< \omega^\omega$-c.a. In this chapter we flesh out the details of this construction. Apart from the intrinsic interest in this result, this argument will serve as a preparation for the next chapter.

We remark that unlike the 1-3-1 embedding, the $m$-topped phenomenon cannot be captured precisely by the hierarchy of totally $< \alpha$-c.a. degrees. This is because no $m$-topped degree can be low [13]. As a result, at every level of our hierarchy there are degrees which do not bound any $m$-topped degrees. It would be interesting to see if there is a permitting argument combining non total $< \omega^\omega$-c.a.-ness and non-lowness that would yield bounding of $m$-topped degrees.

Before we give the full argument we start with easier, weaker results. We show that no totally $\omega$-c.a. degree is $m$-topped; then that no totally $\omega^2$-c.a. degree is $m$-topped; and then give the full proof.

## 1. Totally $\omega$-c.a. degrees are not $m$-topped

Let $\mathbf{d}$ be a totally $\omega$-c.a. c.e. degree. To show that $\mathbf{d}$ is not $m$-topped we need, given a c.e. $D \in \mathbf{d}$, to enumerate some c.e. set $V \leqslant_{\mathrm{T}} D$ which is not many-one reducible to $D$.

The basic module is as follows. Suppose that we want to show that the $d^{\text{th}}$ computable function $\varphi_d$ is not a many-one reduction of $V$ to $D$. We set up a finite set $X$ of followers and wait for them to be realised, which means that $\varphi_d(x)\downarrow$ for all $x \in X$. While we wait we prevent the enumeration of the followers into $V$. When they get realised we may assume that $\varphi_d(x) \notin D_s$ for all $x \in X$; otherwise we get an easy win. We then attack by enumerating some $x \in X$ into $V$. The opponent can respond by enumerating $\varphi_d(x)$ into $D$, in which case we will attack with another follower in $X$. We need to ensure two things:

- $V$ is Turing reducible to $D$; and
- $X$ is sufficiently large so that the opponent cannot always respond.

For the first we will define a functional $\Psi$ with the intention of having $\Psi(D) = V$. To be able to attack without violating this reduction we will ensure that the use $\psi_s(x)$ of any follower is greater than $\varphi_d(y)$ for any other follower. Thus a response by our opponent to our attack with $y$ will be the $D$-change which allows us to attack next with $x$.

For the second we use the "anti-permitting" method used in Chapter V. We tie the set of followers $X$ with some input $n$ for a function $\Gamma(D)$ we build which will serve as an "anchor" (or "anti-permitting number"). Since $\Gamma(D)$ is $\omega$-c.a. we find a bound $m$ on the number of times an approximation for $\Gamma(D, n)$ changes. We ensure that the use $\gamma(n)$ of $\Gamma(D, n)$ is the same as the use $\psi_s(x)$ for followers $x \in X$. So the opponent's $D$-change that allows us to attack with another follower also allows us to redefine $\Gamma(D, n)$ to have a new value and so reduce the number of changes left to the opponent. If $|X| > m$ then the opponent will not be able to always respond. See Figure 1.

As in the previous chapter we need to add a simple permitting step. Previously this was only necessary because we were working with a degree which is totally $\omega$-c.a. and not neccessarily an array computable one: the number $m$ is revealed to us eventually but is not fixed in advance; if we guess incorrecty about our approximation for $\Gamma(D)$ it may never be given. We nonetheless must make sure that $\Gamma(D, n)\downarrow$ (so that $\Gamma(D)$ is total) even if the guess using $n$ is wrong. In the current construction there is another reason to use simple permitting. We do not know whether $\varphi_d$ is total or not. This means that we need to set the uses $\psi_s(x)$ for $x \in X$ immediately when we appoint these followers. Before we attack we need to lift these uses beyond $\varphi_d(y)$ for $y \in X$, and these values are revealed to us after we already appoint the followers and define the $\Psi$-computations. So we wait for a "free pass" to raise these markers, and this will be given as usual by assuming that $D$ is noncomputable.

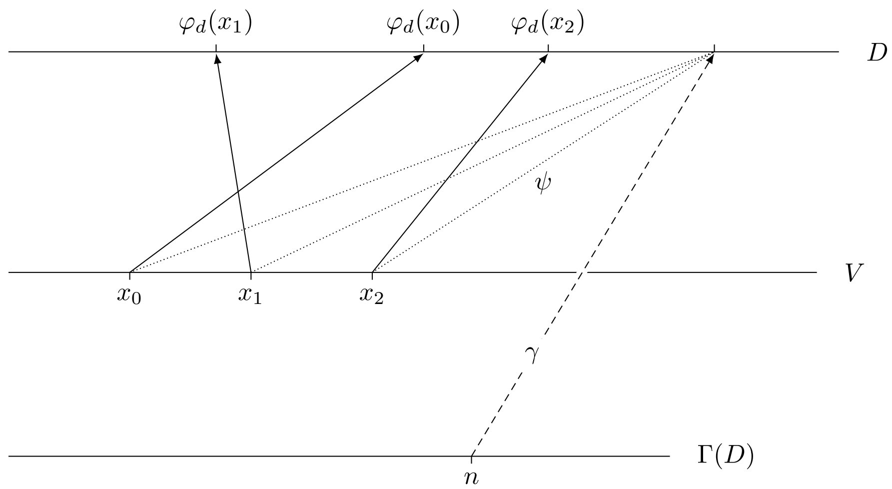


FIGURE 1. $\omega$-c.a. degrees are not $m$-topped

**1.1. Construction.** We are given a c.e. set $D$ whose Turing degree is totally $\omega$-c.a. We use a list $\langle g^e \rangle$ of all $\omega$-c.a. functions, with tidy $(\omega + 1)$-computable approximations $\langle g^e_s, o^e_s \rangle$. We enumerate a Turing functional $\Gamma$ with intended oracle $D$.

For every $e < \omega$ we perform an $e^{\text{th}}$ construction. These constructions are independent of each other, except that as usual they together define the functional $\Gamma$. For every $d < \omega$ the $e^{\text{th}}$ construction will employ an agent $d$. The action of distinct agents is independent of each other; we only need to ensure that they don't share followers.

The $e^{\text{th}}$ construction will enumerate a c.e. set $V^e$. It also defines a Turing functional $\Psi^e$ with the aim of having $\Psi^e(D) = V^e$.

An agent $d$ for construction $e$ aims to define a finite set $X$ of followers. The sets of followers for distinct agents are pairwise disjoint. The agent will choose an *anchor* $n$ (distinct from the numbers chosen by any other agent for any construction). The agent will be responsible for defining $\Gamma(D, n)$ and for defining $\Psi^e_s(D_s, x)$ for $x \in X$. The use $\psi^e_s(x)$ for all $x \in X$ will be the same, namely $\gamma_s(n)$.

We note that the agent must ensure that $n \in \operatorname{dom}\Gamma_s(D_s)$ at every stage $s$ (and that the uses $\gamma_s(n)$ are bounded). However $\Psi^e(D) = V^e$ is required only if the hypothesis that $\Gamma(D) = g^e$ is correct. The agent is thus allowed to leave computations $\Psi^e_s(D_s, x)$ undefined until it gets further evidence that the hypothesis holds.

In this chapter we simplify our notation as follows.

NOTATION 1.1. The intended oracle for the functionals $\Gamma$ and $\Psi^e$ is $D$; At stage $s$ we only define computations $\Gamma_s(D_s, n)$ and $\Psi^e_s(D_s, x)$. Further, the value of these computations is also fixed: at stage $s$, the value of a new $\Gamma_s(D_s, n)$ computation is always $s$, and the value of a new $\Psi^e_s(D_s, x)$ computation is $V^e_s(x)$. Thus to specify a computation all we need to provide is the use $\gamma_s(n)$ or $\psi^e_s(x)$. Instead of mentioning the functionals we only mention the uses (which can be thought of as moving markers). So for example we write $\psi^e_s(x)\downarrow$ if $\Psi^e_s(D_s, x)\downarrow$, and when a new computation is defined, we simply say that we define $\psi^e_s(x)$.

As mentioned above, before we can use any followers to diagonalise against many-one reductions we need them to be simply permitted by $D$. Thus before commencing the attacks, the agent will define distinct sets of followers $X_0, X_1, \dots$ associated with anchors $n_0, n_1, \dots$, one of which we hope will become the $X$ and $n$ we eventually use.

To carry out the construction we need the following, which we will verify after we specify the construction.

LEMMA 1.2. *Suppose that at some stage $s$, an agent $d$ for the $e^{\text{th}}$ construction is attacking with a set of followers $X$. Then $X \nsubseteq V^e_s$.*

*The action of agent $d$ for the $e^{\text{th}}$ construction.* We now describe two cycles (subroutines) detailing the action of an agent $d$ for the $e^{\text{th}}$ construction. The agent starts with set-up cycles; if some set of followers is set-up and permitted then the agent moves to attack cycles. During either cycle the agent is instructed to wait for some event. It is possible that the event does not happen, in which case the agent will wait for ever and not act again, other than maintaining the convergence of some functionals. In fact we will show that either we get an easy win, or the agent will get stuck waiting indefinitely from some point onwards, either because $g^e$ is not the correct guess, $\varphi_d$ is not total, or because some attack succeeds.

The agent starts with setting up the first set of followers.

SETTING UP THE $k^{\text{th}}$ SET OF FOLLOWERS.

> **1.** Let $s_0$ be the stage at which this set-up cycle begins. Choose a large anchor $n_k$. Define $\gamma_{s_0}(n_k) = n_k$.
>
> **2.** We wait for a stage $s_1$ at which $o^e_{s_1}(n_k) < \omega$. At that stage we choose a set $X_k$ of $(o^e_{s_1}(n_k) + 2)$-many large followers. For each $x \in X_k$ we define $\psi^e_{s_1}(x) = n_k$.
>
> **3.** We wait for a stage $s_2 > s_1$ at which $\varphi_{d,s_2}(x)\downarrow$ for all $x \in X_k$.
>
> **4.** We then wait for a stage $s_3 > s_2$ at which $D_{s_3} \restriction_{n_k} \neq D_{s_3-1} \restriction_{n_k}$.
>
> While waiting we (recursively) set up the $(k+1)^{\text{th}}$ set of followers.
>
> When such a stage $s_3$ is found, we interrupt all set-up cycles. We discard all anchors $n_{k'}$ and sets of followers $X_{k'}$ for $k' \neq k$. We let $X = X_k$ and $n = n_k$. We let $u = 1+\max\{\varphi_d(x) : x \in X\}$. We start an attack with some $x \in X$.

Throughout the set-up phase, if some anchor $n_k$ is already chosen and $D_s \restriction_{n_k} \neq D_{s-1} \restriction_{n_k}$ then unless we start an attack at stage $s$, we redefine $\gamma_s(n_k) = n_k$ and if also $X_k$ is defined, $\psi^e_s(z) = n_k$ for all $z \in X_k$.

If we start an attack at some stage $t$ then we will ensure that $\Gamma_{t-1}(D_t, n)\uparrow$ and that $\Psi^e_{t-1}(D_t, z)\uparrow$ for all $z \in X$.

Attacking with a follower $x$.

> **1.** Let $t_0$ be the stage at which the attack begins. We define a new $\Gamma$ computation by setting $\gamma_{t_0}(n) = u$.
>
> **2.** We wait for a stage $t_1 > t_0$ at which $g^e_{t_1}(n) = \Gamma_{t_1}(D_{t_1}, n)$. While waiting, the markers $\psi^e_s(z)$ for all $z \in X$ remain undefined.
>
> If $\varphi_d(x) \in D_{t_1}$ then we interrupt the attack cycle and discard both $n$ and $X$; all action for the agent ceases. In this case we get an easy win by keeping $x$ out of $V^e$.
>
> Otherwise, we enumerate $x$ into $V^e_{t_1}$; we define $\psi^e_{t_1}(z) = u$ for all $z \in X$.
>
> **3.** We wait for a stage $t_2 > t_1$ at which $\varphi_d(x) \in D_{t_2}$. At that stage we end the current attack and commence a new attack with some $x' \in X \backslash V^e_{t_2}$.

Throughout the attack phase, if $D_s \restriction_u \neq D_{s-1} \restriction_u$ and we do not start a new attack at stage $s$ then we define $\gamma_s(n) = u$, and if further $\psi^e_{s-1}(z)\downarrow$ for $z \in X$ (i.e. if $s > t_1$) then we define $\psi^e_s(z) = u$.

Globally, if $n < s$ and $n$ is at stage $s$ not used as anchor by any agent for any construction (either it was never chosen, or was chosen and later discarded) then we define $\gamma(n) = 0$. For all $e < s$, if $x < s$ and $x$ is not at stage $s$ used as a follower by any agent for the $e^{\text{th}}$ construction then we define $\psi^e(x) = 0$.

**1.2. Verification.** We first need to show that the construction can be performed as described. Fix an agent $d$ for the $e^{\text{th}}$ construction.

Let $t$ be a stage at which an attack cycle begins. We need to show that $\Gamma_{t-1}(D_t, n)\uparrow$ and that $\Psi^e_{t-1}(D_t, z)\uparrow$ for all $z \in X$. Suppose that the set-up phase ended at stage $t$. Then $D_t \upharpoonright_n \neq D_{t-1} \upharpoonright_n$ and $n$ equals both $\gamma_{t-1}(n)$ and $\psi^e_{t-1}(z)$ for $z \in X$. If on the other hand an attack cycle (with some follower $x$) ends at stage $t$ then $\varphi_d(x) \in D_t \backslash D_{t-1}$ and $\varphi_d(x) < u$, and $u$ equals both $\gamma_{t-1}(n)$ and $\psi^e_{t-1}(z)$ for $z \in X$.

Proof of Lemma 1.2. During each attack cycle at most one follower is enumerated into $D$. Let $t < s$ be two stages at which an attack cycle begins. Since $g^e_r(n) = \Gamma_t(D_r, n) \geqslant t$ at some stage $r \in (t, s)$ and by convention $g^e_t(n) < t$ we see that $o^e_s(n) < o^e_t(n)$. It follows that at most $o_{s_0}(n) + 1$ many attack cycles are started, where $s_0$ is the stage at which $X$ is appointed. The lemma follows from the choice $|X| = o_{s_0}(n) + 2$. □

We also observe that $\Gamma(D)$ is total. For let $n < \omega$. If $n$ is not chosen as an anchor by any agent for any construction, or is chosen but is later discarded, then we arranged that $n \in \operatorname{dom}\Gamma(D)$ (with use 0). Otherwise $n = n_k$ for some unique agent for a unique construction. If the agent never enters the attack phase then $\gamma_s(n)$ is defined at every stage after $n$ is chosen, always with use $n$, and so eventually a correct computation is defined. If the agent enters the attack phase with $n$ then at every stage $s$ during this phase the computation $\gamma_s(n)$ is defined, with use $u$; so again a correct computation is eventually defined.

We fix some $e$ such that $\Gamma(D) = g^e$ and $\langle f^e_s, o^e_s\rangle$ is eventually $\omega$-computable. We will show that the $e^{\text{th}}$ construction succeeds. We drop all supserscripts $e$ from now on.

Lemma 1.3. $\Psi(D) = V$.

Proof. Let $x < \omega$. If $x$ is enumerated into $V$ at some stage $t$ then $\Psi_{t-1}(D_t, x)\uparrow$ and a computation with a correct value is defined at stage $s$. So it suffices to show that $x \in \operatorname{dom}\Psi(D)$.

If $x$ is never chosen as a follower by any agent for the $e^{\text{th}}$ construction, or if it is chosen and later discarded, then we arrange that $\Psi(D, x)\downarrow$ with use 0. Suppose that $x$ is chosen by some agent $d$ and is never discarded.

During the set-up phase we ensure that $\psi_s(x)\downarrow$ at every stage after the stage at which $x$ was appointed, with use $n_k$ (if $x \in X_k$). As with $\Gamma(D)$, if the attack phase never begins then this ensures that $n \in \operatorname{dom}\Psi(D)$.

Suppose that the attack phase eventually begins and that $x \in X$. Suppose that $s$ is a stage during the attack phase and that $\psi_s(x)\uparrow$. Let $t \leqslant s$ be the stage at which the attack cycle began which is running at stage $s$. At stage $s$ we are still waiting to see $g_r(n) = \Gamma_r(D_r, n)$. Since we assume that $x$ is never discarded, the attack phase is never interrupted. Since $g = \Gamma(D)$ we see that a stage $r$ as required will occur, and at that stage we will define $\psi_r(x) = u$. Again we see that eventually a correct computation will be defined. □

Lemma 1.4. $V \nleqslant_m D$.

Proof. Suppose that $\varphi_d$ is total; we show that there is some $x$ such that $x \in V \Leftrightarrow \varphi_d(x) \notin D$.

We claim that agent $d$ will enter the attack phase. For otherwise, the fact that $\varphi_d$ is total and that $\langle f^e_s, o^e_s \rangle$ is eventually $\omega$-computable ensures that anchors $n_k$ are defined for every $k < \omega$. But then we compute $D$: if $X_k$ is appointed and $X_k \subseteq \operatorname{dom} \varphi_d$ at stage $s$, then $D_s \restriction_{n_k}$ is correct.

We have argued that only finitely many attack cycles are started by the agent. Let $x$ be the last follower with which we start an attack. If the attack is interrupted then $\varphi_d(x) \in D$ but we keep $x \notin V$. Otherwise, as argued above, we eventually enumerate $x$ into $D$. Since no new attack is ever started, $\varphi_d(x) \notin D$. □

## 2. Totally $\omega^2$-c.a. degrees are not $m$-topped

**2.1. An easy proof.** Consider how the construction in the previous section needs to change if $\deg_{\mathrm{T}}(D)$ is totally $\omega^2$-c.a. In this case the ordinal $o_{s_0}(n)$ that we discover is not a natural number $m$ but an ordinal of the form $\omega \cdot m_0 + m_1$, where $m_0$ and $m_1$ are natural numbers.

The most natural adaptation is the following. When the ordinal $\omega \cdot m + k$ is revealed, we appoint a set $X$ of followers of size $k+1$. We wait for $\varphi_d$ to converge on the followers in $X$ and then for permission to lift the uses $\gamma_s(n) = \psi^e_s(z)$ (for $z \in X$) above the values of $\varphi_d(z)$ for $z \in X$. When permission is granted we attack as above; but it is possible that eventually we exhaust all the followers in $X$. But when that happens, since $|X| > k$, the ordinal we see when $X$ is exhausted is $\omega \cdot m' + k'$, with $m' < m$: we dropped below the limit ordinal $\omega \cdot m$. We then would like to repeat the process: appoint a new set $X'$ of followers of size $k' + 1$; wait for $\varphi_d$ to converge on $X'$, and then for permission to lift $\gamma_s(n) = \psi^e_s(z)$ above the values of $\varphi_d$; and then attack again. We can go through at most $m$ many cycles of cycles of attacks, and so eventually the opponent will not be able to respond.

The only question is why we would get enough permissions. Simple permitting is insuficient here; we need multiple permitting for each attempt to meet the requirement. But it is hopefully clear that the kind of permitting which we need to carry this plan out is non-total $\omega$-c.a. permitting. That is, if we assume that $\deg_{\mathrm{T}}(D)$ is totally $\omega^2$-c.a. but not totally $\omega$-c.a. then this argument will actually work. If $\deg_{\mathrm{T}}(D)$ does happen to be totally $\omega$-c.a. then we just refer to the construction in the previous section.

We can also see how to generalise this argument to show to $n > 2$, to show that every c.e. degree which is totally $\omega^n$-c.a. is not $m$-topped. This approach however does not seem to work when we consider degrees which are totally $< \omega^\omega$-c.a. but not totally $\omega^n$-c.a. for any $n$ (see Theorem III.4.2). In that argument we define a single function $\Gamma(D)$ and guess some $n$ such that $\Gamma(D)$ is $\omega^n$-c.a.; and guess an appropriate approximation. However the for the permitting part of the argument we cannot just guess some function $\Theta(D)$ which is not $\omega^{n-1}$-c.a.: the point is that to set $\gamma(m)$ in the first place we need $\theta(k)$ where $k$ is the associated permitting number; if $\Theta(D,k)$ never converges then we will fail to make $\Gamma(D)$ total.

We thus give even for the case $n = 2$ a more complicated argument which we will be able to generalise to give the full result.

*Nonuniformity.* Rather than hope for a voluntary $D$-change, we manufacture it by using more than one set. Returning to the $\omega^2$ case, suppose that we enumerate two c.e. sets $V$ and $W$. It suffices to ensure for every pair $(c,d)$ of indices that either $\varphi_d$ is not a many-one reduction of $V$ to $D$ or $\varphi_c$ is not a many-one reduction of $W$ to $D$. The rough idea is to use two sets of followers $Y$ and $X$. We associate an anchor $n$ with the requirement; if we guess that $\Gamma(D,n)$ will not change more than $\omega \cdot m + k$ many times then we set $|Y| > m$ and $|X| > k$. We attack with the followers $x \in X$ against $\varphi_d$ (and so enumerate them into $V$). When $X$ runs out, as discussed above, the new ordinal is smaller than $\omega \cdot m$; we then attack with one follower $y \in Y$ against $\varphi_c$ (and so aim to enumerate it into $W$). Before the attack with $y$ commences we appoint a new set of followers to take the role of the new $X$, sufficiently large to last until we drop below the next limit ordinal. We wait for realisation of the new followers and then attack with $y$. The failure of this attack will give us the $D$-change that allows us to lift the new $\Gamma(D)$-use (and $\Psi^e(D)$-use for computing $V$ from $D$) beyond $\varphi_d(x)$ for all $x$ in the new $X$.

While we wait for the realisation of the new followers we must leave open the reduction of $W$ to $D$ (in the same way that in the $\omega$-construction, while we wait for a new agreement between $g^e$ and $\Gamma(D)$ to appear we leave the reduction of $V$ to $D$ open). This means that the totality of the reduction of $W$ to $D$ must rely on the totality of $\varphi_d$. We thus enumerate not a single set $W$ but infinitely many, one for each $\varphi_d$, and we rename the sets $V_d$. Assume that the guess $g^e$ is correct. Then we will in any case ensure that $V \leqslant_{\mathrm{T}} D$; and if $\varphi_d$ is a many-one reduction of $V$ to $D$, then we will ensure that $V_d \leqslant_{\mathrm{T}} D$ and that it is not many-one reducible to $D$. See Figure 2.

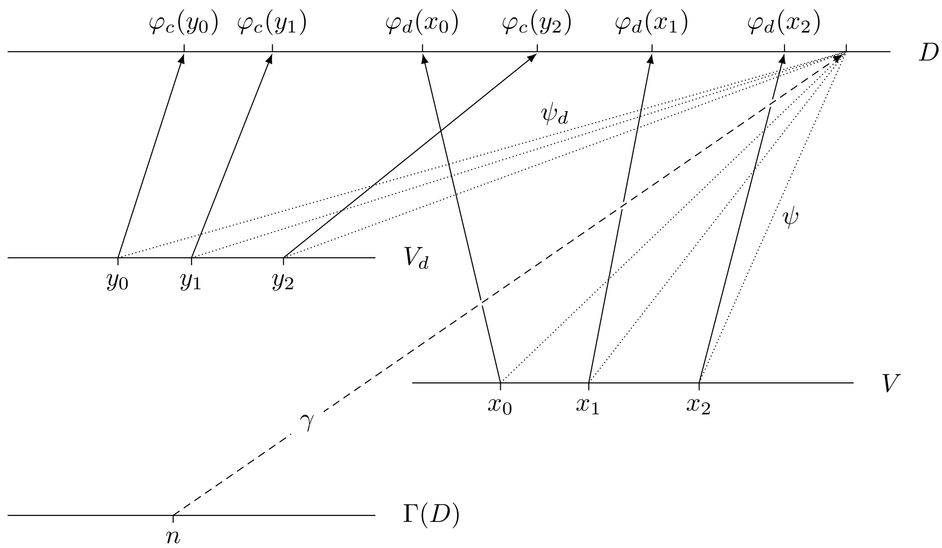


FIGURE 2. $\omega^2$-c.a. degrees are not $m$-topped

**2.2. Construction.** We are given a c.e. set $D$ whose Turing degree is totally $\omega^2$-c.a. We use a list $\langle g^e \rangle$ of all $\omega^2$-c.a. functions, with tidy $(\omega^2+1)$-computable approximations $\langle g^e_s, o^e_s \rangle$. We enumerate a Turing functional $\Gamma$ with intended oracle $D$.

For every $e < \omega$ we perform an $e^{\text{th}}$ construction. As above, these constructions are independent of each other. For every pair $(d, c)$ of natural numbers, the $e^{\text{th}}$ construction will employ an agent $(d, c)$. The action of distinct agents is independent of each other; we only need to ensure that they don't share followers.

The $e^{\text{th}}$ construction will enumerate a c.e. set $V^e$, and for all $d < \omega$, a c.e. set $V^e_d$. It also defines a Turing functional $\Psi^e$ with the aim of having $\Psi^e(D) = V^e$, and Turing functionals $\Psi^e_d$ with the aim of having $\Psi^e_d(D) = V^e_d$. We continue to follow Notation 1.1 and mostly refer to the uses of these computations.

As disccused, an agent $(d, c)$ for the $e^{\text{th}}$ construction plans to set up sets of followers $Y$ and $X$. Once it enters the attack phase, the set $Y$ is fixed, but the set $X$ is not: once the followers in $X$ are exhausted, we attack with another follower from $Y$ and appoint a new set of followers to play the role of $X$. While it is not precise, during the construction we refer to the currect version of $X$ simply by "$X$" rather than give it an index. During the verification we may refer to the version of $X$ at stage $s$ by $X_s$.

During the set-up phase we appoint a sequence $Y_1, Y_2, \dots$ of sets, one of which may be chosen to be the set $Y$ we use for attack.

*The action of agent $(d, c)$ for the $e^{\text{th}}$ construction.* The agent starts with setting up the first set $Y_1$.

Setting up $Y_k$.

**1.** Let $s_0$ be the stage at which this set-up cycle begins. We choose a large anchor $n_k$. Define $\gamma_{s_0}(n_k) = n_k$.

**2.** We wait for a stage $s_1$ at which $o^e_{s_1}(n_k) < \omega^2$. Suppose that $o^e_{s_1}(n_k) = \omega \cdot m + p$. At stage $s_1$ we choose a set $Y_k$ of $(m+2)$-many large followers. For each $y \in Y_k$ we define $\psi^e_{d,s_1}(y) = n_k$.

**3.** We wait for a stage $s_2 > s_1$ at which $\varphi_{c,s_2}(y)\downarrow$ for all $y \in Y_k$.

**4.** We then wait for a stage $s_3 > s_2$ at which $D_{s_3}\restriction_{n_k} \neq D_{s_3-1}\restriction_{n_k}$. While waiting we (recursively) set up the set $Y_{k+1}$.

When such a stage $s_3$ is found, we interrupt all set-up cycles. We discard all anchors $n_{k'}$ and sets of followers $Y_{k'}$ for $k' \neq k$. We let $Y = Y_k$ and $n = n_k$. We let $u = 1 + \max\{\varphi_c(y) : y \in Y\}$. We start an attack with some $y \in Y$.

Throughout the set-up phase, if some anchor $n_k$ is already chosen and $D_s\restriction_{n_k} \neq D_{s-1}\restriction_{n_k}$ then unless we start an attack at stage $s$ we redefine $\gamma_s(n_k) = n_k$ and if also $Y_k$ is defined, $\psi^e_{d,s}(y) = n_k$.

Attacking with a follower $y \in Y$.

**1.** Let $r_0$ be the stage at which the attack begins. We define $\gamma_{r_0}(n) = u$. We appoint a set $X$ of $(p+2)$-many large followers, where $o^e_{r_0}(n) = \omega \cdot m + p$. For each $x \in X$ we define $\psi^e_{r_0}(x) = u$. For now, we leave $\psi^e_{d,s}(y')$ for $y' \in Y$ undefined.

**2.** We wait for a stage $r_1 > r_0$ at which $\varphi_{d,r_1}(x)\downarrow$ for all $x \in X$. If $\varphi_c(y) \in D_{r_1}$ then we interrupt the attack cycle, discard all associated followers and anchor, and cease all action for the agent.

Otherwise we enumerate $y$ into $V^e_{d,r_1}$. For all $y' \in Y$ we define $\psi^e_{d,r_1}(y') = u$.

**3.** We wait for a stage $r_2 > r_1$ at which $\varphi_c(y) \in D_{r_2}$. At that stage we end the current attack and commence an attack with some $x \in X$; we let $v = 1 + \max\{\varphi_d(x) : x \in X\}$.

Throughout this attack phase, if $D_s \upharpoonright_u \neq D_{s-1} \upharpoonright_u$ and we do not start an attack with some $x \in X$ at stage $s$, then we redefine $\gamma_s(n) = u$ with use $u$; and we redefine $\psi^e_s(x) = u$ for $x \in X$. If $s > r_1$ then we also define $\psi^e_{d,s}(y') = u$ for all $y' \in Y$.

Attacking with a follower $x \in X$.

**1.** Let $t_0$ be the stage at which the attack begins. We define $\gamma_{t_0}(n) = v$.

**2.** We wait for a stage $t_1 > t_0$ at which $g^e_{t_1}(n) = \Gamma_{t_1}(D_{t_1}, n)$. While waiting, we leave $\psi^e_s(x')$ for $x' \in X$ undefined. Note that $\psi^e_{d,s}(y)$ for $y \in Y$ will be undefined throughout the attack with $x$.

If $\varphi_d(x) \in D_{t_1}$ then we interrupt the attack cycle, discard all associated followers and anchor, and cease all action for the agent.

Otherwise, we enumerate $x$ into $V^e_{t_1}$. We define $\psi^e_{t_1}(x') = v$ for all $x' \in X$.

**3.** We wait for a stage $t_2 > t_1$ at which $\varphi_d(x) \in D_{t_2}$.

If $X \subseteq V^e_{t_2}$ then we discard $X$ and start a new attack with some $y \in Y \backslash V^e_{d,t_2}$. Otherwise we commence a new attack with some $x' \in X \backslash V^e_{t_2}$.

The functionals $\Gamma(D, n)$ and $\Psi^e(D, x')$ are maintained as above.

Also as in the $\omega$ case, we ensure totality of functionals by defining them with use 0 on all inputs which are not used as anchors or followers.

**2.3. Verification.** We need to show that the construction can be preformed as described. Fix an agent $(d, c)$ for the $e^{\text{th}}$ construction.

First we observe that if an attack cycle begins at some stage $w$ then all functionals are divergent at that stage. Namely:

- If an attack with $y \in Y$ begins at stage $w = r_0$ then $\Gamma_{w-1}(D_w, n)\uparrow$, and $\Psi^e_{d,w-1}(D_w, y')\uparrow$ for all $y' \in Y$; and
- If an attack with $x \in X$ begins at stage $w = t_0$ then also $\Psi^e_{w-1}(D_w, x')\uparrow$ for all $x' \in X$.

But as above these are ensured by the $D$-change encountered at the last stage of the previous cycle. If the set-up phase ended at stage $w$, then we just saw a change on $D \upharpoonright_n$, and all uses are $n$; at the end of an attack with $y \in Y$, we just saw a change on $D \upharpoonright_u$, and all uses are $u$; at the end of an attack with $x \in X$, we just saw a change on $D \upharpoonright_v$, and the uses $\gamma(n)$ and $\psi^e(x')$ are $v$, while $\psi^e_d(y)$ are undefined throughout the attack with $x$.

We also obtain an analogue of Lemma 1.2: if $Y$ is already defined at stage $w$ then $Y \nsubseteq V^e_{d,w}$. Suppose that the set $Y$ is chosen at some stage $s_1$, with $o^e_{s_1}(n) = \omega \cdot m^* + p$, so $|Y| = m^* + 2$. We argue that an attack with some follower in $Y$ is started at most $m^* + 1$ many times. For $t < \omega$ let $o^e_t(n) = \omega \cdot m_t + p_t$. We claim that if two attacks with followers in $Y$ start at stages $s < t$ then $m_t < m_s$. This in turn is done by examining attacks started with elements of $X$. We have $|X_s| = p_s + 2$. The argument in the $\omega$-case shows that if $w < r$ are stages in $(s, t)$ at which we start an attack with an element of $X_s$ then $o^e_r(n) < o^e_w(n)$. The fact that $p_s + 2$ many such attacks occur implies that $m_t < m_s$ as required.

Next we observe that $\Gamma(D)$ is total. The argument is similar to the one in the $\omega$-case. Suppose that $n$ is an anchor for some agent, and is never discarded. A computation $\Gamma_s(D_s, n)$ is defined at every stage $s > n$. The use is bounded. There are three possibilities. An attack may never begin; in this case $\gamma_s(n) = n$ for all $s$. Alternatively, an attack with some $y \in Y$ is never ended; we then eventually have $\gamma_s(n) = u$. Finally it is possible that an attack with some $x \in X$ for some version of $X$ is never ended; we then eventually have $\gamma_s(n) = v$ (and $v$ is never redefined).

We fix some $e$ such that $\Gamma(D) = g^e$ and $\langle g^e_s, o^e_s \rangle$ is eventually $\omega^2$-computable. We will show that the $e^{\text{th}}$ construction succeeds. We drop all supserscripts $e$ from now on.

The argument proving Lemma 1.3 shows that $\Psi(D) = V$. If $V \not\leqslant_m D$ then we are done. Assume this fails; fix some total computable function $\varphi_d$ such that $\varphi_d^{-1}[D] = V$.

We argue that $\Psi_d(D) = V_d$. Observing that we only enumerate $y \in Y$ into $V_d$ at stages at which $\Psi_d(D, y)$ diverges, again it suffices to show that $\Psi_d(D)$ is total. We focus on some $y$ which is a follower in some set of followers $Y$ for some agent $(d, c)$. If no attack by the agent is every started (it is always in the set-up phase) then $\psi_d(y)\downarrow$ at every stage after $y$ is appointed, with a bounded use $n_k$. Otherwise, the key is that since $\varphi_d^{-1}[D] = V$, every attack by this agent with a follower $x \in X$ must end. So there is an attack with some $y' \in Y$ by the agent which never ends. However the assumption that $\varphi_d^{-1}[D] = V$ implies that the attack is not stuck waiting for a stage $r_1$; $\varphi_d$ is total. So we are eventually stuck waiting for a stage $r_2$; while waiting, we keep defining $\psi_d(y) = u$.

Finally, the argument of Lemma 1.4 shows that $V_d \not\leqslant_m D$. Fix some total $\varphi_c$. The simple permitting argument shows that the agent $(d, c)$ will enter the attack phase; we just observed that an attack with some $y \in Y$ must succeed.

## 3. Totally $< \omega^\omega$-c.a. degrees are not $m$-topped

The general case follows the structure of the $\omega^2$ case. Each construction guesses the $m$ such that $\Gamma(D)$ is $\omega^m$-c.a., and an appropriate approximation. It builds sets in $m$ layers of nonuniformity.

**3.1. Construction.** We are given a c.e. set $D$ whose Turing degree is totally $< \omega^\omega$-c.a. We use uniform lists $\langle g^{e,m} \rangle$ of all $\omega^m$-c.a. functions, with tidy $(\omega^m + 1)$-computable approximations $\langle g^{e,m}_s, o^{e,m}_s \rangle$, for all $m < \omega$. We enumerate a Turing functional $\Gamma$ with intended oracle $D$.

For every pair $(e, m)$ we perform an $(e, m)$-construction. These constructions are independent of each other. For every $m$-tuple $\bar{d} = (d_0, \ldots, d_{m-1})$, the $(e, m)$-construction will employ an agent $\bar{d}$. The construction enumerates c.e. sets $V^{e,m}_{\bar{c}}$ for all tuples $\bar{c}$ of numbers of length strictly smaller than $m$. For each such sequence $\bar{c}$, the construction also enumerates a functional $\Psi^{e,m}_{\bar{c}}$, as usual with the aim of having

$\Psi^{e,m}_{\bar{c}}(D) = V^{e,m}_{\bar{c}}$, so as usual, to define a computation for one of these functionals, we only need to specify its use.

*The action of agent $\bar{d}$ for the construction $(e,m)$.* The agent aims to establish $m$ sets of followers $X_{m-1}, X_{m-2}, \dots, X_0$. The followers in $X_k$ are targeted for $V^{e,m}_{\bar{d}\restriction_k}$. After receiveing simple permission, the set $X_{m-1}$ is fixed but the sets $X_{m-2}$, $X_{m-3}, \dots$ are not fixed. When all followers in $X_{k-1}, X_{k-2}, \dots, X_0$ are used, we discard these sets and attack with a new follower from $X_k$.

Before we receive our simple permission though we need to appoint a sequence of candidates for $X_{m-1}$. These will be denoted by $Y_1, Y_2, \dots$.

The agent starts with setting up the first set $Y_1$.

Setting up $Y_i$.

**1.** Let $s_0$ be the stage at which this set-up cycle begins. We choose a large anchor $n_i$. Define $\gamma_{s_0}(n_i) = n_i$.

**2.** We wait for a stage $s_1$ at which $o^{e,m}_{s_1}(n_i) < \omega^m$. Suppose that $o^{e,m}_{s_1}(n_i) = \omega^{m-1} \cdot p + \beta$ (for some $\beta < \omega^{m-1}$). At stage $s_1$ we choose a set $Y_i$ of $(p+2)$-many large followers. For each $y \in Y_i$ we define $\psi^{e,m}_{\bar{d}\restriction_{m-1},s_1}(y) = n_i$.

**3.** We wait for a stage $s_2 > s_1$ at which $\varphi_{d_{m-1},s_2}(y)\downarrow$ for all $y \in Y_i$.

**4.** We then wait for a stage $s_3 > s_2$ at which $D_{s_3}\restriction_{n_i} \neq D_{s_3-1}\restriction_{n_i}$. While waiting we (recursively) set up the set $Y_{i+1}$.

When such a stage $s_3$ is found, we interrupt all set-up cycles. We discard all anchors $n_{i'}$ and sets of followers $Y_{i'}$ for $i' \neq i$. We let $X_{m-1} = Y_i$ and $n = n_i$. We start an attack with some $x \in X_{m-1}$.

Throughout the set-up phase, if some anchor $n_i$ is already chosen and $D_s\restriction_{n_i} \neq D_{s-1}\restriction_{n_i}$ then unless we start an attack at stage $s$ we defne $\gamma_s(n_i) = n_i$ and if also $Y_i$ is defined, $\psi^{e,m}_{\bar{d}\restriction_{m-1},s}(y) = n_i$ for $y \in Y_i$.

Throughout the attack phase we let

$$o^{e,m}_s(n) = \omega^{m-1}p_{m-1,s} + \omega^{m-2}p_{m-2,s} + \cdots + \omega \cdot p_{1,s} + p_{0,s}.$$

When we start an attack with some element of $X_k$ (for $k < m$) the sets $X_{m-1}, \dots, X_k$ are defined but $X_{k-1}, \dots, X_0$ are not. If $X_k$ is defined then so is $u_k = 1 + \max\left\{\varphi^{e,m}_{d_k}(x) : x \in X_k\right\}$. During an attack with some $x \in X_k$, the computations $\Psi^{e,m}_{\bar{d}\restriction_{k'}}(y)$ for all $k' > k$ and $y \in X_{k'}$ are undefined.

Attacking with a follower $x \in X_k$ for $k > 0$.

**1.** Let $r_0$ be the stage at which the attack begins. We define $\gamma_{r_0}(n) = u_k$. We appoint a set $X_{k-1}$ of $(p_{k-1,r_0}+2)$-many large followers. For each $z \in X_{k-1}$ we define $\psi^{e,m}_{\bar{d}\restriction_{k-1},r_0}(z) = u_k$. For now we leave $\psi^{e,m}_{\bar{d}\restriction_k,s}(x')$ for all $x' \in X_k$ undefined.

**2.** We wait for a stage $r_1 > r_0$ at which $\varphi_{d_{k-1},r_1}(z)\downarrow$ for all $z \in X_{k-1}$. If $\varphi_{d_k}(x) \in D_{r_1}$ then we interrupt the attack cycle, discard all associated followers and anchor, and cease all action for the agent.

Otherwise we enumerate $x$ into $V^{e,m}_{\bar{d}\upharpoonright_k, r_1}$. For all $x' \in X_k$ we define $\psi^{e,m}_{\bar{d}\upharpoonright_k, r_1}(x') = u_k$.

**3.** We wait for a stage $r_2 > r_1$ at which $\varphi_{d_k}(x) \in D_{r_2}$. At that stage we end the current attack and commence an attack with some $y \in X_{k-1}$.

As usual we respond to spontaneous $D\upharpoonright_{u_k}$-changes by rectifying existing computations with the same use $u_k$.

Attacking with a follower $x \in X_0$.

**1.** Let $t_0$ be the stage at which the attack begins. We define $\gamma_{t_0}(n) = u_0$.

**2.** We wait for a stage $t_1 > t_0$ at which $g^{e,m}_{t_1}(n) = \Gamma_{t_1}(D_{t_1}, n)$. While waiting, we leave $\psi^{e,m}_{\langle\rangle,s}(y)$ for $y \in X_0$ undefined. If $\varphi_{d_0}(x) \in D_{t_1}$ then we interrupt the attack cycle, discard all associated followers and anchor, and cease all action for the agent.

Otherwise we enumerate $x$ into $V^{e,m}_{\langle\rangle, t_1}$. For all $x' \in X_0$ we define $\psi^{e,m}_{\langle\rangle, t_1}(x') = u_0$.

**3.** We wait for a stage $t_2 > t_1$ at which $\varphi_{d_0}(x) \in D_{t_2}$. At that stage we end the current attack. Let $k \geqslant 0$ be the least such that $X_k \nsubseteq V^{e,m}_{\bar{d}\upharpoonright_k, t_2}$. Discard $X_{k'}$ (and so $u_{k'}$) for all $k' < k$. Start a new attack with some $y \in X_k \backslash V^{e,m}_{\bar{d}\upharpoonright_k, t_2}$.

As above, we maintain functionals, and define them on numbers that are not used by any construction.

**3.2. Verification.** These are similar to the previous verifications. First we need to ensure that the construction can be carried out as described. As above we observe that at the end of any cycle (set up or attack), all related computations are undefined. We also prove that if $X_{m-1}$ is defined at a stage $s$ (for some agent $\bar{d}$ for a construction $(e, m)$) then $X_{m-1} \nsubseteq V^{e,m}_{\bar{d}\upharpoonright_{m-1}}$. To see this, by induction on $k < m$ we observe that if $s < t$ are stages at which at attck with some $x \in X_k$ is started, then $o^{e,m}_s(n) - o^{e,m}_t(n) \geqslant \omega^k$.

The proof that $\Gamma(D)$ is total is as above. Fixing $(e, m)$ which is a correct guess ($\Gamma(D) = g^{e,m}$ and $\langle g^{e,m}_s, o^{e,m}_s \rangle$ is eventually $\omega^m$-computables), and dropping the superscripts $(e, m)$, we argue that the $(e, m)$ construction is successful. As above we argue that $\Psi(D) = V$. If $V$ is not as required, we fix some $d_0$ such that $\varphi^{-1}_{d_0}[D] = V$. Then for any agent $\bar{c}$ such that $c_0 = d_0$, no attack with some $x \in X_0$ can succeed. This shows that $\Psi_{d_0}(D) = V_{d_0}$. If $V_{d_0}$ is not as required then we fix some $d_1$ such that $\varphi^{-1}_{d_1}[D] = V_{d_0}$. Then for any agent $\bar{c}$ with $(c_0, c_1) = (d_0, d_1)$, no attack with some $x \in X_1$ can succeed. This shows that $\Psi_{d_0,d_1}(D) = V_{d_0,d_1}$. And so on… this process must end at some $k < m$, giving some $V_{d_0,d_1,\ldots,d_k}$ which shows that $\deg_{\mathrm{T}}(D)$ is not $m$-topped.

CHAPTER VII

# Embeddings of the 1-3-1 lattice

In this chapter we prove Theorem I.0.7: the 1-3-1 lattice is embeddable in the c.e. degrees below a c.e. degree $\mathbf{d}$ if and only if $\mathbf{d}$ is not totally $< \omega^\omega$-c.a.

## 1. Embedding the 1-3-1 lattice

We prove the first direction: if $\mathbf{d}$ is not totally $< \omega^\omega$-c.a. then the 1-3-1 lattice is embeddable below $\mathbf{d}$.

**1.1. Lachlan's construction.** To prove this we use the construction of Downey and Shore's [19] which shows that the 1-3-1 lattice can be embedded below any non-low$_2$ degree. This is an elaboration on Lachlan's original embedding of the 1-3-1 lattice into the c.e. degrees. We briefly recall a version of the construction given by Stob (unpublished notes), using Lerman's pinball machine technique [46]. This is one of the few infinite-injury constructions which does not benefit much from the use of a tree of strategies.

In this construction we enumerate three c.e. sets $A_0$, $A_1$ and $A_2$. To ensure that their degrees form the middle section of an embedding of the 1-3-1 lattice (with bottom $\mathbf{0}$) we need to ensure that they are noncomputable, any two form a minimal pair (which also implies that they must be Turing incomparable), and each is computable from the join of the other two. The requirements to meet are:

$P^i_e$**:** $A_i \neq \Phi_e$,

where $\langle \Phi_e \rangle$ is an enumeration of all partial computable functions; and for $i \neq j$ from $\{0, 1, 2\}$,

$N^{i,j}_e$**:** If $\Theta_e(A_i)$ and $\Psi_e(A_j)$ are total and equal, then they are computable;

here $\langle \Theta_e, \Psi_e \rangle$ is an enumeration of all pairs of Turing functionals.

The global requirement that $A_i \leqslant_{\mathrm{T}} A_j \oplus A_k$ when $\{i, j, k\} = \{0, 1, 2\}$ is met by the mechanism of appointing traces. A requirement $P^i_e$ will appoint a follower $x$, targeted for $A_i$, and wait for it to be realised, which means $\Phi_e(x)\downarrow = 0$; as usual, when the follower is realised the requirement will want to enumerate it into $A_i$. Before $x$ is realised, it is assigned a *trace* $y > x$, another number, which is targeted for either $A_j$ or $A_k$. This is essentially the current $A_j \oplus A_k$-use of computing $A_i(x)$. The main rule is that we cannot enumerate $x$ into $A_i$ before we enumerate $y$ into the set it is targeted for, $A_j$ or $A_k$. Sometimes we will be able to enumerate $y$ into the required set, but not be yet able to enumerate $x$ into $A_i$; in this case, we will appoint a new trace $y'$, again targeted for $A_j$ or $A_k$, but not necessarily to the same set for which $y$ was targeted. Indeed it is switching between $A_j$ and $A_k$ which is the key idea which makes the construction work.

Say that currently (at some stage $s$) $x$ has a trace $y$, targeted for $A_j$. Another global requirement is $A_j \leqslant_{\mathrm{T}} A_i \oplus A_k$. And so we need to repeat: the number $y$

receives a trace $z > y$ of its own, targeted for either $A_i$ or $A_k$. Overall, the follower $x$ is accompanied by an *entourage* of traces $y, z, \dots$, each element of the sequence being a trace for the number appearing before. At any stage, only numbers in a final segment of the entourage may be enumerated into the sets for which they are targeted. No two successive elements of the entourage are targeted for the same set. At stage $s$, the last element $w$ of the entourage is a number of size at least $s$, and so does not yet require a trace. At the end of the stage, if $w < s+1$ then we will assign it a new, large trace. The construction will specify the set for which the new trace will be targeted. For simplicity of expression, we abuse the term a little by letting the word *entourage* refer to the entire sequence $x, y, z, \dots$, including the follower.

All numbers we use in the construction as potential elements of the three sets $A_0$, $A_1$ and $A_2$ are represented as balls which will move in a pinball machine (see Figure 1). The main components of the machine are gates and holes. Some balls drop through holes to the main track of the machine. The balls move downwards. Along their journey they encounter gates. A gate may allow a ball to pass, or stops its movement. In the latter case, the ball is placed in a *corral* associated with the gate. Balls in the corral may later be released and allowed to resume their journey. When a ball arrives at the bottom of the machine we imagine that it lands in one of the pockets associated with one of the sets $A_i$, namely the set the ball is targeted for. When a ball marked with the number $x$ lands in the pocket associated with $A_i$, the number $x$ is enumerated into the set $A_i$.

Holes $H_0, H_1, H_2, \dots$ are associated with positive requirements $P_e^i$ (much like strategies on a tree are assigned to requirements). As described above, the requirement appoints a follower $x = t_0$. While waiting for the follower to be realised, an entourage of traces $t_1, t_2, t_3, \dots$ is appended to $x$. Once the follower is realised, the entourage $t_0, t_1, \dots$ drops through the associated hole $H_n$ and starts moving down through the machine. The entourage may be stopped by one of the gates $G_m$ for $m \leqslant n$, in which case it enters the corral $C_m$. The last ball $y = t_\ell$ in the entourage rolls out of the corral and waits at the gate $G_m$. While waiting, the entourage is extended with more traces, all of which wait at the gate with $y$. At some point the gate opens and $y$ and its sequence of traces (the final segment of the current entourage waiting at the gate) continue their journey down the machine. This sequence of balls may be stopped at a lower gate $G_{m'}$ (so $m' < m$). All of the balls enter the corral $C_{m'}$ and the last element $z = t_{\ell'}$ rolls out to the gate. Again while waiting, new traces are added to the entourage beyond $z$. When the gate opens, $z$ and its traces continue their fall. Eventually some of these balls, in a final segment of the entourage, pass all of the gates and land in their pockets (with numbers enumerated into the sets they are targeted for). These balls are removed from the entourage. Say the final segment starting with $t_k$ has just landed in the pockets, and $k > 0$. The ball $t_{k-1}$ is now the last element of the entourage. It has been waiting in some corral $C_n$. It now rolls out to the gate $G_n$ and waits for the gate to open. While it is waiting, new traces $t_k, t_{k+1}, \dots$ are added to the entourage; they wait at the gate $G_n$ together with $t_{k-1}$. The process continues... in general the structure is as described in the following lemma.

LEMMA 1.1. *Let $x$ be a follower for some requirement $P_e^i$ associated with the hole $H_m$. Suppose that $x$ has already been dropped through its hole but has not yet been enumerated into $A_i$, so all balls in $x$'s entourage are currently lying at*

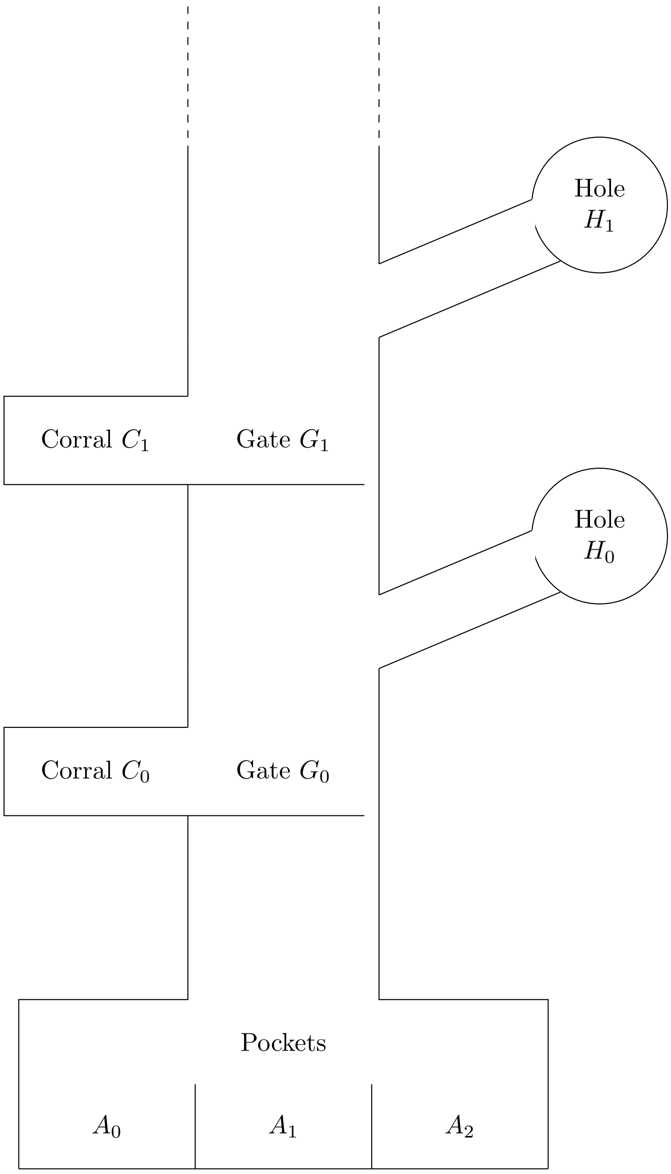


FIGURE 1. A pinball machine

*various corrals and gates below the hole. The entourage is partitioned into segments $I_m < I_{m-1} < \cdots < I_0 < I^*$, where each $I_k$ lies in the corral $C_k$ and $I^*$ waiting at some gate $G_n$. Some of the segment $I_k$ may be empty; indeed all segment $I_k$ for $k < n$ are empty. $I^*$ however is nonempty.*

We need to address two issues:

(1) we need to describe when gates open and when balls are stopped at some gate; and

(2) we need to explain why the follower will eventually be enumerated into its set.

We first explain (1). The gates $G_0$, $G_1$, $G_2, \dots$ are associated with negative requirements. Let $G_n$ be a gate and suppose that it is associated with the requirement $N_e^{i,j}$. The requirement is met by following Lachlan's minimal pair strategy of freezing a computation on one side or the other until it recovers on the other side. Suppose that $s$ is a stage and that $t < s$ was the previous stage at which the gate $G_n$ was open. Then $G_n$ opens at stage $s$ if the length of agreement between $\Theta_e(A_i)$ and $\Psi_e(A_j)$ exceeds $t$. That is, if for all $x \leqslant t$, $\Theta_e(A_i, x)\downarrow = \Psi_e(A_j, x)\downarrow$ $[s]$. When open, the gate $G_n$ cannot allow both balls targeted for $A_i$ and balls targeted for $A_j$ to drop below it. For this reason we need to ensure that if a final segment $I^*$ of an entourage is waiting at the gate $G_n$ at the beginning of stage $s$ then either no ball in $I^*$ is targeted for $A_i$ or no ball in $I^*$ is targeted for $A_j$. This is achieved by appointing new traces correctly: say that the first ball $z$ in $I^*$ rolled out to the gate $G_n$ from the corral $C_n$ at stage $r < s$. Suppose that $z$ is targeted for $A_i$. Then the next trace $w$ that we appoint for $z$ will be targeted not for $A_j$ but for $A_k$. And the next trace that we appoint for $w$ will be targeted for $A_i$; and so on, so no ball waiting at the gate at stage $s$ is targeted for $A_j$. The segment $I^*$ is sometimes called an $(i, k)$-*stream*. If $z$ is targeted for $A_j$ then we build $I^*$ to be a $(j, k)$-stream. Of course if $z$ is targeted for $A_k$ then we can build $I^*$ to be either a $(k, i)$-stream or a $(k, j)$-stream.

The whole process can be thought of as *retargeting* of traces. Say that the segment $I_n$ waiting in $C_n$ is an $(i, j)$-stream. Each ball in that segment waits until its trace, its successor in $I_n$, is enumerated into its set; we then appoint a new trace, targeted for $A_k$.

This brings us to question (2) above. We need to show that progress is made at every step. Let $x$ be a follower. On the face of it, it would appear that because we keep extending the entourage, it is possible that balls in $x$'s entourage move down at infinitely many stages (but $x$ itself is never enumerated). This is not so. Consider as a first example the case of one gate: suppose that $x = t_0$ and its entourage $I = (t_0, t_1, \dots, t_\ell)$ at stage $r$ arrive at the corral $C_0$ at that stage. The last ball $t_\ell$ in $I$ rolls out to the gate $G_0$. We keep appointing traces and extend the entourage beyond $t_\ell$, but when the gate opens, $t_\ell$ and all of these new balls fall to the pockets and are removed from the entourage. Next, the ball $t_{\ell-1}$ rolls out to the gate and the process resumes. We see that after $t_\ell + 1$ many iterations, the follower $x = t_0$ will be enumerated into the set it is targeted for, and the process will end.

Now consider two gates $G_0$ and $G_1$. At some stage $r_0$, $x$ and its entourage $I = (t_0, \dots, t_\ell)$ arrives to the corral $C_1$. The ball $t_\ell$ rolls out to the gate. While waiting the entourage is extended to $I^* = (t_\ell, t_{\ell+1}, \dots, t_{\ell+p})$. At some stage the gate $G_1$ opens, this segment is allowed to proceed, but is placed in the corral $C_0$. As discussed above, after $p - \ell + 1$ many times at which $G_0$ opens, $t_\ell$ will be enumerated into its set and the ball $t_{\ell-1}$ will roll out to the gate $G_1$. After $\ell + 1$ many iterations of this longer process, the follower lands in $C_0$, and we are back in the first case. This kind of nested analysis can be extended to any finite number of gates.

This argument can be coded succinctly using ordinals below $\omega^\omega$. Say $x$ is a follower, and let $I_m < I_{m-1} < \cdots < I_0 < I^*$ be a partition of its entourage as in Lemma 1.1. Consider the ordinal $\omega^m|I_m| + \omega^{m-1}|I_{m-1}| + \cdots + \omega^0|I_0| + \omega^n$, where $I^*$ lies at the gate $G_n$. The analysis above shows that each time a gate opens and part of $x$'s entourage moves, this ordinal decreases. The well-foundedness of $\omega^\omega$ guarantees that parts of $x$'s entourage move only finitely many times. In the next subsection we will see that this "ordinal analysis" corresponds to the kind of permitting which is required to get the argument to work below a given c.e. degree.

We also remind the reader of Theorem I.0.9, part of which relies on the fact that for most admissible ordinals $\alpha > \omega$, the 1-3-1 lattice cannot be embedded (at least with an incomplete top). The reason the argument fails is the instruction that the *last* ball of the entourage roll out to the gate. Since entourages may keep growing, it is perfectly possible that some will have order-type a limit ordinal. The only way to overcome this is if an $\alpha$-c.e. degree can compute a bijection between $\alpha$ and $\omega$. In that case the construction is essentially rearranged to resemble the standard $\omega$-construction, with finite entourages at every stage.

The main ideas of this construction have been described, but we mention a couple of aspects which we missed. In the analysis above we ignored the possibility that the last segment of an entourage is waiting at a gate which will never open again, because the hypothesis of the associated negative requirement fails. In this case the follower will not be enumerated into its set. For this reason, a positive requirement needs to appoint more followers and hope that one of them succeeds. We need to ensure that not all the followers will get stuck in this way. A good way to do this is to let each gate apprehend the entourage-segment of at most one follower. This is made possible by a process of cancellation. Followers are assigned priorities based on the time they were appointed. When a positive requirement receives attention (for example when appointing a new follower or when one of its followers receives attention), all followers for weaker requirements are cancelled. Thus the priority ordering between followers respects the ordering between requirements. When a follower receives attention (when its last entourage segment moves), all weaker followers, even for the same requirement, are cancelled. As usual, since new followers are appointed large, a follower $x$ is stronger than a follower $y$ if and only if $x < y$. Suppose that the last segment of $x$'s entourage is waiting at a currently closed gate $G_n$, and that the segment of $y$'s entourage is currently moving down. The gate can let $y$'s segment pass even though it is not currently open and even though $y$'s segment may contain balls targeted for both sets $A_i$ and $A_j$ that the gate cares about. The reason is the following. The fact that $x$'s segment is still waiting at the gate when $y$'s segment is moving (and so when $y$ receives attention) shows that $x$ is stronger than $y$; otherwise $x$ would be cancelled at this stage. The last stage $r$ at which $x$ received attention is no earlier than the last stage $u$ at which $G_n$ was open. The computations currently protected by the gate have been observed at stage $u$. At stage $r$, followers weaker than $x$ are cancelled, and so $y$ was appointed later than stage $u$. It is therefore much too large to disturb any of the computations that the gate is currently trying to protect, and it (or part of its entourage) can pass without let or hindrance. Overall, this shows that if a positive requirement is using the hole $H_n$, then at most $n + 1$ many of its followers could be permanently stuck at some gate. One of its followers will therefore either never get realised, or successfully enumerated into its set.

We remark that the necessity for appointing more than one follower could be avoided if we put the construction on a tree of strategies. The tree now acts as the track of the machine, with positive nodes acting as holes and others as gates. A positive node on the true path guesses correctly which gates will open infinitely often and so its follower cannot get permanently stuck. However, when we add permitting in the next section we will need to let positive requirements appoint many followers; even if they do not get stuck at gates, they can wait in vain for a permission. For the permitting argument it seems that adding a tree of strategies does not help simplify the construction.

**1.2. Embedding the 1-3-1 lattice with non-total $< \omega^\omega$-c.a. permitting.** Recall the argument above for why every follower $x$ receives attention only finitely many times. The ordinals used to show that the progress was well-founded correspond to the amount of permissions required to get the follower to its pocket. First note that for that argument, it is crucial that when part of $x$'s entourage lands in the pockets, that the numbers are actually enumerated into their sets. We cannot appoint a new trace for the last element of the entourage still waiting in a corral without first enumerating its current trace. Further, before a gate opens again, we need to ensure that the numbers that it allowed to pass at the last time it was open are actually enumerated into the sets. Otherwise it may let balls potentially injuring the other side pass, and then both sides may get injured before the next time the gate opens. So the number of permissions we need to get until $x$ is enumerated is close to the number of times the follower actually receives attention. The fact that $x$ receiving attention corresponds to a decrease in the ordinal shows us that a bound on the ordinal also bounds the number of permissions required. For the hole $H_{m-1}$ the bound is $\omega^m$.

This can be explained in detail looking at the simple cases. In the one-gate-case, once the follower is realised, we know the size of its entourage that enters the corral $C_0$, and so the number of times the gate $G_0$ needs to open until the follower arrives in its pocket. If the gate opens at some stage and releases one of the balls in the entourage, then we need a permission before the next such stage. So the number of permissions required is the same as the size of the entourage. This corresponds to non-total $\omega$-c.a. permitting. (It is not array noncomputable permitting because we need to wait until the follower is realised before we know the eventual size of the entourage that enters $C_0$; we cannot tell it in advance.) When two gates are involved, when the follower is realised we know how many times we need $G_1$ to open. Each time it does open (and not before) we find out how many $G_0$-openings, and so how may permissions, we need until the next $G_1$-opening. Even though we don't need a permission between $G_1$ opening and the first time after that when $G_0$ opens, the size of the entourage in $C_1$ does tell us how many times we need to update the bound on the number of permissions required. This is precisely non-total $\omega^2$-c.a. permitting. Weaker holes need to pass more and more gates, so overall for permitting we need a function which is not $\omega^n$-c.a for any $n < \omega$.

This analysis shows that to pass $m$ gates, *a single ball* requires $\omega^m$ permissions. However, the situation becomes more complicated when more than one ball is involved. As usual, a positive requirement will issue many followers, because some of them may get stuck at gates that don't open, and some of them will get stuck waiting for permissions. When one ball receives attention, weaker balls for the same requirement are cancelled. In many other $\alpha$-c.a. permitting constructions, if a follower $x$ cancels a follower $y$ then $x$ takes over the "permitting number" of $y$. That is, from that point on, every $y$-permission should be also counted as an $x$-permission. We cannot do this in this construction. The reason is that in order to increase $x$'s permission number we first need to actually receive $x$ permission (with the old number). Otherwise the whole process of requiring permissions does not help us show that the permitting degree bounds all the sets being constructed. However in the 1-3-1 construction below a nonlow$_2$ degree we cannot require permission during every movement of a follower; this is only possible with high permitting. (This has to do with the question of what happens when a gate opens but the corresponding follower is waiting for permission to move.) In a nonlow$_2$ or weaker construction we can only require permissions when attempting to enumerate numbers into sets. So whenever $x$ receives attention but does not try to enumerate numbers into sets, weaker followers $y$ will be cancelled, but their permitting numbers cannot be taken over by $x$.

Our solution is to abandon the technique of taking over permitting numbers. Essentially this means that if $y$ is a follower with permitting number $k$, and $y$ is cancelled, then the next follower $y'$ appointed gets the permitting number $k$ as well (technically this is not quite so, but for nonessential reasons). However the first ordinal we compute for $y'$ may be much larger than the ordinals we observed for $y$ while $y$ was still alive. When arguing that the positive requirement is met we need to threaten to give an $\omega^n$-computable approximation (for some $n$) for a function which doesn't have one. During this approximation we are not allowed to increase the ordinals. However we notice that $y$ was cancelled because a stronger follower $x$ received attention. This means that $x$'s ordinal count went down. Multiplying $x$'s ordinal by the bound $\omega^m$ (on the left) and adding to $y$'s ordinals we see that a single drop in $x$'s ordinal allows us to increase the $y$-ordinal to the $y'$-ordinal. Overall, to pass $m$ gates, we need $\omega^{2m}$-permission. The details are given in the proof of Lemma 1.5.

The permitted embedding cannot be done while preserving the least element. Our embedding will have a bottom degree $\mathbf{b} > \mathbf{0}$. This is similar to the non-low$_2$ construction of Downey and Shore's [19]. The reason is an aspect of the construction that we glossed over in the previous section. Let $G_n$ be a gate, working for requirement $N_e^{i,j}$, and suppose that the requirement's hypothesis holds: $\Theta_e(A_i) = \Psi_e(A_j)$. We need to show how to compute this common function. We look at a stage $s$ at which the gate opens; we need to argue that if no balls targeted for $A_j$ (say) drop from the gate at this stage, then the computation $\Psi_e(A_j)[s]$ up to the length of agreement will survive until the next stage $t$ at which the gate opens. This is not actually always true, the reason being that small balls targeted for $A_j$ are currently waiting at a gate $G_m$ below $G_n$ and may be enumerated between stages $s$ and $t$. We only certify the computation at stage $s$ if we know that no small balls targeted for $A_j$ that are at stage $s$ waiting at gates below $G_n$ will ever be enumerated into $A_j$. Note that some such balls may be stuck for ever at a gate below $G_n$. So $G_n$ cannot wait for a stage at which there are no small balls targeted for $A_j$ at any gate below. It only needs to ensure that such balls will not enter $A_j$. How can $G_n$ tell? Well, there are only finitely many gates below $G_n$, and each can have at most one segment as a permanent resident. The information which of the gates below has permanent residents can be given to $G_n$ non-uniformly, and we can wait for stages at which below $G_n$, only gates with permanent members are occupied. Again, a tree of strategies is equivalent to non-uniformly giving this advice to $G_n$; but as we will now see, this advice will be insufficient in the permitted construction.

In the permitted construction, many more balls can get stuck below $G_n$: those which passed all the gates, are lying in their pockets, but are still waiting for permission to be enumerated (the pockets act as a "permission bin"). Over all the construction, there will be infinitely many such balls. We need some uniform way to tell $G_n$ which of those are dangerous. For this reason we introduce the new c.e. set $B$. To ensure that $\deg_{\mathrm{T}}(A_0 \oplus B)$, $\deg_{\mathrm{T}}(A_1 \oplus B)$ and $\deg_{\mathrm{T}}(A_2 \oplus B)$ form the middle of an embedding of the 1-3-1 lattice with bottom $\deg_{\mathrm{T}}(B)$ we need to meet the modified requirements:

- **$P^i_e$:** $A_i \neq \Phi_e(B)$; and
- **$N^{i,j}_e$:** If $\Theta_e(A_i, B)$ and $\Psi_e(A_j, B)$ are total and equal, then they are computable from $B$.

The global requirements are now $A_i \leqslant_{\mathrm{T}} A_j \oplus A_k \oplus B$.

When an entourage segment lands into the pockets, we attach a new trace to the end of the entourage; this new trace is targeted for $B$. When permitted, the balls in that segment, together with the new trace, are enumerated into their sets. A gate $G_n$ now can look at the pockets and consulting $B$ can tell which entourage segments will be enumerated in the future into their sets, and so find whether a computation it is examining may be injured by balls waiting in the pockets.

Note that a number targeted for $B$ does not need a trace of its own. We may be tempted to close off entourages with a trace targeted for $B$ before they land in the pockets. We cannot appoint such a trace while the ball is waiting to be realised: since we are now diagonalising against $B$, a positive requirement will protect the $B$-computation which realises the follower; it can certainly not plan to enumerate a number into $B$ before it sees the use of such a computation. Suppose that the follower dropped through the hole, is moving down the machine, and its entourage has a final segment $I^*$ waiting at a gate. When the gate will open it will want to protect a computation on one side. However now both sides use $B$, so again, the gate cannot allow the appointing of a small number targeted for $B$ before it sees the use of these computations. Thus only an entourage segment which passed all the gates and is waiting in the pockets can appoint a trace targeted for $B$.

The reader may want to compare this construction with the permitted construction of a critical triple below a non-totally $\omega$-c.a. degree in [17]. In that construction the gates do not look at computations involving the "centre" $B$, and so a $B$-trace can be appointed at the node working for the positive requirement, once the $B$-computation realising the follower is discovered.

*Toward the construction.* Let $\mathbf{d}$ be a c.e. degree which is not totally $< \omega^\omega$-c.a. Let $g \in \mathbf{d}$ be a function which is not $\omega^n$-c.a. for any $n < \omega$. As in the argument in Chapter V, since $\mathbf{d}$ is c.e., we may replace $g$ by its modulus, and obtain an approximation $\langle g_s \rangle$ which is non-decreasing and such that changes in $g(n)$ force changes in $g(m)$ for all $m \geqslant n$.

List both kinds of requirements in order-type $\omega$; associate the hole $H_n$ with the $n^{\text{th}}$ positive requirement $P^i_e$ and the gate $G_n$ with the $n^{\text{th}}$ negative requirement $N^{i,j}_e$.

As discussed, each positive requirement appoints followers. Each follower $x$ for a positive requirement will be assigned a permitting number $a(x)$. We say that a follower $x$ for the requirement $P^i_e$ is *realised* at stage $s$ if $\Phi_{e,s}(B_s, x)\downarrow = 0$. An uncancelled follower may, at a given stage, either still reside above its hole; occupy some gate or corral; lie in a pocket; or already be enumerated into the set $A_i$. We say that the requirement is *satisfied* at stage $s$ if there is a follower $x$ for $P^i_e$ which is both realised and has already been enumerated into $A_i$. We say that a follower $x$ is *permitted at stage* $s$ if $g_{s+1}(a(x)) \neq g_s(a(x))$.

As discussed, followers are linearly ordered by priority. When a follower $x$ receives attention, all weaker followers are cancelled. When a follower is cancelled, all of its entourage is cancelled with it.

At each stage, a gate may be *occupied* by a final segment of some entourage. We will ensure the following.

LEMMA 1.2. *Let $G_n$ be a gate, associated with the requirement $N_e^{i,j}$. At a stage $s$ the gate may be occupied by at most one final segment of an entourage. That entourage segment does not contain both a ball targeted for $A_i$ and a ball targeted for $A_j$.*

*The associated corral may contain segments of more than one entourage. However, if the gate is occupied by the final segment of the entourage of some follower $x$, then $x$ is weaker than any other follower which has a segment of its entourage in the corral.*

We also ensure the following:

LEMMA 1.3. *Let $x$ be a follower for a requirement associated with the hole $H_m$. Suppose that at stage $s$, $x$ is on the machine. Then $x$'s entourage at stage $s$ is increasing and is partitioned into intervals $I_m < I_{m-1} < \cdots < I_0 < I^*$ such that:*

- *For each $k \leqslant m$, $I_k$ is in the corral $C_k$; and*
- *$I^*$ is nonempty, and either occupies a gate $G_k$ for some $k \leqslant m$, or is lying in the pockets. If $I^*$ is at gate $G_k$ then $I_n = \varnothing$ for all $n < k$.*

*Every ball in the entourage, except possibly for the last one, is targeted for one of the sets $A_0$, $A_1$ or $A_2$, with no two successive ball in the entourage targeted for the same set. The last ball of the entourage is targeted for $B$ if and only if $I^*$ lies in the pockets.*

*Construction.* At stage $s$ a gate $G_n$, associated with the requirement $N_e^{i,j}$, *opens* if for all $y \leqslant t$, $\Theta_e(B, A_i, y) = \Psi(B, A_j, y)\,[s]$, where $t$ is the previous stage at which the gate opened, $t = 0$ if there was no such stage.

At stage $s$, a follower $x$ *requires attention* if one of the following holds:

(1) $x$ is still waiting above its hole, and is now realised;
(2) $x$ is on the machine, and the final segment $I^*$ of its entourage (as in Lemma 1.3) is waiting at a gate $G_n$, which is now open; or
(3) $x$ is on the machine, the final segment $I^*$ of its entourage is waiting in the pockets, and $x$ is permitted at stage $s$.

A positive requirement $P_e^i$ requires attention if either one of its followers requires attention, or if it is not currently satisfied, and no follower for this requirement is currently waiting above the hole.

Let $P_e^i$ be the strongest requirement which requires attention at stage $s$. We cancel the followers for all weaker requirements. If no follower for $P_e^i$ requires attention at this stage, then we appoint a new, large follower $x$ for $P_e^i$, and place it over the hole. Define $a(x)$ to be large.

Otherwise, let $x$ be the strongest follower for $P_e^i$ which requires attention at stage $s$. We cancel all weaker followers for $P_e^i$.

Let $I^*$ be the final segment of $x$'s entourage given by Lemma 1.3; if $x$ currently lies above its hole let $I^*$ be all of $x$'s current entourage. In cases (1) and (2), the segment $I^*$ drops to the highest gate below its current location which is now unoccupied (this is measured after the cancellation of weaker followers). The segment $I^*$ is put in the corresponding corral, and the last ball in $I^*$ rolls out to wait at the gate.

However, if there are no unoccupied gates below $I^*$'s current location, then the balls in the segment $I^*$ are put into the pockets. A new, large trace, targeted for $B$, is appended to this segment.

In case (3), all of the balls in $I^*$ are enumerated into the sets for which they are targeted; they are removed from $x$'s entourage. If $I^*$ consisted of the entirety of $x$'s entourage then $x$ has just been enumerated and the requirement is now satisfied; we can cancel all other followers for $P^i_e$. Otherwise, the last ball in the remaining entourage is waiting in some corral. That last ball now rolls out of the corral and waits at the gate.

At the end of the stage, for any follower $z$ which is still uncancelled, if the last ball $w$ in $z$'s entourage is smaller than $s+1$, and is not targeted for $B$, then we appoint a new, large trace and append it to the end of $z$'s entourage. The location on the machine of the new trace is the same as the location of the previously last ball $w$. Say $w$ is targeted for a set $A_i$. The new trace is targeted for one of the two sets $A_j$ or $A_k$ (where $\{i, j, k\} = \{0, 1, 2\}$) so that Lemma 1.2 still holds.

*Verification.* Before we embark on the verification, we need to ensure that the construction can actually be carried out as described. We need to show that Lemmas 1.2 and 1.3 hold at every stage. The are proved by simultaneous recursion on the stage. Most parts are immediate. We verify two parts of Lemma 1.2:

(1) If $x$ and $y$ are distinct followers, and at stage $s$, part of $x$'s entourage lies in the corral $C_n$ and part of $y$'s entourage waits at the gate $G_n$, then $y$ is weaker than $x$; and
(2) If a ball $z$ rolls out to a gate $G_n$ at stage $s$, then at that time, the gate is unoccupied.

For (1), consider the stage $r < s$ at which the segment of $y$'s entourage which occupies the gate $G_n$ at the beginning of stage $s$ arrived at the gate. Between stages $r$ and $s$ the gate is occupied so no new entourage segments are added to the gate or corral. Hence $x$'s entourage segment already lay in the corral at the beginning of stage $r$. Since $x$ was not cancelled at stage $r$, it must be stronger than $y$.

For (2), Let $x$ be the follower of whose entourage $z$ is a member. The follower $x$ receives attention at stage $s$. If at that stage the final segment of $x$'s entourage arrives at the corral $C_n$, then by the instructions, $G_n$ is empty when that segment moves. Otherwise, balls in a final segment of $x$'s entourage are enumerated into their sets at stage $s$. The new final segment (of which $z$ is the last element) has been waiting in the corral $C_n$ at the beginning of the stage. Suppose that $G_n$ was occupied at the beginning of the stage. Then we know it was occupied by the final segment of the entourage of some other follower $y$. By (1), $x$ is stronger than $y$. And so all the balls in $y$'s entourage are cancelled at stage $s$ (as $x$ receives attention), and the gate becomes unoccupied.

Let $x$ be a follower which at stage $s$ has already been issued from the hole $H_m$ but is not yet cancelled or enumerated into the set it is targeted for. Let $I_{m,s}(x) < I_{m-1,s}(x) < \cdots < I_{0,s}(x) < I^*_s(x)$ be the decomposition of $x$'s entourage at that stage given by Lemma 1.3. We define an ordinal $\beta_s(x)$. Let

$$\bar{\beta}_s(x) = \omega^m \cdot 2|I_{m,s}(x)| + \cdots + \omega^1 \cdot 2|I_{1,s}(x)| + \omega^0 \cdot 2|I_{0,s}(x)|.$$

If $I^*_s(x)$ resides at gate $G_k$ at stage $s$ then we let $\beta_s(x) = \bar{\beta}_s(x) + \omega^k$. If $I^*_s(x)$ resides in the pockets then we let $\beta_s(x) = \bar{\beta}_s(x)$.

Considering various cases, we observe:

LEMMA 1.4. *Suppose that $x$ is on the machine at stage $s$ and is not cancelled at stage $s$, nor is it enumerated into its set a stage $s$. Then $\beta_{s+1}(x) \leqslant \beta_s(x)$; if $x$ receives attention at stage $s$ then $\beta_{s+1}(x) < \beta_s(x)$.*

It follows that every follower receives attention only finitely many times.

LEMMA 1.5. *Every positive requirement $P^i_e$ receives attention finitely many times, and is met.*

PROOF. Suppose that the requirement $P^i_e$ is associated with the hole $H_{m-1}$.

To begin, we note that if $x$ is a follower for $P^i_e$ which is realised at some stage $r$ and is still not cancelled at a stage $s > r$ then $\Phi_e(B, x)\downarrow = 0[s]$ by the same computation which was present at stage $r$. This is standard: suppose that a number $b < \varphi_{e,s}(B_s, x)$ enters $B$ at stage $s$. The number $b$ is the last element of an entourage of some follower $y$. If $y$ is stronger than $x$ then $x$ is cancelled at stage $s$. Otherwise, the trace $b$ is chosen after stage $r$, and so is greater than $\varphi_{e,r}(B_r, x)$, which by induction equals $\varphi_{e,s}(B_s, x)$.

By induction, all positive requirements stronger than $P^i_e$ eventually cease all action; in particular, they stop cancelling followers for $P^i_e$. Let $r^*$ be the last stage at which a requirement stronger than $P^i_e$ receives attention.

If some follower for $P^i_e$ enters $A_i$ after stage $r^*$ then the lemma holds. This is also the case if some follower $x$ for $P^i_e$ is never cancelled but never realised. We will show that one of these two cases must hold. Suppose otherwise, for a contradiction. We will give an $\omega^{2m}$-computable approximation for $g$.

Suppose that $x$ is a follower for $P^i_e$ which is never cancelled. By assumption, it is realised at some stage. By Lemma 1.4 the follower receives attention finitely many times. We assumed that $x$ is not enumerated into $A_i$. This means that the final configuration for $x$ (given by Lemma 1.3) contains an ever-increasing final segment $I^*(x)$ which is either a permanent resident of some gate $G_n$, or a permanent resident of the pockets. In the first case, we say that $x$'s entourage is *stuck* at the gate $G_n$; in the second case, that it is stuck in the pockets.

There are only finitely many followers for $P^i_e$ whose entourage gets stuck at some gate. Indeed there are at most $m$ many. This is because each gets stuck at some gate $G_n$ for some $n < m$, and each gate contains at most one segment as a permanent resident.

We let $r^{**} > r^*$ be the last stage at which a follower, whose entourage is eventually stuck at some gate, receives attention; $r^{**} = r^*$ if there is no such stage. Every follower which receives attention after stage $r^{**}$ was also appointed after stage $r^{**}$. Every such follower is either eventually cancelled, or eventually its entourage is stuck in the pockets, awaiting permission which is never given.

Infinitely many followers are appointed for $P_e^i$, and of those, infinitely many are never cancelled. The argument is again standard: for any stage $t$ consider the strongest follower $x$ which requires attention after stage $t$. Then $x$ is never cancelled, and after the last stage at which $x$ receives attention, a new follower is appointed, and eventually receives attention as it is eventually realised.

Let $p > r^{**}$. To approximate $g(p)$ we let, for $s > p$, $X_s(p)$ be the set of followers $y > r^{**}$ for $P_e^i$ which are uncancelled at stage $s$ such that $a(y) \leqslant p$. This set is naturally ordered (in an increasing fashion). If $s < t$ then $X_t(p)$ is an initial segment of $X_s(p)$; some followers in $X_s(p)$ may get cancelled; new permitting numbers are always assigned to be large.

Let $S(p)$ be the set of stages $s > r^{**}, p$ such that:

- at stage $s$ there is some follower $x > r^{**}$ for $P_e^i$ such that $a(x) > p$; and
- if $x = x_s(p)$ is the least such follower, then the final segment $I_s^*(x)$ of $x$'s entourage is waiting in the pockets at stage $s$.

The set $S(p)$ is infinite, indeed it is cofinite. The sets $X_s(p)$ stabilise to some $X(p)$; let $s$ be the last stage at which any follower in $X(p)$ receives attention. The next follower $x$, appointed at stage $s+1$, is never cancelled and $a(x) > p$, so $x = x_t(p)$ for all $t > s$; $x$'s entourage is eventually stuck in the pockets.

Let $s \in S(p)$; let $y_{1,s}, y_{2,s}, \dots, y_{\ell(s),s}$ be the increasing enumeration of $X_s(p)$. We let

$$\sum_{y \in X_s(p)} \beta_s(y) = \beta_s(y_{1,s}) + \beta_s(y_{2,s}) + \cdots + \beta_s(y_{\ell(s),s})$$

and

$$\gamma_s(p) = \omega^m \cdot \left( \sum_{y \in X_s(p)} \beta_s(y) \right) + \beta_s(x_s(p)).$$

Since $\beta_s(x) < \omega^m$ for all $x$, we see that $\gamma_s(p) < \omega^{2m}$.

Let $s \in S(p)$, and let $t$ be the next stage in $S(p)$ after stage $s$. We show that $\gamma_t(p) \leqslant \gamma_s(p)$, and that if $g_t(p) \neq g_s(p)$ then $\gamma_t(p) < \gamma_s(p)$.

Suppose that $x_t(p) \neq x_s(p)$. Then the follower $x_s(p)$ must be cancelled by stage $t$. This means that one of the followers in $X_s(p)$ received attention between stages $s$ and $t$; let $z$ be the least such follower. Then $z$ is the last (greatest) element of $X_t(p)$. By Lemma 1.4, $\beta_t(z) < \beta_s(z)$. This shows that $\sum_{y \in X_t(p)} \beta_t(y) < \sum_{y \in X_s(p)} \beta_s(y)$. Even though $\beta_t(x_t(p))$ may be much larger than $\beta_s(x_s(p))$, it is smaller than $\omega^m$, and this shows that $\gamma_t(p) < \gamma_s(p)$.

So we assume that $x_t(p) = x_s(p)$; let $x = x_s(p)$. For all $y \in X_s(p) = X_t(p)$, $\beta_t(y) \leqslant \beta_s(y)$, and $\beta_t(x) \leqslant \beta_s(x)$, so $\gamma_t(p) \leqslant \gamma_s(p)$. Suppose that $g_t(p) \neq g_s(p)$. Since $x$ is not cancelled between stages $s$ and $t$ and $a(x) > p$, it follows that $x$ is permitted at some stage between $s$ and $t$. At the first such stage, $x$'s final entourage segment is still waiting in the pockets, and so $x$ receives attention between stages $s$ and $t$. Lemma 1.4 guarantees that $\beta_t(x) < \beta_s(x)$, and this implies that $\gamma_t(p) < \gamma_s(p)$ as required. □

Lemma 1.6. *All sets $A_0, A_1, A_2$ and $B$ are computable from* $\mathbf{d}$.

Proof. To determine if a number $z$ is an element of one of these sets or not, we first go to stage $z$. We then see if $z$ has already been chosen as a follower or a trace; and if so, to which set it is targeted. If not, then $z$ does not enter any set, since new followers and traces are chosen to be large.

Suppose that $z$ is an element of an entourage of a follower $x$ (possibly $x = z$) at some stage $t \leqslant z$. The number $a(x)$ is already determined by stage $z$. With oracle $g$ we can find a stage after which the follower $x$ is never permitted. The function $g$ can thus calculate a stage after which $z$ cannot enter any set. □

The verification concludes with the following two lemmas, which are standard, but are added for completeness.

LEMMA 1.7. *If* $\{i, j, k\} = \{0, 1, 2\}$ *then* $A_i \leqslant_T A_j \oplus A_k \oplus B$.

PROOF. We ensured that if $y$ is targeted for $A_i$ then at all stages $s > y$ at which $y$ is on the machine, $y$ has a trace $z$ targeted to one of the sets $A_j$, $A_k$ or $B$, and $y$ does not enter $A_i$ unless the trace $z$ enters the set it is targeted for. Further, $y$ is either cancelled or eventually receives a trace which is never cancelled; this is due to Lemma 1.4. □

LEMMA 1.8. *Every negative requirement* $N_e^{i,j}$ *is met.*

PROOF. Suppose that $\Theta_e(A_i, B) = \Psi_e(A_j, B)$ are total. Let $G_n$ be the gate associated with the requirement $N_e^{i,j}$.

By Lemma 1.5, let $r^*$ be the last stage at which either:

- Any positive requirement which is a associated with a hole $H_m$ for some $m < n$ receives attention; or
- Any follower whose entourage is eventually stuck at some gate $G_m$ for some $m < n$ receives attention.

Let $M$ be the set of $m \leqslant n$ such that the gate $G_m$ does not have a permanent resident. We assumed that the hypothesis of $N_e^{i,j}$ holds; this implies that $G_n$ opens infinitely often, and so $n \in M$.

We let $S^*$ be the set of stages $s > r^*$ at the beginning of which:

- For all $m \in M$, the gate $G_m$ is unoccupied;
- If $I_s^*(x)$ is the final segment of an entourage of a follower $x$ which lies in the pockets, then $x$ will not receive attention at stage $s$ or after stage $s$.

The set $S^*$ is computable from $B$; this is because entourage segments in the pockets end with traces targeted for $B$. We note that if $s \in S^*$ and $x$ is a follower, part of whose entourage resides anywhere below the gate $G_n$, then $x$ does not receive attention after stage $s$; the last segment of $x$'s entourage is either permanently at a gate or in the pockets.

The set $S^*$ is infinite. Let $t$ be a large stage. As usual, let $x$ be the strongest follower which ever receives attention after stage $t$; say $x$ last receives attention at stage $s - 1 > t$. All balls on the machine at the beginning of stage $s$ will never move again; if a gate $G_m$ is occupied at the beginning of stage $s$ then the residents of $G_m$ at stage $s$ are permanent. Hence $s \in S^*$.

Let $p < \omega$. We let $s(p)$ be the least stage $s \in S^*$ such that $s > p$, $G_n$ was open at some stage in the interval $(p, s)$, and $\Theta_e(A_i, B, p)\downarrow = \Psi_e(A_j, B, p)\downarrow\ [s]$. Such a stage exists because we assume that the hypothesis of $N_e^{i,j}$ holds. Let $a = \Theta_e(A_i, B, p)[s(p)]$. We claim that $a = \Theta_e(A_i, B, p)$. To show this we prove by induction that for all $s > s(p)$, either $\Theta_e(A_i, B, p)\downarrow\ [s] = a$ or $\Psi_e(A_j, B, p)\downarrow\ [s] = a$.

Let $s > s(p)$ and suppose that the claim is already established for all stages in the interval $[s(p), s)$. Let $x$ be the strongest follower which receives attention at any stage in the interval $[s(p), s)$ (if no follower receives attention then the computations which were observed at stage $s(p)$ were not destroyed by stage $s$).

Since $s(p) \in S^*$, no part of $x$'s entourage lies below $G_n$ at stage $s(p)$. Suppose that no part of $x$'s entourage crosses the gate $G_n$ at any stage in the interval $[s(p), s)$. In this case let $t < s$ be the last stage before stage $s$ at which $x$ received attention. By induction either $\Theta_e(A_i, B, p)\downarrow [t] = a$ or $\Psi_e(A_j, B, p)\downarrow [t] = a$; without loss of generality, assume the former. No numbers are enumerated into sets during stage $t$. If a number from some follower $y$'s entourage is enumerated into any set between stages $t$ and $s$, then $y$ is weaker than $x$, and so was appointed after stage $t$, and so is greater than the use $\theta_{e,t}(p)$. Thus the computation $\Theta_e(A_i, B, p)[t]$ is preserved until stage $s$.

Suppose then that parts of $x$'s entourage do cross the gate $G_n$ at some stages in the interval $[s(p), s)$. Let $t$ be the last stage in that interval at which any part of $x$'s entourage crosses the gate. We note that whenever $x$ receives attention, all other followers that were appointed after stage $s(p)$ are cancelled. In particular, $G_n$ becomes unoccupied. We conclude that no segments of $x$'s entourage ever pass by the gate without stopping first. Hence, at stage $t$, the gate opens, and part of $x$'s entourage that was waiting at the gate is allowed to proceed downwards.

This implies two things: the first, that $\Theta_e(A_i, B, p)\downarrow [t] = \Psi_e(A_i, B, p)\downarrow [t]$; by induction, the common value is $a$. The second is that the segment of $x$'s entourage which is released from the gate at stage $t$ does not contain both balls targeted for $A_i$ and balls targeted for $A_j$. Without loss of generality, suppose it does not contain any balls targeted for $A_j$. We claim that the computation $\Psi_e(A_j, B, p)[t]$ is not destroyed by stage $s$.

For suppose that some number $u$ below the use $\psi_{e,y}(p)$ of that computation is enumerated into $A_j$ or $B$ at some stage in the interval $[t, s)$. Let $y$ be the follower to whose entourage $u$ belongs. By the choice of $x$, either $y = x$ or $y$ is weaker than $x$. If $y$ is weaker than $x$ then $y$ is appointed after stage $t$, and so $y$, and all of the balls in its entourage, are greater than the use $\psi_{e,t}(p)$. But $y = x$ is impossible too: $u$ must be appointed before stage $t$, and so is already an element of $x$'s entourage at stage $t$. But it does not cross the gate at stage $t$: no balls targeted for either $A_j$ or $B$ proceed from the gate at stage $t$. All other balls in $x$'s entourage at stage $t$ remain above the gate until stage $s$. □

**1.3. The 1-4-1 lattice.** The embedding technique used above actually shows:

THEOREM 1.9. *If* $\mathbf{d}$ *is a totally* $< \omega^\omega$*-c.a. c.e. degree then for all* $n \geqslant 3$*, the 1-n-1 lattice can be embedding into the c.e. degrees below* $\mathbf{d}$.

Take for example the case $n = 4$. We enumerate sets $A_0$, $A_1$, $A_2$ and $A_3$, and a bottom set $B$. The requirements are as above, except for the pairwise joins: if $i, j, k$ are distinct indices from $\{0, 1, 2, 3\}$ then $A_i \leqslant_{\mathrm{T}} A_j \oplus A_k \oplus B$. The rule for traces now is that if $\{i, j, k, l\} = \{0, 1, 2, 3\}$ then every number targeted for $A_i$ needs to have two traces, for two of the sets $A_j$, $A_k$ and $A_l$.

It would seem that an entourage in this construction will be a binary branching tree, but we can actually make do with linear entourages as in the construction above; the two balls following a ball in a (linear) sequence of balls are considered its traces. That is, if the follower is $t_0$ and the entourage is $t_0, t_1, t_2, \ldots, t_\ell$ then

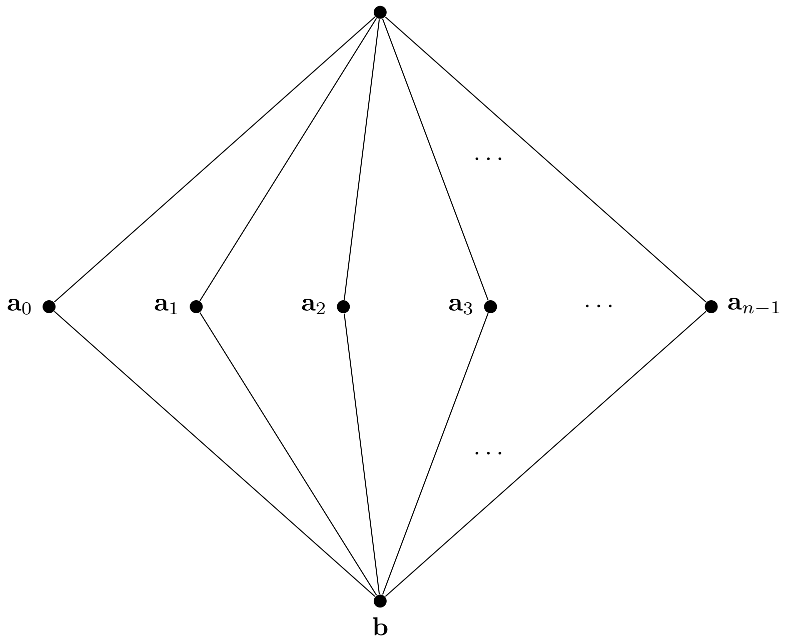


FIGURE 2. The 1-$n$-1 lattice

for all $i \leqslant \ell - 2$, $t_{i+1}$ and $t_{i+2}$ are the traces for $t_i$. For the tracing to work we need to require that for any such $i$, no two of the three balls $t_i$, $t_{i+1}$ and $t_{i+2}$ are targeted for the same set. Given two previous balls $t_{i-2}$ and $t_{i-1}$, this still leaves two options for choosing a target for the next ball $t_i$, and this allows us to retarget followers at gates. A sequence of balls waiting at a gate working for $N_e^{i,j}$ will be an $(i,k,l)$-stream or a $(j,k,l)$-stream. The rest of the construction is identical.

In fact, we can string together these constructions to obtain an embedding of the 1-$\omega$-1 lattice; the $n^{\text{th}}$ follower appointed (across all requirements) and its entourage will only concern itself with the first $n$ middle sets; reductions $A_i \leqslant_{\mathrm{T}} A_j \oplus A_k \oplus B$ will be non-uniform.

## 2. Non-embedding critical triples

A *critical triple* in an upper semi-lattice consists of three incomparable elements $\mathbf{a}_0, \mathbf{a}_1$ and $\mathbf{b}$ such that $\mathbf{a}_i \leqslant \mathbf{b} \vee \mathbf{a}_{1-i}$ for $i = 0, 1$, and such that any $\mathbf{e}$ lying below both $\mathbf{a}_0$ and $\mathbf{a}_1$ lies below $\mathbf{b}$ as well. That is, $\mathbf{a}_0 \wedge \mathbf{a}_1 \leqslant \mathbf{b}$, except that we don't actually require the meet $\mathbf{a}_0 \wedge \mathbf{a}_1$ to exist. The element $\mathbf{b}$ is called the *centre* of the triple.

In [17] the authors show that a c.e. degree bounds a critical triple (in the c.e. Turing degrees) if and only if it is not totally $\omega$-c.a. The proof shows that the same holds for weak critical triples, a related concept we will not use here. The proof that no totally $\omega$-c.a. c.e. degree bounds a weak critical triple is an "anti-permitting" elaboration on an argument from [8] that constructs a c.e. degree which bounds no weak critical triple. That argument in turn is a simplification of an argument from [14], which constructs a c.e. degree that bounds no critical triple. Toward the proof of the second half of Theorem I.0.7, we now give an anti-permitting elaboration on Downey's original argument in [14]. It is somewhat more complicated than the weak critical triple argument, and gives a weaker result. But it will be the argument that we need to generalise in order to prove our theorem. To avoid an extra step of simple permitting we work with array computable degrees rather than totally $\omega$-c.a.

**2.1. Layering.** The fundamental notion from [14] is that of protecting computations by layers. In our setting, let $D$ be a c.e. set whose Turing degree is array computable; and let $A_0, A_1, B \leqslant_{\mathrm{T}} D$ be sets whose degrees potentially form a critical triple. To show that they in fact do not form a critical triple we will build a c.e. set $Q \leqslant_{\mathrm{T}} A_0, A_1$ such that $Q \nleqslant_{\mathrm{T}} B$; or we may fail to do so, but in that case we will show that $A_0$ is computable from $B$. We fix functionals $\Lambda$, $\Phi_0$ and $\Phi_1$ such that $\Lambda(D) = (B, A_0, A_1)$, and such that $\Phi_i(B, A_{1-i}) = A_i$ for $i = 0, 1$.

The general idea of the construction is as follows. We define an auxiliary function $\Delta(D)$, and as in the anti-permitting arguments in the previous chapters, non-uniformly we know an id-computable approximation for $\Delta(D)$. We enumerate the set $Q$, together with reductions $\Gamma_i$ of $Q$ to $A_i$. For each $d < \omega$, to ensure that $\Psi_d(B) \neq Q$ we appoint a follower $x$, and after it is realised ($\Psi_d(B, x)\downarrow = 0$) we hope for double permission — changes in both $A_0$ and $A_1$ below the uses of reducing $Q(x)$ to these sets — so that we can enumerate $x$ into $Q$. The natural two questions are: (a) why would we get double permission? (b) if we do, how do we protect the realisation of the follower — i.e., how do we ensure that indeed $\Psi_d(B, x) = 0$?

The idea is to have a backup strategy. We build a functional $\Xi_d$; if the $d^{\text{th}}$ requirement fails, that is, if $\Psi_d(B) = Q$, then we will ensure that $\Xi_d(B) = A_0$. Suppose that $x$ is a follower. When we see that $x$ is realised then we set up a computation of $A_0 \upharpoonright_x$ from $B$, with use at least $\psi_d(B, x)$. If later we attack with $x$ and then $x$ becomes unrealised, then we will be able to cancel $x$, because any incorrect computation of $A_0 \upharpoonright_x$ from $B$ can be discarded as well. This solves the problem (b) above. However, this process introduces two analogous problems (assuming that indeed $\Psi_d(B) = Q$): (b') how do we protect the correctness of a computation $\Xi_d(B) = A_0 \upharpoonright_x$ (when $x$ is not cancelled); and (a') how to ensure that infinitely many followers are not cancelled so that $\Xi_d(B)$ is total?

This is where anti-permitting comes in. We associate a follower $x$ with an anchor $n$, an input for $\Delta(D)$. As long as we keep $\Delta(D)$ total, having guessed the correct approximation, we know there will be no more than $n$ many changes to $D \upharpoonright_{\delta(n)}$. If we can arrange $\delta(n)$ to be large enough, beyond $\lambda(u)$, then we can ensure that there are at most $n$ many changes to $A_i \upharpoonright_u$ or $B \upharpoonright_u$ (recall that $\Lambda$ is the functional computing $A_0, A_1$ and $B$ from $D$).

A single layer above $x$ is the length $u > x$ required to ensure that a change in one of the sets $A_0$ or $A_1$ below $x$ necessitates a change in at least one other set among $A_0, A_1$ and $B$ below $u$. Formally we define

$$x^{(1)} = \max\{\varphi_0(B, A_1, x), \varphi_1(B, A_0, x)\};$$

see Figure 3. We then let

$$x^{(n+1)} = (x^{(n)})^{(1)}$$

(see Figure 4).

When we set up $x$, we define the use of reducing $Q(x)$ to the sets $A_i$ to be $x^{(n)}$; and set $\delta(n) = \lambda(x^{(n)})$. When $x$ is realised, we set the use $\xi_d(x)$ of reducing

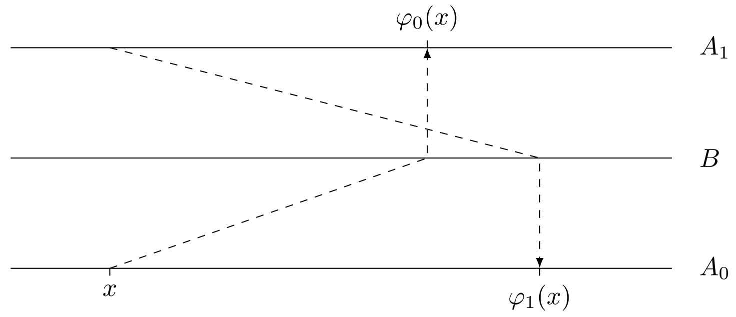


FIGURE 3. One layer.

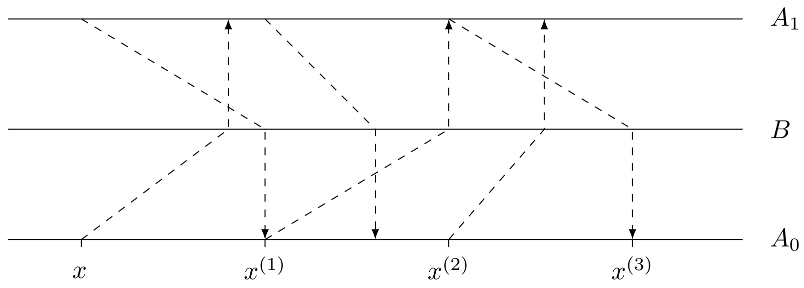


FIGURE 4. Three layers.

$A_0 \restriction_x$ to $B$ to be $\max\{x^{(n)}, \psi_d(x)\}$. We consider what the next change could be. Assuming that $x$ remains realised, we are concerned about $A_i$-changes. The key, again, is that $y^{(1)}$ is chosen so that a change in some $A_i$ below $y$ forces a change in either $B$ or $A_{1-i}$ below $y^{(1)}$. So now there can be two kinds of $A_i$-changes. If one $A_i$ changes below $x^{(n-1)}$, then (again assuming that $x$ remains realised, so $B$ does not change), there must be a change in $A_{1-i}$ below $x^{(n)}$. But $x^{(n)} = \gamma_i(x) = \gamma_{1-i}(x)$, the uses of reducing $Q(x)$ to $A_i$ and $A_{1-i}$; so in this case we get the double change we wished for, and we can attack with $x$: enumerate it into $Q$, and hopefully win the $e^{\text{th}}$ requirement $\Psi_d(B) \neq Q$. Otherwise, the $A_i$-change that concerns us happens below $x^{(n)}$ but above $x^{(n-1)}$. We say that the $n^{\text{th}}$ layer is *peeled*. Since $\delta(n) = \lambda(x^{(n)})$, the $A_i$-change allows us to redefine $\Delta(n)$ and extract one $D \restriction_{\delta(n)}$-change from our opponent. And the opponent's captial is bounded: at most $n$ changes are possible. The $n^{\text{th}}$ layer is gone, but we now repeat the argument with the $(n-1)^{\text{st}}$ layer instead: a change below $x^{(n-2)}$ leads to an attack; a change below $x^{(n-1)}$ but not below $x^{(n-2)}$ means that the next layer is peeled, and another change in $\Delta(D, n)$ is paid by the opponent. Since we have set up sufficiently many layers, if an attack never occurs, the opponent cannot peel all of the layers, which in particular means that no changes to $A_0 \restriction_x$ are possible — ensuring the correctness of the reduction $\Xi_d(B)$ on $x$.

Finally, the anchor $n$ is also used to solve problem (a'): if we can ensure that each time that we cancel $x$, $D$ changes below $\delta(n)$, then we can cancel $x$ and appoint a new follower $x'$, but keep the same anchor $n$. For each anchor $n$, at most $n$ followers can be cancelled, and so one will be permanent. There are some delicate details involved, though, and we discuss them below.

**2.2. Four procedures.** Let us give more details and fix notation. For every $e < \omega$ we will perform an $e^{\text{th}}$ construction. All constructions together define a functional $\Delta$, and ensure that $\Delta(D)$ is total. Let $\langle f^e, o^e \rangle$ be an effective enumeration of all id-c.a. functions (with tidy (id $+1$)-computable approximations). The $e^{\text{th}}$ construction guesses that $\Delta(D) = f^e$. The $e^{\text{th}}$ construction enumerates a c.e. set $Q^e$. For each $d$, an agent $d$ for the $e^{\text{th}}$ construction tries to ensure that $\Psi_d(B) \neq Q^e$. The construction builds two functionals $\Gamma_0^e$ and $\Gamma_1^e$, with the aim of ensuring that $\Gamma_i^e(A_i) = Q^e$. The $d^{\text{th}}$ agent also enumerates a functional $\Xi_d^e$.

We adopt the conventions of Notation VI.1.1; for example, we write $\xi_{d,s}^e(x)\downarrow$ to indicate that $\Xi_d^e(B, x)\downarrow\ [s]$, and when we define the computation we just assign a value to the use; we know that we always define $\Xi_d^e(B, x) = A_0 \upharpoonright_x\ [s]$, $\Gamma_i^e(A_i, x) = Q^e(x)\,[s]$, and $\Delta_s(D_s, n) = s$.

We go one step further and omit mentioning the stage number during the construction; so we just write $\xi_d^e(x)\downarrow$ and understand that this is to be evaluated at the present, i.e., at the stage currently under consideration. To further simplify the notation we omit the superscript $e$.

As discussed above, we are given funcitonals $\Lambda$ and $\Phi_i$ such that for $i = 0, 1$, $\Phi_i(B, A_{1-i}) = A_i$, and $\Lambda(D) = (B, A_0, A_1)$. At a given stage of the construction we may refer to uses such as $\lambda(u)$ for some number $u$. When we do this we understand that we are speeding up the enumerations of the sets and functionals which are given to us so that we see a convergence of the relevant computation (in the example, $\Lambda(D, u)$). Applying this to the uses $\varphi_i$, this allows us to refer to numbers such as $x^{(n)}$ defined above.

At each stage, agent $d$ will appoint a new anchor $n$ (using the next unused number). Each anchor will start a process which will be indpendent of all other processes for all agents and all constructions. The process cycles between four procedures (or *phases*):

- **Set-up:** Appointing a follower $x$; defining a parameter $u = x^{(n)}$, and defining $\delta(n) = \lambda(u)$; waiting for $\Delta(D, n) = f^e(n)$. Once this is observed, defining $\gamma_i(x) = u$.
- **Realisation:** Waiting for $\Psi_d(B, x)\downarrow$. When convergence is obtained, defining $\xi_d(x) = \max\{u, \psi_d(x)\}$.
- **Maintenance:** Waiting for double permission: both $\gamma_i(x)\uparrow$. (While waiting, demanding payment for layers being peeled.)
- **Attack:** When double permission is received, enumerating $x$ into $Q$. Then, monitoring the correctness of the realising computation $\Psi_d(B, x)$.

To understand the construction we need to explain under what circumstances we move from one procedure to another, and how we react to changes when we see them. We discuss some of the principles involved.

*Cancelling a follower.* We cancel a follower $x$ if both $\delta(n)\uparrow$ and $\xi_d(x)\uparrow$, except during the set-up procedure. We need $\delta(n)\uparrow$ so that we will be free to redefine $\delta(n) = \lambda((x')^{(n)})$ for a new follower $x'$ which will be appointed once $x$ is cancelled. We need $\xi_d(x)\uparrow$ as while $\Xi_d(x)\downarrow$ we need to maintain the correctness of this computation. We are not allowed to cancel the follower during the set-up phase, because during set-up we are still waiting for our opponent to make a payment; each cancellation will be charged against a change in $f^e(n)$, and during set-up we have not seen this change yet.

Why would we need to cancel $x$? While we are in set-up, both $\gamma_i(x)$ are undefined, and so any change to any of the sets $A_i$ or $B$ below $u$ will cause us to simply recalculate a new value for $u = x^{(n)}$ and restart the set-up procedure; note that this change forces $\delta(n)\uparrow$. However once we exit set-up, a change in $B$ below $u$ might cause many layers to disappear but it is still possible that one of $\gamma_i(x)$ remains defined; so we cannot return to a fresh set-up for $x$. And certainly, once we have attacked, if realisation is destroyed ($\Psi_d(B,x)\uparrow$) then we need to get rid of $x$, as we cannot extract it from $Q$.

*The value of* $u$. As discussed above, during the set-up phase, any changes to sets $A_i$ or $B$ may increase the value of $x^{(n)}$; we need to keep track of these changes and update the value of $u$. Once we leave set-up we cannot update the value of $u$ anymore; peeling the layers one by one would result in increases to $x^{(n)}$, but at least one of $\gamma_i(x)$ is still defined, so we cannot increase this use to be the new $x^{(n)}$. Once we leave set-up, the value of $u$ is fixed (until the follower $x$ is cancelled).

Actually, one could ask why we ever need to give up on any layer. When the last layer is peeled — say $A_0 \upharpoonright_u$ changes but not $A_1 \upharpoonright_u$ — why shouldn't we just redefine $\gamma_0(x)$ to be the new $x^{(n)}$ and leave $\gamma_1(x) = u$? And later if $A_1 \upharpoonright_u$ changes we could update $\gamma_1(x)$ as well. However the change causes $x^{(n)} > \xi_d(x)$. A change now in $A_1$ below $x^{(n-1)}$ would cause a change in $B$ (rather than $A_0$) below the new $x^{(n)}$ but not below $\xi_d(x)$; we cannot cancel $x$, so we are peeling another layer even though we tried to resurrect the last layer. In other words, there is no way to actually revive the last layer: one change means it is gone.

*The value of* $\delta(n)$. To keep $\Delta(D)$ total, as usual, we need to ensure that $\delta(n)\downarrow$ at every stage (even if the guess $\Delta(D) = f^e$ is wrong), and we need to ensure that the value of this use is bounded. When exiting set-up we have $\delta(n) = \lambda(u)$; when we see that $x$ is realised we will likely have $\psi_d(x) > u$ so will not have $\delta(n) \geqslant \lambda(\xi_d(x))$. This means that during maintenance it is quite possible that a $B$-change causes the realising computation $\Psi_d(B,x)$ to disappear, but $D$ does not change below $\delta(n)$. In this case we need to go back to the realisation procedure and cannot cancel $x$.

However, once we attack, it is important that $\delta(n) \geqslant \lambda(\xi_d(x))$; the reason is that if $B \upharpoonright_{\xi_d(x)}$ changes we must be able to cancel $x$, as it is already enumerated into $Q$. However the double change in $A_i \upharpoonright_u$ that enabled that very attack caused $\delta(n)\uparrow$, and this allows us to redefine $\delta(n)$ to be at least $\lambda(\xi_d(x))$ as required.

Further, during maintenance, if we see one layer peeled the we must update $\delta(n)$ to be $\lambda(\xi_d(x))$. The reason is that while waiting for the opponent to pay for this peeling we may see that $\Psi_d(B,x)\uparrow$. We would then like to cancel $x$: if we do not do so, while waiting we may see more layers unravel, so we would like to attack, but obviously cannot do so if $x$ is no longer realised.

**2.3. Construction.** We detail how to react to changes during each procedure for an anchor $n$ for an agent $d$ (for construction $e$). Recall that during the construction, at each stage, every agent for every construction appoints a new anchor $n$ and starts cycling through the procedures for $n$. The following description of these procedures therefore describes the entire construction.

Set-up.

**1.** Appoint a new follower $x$. Define $\delta(n) = \lambda(x^{(n)})$. Wait for $\Delta(D, n) = f^e(n)$.

- While waiting, if $D$ changes below $\delta(n)$, we redefine $\delta(n)$ using the current value of $x^{(n)}$.

**2.** Once we see that $\Delta(D, n) = f^e(n)$, we define $u = x^{(n)}$ and $\gamma_i(x) = u$, and move to realisation.

Realisation.

**1.** Wait for $\Psi_d(B, x)\downarrow= 0$.

- If, while waiting, we see that $D$ changes below $\delta(n)$, then we cancel $x$ and return to set-up.

**2.** Once we see that $\Psi_d(B, x)\downarrow= 0$, we define $\xi_d(x) = \max\{u, \psi_d(x)\}$, and move to maintenance.

Maintenance.

We wait for a change in $D$ below $\delta(n)$ or in $B$ below $\xi_d(x)$. When we see such a change we react according to the first case which applies:

(a) *Cancellation:* If both $\delta(n)\uparrow$ and $\xi_d(x)\uparrow$ then we cancel $x$ and return to set-up.

(b) *Realisation:* If $\xi_d(x)\uparrow$ (but $\delta(n)\downarrow$), we return to the realisation phase.

(c) *Attack:* If both $\gamma_i(x)\uparrow$ (but $\xi_d(x)\downarrow$) then we move to the attack phase.

(d) *Layer peeled:* If only one $\gamma_i(x)\uparrow$ then we redefine $\delta(n) = \lambda(\xi_d(x))$ and wait for $\Delta(D, n) = f^e(n)$.

- While waiting, if one of the cases (a), (c) or (e) applies, we react accordingly. (b) cannot happen anymore.

When we see the required agreement we redefine $\gamma_i(x) = u$, $\delta(n) = \lambda(u)$, and stay at the maintenance phase.

(e) *Trivial change:* If only $\delta(n)\uparrow$ then we redefine $\delta(n) = \lambda(\xi_d(x))$ and stay at the maintenance phase.

Attack.

**1.** We enumerate $x$ into $Q$. We define $\delta(n) = \lambda(\xi_d(x))$.

**2.** We wait for $\xi_d(x)\uparrow$. When this is observed, we cancel $x$ and return to set-up.

- While waiting, if we see that $\delta(n)\uparrow$, we redefine $\delta(n) = \lambda(\xi_d(x))$, and keep waiting.

**2.4. Verification.**

LEMMA 2.1. *Let $e$ be a construction, $d$ an agent for $e$, and $n$ an anchor for $d$. There is a follower which is appointed for $n$ and is never cancelled.*

PROOF. Let $s_0$ be a stage after which $f^e(n)$ does not change. Suppose that at some stage $s_1 > s_0$ a follower $x$ is appointed for $n$. Then the set-up phase is never exited, and so $x$ is never cancelled. □

LEMMA 2.2. *$\Delta(D)$ is total.*

PROOF. Let $n < \omega$ be an anchor for some agent $d$ (for construction $e$). We note that $\delta(n)$ is never left undefined at the end of a stage, so we just need to show that the value of $\delta(n)$ is bounded (over all stages).

By Lemma 2.1, let $x$ be the last follower appointed for $n$. There are several possibilities for where we can end up with $x$.

(1) It is possible to get stuck for ever waiting for realisation. In this case, we know that $\delta(n)$ can never get undefined after starting the realisation run, as that would cancel $x$.
(2) An attack with $x$ is performed. We would never end this attack. The value $\xi_d(x)$ is constant during the attack. During the attack we let $\delta(n) = \lambda(\xi_d(x))$. Since $\Lambda(D)$ is total, the value $\lambda(v)$ stabilizes for all $v$.
(3) It is possible to be left in the set-up cycle, never getting a correct $f^e$ guess. The value of $x^{(n)}$ may change a number of times, but since $\Phi_i(B, A_{1-i})$ are both total, it eventually stabilises. We always define $\delta(n) = \lambda(x^{(n)})$, and so again since $\Lambda(D)$ is total, this value is eventually constant.
(4) After we enter the maintenance phase, $D\restriction_{\delta(n)}$ never changes. In this case obviously $\delta(n)$ is constant after we enter maintenance.
(5) We enter maintenance with $x$, and at some stage $s_1$ after that we see a $D\restriction_{\delta(n)}$-change. We then define $\delta(n) = \lambda(\xi_d(x))$. After stage $s_1$ there cannot be a change in $B\restriction_{\xi_d(x)}$ — such a change would cause us to cancel $x$. We will therefore remain at maintenance and always define $\delta(n) = \lambda(\xi_d(x))$; again, this reaches a limit. □

We fix some $e$ such that $\Delta(D) = f^e$, and continue with omitting the superscript $e$.

LEMMA 2.3. *$Q$ is computable from both $A_0$ and $A_1$.*

PROOF. The construction ensures that a follower $x$ never enters $Q$ unless both $\Gamma_0(A_0, x)\uparrow$ and $\Gamma_1(A_1, x)\uparrow$. We always define $\Gamma_i(A_i, x)$ to agree with $Q(x)$; so we just need to show that $\gamma_i(x)\downarrow$, or $x$ is cancelled, or is enumerated into $Q$. Suppose that $x$ is a follower (for some anchor $n$ for some agent $d$) which is never cancelled and is never enumerated into $Q$. We show that $\gamma_i(x)$ is defined at infintiely many stages, and that the value is bounded. (As usual we assume that if $x$ is cancelled, or never chosen as a follower, or is enumerated into $Q$, then we eventually define both computations $\Gamma_i(A_i, x)$ with use 0.)

Since the guess $\Delta(D) = f^e$ is correct, we successfully exit the set-up phase for $x$. After set-up, the parameter $u$ is fixed, and $\gamma_i(x)$, when defined, is always defined to equal $u$, and is thus bounded. The only time after set-up at which $\gamma_i(x)$ is undefined is when a layer is peeled, and we wait for agreement between $\Delta(n)$ and $f^e(n)$; such agreement will eventually be found, and then $\gamma_i(x)$ will be redefined.

Since whenever $\gamma_i(x)\uparrow$ we also get $\delta(n)\uparrow$, any other context at which $\gamma_i(x)\uparrow$ causes $x$ to be cancelled (or attacked with). □

If $Q \nleqslant_{\mathrm{T}} B$ then we are done. Otherwise, we fix some $d$ such that $\Psi_d(B) = Q$; we will show that $\Xi_d(B)$ computes $A_0$ successfully. We made sure that if a follower $x$ for agent $d$ is ever cancelled, then $\xi_d(x)\uparrow$ when we do so. The agent $d$ appoints a new anchor at every stage; by Lemma 2.1, for each one there is a follower which is never cancelled. So it suffices to show that if $x$ is a follower for agent $d$ which is never cancelled, then eventually a permanenet computation $\Xi_d(B, x)$ is defined, and this computation correctly computes $A_0 \restriction_x$. Fix a never-cancelled follower $x$ for an anchor $n$ for agent $d$.

Since $e$'s guess that $\Delta(D) = f^e$ is correct, we exit the set-up phase with $x$. Since $\Psi_d(B) = Q$, every time we enter the realisation phase with $x$ we will also exit it. Further, the use $\psi_d(x)$ reaches a limit, which implies that the use $\xi_d(x)$ reaches a limit; whence we eventually define a permanent computation $\Xi_d(B, x)$. We need to verify its correctness. We note that since $x$ is never cancelled, we do not enter the attack phase with $x$. And so after the permanent computation $\Xi_d(B, x)$ is defined, we will for ever be in maintenance with $x$, potentially observing layers being peeled. Again, since $e$ is correct, after each peeling we will observe agreement between $\Delta(D, n)$ and $f^e(n)$.

Let $s^*$ be the stage at which the permanent computation $\Xi_d(B, x)$ is defined. We need to show that $A_0 \restriction_x = A_{0,s^*} \restriction_x$. This is the heart of the argument: showing that setting up sufficiently many layers protects the correctness of $\Xi_d(B, x)$. First we observe again that between set-up and last realisation we do not see $D \restriction_{\delta(n)}$-changes. That is, if $t^*$ is the stage at which set-up of $x$ is exited, then $D_{s^*} \restriction_{\delta(n)} = D_{t^*} \restriction_{\delta(n)}$; otherwise, we would increase $\delta(n)$ to be $\xi_d(x)$, and then at some stage before stage $s^*$, $x$ would be cancelled. This implies that $A_{i,s^*} \restriction_u = A_{i,t^*} \restriction_u$ and $B_{s^*} \restriction_u = B_{t^*} \restriction_u$; since $u = x^{(n)}$ as calculated at stage $t^*$, we have $u = x^{(n)}$ at stage $s^*$ as well.

For $k \leqslant n$ we let $v_k = x^{(k)}$ as calculated at stage $s^*$ (or $t^*$); and we let $s_1 < s_2 < s_3 < \cdots < s_m$ be the stages at which a layer for $x$ is peeled (stages at which we observe case (d) of the maintenance cycle for $x$). So for some $i < 2$, $A_{i,s_k+1} \restriction_u \neq A_{i,s_k} \restriction_u$.

Since $o^e_0(n) \leqslant n$ and during the set-up stage we force one change in $\Delta(D, n)$, we have $o^e_{s_1}(n) \leqslant n - 1$. Every time a layer is peeled we force one more change in $\Delta(D, n)$; this implies that for all $k$, $o^e_{s_k}(n) \leqslant n - k$. It follows that $m \leqslant n$.

Lemma 2.4. *For all $k \leqslant m$, for both $i = 0, 1$,*

$$A_{i,s_k} \restriction_{v_{n-k+1}} = A_{i,s^*} \restriction_{v_{n-k+1}} \tag{1}$$

Proof. The stage $s_1$ is the least stage after stage $s^*$ at which we see any change in either $A_i$ below $u$. In other words, $A_{i,s_1} \restriction_u = A_{i,s^*} \restriction_u$; since $u = v_n$, the equalities (1) hold for $k = 1$.

Now by induction let $k < m$ and suppose that Eq. (1) holds for $k$ (for both $i < 2$). We note that for all $s > s^*$ and $r \leqslant n$, if $A_{i,s} \restriction_{v_r} = A_{i,s^*} \restriction_{v_r}$ for both $i$ then $x^{(r)} = v_r$ when calculated at stage $s$. Fix $i$ such that $A_{i,s_k+1} \restriction_u \neq A_{i,s_k} \restriction_u$. Since at the beginning of stage $s_k$, $x^{(n-k+1)} = v_{n-k+1}$, the fact that $A_{1-i} \restriction_u$ does not change at stage $s_k$ implies that the change in $A_i$ at that stage is neccessarily above $v_{n-k}$. Now, by induction on $s \in (s_k, s_{k+1})$ we show that for both $j < 2$, $A_{j,s} \restriction_{v_{n-k}} = A_{j,s_k} \restriction_{v_{n-k}}$.

Let $t_k > s_k$ be the stage at which we exit the peeling subroutine (d) of the maintenance cycle that we enter at stage $s_k$. Suppose that $s \in (s_k, t_k)$. Between stages $s_k$ and $t_k$ we see no changes in $A_{1-i} \upharpoonright_u$ as such a change would open an attack. Recall that we are assuming that $B \upharpoonright_{\xi_d(x)}$, and hence $B \upharpoonright_u$, is correct from stage $s^*$ onwards. This, and the fact that $A_{1-i,s} \upharpoonright_u = A_{1-i,s_k} \upharpoonright_u$, implies that $\varphi_i(B, A_{1-i}, v_{n-k})[s] \leqslant v_{n-k+1}$, and that $A_{i,s} \upharpoonright_{v_{n-k}} = A_{i,s_k} \upharpoonright_{v_{n-k}}$.

After stage $t_k$ and before stage $s_{k+1}$ we see no changes in $A_j \upharpoonright_u$ for either $j < 2$; this follows from the definition of $s_{k+1}$. It follows that for both $j < 2$, $A_{j,s_{k+1}} \upharpoonright_{v_{n-k}} = A_{j,s_k} \upharpoonright_{v_{n-k}} = A_{j,s^*} \upharpoonright_{v_{n-k}}$ as required. □

## 3. Two gates

We go up one level in our hierarchy: in this section we show that a uniformly totally $\omega^2$-c.a. c.e. degree does not bound a copy of the 1-3-1 lattice in the c.e. degrees.

Of course the main difference between this and the previous section must come from the fact that some uniformly totally $\omega^2$-c.a. degrees do bound critical triples (those which are not totally $\omega$-c.a.). We observe that if $\mathbf{a}_0$, $\mathbf{a}_1$ and $\mathbf{a}_2$ are the middle elements of the 1-3-1 lattice then each of the $\mathbf{a}_i$ is the centre of a critical triple (consisting of these three elements). Given a c.e. set $D$ of uniformly totally $\omega^2$-c.a. degree and $B_0, B_1, A \leqslant_{\mathrm{T}} D$ we show that either $B_0$ is not the centre of a critical triple $B_1, B_0, A$; or $B_1$ is not the centre of a critical triple $B_0, B_1, A$. As expected, this adds one more level of non-uniformity.

The main idea is the following. We enumerate a c.e. set $Q = Q^e$ which will be computable from $A$ and $B_0$, and try to ensure that $Q \nleqslant_{\mathrm{T}} B_1$. If we fail, say $\Psi_d(B_1) = Q$, then we enumerate a back-up set $Q_d = Q_d^e$, this time computable from $A$ and $B_1$, and hope that $Q_d \nleqslant_{\mathrm{T}} B_0$. If we fail then we will ensure that $B_1 \leqslant_{\mathrm{T}} B_0$.

The number of times that $D \upharpoonright_{\delta(n)}$ could change will be at most $\omega n$. We will appoint two followers $x$ and $y$; the latter targeted for $Q$, the former for $Q_d$. We will ensure that if the remaining number of changes is $\omega m + k$ then $y > u_x \geqslant x^{(m)}$ and $u_y \geqslant y^{(k)}$, where $u_x$ and $u_y$ are our analogus of $u$ of the previous construction. The peeling as above will happen from outside in: first, $y$ layers will be peeled by successive $A$- and $B_0$ changes, while $B_1$ remains unchanged. When all the $y$-layers have been peeled, one or two $x$-layers will be peeled. But peeling the $x$-layers happens in successive $A$- and $B_1$-changes, not $B_0$-changes. Such a $B_1$-change will allow us to cancel our follower $y$ (while keeping $x$), and set up a new version of $y$, with however many new layers we might need (the new ordinal is now $\omega(m-1)+k'$, with $k'$ as large as our opponent may like).

An overall intuition is that the alternation between $A, B_0$-peeling and $A, B_1$-peeling reflects the retargeting of traces in two gates of the pinball machine used for constructing an embedding of the 1-3-1 lattice. Speaking vaguely, we say that a degree which is not totally $\omega$-c.a. has enough power to pass one gate, but may run out of gas when trying to pass two gates.

**3.1. Discussion.** We start with some details. Let $D$ be a c.e. set whose Turing degree is uniformly totally $\omega$-c.a. Let $A, B_0, B_1 \leqslant_{\mathrm{T}} D$; fix a functional $\Lambda$ such that $\Lambda(D) = (A, B_0, B_1)$. We further suppose that any two of of these sets compute the third; we fix functionals $\Phi$, $\Phi_0$ and $\Phi_1$ such that $\Phi_0(B_0, B_1) = A$, $\Phi_0(A, B_1) = B_0$

and $\Phi_1(A, B_0) = B_1$. For $x < \omega$ we define

$$x^{(1)} = \max\left\{\varphi(B_0, B_1, x), \varphi_0(A, B_1, x), \varphi_1(A, B_0, x)\right\}$$

and $x^{(n+1)} = (x^{(n)})^{(1)}$.

Again the idea is that a change in one of the sets $A$, $B_0$ or $B_1$ below $x$ neccessitates a change in one other of these sets below $x^{(1)}$.

Let $h(n) = \omega n$; let $\langle f^e, o^e\rangle$ be an effective listing of all $h$-c.a. functions (with tidy $(h+1)$-computable approximations). We will define a functional $\Delta$; the $e^{\text{th}}$ construction will guess that $\Delta(D) = f^e$.

The $e^{\text{th}}$ construction will enumerate a c.e. set $Q = Q^e$ and functionals $\Gamma = \Gamma^e$ and $\Theta = \Theta^e$ with the aim of having $\Gamma(A) = Q$ and $\Theta(B_0) = Q$. Further, for each $d < \omega$ the construction will enumerate a c.e. set $Q_d = Q_d^e$ and functionals $\Gamma_d = \Gamma_d^e$ and $\Theta_d = \Theta_d^e$ with the aim of having $\Gamma_d(A) = Q_d$ and $\Theta_d(B_1) = Q_d$. The action for the construction will be done by agents indexed by pairs of natual numbers. An agent $(d, c)$ for the $e^{\text{th}}$ construction will enumerate a funcitonal $\Xi_{d,c} = \Xi^e_{d,c}$ with the aim of computing $B_1$ from $\Xi_{d,c}(B_0)$.

As mentioned, each anchor for each agent will try to appoint a pair of followers $x$ and $y$. The movement between the four procedures is now complicated by the fact that each $x$ can have several $y$'s. In other words we will sometimes cancel $y$ but not $x$ (we always cancel $y$ if we cancel $x$). So for example we may need to return to the set-up procedure to set up a new $y$; but a change may cause us to interrupt the set-up and either cancel $x$ or attack with it.

How should we set up our uses? On top of the principles applied in the simpler construction above, we have the following. Recall that the idea is to set up $x < u_x < y < u_y$ and to arrange that if at the current stage we have $o^e(n) = \omega m + k$ then $u_y \geqslant y^{(k)}$ and $u_x \geqslant x^{(2m)}$. We need to think about the possible changes and at which times they occur.

The follower $y$ behaves similarly to the follower in the previous construction. It is targeted for $Q$; we will define $\gamma(y) = \theta(y) = u_y$ once we leave the set-up procedure (and define $\delta(n) \geqslant \lambda(u_y)$). After $y$ is realised ($\Psi_d(B_1, y)\downarrow = 0$), when both $A$ and $B_0$ change below $u_y$ we will be able to attack with $y$: enumerate it into $Q$. Changes in $B_1$ below $\psi_d(y)$ will either cause us to return to the realisation phase or to cancel $y$; when a single layer is peeled (either $\gamma(y)\uparrow$ or $\theta(y)\uparrow$) then we redefine $\Delta(D, n)$ and wait for the opponent to catch-up.

The follower $x$ is targeted for $Q_d$; we will be able to attack with $x$ if we see that $\Psi_c(B_0, x)\downarrow = 0$ and then both $\gamma_d(x)$ and $\theta_d(x)$ are undefined. As discussed, the idea is that if two layers below $u_x$ are peeled and $x$ is still realised (no change in $B_0$) then we are guaranteed a change in $B_1$ (and in $A$); so we would be able to cancel $y$ and set up many layers for the new $y$.

One role of $x$ of the simpler construction is taken up in this construction by $x$ and not by $y$: we will define $\xi_{d,c}(x) \geqslant \psi_c(x)$, and will use the peeling of $x$ layers to protect the computation $\Xi_{d,c}(B_0, x) = B_1 \upharpoonright_x$. The role of the $y$-layers is secondary; they protect the $x$-layers. As before, we can only cancel $x$ if it becomes unrealised ($\psi_c(x)\uparrow$) — otherwise we need to keep protecting the correctness of the $\Xi_{d,c}$-computation. However, we will also only be allowed to cancel $y$ if it is unrealised ($\psi_d(y)\uparrow$); while it is realised, it needs to keep protecting the outermost $x$-layers.

*A threat.* The success of this process relies on the layers between $y$ and $u_y$ to be peeled one at a time, so that when the two layers below $u_x$ are peeled, we will have already seen $o^e(n)$ drop below the next limit ordinal (we see $\omega m' + k'$ for some $m' < m$). Consider though the situation in which layers between $y$ and $u_y$ are still unpeeled, but the last layer below $u_x$ is peeled due to an $A$-change. Of course there is a change in either $B_0$ or $B_1$ on the first $y$-layer; the former would allow us to attack with $y$. The latter would allow us to cancel $y$. But our opponent will pay by dropping but not below the limit ordinal $\omega m$ (say to $\omega m + k'$, for $k' < k$). We are now left with insufficiently many $x$-layers.

In this situation what we would really like to do is attack with $x$. For this reason we will define the use $\theta_d(x)$ to be at least $u_y$, not $u_x$.

In fact, we will want to define $\theta_d(x) \geqslant \psi_d(y)$ as well. This is done to prepare the ground for the new follower. When $y$ is cancelled we appoint a new one, say $y'$, and then we would like to define $\theta_d(x) \geqslant u_{y'}$. For us to be able to do so, we need $\theta_d(x)\uparrow$ when $y$ is cancelled. The cancellation of $y$ of course follows from $\psi_d(y)\uparrow$.

This requirement in turn means that while we are waiting for $y$ to be realised, we must leave $\theta_d(x)$ undefined. This is ok because we only need to use the set $Q_d$ if our first attempt with $Q$ has failed; we only need $\Theta_d(B_1) = Q_d$ if $\Psi_d(B_1) = Q$.

Similarly, if during an attack with $y$ we see that $\gamma_d(x)\uparrow$, then we leave it undefined for the duration of the attack. The attack is prompted by changes in $A$ and in $B_0$, but $B_1$ remains fixed; in particular, $\theta_d(x)\downarrow$. The $A$-change below $u_x = \gamma_d(x)$ causes an $x$-layer to be peeled; the opponent has not paid for this by successive peeling of $y$-layers. If the attack later fails ($B_1$ changes below $\psi_d(y) \leqslant \theta_d(x)$) then the fact that $\gamma_d(x)\uparrow$ will allow us to attack with $x$ instead.

**3.2. Construction.** At every stage, every agent $(d, c)$ for a construction $e$ appoints a new anchor $n$ and starts a new set-up procedure for $n$. We then cycle through the four procedures for $n$ as soon described. For brevity:

- We say that $x$ is *realised* if $\xi_{d,c}(x)\downarrow$. We say that $y$ is realised if $\theta_d(x)\downarrow$.
- We say that a follower is *confirmed* if we have already exited the set-up cycle during which it was appointed.
- We *may cancel* a follower if it is confirmed, unrealised and $\delta(n)\uparrow$.
- We *may attack* with $x$ if it is realised, and both $\gamma_d(x)\uparrow$ and $\theta_d(x)\uparrow$. We may attack with $y$ if it is realised, and both $\gamma(y)\uparrow$ and $\theta(y)\uparrow$.

We stipulate that throughout the construction, including the set-up cycle, if we may cancel $x$ or attack with it then we do so; *in either case we cancel* $y$. Otherwise, if we may cancel $y$ or attack with it we do so, except during the set-up of $y$. If we cancel a follower but are not attacking, then we return to the set-up cycle. These instructions override all other instructions during the construction.

We now describe the procedures. We also list facts about divergence of functionals at the beginning of each procedure, to be verified later.

SET-UP: $\delta(n)\uparrow$ AND $\theta_d(x)\uparrow$.

> **1.** If $x$ is not currently defined, appoint a new follower $x$. In either case, appoint a new follower $y > x^{(2n)}$. Define $\delta(n) = \lambda(y^{(k)})$, where currently $o^e(n) = \omega m + k$. Wait for $\Delta(D, n) = f^e(n)$. Note that if $x$ is already defined, then it is realised, and we choose $y > \xi_{d,c}(x)$, so $\delta(n) \geqslant \lambda(\xi_{d,c}(x))$.

While waiting, we react to changes as follows.

- If $x$ was appointed during this set-up cycle, and one of $A$, $B_0$ or $B_1$ changes below $x^{(2n)}$, we cancel $y$, appoint a new $y$, and redefine $\delta(n)$ accordingly.
- Otherwise, if $\delta(n)\uparrow$ then we redefine $\delta(n) = \lambda(y^{(k)})$ (using the current value of $y^{(k)}$).

**2.** Once we see that $\Delta(D, n) = f^e(n)$, we define $u_y = y^{(k)}$ and $\gamma(y) = \theta(y) = u_y$. If $x$ was appointed during this cycle, then we define $u_x = x^{(2n)}$. If $\gamma_d(x)\uparrow$ then we define $\gamma_d(x) = u_x$. As discussed, we leave $\theta_d(x)$ undefined.

We move to realisation.

REALISATION: $\theta_d(x)\uparrow$ or $\xi_{d,c}(x)\uparrow$.

**1.** If $y$ is unrealised, wait for $\Psi_d(B_1, y)\downarrow = 0$. Once this is observed, define $\theta_d(x) = \max\{u_y, \psi_d(y)\}$.

**2.** If $x$ is unrealised, wait for $\Psi_c(B_0, x)\downarrow = 0$. Once this is observed, define $\xi_{d,c}(x) = \max\{u_x, \psi_c(x)\}$; move to maintenance. We could have defined $\xi_{d,c}(x) \geqslant u_y$ but this cannot be maintained, since we may later cancel $y$ but be unable to move $\xi_{d,c}(x)$.

MAINTENANCE: all functionals defined.

We wait for a change in $D$ below $\delta(n)$ or for $x$ or $y$ to become unrealised. When this occurs:

(a) If $x$ or $y$ are unrealised, move to realisation.

(b) If a layer is peeled: either $\gamma(y)\uparrow$ or $\theta(y)\uparrow$, but not both — redefine $\delta(n) = \lambda\left(\max\{\theta_d(x), \xi_{d,c}(x)\}\right)$. Wait for $\Delta(D, n) = f^e(n)$. While waiting, if $\delta(n)\uparrow$ (but no attack or cancellation are possible) then we just redefine it by the same formula. When $\Delta(D, n) = f^e(n)$ is observed we redefine all the markers $\gamma(y), \theta(y), \gamma_d(x)$ which are undefined, with value $u_y$ or $u_x$ as appropriate.

(c) If only $\delta(n)\uparrow$ then we redefine $\delta(n) = \lambda\left(\max\{\theta_d(x), \xi_{d,c}(x)\}\right)$ and stay at the maintenance phase.

ATTACK WITH $y$: $\theta(y)\uparrow$, $\gamma(y)\uparrow$, $\delta(n)\uparrow$.

We enumerate $y$ into $Q$. We define $\delta(n) = \lambda\left(\max\{\theta_d(x), \xi_{d,c}(x)\}\right)$. We wait for changes. If $\delta(n)\uparrow$ we redefine it according to the formula above. As discussed, if $\gamma_d(x)\uparrow$ we leave it undefined.

ATTACK WITH $x$: $\theta_d(x)\uparrow$, $\gamma_d(x)\uparrow$, $\delta(n)\uparrow$.

We enumerate $x$ into $Q_d$. We define $\delta(n) = \lambda\left(\xi_{d,c}(x)\right)$. If $\delta(n)\uparrow$ we redefine it according to the same formula.

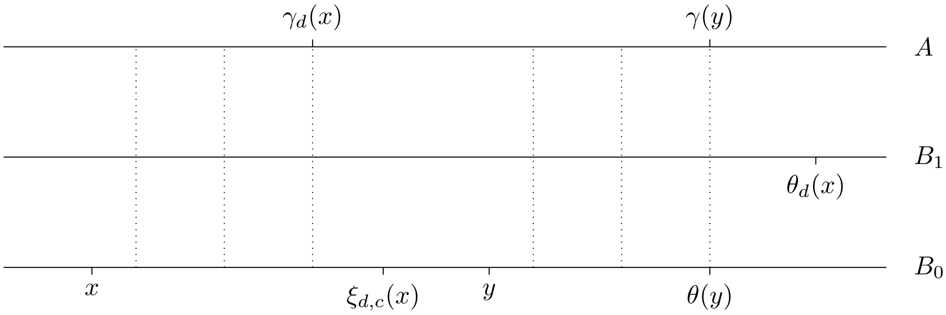


FIGURE 5. Two gates: a typical configuration.

**3.3. Verification.** First, we observe that functionals discussed indeed diverge as promised at the beginning of each cycle. For example, we indeed have $\delta(n)\uparrow$ at the beginning of an attack because we always define $\delta(n) \geqslant \lambda(u_y)$ (which in turn is at least $\lambda(u_x)$), and $\gamma(y) = u_y$ and $\gamma_d(x) = u_x$ whenever they are defined. Similarly, when we return to a set-up and $x$ is not cancelled, it is because $y$ is cancelled; $y$ became unrealised, which means that $\theta_d(x)\uparrow$.

We also observe that the instructions described cover all possible occurences. Consider for example the maintenance cycle. We stipulated that if $x$ or $y$ can be either cancelled or attacked with then we do so (with $x$ having precedence over $y$ in that respect). Suppose that $\delta(n)\uparrow$ during maintenance. If $x$ or $y$ are unrealised, then they are cancelled. Otherwise, at most one of $\gamma(y)\uparrow$ or $\theta(y)\uparrow$, in which case a $y$-layer is peeled; and possibly $\gamma_d(x)\uparrow$ but as $y$ is realised, $\theta_d(x)\downarrow$, so an $x$-layer is peeled.

Also observe that during an attack with $x$, if $x$ becomes unrealised then it is cancelled, as $\delta(n) \geqslant \lambda(\xi_{d,c}(x))$. Similarly, during an attack with $y$, if either $x$ or $y$ becomes unrealised then it is cancelled. And similarly, if $\delta(n)\uparrow$ during maintenance then we never return to the realisation cycle without passing through set-up again.

We note that if we attack with $x$ then we may indeed cancel $y$, as $\theta_d(x)\uparrow$ implies that $y$ is unrealised, and $\gamma_d(x)\uparrow$ implies that $\delta(n)\uparrow$.

Finally note that $\theta_d(x) \geqslant \psi_d(y)$ so if $y$ is realised then $\Psi_d(B_1, y)\downarrow= 0$; if $x$ is realised then $\Psi_c(B_0, x)\downarrow= 0$.

We extend Lemma 2.1.

LEMMA 3.1. *Let $e$ be a construction, $d$ an agent for $e$, and $n$ an anchor for $d$. There is a follower $x$ for $n$ which is never cancelled. There is a last follower $y$ for $n$ which is ever appointed; it is only cancelled if we attack with $x$.*

PROOF. As in the proof of Lemma 2.1, let $s_0$ be a stage after which $f^e(n)$ does not change. Suppose that at some stage $s_1 > s_0$ we are in the set-up cycle. the follower $x$ at that time will never be cancelled. The follower $y$ may be cancelled, but only if one of the sets $A$, $B_0$ or $B_1$ change below $x^{(2n)}$. Eventually, the value of $x^{(2n)}$ stabilizes. $\square$

LEMMA 3.2. *$\Delta(D)$ is total.*

Proof. Let $n < \omega$ be an anchor for some agent $d$ (for a construction $e$). Again we note that $\delta(n)$ is never left undefined at the end of a stage, so we just need to show that the value of $\delta(n)$ is bounded (over all stages).

By Lemma 3.1, let $x$ and $y$ be the last followers appointed for $n$. There are several possibilities for where we can end up.

(1) It is possible to get stuck for ever waiting for realisation for either $x$ or $y$. In this case, we know that $\delta(n)$ can never get undefined after starting the realisation run, as that would cancel $x$ or $y$.
(2) An attack with $x$ or with $y$ is performed. The attack with $y$ can be exited only if we start an attack with $x$ (otherwise, $y$ is cancelled). The attack with $x$ cannot be exited. The value $\theta_d(x)$ is constant during an attack with $y$; the value $\xi_{d,c}(x)$ is constant during an attack with $y$ or with $x$. And $\Lambda(D)$ is total.
(3) It is possible to be left in the set-up cycle, never getting a correct $f^e$ guess. The value of $o^e(n)$ and so of $y^{(k)}$ eventually stabilizes; we again then use the totality of $\Lambda(D)$.
(4) After we enter the maintenance phase, $D\restriction_{\delta(n)}$ never changes. In this case obviously $\delta(n)$ is constant after we enter maintenance.
(5) We enter maintenance with $x$, and at some stage $s_1$ after that we see a $D\restriction_{\delta(n)}$-change. After that stage, $x$ and $y$ are always realised. □

As above we fix $e$ such that $\Delta(D) = f^e$.

Lemma 3.3. *$Q$ is computable from both $A$ and $B_0$.*

Proof. The proof is pretty much identical to the proof of Lemma 2.3: if $y$ a permanent follower for some anchor $n$ for some agent for $e$, then $u_y$ is eventually defined; if we never attack with $y$ then we only leave $\gamma(y)$ or $\theta(y)$ undefined when waiting for agreement between $\Delta(D, n)$ and $f^e(n)$ (after a layer is peeled). □

If $Q \not\leq_\mathrm{T} B_1$ then we are done. Otherwise fix some $d$ such that $\Psi_d(B_1) = Q$.

Lemma 3.4. *$Q_d$ is computable from both $A$ and $B_1$.*

Proof. The proof is slightly more elaborate; let $x$ be a follower for an anchor $n$ for an agent $(d, c)$, and suppose that $x$ is neither cancelled nor attacked with. We consider stages during which $\gamma_d(x)\uparrow$ or $\theta_d(x)\uparrow$.

We possibly have $\gamma_d(x) \uparrow$ while waiting for agreement between $\Delta(D, n)$ and $f^e(n)$. As for $y$, during a realisation cycle, if $\gamma_d(x)\uparrow$ then $\delta(n)\uparrow$ and then we cancel $x$ or $y$; eventually this stops happening. We may also have $\gamma_d(x)\uparrow$ during an attack with some $y$. But such an attack must end, as $\Psi_d(B_1) = Q$. So $\gamma_d(x)$ is defined at all but finitely many stages, and its value is constant $u_x$.

Usually, when $\theta_d(x)\uparrow$ we can cancel $y$. Otherwise, we can have $\theta_d(x)$ while we are waiting for some $y$ to be realised (here it is important that if both $y$ and $x$ are unrealised, we first realise $y$, then $x$); but $\Psi_d(B_1) = Q$ implies that every $y$ is eventually realised or cancelled. There will be a last $y$ appointed, and never cancelled (as we assumed that we do not attack with $x$); and the value $\psi_d(x)$ will eventually stabilise. This implies that the values of $\theta_d(x)$ are bounded. □

If $Q_d \not\leq_\mathrm{T} B_0$ then we are done. Otherwise fix some $c$ such that $\Psi_c(B_0) = Q_d$. We will show that with $\Xi_{d,c}$, $B_0$ correctly computes $B_1$. As in the simpler construction, we need to show that if $x$ is a follower for some anchor for the agent $(d, c)$, and $x$ is never cancelled, then eventually we define a computation $\Xi_{d,c}(B_0, x)$ which always converges, and that $B_1 \upharpoonright_x$ is constant from the stage at which this computation is defined. Fix such $x$. The argument of the simpler construction shows that $\xi_{d,c}(x)$ is bounded and defined at infinitely many stages. We only need to notice that if $y$ is the last follower appointed for $x$'s anchor, then every realisation cycle that we enter after appointing $y$ must be exited, as both $\Psi_d(B_1) = Q$ and $\Psi_c(B_0) = Q_d$.

So it all comes down to correctness, which as above is the heart of the argument. Let $s^*$ be the stage at which the permanent computation $\Xi_{d,c}(B_0, x)$ is defined. For $k \leqslant 2n$ let $v_k = x^{(k)}$ as calculated at stage $s^*$. As $x$ is not cancelled, $\delta(n)\downarrow$ at all stages from the end of the set-up of $x$ and stage $s^*$; it follows that $u_x = v_{2n}$.

The key observation is that the peeling of the $x$-layers has to alternate between $A$ and $B_1$. For $k \leqslant n$ let $s_k$ be the least stage $s \geqslant s^*$ such that $B_{1,s+1} \upharpoonright_{v_{2k}} \neq B_{1,s} \upharpoonright_{v_{2k}}$; otherwise let $s_k = \infty$. By induction on $s \in [s^*, s_k]$ we see that $v_{2k-1} = x^{(2k-1)}$ at stage $s$ and that $A_s \upharpoonright_{v_{2k-1}} = A_{s^*} \upharpoonright_{v_{2k-1}}$; but $A_{s_k} \upharpoonright_{v_{2k+1}} \neq A_{s^*} \upharpoonright_{v_{2k+1}}$. Let $t_k$ be the least stage $t \geqslant s^*$ such that $A_{t+1} \upharpoonright_{v_{2k-1}} \neq A_{s^*} \upharpoonright_{v_{2k-1}}$; the fact that we never attack with $x$ implies that $s_n < t_n < s_{n-1} < t_{n-1} < \cdots$.

Lemma 3.5. *For all $k < n$ such that $s_k < \infty$,*

$$o_{s_k}(n) < \omega k \tag{2}$$

*(where $o = o^e$).*

The inequality will imply that $s_0$ must equal $\infty$, and so $B_{1,s^*} \upharpoonright_x = B_1 \upharpoonright_x$ as required.

Proof. Since we start with $o_0(n) = \omega n$ and we redefine $\Delta(D, n)$ when setting $x$ up, we have $o_{s_n}(n) < \omega n$; so Eq. (2) holds for $k = n$.

We prove Eq. (2) by induction on $k$. Fix $k \leqslant n$ such that $s_{k-1} < \infty$, and suppose that $o_{s_k} < \omega k$. Since $v_{2k} \leqslant u_x \leqslant \theta_d(x)$, $y$ is unrealised at stage $s_k$ and $\delta(n)\uparrow$ at that stage; so we cancel $y$ at stage $s_k$. At stage $t_k$ we must have $\theta_d(x)\downarrow$, since otherwise we attack with $x$ at that stage. So there is some last stage $r_k \in (s_k, t_k)$ at which we realise a follower $y = y_k$. The familiar argument shows that at stage $r_k$ we have $u_y = y^{(m)}$ where $o_{r_k}(n) = \omega(k-1) + m'$ for some $m' < m$ (we may assume that $o_{r_k}(n) \geqslant \omega(k-1)$, otherwise we are done for this inductive step). The follower $y_k$ is not cancelled before stage $t_k$. An important point is that we do not attack with $y_k$ before or at stage $t_k$. To see this, observe that every attack with $y_k$ must eventually fail, and $y_k$ is then cancelled; so this failure does not happen before stage $t_k$. But then, as $\gamma_d(x)\uparrow$ at stage $t_k$, it remains undefined until the attack with $y$ fails — and then we would attack with $x$.

At stage $t_k$ we do not start an attack with $x$ so at that stage $\theta_d(x)\downarrow$ (and recall that $\theta_d(x) \geqslant u_y$). We do not start an attack with $y$ at that stage, whereas $\gamma(y)\uparrow$ at $t_k$; so $\theta(y)\downarrow$ at $t_k$. So $y^{(1)} > u_y$ at stage $t_k$. The only way this could happen is that between stages $r_k$ and $t_k$, all the layers between $y$ and $u_y$ were peeled. Each time this happens we extract another $\Delta(D, n)$ change; we have $m$ such changes, which drives the ordinal $o_{t_k}(n)$ below $\omega(k-1)$ as required. □

## 4. The general construction

No new ideas are required for the general construction. The general idea that if we guess that $\Delta(D)$ is $\omega^m$-c.a. then we set up $m$ many followers. We go straight to the details. We are presented with a c.e. set $D$ of totally $< \omega^\omega$-c.a. degree, three c.e. sets $A, B_0$ and $B_1$, and reductions $\Lambda(D) = (A, B_0, B_1)$, $\Phi(B_0, B_1) = A$, and $\Phi_i(A, B_{1-i}) = B_i$ for $i = 0, 1$. For $x < \omega$ we define $x^{(1)}$ and $x^{(n)}$ as in the previous section.

For $m < \omega$ define $h_m(n) = \omega^m \cdot n$. Every function computable from $D$ is $h_m$-c.a. for some $m$. Fix (uniformly in $m$) an effective list $\langle f^{e,m}, o^{e,m}\rangle$ of all $h_m$-c.a. functions, with the usual tidy approximations. For simplicity of notation we will only use odd $m$'s. We enumerate a functional $\Delta$; a construction $(e, m)$ for $e < \omega$ and odd $m$ will guess that $\Delta(D) = f^{e,m}$. Agents for the $(e, m)^{\text{th}}$-construction are indexed by $m+1$-tuples $\bar{d} = (d_0, d_1, \dots, d_m)$ of natural numbers. For each sequence $\bar{c}$ of length at most $m$ the construction enumerates a c.e. set $Q_{\bar{c}} = Q^{e,m}_{\bar{c}}$ and functionals $\Gamma_{\bar{c}} = \Gamma^{e,m}_{\bar{c}}$ and $\Theta_{\bar{c}} = \Theta^{e,m}_{\bar{c}}$; we plan for $\Gamma_{\bar{c}}(A) = Q_{\bar{c}}$ and for $\Theta_{\bar{c}}(B_{[\bar{c}]}) = Q_{\bar{c}}$, where we let $[\bar{c}] = |\bar{c}| \mod 2$. For simplicity we will also write $[k]$ for $k \mod 2$. Each agent $\bar{d}$ defines a functional $\Xi_{\bar{d}} = \Xi^{e,m}_{\bar{d}}$, hoping that $\Xi_{\bar{d}}(B_0) = B_1$ (if $m$ were even we would need to exchange $B_0$ and $B_1$, all the rest would be identical). We write $\theta, \gamma, Q$ for $\theta_{\langle\rangle}, \gamma_{\langle\rangle}, Q_{\langle\rangle}$.

An agent $\bar{d}$ will appoint anchors $n$, inputs for $\Delta(D)$. Each anchor will try to appoint a sequence of followers $x_m < x_{m-1} < \cdots < x_1 < x_0$, with $x_k$ targeted for $Q_{\bar{d}\restriction k}$. When a follower $x_k$ is cancelled or attacked with, we cancel all the larger followers $x_{k'}$ for $k' < k$. The main idea will be to ensure that if $o^{e,m}(n) = \omega^m p_m + \omega^{m-1} p_{m-1} + \cdots + \omega p_1 + p_0$ then $x_{k-1}$ will bound at least $p_k$ many layers above $x_k$.

To streamline the description of the construction we define, for $k = 0, \dots, m-1$, $\chi(x_k) = \theta_{\bar{d}\restriction k+1}(x_{k+1})$; and define $\chi(x_m) = \xi_{\bar{d}}(x_m)$. See Figure 6. We will say that the follower $x_k$ is *realised* if $\chi(x_k)\downarrow$.

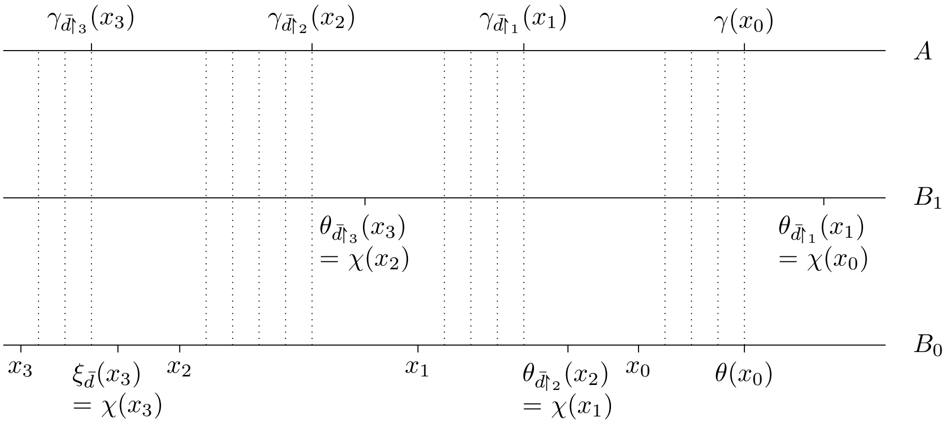


FIGURE 6. Four gates: a typical configuration.

As before, we say that we *may attack* with a follower $x_k$ if it is realised, and both $\gamma_{\bar{d}\restriction k}(x_k)\downarrow$ and $\theta_{\bar{d}\restriction k}\downarrow$. We say that a follower $x_k$ is *confirmed* if the set-up cycle at which it was appointed has already finished. We *may cancel* a confirmed follower $x_k$ if it is unrealised and $\delta(n)\uparrow$. Throughout the construction, if we may cancel a follower or attack with it then we do so, always choosing the smallest follower (the one with largest index) with which to attack or cancel. If we cancel a follower and do not start an attack, then we return to the set-up cycle.

We now describe the procedures undertaken by an anchor $n$.

SET-UP.

**1.** Say that $x_m, x_{m-1}, \ldots, x_{k+1}$ are defined and confirmed. We appoint new followers $x_k < x_{k-1} < x_{k-2} < \cdots < x_0$ so that $x_k > u_{k+1}$, and for all $j < k$, $x_j > x_{j+1}^{(2p_{j+1})}$, where currently $o^{e,m}(n) = \omega^m p_m + \cdots + \omega p_1 + p_0$. We then define $\delta(n) = \lambda(x_0^{(p_0)})$, and wait for $\Delta(D, n) = f^{e,m}(n)$. While waiting, we update the values of $x_j$ for $j < k$ and of $\delta(n)$ to keep the desired inequalities. We do so in a conservative way: only cancel $x_j$ if there is a change in $A$, $B_0$ or $B_1$ below $x_{j+1}^{(2p_{j+1})}$.

**2.** Once we see that $\Delta(D, n) = f^{e,m}(n)$ we define for all $j = 1, \ldots, k$, $u_j = x_j^{(2p_j)}$, and define $u_0 = x_0^{(p_0)}$. For each $j$ such that $\gamma_{\bar{d}\upharpoonright j}(x_j)\uparrow$ we define this marker to equal $u_j$. We also define $\theta(x_0) = u_j$.

We move to realisation.

REALISATION.

For each $k \leqslant m$, if $x_k$ is unrealised, wait for $\Psi_{d_k}(B_{1-[k]}, x_k)\downarrow = 0$. Once this is observed we define $\chi(x_k) = \max\{u_k, \psi_{d_k}(x_k)\}$. That is, we define $\theta_{\bar{d}\upharpoonright k+1}(x_{k+1})$ or $\xi_{\bar{d}}(x_m)$ depending if $k = m$ or $k < m$.

Note that the search is done in parallel, and we define $\chi(x_k)$ immediately when the realising computation is discovered. Once all followers are realised we move to maintenance.

MAINTENANCE.

We wait for a change in $D$ below $\delta(n)$ or for some follower to become unrealised. When this occurs:

(a) If a follower is unrealised, move to realisation. As above this assumes that $\delta(n)\downarrow$, otherwise we would cancel the follower.

(b) If either $\gamma(x_0)\uparrow$ or $\theta(x_0)\uparrow$, but not both, redefine

$$\delta(n) = \lambda\left(\max\{\chi(x_k) \,:\, k \leqslant m\}\right).$$

Wait for $\Delta(D, n) = f^{e,m}(n)$. While waiting, if $\delta(n)\uparrow$ (but no attack or cancellation are possible) then we just redefine it by the same formula. When $\Delta(D, n) = f^{e,m}(n)$ is observed we redefine all the markers $\gamma(x_k)$ and $\theta(x_0)$ which are undefined (with value $u_k$).

(c) If only $\delta(n)\uparrow$ then we redefine $\delta(n) = \lambda\left(\max\{\chi(x_k) \,:\, k \leqslant m\}\right)$ and stay at the maintenance phase.

ATTACK WITH $x_k$.

We enumerate $x_k$ into $Q_{\bar{d}\upharpoonright k}$. We define $\delta(n) = \lambda\,(\max\{\chi(x_j)\ :\ j \geqslant k\})$. We wait for changes. If $\delta(n)\uparrow$ we redefine it according to the formula above. If $\gamma_d(x_j)\uparrow$ for some $j < k$ we leave it undefined.

**4.1. Verification.** The verification is identical to the two-gate case and so we omit it.

CHAPTER VIII

# Prompt permissions

In this chapter we consider prompt versions of the permitting notions we investigated in this monograph. These can be used to obtain results more akin to what is obtained in constructions. For example, in the usual embedding of the 1-3-1 lattice one gets the bottom element to be $\mathbf{0}$; this does not appear to be consistent with permitting at the level of non-total $< \omega^\omega$-c.a.-ness, but can be done if the permitting is obtained promptly.

## 1. Prompt classes

Recall that a c.e. set $A$ *permits promptly* if is has a enumeration $\langle A_s\rangle$ such that for some computable function $p \geqslant \mathrm{id}$, for any $e$, if $W_e$ is infinite then there is some $n$ which enters $W_e$ at some stage $s$ such that $A_s \upharpoonright_n \neq A_{p(s)} \upharpoonright_n$. This notion is invariant under Turing equivalence; a degree permits promptly if and only if it contains a promptly simple set; see [2]. Prompt permitting is the prompt version of simple permitting; a set which permits promptly is in some sense promptly non-computable.

For considering the prompt version of non-total $\alpha$-c.a. permitting, fix an effective listing $\langle f^{e,\alpha}\rangle$ of all $\alpha$-c.a. functions, each equipped (uniformly) with tidy $(\alpha+1)$-computable approximations $\langle f^{e,\alpha}_s, o^{e,\alpha}_s\rangle$ as in Proposition II.1.7. We will shortly use more properties of this list. However to motivate these properties we first give our definitions.

Definition 1.1. Call a function $g$ *self-modulating* if there is a computable approximation $\langle g_s\rangle$ of $g$ such that:

- for all $s$ and $n$, $g_s(n) \leqslant s$;
- for all $s$ and $n$, if $g_s(n) \neq g_{s-1}(n)$ then $g_s(n) = s$ and in fact for all $m \geqslant s$, $g_s(m) = s$.

It follows that for all $s$, $g_s \leqslant g_{s+1}$ (pointwise) and that if $g_s(n) \neq g_{s-1}(n)$ then $g_s(m) \neq g_{s-1}(m)$ for all $m \geqslant n$. The idea is that $g$ is the modulus of the approximation $\langle g_s\rangle$. Above we used the fact that if $\mathbf{d}$ is c.e. but not totally $\alpha$-c.a. then there is a self-modulating function $g \in \mathbf{d}$ which is not $\alpha$-c.a. Note that every self-modulating function has a c.e. degree. Below we assume that each self-modulating function $g$ "comes with" the approximation $\langle g_s\rangle$ of which it is the modulus.

Definition 1.2. A *speed-up function* is a non-decreasing, computable function $p$ such that $p(n) \geqslant n$ for all $n$.

Definition 1.3. Let $g$ be a self-modulating function and let $p$ be a speed-up function. Let $n < \omega$. Let $\langle f_s, o_s\rangle$ be a tidy $(\alpha+1)$-coputable approximation. We say that $g$ promptly $p$-escapes $\langle f_s, o_s\rangle$ on input $n$ if for all $s$, if $o_s(n) < \alpha$ and

$f_s(n) = g_s(n)$ then $g_{p(s)}(n) \neq g_s(n)$. We say that $g$ promptly $p$-escapes $\langle f_s, o_s\rangle$ if it promptly $p$-escapes it on some input.

A self-modulating function $g$ is *promptly not $\alpha$-c.a.* if there is some speed-up function $p$ such that $g$ promptly $p$-escapes each $\langle f_s^{e,\alpha}, o_s^{e,\alpha}\rangle$.

A c.e. degree $\mathbf{d}$ is *promptly not totally $\alpha$-c.a.* if there is a self-modulating function $g \leqslant_{\mathrm{T}} \mathbf{d}$ which is promptly not $\alpha$-c.a.

Note that if an approximation $\langle f_s^{e,\alpha}, o_s^{e,\alpha}\rangle$ is not eventually $\alpha$-computable then vacuously, for almost all $n$, $g$ promptly $p$-escapes this approximation on $n$; the power of promptness is when it is applied to "total" approximations (approximations which are eventually $\alpha$-computable).

**1.1. Slow-down lemma.** Recall how prompt permitting is used in constructions. Suppose for example that we want to show that a promptly permitting degree $\mathbf{d}$ is not half of a minimal pair. Let $D \in \mathbf{d}$ and let $B$ be c.e. and non-computable. We build a c.e. set $Q$ computable from both $D$ and $B$ and plan to make $Q$ non-computable. To diagonalise against the $e^{\text{th}}$ computable set, a Friedberg-Muchnik requirement appoints a follower $x$ and waits for it to be realised ($\varphi_e(x){\downarrow}= 0$). When it is realised we wait for simple permitting from $B$; $B_{s+1}\restriction_x \neq B_s\restriction_x$. When we see this we ask for prompt permission from $D$, namely $D_{p(s)}\restriction_x \neq D_s\restriction_x$. If both are granted then we can enumerate $x$ into $Q$ and meet the requirement. Why will permission be granted? Of course we potentially appoint infinitely many followers. Since $B$ is non-computable, infinitely many of them will be permitted by $B$. Let $U_e = W_{g(e)}$ be the c.e. set of followers for this requirement which will be permitted by $B$. Applying prompt permission to this set $U_e$ guarantees prompt permission from $D$ for one of the followers in $U_e$.

This sketch of an argument involved a little cheating. While indeed we know, by the recursion theorem, an index $g(e)$ for $U_e$, the effective enumeration of $W_{g(e)}$ may be different from our enumeration of $U_e$. We put $x$ into $U_e$ at the stage at which $B$ permits $x$. It is conceivable that $x$ is enumerated into $W_{g(e)}$ at an earlier stage; so the prompt permission for $x$ was given in the past, and is useless for us now. We need to find $g(e)$ such that not only $W_{g(e)} = U_e$ but every number enters $W_{g(e)}$ not before we put it into $U_e$.

This "slow-down lemma" can be obtained by a more sophisticated use of the recursion theorem (see [57, Thm.XII.1.5]). Actually this is not quite necessary. Interpret the $e^{\text{th}}$ partial computable function $\varphi_e$ as a function of two variables. We can transform this function into an effective enumeration of a c.e. set (call it $W_e$) such that if $\varphi_e$ is an effective enumeration $\langle V_{e,s}\rangle$ of a c.e. set $V_e$ (that is, $\varphi_e$ is total and for all $s$, $\varphi_e(-, s)$ is the characteristic function of $V_{e,s}$) then $W_e = V_e$ and further, for all $s$, $W_{e,s} \subseteq V_{e,s}$. Namely, we put $x$ into $W_e$ at stage $s$ if at that stage we have seen sufficiently much convergence from $\varphi_e$ to see that $x \in V_e$. The slow-down lemma can now be obtained by using the recursion theorem to obtain an index $g(e)$ such that $\varphi_{g(e)}$ is our enumeration of $U_e$; we then apply prompt permitting to $W_{g(e)}$.

In our usage of prompt permitting of the form given by Definition 1.3 we need a similar form of a slow-down lemma. Namely, to force changes we will define, for some requirement, an $\alpha$-computable approximation $\langle h_s\rangle$ attempting to trail the function $g$ given by the definition, and ask for immediate changes in $g$. To do this we will need to find one of the functions $f^{e,\alpha}$ on the list such that for all $n$, for all $s$ there is some $t \geqslant s$ such that $f_t^{e,\alpha}(n) = h_s(n)$. To obtain this we follow the construction proving Proposition II.1.7. Using the notation of the proof of that proposition, we think of $\varphi_e$ as giving the sequence $\langle h_s, m_s\rangle$ which we transform into the partial approximation $\langle f_s^{e,\alpha}, o_s^{e,\alpha}\rangle$, making sure that as long as $\langle h_s, m_s\rangle$ appears to be a tidy $(\alpha+1)$-computable approximation, we copy every value that shows up. It is this sequence of approximations that we use in Definition 1.3. This sequence will be acceptable in a strong way.

Call a pair $\langle h_s, m_s\rangle$ of partial computable functions a *partial tidy* $(\alpha+1)$-*computable approximation* if for all $x$ and $s$, $h_s(x)\downarrow \Leftrightarrow m_s(x)\downarrow$ and if so, for all $y \leqslant x$ and $r \leqslant s$, $h_r(y)\downarrow$ and the array $\langle h_r(y), m_r(y)\rangle_{r\leqslant s, y\leqslant x}$ satisfies the conditions for being an initial segment of such an approximation: that is, $m_0(y) = \alpha$, $m_r(y) \leqslant \alpha$, $h_0(y) = 0$, $m_{r(y)} \leqslant m_{r-1}(y)$, and if $h_r(y) \neq h_{r-1}(y)$ then $m_r(y) < m_{r-1}(y)$. The sequence $\langle f_s^{e,\alpha}, o_s^{e,\alpha}\rangle$ is acceptable in the following sense:

- if $\langle h_s^e, m_s^e\rangle_{e,s<\omega}$ is a sequence of (uniformly) partial tidy $(\alpha+1)$-computable approximations then there is a computable function $k$ (obtained uniformly from an index for the sequence) such that for all $e$, $x$ and $s > 0$, if $h_s^e(x)\downarrow$ then there is some $t \geqslant s$ such that $o_t^{k(e),\alpha}(x) = m_s^e(x)$ and $f_t^{k(e),\alpha}(x) = h_s^e(x)$.

In particular, for each $e$, if $\langle h_s^e, m_s^e\rangle$ is a (total) $\alpha$-computable approximation, then $\left\langle f_s^{k(e),\alpha}, o_s^{k(e),\alpha}\right\rangle$ is eventually $\alpha$-computable and further, for all $n$ and $s$ there is some $t \geqslant s$ such that $h_s^e(n) = f_t^{k(e),\alpha}(n)$.

Finally, in some arguments it would be useful to assume that like the enumeration of the sets $W_e$, at each stage $s$ we have only said finitely much about all functions. Formally,

- For all $s$, $e$ and $n$, $f_s^{e,\alpha}(n) \leqslant s$, and $o_s^{e,\alpha}(n) < \alpha$ implies $e, n < s$.

**1.2. Counting down $\alpha$.** The functions $f_s^{e,\alpha}$ are not really important for promptness; it is the ways $o_s^{e,\alpha}$ of counting down $\alpha$ that we need to escape.

Definition 1.4. A *counting down* $\alpha$ is a sequence of uniformly computable functions $\langle o_s\rangle$ from $\omega \to \alpha+1$ such that for all $n$, $o_0(n) = \alpha$; $o_s(n) = \alpha$ if $s \leqslant n$; $o_s(n) \leqslant o_{s-1}(n)$ for all $n$ and $s$; and if $o_s(n) < \alpha$ then $o_s(n-1) < \alpha$ as well.

In other words, it appears as the ordinal part in a tidy $(\alpha+1)$-computable approximation $\langle f_s, o_s\rangle$.

Definition 1.5. Let $g$ be a self-modulating function and let $p$ be a speed-up function; let $\langle o_s\rangle$ be a counting down $\alpha$. We say that $g$ promptly $p$-escapes $\langle o_s\rangle$ on an input $n$ if for all $s > 0$, if $o_s(n) \neq o_{s-1}(n)$ then $g_{p(s)}(n) \neq g_s(n)$. We say that $g$ promptly $p$-escapes $\langle o_s\rangle$ if it does so on some input.

Lemma 1.6. *Let $g$ be a self-modulating function. Then $g$ is promptly not $\alpha$-c.a. if and only there is a speed-up function $q$ such that $g$ promptly $q$-escapes each $\langle o_s^{e,\alpha}\rangle$.*

One direction is short.

Lemma 1.7. *Let $\langle f_s, o_s\rangle$ be a tidy $(\alpha+1)$-computable approximation such that $f_s(n) \leqslant s$ for all $s$ and $n$. Suppose that a self-modulating function $g$ promptly $p$-escapes $\langle o_s\rangle$ on input $n$. Then it also promtply $p$-escapes $\langle f_s, o_s\rangle$ on input $n$.*

Proof. Suppose that $o_s(n) < \alpha$ and that $f_s(n) = g_s(n)$. Let $t \leqslant s$ be the least such that $o_t(n) = o_s(n)$. So $f_s(n) = f_t(n)$. Since $f_t(n) \leqslant t$ we see that

$g_s(n) \leqslant t$; since $g$ is self-modulating, this implies that $g_t(n) = g_s(n)$. By assumption, $o_t(n) \neq o_{t-1}(n)$, and so $g_{p(t)}(n) \neq g_t(n) = g_s(n)$. But $p(t) \leqslant p(s)$ and $g$ is non-decreasing so $g_{p(s)}(n) \geqslant g_{p(t)}(n) > g_s(n)$ as required. □

Proof of Lemma 1.6. One direction is provided by Lemma 1.7 and one of our conditions on the listing of approximations $\langle f_s^{e,\alpha}, o_s^{e,\alpha}\rangle$. In the other direction suppose that $p$ witnesses that $g$ is promptly not $\alpha$-c.a. For brevity we write $f_s^e$ and $o_s^e$ for $f_s^{e,\alpha}$ and $o_s^{e,\alpha}$. For each $e$ we define an approximation $\langle h_s^e\rangle$ which chases $g$ as much as $o^e$ allows it. Namely, we define

$$h_s^e(n) = \begin{cases} 0, & \text{if } s = 0;\\ h_{s-1}^e(n), & \text{if } s > 0 \text{ and } o_s^e(n) = o_{s-1}^e(n); \text{ and}\\ g_s(n), & \text{otherwise.}\end{cases}$$

The approximation $\langle h_s^e, o_s^e\rangle$ is $(\alpha + 1)$-computable and tidy. By the $\alpha$-slow-down lemma find some computable function $k$ such that for all $e$, $n$ and $s$ there is some $t = t(e, n, s) \geqslant s$ such that $o_t^{k(e)}(n) = o_s^e(n)$ and $f_t^{k(e)}(n) = h_s^e(n)$. For $s < \omega$ define $t^*(s) = \max\{t(e, n, s) : e, n \leqslant s\}$, and let $q(s) = p(t^*(s))$.

Fix $e$. There is some $n$ such that $g$ promptly $p$-escapes $\left\langle f_s^{k(e)}, o_s^{k(e)}\right\rangle$ on input $n$. We claim that $g$ promptly $q$-escapes $\langle o_s^e\rangle$ on input $n$. For let $s > 0$ be a stage such that $o_s^e(n) \neq o_{s-1}^e(n)$. Then $h_s^e(n) = g_s(n)$; so $f_t^{k(e)}(n) = g_s(n)$ for $t = t(e, n, s)$. We need to show that $g_{q(s)}(n) \neq g_s(n)$. Note that $o_s^e(n) < \alpha$ implies that $e, n < s$, so $t \leqslant t^*(s)$. If $g_t(n) \neq g_s(n)$ then we are done, as $q(s) \geqslant t$. Otherwise $f_t^{k(e)}(n) = g_t(n)$ (and $o_t^{k(e)}(n) = o_s^e(n) < \alpha$) so by our assumption, $g_{p(t)}(n) \neq g_t(n)$; but $q(s) \geqslant p(t)$. □

Therefore for the purposes of promptness we from now on ignore the function part $f_s$. We state the slow-down lemma in this context. As expected, define a *partial counting down* $\alpha$ to be a partial computable sequence $\langle o_s\rangle$ such that for all $s$ and $x$: (a) if $o_s(x)\downarrow$ then $o_s(x) \leqslant \alpha$ and $o_t(y)\downarrow$ for all $t \leqslant s$ and $y \leqslant x$; (b) if $o_0(x)\downarrow$ then $o_0(x) = \alpha$; (c) if $s > 0$ and $o_s(x)\downarrow$ then $o_s(x) \leqslant o_{s-1}(x)$; if $o_s(x)\downarrow$, $y < x$ and $o_s(y) = \alpha$ then $o_s(x) = \alpha$.

Lemma 1.8. *Suppose that $\langle m_s^e\rangle$ is a uniform sequence of partial countings down $\alpha$. There is a computable function $k$ such that for all $e$, $s$ and $x$, if $m_s^e(x)\downarrow$ then there is some $t \geqslant s$ such that $o_t^{k(e),\alpha}(x) = m_s^e(x)$. The function $k$ can be obtained effectively.*

We can conclude that promptness does not really depend on the choice of list $\langle o_s^{e,\alpha}\rangle$ (as long as it is acceptable).

Corollary 1.9. *Suppose that $\langle m_s^e\rangle$ is a uniformly computable sequence of (total) countings down $\alpha$ such that for all $s$ the set $\{(e, x) : m_s^e(x) < \alpha\}$ is bounded, computably in $s$. Suppose that a function $g$ is promptly not $\alpha$-c.a. Then there is a speed-up function $q$ such that $g$ promptly $q$-escapes $\langle m_s^e\rangle$ for each $e$.*

Proof. As in the proof of Lemma 1.6 let $t^*(s)$ be a bound on stages $t = t(e, x, s) \geqslant s$ such that $o_t^{k(e)}(x) = m_s^e(x)$ for all $e, x$ such that $m_s^e(x) < \alpha$ (where $k$ is given by the slow-down Lemma 1.8; as above $o^e = o^{e,\alpha}$). Suppose that $p$ witnesses that $g$ is promptly not $\alpha$-c.a.; let $q(s) = p(t^*(s))$. To see that this works, suppose that $g$ promptly $p$-escapes $\left\langle o_s^{k(e)} \right\rangle$ on an input $x$. Let $s > 0$ and suppose that $m_s^e(x) \neq m_{s-1}^e(x)$. Then $m_s^e(x) < \alpha$, so $t^*(s) \geqslant t(e,x,s)$. Let $u$ be the least such that $o_u^{k(e)}(x) = m_s^e(x)$; so $u \leqslant t(e,x,s)$. But also $u > t(e,x,s-1) \geqslant s-1$ so $u \geqslant s$. By assumption, $g_{p(u)}(x) \neq g_u(x)$, and $q(s) \geqslant p(u)$. ☐

We can escape infinitely many inputs.

Lemma 1.10. *Suppose that $g$ is promptly not $\alpha$-c.a. Then there is some speed-up function $q$ such that for all $e$ there are infinitely many $x$ such that $g$ promptly $q$-escapes $\langle o_s^{e,\alpha} \rangle$ on input $x$.*

Proof. Note that the first attempt that comes to mind to prove this does not work. Non-uniformly we could guess an initial segment of $g$ and change an approximation to make sure that permission is not given on the first $n$ locations. But there are infinitely many possible initial segments of a fixed finite length, and we cannot define our speed-up taking into account all of them (see the proof of [57, Thm.XII.1.7(iii)]). What we do is shift by $n$.

Namely, for all $e$ and $n$ define $m_s^{e,n}(x) = o_s^e(x+n)$ (for brevity let $o_s^e = o_s^{e,\alpha}$). Note that $m_s^{e,n}(x) < \alpha$ implies $e, x, n < s$. Let $q$ be given by Corollary 1.9. Suppose that $g$ promptly $q$-escapes $\langle m_s^{e,n} \rangle$ on input $x$; we conclude that $g$ promptly $q$-escapes $\langle o_s^e \rangle$ on input $x+n$, the reason being that if $g_{q(s)}(x) \neq g_s(x)$ then $g_{q(s)}(y) \neq g_s(y)$ for all $y > x$.

☐

The proof of this lemma shows that we can effectively, given a uniform list $\langle m_s^e \rangle$ of tidy $(\alpha+1)$-computable approximation and a speed up-function $p$ such that $g$ promptly $p$-escapes each $\langle m_s^e \rangle$, find a speed-up function $q$ such that $g$ promtply $q$-escapes each $\langle m^{e,s} \rangle$ on infinitely manu inputs.

**1.3. Powers of $\omega$.** Let $\alpha \leqslant \epsilon_0$. For brevity let $\mathsf{PN}(\alpha)$ denote the class of degrees which are promptly not totally $\alpha$-c.a.

Lemma 1.11. *If $\beta < \alpha$ then every function which is promptly not $\alpha$-c.a. is also promptly not $\beta$-c.a.*

Hence $\mathsf{PN}(\alpha) \subseteq \mathsf{PN}(\beta)$.

Proof. Define $m_t^e(x) = o_t^{e,\beta}(x)$ if this value is smaller than $\beta$; otherwise let $m_t^e(x) = \alpha$. Now apply Corollary 1.9. ☐

Proposition 1.12. *Suppose that $g$ is promptly not $\alpha$-c.a. Then for all $m < \omega$, $g$ is promptly not $\alpha \cdot m$-c.a.*

So $\mathsf{PN}(\gamma) = \mathsf{PN}(\alpha)$ for all $\gamma \in [\alpha, \alpha\cdot\omega)$. As for the non-prompt case, this means that each prompt class is $\mathsf{PN}(\alpha)$ for an ordinal $\alpha$ which is a power of $\omega$. Below we will see that this is sharp.

Proof. We need to uniformise Lemma III.2.2. We define a list $\left\langle m_s^{e,k} \right\rangle_{e<\omega, k<m}$ of countings down $\alpha$. We claim that by the recursion theorem we have a speed-up function $q$ such that $g$ promptly $q$-escapes each $\left\langle m_s^{e,k} \right\rangle$ (and further we require that this happens on infinitely many inputs).

Actually this relies on a property of the construction. By stage $s$ we will have already defined $m_r^{e,k}$ for all $r < s$ (for all $e$ and $k < m$). The finiteness condition of

Corollary 1.9 will be obtained by ensuring that $m_r^{e,k}(x) = \alpha$ unless $e, x < r$. During stage $s$ we define the functions $m_s^{e,k}$, but in the process of doing so we only consult $q$ on values strictly smaller than $s$. Then the fact that $m_s^{e,k}$ is defined for all $s$ implies that $q(s)$ is defined (as in the proof of Corollary 1.9), and the construction can proceed to the next stage.

The counting $m_s^{e,k}$ guesses that $k = k^*$ (in the notation of Lemma III.2.2). However it is not sufficient for $g$ to escape $\langle o_s^{e,\alpha m}\rangle$ on some input only from the stage at which $o_s^{e,\alpha m}(x) < \alpha(k+1)$; we need it to escape earlier as well. So it looks for inputs which have already been escaped up to that point (using $q$) and only copies them. Inductively, Lemma 1.10 says there will be infinitely many such inputs.

Now to the details. To define $m_s^{e,k}(x)$ we search for some $y \geqslant x$ such that:

- $o_s^{e,\alpha m}(y) \in [\alpha k, \alpha(k+1))$ but $o_{s-1}^{e,\alpha m}(y) \geqslant \alpha(k+1)$ (note that this implies $y < s$); and
- for all $t < s$ at which $o_t^{e,\alpha m}(y) \neq o_{t-1}^{e,\alpha m}(y)$ we have $g_{q(t)}(y) \neq g_t(y)$.

If such $y$ is found then we declare $y = y^{e,k}(x)$ and $s = s^{e,k}(x)$. If such $y$ is never found we let $s^{e,k}(x) = \omega$. Now we can define:

$$m_t^{e,k}(x) = \begin{cases} \alpha, & \text{if } t < s^{e,k}(x); \\ \beta, & \text{if } t \geqslant s^{e,k}(x) \text{ and } o_t^{e,\alpha m}(y^{e,k}(x)) = \alpha k + \beta; \text{ and} \\ 0, & \text{if } t \geqslant s^{e,k}(x) \text{ and } o_t^{e,\alpha m}(y^{e,k}(x)) < \alpha k. \end{cases}$$

Fix $e$. For $k \leqslant m$ we let $I_k = I_k^e$ be the set of inputs $x$ such that for all $s$ such that $o_s^{e,\alpha m}(x) \neq o_{s-1}^{e,\alpha m}(x)$ and $o_s^{e,\alpha m}(x) \geqslant \alpha k$, we have $g_{q(s)}(x) \neq g_s(x)$. Vacuously we have $I_m = \omega$; and our aim is to show that $I_0$ is nonempty. In fact we show by decreasing induction on $k = m, m-1, \dots, 0$ that each $I_k$ is infinite.

Let $k < m$ and suppose that we know that $I_{k+1}$ is infinite. There are two cases. It is possible that for almost all $x \in I_{k+1}$, for all $s$, $o_s^{e,\alpha m}(x) \geqslant \alpha(k+1)$. Each such $x$ is in $I_k$ (in fact in $I_0$). Otherwise, for all $x < \omega$, $s^{e,k}(x)$ (and $y^{e,k}(x)$) are defined. There are infinitely many $x$ on which $g$ promptly $q$-escapes $\langle m_s^{e,k}\rangle$. Let $x$ be such an input and let $y = y^{e,k}(x)$, $s^* = s^{e,k}(x)$. So $y \in I_{k+1}$ and we claim that in fact $y \in I_k$: if $s \geqslant s^*$, $o_s^{e,\alpha m}(y) \geqslant \alpha k$ and $o_s^{e,\alpha m}(y) \neq o_{s-1}^{e,\alpha m}(y)$ then $m_s^{e,k}(x) \neq m_{s-1}^{e,k}(x)$ and so $g_{q(s)}(x) \neq g_s(x)$; since $y \geqslant x$, $g_{q(s)}(y) \neq g_s(y)$. □

**1.4. Relation to prompt simplicity.** A counting $\langle o_s\rangle$ down the ordinal 1 is essentally a computable function. Namely let $h(n)$ be the unique stage $s$ such that $o_s(n) = 0$ but $o_{s-1}(n) = 1$. The domain of $h$ is an initial segment of $\omega$. As mentioned above, the property $\mathsf{PN}(1)$ can be thought of as being "promptly non-computable": it forces that $g(n) \neq h(n)$, and this is observed promptly.

LEMMA 1.13. *A c.e. degree is promptly simple if and only if it is in* $\mathsf{PN}(1)$.

PROOF. Suppose that $A$ permits promptly; let $\langle A_s\rangle$ be an enumeration of $A$ which witnesses this fact. Let $g$ be the modulus of the enumeration of $A$: $g_s(n) = t$ if $t \leqslant s$ is least such that $A_s \upharpoonright_n = A_t \upharpoonright_n$.

For each $e$ and $n$ let $h^e(n) = s$ if $o_s^{e,1}(n) = 0$ but $o_{s-1}^{e,1}(n) = 0$. If $h^e(n) = s$ then enumerate $n$ into a c.e. set $U^e$ at stage $s$. By the promptly simple slow-down lemma there is a non-decreasing computable function $q$ such that for all $e$, if $U^e$ is infinite then there is some $n$ which enters $U^e$ at some stage $s$ such that $A_s \upharpoonright_n \neq A_{q(s)} \upharpoonright_n$, so $g$ promptly $q$-escapes $\langle o_s^{e,1}\rangle$ on $n$. We only care about the case $U^e = \omega$.

In the other direction suppose that $\deg_{\mathrm{T}}(A) \in \mathsf{PN}(1)$, witnessed by some $g$ (which recall comes with an approximation $\langle g_s \rangle$). Let $\Gamma$ be a functional such that $\Gamma(A) = g$. Let $\langle A_s \rangle$ be some enumeration of $A$ such that for all $s$, $\operatorname{dom}\Gamma_s(A_s) \geqslant s$. Define a subsequence $0 = s(0) < s(1) < \dots$ such that for all $k$, $\Gamma_{s(k)}(A_{s(k)}) \restriction_k = g_{s(k)} \restriction_k$.

For each $x < \omega$ search for an index $k = k^e(x) > x, e$ such that some number $n$ enters $W_e$ at stage $k$ and the use $\gamma_{s(k)}(x)$ is smaller than $n$. We then define $h^e(x) = s(k)$. The domain of $h^e$ is an initial segment of $\omega$. We translate this to a counting down the ordinal 1: $m^e_t(x) = 1$ iff $t < h^e(x)$ (or $h^e(x)\uparrow$). Note that the counting $\langle m^e_t \rangle$ is total even if $h^e$ is partial. Further, $m^e_t(x) = 0$ implies $e, x < t$. So by Corollary 1.9 find a computable function $q$ such that for all $e$, if $h^e$ is total then there is some $x$ such that $g_{q(h^e(x))}(x) \neq g_{h^e(x)}(x)$.

Fix $e$. If $W_e$ is infinite then $h^e$ is total. Suppose that $g$ escapes $h^e$ on $x$ (as described above). If $h^e(x) = s(k)$ then find the $k' > k$ such that $q(s(k)) \in (s(k'-1), s(k')]$. Define $p(k) = k'$. Let $n$ be a number which enters $W_e$ at stage $k$ such that $\gamma_{s(k)}(x) < n$. The fact that $g(x)$ changes between stages $s(k)$ and $p(s(k))$ means that $A_{s(k')} \restriction_{\gamma_{s(k)}(x)} \neq A_{s(k)} \restriction_{\gamma_{s(k)}(x)}$. We conclude that the enumeration $\langle A_{s(k)} \rangle$ and the function $p$ witness that $A$ permits promptly. □

**1.5. A prompt hierarchy theorem.** Let $\mathsf{N}(\alpha)$ denote the class of c.e. degrees which are not totally $\alpha$-c.a. The class $\mathsf{N}(1)$ consists of the nonzero degrees.

$$\begin{tikzcd}
\vdots \arrow[d] & \vdots \arrow[d] \\
\mathsf{PN}(\omega^3) \arrow[r] \arrow[d] & \mathsf{N}(\omega^3) \arrow[d] \\
\mathsf{PN}(\omega^2) \arrow[r] \arrow[d] & \mathsf{N}(\omega^2) \arrow[d] \\
\mathsf{PN}(\omega^1) \arrow[r] \arrow[d] & \mathsf{N}(\omega^1) \arrow[d] \\
\mathsf{PN}(\omega^0) \arrow[r] & \mathsf{N}(\omega^0)
\end{tikzcd}$$

FIGURE 1. Prompt and regular classes. Arrows indicate containment.

The following proposition implies that no further implications hold between these classes.

Proposition 1.14. *Suppose that $\alpha \leqslant \beta \leqslant \epsilon_0$ are powers of $\omega$. Then there is a c.e. degree* $\mathbf{d}$ *such that:*

- $\mathbf{d} \in \mathsf{PN}(\gamma)$ *if and only if* $\gamma < \alpha$*; and*
- $\mathbf{d} \in \mathsf{N}(\gamma)$ *if and only if* $\gamma < \beta$.

For example, by choosing $\alpha = \beta$ we obtain:

Corollary 1.15. *Let $\alpha \leqslant \epsilon_0$. There is a degree which is promptly not totally $\omega^\alpha$-c.a. but is totally $\omega^{\alpha+1}$-c.a.*

On the other hand by choosing $\beta > \alpha$ we see:

Corollary 1.16. *Let $\alpha \leqslant \epsilon_0$. There is a degree which is not totally $\omega^\alpha$-c.a., but not promptly so (i.e. not in $\mathsf{PN}(\omega^\alpha)$), but is promptly not $\gamma$-c.a. for all $\gamma < \omega^\alpha$. (In particular if $\alpha > 0$ then the degree is promptly simple.)*

In this subsection we prove Proposition 1.14.

We define an approximation $\langle g_s \rangle$ witnessing that $g = \lim_s g_s$ is self-modulating, and intend to let $\mathbf{d} = \deg_{\mathrm{T}}(g)$.

For the positive side, for each $\gamma < \beta$ and $e < \omega$ we need to find some $p < \omega$ such that:

- $g(p) \neq f^{e,\gamma}(p)$;
- and if $\gamma < \alpha$ then in fact $g$ promptly id-escapes $\langle o_s^{e,\gamma} \rangle$ on the input $p$.

Call this requirement $P^{e,\gamma}$.

For the negative side, we need to meet the usual requirements $N^e$: if $\Phi_e(g)$ is total then it is $\beta$-c.a. But now we also have new requirements ensuring that $\mathbf{d}$ is not in $\mathsf{PN}(\alpha)$. Let $\langle \Gamma^j, \psi^j, h^j \rangle$ be an effective list of all triples of functionals, partial computable functions and partial computable approximations. We will build a family $m_s^j$ of (total) countings down $\alpha$; we will need to enjure, for each $j$, that if $\langle h^j(n,s) \rangle$ is a (total) approximation of a self-modulating function $\Gamma^j(g)$, and if $\psi^j$ is total, then $\Gamma^j(g)$ (equipped with the approximation $h^j$) does not promptly $\psi^j$-escape $m_s^j$ on any input (we then appeal to Corollary 1.9). The plan to meet this requirement is the following. One $n$ at a time we:

(1) Wait for a stage at which we see $\Gamma^j(g,n)\downarrow$, say with value $q$; until the end of the module for $n$ we restrain $g$ from changing below the use.
(2) Wait for a stage $s$ at which we see that $h^j(n,r)\downarrow = q$ for some $r \leqslant s$;
(3) Define $m_s^j(n) \neq m_{s-1}^j(n)$;
(4) Wait until we see that $\psi^j(s)\downarrow$ and $h^j(n,\psi^j(s))\downarrow = q$. When this is observed we end the module for $n$, lift the restraint, and move to $n+1$.

The main conflict is between the actions that must be done promptly and those that must wait until they become accessible again. We argued above that to meet $N^e$ we must use a tree of strategies. However to meet $P^{e,\gamma}$ for $\gamma < \alpha$ we need to change $g(p)$ immediately when we see that $o_s^{e,\gamma}(p)$ changes. The main observation here is that while action with existing followers must be immediate, the *appointment* of followers need not be: it can respect the priority tree. We will argue that this is sufficient to resolve the conflict between $P^{e,\gamma}$ and $N^e$.

Another conflict is between $M^j$ and $P^{e,\gamma}$ for $\gamma \geqslant \alpha$, in particular when $M^j$ is stronger. When $\gamma < \alpha$ we can allow action for $P^{e,\gamma}$ injure the action for $M^j$. We restart the module above (for the same $n$). If we started with a large enough ordinal $m^e(n)$ then we have room to keep decreasing it. We just need to distribute priorities so that for all $n$, only finitely many $P^{e,\gamma}$ can disturb the module for $n$. If $\gamma \geqslant \alpha$ then we cannot allow $P^{e,\gamma}$ to injure $M^j$. However, if $\gamma \geqslant \alpha$ then we do not need to act promptly for $P^{e,\gamma}$. And between ending the module for $n$ and starting the module for $n+1$, $M^j$ can drop all restraint. On a tree, this is enough to ensure that $P^{e,\gamma}$ eventually succeeds.

*Construction.* On the tree of strategies we apportion to each requirement all nodes of some level of the tree. The outcomes for nodes working for $N^e$ and $M^j$ are $\infty < \mathtt{fin}$; nodes working for $P^{e,\gamma}$ have a single outcome.

We start with $g_0$ being the constant function 0. At a stage $s > 0$ we define $g_s$. This is done by determining a number $p^*_s$ and letting $g_s(p) = s$ for $p \geqslant p^*_s$, and $g_{s-1}(p) = g_s(p)$ for $p < p^*_s$. If the stage is ended without determining $p^*_s$ then we let $g_s = g_{s-1}$.

Nodes $\sigma$ working for some $P^{e,\gamma}$ will appoint followers. If a node $\sigma$ is initialised then its follower is cancelled.

Nodes $\rho$ working for some $M^j$ will define a counting $\langle m^\rho_s \rangle$ down $\alpha$. We start with $m^\rho_0$ being the constant function $\alpha$. At stage $s > 0$ we define $m^\rho_s$ for all $\rho$. If $\rho$ is initialised then we throw the counting $\langle m^\rho_s \rangle$ out and start a new one (we complete the old counting trivially, say with zeros everywhere, so that at the end we do get a uniformly computable sequence of total countings.) If $\rho$ is initialised at stage $s$ then we (re)define $m^\rho_t$ to be the constant function $\alpha$ for all $t \leqslant s$. If $\rho$ is not initialised at stage $s$ but is not accessible at stage $s$ then we define $m^\rho_s = m^\rho_{s-1}$.

At each stage $s$, each node $\rho$ working for $M^j$ will be trying to meet the subrequirement $M^\rho_n$ for some $n$; we denote this $n$ by $n_s(\rho)$. We set $n_0(\rho) = 0$, and reset $n_s(\rho) = 0$ if $\rho$ is initialised at stage $s$. Unless otherwise stated, we let $n_s(\rho) = n_{s-1}(\rho)$.

At stage $s$ we first tend to promptness requirements. We ask if there is some node $\sigma$, working for some $P^{e,\gamma}$ for $\gamma < \alpha$, which has a follower $p$ defined, and $o^{e,\gamma}_s(p) \neq o^{e,\gamma}_{s-1}(p)$. If so, we let $\sigma$ be the strongest such node; we determine $p^*_s = p$, and initialise all nodes weaker than $\sigma$. No node is accessible, and we move to the next stage.

If there is no such node $\sigma$ then we build the path of accessible nodes.

Suppose that a node $\tau$, working for some $N^e$, is accessible at stage $s$. We let $t$ be the greatest stage before $s$ at which $\tau\hat{\ }\infty$ was accessible, $t = 0$ if there was no such stage. If $\operatorname{dom} \Phi_{e,s}(g_{s-1}) \geqslant t$ then we let $\tau\hat{\ }\infty$ be next accessible. Otherwise we let $\tau\hat{\ }\mathtt{fin}$ be next acccessible.

Suppose that a node $\sigma$, working for some $P^{e,\gamma}$, is accessible at stage $s$. If $\sigma$ has no follower then it appoints a new, large follower, initialises all weaker nodes, and ends the stage. If $\sigma$ already has a follower $p$, $\gamma \geqslant \alpha$ and $f^{e,\gamma}_s(p) = g_{s-1}(p)$ then we determine that $p^*_s = p$, initialise all weaker nodes, and end the stage. Otherwise, we let the unique successor of $\sigma$ on the tree be next accessible.

Suppose that a node $\rho$, working for some $M^j$, is accessible at stage $s$. Let $n = n_{s-1}(\rho)$. The subrequirement $M^\rho_n$ is currently seen to be *satisfied* if there is some stage $r < s$ such that $m^\rho_r(n) \neq m^\rho_{r-1}(n)$, $\psi^j(r)\downarrow$ by stage $s$, and $h^j(n, \psi^j(r))\downarrow = h^j(n, r)$. If this subrequirement is currently seen to be satisfied then we let $n_s(\rho) = n + 1$, and let $\rho\hat{\ }\infty$ be next accessible; we let $m^\rho_s(k) = 0$ for $k \leqslant n$ and $m^\rho_s(k) = \alpha$ for $k > n$.

Suppose that this is not the case. If $\Gamma^j_s(g_{s-1}, n)\uparrow$, let $m^\rho_s = m^\rho_{s-1}$ and let $\rho\hat{}\mathtt{fin}$ be next accessible. Suppose that $\Gamma^j_s(g_{s-1}, n)\downarrow= q$, and let $\gamma^j_s(n)$ be the use. If there is some node $\sigma \geqslant \rho\hat{}\mathtt{fin}$, working for some $P^{d,\gamma}$, which has a follower $p < \gamma^j_s(n)$, then we initialise all nodes to the right of $\rho\hat{}\infty$, let $m^\rho_s = m^\rho_{s-1}$, and end the stage.

Otherwise, if there is no $r < s$ such that currently we see that $h^j(n, r) = q$ then again we let $m^\rho_s = m^\rho_{s-1}$ and let $\rho\hat{}\mathtt{fin}$ be next accessible. If there is such $r$, let $t$ be the last stage at which $\rho\hat{}\infty$ was accessible, $t = 0$ if there was no such stage. Let $\sigma_1, \sigma_2, \dots, \sigma_k$ be the list, with descending priority, of the nodes extending $\rho\hat{}\infty$, working for some $P^{e,\gamma}$ for some $\gamma < \alpha$, which currently have a follower $p$; let $p_i$ be the follower for node $\sigma_i$ and say that $\sigma_i$ works for $P^{e_i,\gamma_i}$. We let

$$m^\rho_s(n) = \sum_{i\leqslant k} o^{e_i,\gamma_i}_s(p_i);$$

we let $m^\rho_s(n') = 0$ for all $n' < n$ and $m^\rho_s(n') = \alpha$ for all $n' > n$. We let $\rho\hat{}\mathtt{fin}$ be next accessible.

*Verification.* Let $\rho$ be a node, working for some $M^j$. Our first task is to prove:

LEMMA 1.17. *$\langle m^\rho_s\rangle$ is a counting down $\alpha$.*

Let $s < \omega$, and let $r^*$ be the last stage prior to stage $s$ at which $\rho$ was accessible. We need to show that the conditions for $m^\rho$ for being a counting have not been violated by stage $s$. We observe:

- If $r^* \leqslant t \leqslant s$ then $n_t(\rho) \leqslant n_s(\rho)$;
- For all $n' < n_s(\rho)$, $m^\rho_s(n') = 0$;
- For all $n' > n_s(\rho)$, $m^\rho_s(n') = \alpha$.

So the only question is what happens on $n = n_s(\rho)$. Let $u^* \geqslant r^*$ be the least stage such that $n_{u^*}(\rho) = n$. For $t \in (u^*, s]$ let $\sigma^t_1, \sigma^t_2, \dots, \sigma^t_{k(t)}$ be the list, with descending priority, of the nodes extending $\rho\hat{}\infty$, working for some $P^{e,\gamma}$ for some $\gamma < \alpha$, which at stage $t$ have a follower. Since $\rho\hat{}\infty$ is not accessible on the interval $(u^*, s]$, we in fact know that the node $\sigma^t_i$ does not depend on $t$, so we write $\sigma_1, \sigma_2, \dots, \sigma_{k(t)}$; and the follower $p_i$ for $\sigma_i$ does not change. Say $\sigma_i$ works for $P^{e_i,\gamma_i}$; for brevity let, for $t > u^*$,

$$\eta^\rho_t = \sum i \leqslant k o^{e_i,\gamma_i}_t(p_i).$$

So $\eta^\rho_t$ is non-increasing, and if some node $\sigma_i$ acts at a stage $w \in (u^*, s]$ then as $o^{e_i,\gamma_i}_w(p_i) < o^{e_i,\gamma_i}_{w-1}(p_i)$ we have $\beta^\rho_w < \beta^\rho_{w-1}$. Further, since $\alpha$ is closed under addition and each $\gamma_i$ is smaller than $\alpha$, we have $\beta^\rho_t < \alpha$ for all $t \in (u^*, s]$. Now let $t < s$. Either $m^\rho_t(n) = \alpha$, in which case certainly $m^\rho_s(n) \leqslant m^t_\rho(n)$; or there are stages $t' \leqslant t$ and $s' \geqslant t'$ such that $m^\rho_t(n) = \beta^\rho_{t'}$ and $m^\rho_s(n) = \beta^\rho_{s'}$; so we get $m^\rho_s(n) \leqslant m^\rho_t(n)$ as required. This proves Lemma 1.17.

Keeping with the same notation, say that $\rho$ *acts* at a stage $t > u^*$ if it is accessible at stage $t$ and ends the stage (initialising all extensions of $\rho\hat{}\mathtt{fin}$).

LEMMA 1.18. *Suppose that $\rho$ acts at two stages $s > t$, that $n_s(\rho) = n_t(\rho)$, and that $\rho$ is not initialised at any stage in the interval $[t, s]$. Then $\beta^\rho_{s-1} < \beta^\rho_t$.*

PROOF. Let $n = n_t(\rho)$. The action of $\rho$ at stage $t$ ensures that the computation $\Gamma^j_t(g_{t-1}, n)$ is injured between stages $t$ and stage $s$. This action, and the fact that $\rho$ itself is not initialised between stages $t$ and $s$, means that some node $\sigma$ extending $\rho\hat{}\infty$ acts at some stage $w \in (t, s)$ and changes $g$ below the use of the computation.

Since $\rho\hat{}\infty$ is not accessible at that interval, $\sigma$ must work for some $P^{e,\gamma}$ where $\gamma < \alpha$. We observed that this means that $\beta^\rho_w < \beta^\rho_{w-1}$. □

LEMMA 1.19. *The true path is infinite, and the construction is fair to every node on the true path.*

PROOF. As usual, if $p$ is a follower for some node $\sigma$ then $\sigma$ acts for $p$ only finitely often. This shows that there are infinitely many stages at which we build the path of accessible nodes. Hence the root node lies on the true path, and of course is never initialised. Also this shows that a node that lies to the left of the true path can act at most finitely often.

Further, the usual arguments show that if a node working for either $P^{e,\gamma}$ or $N^e$ is on the true path and is initialised only finitely many times, then some immediate successor of the node on the tree lies on the true path, and is only initilaised finitely many times.

So we consider a node $\rho$ on the true path, working for some $M^j$. The node $\rho$ never initialises nodes extending $\rho\hat{}\infty$, so if $\rho\hat{}\infty$ is accessible infinitely often then we are done. Suppose that this is not the case. Then we can let $t^*$ be the stage at which the last value $n^*$ for $n_s(\rho)$ is set (either the last stage at which $\rho\hat{}\infty$ is accessible, or the last stage at which $\rho$ is initialised). Now Lemma 1.18 implies that $\rho$ acts only finitely many times after stage $t^*$. □

It is not difficult to see that every positive requirement is met. Further, following the proof of Theorem III.2.1 we can see that each requirement $N^e$ is met. As we mentioned above, it is not actually important that a computation $\Phi_e(g,x)$, already certified by a node $\tau$ on the true path, is injured only during $\tau\hat{}\infty$-stages; it is only important that the node injuring the computation extends $\tau\hat{}\infty$. We are left therefore with verifying that each $M^j$ is met. Fix $j$, let $\rho$ be a node on the true path working for $M^j$, and suppose that $\psi^j$ is a total speed-up function, $h^j$ is a (total) approximation witnessing that $\Gamma^j(g)$ (which is total) is self-modulating. We show that every subrequirement $M^\rho_n$ is satisfied: for every $n$ there is some stage $r$ such that $m^\rho_r(n) \neq m^\rho_{r-1}(n)$ and $h^j(n,\psi^j(r)) = h^j(n,r)$. Of course if the subrequirement is every seen to be satisfied then it is indeed satisfied. So by induction we show that $\langle n_s(\rho)\rangle$ is unbounded, equivalently that $\rho\hat{}\infty$ lies on the true path.

Suppose that this is not the case; let $n = \lim_s n_s(\rho)$; let $t^*$ be the least stage (not before the last stage at which $\rho$ was initialised) such that $n_{t*}(\rho) = n$. The fact that $\Gamma^j(g,n)c\downarrow$ and that $\lim_s h^j(n,s) = \Gamma^j(g)$ implies that $\lim_s m^\rho_s(n) = \lim_s \beta^\rho_s(n)$; let $\delta$ be that common value. Let $s$ be the least stage at which $m^\rho_s(n) = \delta$; since $\delta < \alpha$, $m^\rho_s(n) \neq m^\rho_{s-1}(n)$. Also, by our instructions, $m^\rho_s(n) = \beta^\rho_s(n)$ so $\beta^\rho_s(n) = \delta$ and in fact $\beta^\rho_t(n) = \delta$ for all $t > s$.

Suppose that the computation $\Gamma^j_s(g_{s-1},n) = q$ is correct. There is some $r < s$ such that $h^j(n,r) = q$; since $h^j$ correctly approximates $\Gamma^j(g)$, and in a non-decreasing way, it must be that $h^j(n,w) = q$ for all $w \geqslant r$. But then since $\psi^j$ is total, we will eventually see that $M^\rho_n$ is satisfied, contrary to our hypothesis. Hence the computation $\Gamma^j_s(g_{s-1},n)$ is injured at some stage $w > s$. The fact that $\rho$ does not act at stage $s$ implies, as in the arguments above, that some node $\sigma$ extending $\rho\hat{}\infty$ does this injury, and that it must work for $P^{e,\gamma}$ for some $\gamma < \alpha$; this implies that $\beta^\rho_w < \beta^\rho_s$. This is the desired contradiction, showing that $M^j$ is met, and concluding the proof of Proposition 1.14.

**1.6. Uniform prompt classes.** The uniform layers in our hierarchy also have prompt versions. Let $\alpha \leqslant \epsilon_0$ be an infinite power of $\omega$. Recall the definition of an $\alpha$-order function $h$ and of $h$-computable approximations (Definition III.3.1). Recall also that we have a uniform listing $\left\langle f_s^{e,h}, o_s^{e,h}\right\rangle$ of tidy $(h+1)$-computable approximations of all $h$-c.a. functions. To avoid technical annoyances we define:

DEFINITION 1.20. A self-modulating function $g$ is *promptly not $h$-c.a.* if there is a speed up function $p$ such that $g$ promptly $p$-escapes each counting $\left\langle o_s^{e,h}\right\rangle$ on infinitely many inputs.

An elaboration on the argument giving Lemma III.3.2 yields the following.

LEMMA 1.21. *The following are equivalent for a c.e. degree* $\mathbf{d}$*:*

(1) *For some $\alpha$-order function $h$, some $g \leqslant_{\mathrm{T}} \mathbf{d}$ is promptly not $h$-c.a.;*
(2) *For every $\alpha$-order function $h$, some $g \leqslant_{\mathrm{T}} \mathbf{d}$ is promptly not $h$-c.a.*

If these conditions hold then we say that $\mathbf{d}$ is *promtply not uniformly $\alpha$-c.a.* When $\alpha = \omega$ we say that $\mathbf{d}$ is *promptly array noncomputable.*

PROOF. Let $h$ and $\bar{h}$ be $\alpha$-order functions; let $f$ be a function which is promptly not $h$-c.a. As in the proof of Lemma III.3.2 partition $\omega$ into an increasing sequence of finite intervals $I^* < I_0 < I_1 < I_2 < \dots$ such that for all $n$, for all $x \in I_n$ we have $h(x) \geqslant \bar{h}(n)$.

Define a self-modulating function $g$ by setting $g_s(n) = s$ if $f_s(x) = s$ for some $x \in I_m$ for some $m \leqslant n$.

For each $e$, define a counting $\langle m_s^e(x)\rangle$ down $h$ by letting

$$m_s^e(x) = \begin{cases} 0, & \text{if } x \in I^*; \\ h(x), & \text{if } x \in I_n \text{ and } o_s^{e,\bar{h}}(n) = \bar{h}(n); \text{ and} \\ o_s^{e,\bar{h}}(n) & \text{if } x \in I_n \text{ and } o_s^{e,\bar{h}}(n) < \bar{h}(n). \end{cases}$$

The slow-down lemma holds for $h$ and so an analogue of Corollary 1.9 ensures that there is a speed-up function $p$ such that $f$ promptly $p$-escapes each $\langle m_s^e\rangle$ on infinitely many inputs.

Fix $e$ and suppose that $f$ promptly $p$-escapes $\langle m_s^e\rangle$ on an input $x \notin I^*$; say $x \in I_n$. Then $g$ promptly $p$-escapes $\left\langle o_s^{e,\bar{h}}\right\rangle$ on the input $n$. ☐

We can also define the prompt version of the class of not totally $< \alpha$-c.a. functions; the definition carries no surprises. The techniques used above allow us to prove hierarchy theorems for these classes; we do not elaborate here.

## 2. Incomparable pairs of separating classes

To demonstrate the dynamic power encapsulated by prompt classes we discuss separating classes. Recall that a *separating class* $\mathcal{P} = \mathcal{P}(A_0, A_1)$ is the $\Pi_1^0$ class of all sets separating two disjoint c.e. sets $A_0$ and $A_1$; for non-triviality we require that $A_0 \cup A_1$ is co-infinite, so that the separating class is uncountable rather than finite. Recall that we say that a Turing degree $\mathbf{d}$ *computes* a $\Pi_1^0$ class $\mathcal{P}$ if it computes the finite binary strings extendible to elements of $\mathcal{P}$. A Turing degree $\mathbf{d}$ computes a separating class $\mathcal{P}(A_0, A_1)$ if and only if it computes both $A_0$ and $A_1$.

Downey, Jockusch and Stob [23] proved that a c.e. degree is array noncomputable if and only if it computes two separating classes $\mathcal{P}$ and $\mathcal{Q}$ which are incomparable in the sense that any element of $\mathcal{P}$ is Turing incomparable with any element of $\mathcal{Q}$. In one direction they showed that any separating class computed by an array computable degree has an element of degree $\mathbf{0}'$. Here we prove:

THEOREM 2.1. *Every c.e. degree which is promptly array noncomputable computes two separating classes $\mathcal{P}$ and $\mathcal{Q}$ such that any element of $\mathcal{P}$ forms a minimal pair with any element of $\mathcal{Q}$.*

**2.1. The Jockusch-Soare construction.** To prove the theorem, we first recall how to construct such classes $\mathcal{P}$ and $\mathcal{Q}$, as was first done by Jockusch and Soare in [38]. We are not aware of a modern presentation of this construction, so we discuss it in some detail.

We wish to enumerate four c.e. sets $A_0$, $A_1$, $B_0$ and $B_1$ with the intention of letting $\mathcal{P} = \mathcal{P}(A_0, A_1)$ and $\mathcal{Q} = \mathcal{P}(B_0, B_1)$. The minimality requirements we need to meet are:

$R_e$**:** If $X \in \mathcal{P}$, $Y \in \mathcal{Q}$ and $\Phi_e(X) = \Psi_e(Y)$ is total, then it is computable.

(Here as usual $\langle \Phi_e, \Psi_e \rangle$ is a list of all pairs of functionals). To meet one requirement on its own we can follow the forcing constuction of a minimal pair of degrees: we look for splits and take them if possible. Namely, if we find two strings $\sigma$ and $\tau$ such that $\Phi_e(\sigma) \perp \Psi_e(\tau)$ then we ensure that $\mathcal{P} \subseteq [\sigma]$ and that $\mathcal{Q} \subseteq [\tau]$. This is done by enumerating into $A_0$ every $x$ such that $\sigma(x) = 0$ and into $A_1$ every $x$ such that $\sigma(x) = 1$; and similarly with $\tau$ and the sets $B_0$ and $B_1$.

In the forcing construction, we argue that if we never find splits then the only value we ever see must be correct. When more than one requirement is acting this is no longer true. While $R_e$ is searching for splits, weaker requirements shrink the classes $\mathcal{P}$ and $\mathcal{Q}$ by enumerating numbers into the various sets. It is possible that $e$-splits appear too late for us to be able to take them. In particular, it is possible that we first see $\Phi_e(X)$ computations (for clopen sets of $X \in \mathcal{P}_s$) agreeing with $\Psi_e(Y)$ computations (for $Y \in \mathcal{Q}_s$), say giving value 0 on some input $n$; then the action of weaker requirements extracts all these $X$'s and $Y$'s from $\mathcal{P}$ and $\mathcal{Q}$; and only afterwards we see new computations on the remaining elements of the classes, giving value 1 on the input $n$.

The solution is to add Lachlan's idea for constructing a minimal pair of c.e. sets. Rather than only searching for splits, we will actively declare values that we believe and act to preserve them. Requirements weaker than $R_e$ will only act during $e$-expansionary stages; and at each such stage either shrink $\mathcal{P}$, or shrink $\mathcal{Q}$, but not both. In other words, at all times we keep one side of the computation alive, and wait for the other side to reveal a new agreement with the constant value (likely with a different computation). Note that we still need to keep looking for splits and take them when we can: preserving a computation on the $\mathcal{P}$-side means that we keep ensuring that $\Phi_e(X, n)\downarrow = i$ for *some* $X \in \mathcal{P}_s$; we cannot expect that we will see the same value for all $X \in \mathcal{P}_s$.

We also need to ensure that the classes $\mathcal{P}$ and $\mathcal{Q}$ are infinite: that the complements of $A_0 \cup A_1$ and $B_0 \cup B_1$ are infinite. A simple way to arrange for this is to assign a number $k$ to each strategy $\tau$ working for $R_e$ and prevent that strategy $\tau$ from enumerating the $k$-many smallest elements of those complemenets into the sets. This will force us to consider $(2^k)^2$ many different subrequirements of $R_e$

at the strategy $\tau$. Suppose that the strategy sees a split $\alpha$ and $\beta$. Let $C$ and $D$ be the sets of numbers that $R_e$ is prevented from enumerating into $A_0 \cup A_1$ and $B_0 \cup B_1$. Except for on numbers in these sets, we can make sure that all separators of $A_0$ and $A_1$ agree with $\alpha$, and the same on the other side. What this means is that we have ensured that the requirement is met for all separators $X$ and $Y$ that agree with $\alpha$ and $\beta$ on $C$ and $D$. So for any possible configuration on $C$ and $D$, a substrategy of $\tau$ is restricted to work on the clopen sets determined by these configurations.

Recall that the Lachlan strategy posits that between expansionary stages, weaker requirements are restrained from destroying the computation remaining on one side. In this construction, this means that they are prevented from completely extracting from $\mathcal{P}$ (or perhaps $\mathcal{Q}$) oracles that give the constant value. This is done by restraining these weaker nodes from enumerating numbers below the length of $\alpha$ (or perhaps $\beta$) above into the sets. In other words, by setting the restraint $k$ of these nodes to be larger than the use. Of course this means that these nodes have to consider more sub-strategies. Which in turn means that substrategies of the same strategy cannot restrain each other; it seems that we cannot consider them as independent nodes on the tree os strategies.

On the other hand, $\Pi_2/\Sigma_2$ outcomes for different sub-strategies are independent; we can see more convergences and agreement on some clopen subsets but not on others. As a result, when calculating the restraint that a node imposes on weaker strategies, it is important to see all the subrequirements as independent nodes on the tree. It is just that these nodes will only contribute toward meeting the subrequirement *negatively*, by extending the length of agreement and imposting restraint; the positive parts of meeting the subrequirement, that of seeking splits and following them, must all be shared at one "primary" node.

*The tree of strategies.* Nodes on the tree of strategies will be finite sequences of numbers and the symbol $\infty$. With every node $\sigma$ we will associate a restraint $r(\sigma)$ (imposed on $\sigma$). There will be two kinds of nodes: *primary nodes* $\tau$ which work for some requirement $R_e$; and auxiliary nodes whose job is to help calculate the restraint imposed by subrequirements. For brevity for a primary node $\tau$ we let $m(\tau) = 2^{2r(\tau)}$. The tree and the restraint are defined recursively.

We start with the empty string $\langle\rangle$ which is a primary node, working for $R_0$. We let $r(\langle\rangle) = 0$. Suppose that $\tau$ is a primary node, working for a requirement $R_e$. The outcomes of $\tau$ are all the numbers $k > r(\tau)$ (ordered naturally). We let $r(\tau\hat{\ }k) = k$. For each $i < m(\tau)$, all extensions of $\tau$ of length $|\tau| + (i + 1)$ are auxiliary nodes associated with a subrequirement $R_{\tau,i}$ (which will be the restriction of $R_e$ to a pair of clopen sets). If $\sigma$ is such a node then the outcomes of $\sigma$ are $\infty$ and all natural numbers $k \geqslant r(\sigma)$, ordered $\infty < r(\sigma) < r(\sigma) + 1 < \cdots$. We let $r(\sigma\hat{\ }\infty) = r(\sigma)$ and $r(\sigma\hat{\ }k) = k$.

All extensions of $\tau$ of length $|\tau| + m(\tau) + 1$ are primary nodes, each working for $R_{e+1}$.

*Construction.* For definiteness, for a pair of disjoint sets $E$ and $F$ we let

$$\mathcal{P}(E, F) = \{X \in 2^\omega : (n \in E \to X(n) = 0) \ \ \& \ \ (n \in F \to X(n) = 1)\}.$$

We enumerate four sets $A_0$, $A_1$, $B_0$ and $B_1$, and make sure to keep $A_0$ and $A_1$ disjoint, and $B_0$ and $B_1$ disjoint. At stage $s$ we let $\mathcal{P}_s = \mathcal{P}(A_{0,s}, A_{1,s})$ and $\mathcal{Q}_s = \mathcal{P}(B_{0,s}, B_{1,s})$.

At stage $s \geqslant 1$ we describe the path of accessible nodes. The root is always accessible. Let $\tau$ be a primary node which is accessible at stage $s$. If $|\tau| \geqslant s$ we end the stage. Suppose that $|\tau| < s$.

Let $\bar{A}_s$ be the complement of $A_{0,s} \cup A_{1,s}$; similarly define $\bar{B}_s$. We let $\bar{A}_s(\tau)$ be the collection of the $r(\tau)$-many smallest elements of $\bar{A}_s$; we similarly define $\bar{B}_s(\tau)$. Let $\mathsf{C}_s(\tau)$ be the collection of all clopen subsets of $2^\omega$ determined by a choice of bits on $\bar{A}_s(\tau)$; let $\mathsf{D}_s(\tau)$ be the collection of all clopen subsets of $2^\omega$ determined by a choice of bits on $\bar{B}_s(\tau)$; and let $\langle \mathcal{C}_i, \mathcal{D}_i \rangle_{i<m(\tau)}$ be an enumeration of $\mathsf{C}_s(\tau) \times \mathsf{D}_s(\tau)$. The subrequirement $R_{\tau,i}$ is the restriction of $R_e$ to the clopen sets $\mathcal{C}_i$ and $\mathcal{D}_i$.

Suppose that $\tau$ works for $R_e$. Recall that an *e-split* is a pair $(\alpha, \beta)$ of binary strings such that $\Phi_e(\alpha) \perp \Psi_e(\beta)$. Let $i < m(\tau)$.

- We say that the subrequirement $R_{\tau,i}$ is *already met* at stage $s$ if there is an $e$-split $(\alpha, \beta)$ (observed by this stage) such that $\mathcal{P}_s \cap \mathcal{C}_i \subseteq [\alpha]$ and $\mathcal{Q}_s \cap \mathcal{D}_i \subseteq [\beta]$.
- We say that $R_{\tau,i}$ *admits a split* at stage $s$ if there is an $e$-split $(\alpha, \beta)$, observed by this stage, such that $[\alpha] \subseteq \mathcal{C}_i$, $[\beta] \subseteq \mathcal{D}_i$, $[\alpha] \cap \mathcal{P}_s \neq \varnothing$ and $[\beta] \cap \mathcal{Q}_s \neq \varnothing$.
- Suppose that $R_{\tau,i}$ does not admit a split at stage $s$. We then define $\zeta_s(\tau, i)$, the $(\tau, i)$-*agreement* at stage $s$, to be the longest binary string $\zeta$ such that $\zeta \preccurlyeq \Phi_e(X)$ and $\zeta \preccurlyeq \Psi_e(Y)$ for some $X \in \mathcal{P}_s \cap \mathcal{C}_i$ and some $Y \in \mathcal{Q}_s \cap \mathcal{D}_i$ (of course, as observed at stage $s$).

At stage $s$, if there is some subrequirement $R_{\tau,i}$ which admits a split but is not already met, then we choose the least such $i$, and we let $(\alpha, \beta)$ be the least split admitted by the subrequirement. We then act as follows:

- If $\mathcal{P}_s \cap \mathcal{C}_i \nsubseteq [\alpha]$ then we enumerate numbers into $A_{0,s+1}$ and $A_{1,s+1}$ so that $\mathcal{P}_{s+1} \cap \mathcal{C}_i \subseteq [\alpha]$. Namely for all $x < |\alpha|$, $x \in \bar{A}_s \backslash \bar{A}_s(\tau)$, we enumerate $x$ into $A_{0,s+1}$ if $\alpha(x) = 0$ and enumerate $x$ into $A_{1,s+1}$ if $\alpha(x) = 1$. We declare that $\tau$ *acted* at stage $s$ and end the stage.
- If $\mathcal{P}_s \cap \mathcal{C}_i \subseteq [\alpha]$ then we act similarly, to ensure that $\mathcal{Q}_{s+1} \cap \mathcal{D}_i \subseteq [\beta]$, declare that $\tau$ acted and end the stage.

If $\tau$ does not act at stage $s$ we extend the path of accessible nodes up to the next primary node. We first determine the immediate extension of $\tau$ by determining $\tau$'s outcome at stage $s$; the outcome is the greatest stage so far at which $\tau$ acted, if there is such a stage; if not, the outcome is $r(\tau) + 1$. Now let $i < m(\tau)$ and suppose that a node $\sigma$ of length $|\tau| + (i + 1)$ (and so associated with $R_{\tau,i}$) is accessible at stage $s$. If this is the first stage at which $\sigma$ is accessible, let the outcome of $\sigma$ at this stage be $\infty$. Otherwise, let $t$ be the greatest stage prior to stage $s$ at which $\sigma\hat{}\infty$ was accessible. If $|\zeta(\tau, i)| > t$ then we let $\sigma\hat{}\infty$ be next accessible. Otherwise we let $\sigma\hat{}t$ be next accessible.

*Verification.* We first work toward showing that the true path is infinite. To do this we will need to show that every (primary) node acts only finitely many times. The following lemma shows that nodes are successful in imposing restraint. Note that for every stage $s$, for every node $\tau$ which is accessible at stage $s$, we have $r(\tau) < s$ (this shows that the outcome of a primary node $\tau$ is indeed always a number greater than $r(\tau)$). Also note that if $\sigma \preccurlyeq \tau$ then $r(\sigma) \leqslant r(\tau)$.

Lemma 2.2. *Let $\tau$ be a node on the tree of strategies. Suppose that $\tau$ is accessible at some stage $t$; suppose that a node $\rho$, which lies to the right of $\tau$, is accessible at some stage $s > t$. Then $r(\rho) \geqslant t$.*

Proof. Let $\sigma$ be the longest common initial segment of $\tau$ and $\rho$; let $o$ and $p$ be the outcomes of $\sigma$ such that $\sigma\hat{\ }o \preccurlyeq \tau$ and $\sigma\hat{\ }p \preccurlyeq \rho$. So $o < p$. If $\sigma$ is a primary node then $p$ is a stage at which $\sigma$ acts, and $p > t$ (or at stage $t$ the outcome would be at least $p$), and $r(\rho) \geqslant p$. The other case is similar. □

Note that the lemma implies that if some node $\rho$ is accessible at some stage $t$, and that at stage $s > t$, some node that lies to the left of $\rho$ is accessible, then $\rho$ is never accessible after stage $s$. This implies that if $\tau$ lies on the true path then no node to the left of $\tau$ is ever accessible.

Lemma 2.3. *Let $\tau$ be a primary node which lies on the true path; let $t$ be a stage at which $\tau$ is accessible.*

(1) *No node $\sigma < \tau$ acts after stage $t$.*
(2) *For all $s > t$, $\bar{A}_s(\tau) = \bar{A}_t(\tau)$ and $\bar{B}_s(\tau) = \bar{B}_t(\tau)$.*
(3) *Suppose that a subrequirement $R_{\tau,i}$ is seen to be already met by stage $t+1$, witnessed by some $e$-split $(\alpha, \beta)$. Then for all $s \geqslant t+1$, $R_{\tau,i}$ is also seen to be met, by the same pair.*
(4) *The node $\tau$ acts at most finitely many times.*

Proof. (1): Say that $\sigma < \tau$ acts at stage $s > t$. Every outcome of $\sigma$ taken after stage $s$ will be at least $s$, which is greater than $r(\tau)$, so $\tau$ will not be accessible after stage $s$.

(2): Suppose that a node $\sigma$ acts at some stage $s \geqslant t$ and enumerates some numbers from $\bar{A}_t(\tau)$ into $A_0 \cup A_1$ (or from $\bar{B}_t(\tau)$ into $B_0 \cup B_1$). Then $r(\sigma) < r(\tau)$. As observed, $\sigma$ cannot lie to the left of $\tau$; and $\sigma$ cannot extend $\tau$. Lemma 2.2 (and the fact that $r(\tau) < t$) implies that $\sigma$ cannot lie to the right of $\tau$. And (1) implies that $\tau$ cannot extend $\sigma$.

(3): The point here is that for all $s$, $\mathcal{P}_s \cap \mathcal{C}_i \neq \varnothing$ and $\mathcal{Q}_s \cap \mathcal{D}_i \neq \varnothing$.

(4): by induction on $i$, we show that $\tau$ acts on behalf of $R_{\tau,i}$ only finitely many times. Fix some $i < m(\tau)$ and suppose that after stage $t_0$, $\tau$ does not act on behalf of $R_{\tau,j}$ for any $j < i$. Suppose that $\tau$ acts on behalf of $R_{\tau,i}$ at stage $t_1 \geqslant t_0$. If $R_{\tau,i}$ is met by this action then we are done by (1). If not, then the action at stage $t_1$ was on the $\mathcal{P}$-side (rather than the $\mathcal{Q}$-side). Let $(\alpha, \beta)$ be the split that prompted the action; let $s$ be the next stage after stage $t_1$ at which $\tau$ is accessible. We claim that $[\beta] \cap \mathcal{Q}_s \neq \varnothing$; this would imply that at stage $s$, $\tau$ will act again on behalf of $R_{\tau,i}$ and cause it to be met. The fact that $[\beta] \cap \mathcal{Q}_s \neq \varnothing$ follows from the success of imposing restraint: if a node $\sigma$ acts at some stage $u \in (t, s)$ then by (1) and Lemma 2.2, $r(\sigma) > t$, so $\sigma$ cannot enumerate any numbers below $|\beta|$ into $B_0 \cup B_1$. □

Lemma 2.3 implies that the true path is infinite.

Lemma 2.4. *The classes $\mathcal{P}$ and $\mathcal{Q}$ are uncountable.*

PROOF. Let $e < \omega$. Let $\tau$ be a node on the true path which works for $R_e$. Then $r(\tau) \geqslant e$. If $\tau$ is accessible at stage $t$ then no number from $\bar{A}_t(\tau)$ is ever enumerated into $A_0 \cup A_1$, and so the complement of $A_0 \cup A_1$ contains at least $e$ many elements. The same holds for $B$. □

LEMMA 2.5. *Every requirement $R_e$ is met.*

PROOF. Let $e < \omega$; let $\tau$ be the primary node on the true path which works for $R_e$.

Let $X \in \mathcal{P}$ and $Y \in \mathcal{Q}$, and suppose that $\Phi_e(X) = \Psi_e(Y)$. There is a unique $i < m(\tau)$ such that $X \in \mathcal{C}_i$ and $Y \in \mathcal{D}_i$. The subrequirement $R_{\tau,i}$ is never seen to be met, and in fact, by Lemma 2.3, at no stage $t$ at which $\tau$ is accessible does $R_{\tau,i}$ admit a split.

Let $\sigma$ be the auxiliary node on the true path which is associated with $R_{\tau,i}$. The reals $X$ and $Y$ show that $\sigma\hat{}\infty$ lies on the true path. We show that if $s > t$ are stages at which $\sigma\hat{}\infty$ is accessible then $\zeta_t(\tau, i) \prec \zeta_s(\tau, i)$; the fact that no splits are ever observed will imply that $\zeta_t(\tau, i) \prec \Phi_e(X)$ for all such $t$. Note that the node $\tau$ does not act after stage $t$ (or $\sigma$ would not be on the true path).

As discussed above, the argument is really the Lachlan minimal pair argument. At stage $t$, at most one node extending $\sigma$ acts. That node enumerates numbers into $A_0 \cup A_1$, or into $B_0 \cup B_1$, but not both. Without loss of generality, say it is the former. The arguments above show that any node $\sigma$ that acts between stages $t$ and $s$ has restraint $r(\sigma) \geqslant t$. This implies that if $[\beta] \subseteq \mathcal{Q}_t \cap \mathcal{D}_i$ has length $t$ and $\Psi_e(\beta) \succcurlyeq \zeta_t(\tau, i)$ then $[\beta] \cap \mathcal{Q}_s \neq \varnothing$ as well. □

**2.2. Adding prompt permissions.** To prove Theorem 2.1 we observe that the proof of Lemma 2.3 shows that in fact we can computably bound the number of times a primary node will need to act: at most twice for each $R_{\tau,i}$, once all action for $R_{\tau,j}$ for $j < i$ has ceased. The total is $\sum_{i<m(\tau)} 2^{i+1} \leqslant 2^{m(\tau)+1} = 2^{1+2^{2r(\tau)}}$. So we let $h(r) = 2^{1+2^{2r}}$. We need the permissions to be prompt: otherwise the Lachlan mechanism of keeping one side of the computation alive cannot work. Let $\mathbf{d}$ be a c.e. degree which is promptly array noncomputable; by Lemma 1.21 there is some function $g \leqslant_{\mathrm{T}} \mathbf{d}$ which is promptly not $h$-c.a.

The idea is to use $g$ to permit the action of a node $\tau$. Each time $\tau$ wants to act we will seek a change in $g(r(\tau))$. If we do not get it we will of course notice that immediately; we will then essentially want to increase $r(\tau)$ by 1 and try all over again. Of course this means that we need to break the requirement up into more subrequirements. Rather than increase $r(\tau)$ we will incorporate into the tree the guess as to where permission will be given.

To the details. We will define our tree of strategies as above, except that instead of one primary node we will have a whole layer of them. Call the root of the tree a *super-primary* node, working for $R_0$. If $\mu$ is a super-primary node, working for $R_e$, then its immediate successors are $\mu\hat{}k$ for $k \geqslant r(\mu)$; these nodes are now called primary nodes, working for the same requirement. We let $r(\mu\hat{}k) = k$. Beyond the primary nodes we build the tree as above; if $\tau$ is a primary node then its extensions are $\tau\hat{}k$ for $k > r(\tau)$, and the extensions of length $|\tau| + (i + 1)$ work for $R_{\tau,i}$, defined as above, with the same outcomes; extensions of length $|\tau| + (m(\tau) + 1)$ are super-primary nodes working for $R_{e+1}$.

For each super-primary node $\mu$ we will build a (total) counting $\langle o^\mu_s \rangle$ down $h$. By the recursion theorem (and the slow-down lemma) we can find a speed-up function $p$ such that for all $\mu$, the function $g$ promptly $p$-escapes each $\langle o^\mu_s \rangle$, each on infinitely many inputs.

For each super-primary node $\mu$ we will have a counter $n_s(\mu)$, starting with $n_0(\mu) = r(\mu)$. If $\mu$ is accessible at stage $s$ then its successor $\mu\hat{\ }n_s(\mu)$ is next accessible. We will let $o^\mu_s(n) = 0$ whenever $n < n_s(\mu)$, and $o^\mu_s(n) = h(n)$ whenever $n > n_s(\mu)$. As usual, unless otherwise stated, we will let $o^\mu_s(n) = o^\mu_{s-1}(n)$.

*Construction.* Suppose that a super-primary node $\mu$, working for $R_e$, is accessible at stage $s$, and that $|\mu| < s$. As mentioned above, we let $\tau = \mu\hat{\ }n_s(\mu)$ be next accessible. We define $\bar{A}_s(\tau)$ and $\bar{B}_s(\tau)$ as above, and so get the list $\langle \mathcal{C}_i, \mathcal{D}_i \rangle_{i<m(\tau)}$ and the subrequirements $R_{\tau,i}$. The instructions for $\tau$'s action are as above, except that whenever $\tau$ decides it would like to act, it first defines $o^\mu_s(n) = o^\mu_{s-1}(n)$ where $n = n_s(\mu)$. We then check to see if $g_{p(s)}(n) \neq g_s(n)$. If so, then $\tau$ can carry out the desired action. If not, then $\tau$ stops the stage and we set $n_{s+1}(\mu) = n_s(\mu) + 1$.

The rest of the construction is identical to the one above.

*Verification.* We observe how we need to augment the proofs of the lemmas above. Lemma 2.2 does not hold as written, but does hold provided that we assume that $\tau$ lies on the true path. The point is that if $\sigma$ from the proof of the lemma is a super-primary node then $\tau$ will not be accessible after stage $s$; the outcomes of a super-primary node only increase with time. If $\tau$ lies on the true path then nodes to the left of $\tau$ may be accessible, but not after the least stage at which $\tau$ is accessible.

Lemma 2.3 holds, with the same proof. However to show that the true path is infinite we now need to consider super-primary nodes. If $\mu$ is a super-primary node on the true path then the sequence $\langle n_s(\mu) \rangle$ must come to a limit: if $g$ promptly $p$-escapes $\langle o^\mu_s \rangle$ on an input $n \geqslant r(\mu)$ then we will never have $n_s(\mu) > n$. Of course we need to note that the calculation above ensures that $h(n)$ is large enough, so that whenever a primary node $\tau = \mu\hat{\ }n$ wants to act, we have $o^\mu_s(n) > 0$. The rest of the verifications follow as above to see that all requirements are met. Finally, we observe that all sets enumerated are computable from $g$: if $g_s(r) = g(r)$ then the first $r$ elements of $\bar{A}_s$ will never enter $A_0 \cup A_1$, and the same holds for $B$.

## 3. Prompt permission and embedding results

Prompt versions of permitting can also be adapted to other constructions we have been discussing. The main example is the embedding of the 1-3-1 lattice. The main idea is that if non-total $< \omega^\omega$-permission is given promptly then when balls enter the permitting bin, instead of appointing a trace for the bottom set $B$, we ask for prompt permission. If this is not given then the follower is cancelled. This yields:

THEOREM 3.1. *If* $\mathbf{d}$ *is promptly not totally* $< \omega^\omega$*-c.a. then there is an embedding of the 1-3-1 lattice in the c.e. degreed below* $\mathbf{d}$ *which maps the bottom element to* $\mathbf{0}$.

As mentioned above, a full reversal is impossible, since every high degree bounds such an embedding as well.

# Bibliography


[1] Klaus Ambos-Spies and Peter A. Fejer. Embeddings of $N_5$ and the contiguous degrees. *Ann. Pure Appl. Logic*, 112(2-3):151–188, 2001.

[2] 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.

[3] Bernie A. Anderson and Barbara F. Csima. A bounded jump for the bounded turing degrees. Submitted.

[4] Chris J. Ash and Julia F. Knight. *Computable structures and the hyperarithmetical hierarchy*, volume 144 of *Studies in Logic and the Foundations of Mathematics*. North-Holland Publishing Co., Amsterdam, 2000.

[5] George Barmpalias, Rodney G. Downey, and Noam Greenberg. Working with strong reducibilities above totally $\omega$-c.e. and array computable degrees. *Trans. Amer. Math. Soc.*, 362(2):777–813, 2010.

[6] Paul Brodhead, Rod Downey, and Keng Meng Ng. Bounded randomness. In *Computation, physics and beyond*, volume 7160 of *Lecture Notes in Comput. Sci.*, pages 59–70. Springer, Heidelberg, 2012.

[7] Peter Cholak, Richard Coles, Rod Downey, and Eberhard Herrmann. Automorphisms of the lattice of $\Pi^0_1$ classes: perfect thin classes and anc degrees. *Trans. Amer. Math. Soc.*, 353(12):4899–4924, 2001.

[8] Peter Cholak, Rod Downey, and Richard Shore. Intervals without critical triples. In *Logic Colloquium '95 (Haifa)*, pages 17–43. Springer, Berlin, 1998.

[9] Peter Cholak, Rodney G. Downey, and Stephen Walk. Maximal contiguous degrees. *J. Symbolic Logic*, 67(1):409–437, 2002.

[10] Richard J. Coles, Rodney G. Downey, and Geoff L. Laforte. Notes on wtt-jump and ordinal notations. Manuscript, 1998.

[11] Adam R. Day. Indifferent sets for genericity. *J. Symbolic Logic*, 78(1):113–138, 2013.

[12] David Diamondstone, Noam Greenberg, and Daniel Turetsky. Natural large degree spectra. *Computability*, 2(1):1–8, 2013.

[13] R. G. Downey and Carl G. Jockusch, Jr. **T**-degrees, jump classes, and strong reducibilities. 30:103–137, 1987.

[14] Rod Downey. Lattice nonembeddings and initial segments of the recursively enumerable degrees. *Ann. Pure Appl. Logic*, 49(2):97–119, 1990.

[15] Rod Downey and Noam Greenberg. Totally $< \omega^\omega$ computably enumerable and $m$-topped degrees. In *Theory and applications of models of computation*, volume 3959 of *Lecture Notes in Comput. Sci.*, pages 46–60. Springer, Berlin, 2006.

[16] Rod Downey and Noam Greenberg. Turing degrees of reals of positive effective packing dimension. *Inform. Process. Lett.*, 108(5):298–303, 2008.

[17] Rod Downey, Noam Greenberg, and Rebecca Weber. Totally $\omega$-computably enumerable degrees and bounding critical triples. *J. Math. Log.*, 7(2):145–171, 2007.

[18] Rod Downey and Richard A. Shore. Degree-theoretic definitions of the low$_2$ recursively enumerable sets. *J. Symbolic Logic*, 60(3):727–756, 1995.

[19] Rod Downey and Richard A. Shore. Lattice embeddings below a nonlow$_2$ recursively enumerable degree. *Israel J. Math.*, 94:221–246, 1996.

[20] Rod Downey and Sebastiaan A. Terwijn. Computably enumerable reals and uniformly presentable ideals. *MLQ Math. Log. Q.*, 48(suppl. 1):29–40, 2002. Dagstuhl Seminar on Computability and Complexity in Analysis, 2001.

[21] Rod G. Downey and Denis R. Hirschfeldt. Algorithmic randomness and complexity. 2009.

[22] Rodney G. Downey, Denis R. Hirschfeldt, André Nies, and Frank Stephan. Trivial reals. In *Proceedings of the 7th and 8th Asian Logic Conferences*, pages 103–131, Singapore, 2003. Singapore Univ. Press.

[23] 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.

[24] Rodney G. Downey, Carl G. Jockusch, Jr., 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.

[25] Rodney G. Downey and Geoffrey L. LaForte. Presentations of computably enumerable reals. *Theoret. Comput. Sci.*, 284(2):539–555, 2002. Computability and complexity in analysis (Castle Dagstuhl, 1999).

[26] Rodney G. Downey and Steffen Lempp. Contiguity and distributivity in the enumerable Turing degrees. *J. Symb. Logic*, 62(4):1215–1240, 1997.

[27] Richard L. Epstein, Richard Haas, and Richard L. Kramer. Hierarchies of sets and degrees below $\mathbf{0}'$. 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 32–48. Springer, Berlin, 1981.

[28] Y. L. Ershov. A certain hierarchy of sets. I. *Algebra i Logika*, 7(1):47–74, 1968.

[29] Yuri L. Ershov. A certain hierarchy of sets. II. *Algebra i Logika*, 7(4):15–47, 1968.

[30] Yuri L. Ershov. A certain hierarchy of sets. III. *Algebra i Logika*, 9:34–51, 1970.

[31] Santiago Figueira, Denis Hirschfeldt, Joseph S. Miller, Keng Meng Ng, and André Nies. Counting the changes of random $\Delta^0_2$ sets. In *Programs, proofs, processes*, volume 6158 of *Lecture Notes in Comput. Sci.*, pages 162–171. Springer, Berlin, 2010.

[32] Johanna N. Y. Franklin, Noam Greenberg, Frank Stephan, and Guohua Wu. Anti-complex sets and reducibilities with tiny use. *J. Symbolic Logic*, 78(4):1307–1327, 2013.

[33] Johanna N. Y. Franklin and Keng Meng Ng. $\omega$-change randomness and weak Demuth randomness. *J. Symb. Log.*, 79(3):776–791, 2014.

[34] Noam Greenberg. The role of true finiteness in the admissible recursively enumerable degrees. *Mem. Amer. Math. Soc.*, 181(854):vi+99, 2006.

[35] Noam Greenberg, Denis R. Hirschfeldt, and André Nies. Characterizing the strongly jump-traceable sets via randomness. *Adv. Math.*, 231(3-4):2252–2293, 2012.

[36] Noam Greenberg and André Nies. Benign cost functions and lowness properties. *J. Symbolic Logic*, 76(1):289–312, 2011.

[37] 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.

[38] Carl G. Jockusch, Jr. and Robert I. Soare. A minimal pair of ${\Pi_1}^0$ classes. *J. Symbolic Logic*, 36:66–78, 1971.

[39] Bjorn Kjos-Hanssen, Joseph S. Miller, and Reed Solomon. Lowness notions, measure and domination. *J. Lond. Math. Soc. (2)*, 85(3):869–888, 2012.

[40] Stephen C. Kleene. On notations for ordinal numbers. *J. Symbolic Logic*, 3(4):150–155, 1938.

[41] Martin Kummer. Kolmogorov complexity and instance complexity of recursively enumerable sets. *SIAM J. Comput.*, 25(6):1123–1143, 1996.

[42] Alistair H. Lachlan. Embedding nondistributive lattices in the recursively enumerable degrees. In *Conference in Mathematical Logic—London '70 (Proc. Conf., Bedford Coll., London, 1970)*, pages 149–177. Lecture Notes in Math., Vol. 255. Springer, Berlin, 1972.

[43] Alistair H. Lachlan and Robert I. Soare. Not every finite lattice is embeddable in the recursively enumerable degrees. *Adv. in Math.*, 37(1):74–82, 1980.

[44] Steffen Lempp and Manuel Lerman. A finite lattice without critical triple that cannot be embedded into the enumerable Turing degrees. *Ann. Pure Appl. Logic*, 87(2):167–185, 1997. Logic Colloquium '95 Haifa.

[45] Steffen Lempp, Manuel Lerman, and D. Reed Solomon. Embedding finite lattices into the computably enumerable degrees—a status survey. In *Logic Colloquium '02*, volume 27 of *Lect. Notes Log.*, pages 206–229. Assoc. Symbol. Logic, La Jolla, CA, 2006.

[46] Manuel Lerman. Admissible ordinals and priority arguments. In *Cambridge Summer School in Mathematical Logic (Cambridge, 1971)*, pages 311–344. Lecture Notes in Math., Vol. 337. Springer, Berlin, 1973.

[47] Manuel Lerman. The embedding problem for the recursively enumerable degrees. In *Recursion theory (Ithaca, N.Y., 1982)*, volume 42 of *Proc. Sympos. Pure Math.*, pages 13–20. Amer. Math. Soc., Providence, RI, 1985.

[48] Donald A. Martin. Classes of recursively enumerable sets and degrees of unsolvability. *Z. Math. Logik Grundlagen Math.*, 12:295–310, 1966.

[49] Joseph S. Miller. Degrees of unsolvability of continuous functions. *J. Symbolic Logic*, 69(2):555–584, 2004.

[50] André Nies. Lowness properties and randomness. *Adv. Math.*, 197(1):274–305, 2005.

[51] André Nies. Reals which compute little. In *Logic Colloquium '02*, volume 27 of *Lect. Notes Log.*, pages 261–275. Assoc. Symbol. Logic, La Jolla, CA, 2006.

[52] 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.

[53] Gerald E. Sacks. *Higher recursion theory.* Perspectives in Mathematical Logic. Springer-Verlag, Berlin, 1990.

[54] Joseph R. Shoenfield. On degrees of unsolvability. *Ann. of Math. (2)*, 69:644–653, 1959.

[55] Richard A. Shore. The recursively enumerable $\alpha$-degrees are dense. *Ann. Math. Logic*, 9(1-2):123–155, 1976.

[56] Richard A. Shore. Natural definability in degree structures. In *Computability theory and its applications (Boulder, CO, 1999)*, volume 257 of *Contemp. Math.*, pages 255–271. Amer. Math. Soc., Providence, RI, 2000.

[57] Robert I. Soare. *Recursively enumerable sets and degrees.* Perspectives in Mathematical Logic. Springer-Verlag, Berlin, 1987. A study of computable functions and computably generated sets.

[58] Clifford Spector. Recursive well-orderings. *J. Symb. Logic*, 20:151–163, 1955.

[59] Frank Stephan and Guohua Wu. Presentations of $k$-trivial reals and kolmogorov complexity. In *Computability in Europe 2005: New Computational Paradigms*, volume 3526 of *Lecture Notes in Comput. Sci.*, pages 461–469. Springer, 2005.

[60] Sebastiaan A. Terwijn and Domenico Zambella. Computational randomness and lowness. *J. Symbolic Logic*, 66(3):1199–1205, 2001.

[61] Stephen M. Walk. *Toward the definability of the array noncomputable degrees.* PhD thesis, University of Notre Dame, 1999.

[62] Barry Weinstein. *On Embedding of the Lattice 1-3-1 into the Recursively Enumerable Degrees.* PhD thesis, University of California, Berkeley, 1988.

[63] Guohua Wu. Prefix-free languages and initial segments of computably enumerable degrees. In *Computing and combinatorics (Guilin, 2001)*, volume 2108 of *Lecture Notes in Comput. Sci.*, pages 576–585. Springer, Berlin, 2001.