WORKING WITH STRONG REDUCIBILITIES ABOVE TOTALLY ω-C.E. AND ARRAY COMPUTABLE DEGREES

GEORGE BARMPALIAS, ROD DOWNEY, AND NOAM GREENBERG

Abstract. We investigate the connections between the complexity of a c.e. set, as calibrated by its strength as an oracle for Turing computations of functions in the Ershov hierarchy, and how strong reducibilities allows us to compute such sets. For example, we prove that a c.e. degree is totally ω-c.e. iff every set in it is weak truth-table reducible to a hypersimple, or ranked, set. We also show that a c.e. degree is array computable iff every left-c.e. real of that degree is reducible in a computable Lipschitz way to a random left-c.e. real (an Ω-number).

1. Introduction

Whilst this paper is underpinned by a number of technical results, its principal goal is to bring into focus a new and potentially important programme in computability theory; and, in particular, computability theory applied to algorithmic randomness and computable model theory. As we will see, this paper will demonstrate interactions between a number of distinct areas of computability theory, including strong reducibilities, Turing degrees on the computably enumerable sets, dynamic and static properties of computably enumerable sets, various natural classes of degrees, algorithmic randomness, Π^0_1 classes and their ranked points, and computable model theory. We mention that the classes of degrees discussed include the totally ω-c.e. degrees and the array computable degrees[1], classes which have had significant recent interest and have deep connections with algorithmic randomness as witnessed by, for example, Kummer [29], Downey and Greenberg [14], and several other articles (see Downey and Hirschfeldt [16]).

Traditionally, one classifies classes of Turing degrees by investigating their computational power as oracles, as is reflected by the structure of the Turing degrees *below* the degree in question. There are very well known classical examples of this phenomenon. For example, a c.e. set A is *high* if it is indistinguishable from the halting problem as measured by the jump operator, in that $A' \equiv_T \emptyset''$. Used as oracles, we would expect that high sets should have computational power similar to that of $\emptyset'$ and this intuition is justified by results such as the coincidence of the high degrees with the degrees which contain maximal c.e. sets[2]. Thus, high sets have enough computational power to enable us to carry out the construction of a maximal set. Another illustration comes from lowness properties, where L is low if $L' \equiv_T \emptyset'$. Now in terms of the jump operator, low sets are indistinguishable

We wish to thank Frank Stephan for his permission to include Theorem 1.12 and its proof. All authors were supported by the Marsden fund of New Zealand.

[1]Precise definitions will soon follow.

[2]Recall that a co-infinite set M is maximal if for all c.e. $W \supset M$ either $W - M$ is finite or $\omega - W$ is finite.

from the computable sets. Tying this to our example above, no low c.e. set can be maximal.

The focus of this paper is not on what can be performed *below* some degree, but the dual question: how complicated is it to *compute* a set outside a given class, and so how do degrees in these classes behave with respect to degrees *above* them? A nice example in algorithmic randomness comes from the work on algorithmic lowness, where it has been shown that the only sets D so computationally feeble that there is an $R \geqslant_T D$ which is random relative to D, are precisely the class of "K-trivial" reals. The precise definitions here are not important to our task at hand, but it suffices to say that this class is an important subclass of the low sets, whose members are low in terms of their initial segment complexity (more details can be found in Nies [32]).

For our paper, we begin with the following examples of totally ω-c.e. degrees and their uniform analogue, the array computable degrees. To make the definitions precise, recall that Shoenfield's limit lemma states that a function $f\colon \omega \to \omega$ is computable from the degree $\mathbf{0}'$ of the halting problem if and only if it has a *computable approximation*: there is a uniformly computable sequence of functions $\langle f_s \rangle$ which converge pointwise to f (under the discrete topology on ω). That is, for all n, the mind-change set

$$\{s : f_s(n) \neq f_{s+1}(n)\}$$

is finite, and $f_s(n) = f(n)$ for almost all s. Every computable approximation $\langle f_s \rangle$ gives rise to the mind-change function:

$$m_f(n) = \#\{s : f_s(n) \neq f_{s+1}(n)\}.$$

Following Ershov, we classify the complexity of a Δ^0_2 function by how quickly a computable approximation for f can settle; that is, what kind of functions dominate the mind-change function of some approximation for f. An *order* is a non-decreasing and unbounded function; we say that a function f is h-c.e. if it has some computable approximation such that h bounds the mind-change function of that approximation. Finally, we say that a function f is *ω-c.e.* if it is h-c.e. for some computable order h.

Array computability and total ω-c.e.ness are defined as notions of oracular weakness, in the sense that the definitions say that all functions that can be computed by the degree are simple in the sense that they have reasonable approximations. A c.e. degree $\mathbf{d}$ is *array computable* if there is some computable order h such that every $f \leqslant_T \mathbf{d}$ is h-c.e. This notion was defined and investigated by Downey, Jockusch and Stob [19], who showed that the array computable c.e. degrees are exactly the c.e. degrees for which a certain notion of permitting (called "multiple permitting") fails, with immediate consequences for the computational power of $\mathbf{d}$. For example, they showed that a c.e. degree is array non-computable iff it computes a Martin-Pour-El theory. Since it is again not important to our results, we will simply mention that a Martin-Pour-El theory is a special type of an essentially undecidable axiomatizable theory with very few axiomatizable extensions, and is an analog of a maximal set for theories. Thus we see that the array non-computable degrees have precisely the power to compute these objects.

Later, this notion of array non-computability was expanded in [21] to the degrees in general by replacing approximations with domination; a Turing degree $\mathbf{d}$ is array computable if there is some ω-c.e. function which dominates all functions

computable in $\mathbf{d}$. It was shown in [21] that every array non-computable degree is the join of a minimal pair, and a degree is array non-computable iff it bounds a *pb-generic* real, which is defined as a real which is Cohen generic for dense sets which can be accessed by *primitive recursive* functions.

The class of *totally ω-c.e.* degrees was defined and investigated by Downey, Greenberg and Weber in [15] and later in [12]. These are the c.e. degrees $\mathbf{d}$ such that every function $f \leqslant_T \mathbf{d}$ is ω-c.e. In [15], methods of permitting and anti-permitting arguments were developed for the class of totally ω-c.e. degrees and it was shown that a c.e. degree is not totally ω-c.e. iff it bounds a critical triple (in the c.e. degrees). Here a critical triple is a configuration in the c.e. degrees representing a certain blockage for embeddings. To wit, $\mathbf{a}, \mathbf{b}, \mathbf{c}$ form a critical triple iff $\mathbf{b} \cup \mathbf{a} = \mathbf{b} \cup \mathbf{c}$ and for all $\mathbf{d} \leqslant \mathbf{a}, \mathbf{c}$, $\mathbf{d} \leqslant \mathbf{b}$. Notice that this gives a natural definition of the totally ω-c.e. degrees in the c.e. degrees. A generalisation of the notion of totally ω-c.e. degree to the degrees has not yet been investigated, but we mention that it is true that a c.e. degree $\mathbf{d}$ is totally ω-c.e. iff every $f \leqslant_T \mathbf{d}$ is dominated by some ω-c.e. function.

Both classes are defined in terms of strength as an oracle for computations, and so was the flavour of the results we mentioned about highness and lowness. Focusing on the dual question for these classes: how do these classes behave with respect to degrees *above* them? A result of this type is that array non-computable degrees join to every degree above them (Downey, Jockusch and Stob [21]) and so cannot have strong minimal covers; Ishmukhametov [24] improved this to show that the c.e. degrees which have strong minimal covers are exactly the array computable ones.

In this paper we continue this line of investigation. We tie these classes with another theme that has been central in the investigation of the array computable and totally ω-c.e. degrees: the interplay between Turing and stronger reducibilities, such as weak truth-table reducibility. Recall that a *wtt-functional* is a Turing functional Γ which is accompanied by a computable function γ which bounds the use of Γ-computations; in other words, for all $(\sigma, \tau) \in \Gamma$ (which express that if σ is an initial segment of an oracle X, then τ is an initial segment of Γ^X) we have $|\sigma| \leqslant \gamma(|\tau|)$. For two sets A, B of natural numbers, $A \leqslant_{\mathrm{wtt}} B$ if there is a wtt-functional Γ such that $\Gamma^B = A$.

The fact that a function is ω-c.e. iff it is wtt-reducible to the halting problem signaled an interesting relationship between the weakness notions in the Turing degrees and weak truth-table reducibility. For example, a c.e. degree is not totally ω-c.e. iff it computes a *wtt-triple*: sets A_0, A_1 and B such that $A_0 \equiv_T A_1$, $A_i \not\leqslant_T B$ but if $C \leqslant_{\mathrm{wtt}} A_0, A_1$ then $C \leqslant_{\mathrm{wtt}} B$. There are similar results for array computability.

As we mentioned, in this paper we combine weakness notions for Turing degrees, question the effect of weakness for being the *result* of a computation, rather than its oracle, and put strong reducibilities in the mix. Thus, the main question we ask is: how hard is it for reals in certain classes to strongly compute strong or weak reals? Our results express that particular strong reducibilities help understand and characterise our two lowness notions. Thus total ω-c.e.-ness is intimately related with weak truth-table reducibility, and array computability is similarly related to the stronger computable Lipschitz reducibility.

In the first part of the paper, we concern ourselves with weak truth-table reductions and totally ω-c.e. degrees. The first result here was obtained by Chisholm,

Chubb, Harizanov, Hirschfeldt, Jockusch, McNicholl and Pingrey [8], who showed that every c.e. degree which is not totally ω-c.e. contains a c.e. set which is not wtt-reducible to any ranked set. Independently, Afshari, Barmpalias, Cooper and Stephan [1] showed that if $\mathbf{d}$ is totally ω-c.e. then every $A \leqslant_T \mathbf{d}$ is wtt-reducible to a hypersimple set; but Barmpalias [2, 3] showed that not every c.e. set is wtt-reducible to a hypersimple set. We prove:

Theorem 1.1. *The following are equivalent for a c.e. degree* $\mathbf{d}$*:*

(1) *Every set in* $\mathbf{d}$ *is wtt-reducible to a ranked set.*
(2) *Every set in* $\mathbf{d}$ *is wtt-reducible to a hypersimple set.*
(3) *Every set in* $\mathbf{d}$ *is wtt-reducible to a proper initial segment of a computable, scattered linear ordering.*
(4) $\mathbf{d}$ *is totally* ω*-c.e.*

Moreover, the equivalence still holds if in any of (1), (2) or (3), "set" is replaced by "c.e. set".

We recall the definitions. A set is *ranked* if it is an element of some countable Π^0_1 class (and so it has a Cantor-Bendixson rank). A linear ordering is *scattered* if it doesn't contain a copy of the rationals (and so repeating the Hausdorff derivative leaves an empty kernel at the end). A set $A \subset \omega$ is *hyperimmune* if it is infinite, and whenever $\langle F_n \rangle$ is a computable sequence of pairwise disjoint finite sets, there is some n such that $F_n \cap A$ is empty. Equivalently, the function which maps n to the n^{th} element of A (by magnitude) is not dominated by any computable function. Finally, a c.e. set is *hypersimple* if its complement is hyperimmune.

We remark that weak truth-table reducibility is exactly the right kind of reducibility which gives non-trivial results in this context. This is because every non-zero c.e. degree contains a hypersimple set and every c.e. set is Turing reducible to a ranked set; but if $A \leqslant_{\text{tt}} B$ and B is ranked then so is A.

Some connections between the notions are known:

Lemma 1.2 (Chisholm et. al. [8])**.** *Any initial segment of a scattered, computable linear ordering is ranked.*

Lemma 1.3. *If A is c.e. and non-computable, and is the ω-part of a computable linear ordering of order-type $\omega + \omega^*$, then A is hypersimple.*

Proof. We note that if there is a computable function f such that for all n,

$$|(\omega \setminus A) \cap f(n)| \geqslant n,$$

then we can enumerate $\omega \setminus A$ by listing, for all n, the n rightmost points of $L \upharpoonright f(n)$. □

Theorem 1.1 follows from the following two propositions, which in fact prove a little more.

Proposition 1.4. *If* $\mathbf{d}$ *is a c.e. degree which is not totally ω-c.e., then there is some c.e. set $B \leqslant_T \mathbf{d}$ which is not wtt-reducible to any hypersimple set.*

Proposition 1.5. *If* $\mathbf{d}$ *is a c.e. degree which is totally ω-c.e., then every $B \leqslant_T \mathbf{d}$ is wtt-reducible to a c.e. set which is the ω-part of a computable linear ordering of order-type $\omega + \omega^*$.*

These propositions are proved in section 2.

Proof of Theorem 1.1. Let $\mathbf{d} > \mathbf{0}$ be a c.e. degree. If $\mathbf{d}$ is totally ω-c.e., then by Proposition 1.5, every $C \in \mathbf{d}$ is wtt-reducible to a c.e. set A which is a proper, non-empty initial segment of a computable linear ordering of order-type $\omega + \omega^*$. Of course A is not computable, and so A is both ranked and hypersimple.

If $\mathbf{d}$ is not totally ω-c.e., then by Chisholm et. al., there is some c.e. $A \in \mathbf{d}$ which is not wtt-reducible to any ranked set, and hence not to an initial segment of a scattered, computable linear ordering. By Proposition 1.4, there is some c.e. $B \leqslant_T \mathbf{d}$ which is not wtt-reducible to any hypersimple set; if $D \in \mathbf{d}$ is any c.e. set then $B \oplus D$ is in $\mathbf{d}$ and again not wtt-reducible to any hypersimple set. □

We mention that Proposition 1.4 does not simply follow from the result of Chisholm et. al.; this is because there are hypersimple sets which are not wtt-reducible to any ranked sets. We prove that fact in section 3. On the other hand, Proposition 1.4 does not imply the result of Chisholm et. al.; in the same section, we show that there is a ranked c.e. set which is not wtt-reducible to any hypersimple set.

In the second part of the paper, we turn to algorithmic randomness and in particular some natural notions of relative randomness. Recall that a Martin-Löf test is a method of coding all effective statistical tests. To wit, a computable collection of c.e. open sets $T = \{U_n : n \in \omega\}$ is a Martin-Löf test iff the Lebesgue measure of U_n, $\mu(U_n)$ is bounded by 2^{-n} for all n. Then a real A passes the test T iff $A \notin \cap_n U_n$. We say that a real is Martin-Löf random iff it passes all Martin-Löf tests[3]. A universal Martin-Löf test (V_n) is a Martin-Löf test with the property that $\cap_n V_n$ contains (and so, coincides with the set of) all non-random reals. The existence of a universal Martin-Löf test was an early result of Martin-Löf (see e.g. [32]). Martin-Löf randomness is a natural and robust notion of randomness in that it coincides with other methods of defining algorithmic randomness. For example a real should be random iff all its initial segments are incompressible, so that (like white noise) they have no patterns allowing compression. This intuition is proven correct by Schnorr's theorem, which says that A is Martin-Löf random if and only if for all n, $K(A \upharpoonright n) = n + O(1)$, where $A \upharpoonright n$ denotes the first n bits of A and K denotes prefix free Kolmogorov complexity[4].

Schnorr's theorem suggests a natural method of calibrating randomness of reals: $A \leqslant_K B$ iff for all n, $K(A \upharpoonright n) \leqslant K(B \upharpoonright n) + O(1)$. Many measures of relative randomness implying this measure have been analysed. Of interest to us here is one inspired by strong reducibilities. In [17, 18] a strengthening of weak truth-table reducibility, namely computations where the use on the oracle on argument n is $n + c$ for some constant c, was suggested as a possible measure of relative randomness. This reducibility has appeared in the literature with various names, e.g. *strong weak truth table* [17, 18], *computable Lipschitz* (due to a characterization

[3]in the following, we sometimes say 'random' meaning 'Martin-Löf random'.

[4]Prefix-free Kolmogorov complexity is a modification of plain complexity C which captures the intentional meaning of Kolmogorov complexity. Recall that for a machine M the Kolmogorov complexity $C_M(\sigma)$ of a string σ is the length of the shortest τ with $M(\tau) = \sigma$. Up to a constant, there is a universal machine U with $C_U(\sigma) \leq C_M(\sigma) + O(1)$, for all σ, allowing us to define the (plain) Kolmogorov complexity. Prefix-free complexity solves certain deficiencies in this definition, and simply asks that the domains of the machines have the property that if $U(\tau) \downarrow$ then for all strings ρ comparable with τ, $U(\rho) \uparrow$. We refer the reader to Li-Vitanyi [30], Downey-Hirschfeldt [16] for more details.

of it in terms of effective Lipschitz functions) [4] and *linear* [6]. We will adopt the terminology in [4] and note it as $\leqslant_{\mathrm{cl}}$. It is nearly obvious that $\leqslant_{\mathrm{cl}}$ is a measure of relative randomness since a programme computing $n + c$ bits of B will compute n bits of A.

In [35], Yu and Ding proved that there are two left-c.e. reals (i.e. limits of computable increasing sequences of rationals) which had no common upper bound in the cl-degrees. In [7], Barmpalias and Lewis showed that there is a left-c.e. real which is not cl-reducible to any Martin-Löf random left-c.e. real.

The Yu-Ding Theorem and the Barmpalias-Lewis Theorem should be viewed in the context of the Kučera-Slaman Theorem [28] which shows that that all random left-c.e. reals have the same Solovay degree (which is the greatest Solovay degree of left-c.e. reals). Here α is Solovay reducible to β iff there is a constant c and a partial computable φ such that for all rationals $q < \beta$, $\varphi(q) \downarrow$ and $c|\beta - q| > \alpha - \varphi(q)$. Thus, the Kučera-Slaman Theorem says that a computable sequence of rationals converging to some version of Ω can be efficiently converted into a sequence converging to any other version of Ω at more or less the same rate. The Yu-Ding-Barmpalias-Lewis material shows that there is no way to efficiently convert the *bits* of one version of Ω to some arbitrary left-c.e. real.

Both the Yu-Ding theorem and the Barmpalias-Lewis theorem characterise array computability.

Theorem 1.6. *The following are equivalent for a c.e. degree* $\mathbf{d}$*:*

(1) *There are left-c.e. reals* $\alpha_0, \alpha_1 \in \mathbf{d}$ *which have no common upper bound in the cl-degrees of left-c.e. reals.*
(2) *There is a left-c.e. real* $\alpha \in \mathbf{d}$ *which is not cl-reducible to any random left-c.e. real.*
(3) *There is a set* $A \in \mathbf{d}$ *which is not cl-reducible to any random left-c.e. real.*
(4) $\mathbf{d}$ *is array non-computable.*

To make the technical details of the proofs slightly simpler, we often work with an even more restrictive reducibility than computable Lipschitz: *identity bounded Turing* reducibility (ibT or $\leqslant_{\mathrm{ibT}}$ for short) is a computable Lipschitz reduction for which the constant c is 0. This reducibility was introduced by Soare in connection with applications of computability theory to differential geometry (see [10, 5]). Even though the degree structures are different, for our purposes, we may work with either:

Lemma 1.7.

(1) *Left-c.e. reals* α_0 *and* α_1 *have a common upper bound in the cl-degrees of left-c.e. reals iff they have a common upper bound in the ibT-degrees of the left-c.e. reals.*
(2) *A set* A *is cl-reducible to a random left-c.e. real iff it is ibT-reducible to a random left-c.e. real.*

Proof. If $\alpha \leqslant_{\mathrm{cl}} \beta$ with constant c then $\alpha \leqslant_{\mathrm{ibT}} \beta \upharpoonright [c, \infty)$. If β is random then so is $\beta \upharpoonright [c, \infty)$. □

Theorem 1.6 follows from the following four Propositions, which will be proved in section 4. Again we prove slightly more.

Proposition 1.8. *If* $\mathbf{d}$ *is c.e. and array non-computable, then there are left-c.e. reals* $\alpha_0, \alpha_1 \in \mathbf{d}$ *which have no common upper bound in the ibT degrees of left-c.e. reals.*

For the proof of this proposition, we give a significantly simplified account of the verifications of the construction of Yu and Ding, which appeared in [35] and [4].

Proposition 1.9. *If* A_0 *and* A_1 *are sets, both of which have c.e., array computable degree, then there is a left-c.e. real* β *such that* $A_0, A_1 \leqslant_{\mathrm{ibT}} \beta$.

The fact that we don't need $A_0 \oplus A_1$ to have array computable degree may be somehow related to the fact [19] that the wtt degrees of c.e. sets which do not contain array non-computable *sets* are closed under join.

Proposition 1.10. *If* $\mathbf{d}$ *is c.e. and array non-computable, then there is some left-c.e. real* $\alpha \in \mathbf{d}$ *which is not ibT-reducible to any random left-c.e. real.*

Here we give a construction which is significantly simpler than that which appeared in [7].

Proposition 1.11. *If* $\mathbf{d}$ *is c.e. and array computable, and* $A \leqslant_T \mathbf{d}$*, then* A *is ibT-reducible to some random left-c.e. real.*

To clarify the situation regarding the cl-degrees of random left-c.e. reals, Frank Stephan has proved the following fact:

Theorem 1.12. *There are two random left-c.e. reals which are not cl-equivalent.*

We give Stephan's proof in section 6.

Finally, we somewhat extend Proposition 1.10 beyond the c.e. degrees. We remark that Hirschfeldt (see [16]) has constructed a real (not left-c.e.) which is not ibT-reducible to any random real, indeed to any complex real. Recall that a set A is called *complex* (by Kjos-Hanssen, Merkle and Stephan, [25]) if there is a nondecreasing unbounded computable function f such that $K(A \upharpoonright n) > f(n)$ for all n; thus this is a weakening of the notion of randomness. Using Hirschfeldt's technique and sufficient genericity which is available from non-GL_2 permitting (recall that a degree $\mathbf{d}$ is called *generalised low*$_2$ if $\mathbf{d}'' \leqslant (\mathbf{d} \vee \mathbf{0}')'$; equivalently, every $f \leqslant_T (\mathbf{d} \vee \mathbf{0}')$ is not dominated by some $g \leqslant_T \mathbf{d}$), we prove the following in section 5:

Theorem 1.13. *If* $\mathbf{d}$ *is not generalised low*$_2$ *then there is some* $A \leqslant_T \mathbf{d}$ *which is not ibT-reducible to any complex real.*

This result complements a line of recent knowledge about using strong reducibilities with random oracles. Early on, Gács [23] and Kučera [26, 27] showed that every real is wtt-reducible to a random one. In fact, Gács proved that the use of the reduction can be $n + o(n)$, that is, asymptotically close to an ibT reduction (see [31]). Recently, Doty [11] showed that for every $A \in 2^\omega$ there is some random $R \geqslant_T A$ such that the limit superior of the use of computing $A \upharpoonright n$ from R, divided by n, is exactly the effective packing dimension $\mathrm{Dim}(A)$ of A. Thus if $\mathrm{Dim}(A) < 1$ then there is some random $R \geqslant_{\mathrm{ibT}} A$. For example, if $\mathbf{d}$ is c.e. traceable[5] then every

[5] i.e. there is an order function h such that every function f which is computable from $\mathbf{d}$ has a c.e. trace with bound h. A c.e. trace with bound h is a uniformly c.e. sequence of sets $\langle T_n \rangle$ such that $|T_n| \leq h(n)$ for all n. We say that $\langle T_n \rangle$ is a trace for f if $f(n) \in T_n$ for each n.

$A \in \mathbf{d}$ has effective packing dimension zero [14] and so every $A \in \mathbf{d}$ is ibT-reducible to a random, hence to a complex, real. The space between the c.e. traceable degrees and the non-GL_2 degrees remains a mystery. So we ask:

Question 1.14. *Which Turing degrees contain sets which are not ibT-reducible to random (or complex) reals?*

The first step, we suspect, would be to analyse the array non-computable degrees.

2. Weak truth-table reductions and totally ω-c.e. degrees

2.1. **Non-totally ω-c.e. permitting.** Here we prove Proposition 1.4: if $\mathbf{d}$ is c.e. and not totally ω-c.e. then there is some c.e. $A \leqslant_T \mathbf{d}$ which is not weak-truth-table reducible to any hypersimple set. To do so, we apply non-totally ω-c.e. permitting to the construction of a c.e. set which is not weak-truth-table reducible to any hypersimple set. To make the argument clear, we first review that construction, which appeared in [2, 3].

Let $\langle \Phi_e \rangle$ be an effective enumeration of all truth-table functionals: Φ_e is a Turing functional whose use function is bounded by the computable function φ_e (of course, some of the use functions may be partial, in which case we assume the corresponding computations do not converge). In enumerating the desired set A, the requirements we need to meet are

$R_{e,i}$**:** If $\Phi_e^{W_i} = A$ then W_i is not hypersimple.

The strategy for a single requirement $R_{e,i}$ is the following. We inductively define sets of followers $X_0, X_1, \ldots$ such that $\max X_n < \min X_{n+1}$ which will be used for diagonalisation against $\Phi_e^{W_i} = A$. If the wtt reduction $\Phi_e^{W_i}$ is total, then we can compute an increasing sequence $u_0 < u_1 < u_2 < \cdots$ such that for all n, for all $x \in X_n$, the use of the computation $\Phi_e^{W_i}(x)$ is smaller than u_n. The key is that the size of the sets of followers X_n grows fast: we ensure that $|X_n| > u_{n-1}$. Then if W_i is indeed hypersimple then at some stage of the construction we will discover some n such that the entire interval $[u_{n-1}, u_n)$ is a subset of W_i. The size of X_n then allows us to keep diagonalising against any future configuration of $W_i \upharpoonright u_n$, since there are at most u_n of those.

The sets of followers for different requirements can be kept disjoint, and so there is no interaction between distinct requirements, which act completely independently. Thus we have no need for the priority mechanism.

We now add the permitting component and describe the proof. It is quite straightforward. Say we are given a c.e. set D and some $g = \Gamma(D)$ which has no ω-c.e. approximation. Each requirement proceeds as before, setting up sets of followers X_n and computing the sequence of u_n's. When we want to start enumerating followers from some X_n into A (upon discovery that $[u_{n-1}, u_n) \subseteq W_i$) we know that we need at most $|X_n|$ many permissions from D to successfully meet the requirement. Now we know that if W_i is hypersimple then in fact there are infinitely many n such that $[u_{n-1}, u_n) \subseteq W_i$. So if we tie the D-use for permitting $x \in X_n$ to the D-use of computing $g \upharpoonright a_n$ (via Γ) then failure to obtain permission on each of these n's will yield an ω-c.e. approximation for g.

Construction. We now give the formal details. We give the instructions for a single requirement $R_{e,i}$, because again, there is no interaction between distinct requirements, even though they implicitly collaborate in building the reduction from A to

D. Let, for $a < \omega$, $\gamma(a)$ (at a stage s) be the current use of computing $g \upharpoonright a$ via $\Gamma(D)$. [Whenever the construction asks for a value $\gamma(a)$ we run the enumeration of D forward until we get $|\mathrm{dom}\,\Gamma(D)| > a$.]

Setting up X_n:

> Let X_n consist of $u_{n-1}+1$ large numbers which have not been used before. Also choose a large number a_n.

Attacking with X_n at stage s:

> Pick any $x \in X_n \setminus A$ and let $v = \gamma(a_n)$ (at stage s). Wait for the least stage $t > s$ such that $D_s \upharpoonright v \neq D_t \upharpoonright v$. If such a stage appears, enumerate x into A at that stage.

The *instructions for $R_{e,i}$* consist of two parts, which are performed *in parallel*.

(1) Let $u_{-1} = 0$. Set up X_0. After setting up X_n, wait for a stage at which $\Phi_e^{W_i}$ agrees with $A \upharpoonright \max X_n + 1$. Let $u_n = \varphi_e(\max X_n + 1)$ be the use of that computation, and use that value to set up X_{n+1}.
(2) For any n, if $[u_{n-1}, u_n) \subseteq W_i$ and $\Phi_e^{W_i}$ agrees with $A \upharpoonright \max X_n + 1$ (and we are not currently attacking with X_n), then attack with X_n.

Again note that setting up new X_n's proceeds while possible attacks are made with older X_m's. Also several attacks (with distinct X_m's) may be performed simultaneously.

Verification.

Lemma 2.1. *$A \leqslant_T D$.*

Proof. Let $x < \omega$. To find whether $x \in A$, first go to stage x and see if there is some requirement $R_{e,i}$ which has chosen x to be an element of a follower set X_n belonging to this requirement. If not, then $x \notin A$. Otherwise, let a_n be the parameter chosen when X_n was set up. Wait for a stage s at which $g \upharpoonright a_n \subset \Gamma(D)$ via a D-correct computation. Then $x \in A$ iff $x \in A_s$. □

Lemma 2.2. *A is not wtt-reducible to any hypersimple set.*

Proof. Suppose for contradiction that W_i is hypersimple and $\Phi_e^{W_i} = A$ (with computable bound φ_e on the use). Since $\Phi_e^{W_i}$ is total, the requirement $R_{e,i}$ sets up follower sets X_n for every $n < \omega$ and computes the associated sequence $\langle u_n \rangle$. By assumption, the set

$$B = \{n < \omega : [u_{n-1}, u_n) \subset W_i\}$$

is infinite. For each $n \in B$, an attack with X_n commences at some stage. Since $\Phi_e^{W_i}$ computes A correctly, for no $n \in B$ do we get to enumerate all of X_n into A; we cannot get stuck waiting for another agreement between $\Phi_e^{W_i}$ and $A \upharpoonright \max X_n + 1$, so it must be that for each $n \in B$, an attack with X_n gets stuck waiting for permission from D.

This means that if we approximate, for $n \in B$, $g \upharpoonright a_n$ to be its value at a stage at which an attack with X_n is started, then for each $n \in B$ there are at most $|X_n|$ many mind-changes about $g \upharpoonright a_n$. Since the sequence $\langle a_n \rangle_{n \in B}$ is computable, this gives us an ω-c.e. approximation for g, and a contradiction. □

2.2. **Totally ω-c.e. anti-permitting.** We prove Proposition 1.5 under the simplifying hypothesis that the set B is c.e. Since the totally ω-c.e. degrees are downward closed, we simply let $B = D$ be a c.e. set of totally ω-c.e. degree, and show that there is some computable linear ordering L of order-type $\omega + \omega^*$ such that the ω-part of L is c.e. and wtt-computes D. The reader may be interested to note how the proof behaves like a mirror image of the proof of the previous section.

The proof follows the style of the "anti-permitting" arguments of [15, 12, 13]. The general idea is that we globally construct some function $\Gamma(D)$. If we knew some ω-c.e. approximation to $\Gamma(D)$ then we could use that approximation to limit changes in D (as we would not be able to change $\Gamma(D)$ too many times). In the present case we would use this to correctly compute D from a set A we are building. There is a small inconvenience in that although we know that $\Gamma(D)$ is ω-c.e., we cannot effectively get an ω-c.e. approximation to $\Gamma(D)$. That is why we opt for a non-uniform construction: for every guess at such an approximation, we build a separate set A and a reduction from D to A. The correct guess will build a successful reduction and a set with the desired property (here, being the ω-part of a computable linear ordering of type $\omega + \omega^*$). The only interaction between the different guesses is that together they need to construct $\Gamma(D)$; thus even if a guess is incorrect, we need to ensure that on the inputs under this guess' responsibility, $\Gamma(D)$ is defined.

The heart of the proof is the strategy, given an approximation for $\Gamma(D)$. Suppose that we are given a correct approximation $g_s(x)$ to $\Gamma(D)$ where the value of $g_s(x)$ changes at most $h(x)$ many times (with h computable). To begin, we choose a large number a_0, and set up $h(a_0) + 2$ many points in the computable linear ordering $<_L$ we are building. We call the collection of points we set up the first *layer* L_0 of $<_L$. To begin, we declare that A (which will be the ω-part of $<_L$) contains only the leftmost point of L_0. We define $\Gamma(D, a_0)$ with large use $\gamma(a_0)$. When we get confirmation that the current version of $D \upharpoonright \gamma(a_0)$ is correct by obtaining a stage s at which $g_s(a_0) = \Gamma(D, a_0)$, then we declare that $A \upharpoonright L_0$ (that is, the information that from the points in L_0, only the leftmost one is in A) computes $D \upharpoonright \gamma(a_0)$. [If D changes before we get such confirmation, then we start again with the new value of $D \upharpoonright \gamma(a_0)$. Note that if g_s is not a correct approximation for $\Gamma(D)$, then $D \upharpoonright \gamma(a_0)$ could be correct yet we would never receive confirmation for this fact. The construction for this incorrect guess at an approximation is then stuck. But note that $\Gamma(D, a_0)$ is nevertheless well-defined and so this eventuality doesn't spoil the totality of $\Gamma(D)$.]

We then proceed to choose the next "unpermitting number" a_1 and lay the next layer L_1 of $h(a_1) + 2$ points of $<_L$ which will be put in the cut between A and its complement: namely, between the leftmost point of L_0 and the next point of L_0. We now repeat the process by enumerating the leftmost point of L_1 into A, defining $\Gamma(D, a_1)$ with large use $\gamma(a_1)$ and upon confirmation by $g_s(a_1)$, we stipulate that $A \upharpoonright (L_0 \cup L_1)$ computes $D \upharpoonright \gamma(a_1)$.

Of course, the main issue is what happens if D does change, making the reduction to A incorrect. We then need to change A to react to the D-change. Suppose, for example, that $D \upharpoonright \gamma(a_0)$ changed from its original value. We then decide to enumerate the second leftmost point of L_0 into A. This means that all the subsequent layers we put down have to be abandoned (and enumerated into A). This does not affect the order-type of $<_L$ because it means just adding finitely many

points to the ω-part A. But it does invalidate all the computations we commited to, of $D \upharpoonright \gamma(a_0)$ from $A \upharpoonright L_0$, $D \upharpoonright \gamma(a_1)$ from $A \upharpoonright (L_0 \cup L_1)$, etc., and allows us to define new ones. The fact that we only make such commitments when a confirmation of D is obtained, and the fact that g_s is indeed an h-c.e. approximation for $\Gamma(D)$, tells us that the process will halt before we run out of points in the layer.

There is one last delicate matter we need to address. We need to build a *weak truth-table* reduction of D to A. When we set up a layer such as L_1, we commit to using only $L_0 \cup L_1$ for computing $D \upharpoonright \gamma(a_1)$. If later we abandon that version of L_1, we cannot lift the use to compute $D \upharpoonright \gamma(a_1)$ from the new L_1. The only thing that we can do is require that $D \upharpoonright \gamma(a_1)$ (for the old value of $\gamma(a_1)$) be computable by $A \upharpoonright L_0$, because we enumerated the old L_1 in its entirety into A. How can we ensure that $D \upharpoonright \gamma(a_1)$ doesn't change more than $h(a_0)$ many times? This we do by updating the value of $\gamma(a_0)$ to be larger than the old $\gamma(a_1)$, and we can do this exactly because $D \upharpoonright \gamma(a_0)$ has just changed. We note that this is necessary: without this feature, we wouldn't need to ever change $\gamma(a_n)$, and then we'd have $\Gamma(D) \leqslant_{\mathrm{wtt}} D$ and we'd be able to prove the lemma for all c.e. sets D, which is of course false.

Construction. Let $\langle g_e, h_e \rangle$ be an effective enumeration of all pairs of partial ω-c.e. approximations: so $g_e(x,s)$ is a total computable function, h_e is a partial computable function, and for each x, the number of s such that $g_e(x,s) \neq g_e(x,s+1)$ is bounded by $h_e(x)$ (and is 0 if $h_e(x) \uparrow$). We are given a c.e. set D; we define a Turing functional Γ. For each e we also define a computable linear ordering $<_e$ and other auxiliary sets. As discussed, apart from building Γ, the construction for each e is completely independent, so we describe the instructions for e and drop the subscript e everywhere. So fix e, let $g_s(x) = g_e(x,s)$ and $h(x) = h_e(x)$. We build the linear ordering $<_L = <_e$ and approximate finite sets $L_n = L_{e,n}$ and enumerate a set $A = A_e$.

The *main module* for e is:

(1) Let n be the least such that L_n is currently undefined. Let a_n be the least number not picked as an unpermitting number by any e' (including e itself). Define $\Gamma(D, a_n)$ be the stage number s with use $\gamma(a_n) = s$. Wait for $h(a_n)$ to converge and for $g_t(a_n) = s$. If, while waiting, $D \upharpoonright s$ changes, redefine $\Gamma(D, a_n) = s$ with use s (note, this is the original s, not the new stage number).

(2) Let L_n be a collection of $h(a_n)+2$ many points not currently in the domain of $<_L$. Extend the definition of $<_L$ to linearly order L_n (say to agree with the natural ordering on $\mathbb{N}$) and place L_n between $\max_{<_L} A$ and its successor in $<_L$. Enumerate $\min L_n$ into A. Return to (1).

There is one *interrupt procedure* for e. The *n-interrupt condition* at a stage t is:

> $s < t$ is the last stage before t at which L_n is defined (step (2) of the main module) or a single number is enumerated into $A \cap L_n$ (step (2) of an n-interrupt), L_n is not cancelled between stages s and t, and
>
> $$D \upharpoonright \gamma(a_n)[s] \neq D \upharpoonright \gamma(a_n)[t].$$

The instructions for the n-interrupt are:

(1) For all $n' > n$, enumerate $L_{n'}$ into A and declare $L_{n'}$ undefined. Also, set $\Gamma(D, a_{n'}) = 0$ with use 0 and declare $a_{n'}$ undefined. Redefine $\Gamma(D, a_n) = t$

with new use $\gamma(a_n) = t$. Wait for $g_r(a_n) = t$. In the meanwhile, if $D \upharpoonright t$ changes, redefine $\Gamma(D, a_n) = t$ with use t.

(2) Enumerate $\min(L_n \setminus A)$ into A. Return to the main module.

Verification. First, we note that the instructions are consistent. Namely, that step (2) of the interrupt process can be always carried out.

Lemma 2.3. *For every e, at every stage s, for every n such that $L_n = L_{n,e}$ is defined, $L_n \not\subseteq A$ (where $A = A_e$).*

Proof. As long as L_n is not cancelled, the size of $A \cap L_n$ is 1+the number of times (since the current version of L_n was set up) an n-interrupt arrived at step (2). For each such step we have a new guess for the value of $g_s(a_n)$ and so

$$|A \cap L_n| \leqslant 1 + h(a_n) < |L_n|. \qquad \square$$

Lemma 2.4. *$\Gamma(D)$ is total.*

Proof. Let $a < \omega$. By induction assume that every $b < a$ is in the domain of $\Gamma(D)$. Then there is some e, some stage s and some n such that e picks a as a_n at stage s. If e ever cancels L_n then we define $\Gamma(D, a)$ with use 0 and so $\Gamma(D, n)\downarrow$. If e ever gets stuck at step (1) of the main module (when setting up L_n) or at step (1) of an n-interrupt, then we keep defining $\Gamma(D, a) = t$ with use t for some stage t; eventually, $D \upharpoonright t$ stabilizes and so this definition becomes permanent.

Otherwise, we first define $\Gamma(D, a) = s$ with use s when setting up L_n, and at every n-interrupt after that, we redefine $\Gamma(D, a) = t$ with use t for some t. As we've seen before, this happens finitely many times. After each time, if $D \upharpoonright t$ changes, then we call an n-interrupt again. Thus eventually one of these definitions is permanent. $\square$

By our assumption that $\deg_T(D)$ is totally ω-c.e., we can fix an e such that (g_e, h_e) is an ω-c.e. approximation for $\Gamma(D)$. From now, drop all subscripts e. Note that neither the main module for e nor any interrupt for e ever get stuck waiting at step (1).

Lemma 2.5. *$<_L$ is a computable linear ordering of order-type $\omega + \omega^*$, and A is the ω-part of $<_L$.*

Proof. By induction on n we see that for every n, there is some version of L_n which is never cancelled. For suppose that a version of L_{n-1} is set up at stage s and is not later cancelled. Then there is a stage $t \geqslant s$ at which the last element of $A \cap L_{n-1}$ gets enumerated into A. Immediately after that stage, a new L_n is set up. This L_n will never be cancelled, because an $n-1$ interrupt does not get stuck at step (1) and so necessarily enumerates another point into $A \cap L_{n-1}$.

For every version of L_n, including the final one, $A \cap L_n$ is non-empty, and so A is infinite. On the other hand, if a point x is enumerated into A at stage s, then no points are later placed in $<_L$ to the left of x, so the initial segment of $<_L$ determined by x is finite. Thus the order type of A is ω. It is also easy to see that the instructions ensure that A is an initial segment of $<_L$.

For every permanent version of any L_n, $\max L_n$ never gets enumerated into A and so the complement of A is infinite. In fact, the complement of A is the union of $L_n \setminus A$ for all *permanent* L_n's. For each point in that complement, after it is added to $<_L$ (when the permanent L_n is set up), no new points to the right are ever added. Thus the order-type of the complement of A is ω^*. $\square$

Lemma 2.6. $D \leqslant_{\mathrm{wtt}} A$.

Proof. Suppose that at stage s, a (not necessarily permanent) L_n is set up and a_n is chosen. Then $s = \gamma_s(a_n)$. We show how to compute $D \upharpoonright s$ from $A \upharpoonright \bigcup_{m \leqslant n} L_m$, where the L_m's are their versions at stage s.

Let $m \leqslant n$ be the greatest such that after stage s, L_m is never cancelled (such an m exists, because L_0 is never cancelled, and can be calculated from $A \upharpoonright \bigcup_{m \leqslant n} L_m$ by looking at the last element of each L_m). Let t be the last stage at which we go through step (2) of the main module (setting up L_m) or an m-interrupt, and let $r < t$ be the stage at which the corresponding step (1) is started. $A \upharpoonright L_m$ can calculate these stages. The main point is that $r = \gamma_r(a_m) = \gamma(a_m)$ is not smaller than s. For if L_n is never cancelled, then $m = n$ and then $r \geqslant s$; or $m < n$, in which case L_n is cancelled at stage r and so $r > s$.

Since there is no m-interrupt after stage t, we know that $D \upharpoonright r = D_t \upharpoonright r$. So $D \upharpoonright s$ doesn't change after stage t. □

2.3. **What if B is not c.e.?** We now explain how to change the proof above in the case that the given set $B \leqslant_T \mathbf{d}$ is not c.e., and thus indicate how to prove the full version of Proposition 1.5. Let $D \in \mathbf{d}$ be c.e., and let Θ be a Turing functional such that $\Theta(D) = B$. The construction now is almost identical to the construction above, building $\Gamma(D)$, etc., except that we tie the use of Θ computations to that of Γ computations. Namely, we require that if a_n is chosen as an unpermitting number for some guess (g_e, h_e), then at all stages s, the use $\gamma_s(a_n)$ is greater than the use of computing $\Theta(D) \upharpoonright s$ at that stage. [Because $\Theta(D)$ is total, by speeding up the enumeration of D, we may assume that for all s, $\Theta(D) \upharpoonright s\!\downarrow$ at stage s.] So instead of computing $D \upharpoonright s$, $A \upharpoonright \bigcup_{m \leqslant n} L_m$ is responsible for computing $\Theta(D) \upharpoonright s$. Before confirmation, if D changes below this $\gamma(a_n)$, then we may need to update $\gamma(a_n)$ because of an increase in the use of $\theta(s)$. But this doesn't spin out of control because $\Theta(D)$ is total, so $\theta(s)$ reaches a limit.

When D changes after confirmation, and an interrupt is called, then we need to shift responsibilities as above: some smaller m is handed with the burden of computing $\Theta(D) \upharpoonright s$. The general geometry of the construction is not changed much.

Construction. Again we are given $\langle g_e, h_e \rangle$ and D; and this time, also a Turing functional Θ such that $\Theta(D)$ is total. We define a Turing functional Γ, and computable linear orderings $<_e$, and other auxiliary sets. Apart from the use markers $\gamma_s(a)$, we also need to define placement markers $\delta(n) = \delta_e(n)$; the idea is that $A \upharpoonright \bigcup_{m \leqslant n} L_m$ computes $\Theta(D) \upharpoonright \delta(n)$.

The *main module* for e is:

(1) Let n be the least such that L_n is currently undefined. Let a_n be the least number not picked as an unpermitting number by any e' (including e itself). Define $\delta(n) = s$ (the current stage number), and define $\Gamma(D, a_n) = s$ with use $\theta_s(\delta(n))$. Wait for $g_t(a_n) = s$.

If, while waiting, $D \upharpoonright \gamma_s(a_n)$ changes, redefine $\Gamma(D, a_n) = s$ with use $\theta_t(\delta(n))$ (for the current stage t).

(2) Let L_n be a collection of $h(a_n)+2$ many points not currently in the domain of $<_L$. Extend the definition of $<_L$ to linearly order L_n (say to agree with the natural ordering on $\mathbb{N}$) and place L_n between $\max_{<_L} A$ and its successor in $<_L$. Enumerate $\min L_n$ into A. Return to (1).

The *n-interrupt condition* for e at a stage t is:

> $s < t$ is the last stage before t at which L_n is defined (step (2) of the main module) or a single number is enumerated into $A \cap L_n$ (step (2) of an n-interrupt), L_n is not cancelled between stages s and t, and
>
> $$D \upharpoonright \gamma(a_n)[s] \neq D \upharpoonright \gamma(a_n)[t].$$

The instructions for the n-interrupt are:

(1) For all $n' > n$, enumerate $L_{n'}$ into A and declare $L_{n'}$ undefined. Also, set $\Gamma(D, a_{n'}) = 0$ with use 0 and declare $a_{n'}$ and $\delta(n')$ undefined.
    Redefine $\delta(n) = t$ and $\Gamma(D, a_n) = t$ with new use $\gamma(a_n) = \theta_t(\delta(n))$. Wait for $g_r(a_n) = t$. In the meanwhile, if $D \upharpoonright t$ changes, redefine $\Gamma(D, a_n) = t$ with use the current $\theta(\delta(n))$.
(2) Enumerate $\min(L_n \setminus A)$ into A. Return to the main module.

Verification. Again, the instructions are consistent:

Lemma 2.7. *For every e, at every stage s, for every n such that $L_n = L_{n,e}$ is defined, $L_n \not\subseteq A$ (where $A = A_e$).*

The proof is identical to that of Lemma 2.3.

Lemma 2.8. *$\Gamma(D)$ is total.*

Proof. Identical to the proof of Lemma 2.4, except that if the guess is wrong and we get stuck, after defining $\delta(n) = s$ and $\Gamma(D, a_n) = s$ with use $\theta(\delta(n))$, waiting for $g_t(a_n) = s$, then $\delta(n)$ (for this e) is not redefined after stage s, and so $\theta(\delta(n))$ eventually reaches a limit u; after that, we keep redefining $\Gamma(D, a_n) = s$ with use u, and this process halts once $D \upharpoonright u$ stabilizes. □

Again we pick a correct guess (g, h) and get that the ordering built for the correct guess is of type $\omega + \omega^*$, with A being the ω-part.

Lemma 2.9. *$\Theta(D) \leqslant_{\mathrm{wtt}} A$.*

Proof. Suppose that at stage s, a (not necessarily permanent) L_n is set up and a_n is chosen. Then $s = \delta_s(n)$. We show how to compute $\Theta(D) \upharpoonright s$ from $A \upharpoonright \bigcup_{m \leqslant n} L_m$, where the L_m's are their versions at stage s.

Let $m \leqslant n$ be the greatest such that after stage s, L_m is never cancelled. Let t be the last stage at which we go through step (2) of the main module (setting up L_m) or an m-interrupt, and let $r < t$ be the stage at which the corresponding step (1) is started. Then $r = \delta(n)$ (as is set at stage r and is not altered later) is not smaller than s; the proof is the same as in Lemma 2.6 for $\gamma(a_n)$.

Let $u = \theta_t(r)$. Since there is no m-interrupt after stage t, we know that $D \upharpoonright u = D_t \upharpoonright u$. So $\Theta(D) \upharpoonright s$ doesn't change after stage t. □

3. Hypersimplicity and ranked sets in the wtt degrees

3.1. **A hypersimple set which is not wtt-reducible to a ranked set.** Here we prove the following fact:

Proposition 3.1. *There is a hypersimple set which is not wtt-reducible to any ranked set.*

To prove this proposition, we would like to closely follow the construction (from [8]) of a c.e. set which is not wtt-reducible to any ranked set. If the set we enumerate is A, the diagonalisation requirements to meet are:

R_e**:** If $X \in \mathcal{P}_e$ and $\Phi_e^X = A$ then $\mathcal{P}_e$ is uncountable.

Here $(\mathcal{P}_e, \Phi_e)$ is a list of all pairs of Π^0_1 classes and wtt-functionals (as usual, φ_e denotes a computable bound on the use of Φ_e). The general plan from [8] is to devote a column of ω to each requirement, and then diagonalise on finitely many elements of the column against a finite clopen covering of $\mathcal{P}_e$ whose elements are small enough (that is, the strings are longer than the use of Φ_e); a combinatorial lemma says that if $\mathcal{P}_e$ is countable then such a covering must show up eventually.

In our case, we also want to make A hypersimple. So we want to meet the following requirements:

P_e**:** If g_e is total and $\langle D_{g_e(n)} \rangle_{n<\omega}$ consists of pairwise disjoint sets, then there is some n such that $D_{g_e(n)} \subset A$.

Here (g_e) is an enumeration of all partial computable functions, and D_x is an enumeration of all finite sets of natural numbers.

The hypersimplicity requirements do not interact well with the general plan; a requirement R_e cannot keep a whole column to itself and expect even lower priority requirements to never enumerate elements of the column into A. It can, however, using finitely much restraint, ensure that infinitely much of ω is available for diagonalisation. Of course the part of ω, namely $\omega \setminus A$, which is left for diagonalisation is not computable; this makes the combinatorial lemma useless. Using incompleteness, in particular Arslanov's criterion and DNR functions, we salvage enough of the lemma and push through the construction. Recall that a function g is DNR (diagonally non-recursive) if $g(e) \neq \varphi_e(e)$ for all e such that $\varphi_e(e)$ is defined.

Relativised complexity. We start with the combinatorial lemma. For any function $g\colon \omega \to \omega$, let $\mathcal{Q}_g$ be the collection of sets $X \in 2^\omega$ such that for all n, $C(X \upharpoonright g(n)) \geqslant n$. The class $\mathcal{Q}_g$ is a $\Pi^0_1(g)$-class.

Lemma 3.2. *For all $X \in \mathcal{Q}_g$, $X \oplus g$ computes a DNR function.*

Proof. This is implicit in [25, Proposition 6]; the function $n \mapsto X \upharpoonright g(n)$ differs from $\varphi_n(n)$ for almost all n because the plain complexity of $\varphi_n(n)$ is at most $C(n)$ which is smaller than n. □

The following is analogous to [8, Theorem 4.7].

Lemma 3.3. *If $\mathcal{P}$ is a countable Π^0_1-class and g does not compute a DNR function, then $\mathcal{P} \cap \mathcal{Q}_g = \emptyset$.*

Proof. The point is that no $X \in \mathcal{Q}_g$ is computable in g, and so any $\Pi^0_1(g)$-subclass of $\mathcal{Q}_g$, such as $\mathcal{P} \cap \mathcal{Q}_g$, must be either empty or perfect. □

Definition 3.4. Let $A \subseteq \omega$ be a set and let $f\colon \omega \to \omega$ be a function. Let $\mathcal{P}$ be a Π^0_1-class. An *A, f-covering* of $\mathcal{P}$ is a map $x \mapsto \sigma_x$ from some finite subset D of A such that for all $x \in D$, $|\sigma_x| > f(x)$, and such that

$$\mathcal{P} \subset \bigcup_{x \in D} [\sigma_x].$$

(Compare with [8, Definition 5.1].)

Lemma 3.5. *Suppose that f is a computable function and that A is an infinite set which does not compute a DNR function. Then every countable Π^0_1 class has an A, f-covering.*

Proof. We follow the proof of [8, Theorem 5.2]. We split A into disjoint sets $\langle I_k \rangle$ with each I_k of size 2^k (and for simplicity, $\max I_k < \min I_{k+1}$); we let $g(k) = \max_{x \in I_k} f(x)$. Inductively, for each $x \in A$ we pick a string σ_x of length $g(k)$ (if $x \in I_k$) of minimal plain complexity (which has not been chosen so far). Again, if $\sigma \in 2^{g(k)}$ and $C(\sigma) < k$ then $\sigma = \sigma_x$ for some $x \in I_k$, so $\mathcal{Q}_g$ is disjoint from $\bigcup_{x \in A} [\sigma_x]$.

Now $g \leqslant_T A$ so g does not compute a DNR function. So if $\mathcal{P}$ is countable then $\mathcal{P} \cap \mathcal{Q}_g = \emptyset$ and so $\bigcup_{x \in A} [\sigma_x]$ covers $\mathcal{P}$; so by compactness, some finite sub-cover covers $\mathcal{P}$ and is an A, f-covering. □

Remark 3.6. We could have used the notion of h-complex sets, which is equivalent to being in $\mathcal{Q}_g$ (for some g of the same Turing degree as h) if h has K-trivial degree. We could have proceeded with the proof of Proposition 3.1 because the construction of a hypersimple set is consistent with making it K-trivial, that is, we could obey the standard cost function. Instead, we use the blunter instruments of lowness (and Arslanov's completeness criterion).

Construction. We enumerate a set A and attempt to meet the requirements R_e and P_e above. In addition, we make A incomplete by ensuring that it is low; let N_e be the e^{th} lowness requirement, i.e. the requirement which tries to ensure that the halting or not of $\Phi^A_e(e)$ can be decided given the (unrelativised) halting problem $\emptyset'$. The strategy for N_e is to build a computable approximation with final value 1 if $\Phi^A_e(e)$ is defined, and 0 otherwise. Intuitively, N_e will be trying to preserve $\Phi^A_e(e)$ if such a computation is discovered.

We impose finitary restraint between requirements using initialisation. Generally, whenever a requirement receives attention, we initialise all weaker requirements. We also ensure that A is co-infinite by imposing restraint when certain markers move: at stage s, let $\langle a_{k,s} \rangle$ be an increasing enumeration of $\omega \setminus A_s$; if $a_{k,s} \neq a_{k,s+1}$ then all requirements Q_k (for $Q \in \{R, P, N\}$) are initialised. For any requirement Q_e, we let $r_s(Q_e)$ be the last stage before s at which Q_e was initialised; this is the restraint imposed on Q_e at stage s.

We now describe the instructions for all requirements. We describe when a requirement requires attention, and what it does if it receives it. As usual, at every stage, the strongest requirement which requires attention, receives it.

At stage s:

- N_e**:** A requirement N_e requires attention if it has not received attention since stage $r_s(N_e)$, and $A'(e) \downarrow$ [s]. When it receives attention, it does nothing (except for initialising all weaker requirements).
- P_e**:** A requirement P_e requires attention if there is no $n \in \operatorname{dom} g_{e,s}$ such that $D_{g_e(n)} \subset A_s$, and there is some $n \in \operatorname{dom} g_{e,s}$ such that $\min D_{g_e(n)} > r_s(P_e)$. If it receives attention, then one such n is chosen and the contents of $D_{g_e(n)}$ are enumerated into A.
- R_e**:** A requirement R_e first searches for a suitable covering. So if, at stage s, R_e has no covering, then it searches for a covering $\langle \sigma_x \rangle_{x \in D}$ which is currently suitable:
  - The covering appears by stage s: for all $x \in D$, $x < s$ and $|\sigma_x| < s$.

- D is available for diagonalisation: $D \cap A_s = \emptyset$;
- D is unrestrained: $\min D > r_s(R_e)$.
- This is an $\omega \setminus A_s, \varphi_e$-covering: $D \subset \operatorname{dom} \varphi_{e,s}$ and for all $x \in D$, $|\sigma_x| > \varphi_e(x)$.
- It currently covers $\mathcal{P}_e$: $\mathcal{P}_{e,s} \subseteq \bigcup_{x \in D} [\sigma_x]$.

If such a covering appears, then R_e requires attention. If it receives attention, then it picks the covering to work with (and initialises weaker requirements). Note that if later, R_e is initialised, then the covering is no longer permitted, it is abandoned, and R_e needs to resume the search for a new one.

If a covering $\langle \sigma_x \rangle_{x \in D}$ is already appointed, then R_e requires attention if there is some $x \in D$ such that $\Phi_e^{\sigma_x}(x) \downarrow [s] = 0$ and $x \notin A_s$. If it receives attention, R_e enumerates x into A.

Verification.

Lemma 3.7. *The construction is fair: for every requirement Q_e there is a stage after which it is never initialised; it requires attention only finitely many times. Also, A is co-infinite.*

Proof. The usual proof, by induction on the priority ordering. Suppose that s^* is the last stage at which any requirement which is stronger than a requirement Q_e ever receives attention (and so no such requirements ask for attention after stage s^*) and by s^* all $a_{i,s}$, $i < e$ have reached a limit. Suppose that there is some stage $s > s^*$ such that $a_{e,s} \neq a_{e,s+1}$. By induction hypothesis e is the least number with this property, hence $a_{e,s+1} < s$. Then at stage s all of P_e, R_e and N_e are initialised, so we have $r_t(Q_{e'}) > a_{e,s+1}$ for all $t > s$ and all requirements $Q_{e'}$ which are not stronger than Q_e. Thus $a_{e,s+1}$ (and any smaller number) is never enumerated into A after stage s, so after stage s, Q_e is never initialised.

Now depending on the nature of Q_e, it is easy to check that after it is last initialised, Q_e only receives attention finitely often; in the case of N_e and P_e, it is at most once, and in the case of R_e, it is once for picking a covering $\langle \sigma_x \rangle_{x \in D}$ and then at most once for every $x \in D$. □

Lemma 3.8. *A is low and hypersimple.*

Proof. The requirement P_e is met: if g_e is total and $\langle D_{g_e(n)} \rangle$ are pairwise disjoint then there is some n such that $\min D_{g_e(n)}$ is greater than the last stage at which P_e is initialised; then at some stage, P_e acts.

The requirement N_e is met: if s^* is the last stage at which N_e is initialised, and there is some $s > s^*$ at which $A'_s(e)$ converges, then by initialisation, this computation is preserved. □

Lemma 3.9. *A is not wtt-reducible to any ranked set.*

Proof. We show that every requirement R_e is met. Suppose that $\mathcal{P}_e$ is countable and that φ_e is total. Since A is low, it is incomplete (and has c.e. degree), so by Arslanov's completeness criterion (see, e.g. [34]), A computes no DNR function. Let s^* be the last stage at which R_e is initialised, and let $B = \omega \setminus (A \cup \{n \| \ n \leq s^*\})$, which is of course Turing equivalent to A.

So there is some B, φ_e-covering $\langle \sigma_x \rangle_{x \in D}$ of $\mathcal{P}$. Such a covering will be discovered by some stage $s > s^*$ (and will be permitted), and so we know that R_e picks some

covering at some stage $s > s^*$. Later, R_e may diagonalise whenever it needs to, so it is met. □

3.2. **A ranked set which is not wtt-reducible to a hypersimple set.** Here we prove:

Proposition 3.10. *There is a c.e. ranked set (indeed of rank 1) which is not wtt-reducible to any hypersimple set.*

We call the set we enumerate A. To make A ranked, we make it an initial segment of a scattered, computable linear ordering; it will be the ω-part of an ordering of type $\omega + (\omega^2)^*$. This is the "shortest" possible example, because Lemma 1.3 implies that if $\gamma < \omega^2$ and A is a c.e. and non-computable initial segment of a computable ordering of type $\omega + \gamma^*$ then A is computably isomorphic to a hypersimple set.

The strategy of making A not wtt-reducible to any hypersimple set is similar to that from Section 2.1. Let us remind the reader of the requirements we need to satisfy:

R_e**:** If $\Phi_e^{W_e} = A$ then W_e is not hypersimple.

(Here $\langle W_e, \Phi_e \rangle$ is a list of all pairs consisting of a c.e. set and a weak truth-table functional with use function φ_e.) We combine the strategy from Section 2.1 with a priority argument to get the present construction. Again, a requirement R_e will choose finite sets of followers $X_0^e, X_1^e, X_2^e, \dots$ such that $|X_{n+1}^e| > u_n^e$, where for all $x \in X_n^e$ we have $\varphi_e(x) < u_n^e$. The sets X_n^e are placed in the e^{th} copy of ω^* in the linear ordering we are building, until we decide to attack with some X_n^e; we then move the weaker requirements to build between the elements of X_n^e.

Construction. Each requirement R_e defines an increasing sequence $\langle u_n^e \rangle$ of natural numbers and finite sets of followers $\langle X_n^e \rangle$. When R_e is initialised, we abandon its current sequences and start rebuilding from $n = 0$. A requirement R_e can be in *waiting* mode or in *attack* mode. (If R_e is initialised, then it is returned to waiting mode.)

At stage s, if R_e is in waiting mode, then it requires attention if one of the following holds:

(1) no X_n^e is currently defined.
(2) $X_0^e, \dots, X_n^e$ are currently defined and for all $x \in X_n^e$, $\varphi_e(x)\downarrow$.
(3) There is some n such that u_n^e is defined and $[u_{n-1}^e, u_n^e) \subset W_e$.

If R_e is in attack mode (with X_n^e), then it requires attention if $\Phi_e^{W_e}$ currently agrees with $A \upharpoonright \max X_n^e + 1$.

We let the strongest requirement which requires attention receive it. Suppose that R_e receives attention. If R_e is in waiting mode, we act according to the cases above.

(1) Let X_0^e be some finite set of fresh numbers. In the linear ordering $<^*$ that we are building, place the elements of X_0^e immediately to the right of A_s.
(2) Let $u_n = 1 + \max\{\varphi_e(x) : x \in X_n^e\}$. Let X_{n+1}^e consist of $u_n + 1$ many fresh numbers. In the linear ordering $<^*$ that we are building, place the elements of X_{n+1}^e immediately to the left of the elements of X_n^e.
(3) Move R_e to attack mode with X_n^e.

If R_e is attacking with X_n^e, we enumerate the $<^*$-leftmost element of $X_n^e \setminus A_s$ into A and initialise all weaker requirements. Maintain A as an initial segment of

$<^*$ by enumerating into A any numbers which lie to the left of the number just enumerated into A (these are the elements of $X^{e'}_m$ for $e' > e$).

Verification. It is clear that $<^*$ that we built is a computable linear ordering, since we placed any newly chosen followers into $<^*$ immediately. It is also clear that the order-type of A in $<^*$ is ω, because once an element is enumerated into A, no new element is ever placed to its left. Further, it is also clear that the construction is fair, in that every requirement is initialised only finitely many times; because a requirement R_e only initialises weaker requirements if it is in attack mode, and then it acts only $|X^e_n|$ many times (if it is not later initialised).

Lemma 3.11. *The order-type of $<^*$ is $\omega + (\omega^2)^*$.*

Proof. Every requirement R_e has three possible eventual outcomes (after it is last initialised):

- R_e defines only finitely many sets of followers X^e_n (because φ_e is not total), and does not attack. In that case, R_e adds only finitely many points to $<^*$ and does not change its order-type.
- R_e defines infinitely many X^e_n but never attacks. In that case, it adds to $<^*$ an interval of order-type ω^*. This happens, for example, whenever φ_e is total but W_e is empty.
- R_e eventually attacks with some X^e_n. In this case again R_e adds only finitely many points to $<^*$.

Whenever a requirement R_e attacks and initialises weaker requirements, finitely many points are added to A but the picture for stronger portions of $<^*$ doesn't change. □

Lemma 3.12. *A is not wtt-reducible to any hypersimple set.*

Proof. Suppose for contradiction that W_e is hypersimple and that $\Phi_e^{W_e} = A$. Let s_0 be the last stage at which R_e is initialised. Since φ_e is total, R_e can always define new sets of followers X^e_n, and since W_e is hypersimple, we eventually discover some n such that $[u_{n-1}, u_n) \subset W_e$; so eventually, at some stage $s_1 > s_0$, R_e attacks with some X^e_n. We are then later allowed to change $A \upharpoonright \max X^e_n + 1$ more times than the possible future configurations for $W_e \upharpoonright u_n$ which leads to the desired contradiction. □

4. Computable Lipschitz reductions and array computable degrees

4.1. **Multiple permitting I: common upper bounds.** Here we prove Proposition 1.8: If $\mathbf{d}$ is c.e. and array non-computable, then there are left-c.e. reals α_0 and α_1 in $\mathbf{d}$ which do not have a common upper bound in the ibT-degrees of left-c.e. reals. This is obtained by adding coding and multiple permitting to the construction of two left-c.e. reals with no common upper bound in the cl-degrees from [35, 5], which we now rephrase for our purposes.

The main idea of the Yu-Ding construction is that if β is a left-c.e. real which ibT-computes both α_0 and α_1 then alternatingly adding little bits to α_0 and α_1 (drip-feeding them) is sufficient to drive β to be too large. Furthermore, we can computably reserve, for each requirement, an interval of its own, which is sufficient for its purposes. Here are the details.

Suppose that Γ_0 and Γ_1 are ibT reductions and that β is a left-c.e. real. For all such triples, we need to meet the requirement

$R_{\Gamma_0,\Gamma_1,\beta}$**:** Either $\Gamma_0^\beta \neq \alpha_0$ or $\Gamma_1^\beta \neq \alpha_1$.

We view left-c.e. reals as both infinite binary sequences and as the corresponding elements in the Euclidean interval $[0,1]$ (via binary expansion). Addition means addition in the real field $\mathbb{R}$.

To meet requirement $R = R_{\Gamma_0,\Gamma_1,\beta}$, we describe a family of modules, each indexed by an interval of natural numbers. Let $a \leqslant b < \omega$. A stage s is *expansionary* for the $[a,b)$-module (for R) if for both $i = 0,1$ we have $\Gamma_i^{\beta_s} \supset \alpha_{i,s} \upharpoonright b$. The instructions for the module are:

Repeat the following $2^{b-a} - 1$ times:
(1) At the next expansionary stage, add 2^{-b} to α_0.
(2) At the next expansionary stage, add 2^{-b} to α_1.
At the end, wait for the next expansionary stage and then return.

To meet the requirement R, we run the $[k, k+2^k)$-module for R for some k. We will shortly argue that this module cannot return, and so it must get stuck waiting for some expansionary stage, and so R is met. It is easy to see that assuming that we start with $\alpha_i \upharpoonright [a,b) = 0^{b-a}$, the $[a,b)$-module for any requirement only makes changes in $\alpha_i \upharpoonright [a,b)$ (for both $i = 0,1$) and so if for distinct requirements we run modules on disjoint intervals, there is no interaction between the requirements and so we can meet them all.

The verification relies on the following lemma. For this lemma, we also think of a finite binary string as a natural number (via binary expansion[6]).

Lemma 4.1. *Suppose that $\alpha_{0,t_0} \upharpoonright [a,b) = \alpha_{1,t_0} \upharpoonright [a,b) = 0^{b-a}$. Suppose that at stage t_0, an $[a,b)$-module for R begins and returns at a stage t_1. Then*

$$\beta_{t_1} \upharpoonright a - \beta_{t_0} \upharpoonright a \ \geqslant\ b - a.$$

Proof. By induction on $b-a$. If $b = a$ there is nothing to prove. Assume the lemma holds for n, and let $a < b < \omega$ such that $b - a = n + 1$. The key observation is that the $[a,b)$-module consists of three parts:

- Running the $[a+1,b)$-module;
- Running one iteration of adding 2^{-b} to α_0 and then α_1 (and waiting for expansionary stages).
- Running the $[a+1,b)$-module again.

We also note that both times that we run the $[a+1,b)$-module, we start with $\alpha_i \upharpoonright [a+1,b) = 0^n$: at the first time, by assumption of the lemma, and by the second, because when the first $[a+1,b)$-module halts we have $\alpha_i \upharpoonright [a,b) = 01^n$ for both i; we then add 2^{-b} which makes $\alpha_i \upharpoonright [a,b) = 10^n$.

Let s_0 be the stage at which the first $[a+1,b)$-module returns, and s_1 be the stage at which the second one begins. By induction, therefore, we have

$$\beta_{s_0} \upharpoonright a+1 - \beta_{t_0} \upharpoonright a+1 \ \geqslant\ n$$

and

$$\beta_{t_1} \upharpoonright a+1 - \beta_{s_1} \upharpoonright a+1 \ \geqslant\ n.$$

Between s_0 and s_1 we have a change in $\alpha_0(a)$, which forces a change in $\beta \upharpoonright a+1$ by the next expansionary stage, and then a change in $\alpha_1(n)$, which forces another

[6]with the understanding that a prefix of 0s may occur in the binary digits of a number; so the correspondence is not 1-1.

change in $\beta \upharpoonright a+1$. Each of these changes adds at least 1 to $\beta \upharpoonright a+1$. Thus in total,

$$\beta_{t_1} \upharpoonright a+1 - \beta_{t_0} \upharpoonright a+1 \ \geqslant \ 2(n+1)$$

which implies that

$$\beta_{t_1} \upharpoonright a - \beta_{t_0} \upharpoonright a \ \geqslant \ n+1$$

as required. □

This concludes the verification: the $[k, k+2^k)$-module can never return because that would force $\beta \upharpoonright k \geqslant 2^k$ which is impossible.

Now to add multiple permitting, we use the original definition of array computability from [19]: A c.e. degree $\mathbf{d}$ is array non-computable iff for all (or some) computable partitions $\langle F_n \rangle$ of ω of increasing size, there is a c.e. set $D \in \mathbf{d}$ such that for all c.e. sets W there are infinitely many n such that $W \upharpoonright F_n = D \upharpoonright F_n$. Later, we will calculate the desired size of F_n. Assuming this has been done, fix some $D \in \mathbf{d}$ which satisfies the definition for this $\langle F_n \rangle$. To get sufficiently many permissions, a requirement $R = R_{\Gamma_0,\Gamma_1,\beta}$ as above enumerates an auxiliary c.e. set W_R and ties its actions to permissions from F_n.

Again, a module for requirement R will operate on some interval $[a, b)$; the notion of an expansionary stage for a module for R on an interval $[a, b)$ is defined as before. For such a module, we will assign some n so that $|F_n| > 2(2^{b-a} - 1)$. Note that we choose *distinct* n's for each module for R. To *request permission*, the module picks some $x \in F_n$ which is not yet in W_R and enumerates it into W_R. Permission is *received* when at a later stage, x enters D. The standard modus operandi for multiple permitting is used: if some $x \in F_n$ enters D before it is enumerated into W_R, then F_n can be made incorrect for permitting by withholding x from ever entering W_R, so we assume this never happens.

The new module follows the following instructions:

> Repeat the following $2^{b-a} - 1$ times:
> (1) Wait for an expansionary stage; then request permission. When permission is received, add 2^{-b} to α_0.
> (2) Wait for another expansionary stage; then request permission. When permission is received, add 2^{-b} to α_1.
> At the end, wait for the next expansionary stage and then return.

Note that the choice of n ensures that we can always request new permission, as the module above requests at most $2(2^{b-a} - 1)$ permissions and so we never get $F_n \subseteq W_R$.

The overall construction is as expected. For every R we now pick an infinite set of intervals of the form $[k, k+2^k)$ and run modules on each interval separately (all of these modules together enumerate W_R though). We ensure that these intervals are pairwise disjoint and further that the intervals used by different requirements are also pairwise disjoint. Also, to code in D, we fix a computable set C which is disjoint from every interval used by any requirement. Let c_n be the n^{th} element of C; we declare that for both $i = 0, 1$, $\alpha_i(c_n) = 1$ iff $n \in D$. Since D is a c.e. set and all modules only change α_i on their intervals, we still see that both α_i are left-c.e. reals.

To conclude the construction, we need to specify the required size of F_n so that we can assign, for every requirement R, almost every n as a permitting number

for some module working for R. So after we specify the coding location set C and assign the intervals for every requirement, we define $|F_n|$ to be large enough so that for every $m \leqslant n$, for each of the first m requirements, if $[a, b)$ is the n^{th} interval assigned to R, then $|F_n| > 2(2^{b-a} - 1)$, so F_n can be assigned as a permitting set for the $[a, b)$-module for R.

For the verification, we note that Lemma 4.1 holds for the new construction, with the same proof. Thus, for any requirement R, no $[k, k + 2^k)$-module for R ever returns. The standard permitting argument now shows that it cannot be that every module for some requirement R gets stuck waiting for permission; since for almost all n, F_n is assigned as a permitting set for some module for R, there is some such n such that $W_R \upharpoonright F_n = D \upharpoonright F_n$ and so the module for which F_n is assigned is never stuck waiting for permission. As a result, it must get stuck waiting for an expansionary stage, and so the requirement R is met.

It is clear by coding that $D \leqslant_T \alpha_0, \alpha_1$. The next lemma concludes the verification.

Lemma 4.2. $\alpha_0, \alpha_1 \leqslant_T D$.

Proof. Fix $i < 2$. Let $x < \omega$. To decide $\alpha_i(x)$ with oracle D, it depends in which part of ω x lies. If $x \in C$ then we can easily calculate the n such that $x = c_n$ and then consult D for the value of $\alpha_i(x)$. Otherwise, x belongs to some interval $[a, b)$ on which a module for some requirement R is working. This requirement is assigned some permitting set F_n. Wait for a stage s at which $D_s \upharpoonright F_n = D \upharpoonright F_n$. Then after stage s, $\alpha_i \upharpoonright [a, b)$ is fixed and so $\alpha_{i,s}(x) = \alpha_i(x)$. □

4.2. **Failure of multiple permitting I: common upper bounds.** Here we prove Proposition 1.9: if α_0 and α_1 are left-c.e. reals which have array computable (Turing) degrees, then α_0 and α_1 have a common upper bound in the cl-degrees. The idea is that we can approximate α_0 and α_1 with the mind-changes bound growing as slowly as we like. This follows from the characterisation of array computability as uniform total ω-c.e.-ness (from [19]): A c.e. degree $\mathbf{d}$ is array computable iff for every computable order h, every $f \leqslant_T \mathbf{d}$ has an h-c.e. approximation. We can then use this slow bound to build a left-c.e. real β which changes somewhere on its first n digits whenever either α_0 or α_1 do so.

Again we think of these reals as elements of the interval $[0, 1]$. A request that $\beta \upharpoonright n$ change is ensured by adding a quantity of 2^{-n} to β. If the number of requests for a $\beta \upharpoonright n$ change does not exceed a number $g(n)$, and

$$\sum_{n \geqslant 1} g(n)2^{-n} < 1, \tag{4.1}$$

then we can construct a left-c.e. real that changes appropriately whenever we ask it to. In our case, a request to change $\beta \upharpoonright n$ is made whenever we believe a new value for $\alpha_0 \upharpoonright n$ or $\alpha_1 \upharpoonright n$, so if the number of initial segments of length n which we believe for α_i is at most $g(n)/2$ then the plan will work. So all we need to do is to fix some computable order g which grows sufficiently slowly – say one bounded by a polynomial – so that (4.1) holds, and approximate the functions $n \mapsto \alpha_i \upharpoonright n$ in a $g(n)/2$-c.e. way, which is possible by assumption.

There is not much to add to give a formal construction. Fix a computable order g that satisfies (4.1). For $i = 0, 1$, fix computable functions $f_{i,s}(n) = f_i(s, n)$ such

that for all n, $f_{i,s}(n)$ is a binary string of length n,

$$\#\{s : f_{i,s+1}(n) \neq f_{i,s}(n)\} \leqslant g(n)/2,$$

(if $g(n) < 2$ then $f_{i,s}(n)$ has to be constant in s), and

$$\lim_s f_{i,s}(n) = \alpha_i \upharpoonright n.$$

We define a left-c.e. real β: start with $\beta_0 = 0$; and let

$$\beta_{s+1} = \beta_s + \sum\{2^{-n} : f_{i,s+1}(n) \neq f_{i,s}(n)\}.$$

The fact that g satisfies (4.1) and the restriction on the number of mind-changes for the f_i imply that $\beta = \lim_s \beta_s < 1$, so β is well-defined.

Lemma 4.3. *$\alpha_i \leqslant_{\mathrm{ibT}} \beta$.*

Proof. If s is a stage such that $\beta_s \upharpoonright n = \beta \upharpoonright n$ then we never, after stage s, add a quantity of 2^{-n} to β. This implies that for all $t > s$, $f_{i,t}(n) = f_{i,s}(n)$. Thus $f_{i,t}(n) = \alpha_i \upharpoonright n$. □

Remark 4.4. We didn't use the fact that the α_i are left-c.e. reals; the proof works for any sets which have array computable c.e. degrees because approximations such as the f_i are available for all sets in those degrees.

4.3. **Multiple permitting II: random reals.** In this section we prove Proposition 1.10: if $\mathbf{d}$ is c.e. and array non-computable, then there is some left-c.e. real $\alpha \in \mathbf{d}$ which is not ibT-reducible to any random left-c.e. real.

As before, we only add multiple permitting and coding to a construction of a left-c.e. real which is not ibT-reducible to any random left-c.e. real. We give a simpler construction than the one presented in [7], which we now describe.

We build a left-c.e. real α and try to meet the requirements

$R_{\Gamma,\beta}$**:** If $\Gamma^\beta = \alpha$ then β is not random.

Here Γ ranges over all ibT functionals and β ranges over all left-c.e. reals. To meet R, we will run infinitely many modules, where the n^{th} module enumerates a finite set of strings U_n of measure at most 2^{-n}, such that if $\Gamma^\beta = \alpha$ then $\beta \in U_n$. Thus together, all modules for R enumerate a Martin-Löf test which covers β, and so β is not random.

In the previous construction, we used the leeway we had in the play between two left-c.e. reals to drive a potential common upper bound to be too large. Here we only have one real (α) to play with, and the role of the second real is taken by the element of the Martin-Löf test being enumerated. This is much more limited because the restriction on the size of the enumerated set of strings is much stricter than that on the enumeration of the left-c.e. real.

The modules are built by recursion from smaller and smaller building blocks. Consider, for example, the following module, which only changes α from the a^{th} digit, and makes $\beta \upharpoonright a + 1 \geqslant 2$.[7] As in the Yu-Ding and Barmpalias-Lewis papers, assume (for now) that β is playing its optimal strategy, which is adding the minimal amount which is necessary to match α's movements.

[7] as before, we also think of finite binary strings as natural numbers via their binary expansion, with the understanding that a prefix of 0s may occur in the binary digits of a number.

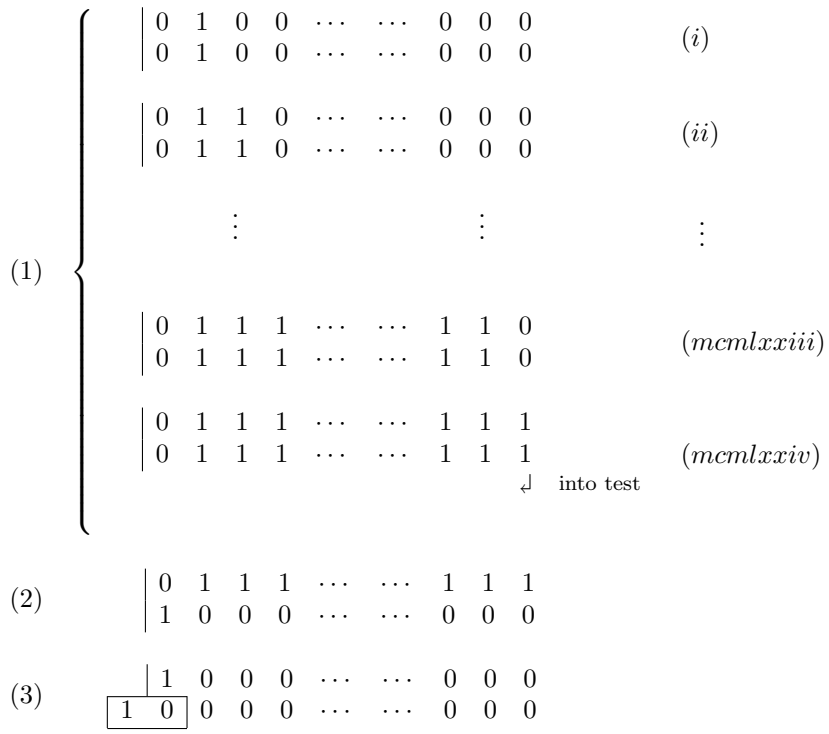

FIGURE 1. The 2-module

(1)
- (i) Set $\alpha(a+1)=1$; wait for $\beta(a+1)=1$.
- (ii) Set $\alpha(a+2)=1$; wait for $\beta(a+2)=1$.
- ⋮
- (mcmlxxiv) Set $\alpha(a+1974)=1$; wait for $\beta(a+1974)=1$.

(2) Enumerate $0^{a+1}1^{1974}$ into the test element; wait for $\beta = 0^a10^\omega$.

(3) Add $2^{-a-1975}$ to α, thus setting $\alpha = 0^a10^\omega$. Wait for $\beta = 0^{a-1}10^\omega$.

The cost, in terms of the measure of strings enumerated into the test element, is $2^{-a-1975}$; as 1975 approaches infinity, we can make the cost as low as we like. Figure 1 shows the state of α and β at the end of every stage of the 2-module. The first digit to the right of the horizontal line is the a^{th}. At the end of the first step, we indicate the initial segment of β which is enumerated into the test element at the second step of the module; and at the end we indicate the amount (2, which is 10 in binary) added to $\beta \upharpoonright a+1$.

Now this module can be iterated to make $\beta \upharpoonright a+1 \geqslant 3$:

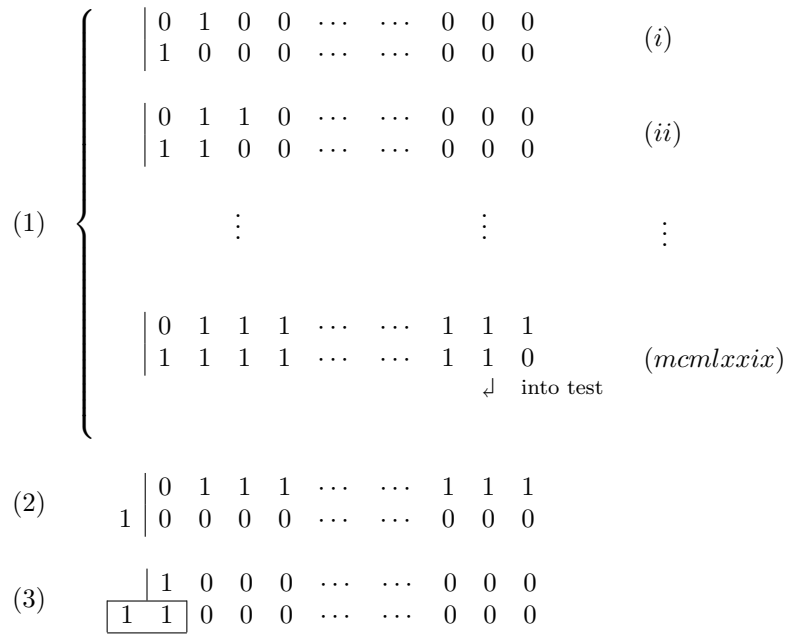

FIGURE 2. The 3-module

(1)
- (i) Run the 2-module from point $a+1$, to get $\alpha(a+1)=1$ and $\beta(a)=1$.
- (ii) Run the 2-module from point $a+2$, to get $\alpha(a+2)=1$ and $\beta(a+1)=1$.
- ⋮
- (mcmlxxx) Run the 2-module from point $a+1979$, to get $\alpha(a+1979)=1$ and $\beta(a+1978)=1$.

(2) Enumerate 0^a1^{1979} into the test element; wait for $\beta = 0^{a-1}10^\omega$.

(3) Set $\alpha = 0^a10^\omega$. Wait for $\beta = 0^{a-1}110^\omega$.

Again the cost can be kept down by making the number 1980 large, and then keeping the cost of every recursive call of the 2-module low as well. Figure 2 follows the 3-module.

For the 4-module (Figure 3), note the growing distance between the last point of change in α and the end of the string of 1's in β which goes into the test. This distance, according to our calculations below, is bounded by n, the level of the module.

We turn to formally describe the modules and investigate their properties, without assuming anything about the behaviour of the opponent.

The module $M_R(a,n,\varepsilon)$ is indexed by: a, the bit of α where it begins acting; n, the level of the module; and ε, the bound on the measure of the c.e. open set U_M which the module enumerates.

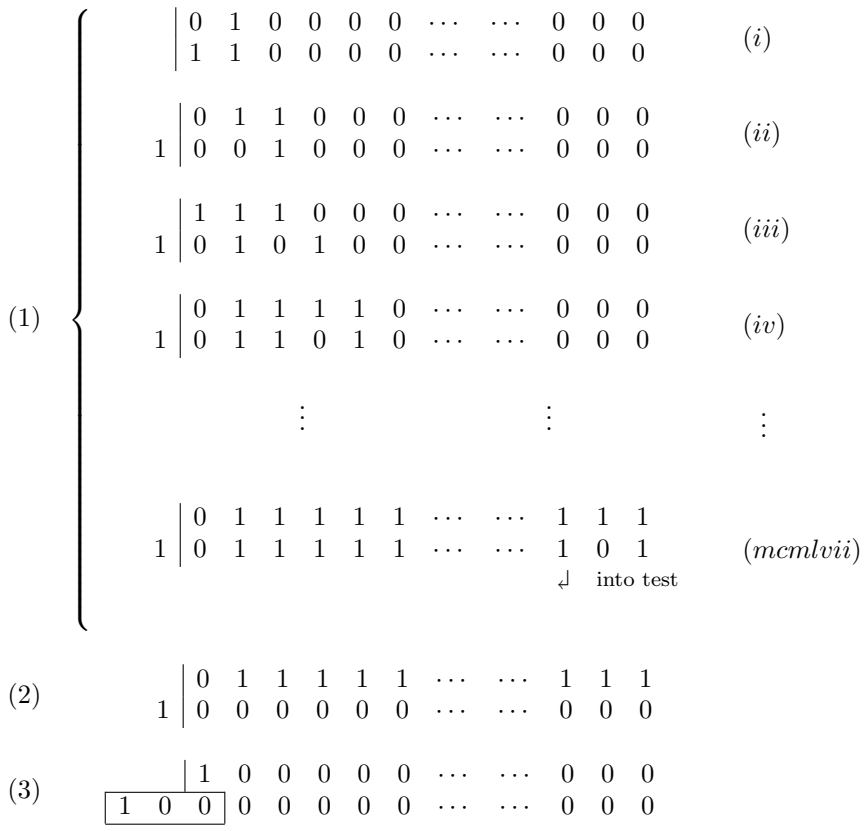

FIGURE 3. The 4-module

By induction on stages, define the notion of an *expansionary stage* for requirement $R = R_{\Gamma,\beta}$: 0 is expansionary, and if s is an expansionary stage for R, and x is the largest number mentioned at stage s (or before), then the next expansionary stage is the least stage t at which $\Gamma^{\beta_t} \supseteq \alpha_t \upharpoonright x$.

The module $M_R(a, 1, \varepsilon)$ is:

> Wait for an expansionary stage, then add 2^{-a-1} to α; wait for another expansionary stage, and return with $U_M = \emptyset$.

For $n > 1$, the module $M_R(a, n, \varepsilon)$ is:

> Let b be the least $b > a$ such that $2^{-b} < \varepsilon/2$; let $\varepsilon' = \varepsilon/2(b+n-a)$.
> (1) For $k = a+1, a+2, \ldots, b+n$, call $M_R(k, n-1, \varepsilon')$, and add the returned set into U_M.
> (2) Add the current version of $\beta \upharpoonright b$ into U_M, and wait for a stage at which $\beta \upharpoonright b$ changes.
> (3) Wait for an expansionary stage, then add 2^{-b-n-1} to α. Wait for another expansionary stage, and return U_M.

We verify some properties of these modules, which will lead to the full construction.

Lemma 4.5. *A module $M_R(a,n,\varepsilon)$ does not change $\alpha \upharpoonright a$. Indeed, there is a computable function $c(a,n,\varepsilon)$ such that for all R, a, n and ε, if $c = c(a,n,\varepsilon)$, and a module $M = M_R(a,n,\varepsilon)$ starts at stage s with $\alpha_s \upharpoonright [a,c) = 0^{c-a}$, then throughout its run, M changes only $\alpha \upharpoonright [a,c)$, and if it returns at a stage t, then $\alpha_t \upharpoonright [a,c) = 10^{c-a-1}$.*

Proof. By induction on n. The module $M_R(a,1,\varepsilon)$ only changes $\alpha(a)$ from 0 to 1 so we can let $c = a+1$. For $n > 2$, we can calculate b (and ε') as in the instructions for the module, and let

$$c(a,n,\varepsilon) = \max\Big\{b+n, c(k,n-1,\varepsilon') \,:\, k \in [a+1,\ldots,b+n]\Big\}.$$

Let $c = c(a,n,\varepsilon)$. Since $c \geqslant c(k,n-1,\varepsilon')$ for all $k \in [a+1,\ldots,b+n]$, if we start with $\alpha_s \upharpoonright [a,c) = 0^{c-a}$ then by induction, after m iterations of part (1) of the module $M_R(a,n,\varepsilon)$, we have $\alpha \upharpoonright [a,c) = 01^m0^{c-a-m-1}$ and so at the end of part (1) we get $\alpha \upharpoonright [a,c) = 01^{b+n-a}0^{c-b-n-1}$. At part (2) of the module, $\alpha \upharpoonright [a,c)$ doesn't change, and at part (3), we set $\alpha \upharpoonright [a,c) = 10^{c-a-1}$ as is promised. □

Lemma 4.6. *The measure of the set of strings U_M enumerated by a module $M_R(a,n,\varepsilon)$ is at most ε.*

Proof. By a pretty easy induction on n. □

Again for the next lemma, we also think of finite binary strings as natural numbers.

Lemma 4.7. *Suppose that a module $M_R(a,n,\varepsilon)$ starts running at stage s (with $\alpha_s \upharpoonright [a,c) = 0^{c-a}$, c as above), and returns at stage t. Then*

$$\beta_t \upharpoonright a+1 - \beta_s \upharpoonright a+1 \geqslant n.$$

Proof. This goes by induction on n. The base case $n = 1$ is easy: if $M_R(a,1,\varepsilon)$ starts running at stage s and $\alpha_s(a) = 0$, then the module changes $\alpha_s(a)$ to 1, and so by the next expansionary stage we get a change in $\beta \upharpoonright a+1$, which implies that $\beta_t \upharpoonright a+1 - \beta_s \upharpoonright a+1 \geqslant 1$.

Now assume that the lemma is proved for some $n \geqslant 1$. Assume that we start running $M_R(a,n+1,\varepsilon)$ at stage s_a with $\alpha_{s_0} \upharpoonright [a,c) = 0^{c-a}$ (where $c = c(a,n+1,\varepsilon)$). Calculate the associated b and ε'. For $k = a+1,\ldots,b+n+1$, let s_k be the stage at which the recursive call of $M_R(k,n,\varepsilon')$ returns. By induction, we have, for all such k,

$$\beta_{s_k} \upharpoonright k+1 - \beta_{s_{k-1}} \upharpoonright k+1 \geqslant n.$$

For any $m < \omega$, let $Q_m = \{z2^{-m} \,:\, z \in \mathbb{Z}\}$ (see figure 4). For $\gamma \in (-\infty,\infty)$, let $\llcorner\gamma\lrcorner_m$ be the greatest element of Q_m which is not greater than γ. Then for $\gamma \in 2^\omega$, again identified with an element of $[0,1]$, we have $\gamma \upharpoonright m = 2^m\llcorner\gamma\lrcorner_m$. Also, for $\gamma < \delta$, let

$$d_m(\gamma,\delta) = 2^m\left(\llcorner\delta\lrcorner_m - \llcorner\gamma\lrcorner_m\right) = \#\left(Q_m \cap (\gamma,\delta]\right).$$

The induction hypothesis thus says that

$$d_{k+1}\left(\beta_{s_{k-1}}, \beta_{s_k}\right) \geqslant n. \tag{4.2}$$

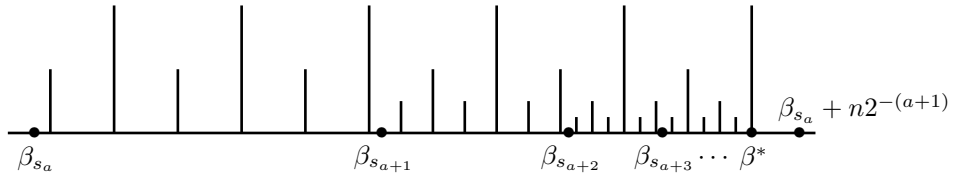

FIGURE 4. The longest lines represent elements of Q_{a+1}, the next longest lines elements of Q_{a+2}, etc. In this example, $n = 6$.

Let $\beta^* = \llcorner \beta_{s_a} \lrcorner_{a+1} + n2^{-a-1}$. So $\beta^* \in Q_{a+1}$.

By induction on $k = a, \ldots, b+n+1$, we show that

$$d_{k+1}\left(\beta_{s_k}, \beta^*\right) \leqslant n. \tag{4.3}$$

For $k = a$ this is immediate. If $k > a$ and (4.3) is true for $k-1$, then because β^* is in Q_k, for every $x \in (Q_{k+1} \setminus Q_k) \cap (\beta_{s_{k-1}}, \beta^*]$ there is some $y > x$ in $Q_k \cap (\beta_{s_{k-1}}, \beta^*]$ and so

$$d_{k+1}\left(\beta_{s_{k-1}}, \beta^*\right) \leqslant 2n.$$

Together with (4.2) we get (4.3) for k: see Figure 5.

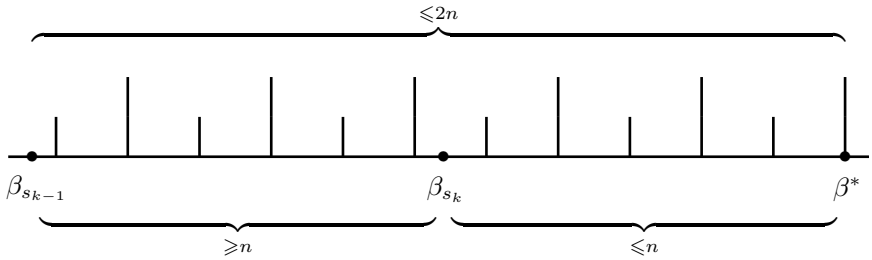

FIGURE 5. The longer lines represent elements of Q_k while the shorter lines represent elements of $Q_{k+1} \setminus Q_k$. Again in this example, $n = 6$.

At the end, we get $d_{a+b+2}(\beta_{s_{b+n+1}}, \beta^*) \leqslant n$, so $\beta^* - \beta_{s_{b+n+1}} \leqslant (n+1)2^{-b-n-2}$. By Cantor's theorem, $2^n \geqslant n+1$, so $\beta^* - \beta_{s_{b+n+1}} \leqslant 2^{-b-2}$ (see Figure 6). It follows that $\beta^* \upharpoonright b = \beta_{s_{b+n+1}} \upharpoonright b$.

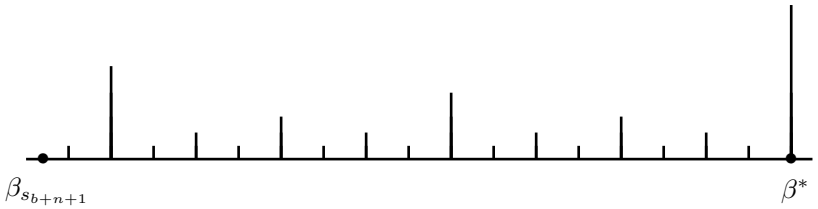

FIGURE 6. Even though $d_{b+n+2}(\beta_{s_{b+n+1}}, \beta^*)$ may be n (in this example, $n \geqslant 17$), we see that β^* is the only element of Q_{b+2} between $\beta_{s_{b+n+1}}$ and β^* itself. Again the shortest lines represent elements of Q_{b+n+2}, the next shortest ones elements of Q_{b+n+1}, etc.

So at the end of step (2) of the module $M_R(a, n+1, \varepsilon)$, say at a stage t_0, we have $\beta_{t_0} > \beta^*$, so $d_{a+1}(\beta_{s_a}, \beta_{t_0}) \geqslant n$, in other words, $\beta_{t_0} \upharpoonright a+1 - \beta_{s_a} \upharpoonright a+1 \geqslant n$. At step (3) of the module, we change $\alpha(a)$ and by the next expansionary stage, we have a change in $\beta \upharpoonright a+1$, which adds at least 1 to $\beta \upharpoonright a+1$. So if the module returns at stage t_1, we have

$$\beta_{t_1} \upharpoonright a+1 - \beta_{s_a} \upharpoonright a+1 \geqslant n+1$$

as required. □

The construction is now clear: we partition ω into disjoint intervals, all of the form $[a, c(a, 2^{a+1}, 2^{-n}))$. A requirement R will be assigned infinitely many such intervals and run $M_R(a, 2^{a+1}, 2^{-n})$ on the n^{th} interval assigned to R. By Lemma 4.7, none of these modules can return, because we cannot have $\beta \upharpoonright a+1 \geqslant 2^{a+1}$. If the hypothesis $\Gamma^\beta = \alpha$ of R holds, then the module cannot ever be stuck waiting for an expansionary stage, and so it must get stuck waiting for β to avoid the set of intervals U_M which it enumerates. In other words, if the n^{th} module for R enumerates U_n, and $\Gamma^\beta = \alpha$, then $\beta \in U_n$ as required; so β is not random.

Finally, we can add multiple permitting to this construction to get the proof of Proposition 1.10. This is done exactly as in section 4.1, so we describe the proof swiftly. Let $\mathbf{d}$ be a c.e., array non-computable degree. Partition ω into an infinite computable coding set C and pairwise disjoint intervals of the form $[a, c(a, 2^{a+1}, 2^{-n}))$ (all disjoint from C). Assign to every requirement R infinitely many intervals, where the n^{th} interval assigned to n is of the form $[a, c(a, 2^{a+1}, 2^{-n}))$. For every module $M_R(a, n, \varepsilon)$, calculate the total number $p(a, n, \varepsilon)$ of stages at which the module changes α. Now partition ω into a very strong array[8] $\langle F_n \rangle$, where for every requirement, for almost every n, $|F_n| > p(a, 2^{a+1}, 2^{-n})$ (where the n^{th} interval for R is $[a, c)$). Let $D \in \mathbf{d}$ be a c.e. $\langle F_n \rangle$-a.n.c.[9]

Every requirement R enumerates an auxiliary c.e. set W_R. To *request permission*, a module $M = M_R(a, 2^{a+1}, 2^{-n})$ enumerates some new $x \in F_n$ into W_R. Permission is later received when x appears in D. The new instructions for the modules are as for the original ones, except that every change in α requires permission:

The new module $M_R(a, 1, \varepsilon)$ is:

> Wait for an expansionary stage, then request permission. When permission is received, add 2^{-a-1} to α; wait for another expansionary stage, and return with $U_M = \emptyset$.

For $n > 1$, the new module $M_R(a, n, \varepsilon)$ is:

> Let b be the least $b > a$ such that $2^{-b} < \varepsilon/2$; let $\varepsilon' = \varepsilon/2(b+n-a)$.
> (1) For $k = a+1, a+2, \ldots, b+n$, call $M_R(k, n-1, \varepsilon')$, and add the returned set into U_M.
> (2) Add the current version of $\beta \upharpoonright b$ into U_M, and wait for a stage at which $\beta \upharpoonright b$ changes.

[8] recall from [19] that a very strong array is a computable sequence (I_n) of disjoint finite sets of natural numbers such that $|I_n| < |I_{n+1}|$ for all $n \in \mathbb{N}$.

[9] recall from [19] that a c.e. set D is $\langle F_n \rangle$-a.n.c. ($\langle F_n \rangle$ array non-computable) if for each c.e. set W there is some k such that $W \cap F_k = D \cap F_k$.

(3) Wait for an expansionary stage, then request permission. When permission is received, add 2^{-b-n-1} to α. Wait for another expansionary stage, and return U_M.

For every R and n, we run $M_R(a, 2^{a+1}, 2^{-n})$ on the n^{th} interval $[a, c)$ assigned for R. We also let $\alpha(c_n) = 1$ iff $n \in D$, where c_n is the n^{th} element of C. That's the construction.

The Lemmas 4.5, 4.6, and 4.7 hold for the new modules as well, so there is no interaction between the requirements. If $\Gamma^\beta = \alpha$, then the R-modules do not get stuck waiting for expansionary stages. For such R and every n such that $W_R \upharpoonright F_n = D \upharpoonright F_n$, every request for permission by the n^{th} module for R is granted, so the n^{th} module for R gets stuck waiting for β to leave U_n (the open set enumerated by R). Thus for infinitely many n we have $\beta \in U_n$. By adjusting the set (letting $U'_m = \bigcup_{n>m} U_n$) we build a Martin-Löf test which covers β, so β is not random.

4.4. **Failure of multiple permitting II: random reals.** Here we show Proposition 1.11: If α has array computable Turing degree, then there is some random left-c.e. real $\beta \geqslant_{\text{ibT}} \alpha$.

Again the idea is to use an approximation to α which doesn't change too much, and build β by requesting that it change by at least 2^{-n} whenever $\alpha \upharpoonright n$ changes. On top of this, we need to ensure that β is random. To accomplish this task, let U be the first element of the universal Martin-Löf test. If we ensure that $\beta \notin U$ then we will have ensured that β is random.

We can think of U as a c.e. antichain of strings (the set of reals is the set of all infinite extensions of strings in U). So $\sum_{\sigma \in U} 2^{-|\sigma|} \leqslant 1/2$. To ensure that β is not in U, whenever we discover that there is some $\sigma \in U$ which is an initial segment of the current version of β, we request that this change by adding $2^{-|\sigma|}$ to β. Thus if the approximation to α is sufficiently tight so that the total amount of quantity added to β on behalf of following α is less than half, this strategy will succeed.

So fix a computable order which satisfies

$$\sum_{n \geqslant 1} g(n) 2^{-n} < 1/2$$

(we allow the value 0 for g for some finitely many inputs; we assume that if $g(n) = 0$ then $\alpha \upharpoonright n$ is simply given to us as a parameter in the construction).

Get a computable approximation $f_s(n)$ for the function $n \mapsto \alpha \upharpoonright n$ such that for all n,

$$\#\{s : f_{s+1}(n) \neq f_s(n)\} \leqslant g(n).$$

The left-c.e. real β is defined by recursion: we start with $\beta_0 = 0$; and let

$$\beta_{s+1} = \beta_s + \sum\{2^{-n} : f_{s+1}(n) \neq f_s(n)\} + \sum\{2^{-|\sigma|} : \sigma \in U_s \ \&\ \sigma \subset \beta_s\}.$$

Lemma 4.8. *$\beta < 1$ (and so is well-defined as an element of 2^ω).*

Proof. There are two kinds of contributions to β: following α and avoiding U. Now g satisfies $\sum_n g(n)2^{-n} < 1/2$ and for all n we have $\#\{s : f_{s+1}(n) \neq f_s(n)\} \leqslant g(n)$, so the total added to β by the clause $\sum\{2^{-n} : f_{s+1}(n) \neq f_s(n)\}$ is less than

one half. Because β is a left-c.e. real, for every $\sigma \in U$, there is at most one stage s at which we have $\sigma \in U_s$ and $\sigma \subset \beta_s$, so the contribution of the clause $\sum\{2^{-|\sigma|} : \sigma \in U_s \ \&\ \sigma \subset \beta_s\}$ is at most $\sum_{\sigma \in U} 2^{-|\sigma|} \leqslant 1/2$. □

Lemma 4.9. *$\beta \notin U$.*

Proof. First note that β is not computable, because otherwise α would be computable. So β is irrational, it has a unique infinite binary expansion and the digits of (β_s) converge pointwise to the digits of β. If $\beta \in U$ then there is some $\sigma \in U$ such that $\sigma \subset \beta$. Then at a late enough stage s we have $\sigma \in U_s$ and $\sigma \subset \beta_s$, so at stage s we add $2^{-|\sigma|}$ to β and force $\sigma \not\subset \beta$ for ever. □

Finally, the first clause in the definition of β shows that $\beta \geqslant_{\mathrm{ibT}} \alpha$ (see Lemma 4.3).

5. cl reductions outside the c.e. degrees

Here we prove Theorem 1.13: every Turing degree $\mathbf{d}$ which is not generalised low_2 computes a real which is not cl-reducible to a random real. As discussed in the introduction, this is an application of non GL_2-permitting to a construction of Hirschfeldt's. The basic Lemma of Hirschfeldt (which can be found in [16]) is:

Lemma 5.1. *If h is a computable order and Γ is an ibT-functional then the set of strings $\tau \in 2^{<\omega}$ such that for all $\eta \in 2^{|\tau|}$, if $\Gamma^\eta = \tau$ then $K(\eta) < h(|\eta|)$, is dense.*

Iterating Lemma 5.1 over all computable orders (since there is a $\mathbf{0}'$-listing of those) and ibT-functionals, Hirschfeldt gets a real below $\mathbf{0}'$ which is not cl-reducible to a complex real. An immediate consequence of Hirschfeldt's lemma is that every 2-generic real is also not cl-reducible to a complex real. To get our result, we use the technical Lemma 5.2 to show that non-GL_2 degrees provide sufficient genericity (compare with the fact that every non-GL_2 degree bounds a 1-generic real).

Lemma 5.2. *Suppose that $\langle D_e \rangle$ is a sequence of subsets of $2^{<\omega}$, uniformly computable from $\mathbf{0}'$. Also suppose that $\{e : D_e \text{ is dense}\}$ is Π^0_2-definable. Then every non-GL_2 degree bounds a real which meets every set D_e which is dense.*

To obtain Theorem 1.13 from Lemma 5.2, let $\langle \Gamma_e \rangle$ be an effective listing of all ibT-functionals and let $\langle h_i \rangle$ be an effective listing of all non-decreasing, partial computable functions (whose domain is an initial segment of ω) such that for all i, if h_i is total then it is unbounded, and so an order (such a listing can be obtained by not recognising following convergences until a larger value is obtained). For $e, i < \omega$, let $D_{e,i}$ be the set of strings τ such that $h_i(m)\downarrow$ for all $m \leqslant |\tau|$ and for all $\eta \in 2^{|\tau|}$, if $\Gamma_e^\eta = \tau$ then $K(\eta) < h_i(|\tau|)$. These sets are uniformly computable in $\mathbf{0}'$. If h_i is total then it is an order and so by Hirschfeldt's 5.1, $D_{e,i}$ is dense. If h_i is not total then $D_{e,i}$ is finite. It follows that the set of pairs (e, i) such that $D_{e,i}$ is dense is Π^0_2-definable. By Lemma 5.2, if $\mathbf{d}$ is non-GL_2, there is some $A \leqslant_T \mathbf{d}$ such that for all e and i such that h_i is an order, there is some $\tau \subset A$ which is in $D_{e,i}$. Then if $B \geqslant_{\mathrm{ibT}} A$ and h is an order, then there is some $\tau \subset A$ such that $K(B \upharpoonright |\tau|) < h(|\tau|)$ and so B is not complex via h; so B is not complex.

We turn to prove Lemma 5.2. We define two fast-growing functions, both computable in $\mathbf{0}'$:

- Let $D_{e,s}$ be a computable approximation for D_e (uniformly in e). Let m_e be a modulus for $D_{e,s}$, uniformly computable from $\mathbf{0}'$. Let f_0 be a function such that for all n, for all $e \leqslant n$ and all $\sigma \in 2^{\leqslant n}$, $f_0(n) \geqslant m_e(\sigma)$.
- The set of indices e such that D_e is dense is co-c.e. in $\mathbf{0}'$. Using an enumeration of the set of e's such that D_e is not dense, we can, with oracle $\mathbf{0}'$, compute a function which takes a string σ to an extension in D_e, or discovers that D_e is not dense. So computably from $\mathbf{0}'$ we can compute a function f_1 such that for all n, for all $e \leqslant n$ and all $\sigma \in 2^{\leqslant n}$, if D_e is dense, then $f_1(n)$ bounds the length of some extension of σ in D_e.

We may further assume that if D_e is not dense then it is finite; because as mentioned, $\mathbf{0}'$ can enumerate those e's such that D_e is not dense, and once e is enumerated, $\mathbf{0}'$ can declare that no more strings are ever accepted as elements of D_e. Indeed, we may assume that if D_e is not dense then there is a stage s such that $D_{e,s}$ is finite and for all $t \geqslant s$, $D_{e,t} = D_{e,s}$; because we can approximate these e's in a Σ^0_2 fashion and refuse to accept changes to $D_{e,t}$ until an "expansionary" stage is discovered. [In detail: say that D_e is dense iff $\forall x \exists y\, C(e,x,y)$, where C is a computable predicate. Inductively define *e-expansionary* stages to be 0, and if s is e-expansionary, then the next e-expansionary stage is the next stage t such that for all $x < s$ there is some $y < t$ such that $C(e,x,y)$ holds. Redefine the approximation to D_e by letting $\hat{D}_{e,t} = D_{e,s}$ for the greatest e-expansionary stage $s \leqslant t$.]

The functions f_0 and f_1 defined above have a property which is somewhat stronger than simply being Δ^0_2: by possibly increasing their values, they each have a computable approximation which is non-decreasing in s (this doesn't change their desired properties). It follows that the function f, defined recursively by letting $f(0) = 0$ and $f(s+1) = f_0(f_1(f(s)))$, also has such an approximation.

Lemma 5.3. *There is some $g \leqslant_T \mathbf{d}$ such that for all n, $g(n) \leqslant f(n)$ and for infinitely many n, $g(n) = f(n)$.*

Proof. Let m_f be a modulus for f_s, the non-decreasing (in s) computable approximation for f. Since $\mathbf{d}$ is non-GL_2, there is some $h \leqslant_T \mathbf{d}$ which is not dominated by m_f. For all n, let $g(n) = f_{h(n)}(n)$. □

We use g to bound the searches in our imitation of the $\mathbf{0}'$ construction: we recursively define an increasing sequence of strings σ_s, starting with $\sigma_0 = \langle\rangle$, aiming to define $A = \bigcup_s \sigma_s$ and try to meet the requirements

R_e**:** There is some $\sigma \subset A$ in D_e.

At stage s, we say that requirement R_e is currently *satisfied* if there is some $\tau \subseteq \sigma_s$ such that $\tau \in D_{e,g(s+1)}$. A requirement R_e *requires attention* at stage s if it is not currently satisfied and there is some $\tau \supset \sigma_s$ of length at most $g(s+1)$ which is in $D_{e,g(s+1)}$. At stage s we act for the strongest requirement which wishes to act. To act for R_e, we choose the shortest extension τ of σ_s which is in $D_{e,g(s+1)}$ and set $\sigma_{s+1} = \tau$. If no requirement wishes to act, we set $\sigma_{s+1} = \sigma_s$. That's the construction.

Lemma 5.4. *For all s, $|\sigma_s| \leqslant g(s)$, and so for all s, $|\sigma_s| \leqslant f(s)$.*

A stage s is called *true* if $f(s+1) = g(s+1)$.

Lemma 5.5. *Suppose that a requirement R_e feels satisfied at a true stage $s > e$. Then it is met (and feels satisfied at every stage $t \geqslant s$).*

Proof. At stage s we identify some $\tau \subseteq \sigma_s$ which is in $D_{e,g(s+1)} = D_{e,f(s+1)}$. Since $f(s+1) \geqslant f_0(f(s))$ and $f(s) \geqslant |\tau|, e$, we have $f(s+1) \geqslant m_e(\tau)$ and so $\tau \in D_e$. For every $t \geqslant s$, since we may assume that $g(t+1) \geqslant g(s+1)$, we have $\tau \in D_{e,g(t+1)}$ and so R_e feels satisfied at stage t. □

Lemma 5.6. *Suppose that D_e is dense and that requirement R_e acts at a true stage $s > e$. Then R_e is met and feels satisfied at every stage $t > s$.*

Proof. At stage s we define $\sigma_{s+1} = \tau$ where τ is the shortest extension of σ_s in $D_{e,g(s+1)} = D_{e,f(s+1)}$. Since D_e is dense and $|\sigma_s| \leqslant f(s)$, we know that there is an extension τ' of σ_s in D_e of length at most $f_1(f(s))$, and that $f(s+1) = f_0(f_1(f(s)) \geqslant m_e(\tau')$, so $\tau' \in D_{e,f(s+1)}$. So $|\tau| \leqslant |\tau'| \leqslant f_1(f(s))$ and so $f(s+1) \geqslant m_e(\tau)$. So $\tau \in D_e$ and for all $t > s$ we have $\tau \in D_{e,g(t+1)}$ (again we assume that $g(t+1) \geqslant g(s+1)$ so R_e feels satisfied at stage t. □

Lemma 5.7. *If D_e is dense, $s > e$ is a true stage, and R_e does not feel satisfied at stage s, then R_e requires attention at stage s.*

Proof. Since D_e is dense, there is some extension of σ_s in D_e. In fact, since $|\sigma_s| \leqslant f(s)$, there is such an extension τ of length at most $f_1(f(s))$, and so $g(s+1) = f(s+1) = f_0(f_1(f(s)) \geqslant m_e(\tau)$ so $\tau \in D_{e,g(s+1)}$ so R_e requires attention at stage s. □

Lemma 5.8. *If D_e is not dense then there is a stage after which R_e never requires attention.*

Proof. If n is the greatest length of a string which appears in any $D_{e,s}$ then after stage n, R_e never finds an extension of σ_n in $D_{e,g(n+1)}$ and so does not require attention. □

Overall we see that every requirement eventually stops requiring attention and so the finite-injury priority argument works and for all e such that D_e is dense, R_e is met. This concludes the proof of Lemma 5.2.

Remark 5.9. Lemma 5.2 holds if the set of dense sets is only Σ^0_3-definable. This can be seen by a modification of the proof, but also directly using the lemma for Π^0_2 and changing the enumeration of the sets: change $\langle D_e \rangle$ to $\langle D_{e,t} \rangle$ where $D_{e,t} = D_e$ if t is a Σ^0_3-witness for the fact that D_e is dense, and otherwise make $D_{e,t}$ finite.

Remark 5.10. Another way to show lemma 5.2 was pointed out by the anonymous referee. Let D_e be as in lemma 5.2 and let $h(e,n)$ be a computable 0-1 function such that D_e is dense iff $f(e,n) = 1$ for infinitely many n. The sets

$$C_e = D_e \cup \{\sigma \mid |\sigma| > n \text{ and } f(e,m) = 0 \text{ for all } m > n\}$$

are uniformly $\emptyset'$-computable and dense. Also, if D_e is dense then $C_e = D_e$, so meeting every C_e is the same as meeting all dense D_e.

Thus lemma 5.2 reduces to showing that for every uniformly $\emptyset'$-computable sequence of dense sets of binary strings, every non-GL_2 degree bounds a real meeting every set in the sequence. This is a consequence of a combination of results in computable model theory and genericity arguments with GL_2 from [20, 9].

6. Proof of a theorem of Frank Stephan's

We give a proof of Theorem 1.12 (due to Frank Stephan) that the left-c.e. random reals inhabit more than one cl-degree. First we give some immunity properties of random sequences.

Definition 6.1 (Fenner and Schaefer [22])**.** A set A is called k-immune ($k \in \mathbb{N}$) if there is no computable sequence $\langle F_n \rangle$ of pairwise disjoint sets such that for all n, $|F_n| \leqslant k$ and $F_n \cap A \neq \emptyset$.

Note that every set is 0-immune and that the 1-immune sets are exactly the immune sets; also, hyperimmune sets are k-immune for all $k < \omega$. Fenner and Schaefer [22] showed that the classes of k-immune sets form a proper hierarchy.

Theorem 6.2 (Folklore)**.**

(1) *For all $k < \omega$, every random set is k-immune (indeed, every Kurtz-random*[10] *set is k-immune).*
(2) *No random set is hyperimmune.*

Proof. For (1) we need the following

Lemma 6.3. *Suppose that $|B| = m$ and $\mathcal{B} \subseteq \mathcal{P}(B)$ is a collection of n pairwise disjoint subsets of B, each of size k. Then there are exactly $2^{m-kn}(2^k-1)^n$ subsets of B which have non-empty intersection with every set in $\mathcal{B}$.*

Proof. Each $B' \in \mathcal{B}$ has $2^k - 1$ non-empty subsets; so there are $(2^k-1)^n$ many subsets of $\bigcup \mathcal{B}$ with non-empty intersection with every $B' \in \mathcal{B}$. The size of $B - \bigcup \mathcal{B}$ is $m - kn$; each subset of B of the desired property is determined by its intersection with $\bigcup \mathcal{B}$ and with $B - \bigcup \mathcal{B}$, which can be chosen independently. □

For (1), let $\langle F_n \rangle$ be a computable collection of pairwise disjoint sets, each of size k, and let $\mathcal{Q}$ be the class of sets which have non-empty intersection with every F_n. Then $\mathcal{Q}$ is Π^0_1. By weeding, we may assume that for all n, $\max F_n < \min F_{n+1}$. By Lemma 6.3, for every n, The number of subsets of $1 + \max F_n$ which intersect every F_m (for $m = 1, \dots, n$) is $2^{1+\max F_n - kn}(2^k-1)^n$ and so the measure of the class $\mathcal{Q}_n$ of infinite sets which intersect every F_m for $m = 1, \dots, n$ is

$$\mu(\mathcal{Q}_n) = \frac{2^{1+\max F_n - kn}(2^k-1)^n}{2^{1+\max F_n}} = \left(1 - \frac{1}{2^k}\right)^n$$

which approaches 0 as $1 - 2^{-k} < 1$. Since $\mathcal{Q} = \bigcap \mathcal{Q}_n$, $\mathcal{Q}$ is a null class. So if A is Kurtz-random then $A \notin \mathcal{Q}$ so A doesn't fail to be k-immune because of $\langle F_n \rangle$.

For (2), let $\langle F_n \rangle$ be a computable sequence of pairwise disjoint sets such that $|F_n| = 2^n$. This time let $\mathcal{Q}_n$ be the class of sets A such that $F_n \cap A$ is empty. So $\mu(\mathcal{Q}_n) = 2^{-n}$. So if we let $U_n = \bigcup_{m>n} \mathcal{Q}_n$ then $\langle U_n \rangle$ is a Martin-Löf test, and $\bigcap_n U_n$ is the collection of sets A such that there are infinitely many n such that $F_n \cap A$ is empty, and so contains every hyperimmune set. □

Proof of Theorem 1.12. Let α be a random left-c.e. real. Note that $\omega \setminus \alpha$ (where α is viewed as a subset of ω) is also random. By Theorem 6.2(2), there is a strictly

[10] recall from e.g. [32] that a set is Kurtz-random iff it is not a member of a null Π^0_1 class. This is a very weak notion of randomness.

increasing computable sequence $\langle t_n \rangle$ such that for every n, $\alpha \upharpoonright (t_n, t_{n+1}]$ contains a 0 digit.

Let $\beta(x) = 1$ iff $x = t_n$ for some $n > 0$ (so $\beta = \sum_{n>0} 2^{-t_n - 1}$). For every $n > 0$,

$$\alpha \upharpoonright (t_n + 1) + \beta \upharpoonright (t_n + 1) = (\alpha + \beta) \upharpoonright (t_n + 1); \tag{6.1}$$

this is because for $m < n$, $\alpha \upharpoonright (t_m, t_{m+1}]$ is not a string of 1's and so adding $2^{-t_m - 1}$ doesn't carry before the $t_m{}^{\text{th}}$ digit. It follows that

$$K((\alpha + \beta) \upharpoonright t_n + 1) =^+ K(\alpha \upharpoonright t_n + 1) =^+ t_n$$

(here $=^+$ denotes equality modulo a constant) and so $\alpha + \beta$ is random (and left-c.e.).

We claim that α does not cl-compute $\alpha + \beta$. For otherwise there would be some constant $c < \omega$ and a partial computable function ψ such that for all n,

$$\psi(\alpha \upharpoonright (n + c)) = (\alpha + \beta)(n).$$

By Theorem 6.2(1), there are infinitely many n such that $\alpha \upharpoonright [t_n - c, t_n) = 1^c$. For each such n, by Equation 6.1, we have

$$\psi(\alpha \upharpoonright t_n) = (\alpha + \beta)(t_n - c) = 1 - \alpha(t_n).$$

But this gives rise to a c.e. martingale[11] which succeeds on α: let $M(\sigma i)$ split $M(\sigma)$ evenly between both outcomes, except when $|\sigma| = t_n$ for some $n > 0$ and σ ends with a string of c 1's; then wait for $\psi(\sigma)$ to converge, in which case all the capital is spent guessing the next bit is $1 - \psi(\sigma)$. This, of course, contradicts the fact that α is random. □

References

[1] Bahareh Afshari, George Barmpalias, S. Barry Cooper and Frank Stephan, Post's programme for the Ershov hierarchy, *Journal of Logic and Computation* 2007 **17** : 1025–1040.

[2] George Barmpalias, *Computability and applications to analysis*, PhD thesis, University of Leeds, U.K., 2004.

[3] George Barmpalias, Hypersimplicity and semicomputability in the weak truth table degrees, *Archive for Math. Logic* **44** (2005), 1045–1065.

[4] George Barmpalias and Andrew E. M. Lewis, Random reals and Lipschitz continuity, *Mathematical Structures in Computer Science* Volume **16** (2006)

[5] George Barmpalias and Andrew E. M. Lewis, The ibT degrees of c.e. sets are not dense, *Annals of Pure and Applied Logic* Volume **141** (2006)

[6] George Barmpalias and Andrew E. M. Lewis, Randomness and the linear degrees of computability, *Annals of Pure and Applied Logic* **145** (2007), 252–257.

[7] George Barmpalias and Andrew E. M. Lewis, A c.e. real that cannot be sw-computed by any Ω-number, *Notre Dame Journal of Formal Logic* Volume **47** Issue 2 (2006)

[8] John Chisholm, Jennifer Chubb, Valentina S. Harizanov, Denis R. Hirschfeldt, Carl G. Jockusch, Jr., Timothy McNicholl, and Sarah Pingrey, Π^0_1 Classes and strong degree spectra of relations, *Journal of Symbolic Logic* **72** (2007), 1003 – 1018.

[9] Chris J. Conidis, Classifying Model Theoretic Properties, *Journal of Symbolic Logic*, Vol. **73**(3) (2008) 885–905.

[10] Barbara F. Csima and Robert I. Soare, Computability results used in differential geometry, *Journal of Symbolic Logic* **71** (2006), 1394–1410

[11] Martingales is another way to define effective randomness. Recall, e.g. from [32], that a martingale is a function d from binary strings to non-negative reals such that $2d(\sigma) = d(\sigma 0) + d(\sigma 1)$ for all strings σ. A martingale is computably enumerable (c.e.) if there is a computable procedure which, given σ, produces a computable increasing sequence of rationals converging to $d(\sigma)$. We say that a martingale succeeds on an infinite binary sequence if the supremum of the values it takes on the sequence is unbounded. It is a basic fact that a sequence Z is Martin-Löf random iff no c.e. martingale succeeds on Z.

[11] David Doty, Every sequence is decompressible from a random one. in Arnold Beckmann et. al. (eds.), *Logical approaches to computational barriers*, proceedings of the second conference on computability in Europe, Lecture Notes in Computer Science *3988*, Springer-Verlag, 2006, 153–162.

[12] Rod Downey and Noam Greenberg, Totally $< \omega^\omega$-computably enumerable degrees and embeddings of the 1-3-1 lattice, in preparation.

[13] Rod Downey and Noam Greenberg, Totally $< \omega^\omega$-computably enumerable degrees and m-topped degrees, In Cai, Cooper, Li (eds.), *Theory and Applications of Models of Computation* (proceedings of TAMC 2006, Beijing), volume 3959 of *Lecture Notes in computer Science*, Springer (2006), 46–60.

[14] Rod Downey and Noam Greenberg, Turing degrees of reals of positive packing dimension, to appear in *Information Processing Letters*.

[15] Rod Downey, Noam Greenberg and Rebecca Weber, Totally ω-computably enumerable degrees and bounding critical triples, preprint.

[16] Rod Downey and Denis R. Hirschfeldt, Algorithmic randomness and complexity, Springer, to appear.

[17] Rod Downey, Denis R. Hirschfeldt and Geoff LaForte, Randomness and reducibility. Mathematical foundations of computer science, 2001, 316–327, *Lecture Notes in Comput. Sci. 2136*, Springer, Berlin, 2001.

[18] Rod Downey, Denis R. Hirschfeldt and Geoff LaForte, Randomness and reducibility. *Journal of Computer and System Sciences* **68** (2004) 96–114

[19] Rod Downey, Carl G. Jockusch, Jr., and Michael Stob, Array nonrecursive sets and multiple permitting arguments, *Recursion Theory Week (Proceedings, Oberwolfach 1989) (K. Ambos-Spies, G.H. Müller and G.E. Sacks, eds.) Lecture Notes in Math. 1432*, Springer-Verlag, 1990, 141–173

[20] Denis R. Hirschfeldt, Richard A. Shore, and Theodore A. Slaman, The atomic model theorem, Preprint.

[21] Rod Downey, Carl G. Jockusch, Jr., and Michael Stob, Array nonrecursive sets and genericity, in *Computability, Enumerability, Unsolvability: Directions in Recursion Theory*. London Mathematical Society Lecture Notes Series (Cambridge University Press) *224* (1996), 93–104.

[22] Stephen Fenner and Marcus Schaefer, Bounded immunity and btt-reductions. *Mathematical Logic Quarterly* **45** (1999), 3–21.

[23] Peter Gács, Every sequence is reducible to a random one, *Information and Control* **70** (1986), 186–192.

[24] Shamil Ishmukhametov, Weak recursive degrees and a problem of Spector, in *Recursion Theory and Complexity* (M. Arslanov and S. Lempp, eds.), de Gruyter, Berlin, 1999, 81–88

[25] Bjørn Kjos-Hanssen, Wolfgang Merkle and Frank Stephan, Kolmogorov complexity and the recursion theorem, in *STACS 2006: Twenty-Third Annual Symposium on Theoretical Aspects of Computer Science*, Marseille, France, February 23-25, 2006. Proceedings. Springer LNCS *3884* 149–161, 2006.

[26] Antonin Kučera, Measure, Π^0_1-classes and complete extensions of PA. In: H.-D. Ebbinghaus et al. (eds.), *Recursion Theory Week*. Lecture Notes in Mathematics *1141*:245–259, Springer, 1985.

[27] Antonin Kučera, On the use of diagonally nonrecursive functions. In: H.-D. Ebbinghaus et al. (eds.), *Logic Colloquium 1987*. Studies in Logic and the Foundations of Mathematics *129*:219–239, North-Holland, 1989.

[28] Antonin Kučera and Theodore A. Slaman, Randomness and recursive enumerability. *SIAM Journal of Computing* **31** (2001), 199–211.

[29] Martin Kummer, Kolmogorov complexity and instance complexity of recursively enumerable sets, *SIAM Journal on Computing* **25** (1996), No. 6, 1123–1143.

[30] M. Li and P. Vitányi, An introduction to Kolmogorov complexity and its applications, *Graduate Texts in Computer Science, Second Edition, Springer-Verlag, New York*, 1997.

[31] Wolfgang Merkle and Nenad Mihailović, On the construction of effectively random sets, *Journal of Symbolic Logic* **69** (2004), 862–878.

[32] André Nies. *Computability and Randomness*. Oxford University Press, in preparation, 2009.

[33] Keng Meng Ng, Frank Stephan and Guohua Wu, Degrees of weakly computable reals, Proceedings of "Second Conference on Computability in Europe", CiE 2006, Swansea (2006), 413–422.
[34] Piergiorgio Odifreddi, *Classical recursion theory*, Volume I, North-Holland, Amsterdam, Oxford, 1989.
[35] Liang Yu and Decheng Ding, There is no SW-complete c.e. real. *J. Symbolic Logic* **69** (2004), 1163–1170.

School of Mathematics, Statistics and Computer Science, Victoria University, P.O. Box 600, Wellington, New Zealand
E-mail address: George.Barmpalias@mcs.vuw.ac.nz

School of Mathematics, Statistics and Computer Science, Victoria University, P.O. Box 600, Wellington, New Zealand
E-mail address: downey@mcs.vuw.ac.nz

School of Mathematics, Statistics and Computer Science, Victoria University, P.O. Box 600, Wellington, New Zealand
E-mail address: greenberg@mcs.vuw.ac.nz