Order-Computable Sets

Denis Hirschfeldt[*], Russell Miller[†], & Sergei Podzorov[‡]

September 3, 2007

Abstract

We give a straightforward computable-model-theoretic definition of a property of Δ^0_2 sets, called *order-computability*. We then prove various results about these sets which suggest that, simple though the definition is, the property defies any easy characterization in pure computability theory. The most striking example is the construction of two computably isomorphic c.e. sets, one of which is order-computable and the other not.

1 Introduction

The *Turing degree* of a countable structure (whose domain is a subset of ω) is the join of the Turing degrees of the domain and of the functions and relations on that structure, in the relevant language. We say that a structure is *computable* if it has Turing degree $\mathbf{0}$, the degree of the computable sets. (For the purposes of this paper, all structures are assumed to have domain ω, because we want the Turing degree to reflect the relative computability of the functions and relations in the structure. For us, choosing the domain to be more complex than ω is cheating.)

[*]The first author was partially supported by NSF grant DMS-05-00590.

[†]The second author was partially supported by a VIGRE postdoc under NSF grant 9983660 to Cornell University, by NSF grant DMS 0075899, and by grants numbered 67182-00-36, 60095-34-35, 68470-00-37, and 80209-04-12 from The City University of New York PSC-CUNY Research Award Program.

[‡]The third author was partially supported by Grant KTsFE PD02-1.1-475 from the Ministry of Education of the Russian Federation, and by the Council for Grants (under RF President) and State Aid of Fundamental Science Schools, project NSh-2112.2003.1.

It is common for two isomorphic structures to have different Turing degrees. (For simplicity, the isomorphic structures are often called *copies* of each other.) This observation has led to a great deal of research into the spectra of structures and of relations on structures, which are essentially measurements of the intrinsic computational complexity of the structures themselves and of additional relations on them. We suggest section 15 of [3] for definitions and an overview of this area. (The two main definitions are given in this paper at the beginning of Section 4.)

Most of this research has attempted to produce spectra with certain desirable characteristics, or to prove that no spectrum can have those characteristics. In this paper we take a different, more concrete approach. We consider the structure $(\omega, <)$, the most straightforward copy of a very simple linear order. We add one additional unary relation A to the language, and ask, for specific sets $A \subset \omega$, whether the structure $(\omega, <, A)$ has a computable copy. Clearly the answer is positive if A itself is computable, but it can be so for certain noncomputable A as well. If the structure does have a computable copy, then we call A an *order-computable set*.

In approaching this question, we initially expected to find a straightforward characterization of order-computability using pure computability-theoretic properties of the set A, such as the Turing degree of A and/or the position of A in the Ershov hierarchy. (It is quickly seen that all order-computable sets are Δ^0_2.) However, the property of order-computability defied all attempts at easy characterization in these terms. We believe that the results in this paper will demonstrate to the satisfaction of all that no easy characterization is possible without resorting to model theory, thus reinforcing the general thesis that computability issues become significantly more complex when placed in the context of model theory than in pure computability theory on subsets of ω.

2 Order-Computable Sets

Definition 2.1. A set $A \subseteq \omega$ is *order-computable* if there exists a computable copy of the structure $(\omega, <, A)$ in the language of linear orders with an additional unary predicate.

Notice that for every computable structure $(\omega, \prec, R)$ such that $(\omega, \prec) \cong$

$(\omega, <)$, there is a unique set A such that $(\omega, \prec, R) \cong (\omega, <, A)$. In particular,

$$A = \{n \in \omega : (\exists x \in R)\ x \text{ has exactly } n \text{ predecessors under } \prec\}.$$

Every order-computable set is Δ^0_2, since we may use a $0'$-oracle to find the unique element with exactly n predecessors, then check whether it lies in R. We say that the ordering $(\omega, \prec, R)$ *order-computes* A. In the case when $R = E$, the set of even numbers, we may say simply that $\prec$ order-computes A. This case is standard in the sense that we can always choose $R = E$:

Lemma 2.2. *For every infinite coinfinite order-computable set A, there exists a computable order $\prec$ on ω such that $(\omega, \prec, E) \cong (\omega, <, A)$, where E is the set of even numbers.*

Proof. If $(\omega, <, A) \cong (\omega, \prec', R)$ with R and $\prec'$ computable, then there exists a computable permutation h of ω with $h(E) = R$. Define $m \prec n$ iff $h(m) \prec' h(n)$. ∎

The same would work for any other infinite coinfinite computable set, of course; choosing E just standardizes certain of our constructions.

Before describing any results regarding order-computable sets, we develop some of the tools to be used. If $\prec$ is the order given by Lemma 2.2 corresponding to an order-computable set A, we define the *predecessor approximation function* f_A for A by:

$$f_A(n, s) = \begin{cases} |\{y \leq s : y \prec n\}| & \text{if } n < s \\ \uparrow & \text{otherwise} \end{cases}$$

Thus, for each s, the function $f_A(\cdot, s)$ is a permutation of the numbers $0, \ldots, s-1$. The limit $g_A(n) = \lim_{s\to\infty} f_A(n, s)$ is the *predecessor function* for A. Thus $g_A(E) = A$. (If $|A| = k$ or $|\overline{A}| = k$ with $k < \omega$, then a similar process works with either $\{0, \ldots, k-1\}$ or its complement in place of E.)

It is easy to give conditions for a function f to serve this role:

Definition 2.3. A binary partial function f is a *predecessor approximation function* if it satisfies, for all n and s:

- $f(n, s)\downarrow$ iff $n < s$;
- the function $f(\cdot, s)$ is a permutation of the set $\{0, \ldots, s-1\}$;

- if $n < s$, then $f(n,s) \leq f(n,s+1) \leq f(n,s)+1$; and
- $\lim_t f(n,t)$ converges.

So every order-computable set A gives rise to a computable predecessor approximation function, as described above. (More specifically, each computable order $\prec$ that order-computes A gives rise to such a function, and for distinct orders the functions will be different.) Conversely, any computable predecessor approximation function f arises in this fashion, since we may build a computable order $\prec$ using f by adding s to the order at stage $s+1$ so that s has exactly $f(s,s+1)$ predecessors among the elements already added at previous stages. The first two conditions in Definition 2.3 make it clear that this is always possible. The third shows that $\prec$ really is a computable linear order (since we never change our mind about the order of elements), and the last condition shows that $(\omega, \prec)$ has order type ω.

This analysis allows us to examine the complexity of order-computability. The original Definition 2.1, correctly stated, is Σ^1_1, since it quantifies over isomorphisms between computable linear orders. However, it is not strictly Σ^1_1, since any two computable orders of order type ω are in fact Δ^0_2-isomorphic. (We can use a $\emptyset'$-oracle to find the least element of each, then the successors of those elements, and so on.) The simplest bound on the complexity uses Definition 2.3. If A is a Δ^0_2 set, suppose that for every x, $\lim_s \varphi_e(x,s)\downarrow = A(x)$. Then A is order-computable if:

- A is finite; or
- $\overline{A}$ is finite; or
- there exists a computable predecessor approximation function φ_i such that $\lim_t \varphi_i(n,t) \in A$ iff n is even.

The first two items are both Σ^0_4 properties of e, and the last is also Σ^0_4, once one works through Definition 2.3. So for Δ^0_2 sets (as given by indices of computable approximations), order-computability is actually a Σ^0_4 property, and similarly for c.e. sets, the index set $\{e \in \omega : W_e \text{ is order-computable}\}$ is Σ^0_4. We conjecture that these sets are actually Σ^0_4-complete.

Often we will ensure that a set A is not order-computable by using a diagonal argument to show that no partial computable function φ_e is a predecessor approximation function for A. We say that an approximation $\varphi_{e,t}$ proves that φ_e is not a predecessor approximation function if there exist

n and s such that $\varphi_{e,t}(n, s+1)\downarrow \notin \{\varphi_{e,t}(n,s), \varphi_{e,t}(n,s)+1\}$, or such that $\varphi_{e,t}(n,s)\downarrow \geq s$, or such that $\varphi_{e,t}(\ \cdot\ , s)$ is not one-to-one, or such that $n \geq s$ and $\varphi_{e,t}(n,s)\downarrow$. If there are no such n and s (and if $\varphi_e(n,s)\downarrow$ for all $n < s$), then our construction must either make $\lim_t \varphi_e(w,t) = \infty$ for some witness element w, or make $\lim_t \varphi_e(2n,t) \notin A$ or $\lim_t \varphi_e(2n+1,t) \in A$ for some n.

Building a set A that is order-computable, on the other hand, uses negative requirements. We define a canonical process for trying to build a computable order on ω that order-computes A. Let A be an infinite coinfinite Δ^0_2 set with computable approximation $\langle A_s\rangle_{s\in\omega}$. (We will assume that every A_s is finite.) First we need a simple lemma.

Lemma 2.4. *Let A and $\langle A_s\rangle$ be as above. Then for every s there exists a stage $t > s$ and a strictly increasing function $g : \{0, \ldots, \max A_s\} \to \omega$ such that for each $n \leq \max A_s$,*

$$n \in A_s \iff g(n) \in A_t.$$

Proof. Since A is infinite and coinfinite, there exists a g as required such that

$$n \in A_s \iff g(n) \in A.$$

But also there must be a t so large that A_t and A agree up to $\max(\text{range}(g))$. ∎

We now define the *derived order* $\mathcal{L} = (\omega, \prec, E)$ for the approximation $\langle A_s\rangle$ to A. (We may assume that for all s, no $x \geq s$ lies in A_s.) To construct $\prec$, we start with $\mathcal{L}_0$ consisting of a single element $\notin E$. At stage $s+1$, we write $b_{0,s} \prec b_{1,s} \prec \cdots \prec b_{n,s}$ for the order $\mathcal{L}_s$, and let σ be the string $\langle E(b_{0,s}), \ldots, E(b_{n,s})\rangle \in 2^{n+1}$. If σ is an initial segment of A_{s+1}, we add one fresh element $b_{n+1,s}$ to $\mathcal{L}_s$, with $b_{n,s} \prec b_{n+1,s}$.

If $\sigma \not\subseteq A_{s+1}$, we search for a strictly increasing function $g : \text{dom}(\mathcal{L}_s) \to \omega$ such that

$$(\forall i \leq n)[b_{i,s} \in E \iff g(b_{i,s}) \in A_{s+1}],$$

If there is no such g, let $\mathcal{L}_{s+1} = \mathcal{L}_s$. If there are such functions g, choose the first in the dictionary order. (That is, choose that g with $g(b_{0,s})$ minimal. If several still remain, choose that g with $g(b_{1,s})$ minimal, and so on.) For each $i < n$, suppose $g(b_{i,s}) < k_1 < \cdots < k_p < g(b_{i+1,s})$ are the consecutive integers from $g(b_{i,s})$ to $g(b_{i+1,s})$. We add one new number to the order $\mathcal{L}_{s+1}$

for each k_j, with the new number being even iff $k_j \in A_{s+1}$. Thus we have "embedded" $\mathcal{L}_s$ into A_{s+1}. This completes the construction.

It is easily shown by induction that each time we find the desired g, we have

$$\langle E(b_{0,s+1}), \ldots, E(b_{m,s+1}) \rangle = A_{s+1} \upharpoonright (m+1)$$

where $m+1 = \max A_{s+1}$. This holds at infinitely many stages, since Lemma 2.4 makes it clear that the required g exists at infinitely many stages, so we do build an infinite computable linear order $\mathcal{L}$ by this process. Moreover, $\mathcal{L}$ has an initial segment of order type ω, since A is Δ^0_2 and g was always chosen lexicographically.

However, the derived order $\mathcal{L}$ may not actually be of order type ω. (This will hold for certain approximations $\langle A_s \rangle$ even if A is order-computable.) If $\mathcal{L}$ is of type ω, then clearly A is order-computable. In the rest of this section, we will build several order-computable sets A. Our technique will be to build a computable approximation to each A, ensuring that the derived order $\mathcal{L}$ (for the approximation we build) is of type ω. To ensure this, we will have requirements

$$\mathcal{N}_x : \quad x \text{ has only finitely many predecessors in } \mathcal{L}.$$

Each $\mathcal{N}_x$ acts by placing a finite restraint on A. If we know that x has only k predecessors at stage s in the construction of $\mathcal{L}$ above, then by making $A_t \upharpoonright (k+1) = A_s \upharpoonright (k+1)$ for all $t > s$, we can ensure that x never acquires any more $\prec$-predecessors, so that $\mathcal{N}_x$ is satisfied.

Our first result uses a simple combination of these requirements with Friedberg-Muchnik requirements, to build a set that is non-computable but order-computable. (Clearly every computable set is order-computable.)

Lemma 2.5. *There exists a non-computable order-computable c.e. set A.*

Proof. This is actually a special case of the later Lemma 2.12, but we give the proof anyway, as an introduction to this type of argument.

The construction of A and the derived order $\prec$ that order-computes A uses the $\mathcal{N}$-requirements defined above, along with Friedberg-Muchnik requirements to ensure $A >_T \emptyset$:

$$\mathcal{P}_e : A \neq \varphi_e.$$

$\mathcal{P}_e$ puts at most one element into A, and then stays satisfied forever.

$\mathcal{P}$-requirements always choose their witness elements $w_{e,s}$ larger than any number yet seen in the construction (hence larger than the total number of elements in the order $\prec$ up to that stage), thus implicitly respecting all higher-priority $\mathcal{N}$-requirements.

Let $A_0 = \emptyset$, with all $w_{e,0}$ undefined. At stage $s+1$, if some $\mathcal{P}_e$ with $e < s$ sees that $\varphi_{e,s}(w_{e,s}) \downarrow = 0$, it enumerates $w_{e,s}$ into A_{s+1}, and declares itself satisfied. Also, if $s+1$ is a stage at which the construction of the derived order for A (using the approximation built in this construction) makes an embedding, let x be the largest element of $\mathcal{L}_s$, and make the witness elements for all unsatisfied $\mathcal{P}_e$ with $e > x$ undefined. If there is no embedding at stage $s+1$, then for the least e for which $\mathcal{P}_e$ is not yet satisfied and $w_{e,s}$ is undefined, choose $w_{e,s+1}$ to be larger than any number yet seen. This completes the construction.

Now for any fixed $x \in \omega$, let s be a stage such that $x \in \mathcal{L}_s$ and no $\mathcal{P}_e$ with $e \leq x$ enumerates any element into A after stage s. Of course, x has only finitely many predecessors in $\mathcal{L}_s$. It may acquire finitely many more at a future stage $t+1$ under some embedding of $\mathcal{L}_t$, but if this happens, then all witness elements for unsatisfied requirements $\mathcal{P}_e$ with $e > x$ will be chosen large after stage $t+1$, hence larger than the number p of predecessors of x in $\mathcal{L}_{t+1}$. Thus $A_{t+1}\restriction p = A\restriction p$, and so no later embeddings can add any more predecessors of x to $\mathcal{L}$. Thus $\mathcal{N}_x$ is satisfied.

But then, for any $e \in \omega$, there is a stage s such that no $x \leq e$ acquires any new predecessors in $\mathcal{L}$ after stage s. So $\mathcal{P}_e$ will eventually have a permanent witness $w_e = \lim_t w_{e,t}$, and will enumerate w_e into A iff $\varphi_e(w_e) \downarrow = 0$. Thus $\mathcal{P}_e$ is satisfied as well. ∎

The set built by this construction can easily be forced to be low, by adding lowness requirements. In fact, though, we have a stronger result.

Theorem 2.6. *Every low c.e. set is order-computable.*

Proof. Let A be a low c.e. set. We use the characterization in [8], p. 229, that therefore there must exist a computable function f such that for every j, $W_j \cap \{n : D_n \subseteq \overline{A}\} = W_{f(j)} \cap \{n : D_n \subseteq \overline{A}\}$, and moreover if $W_j \cap \{n : D_n \subseteq \overline{A}\} = \emptyset$, then $W_{f(j)}$ must be finite. (Here D_n is the nth canonical finite set.)

Take any enumeration $\langle A_s \rangle$ of A. For each $n \in \omega$, we will enumerate a c.e. set $W_{g(n)}$, using the Recursion Theorem to assume that we know indices $g(n)$ in advance, uniformly in n.

Suppose that after stage s, the order $\prec$ is defined on exactly k elements, so that $\prec$ agrees with $A_{t_s}\restriction k$ (i.e. for all $j < k$, $j \in A_{t_s}$ iff the $(j+1)$-st element of $\prec$ at this stage is even). If $A_{t_s+1} = A_{t_s}$, then set $t_{s+1} = 1 + t_s$ and make no change to $\prec$. Otherwise, some (unique) element n entered A at stage $t_s + 1$, so we will need to displace the current $(n+1)$-st element e (which must be odd) of the order $\prec$. We search for the first available embedding of the order $\prec$ into some A_t with $t > t_s$, just as when we build the standard model. Since A may be assumed infinite and coinfinite, we find such a t, by Lemma 2.4. Suppose that e corresponds to the number $n' \in \overline{A}_t$ under this embedding. We pick the index m such that $D_m = \overline{A}_t \cap \{0, \ldots, n'\}$, and enumerate m into $W_{g(e)}$. We then run the enumeration of $W_{f(g(e))}$ and find the least stage t' such that either $m \in W_{f(g(e)),t'}$ or $D_m \cap A_{t'} \neq \emptyset$. If $D_m \cap A_{t'} \neq \emptyset$, we search for the next larger t with an embedding of $\prec$ into A_t and repeat the process for that t. Otherwise $m \in W_{f(g(e)),t'}$, and we set $t_{s+1} = t$, add elements to $\prec$ as dictated by the embedding of $\prec$ into A_t, and add extra elements at the end of $\prec$ (odd or even, corresponding to A_t) until $\prec$ has an odd number as its rightmost element. We declare this new order to be stage $s+1$ in the construction of $\prec$, and note that it agrees with (the appropriate initial segment of) $A_{t_{s+1}}$. This completes the construction.

We claim that at each stage, this procedure must finally terminate with some t. Indeed, there exists an embedding of $\prec$ into A, which we will eventually find as an embedding of $\prec$ into A_t for some t. Clearly the m chosen at this stage will have $D_m \cap \overline{A} = \emptyset$, so by our choice of f, this m must eventually appear in $W_{f(g(e))}$.

Now the ordering computed by $\prec$ does have an initial segment of type ω that matches A, as in all such constructions. We must show that this initial segment actually contains every element of the order. To see this, take any odd element e of $\prec$. We will show that e has only finitely many predecessors in $\prec$. Each embedding adds only finitely many predecessors. If we made infinitely many embeddings in which predecessors were added before e, then $W_{g(e)}$ and $W_{f(g(e))}$ would both be infinite, with every $m \in W_{g(e)}$ satisfying $D_m \cap A \neq \emptyset$. By our conditions on f, this is impossible. Therefore every such e has only finitely many predecessors, and since at every stage $\prec$ ends with an odd number, each even number in $\prec$ can have only finitely many predecessors. Thus $(\omega, \prec) \cong (\omega, <)$, and $\prec$ order-computes A. ∎

We conjecture that there is no uniform proof for Theorem 2.6. That is, there is no computable function h such that for every e, if W_x is low, then

$h(x)$ is the index of a linear order of type ω that order-computes W_x. This would not contradict the construction above, since the index of the function f used in the construction need not be computable from the index of the low c.e. set A. (In fact, if Theorem 2.6 cannot be uniformized, then the index of f could not be computable from the index of A, for that very reason.)

The lowness in Theorem 2.6 can also be avoided. We recall the *difference hierarchy*. A set W is ω-c.e. if there exists a computable function g and a computable approximation $\langle W_s \rangle$ to W such that for all $x \in \omega$,

$$g(x) \geq |\{s : W_{s+1}(x) \neq W_s(x)\}|.$$

If g can be taken to be the constant function n (for $n \in \omega$) and W_0 can be taken to be $\emptyset$, then we say that W is n-c.e. (For further details, see III.3.8 of [8].)

Theorem 2.7. *For every $\alpha \leq \omega$ and every α-c.e. set W, there exists an order-computable α-c.e. set $A \equiv_T W$.*

Corollary 3.17 will show that this result does not extend to $\alpha > \omega$.

Proof. First we define the E-extension of a finite binary string. Fix $\sigma \in 2^k$, for any $k \in \omega$. Write $\sigma_0 = \sigma$ and let $\sigma_1, \sigma_2, \ldots, \sigma_{2^k}$ be a listing of all strings in 2^k, in the dictionary order. We define the E-extension of σ by:

$$\sigma_E(x) = \sigma_i(j), \quad \text{where } x = j + ik,\ j < k, i \leq 2^k.$$

In short, σ_E has initial segment σ, with every possible string of length k appended to it. (Notice that except for the first k bits, σ_E is actually the same for every string $\sigma \in 2^k$.) For future reference, let h be the function

$$h(k) = k \cdot (2^k + 1).$$

Thus, if $\mathrm{lh}(\sigma) = k$, then $\mathrm{lh}(\sigma_E) = h(k)$.

We give details for the proof of the lemma when $\alpha = 1$, i.e. for c.e. sets. The key is the construction of the set A_0, the first element of the computable approximation $\langle A_s \rangle$ to A. We define $a_0 = 0$, and proceed by recursion. Given $\sigma = A_0 \upharpoonright a_x$, set $A_0(a_x) = 0$, and then extend A_0 so that it has initial segment $\tau = (\cdots(\sigma \hat{}\, 0)_E)_E)\cdots)_E$, where we take the E-extension exactly $(x+1)$ times. Then define a_{x+1} to be the length of τ.

Clearly the sequence $\langle a_x \rangle$ increases extremely fast. It would be possible to reduce the rate of increase, but our definition makes the following lemma immediate:

Lemma 2.8. *For every $\sigma \in 2^{<\omega}$, there exists a monotonic function f from k into $lh(\sigma_E)$ with $f(0) \geq k$ and $\sigma(i) = \sigma_E(f(i))$ for all $i < k$.* ∎

This will give us the embeddings we need within A whenever a new element appears in W.

Having built A_0, we continue with A_{s+1}. If $W_{s+1} = W_s$, then $A_{s+1} = A_s$. Otherwise, let x be the unique (WLOG) element of $W_{s+1} - W_s$, and let $A_{s+1} = A_s \cup \{a_x\}$. Thus A is c.e., and indeed W is 1-reducible to A via the function $x \mapsto a_x$. On the other hand, $A \leq_T W$: we know $a_x \in A$ iff $x \in W$, and elements not in the computable sequence $\langle a_x \rangle_{x \in \omega}$ belong to A iff they belong to the computable set A_0.

To see that this A is order-computable, we build an order $\prec$ as follows. Start with the empty order. At each stage $s+1$, we assume inductively that $\prec$ (on its domain of definition D_s, which contains j elements, say) matches the initial segment $A_s \restriction j$. If $A_{s+1} = A_s$, we do nothing. Otherwise, let a_x be the new element of A_{s+1}. If $a_x \geq j$, then simply add new elements to D_{s+1}, to the right of all elements of D_s, until $\prec$ on D_{s+1} matches $A_{s+1} \restriction a_x$. If $j > a_x$, then add a new even number y to $\prec$ with exactly a_x predecessors (under $\prec$) in D_s. Write σ for the binary string generated by D_s under $\prec$, and

$$\tau = (\sigma \restriction a_x)^\frown 1^\frown (\sigma \swarrow a_x)$$

where $\sigma \swarrow a_x$ represents σ with its initial segment $\sigma \restriction a_x$ chopped off:

$$(\sigma \swarrow a_x)(n) = \sigma(n + a_x).$$

Thus, τ is the string corresponding to $D_s \cup \{y\}$ under $\prec$.

By Lemma 2.8, τ embeds into A_{s+1}. Find the shortest such embedding f, i.e. that one minimizing $f(i)$ for every $i < \mathrm{lh}(\tau)$. Since f is an embedding, we can now add new elements to D_{s+1} so that D_{s+1} contains exactly $f(\mathrm{lh}(\tau)-1)$ elements and matches the initial segment of A_{s+1} of that length.

It remains to show that $(\omega, \prec) \cong (\omega, <)$. To see this, we claim that if a number $y \in D_s - D_{s-1}$ has $< a_x$ predecessors under $\prec$ in D_s, then y has $< a_{x+1}$ predecessors under $\prec$ in ω. The first time y acquires new predecessors, resulting from a change to $W \restriction x$, it must wind up with no more than $h(a_x)$ predecessors, by Lemma 2.8. (Recall that for $\sigma \in 2^k$, $h(k) = \mathrm{lh}(\sigma_E)$.) A subsequent change to $W \restriction (x+1)$ could add more predecessors, but y would still have at most $h(h(a_x))$ predecessors. At most $(x+1)$ changes to $W \restriction (x+1)$ can ever occur, since W is c.e., and these changes would leave y with at most

$h^{x+1}(a_x)$ predecessors. However, $h^{x+1}(a_x) < a_{x+1}$ by our construction of the sequence $\langle a_x \rangle$, and so the entrance of any number $> x+1$ into W will not add any predecessors to y. This proves the claim, and the lemma for c.e. sets.

It is clear how to handle the case $\alpha > 1$: instead of iterating the E-extension operation $(x+1)$ times when extending $A_0 \restriction a_x$ in the definition of A_0, one extends it as many times as $A\restriction(x+1)$ could change. For $\alpha \in \omega$, this is just $\alpha \cdot (x+1)$ times; for an ω-c.e. set with bound g (as on p. 9), it would be $\sum_{t \leq x} g(t)$ times. If x appears in the set W at stage $s+1$, we add a_x to A_{s+1}, just as before; but also now, if x leaves W at stage $s+1$, we remove a_x from A_{s+1}. Thus the limit A will also be α-c.e., and the rest of the proof goes through just as before. ∎

The proof shows a somewhat stronger result, in fact, since the reduction from W to A was a 1-reduction, not just a Turing reduction. Indeed, in many cases we have 1-equivalence:

Corollary 2.9. *For every set W that is c.e. but not simple, there exists an order-computable c.e. set $A \equiv_1 W$.*

Proof. We may assume that W itself is not order-computable, hence not computable. Since W is not simple, $\overline{W}$ contains an infinite c.e. set, which in turn contains an infinite computable set Y. Also, W itself contains an infinite computable set X. Moreover, X is coinfinite in W and Y is coinfinite in $\overline{W}$, because W is noncomputable. We now tweak the function $f(x) = a_x$ to get a computable permutation g of ω with $g(W) = A$, with a_x and A as defined in the proof of Theorem 2.7. Define computable bijections $h_X : X \to (A_0 - f(\overline{X \cup Y}))$ and $h_Y : Y \to (\overline{A_0} - f(\overline{X \cup Y}))$. (Since f is increasing, these are all infinite computable sets.) Then define g by:

$$g(n) = \begin{cases} f(n), & \text{if } n \notin X \cup Y \\ h_X(n) & \text{if } n \in X \\ h_Y(n) & \text{if } n \in Y \end{cases}$$

This is the desired computable permutation of ω mapping W onto A. ∎

On the other hand, while order-computable sets must be Δ^0_2, they need not be ω-c.e.

Lemma 2.10. *There exists an order-computable set that is not ω-c.e.*

Proof. We build an order-computable set A satisfying requirements:

$$\mathcal{R}_{\langle e,i\rangle}: \quad \text{If } \varphi_e \text{ and } \varphi_i \text{ are total, then } (\exists w)[A(w) \neq \lim_s \varphi_i(w,s) \text{ or } |\{s : \varphi_i(w,s+1) \neq \varphi_i(w,s)\}| > \varphi_e(w)]$$

Thus, if φ_i gives a computable approximation to the set A, then that approximation cannot have φ_e as computable bound on the number of changes of this approximation. If we can make this hold for all e and i, then A will not be ω-c.e. (Of course, we must also make A order-computable.) Here we only sketch the method for satisfying these requirements, since this lemma is a special case of the next lemma.

Our strategy is to choose a witness w_j (with $j = \langle e,i\rangle$) and wait for $\varphi_e(w_j)$ to converge. When it does, we know how many times we will have to change our approximation $A_s(w_j)$, so we redefine all w_k $(k > j)$ to be sufficiently much larger than w_j as to allow all those changes to A to take place without ruining the order-computability of A. We may put numbers between w_j and w_{j+1} into A and remove them without harming any $\mathcal{R}$-requirements. At the start none of them is in A, and we move them in and out as necessary to create embeddings for the construction of the standard order corresponding to our approximation. Each time $\varphi_i(w_j, s+1)\downarrow \neq \varphi_i(w_j,s)$, we set $A(w_j) \neq \varphi_i(w_j, s+1)$, and change A between w_j and w_{j+1} as needed to ensure order-computability. This continues until the approximation $\varphi_i(w_j,s)$ has changed $\varphi_e(w_j)+1$ times; then we have satisfied $\mathcal{R}_j$, and we leave $A\restriction w_{j+1}$ fixed forever after. ∎

The question then arises whether there exists an order-computable set whose Turing degree contains no ω-c.e. set. We do manage to build such a set, so one conjecture about order-computable sets (that their Turing degrees might be precisely the ω-c.e. degrees) falls by the wayside.

Lemma 2.11. *There exists an order-computable set A whose Turing degree contains no ω-c.e. set.*

Proof. The idea is to satisfy the requirements

$$\mathcal{R}_{\langle e,i,j,k\rangle}: \quad \text{If } \varphi_e \text{ is total and } B = \Phi_j^A \text{ and } A = \Phi_k^B, \text{ then there exists } w \text{ s.t. } [B(w) \neq \lim_s \varphi_i(w,s) \text{ or } |\{s : \varphi_i(w,s+1) \neq \varphi_i(w,s)\}| > \varphi_e(w)]$$

Write $C_s(w) = \varphi_i(w, s)$ for all s and w. (We do not know for sure whether C_s is an approximation to B or not; if it is not, then the procedure below will eventually terminate.)

We choose a witness x_n, where $n = \langle e, i, j, k\rangle$, and wait until $\Phi_k^B(x_n)\downarrow$ with some use u_n and $\varphi_e(w)\downarrow$ for every $w \leq u_n$. Set $m_n = 1 + u_n \cdot \max_{w<u_n} \varphi_e(w)$. We now move x_n into A, thereby forcing a change on $B\upharpoonright u_n$. When $\Phi_k^B(x_n) = A(x_n)$ again and the approximation $C_s \upharpoonright u_n$ matches $B \upharpoonright u_n$, we restore $A\upharpoonright \mathrm{use}(\Phi_j^A(u_n))$ to its state before x_n entered A, which forces $B\upharpoonright u_n$ to return to its state at that time. Eventually $C_s\upharpoonright u_n = B\upharpoonright u_n$ again, meaning that we have forced two C-changes for at least one $w < u_n$. We repeat this entire process m_n times (actually $\frac{m_n+1}{2}$ times would suffice), thereby guaranteeing that some $w < u_n$ has changed at least $\varphi_e(w) + 1$ times, and that $C_s(w)$ has matched every such change. Thus the function φ_e does not show B to be ω-c.e.

Once we have permanently chosen x_n, we wait for u_n and $\varphi_e \upharpoonright u_n$ and $\Phi_j^A(u_n)$ to converge. Once they do, we know the maximum number of times we will have to force changes to $B\upharpoonright u_n$, and we know how much of A we will have to restore each time. Therefore, we know how much of A will have to be left open in order for the necessary embeddings to take place for us to build a standard order (from our approximation $\langle A_s\rangle$) to order-compute A. At this time we choose $x_{n+1}, x_{n+2}, \ldots$ all to be large enough not to interfere with that space (since $\mathcal{R}_{n+1}$ will never change $A\upharpoonright x_{n+1}$). The requirements $\mathcal{R}_{n+1}, \ldots$ are injured every time we make A-changes on behalf of $\mathcal{R}_n$, and the procedure for $\mathcal{R}_{n+1}, \ldots$ must be restarted each such time. ∎

We can also find a noncomputable Turing degree that contains only order-computable sets. Indeed, our result is stronger:

Lemma 2.12. *There exists a noncomputable c.e. set A such that every set Turing-computable in A is order-computable.*

Corollary 2.13. *There exist infinitely many Turing degrees* $\mathbf{d}$ *such that every set of degree* $\mathbf{d}$ *is order-computable.* ∎

Proof of Lemma 2.12. The requirements for constructing A are simple. If Φ_e^A computes a set B_e, we will use

$$B_{e,s}(x) = \begin{cases} \Phi_{e,s}^{A_s}(x), & \text{if } \Phi_{e,s}^{A_s}(y)\downarrow \in \{0,1\} \text{ for all } y \leq x \\ 0, & \text{if not.} \end{cases}$$

as a Δ^0_2 approximation to B_e, and build the derived ordering $\prec_e$ corresponding to B_e. (Notice that our approximations are defined even if Φ^A_e does not compute any set.) We then satisfy:

$\mathcal{P}_e$: If φ_e is total, then $\varphi_e \neq A$.

$\mathcal{N}_{\langle e,k\rangle}$: If $\prec_e$ is defined on k at some stage, then k has only finitely many $\prec_e$-predecessors.

The construction for the $\mathcal{P}$-requirements is standard, once we have defined the restraints for the $\mathcal{N}$-requirements. At stage s, if k has exactly p predecessors in the order $\prec_e$, we set

$$r(e,k,s) = \max_{x \leq p+1} \text{use}(\Phi^{A_s}_{e,s}(x))$$

where the use of a divergent computation is 0 by definition. (The computation must converge for all such x at the first stage s at which the order $\prec_e$ includes k, but might diverge for some such x later if $\mathcal{P}$-requirements injure $\mathcal{N}_{\langle e,k\rangle}$.) If $\prec_e$ does not yet include k, then $r(e,k,s) = 0$.

For every $i \geq \langle e,k\rangle$ and every stage s, the witness element $w_{i,s}$ will be $> r(e,k,s)$. The requirement $\mathcal{P}_i$ waits for $\varphi_{e,s}(w_{i,s})$ to converge to 0, then enumerates $w_{i,s}$ into A, possibly injuring lower-priority $\mathcal{N}$-requirements, but not higher-priority ones. This completes the construction.

Once we reach a stage after which no higher-priority $\mathcal{P}$-requirement ever again injures $\mathcal{N}_{\langle e,k\rangle}$, we wait until k appears in the order $\prec_e$ at some stage s. Once it does (with exactly x predecessors, say), the restraint $r(e,k,s)$ on A forces $B_t\restriction(x+2) = B_s\restriction(x+2)$ for all $t \geq s$. Hence the element k never again acquires another predecessor in the order $\prec_e$, and $r(e,k,t) = r(e,k,s)$ for all $t \geq s$. In turn, this allows the next $\mathcal{P}$-requirement to choose a witness element that will never again be changed, and will enter A if necessary to satisfy P. Thus all requirements are ultimately satisfied.

If $B \leq_T A$, then we have $B = \Phi^A_e$ for some e, and the $\langle B_{e,s}\rangle$ defined above is a computable approximation to B. If B is infinite, then the order $\prec_e$ is the derived order for this approximation. Since every $\mathcal{N}_{\langle e,k\rangle}$ is satisfied, we know $(\omega, \prec_e) \cong (\omega, <)$, and thus B is order-computable. (Clearly every finite B is order-computable as well.) ∎

3 Non-Order-Computable Sets

An order-computable set must not only be Δ^0_2, but must also be the range of a limitwise-monotonic function. Indeed, if $(\omega, \prec, R) \cong (\omega, <, A)$, then A is precisely the set $\{|\{y \prec x\}| : x \in R\}$, and if we enumerate R as $\{x_0 < x_1 < \cdots\}$, then the limit as $s \to \infty$ of the (computable, monotonic) predecessor approximation function $f(k, s) = |\{y \leq s : y \prec x_k\}|$ has range A.

Limitwise-monotonic functions are also called subcomputable or Σ^0_1; a simple definition is that the function must be total and the set of points below its graph must be a c.e. set. The range of such a function is always a Σ^0_2 set, but can fail to be Δ^0_2, so clearly not all such ranges are order-computable.

In [5], Khoussainov, Nies, and Shore proved that there exist Δ^0_2 sets that are not the range of any limitwise-monotonic function (though this result appears to have been proved earlier by Khisamiev [4]). These are the first examples of Δ^0_2 sets that are not order-computable. We can imitate their proof and include lowness requirements, thereby showing that low sets can fail to be order-computable.

Lemma 3.1. *There exists a low set A that is not the range of any limitwise-monotonic function.*

Corollary 3.2. *There exist low sets that are not order-computable.* ∎

This set A cannot itself be made c.e., of course. It is an open question, suggested by the referee of this paper, whether one could build such an A which is Turing-reducible to a low c.e. set.

Proof of Lemma 3.1. We imitate the finite-injury proof in [5], with its requirements

$$\mathcal{R}_e: \quad \text{If } f_e(x) = \lim_{t\to\infty} \varphi_e(x,t) \text{ is total and } \varphi_e \text{ is monotonic,} \\ \text{then range}(f_e) \neq A,$$

which are satisfied by fixing a witness x_e and moving each new $\varphi_e(x_e, t)$ out of A. We add requirements

$$\mathcal{L}_{i,x}: \quad [(\exists^\infty s)\Phi^{A_s}_{i,s}(x)\downarrow] \implies \Phi^A_i(x)\downarrow$$

to ensure that A is low. The idea is that each $\mathcal{L}_{i,x}$ attempts to protect a finite initial segment of A so as to satisfy itself. Each higher-priority $\mathcal{R}_e$ either acts only finitely often, keeping a particular $\varphi_e(x_e, t)$ out of A forever (in which case eventually it no longer interferes with $\mathcal{L}_{i,x}$), or else drives $\varphi_e(x_e, t)$ higher and higher as t increases. In this latter case, for each y there is a stage s so large that $\mathcal{R}_e$ sets no conditions on $A \upharpoonright y$ after stage s, and so, for any string σ such that $\Phi_i^\sigma(x)$ converges, this $\mathcal{R}_e$ will eventually allow $A \upharpoonright \mathrm{lh}(\sigma)$ to be set to σ, leaving $\mathcal{L}_{i,x}$ satisfied at all subsequent stages, so that the restraint function $l(\langle i, x\rangle, s)$ for $\mathcal{L}_{i,x}$ will converge and will not injure any lower-priority requirements from then on.

Define distinct witness elements $m_{e,0}$ for each $e \in \omega$, which we will use to ensure that if $\varphi_e(\cdot, t)$ does converge monotonically to a total function f_e as $t \to \infty$, then A is not the range of f_e. We let $A_0 = \{m_{e,0} : e \in \omega\}$, make all $x_{e,0}$ and $t_{e,0}$ undefined, and set all $l(e, 0) = 0$.

At stage $s+1$, we have a substage e for each $e = \langle i, x\rangle \leq s$, starting with $e = 0$. First, to avoid injuring any higher-priority $\mathcal{R}_j$ or $\mathcal{L}_k$, we search for the shortest $\sigma \in 2^{<s}$ such that:

- $\Phi_{i,s}^\sigma(x)\!\downarrow$; and
- for all $j < e$ such that $x_{j,s}\!\downarrow < \mathrm{lh}(\sigma)$, $\sigma(\varphi_j(x_{j,s}, t_{j,s})) = 0$; and
- for all $j < e$ such that $x_{j,s}\!\uparrow$ and $m_{j,s} < \mathrm{lh}(\sigma)$, $\sigma(m_{j,s}) = 1$; and
- for all $k < e$, $\sigma \upharpoonright l(k, s+1) = A_{s+1} \upharpoonright l(k, s+1)$.

If there is no such σ, then $l(e, s+1) = 0$ and we make no change to A_{s+1}. Otherwise, we define $A_{s+1} \upharpoonright \mathrm{lh}(\sigma) = \sigma$ and $l(e, s+1) = \mathrm{lh}(\sigma)$. (The final condition above ensures that this does not contradict any definition of A_{s+1} at any previous substage of stage $s+1$.) If there are several σ of minimal length satisfying these conditions, use the first in the dictionary order.

We then continue this substage by attending to $\mathcal{R}_e$ as follows:

1. If there exists $j < e$ such that either
   - $l(j, s+1) \geq m_{e,s}$; or
   - $\varphi_{j,s}(x_{j,s+1}, t_{j,s+1})\!\downarrow = \varphi_{e,s}(x_{e,s}, t_{e,s})\!\downarrow$; or
   - $x_{e,s}$ is undefined and $\varphi_{j,s}(x_{j,s+1}, t_{j,s+1})\!\downarrow = m_{e,s}$,

then all $\mathcal{R}_k$ with $k \geq e$ are injured at stage $s+1$. We make all $x_{k,s+1}$ and $t_{k,s+1}$ undefined, redefine each $m_{k,s+1}$ to be the least number larger than $\max_{n<k} l(n, s+1)$ that is not in $\{m_{n,s+1} : n < k\} \cup \{\varphi_n(x_{n,s+1}, t_{n,s+1}) : n < k \ \& \ x_{n,s+1}\downarrow\}$, and end the stage.

2. If there is no such j, but $x_{e,s}$ is defined, then so is $t_{e,s}$, and we ask if $\varphi_{e,s}(x_{e,s}, t_{e,s}+1) \downarrow \geq \varphi_e(x_{e,s}, t_{e,s})$. If not, then $\mathcal{R}_e$ makes no changes at all. If so, then $\mathcal{R}_e$ removes $\varphi_{e,s}(x_{e,s}, t_{e,s}+1)$ from A_{s+1}, and sets $t_{e,s+1} = t_{e,s} + 1$.

3. Otherwise $x_{e,s}$ is undefined. We leave $m_{e,s+1} = m_{e,s}$ and ask if there exists a pair $\langle x, t\rangle \leq s$ such that $\varphi_{e,s}(x,t)\downarrow = m_{e,s}$. If so, then $\mathcal{R}_e$ chooses the elements of the least such pair to be $x_{e,s+1}$ and $t_{e,s+1}$, respectively, and removes $m_{e,s+1}$ from A. If not, we do nothing.

As long as $e < s$ and Case 1 did not apply, we now continue with the next substage $e+1$. If $e = s$ or Case 1 applied, this concludes stage $s+1$.

The proof that the requirements $\mathcal{R}_e$ are satisfied proceeds just as in [5], once we have proven that each restraint built by an $\mathcal{L}$-requirement is bounded. Assume by induction that for each $j < e$, $\lim_s l(j,s)\downarrow < \omega$ and that for each $j \leq e$, $\mathcal{R}_j$ is satisfied. Let s_0 be a stage so large that of all the requirements $\mathcal{R}_j$ with $j \leq e$, the only ones that make any changes to A after stage s_0 are those that make infinitely many such changes, so that $\lim_s \varphi_j(x_{j,s}, t_{j,s}) = \infty$. (Call these requirements $\mathcal{R}_{j_1}, \ldots, \mathcal{R}_{j_n}$.) If there exists a stage $s' > s_0$ at which $l = l(e, s') > 0$, then only these $\mathcal{R}_{j_k}$ can change $A\restriction l$ after stage s'. However, there exists another stage $s_1 > s'$ such that for all $s \geq s_1$ and all $k \leq n$, $\varphi_{j_k}(x_{j_k,s}, t_{j_k,s}) > l$. But then at all stages $s > s_1$, the requirement $\mathcal{L}_e$ is allowed to set $A_s\restriction l = A_{s'}\restriction l$. It does so (unless there is a shorter string it can use for the same purpose). Hence $l(e,s) \leq l$ for cofinitely many s, and $\Phi_e^A(x)\downarrow$, as required by $\mathcal{L}_e$. Also, $\lim_s l(e,s) \leq m_{e+1,s_1+1}$, allowing the induction to proceed on $\mathcal{R}_{e+1}$. ∎

The next lemma gives a straightforward proof for a weak version of Corollary 3.18 below.

Lemma 3.3. *There exists a limitwise-monotonic function g whose range is Δ_2^0, yet is not order-computable. Indeed, we can build one such function g_1 with low range, and another such function g_2 whose range has degree $\mathbf{0}'$.*

Corollary 3.4. *The property of order-computability does not respect Turing equivalence.*

Proof of Corollary. Theorem 2.7 and Lemma 3.3 provide two sets of Turing degree $\mathbf{0}'$, one order-computable and the other not. ∎

Proof of Lemma. Our strategy for building $g = \lim_s h(\cdot, s)$ is to satisfy:

$\mathcal{C}_m$: $(\forall s)\ |\{0, \ldots, 3m-1\} \cap \text{range}(h(\cdot, s))| \leq 2m$.

$\mathcal{R}_e$: If $\varphi_e(x, s)$ is a predecessor approximation function for an order-computable infinite set, then $\text{range}(\lim_s \varphi_e(\cdot, s)) \cap \text{range}(g) \neq \emptyset$.

We write $f_e = \lim_s \varphi_e(\cdot, s)$ and exploit the fact that the set $A = \text{range}(g)$ is order-computable iff its complement $\overline{A}$ is. (If $(\omega, \prec, R)$ is computable, so is $(\omega, \prec, \overline{R})$.) The requirements $\mathcal{C}_m$ make A coinfinite. Therefore, if A were order-computable, then $\overline{A}$ would be infinite and have a computable predecessor approximation function φ_e such that $\text{range}(f_e) = \overline{A}$, contradicting $\mathcal{R}_e$.

To satisfy $\mathcal{R}_e$, we would like to pick a single witness w_e and define $h(w_e, s) = \varphi_e(w_e, t)$ for the greatest t such that $\varphi_{e,s}(w_e, t)\downarrow$. (If $\varphi_e(w_e, 0)\uparrow$, or if $\varphi_e(w_e, t+1) < \varphi_e(w_e, t)$ for some t, then clearly $\mathcal{R}_e$ is satisfied and we leave $h(w_e, s)$ constant for all subsequent s, so as to guarantee that $g = \lim_s h(\cdot, s)$ is limitwise monotonic.) The danger, of course, is that possibly $\lim_t \varphi_e(w_e, t) = \omega$. This would disrupt our strategy, since we must ensure that $\lim_t h(w_e, t) < \omega$. To deal with this problem, we choose a single element w_e, compute $\varphi_e(w_e, t)$ for each t, and assign infinitely many elements $w_{e,k}$ to follow m_e by setting $h(w_{e,k}, s) = \varphi_e(w_e, t)$ as above. For each k and all t, we will ensure that $h(w_{e,k}, t) \leq d_k$, for fixed finite numbers d_k such that $d_1 < d_2 < \cdots$. Thus we will be able to satisfy $\mathcal{R}_e$ if $f_e(w_e)$ turns out to be finite, yet will not have trouble if $f_e(w_e)$ diverges.

For each e for which the hypotheses of $\mathcal{R}_e$ are satisfied, f_e must have infinite range. Therefore, we can wait for arbitrarily high values of $\varphi_e(x, t)$ to appear, and then choose w_e to be an x for which $\varphi_e(x, t)$ is sufficiently large. This allows us to satisfy all $\mathcal{C}_m$ and also any higher-priority finitary requirements such as the $\mathcal{L}$-requirements below, and to guarantee that $\text{range}(g)$ will be Δ^0_2 (since the functions $h(\cdot, t)$ will only move a given element into or out of $\text{range}(g)$ finitely often as $t \to \infty$). Thus satisfying our requirements will indeed prove the lemma.

The elements d_k mentioned above are "dumps" which we decide beforehand will lie in range(g). Specifically, we choose $d_k = 3k$, and assign elements $c_k \in \omega$ such that $h(c_k, t) = d_k$ for every k and t. (This does not interfere with any requirement, since when we need to satisfy an $\mathcal{R}_e$, we do so by putting a number into range(g), never by keeping a number out of range(g).) Whenever we want to remove an element $h(x, s)$ from range(g), we can always find a $d_k > h(x, s)$, and define $h(x, s + 1) = d_k$, thereby removing $h(x, s)$ from range(g) for the present without introducing any new elements to range(g). Thus we can ensure that each $\mathcal{R}_e$ adds only one new element to range(g), so that range(g) is coinfinite.

Our satisfaction of the different requirements $\mathcal{R}_e$ actually creates no priority conflicts, since each such requirement wants only to put certain elements into range(g), not to keep any specific elements out of it. The only negative requirements arise from making range(g) coinfinite and Δ^0_2, and we resolve these conflicts by making each $\mathcal{R}_e$ consider only sufficiently large numbers from range(f_e).

To make g_1 have low range, we simply add in the lowness requirements as in the proof of Lemma 3.1; once again, the strategy used there works. Each $\mathcal{L}_e$ looks for any σ it can find to guarantee convergence for its requirement; the only restraints on its search are that σ must respect the (finitary) restraints of higher-priority $\mathcal{L}_i$ and $\mathcal{R}_i$, must have $\sigma(d_k) = 1$ for all k, and must satisfy the coinfiniteness requirements $\mathcal{C}_m$.

To code the Σ^0_1-complete set K into the range of g_2, on the other hand, we use the dumps: whenever we want to put an element into a dump, we do not just add it to the next higher dump, but rather to the dump d_k, where k is the first element of K we find such that d_k is larger than the current value of the element. Also, rather than guaranteeing that every d_k lies in A, we wait for numbers n to enter K_s. Each time some $n \in K_{s+1} - K_s$, we choose a fresh element x not yet in the domain of $h(\cdot, s)$ and define $h(x, s') = d_n$ for all $s' \geq s$. Thus $K \leq_1 A$, via the function that takes x to d_x. Since $A = \text{range}(g_2)$ is Δ^0_2, this guarantees $K \equiv_T \text{range}(g_2)$. ∎

The predecessor approximation function for A has an important additional property. Since $(\omega, \prec)$ is a linear order, we know that if $x_i \prec x_j$, then every new predecessor of x_i that appears must also precede x_j. Stating this in terms of f_A yields:

$$(\forall i, j, s)\,[f_A(i, s) < f_A(j, s) \implies f_A(i, s{+}1) - f_A(i, s) \leq f_A(j, s{+}1) - f_A(j, s)].$$

We refer to this as the *order property* for f_A (or for $g_A = \lim_s f_A(\cdot, s)$, by abuse of terminology).

Moreover, if f is any monotonic computable function with the order property and $g = \lim_s f$ is total, then $g(E)$ is order-computable. This yields a characterization of order-computability, but hardly a satisfactory one.

Above we constructed a low set that was not order-computable. We now investigate how close such a set can come to being c.e. Recalling the Ershov hierarchy as defined on p. 9, we remind the reader that 2-c.e. sets are also called d.c.e., since they can be expressed as the set-theoretic difference $W_1 - W_2$ of two c.e. sets W_1 and W_2.

Theorem 3.5. *There exist a c.e. set W and a low d.c.e. set A that are not order-computable.*

Proof. We give the construction of the set A. To build W, one simply runs the same construction without the lowness requirements. Since these are the only requirements that ever remove an element from A, it is clear that the set W will indeed be c.e., although not low. The contrast gives a fair insight into the impossibility of building a low c.e. set that is not order-computable (cf. Theorem 2.6).

Our strategy is to build an approximation to a set A satisfying the standard lowness requirements:

$$\mathcal{L}_{\langle e,x\rangle} : \ [(\exists^\infty s)\Phi_{e,s}^{A_s}(x)\downarrow] \implies \Phi_e^A(x)\downarrow$$

while also ensuring that if φ_e appears to be a predecessor approximation function for A, then in fact some element of the corresponding linear order must have infinitely many predecessors:

$$\mathcal{R}_e : \varphi_e \text{ is not a predecessor approximation function for } A.$$

The basic module for satisfying a single $\mathcal{R}_e$ checks at every stage to see whether φ_e has proven itself not to be a predecessor approximation function (as described after Definition 2.3). As long as it has not done so, we pick a witness element w_e to put into A. We choose w_e large enough to have an element of $\overline{A}$ below it, which we use as a "trigger," keeping it out of A for the present. If φ_e responds by matching A up to w_e at some stage s (i.e. putting odd numbers in positions corresponding to $\overline{A}_s$ and even numbers in positions corresponding to A_s), then we define n_e to be the even number in position

w_e, and pull the trigger, by enumerating the trigger element into A_{s+1} (and naming it $t_{e,s+1}$). In order to match this change, φ_e must add an even number to its order in that position, so that n_e acquires a new predecessor, and then must acquire enough more predecessors to correspond to a larger element of A. If there are only finitely many such stages s, then clearly $\mathcal{R}_e$ is satisfied. Otherwise, we wait for a future stage s' at which φ_e matches the new $A_{s'}$ up to $\varphi_e(n_e, s')$ (thus, on a longer initial segment than at stage s). When this happens, we enumerate a new trigger $t_{e,s'+1} < \varphi_e(n_e, s')$ from $\overline{A}_{s'}$ into A, and start the process over. Ultimately this will add infinitely many predecessors of n_e to the order, satisfying $\mathcal{R}_e$.

To make this basic module work, of course, we must ensure that elements of A are far enough apart that triggers will always be available. The interplay between distinct $\mathcal{R}$-requirements is therefore limited to choosing the initial elements $w_{e,0}$ far enough apart. This is a straightforward combinatorial exercise, described in Lemma 3.6. ($\mathcal{L}$-requirements will never be allowed to put elements into A, so they cannot mess up the combinatorics.)

This basic module never puts any number into A more than once. Moreover, once $\varphi_e(n_e, s)$ has increased, the t_e and the w_e that we had put into A can be removed from A if necessary to satisfy a lowness requirement. Thus, if $w_e \to \infty$, we will satisfy each lower priority $\mathcal{L}_{i,x}$-requirement by waiting until $\varphi_e(n_e, s)$ is so large that we can clean up behind it and build an initial segment of A of length $< \varphi_e(n_e, s)$ that makes $\Phi_i^A(x)$ converge.

We start with $A_0 = \emptyset$ and all variables undefined. At stage $s+1$, we have one substage for each $i = \langle e, x\rangle \leq s$ in turn, starting with 0. At substage i, we first search for the shortest $\sigma \in 2^{<s}$ (if any; otherwise choose the empty string) such that:

- $\Phi_{e,s}^{\sigma}(x)\downarrow$; and
- $\sigma\restriction \max_{j<i} l(j, s+1) = A_s\restriction \max_{j<i} l(j, s+1)$; and
- $w_{i,s}$ is defined and $\sigma\restriction w_{i,s} = A_s\restriction w_{i,s}$; and
- for all z such that $\sigma(z) = 1$, $z \in A_s$; and
- for all $z < \mathrm{lh}(\sigma)$ such that $z = t_{j,s}$ for some $j < i$, $\sigma(z) = A_s(z)$.

We set $A_{s+1}\restriction \mathrm{lh}(\sigma) = \sigma$, and set

$$l(i, s+1) = \max_{t \leq s+1} \mathrm{use}(\Phi_{e,t}^{A_t}(x))$$

(where by definition the use of a divergent computation is 0). If we have a choice between two σ of equal length, we take the first in the dictionary order.

If $A_{s+1} \restriction \mathrm{lh}(\sigma) \neq A_s \restriction \mathrm{lh}(\sigma)$, then $\mathcal{L}_i$ has injured all $\mathcal{L}_k$ $(k > i)$ and all $\mathcal{R}_k$ $(k \geq i)$: every $l(k, s+1)$ with $k > i$, and every $n_{k,s+1}$, $w_{k,s+1}$, and $t_{k,s+1}$ with $k \geq i$, becomes undefined, and we end the stage s immediately. Otherwise, we continue the substage i, turning now to the requirement $\mathcal{R}_i$. If $\varphi_{i,s}$ has proven that it is not a predecessor approximation function, we end this substage and go on to substage $i+1$. Otherwise we work to satisfy $\mathcal{R}_i$ as follows:

1. If $w_{i,s}$ is undefined, we pick $w_{i,s+1}$ to equal 2^{i+1} plus the greatest number yet seen in the construction, and enumerate it into A_{s+1}. We also make all $n_{k,s+1}$ with $k \geq i$ and all $w_{k,s+1}$ and $t_{k,s+1}$ with $k > i$ undefined.

2. If $w_{i,s}$ is defined, but $n_{i,s}$ is not, we check whether $A_s \restriction (1 + w_{i,s})$ is an initial segment of the set approximated by $\varphi_{i,s}$. If not, we do nothing. If so, we set $n_{i,s+1}$ to be the n such that $\varphi_{i,s}(n, u_{i,s}) = w_{i,s}$, and let $w_{i,s+1} = w_{i,s}$. (Here $u_{i,s}$ is the greatest u such that $\varphi_{i,s}(x, u) \downarrow$ for all $x < u$.) We choose the trigger $t_{i,s+1}$ to be the greatest number $< w_{i,s}$ such that:

   - $t_{i,s+1} \notin \cup_{t \leq s} A_t$; and
   - no number $w_{j,t}$ with $t \leq s$ and $j \in \omega$ satisfies $t_{i,s+1} \leq w_{j,t} < w_{i,s}$;

   and enumerate $t_{i,s+1}$ into A_{s+1}. Notice that $n_{i,s+1}$ is uniquely defined (since $\varphi_{i,s}$ appears to be a predecessor approximation function) and must be even, since $w_{i,s} \in A_s$. We will prove in Lemma 3.6 below that such a $t_{i,s+1}$ must exist.

3. Otherwise $n_{i,s}$ is defined. With $u_{i,s}$ as above, we check whether $A_s \restriction (1 + \varphi_{i,s}(n_{i,s}, u_{i,s}))$ is an initial segment of the set approximated by $\varphi_{i,s}$. If not, we do nothing. If so, then keep $n_{i,s+1} = n_{i,s}$, redefine $t_{i,s+1}$ to be the greatest element $< \varphi_{i,s}(n_{i,s}, u_{i,s})$ satisfying

   - $t_{i,s+1} \notin \cup_{t \leq s} A_t$; and
   - no number $w_{j,t}$ with $t \leq s$ and $j \in \omega$ satisfies $t_{i,s+1} \leq w_{j,t} < \varphi_{i,s}(n_{i,s}, u_{i,s})$;

   and enumerate $t_{i,s+1}$ into A_{s+1}.

This completes substage i and the construction.

For each i, write $w_i = \lim_s w_{i,s}$. Also, let $\{v_0 < v_1 < \cdots\}$ be the set $\{w_{i,s} : i, s \in \omega\}$. (We will not have $w_i = v_i$ for all i, because of injuries to the $\mathcal{R}$-requirements.)

Lemma 3.6. *At every stage s and substage i of this construction, if substep 2 or 3 requires a trigger $t_{i,s+1}$ to be defined and enumerated into A, then there exists a trigger satisfying the two conditions named in that substep.*

Proof. This is a combinatorial argument. First, notice that the only elements to enter A are the various $w_{j,s}$ and the triggers $t_{k,s}$, which enter A only if some greater element is already in A.

Choose the least j such that $\varphi_i(n_{i,s}, s) \leq v_j$. Now v_j was chosen (as w_{m,s_0} for some $m \leq j$, at some stage s_0) to equal 2^{m+1} plus the greatest element yet seen in the construction, at which time the set

$$T = \{t \notin A_{s_0} : v_{j-1} < t < v_j\}$$

of available triggers contained at least $2^{m+1} - 1$ elements. The only way for any of them ever to enter A is if $v_j = \varphi_k(n_k, s_1)$ at some stage $s_1 > s_0$, for some $k < m$; when this happens, $v_j - 1$ would be chosen as t_{k,s_1+1} and would enter A. At subsequent stages, if $\mathcal{R}_k$ enumerates any more triggers into A, we would have $\varphi_k(n_k, s_1) > v_j$, so the triggers would also be $> v_j$. Thus this $\mathcal{R}_k$ enumerates at most one of the $2^{m+1} - 1$ elements into A.

Subsequently, another $\mathcal{R}_{k'}$ with $k' < m$ may have either $\varphi_{k'}(n_{k'}, s_2) = v_j$ or $\varphi_{k'}(n_{k'}, s_2) = v_j - 1$, and may enumerate $v_j - 2$ into A as a trigger. Moreover, if $\varphi_{k'}(n_{k'}, s_2) = v_j - 1$, then we might subsequently have $\varphi_{k'}(n_{k'}, s_3) = v_j$, in which case $\mathcal{R}_{k'}$ would enumerate another element of T into A. However, this $\mathcal{R}_{k'}$ enumerates at most these two elements of T into A. The inductive step is now clear: the third requirement $\mathcal{R}_{k''}$ $(k'' < m)$ might enumerate as many as four triggers from T into A, corresponding to the four numbers $v_j - 3, \ldots, v_j$ that could already be in A, and the next $\mathcal{R}$-requirement could enumerate eight triggers from T into A and so on. But only $\mathcal{R}_0, \ldots, \mathcal{R}_{m-1}$ can enumerate triggers from T into A, so the total number of triggers needed from T is at most $1 + \cdots + 2^m = 2^{m+1} - 1$, and we saw that T is at least this large. This proves Lemma 3.6. ∎

If $\lim_s t_{i,s}$ is finite, or if $\mathcal{R}_i$ never reaches substep 3, then we say that $\mathcal{R}_i$ is *finite-acting*. Otherwise, $\mathcal{R}_i$ is *infinite-acting*, and $\langle t_{i,s} \rangle_{s \in \omega}$ is an infinite nondecreasing unbounded sequence.

We now show by induction on i that every $\mathcal{L}_i$ (say with $i = \langle e, x\rangle$) and every $\mathcal{R}_i$ is eventually satisfied. Assume this is true for all higher-priority requirements, and let s_0 be a stage so large that none of them injures $\mathcal{L}_i$ after s_0, that $A_{s_0}\restriction l(j, s_0) = A\restriction l(j, s_0)$ for all $j < i$, and that for every finite-acting higher-priority requirement $\mathcal{R}_j$, we have $t_{j,s_0} = t_j$ (or, if $t_{j,s}$ is undefined for all s, then w_{j,s_0} is defined). If there is a stage $s_1 > s_0$ such that $\Phi_{e,s_1}^{A_{s_1}}(x)\downarrow$, let σ be the restriction of A_{s_1} to the use u of this computation. (If there are many such stages s_1, choose the one with the shortest corresponding σ, and leftmost in the dictionary order for that length. If s_1 does not exist, then $\mathcal{L}_{e,x}$ is satisfied.) Now there will be another stage $s_2 > s_1$ such that $t_{j,s} > u$ for every higher-priority infinite-acting $\mathcal{R}_j$ and every $s \geq s_2$. Between stages s_1 and s_2, moreover, no number $< u$ will have been removed from A: no higher-priority $\mathcal{L}$-requirement has acted; the lower-priority $\mathcal{L}$-requirements never change A below $l(i, s_1) \geq u$; and by our choice of s_1, $\mathcal{L}_i$ itself has no reason to redefine A. Hence at stage s_2, $\mathcal{L}_i$ will remove from A any elements added to A by $\mathcal{R}$-requirements in the meantime, setting $A_{s_2} = \sigma$, and will preserve this much of A forever after, so that $\Phi_e^A(x)\downarrow$. Thus $\mathcal{L}_i$ is satisfied, and never again injures any lower-priority requirements.

The requirement $\mathcal{R}_i$ (acting at the i-th substage of each stage) may stop permanently in either substep 2 or substep 3 above. If we stop in substep 2, then w_i stays in A forever (since no lower-priority $\mathcal{L}$-requirement can remove it, and any action by a higher-priority $\mathcal{L}$-requirement would redefine $w_{i,s}$). But the initial segment $A\restriction(1 + w_i)$ does not match that of the set (if any) order-computed by φ_i, for if it did, we would move to substep 3.

If we stop in substep 3 and $\mathcal{R}_i$ is finite-acting, fix s_0 such that $t_i = t_{s_0}$. Then for all $s \geq s_0$, the initial segment $A\restriction(1+\varphi_i(n_i, s))$ does not match that of the set order-computed by $\varphi_{i,s}$. Otherwise $\mathcal{R}_i$ is infinite-acting, so substep 3 must have enumerated the current $t_{i,s}$ into A at infinitely many stages s. Each time, that forced $\varphi_i(n_i, s)$ to increase in order for φ_i to approximate an initial segment of A again, because no lower-priority $\mathcal{L}$-requirement was allowed to remove $t_{i,s}$ from A until a new trigger $t_{i,s'} > t_{i,s}$ was defined. This ensures that $\lim_s \varphi_i(n_i, s) = \infty$. So in both cases, we see that φ is not a predecessor approximation function for A, and $\mathcal{R}_e$ is satisfied.

The only elements that ever enter A are ones that have never before entered A, by our choice of $w_{i,s}$ in substep 1 and our conditions on the triggers in substeps 2 and 3. The $\mathcal{L}$-requirements may remove elements from A, so A will be d.c.e. The same construction without the $\mathcal{L}$-requirements would

never remove elements from A, but would still satisfy all $\mathcal{R}$-requirements, hence would build a c.e. set that is not order-computable. ∎

We may use the same strategy to show that the join of two sets, even of two c.e. sets, need not preserve order-computability.

Theorem 3.7. *There exist order-computable c.e. sets A and B whose join*

$$A \oplus B = \{2n : n \in A\} \cup \{2n+1 : n \in B\}$$

is not order-computable.

Proof. We will build two computable linear orders $\prec^A$ and $\prec^B$ on the domain ω, each isomorphic to $(\omega, <)$, such that the sets A and B order-computed by these orders are c.e., yet the join $C = A \oplus B$ is not. We build C by the same method that was used for building W in Theorem 3.5, with witness elements $w_{e,s}$ and trigger elements $t_{e,s}$ that may be enumerated into either A or B (but not both) to force a potential predecessor approximation function φ_e for C to add new elements preceding the number n_e such that $\varphi_e(n_e, u) = 2w_e$. The requirements are as follows:

$\mathcal{P}_x$: The element x has only finitely many $\prec^A$-predecessors.
$\mathcal{Q}_x$: The element x has only finitely many $\prec^B$-predecessors.
$\mathcal{R}_e$: φ_e is not a predecessor approximation function for C.

$\mathcal{N}$-requirements impose finite restraints on enumeration into A and B at any given stage, and satisfying them will ensure that for each single x, the supremum of the lengths of these restraints is finite. We start with elements w_e already enumerated into A_0, so that $2w_e \in C_0$; these are chosen far enough apart from each other (as in Theorem 3.5) and will never be redefined, since the lowness requirements from that theorem do not apply here. The requirement $\mathcal{R}_e$ waits for φ_e to find an even element n_e with exactly $2w_e$ predecessors, then enumerates triggers into C to force φ_e to add more predecessors to n_e. However, we can satisfy $\mathcal{R}_e$ at stage $s+1$ either by enumerating the trigger $t_{e,s}$ into A_{s+1}, or by enumerating it into B_{s+1}, and this allows us to respect the negative requirements.

We start with $A_0 = \{w_e : e \in \omega\}$ and $B_0 = \emptyset$, where $w_0 = 0$ and $w_{e+1} = w_e + 2^{e+1}$. The real witness elements are therefore the numbers $2w_e$, which are the elements of C_0. At stage $s+1$, we set the restraint functions

$p(x, s+1)$ and $q(x, s+1)$ to be the number of predecessors of x at this stage under $\preceq^A$ and $\preceq^B$, respectively. These are the restraints imposed by $\mathcal{P}_x$ on A and by $\mathcal{Q}_x$ on B. Then, for each $i < s$ in turn, we follow steps 2 and 3 from page 22. The only change is that when we need to enumerate a trigger element $t_{i,s}$, we have to choose whether to enumerate it into A_{s+1} or B_{s+1}. Let $y = \mu x[p(x, s+1) \geq t_{i,s}]$ and $z = \mu x[q(x, s+1) \geq t_{i,s}]$. If $y > z$, then we enumerate $t_{i,s}$ into A_{s+1} (so $2t_{i,s}$ enters C_{s+1}); otherwise $t_{i,s}$ enters B_{s+1} and $2t_{i,s}+1$ enters C_{s+1}. In either case, this trigger forces φ_i to add more predecessors to n_i, since the element entering C_{s+1} is $< \varphi_i(n_i, u_{i,s})$. Moreover, we injured the lowest-priority negative requirement possible (under the convention that $\mathcal{P}_x$ has higher priority than $\mathcal{Q}_x$.) This completes substage i of stage $s+1$.

To finish stage $s+1$ itself, we must also consider $\prec^A$ and $\prec^B$. Suppose that $\prec^A$ currently has a elements in its domain. If $A_{s+1}\restriction a \neq A_s \restriction a$, then starting from the point where they disagree, we add enough fresh elements (say a'-many) to the order $\prec^A$ to make sure that $\prec^A$ on these elements does approximate $A_{s+1}\restriction (a+a')$. (If $A_s\restriction a = A_{s+1}\restriction a$, we add one more element to the right end of the order $\prec^A$, odd or even as dictated by A_{s+1}. This ensures that $\prec^A$ has a domain of at least $s+1$ elements at this stage.) Then we do the same for $\prec^B$ using B_{s+1}. This completes the construction.

The set $C = \cup_s C_s$ built by this construction is c.e. but not order-computable, by the same argument given for the set W in Theorem 3.5, since with $C = A \oplus B$, we now have even more space in C between witness elements $2w_e$ and triggers $2t_{i,s}$ or $2t_{i,s}+1$ than we did in W. It is important that for each i and s, at most one of the two triggers $2t_{i,s}$ and $2t_{i,s}+1$ ever enters C, of course.

However, we claim that $A = \cup_s A_s$ and $B = \cup_s B_s$ both are order-computable. The orders $\prec^A$ and $\prec^B$ were built to order-compute A_{s+1} and B_{s+1} at each stage $s+1$, so we need only show that each $\mathcal{P}_x$ and $\mathcal{Q}_x$ is injured only finitely often. (An injury to $\mathcal{P}_x$ is a change to $A_s\restriction p(x,s)$, of course.) By induction on x, suppose that no $\mathcal{P}_y$ or $\mathcal{Q}_y$ with $y < x$ is injured after stage s_0. Then we have $p(y,s) = p(y,s_0)$ and $q(y,s) = q(y,s_0)$ for all $s > s_0$ and all $y < x$. Let $m = \max(\{p(y,s_0) : y < x\} \cup \{q(y,s_0) : y < x\})$, and fix a stage $s_1 > s_0$ so large that every $\mathcal{R}_e$ with $w_e < m$ either has $t_{e,s_1} > m$ or never enumerates any more triggers into C after stage s_1. Hence no $\mathcal{R}$-requirement at all will try to enumerate any trigger $< m$ into A or B after stage s_1. So, if we need to put a trigger element $t < p(x, s+1)$ into A_{s+1} or B_{s+1} at some stage $s+1 > s_1$, then no $\mathcal{Q}_y$ with $y < x$ will stop us from putting it into B_{s+1}.

Thus $\mathcal{P}_x$ will never be injured after stage s_1. The same argument works for $\mathcal{Q}_x$, so all $\mathcal{P}$- and $\mathcal{Q}$-requirements are eventually satisfied. ∎

It is easy to adapt the requirements of Theorem 3.5 to build a c.e. set that is neither order-computable nor simple; just ensure that all elements $w_{i,s}$ are even and far enough apart that we can always choose even numbers as triggers. Then the complement of A clearly contains an infinite c.e. set. Also, it is straightforward to build both a simple order-computable set and a non-simple one. (Moreover, both can be made non-computable; Friedberg-Muchnik requirements and the negative requirements for order-computability all mesh easily either with simplicity requirements or with a construction on even numbers only. Follow the proof of Lemma 2.5.) The one difficult question about simple sets is resolved by the following lemma.

Lemma 3.8. *There exists a simple set A that is not order-computable.*

Proof. The proof adapts the construction of A in Theorem 3.5 with the same requirements $\mathcal{R}_e$, adding simplicity requirements

$$\mathcal{P}_e: \quad W_e \text{ infinite} \implies W_e \cap A \neq \emptyset.$$

(Of course, we leave out the $\mathcal{L}$-requirements, since we want A to be c.e., rather than low.)

We adapt the construction from Theorem 3.5. The difficulty is that we can no longer control which elements enter A, since $\mathcal{P}_e$ essentially says that we must give carte blanche to cofinitely many elements. However, $\mathcal{P}_e$ will only actually enumerate one of these cofinitely many elements into A, and so we can control the number of elements in a given interval (such as $[w_{i-1}, w_i]$) that could ever be enumerated into A. Then we need only ensure the presence of enough triggers (for each $i < e$) to add extra predecessors to each n_i and make $\varphi_i(n_i) > w_i$ as before. (Since no requirements are injured, w_i is never redefined; hence the subscript s is unnecessary.)

Thus, the only actual alteration to the construction from Theorem 3.5 is that in substep 1, we choose then new w_i to be greater than $2^{i+2} + i$ plus the greatest number yet seen in the construction. Each $\mathcal{P}_e$ is allowed to enumerate into A one number p_e, which must be $> w_e$. (Of course, that one number is all that is needed to satisfy $\mathcal{P}_e$ forever.) Any time $\varphi_{i,s}(n_i, s) = p_e$ for some $i < e$, we enumerate into A the greatest available trigger $< w_e$, without regard to the second of the two requirements in substep 2 or 3. If p_e

is large, there could be a number w_j between this trigger and p_e, but we have included enough available triggers below between w_{e-1} and w_e to ensure that p_e will never require enumeration of a trigger $< w_{e-1}$. The number $2^{i+2} + i$ above allows for i elements between w_{i-1} and w_i to be enumerated into A by requirements $\mathcal{P}_e$ with $e < i$, and then sufficiently many more triggers to move the value of $\varphi_j(n_j, s)$ (for every $j < i$) past w_i, past every trigger already enumerated into A, and past p_i. ∎

Another corollary of Theorem 3.5 strongly restricts the possible global connections between order-computability and most traditional measures of computability.

Corollary 3.9. *There exist computably isomorphic c.e. sets V and W such that V is order-computable and W is not.*

Proof. As noted after the proof of Theorem 3.5, it is easy to ensure that the W in the theorem is not simple. Apply Corollary 2.9 to the set W produced by Theorem 3.5. ∎

The following lemma is closely related; indeed the existence of the sets A and B would follow from Corollary 3.9. We include it because it gives a concrete example of a very straightforward computable permutation of ω that maps an order-computable set to a non-order-computable one.

Theorem 3.10. *Let $a_0 = 1$ and $a_n = (n+1)+(n+2)\cdot a_{n-1}$, and set $m_{0,0} = 1$ and $m_{e+1,0} = m_{e,0} + a_{e+1} + 1$ for all e. Define the computable permutation p of ω by letting $p(m_{e+1,0} - k) = m_{e,0} + k + 1$ for each e and each $k \leq a_{e+1}$. There exist Δ^0_2 sets A and B, computably isomorphic to each other via p, such that B is order-computable but A is not.*

Proof. We build A using the Khoussainov-Nies-Shore strategy to ensure that A is not the range of any limitwise-monotonic function. The requirements are precisely the $\mathcal{R}_e$ defined in the proof of Lemma 3.1. Simultaneously, we will build a computable linear order $(\omega, \prec)$ of type ω, such that the set B order-computed by $(\omega, \prec, E)$ is precisely the image $p(A)$. This will suffice to prove the theorem.

We call the interval $[m_{e,0} + 1, m_{e+1,0}]$ the $(e+1)$-st *p-segment*, and note that p maps each p-segment to itself, reversing the order of the elements. (We consider the two-element segment $[0, 1]$ to be the 0-th p-segment, with $p(0) = 1$ and $p(1) = 0$.)

At stage 0, we enumerate every $m_{e,0}$ into A_0. On the B-side, we immediately define an ordering $\prec$ of type ω on the infinite set $C = \{n \in \omega : n \not\equiv 3(\text{mod } 4)\}$, so that the number $2e$ has exactly $p(m_{e,0})$ predecessors under $\prec$, and the odd elements of C fill in the gaps. Thus the set B_0 order-computed by $(C, \prec, E)$ is just $p(A_0)$. The numbers not in C will all be added to the order later in the construction. The witness elements x_e and the corresponding $t_{e,0}$ are all undefined at stage 0.

At stage $2s$, let $A_s = \{m_{i,s} : i \in \omega\}$ and consider the least $e < s$ (if any) for which x_e and t_e are currently defined and $\varphi_{e,s}(x_e, t_{e,s}+1)\downarrow \geq \varphi_{e,s}(x_e, t_{e,s})$. We make $\varphi_{e,s}(x_e, t_{e,s}+1) \notin A_{s+1}$ and set $t_{e,s+1} = t_{e,s}+1$. The B-side has two possible actions, depending on whether we have changed A or not:

1. If there exists an $i > e$ such that $\varphi_{e,s}(x_e, t_{e,s}+1) = m_{i,s}$, then $m_{i,s}$ has been removed from A_{s+1}, and we define $m_{i,s+1} = m_{i,s} - 1 \in A_{s+1}$ and make x_i undefined. In this case the requirement $\mathcal{R}_i$ has been injured by the higher-priority $\mathcal{R}_e$. To parallel this A-change in B, we add the next available odd number k to the order $\prec$ so that k has exactly $p(m_{i,s})$ $\prec$-predecessors. This forces $p(m_{i,s}) \notin B_{s+1}$ and $p(m_{i,s+1}) \in B_{s+1}$. In turn, this change requires an A-change, since every element of the order to the right of k has now acquired a new predecessor. Therefore, for every $j > i$, we define $m_{j,s+1} = m_{j,s} - 1$ and make x_j undefined, thereby injuring $\mathcal{R}_j$. This leaves $p(m_{j,s+1}) = p(m_{j,s}) + 1$, since p inverts the ordering on each p-segment, so that A_{s+1} agrees with B_{s+1}.

2. If there is no such i, then there has been no change in A_{s+1}, so we do nothing further.

For each i, if $m_{i,s+1}$ is not defined by the process above, then $m_{i,s+1} = m_{i,s}$.

At stage $2s+1$, find the least $e < s$ for which x_e is not currently defined and there exist $x, t \leq s$ such that $\varphi_{e,s}(x,t)\downarrow = m_{e,s+1}$. (If there is no such e, we do nothing.) We choose the least such pair $\langle x, t\rangle$ corresponding to this e, let $x_e = x$ and $t_{e,s+1} = t$, set $m_{e,s+1} = m_{e,s} - 1 \in A_{s+1}$, and make $m_{e,s} \notin A_{s+1}$. As above, we add the next available odd number k to the order $\prec$ so that k has exactly $p(m_{e,s})$ $\prec$-predecessors. Thus $p(m_{e,s}) \notin B_{s+1}$ and $p(m_{e,s+1}) \in B_{s+1}$. For every $j > e$, we define $m_{j,s+1} = m_{j,s} - 1$ and make x_j undefined, thereby injuring $\mathcal{R}_j$ in order to preserve the agreement between A_{s+1} and B_{s+1}. This completes the construction.

Now m_e is redefined (with $m_{e,s+1} = m_{e,s} - 1$) when $m_{e,s}$ first enters the range of φ_e, and again each time a higher-priority $\mathcal{R}_i$ $(i < e)$ injures $\mathcal{R}_e$.

The numbers a_n are defined so that if the requirement $\mathcal{R}_{e-n}$ is never injured, then m_e will be reduced at most a_n times. For $n = 0$, this is clear, since if $\mathcal{R}_e$ is never injured, then m_e is reduced only when we first discover a pair $\langle x, t\rangle$ with $\varphi_e(x, t) = m_e$. The definition $a_n = (n + 1) + (n + 2) \cdot a_{n-1}$ was selected because if $\mathcal{R}_{e-n}$ is never injured, then $\mathcal{R}_{e-n}$ itself may reduce m_e once when x_{e-n} is first defined and $\varphi_{e-n}(x_{e-n}, t_{e-n,s}) = m_{e-n}$, then again when $\varphi_{e-n}(x_{e-n}, t_{e-n,s'}) = m_{e-n+1}$, and so on, for a total of $n + 1$ injuries. In between any two of those injuries, m_e can be reduced at most a_{n-1} times, by inductive hypothesis, and similarly before the first such injury and after the last, yielding a total of at most a_n reductions of m_e. Hence our choice of $m_{e,0} = m_{e-1,0} + a_e + 1$, which guarantees that for all s we have $m_{e-1,0} < m_{e,s}$. Therefore, $m_{e,s}$ always lies in the same p-segment of ω, and is mapped into that same p-segment by p. We see from this that only finitely many predecessors will be added below each element of $\prec$, so that $(\omega, \prec) \cong (\omega, <)$. Also, as noted above, the approximations $A_s = \{m_{e,s} : e \in \omega\}$ and $B_s = \{n \in \omega : (\exists x)[2x \text{ has exactly } n \prec_s\text{-predecessors}]\}$ satisfy $A_s = p(B_s)$ for every s. ■

The next theorem can be seen as a complement to Theorem 2.7 and Lemma 2.10.

Theorem 3.11. *There exists a Turing degree* $\mathbf{a} \leq_{\mathbf{T}} \mathbf{0}'$ *such that no set in* $\mathbf{a}$ *is order-computable.*

Proof. We build a computable approximation $\langle A_s\rangle_{s\in\omega}$ to a set A, and prove that $\deg(A)$ satisfies the theorem. The construction uses an infinite-injury argument, with the usual tree structure. For each e and each pair of oracle Turing functionals Ψ and Θ we will have a witness element $w_{e,\Psi,\Theta}$ satisfying the following requirements:

$\mathcal{R}_{\langle e,\Psi,\Theta\rangle}$: If φ_e appears to be a predecessor approximation function for a set B and $\Psi^A = B$ and $\Theta^B = A$, then $\lim_t \varphi_e(w_{e,\Psi,\Theta}, t) = \infty$ or A is computable.

Additional requirements will prevent A from being computable:

$$\mathcal{P}_e : \quad \varphi_e \neq A.$$

Together with the convergence of the approximation $\langle A_s\rangle_{s\in\omega}$, this clearly suffices to prove the theorem.

We write ψ and θ for the use functionals of Ψ and Θ. By definition $\Psi_s^X(n)\uparrow$ iff the use $\psi_s^X(n) = 0$. Moreover, on the domain of Ψ^X, ψ^X is always an increasing function, for any fixed oracle X.

The tree T that we use to build A has 4-branching nodes α for the $\mathcal{R}$-requirements and 2-branching nodes π for the $\mathcal{P}$-requirements. The nodes π have outcomes f and w, and use simple diagonalization to ensure that $\mathcal{P}_e$ is satisfied. The four successors of a node α, in order from left to right, will be denoted by

$$\alpha\hat{}\infty \ \prec\ \alpha\hat{}b \ \prec\ \alpha\hat{}a \ \prec\ \alpha\hat{}f$$

and this order extends lexicographically to all of T. If $\mathrm{lh}(\alpha) = \langle e, \Theta, \Psi\rangle = x$, then these outcomes attempt to satisfy $\mathcal{R}_x$. The outcome f denotes a finite win in which φ_e proves itself not to be a predecessor approximation function; b is the outcome in which the functional Ψ^A fails to compute the set B_e order-computed by φ_e, and similarly a is the outcome in which Θ^{B_e} fails to compute A. In the remaining outcome ∞, we will force $\lim_t \varphi_e(w_\alpha, t) = \infty$ for some number w_α, thereby ensuring that φ_e is not actually a predecessor approximation function for the set B_e. The basic module for accomplishing all this is explained below.

We partition the elements of ω into countably many infinite computable sets, each of which is the set of *toggle elements* for some node on T. Let α be a node on T of length $2x$. The specific toggle element used by α at stage s will be named $t_{\alpha,s}$, and α will move these elements into and out of A in order to satisfy $\mathcal{R}_x$. Similarly, nodes α of length $2e+1$ will move toggle elements into A to satisfy $\mathcal{P}_e$. We call this α an $\mathcal{R}_x$-node or a $\mathcal{P}_e$-node, accordingly, and we also sometimes speak simply of $\mathcal{R}$-nodes (those of even length) and $\mathcal{P}$-nodes (odd length). If α is on the true path P, i.e. the leftmost path through the tree such that every node on P acts at infinitely many stages, then α will succeed in doing this, and its successor on P will depend on which strategy it uses to satisfy its requirement.

We use $t'_{\alpha,s}$ to denote the least toggle element for α greater than $t_{\alpha,s}$. Of course, to make $A \leq_T \emptyset'$, we must ensure that every element enters or leaves A only finitely often.

For each e and each stage s such that $\varphi_{e,s}$ fails to prove that φ_e is not a predecessor approximation function, we find the greatest $l_{e,s}$ (possibly 0) such that $\varphi_{e,s}(\ \cdot\ ,\ l_{e,s})$ is a permutation of $\{0,\ldots,l_{e,s}-1\}$, and define $B_{e,s}$ to be the image of the even numbers under $\varphi_{e,s}(\ \cdot\ ,\ l_{e,s})$. The length of the approximation $B_{e,s}$ is therefore $l_{e,s}$. Thus, if φ_e really is a predecessor

approximation function, then $\lim_s l_{e,s} = \infty$ and $B_e = \lim B_{e,s}$ will exist. (If $\varphi_{e,s}$ does prove that φ_e is not a predecessor approximation function, then $B_{e,s}$ is undefined.) Moreover, every order-computable set B will appear as B_e for some e.

To initialize a node α at a stage s means to choose $t_{\alpha,s}$ to be a toggle element for α that is larger than any number yet seen in the construction, to remove from A all toggle elements for α, and to make $w_{\alpha,s}$ undefined.

We start with $A_0 = \emptyset$ and every $w_{\alpha,0}$ and $t_{\alpha,0}$ undefined. At stage 0 we initialize the empty node. At stage $s+1$, we have substages numbered 0 through (at most) s. At substage u, we act on behalf of some node α of length u, a successor of the node for which we acted at the preceding substage. This α will be said to be *eligible* to act at stage $s+1$, and $s+1$ will be called an *α-stage*.

At substage u, we let α be that node that was made eligible at the preceding substage. (At substage 0 the empty node is eligible.) If $\mathrm{lh}(\alpha) = 2e+1$ is odd, then we act on behalf of $\mathcal{P}_e$, setting $t_{\alpha,s+1} = t_{\alpha,s}$. If $\alpha\hat{}f$ was eligible at the last α-stage and α has not been initialized since, we simply make $\alpha\hat{}f$ eligible at this stage as well. Otherwise, if $\varphi_{e,s}(t_{\alpha,s})\downarrow = 0$, we enumerate $t_{\alpha,s}$ into A_{s+1} and make $\alpha\hat{}f$ eligible. Finally, if neither of these cases applies, then we make $\alpha\hat{}w$ eligible. This completes this substage.

If $\mathrm{lh}(\alpha) = 2x$ is even, we choose $\langle e, \Theta, \Psi\rangle = x$. We act according to the least-numbered of the following eight steps that applies:

1. If $\varphi_{e,s}$ proves that it is not a predecessor approximation function (so that $B_{e,s}$ is undefined), then we make $\alpha\hat{}f$ eligible and end this substage.

2. If $w_{\alpha,s}$ is undefined and $\theta_s^{B_{e,s}}(t_{\alpha,s}) \geq l_{e,s}$, then again we make $\alpha\hat{}f$ eligible and end this substage.

3. If $\theta_s^{B_{e,s}}(t) = 0$ for any $t \leq t'_{\alpha,s}$, then we make $\alpha\hat{}a$ eligible and end this substage.

4. If $w_{\alpha,s}$ is undefined and $0 < \theta_s^{B_{e,s}}(t_{\alpha,s}) < l_{e,s}$, we choose $w_{\alpha,s+1}$ to be a number such that $\varphi_{e,s}(w_{x,s+1}, l_{e,s})\downarrow \geq \theta_s^{B_{e,s}}(t_{\alpha,s})$, let $t_{\alpha,s+1} = t_{\alpha,s}$, initialize every node $\beta \supsetneq \alpha$, and end the stage, by making no node eligible to act at the next substage. (By the conditions given, such a number $w_{x,s+1}$ must exist.)

5. If $\Theta_s^{B_{e,s}}\restriction t'_{\alpha,s} \neq A_s\restriction t'_{\alpha,s}$, then we end this substage, with $\alpha\hat{}a$ eligible to act at the next substage.

6. If $\Theta_s^{B_{e,s}} \restriction t'_{\alpha,s} = A_s \restriction t'_{\alpha,s}$, but $\Psi_s^{A_s} \restriction \theta_s^{B_{e,s}}(t'_{\alpha,s}) \neq B_{e,s} \restriction \theta_s^{B_{e,s}}(t'_{\alpha,s})$, then we end this substage, with $\alpha\hat{}b$ eligible to act at the next substage.

7. If $t_{\alpha,s} \notin A_s$ (and none of the above cases applies), then we put $t_{\alpha,s}$ into A_{s+1}. Then we make $\alpha\hat{}\infty$ eligible to act at the next substage. However, we initialize all nodes of length at least $n + \mathrm{lh}(\alpha)$ that extend α, where n is the number of toggle elements for α that are $< t_{\alpha,s}$. (Our rule is that these initialized nodes cannot be eligible later in this same stage.) This completes this substage.

8. Otherwise we have $t_{\alpha,s} \in A_s$ and $\Psi_s^{A_s} \restriction \theta_s^{B_{e,s}}(t'_{\alpha,s}) = B_{e,s} \restriction \theta_s^{B_{e,s}}(t'_{\alpha,s})$ and $\Theta_s^{B_{e,s}} \restriction t'_{\alpha,s} = A_s \restriction t'_{\alpha,s}$. In this case we set

$$t_{\alpha,s+1} = \begin{cases} t'_{\alpha,s} & \text{if } \varphi_{e,s}(w_{\alpha,s}, l_{e,s}) > \theta_s^{B_{e,s}}(t'_{\alpha,s}) > 0 \\ t_{\alpha,s} & \text{if not,} \end{cases}$$

   We set $t_{\alpha,s} \notin A_{s+1}$, make $\alpha\hat{}a$ eligible to act at the next substage, and end this substage.

When we have finished the last eligible substage, we initialize all nodes β such that some α to the left of β in T was eligible to act at this stage. We also initialize all nodes of length $s+1$. Any element not mentioned in the foregoing instructions is preserved at stage $s+1$ (so $w_{\beta,s+1} = w_{\beta,s}$, etc.). This completes the construction.

We now describe the basic module for satisfying a requirement $\mathcal{R}_x$ via a node α of length $2x = 2 \cdot \langle e, \Theta, \Psi \rangle$ on the true path P. The node $\alpha\hat{}f$ represents the outcome in which φ_e proves itself not to be a predecessor approximation function. (If this never occurs, but $\lim_s l_{e,s}$ is finite, then again φ_e cannot be a predecessor approximation function, and f may be the outcome, using Step 2 above.) The node $\alpha\hat{}a$ represents the outcome in which $A \neq \Theta^{B_e}$. This can occur either in Step 3, if Θ^{B_e} simply fails to converge on a toggle element for α, or in Step 5, in which Θ^{B_e} converges but disagrees with A or in Step 8, where we actively create a disagreement by removing $t_{\alpha,s}$ from A. If any of these steps holds infinitely often (for some $t_\alpha = \lim_s t_{\alpha,s}$), then clearly $\mathcal{R}_x$ is satisfied. Similarly, the node $\alpha\hat{}b$ represents a disagreement between B and Ψ^A.

Apart from these obvious ways to satisfy $\mathcal{R}_x$, the strategy for α is to drive $\lim_s \varphi_e(w_\alpha, s)$ to infinity. To do so, α waits for $\Theta_s^{B_{e,s}} \restriction t'_{\alpha,s} = A_s \restriction t'_{\alpha,s}$ and $\Psi_s^{A_s} \restriction \theta_s^{B_{e,s}}(t'_{\alpha,s}) = B_{e,s} \restriction \theta_s^{B_{e,s}}(t'_{\alpha,s})$ and ensures that $\theta_{\alpha,s}^{B_{e,s}}(t_{\alpha,s}) < \varphi_{e,s}(w_{\alpha,s}, l_{e,s})$.

Then, by moving its current toggle element $t_{\alpha,s}$ into A in Step 7, α can force B_e to change somewhere below $\theta_{\alpha,s}^{B_{e,s}}(t_{\alpha,s})$, meaning that $\varphi_e(w_{\alpha,s}, l_{e,s})$ must have increased, since φ_e is the predecessor approximation function for B_e. (This is the outcome ∞.) If this change occurs and the functionals Θ and Ψ again appear to compute A and B_e from each other, then α removes $t_{\alpha,s}$ from A again in Step 8.

Of course, α cannot move $t_{\alpha,s}$ into and out of A infinitely often, since A must be Δ_2^0. So α always tries to switch its current toggle element to a larger one (at the start of Step 8). We know that α succeeds in making $\varphi_e(w_\alpha, l_{e,s})$ larger and larger, but it must also ensure that $\theta^{B_e}(t'_{\alpha,s})$ does not go to infinity as well, so that eventually it will be able to switch from $t_{\alpha,s}$ to $t'_{\alpha,s}$. This is where α uses the assumption that $B_e = \Psi^A$: by preserving A up to $\psi^A(\theta^{B_e}(t'_{\alpha,s}))$, it forces B_e to return to its original configuration up to $\theta^{B_e}(t'_{\alpha,s})$, so that the use of $\Theta^{B_e}(t'_{\alpha,s})$ does indeed stay bounded. In fact, α does not even begin toggling $t_{\alpha,s}$ until Θ^{B_e} has converged on $t'_{\alpha,s}$. Thus, by alternately toggling and restoring A, α guarantees that it can keep switching to new toggle elements, so that $\lim_s A_s$ converges.

The requirements fit together very much in the usual way for a tree construction. The *true path* P contains the leftmost node α at each level of T such that α is eligible at infinitely many stages and all predecessors of α lie on P. Lemma 3.13 below shows that P is indeed an infinite path through T.

We now give the formal proof that A is Δ_2^0 and satisfies every $\mathcal{R}_x$ and $\mathcal{P}_e$. We write $w_\alpha = \lim_s w_{\alpha,s}$; this exists for all $\alpha \in P$. We also write $t_\alpha = \lim_s t_{\alpha,s}$ when this limit exists, but it will fail to exist when $\alpha\hat{}\infty \in P$ (and also for all nodes to the right of P). The key to the proof is the very first lemma.

Lemma 3.12. *If $\alpha\hat{}\infty \in P$, then there are infinitely many stages s at which $t_{\alpha,s+1} > t_{\alpha,s}$.*

Proof. Clearly α has even length, say $2x$. Suppose the statement is false, and let s_0 be the last stage at which either initialization or Step 8 changes $t_\alpha = t_{\alpha,s_0}$. (Notice that the construction never makes $t_{\alpha,s+1} < t_{\alpha,s}$; equality and increase are the only possibilities.) Let n be the number of toggle elements for α that are $< t_\alpha$, and let $m = n + \mathrm{lh}(\alpha)$. We prove by induction on n that if every $\gamma \supset \alpha$ with $\mathrm{lh}(\gamma) - \mathrm{lh}(\alpha) \geq n$ is initialized at every α-stage, and $\alpha\hat{}\infty \in P$, then $t_{\alpha,s+1} > t_{\alpha,s}$ at infinitely many stages s.

Let $s+1$ be any $\alpha\hat{}\infty$-stage $> s_0$. Step 7 applies, so we enumerate t_α into A_{s+1}. Before it can be enumerated into A again, Step 8 must apply at

a subsequent α-stage $s'+1$ in order for it to be removed from A. When this happens, we must have $\varphi_{e,s'}(w_\alpha, l_{e,s'}) \leq \theta_{s'}^{B_{e,s'}}(t'_\alpha)$, since otherwise $t_{\alpha,s'+1} > t_{\alpha,s'}$.

Now when $\alpha\hat{}\infty$ is eligible at $s+1$, we initialize all nodes to its right. The same process at the next $\alpha\hat{}\infty$-stage $s''+1$, along with the enumeration of t_α back into $A_{s''+1}$, restores $A_{s''+1}(t) = A_{s+1}(t)$ for all t except possibly $t = t_{\beta,s+1}$ for some nodes β in the set

$$S = \{\beta : \alpha\hat{}\infty \subseteq \beta \text{ and } \mathrm{lh}(\beta) < m\},$$

because α also initializes all nodes $\gamma \supset \alpha$ of length $\geq m$ at each $\alpha\hat{}\infty$-stage. Consider a node $\beta \in S$ and $t = t_{\beta,s+1}$. If β does not lie on P, then it is either initialized infinitely often or eligible only finitely often, so t enters A only finitely often. If $\beta \in P$ but $\beta\hat{}\infty \notin P$ or $\mathrm{lh}(\beta)$ is odd, then t only enters A finitely often, because Step 7 only applies to an $\mathcal{R}$-node β at finitely many β-stages. Finally, if $\beta\hat{}\infty \in P$, then $t_{\beta,s+1}$ must increase infinitely often, by our inductive hypothesis, because every node $\gamma \supseteq \beta$ with $\mathrm{lh}(\gamma) - \mathrm{lh}(\beta) \geq n-1$ satisfies $\mathrm{lh}(\gamma) - \mathrm{lh}(\alpha) \geq n$, so is initialized (by α) at every β-stage.

Now finitely many nodes $\beta \in S$ may have a permanent toggle element t_β. (Such β either lie to the left of P, or lie on P but have odd length, or $\beta\hat{}\infty \notin P$.) Fix an $\alpha\hat{}\infty$-stage $s_1 > s_0$ at which all these nodes β have already moved t_β into or out of A for the last time.

Let β_0 be the greatest node in S such that $\beta_0\hat{}\infty \in P$. Then we eventually reach a $\beta_0\hat{}\infty$-stage $r+1 > s_1$ after which no $t < \varphi^{A_{s_1}}(\theta^{B_{e,s_1}}(t'_\alpha))$ ever again enters or leaves A. (If there is no such β_0, then let $r+1 = s_1$.) Now stage $r+1$ is a $\beta\hat{}\infty$-stage for every $\beta \in S$ such that $\beta\hat{}\infty \in P$, and so $t_{\beta,r} \notin A_r$ for all such β. Fix a stage $s_2 > r$ such that $t_{\beta,s_2} > \varphi^{A_r}(\theta^{B_{e,r}}(t'_\alpha))$ for all $\beta \in S$ with $\beta\hat{}\infty \in P$. Then for each stage $s+1 \geq s_2$, we have $A_s{\restriction}\varphi^{A_r}(\theta^{B_{e,r}}(t'_\alpha)) = A_r{\restriction}\varphi^{A_r}(\theta^{B_{e,r}}(t'_\alpha))$, since every $\beta \in S$ with $\beta\hat{}\infty \in P$ has left all its previous toggle elements out of A when it switched to larger ones or when it was initialized. Therefore, for all α-stages $s+1 \geq s_2$ at which Step 7 or 8 applies, we have

$$B_{e,s}{\restriction}u = \Psi^{A_s}{\restriction}u = \Psi^{A_r}{\restriction}u = B_{e,r}{\restriction}u$$

where $u = \min(\theta^{B_{e,r}}(t'_\alpha), \theta^{B_{e,s}}(t'_\alpha))$. Hence $\theta_s^{B_{e,s}}(t'_\alpha) = \theta_r^{B_{e,r}}(t'_\alpha)$. But as Step 7 continues to apply at infinitely many α-stages $s+1$, $\varphi_e(w_\alpha, l_{e,s})$ must eventually grow larger than $\theta^{B_{e,r}}(t'_\alpha)$, by the argument given in the basic module. When this happens, Step 8 will choose $t_{\alpha,s+1}$ to be $> t_\alpha$, contradicting our hypothesis. ∎

Lemma 3.13. *The true path P is infinite.*

Proof. We induct on the length of nodes on P, starting with the empty node, which is eligible at every stage, and showing that every node on P has a successor that is eligible infinitely often and initialized only finitely often. Since every node has at most four immediate successors, we need only worry about stages $s+1$ at which the construction either makes fewer than $s+1$ nodes eligible, or initializes its own successors. Steps 4 and 7 for $\mathcal{R}$-nodes do create such cases. However, by induction on the length of the node, each node $\alpha \in P$ will go through Step 4 only finitely many times, after which $w_{\alpha,s}$ will remain unchanged. Moreover, initializations by α using Step 7 are a concern only if $\alpha\hat{}\infty$ is eligible infinitely often In this case $\alpha\hat{}\infty \in P$, so Lemma 3.12 shows that $\lim_s t_{\alpha,s} = \infty$. Hence the number n in Step 7 grows without bound at $\alpha\hat{}\infty$-stages, and each node $\beta \supseteq \alpha\hat{}\infty$ on P is initialized by α only finitely many times. ∎

Lemma 3.14. *For every $t \in \omega$, $\lim_s A_s(t)$ converges.*

Proof. A toggle element for a $\mathcal{P}$-node π can enter A at most once; it might later leave A due to initialization, but it will never again be chosen as $t_{\pi,s}$.

For $\mathcal{R}$-nodes α, notice that α moves only its own current toggle element $t_{\alpha,s}$ into or out of A at any stage (and then only in Step 7 or 8). Of course, no number is a toggle element for more than one node. Say that t is a toggle element for the $\mathcal{R}$-node γ. If γ lies to the left of the true path, then γ is eligible at only finitely many stages, so $\lim_s A_s(t)$ must converge. If γ lies to the right of the true path, then it is initialized infinitely often, with $t_{\gamma,s}$ being chosen large each time, so every $t_{\gamma,s}$ enters A only finitely often before γ is initialized again. Finally, suppose γ lies on P. If $\gamma\hat{}\infty \in P$, then by Lemma 3.12, $t_{\gamma,s}$ increases infinitely often, so each individual toggle element for γ enters A only finitely often. If $\gamma\hat{}\infty \notin P$, then there are only finitely many $\gamma\hat{}\infty$-stages, and these are the only stages at which γ enumerates any element into A. ∎

Lemma 3.15. *Every requirement $\mathcal{P}_e$ is satisfied by this construction.*

Proof. Let $\pi \in P$ be a node of length $2e+1$, and let $t_\pi = t_{\pi,s_0}$, where s_0 is a stage after which π is never initialized. If φ_e were the characteristic function of A, then $\varphi_{e,s}(t_\pi)\downarrow$ for some $s > s_0$. But π puts t_π into A precisely if $\varphi_e(t_\pi) = 0$, so this is impossible. ∎

Lemma 3.16. *Every requirement $\mathcal{R}_{\langle e,\Theta,\Psi\rangle}$ is satisfied by this construction.*

Proof. Let α be the node on P of length $2x = 2 \cdot \langle e, \Theta, \Psi\rangle$, and assume α is never initialized after stage s_0. Suppose that φ_e is a predecessor approximation function for a set B_e and $\Psi^A = B_e$ and $\Theta^{B_e} = A$. Then w_α must eventually be chosen by α using Step 4 and never again changed, since $l_{e,s}$ grows arbitrarily large and $\Theta^{B_e}(t_{\alpha,s_0})$ must converge. Thereafter Steps 1, 2, 3, and 4 never again apply. Also, Steps 5 and 6 may apply at some α-stages, but since neither of them ever changes $t_{\alpha,s}$, eventually we will get $\Theta_s^{B_{e,s}} \upharpoonright t'_{\alpha,s} = A_s \upharpoonright t'_{\alpha,s}$ and $\Psi_s^{A_s} \upharpoonright \theta_s^{B_{e,s}}(t'_{\alpha,s}) = B_{e,s} \upharpoonright \theta_s^{B_{e,s}}(t'_{\alpha,s})$, at which stages Steps 7 and 8 will apply in alternation (since Step 7 puts $t_{\alpha,s}$ into A_{s+1} and Step 8 either removes it or redefines $t_{\alpha,s+1}$ to be a new toggle element not yet in A). Thus $\alpha\hat{}\infty \in P$. But we have seen that then $t_{\alpha,s}$ increases at infinitely many α-stages via Step 8. Therefore there are infinitely many s for which $\varphi_{e,s}(w_\alpha, l_{e,s}) > \theta^{B_{e,s}}(t'_{\alpha,s})$. But since φ_e is a predecessor approximation function for B_e, we know that $\lim_s \varphi_{e,s}(w_\alpha, l_{e,s})$ must be finite. Call this limit u. Now for every $t \in \omega$ there is a stage s such that $B_e \upharpoonright \theta^{B_e}(t) = B_{e,s} \upharpoonright \theta^{B_e}(t)$ and so $\theta^{B_{e,s}}(t) = \theta^{B_e}(t)$. But use functions are by definition nondecreasing, so $\theta^{B_{e,s}}(t) \leq \theta^{B_{e,s}}(t'_{\alpha,s}) < u$, assuming we choose s large enough that $t \leq t'_{\alpha,s}$. Thus $A = \Theta^{B_e}$ is actually a computable set, being equal to $\Theta^{B_e \upharpoonright u}$, and $\mathcal{R}_x$ is satisfied. (Of course, Lemma 3.15 excludes the possibility of A being computable, so in fact $\lim_s \varphi_{e,s}(w_\alpha, l_{e,s}) = \infty$.)

This completes the proof of Lemma 3.16, and also of Theorem 3.11. ∎

A close reading of the preceding proof yields more information. Our thanks go to the referee of this paper for pointing out this corollary, which shows that Theorem 2.7 is best possible.

Corollary 3.17. *The degree* **a** *from Theorem 3.11 is $(\omega + 1)$-c.e. Hence there exists an $(\omega + 1)$-c.e. set which is not Turing-equivalent to any order-computable set.*

Proof. We show that the set A constructed in Theorem 3.11 is $(\omega + 1)$-c.e. The only elements t to enter A are the toggle elements, say $t = t_{\alpha,s}$, in Step 7 of the construction. At that stage s we know the use $\theta^{B_{e,s}}(t'_{\alpha,s})$, of course, and the restoration of B_e keeps this use fixed right from the first time t enters A. So we don't know a bound on the number of moves of t in and out of A until that first entrance of t into A, but as soon as it takes place, we can say with certainty that $2 + \theta^{B_{e,s}}(t'_{\alpha,s}) - \varphi_{e,s}(w_{\alpha,s}, l_{e,s})$ is a bound, because the

putative predecessor approximation function φ_e must increase on $w_{\alpha,s}$ each time t enters or leaves A, and as soon as it becomes $> \theta^{B_{e,s}}(t'_{\alpha,s})$, Step 8 will switch from $t = t_{\alpha,s}$ to $t'_{\alpha,s}$ and t will never enter A again. Of course, if α is injured before or after this happens, then after initialization the new toggle element for α will be chosen much larger than t, so t will never enter A again. ∎

Corollary 3.18. *There exists a Δ^0_2 set B such that each of B and $\overline{B}$ is the range of a limitwise monotonic function, yet B is not order-computable.*

Proof. Choose any set A from the degree **a** constructed in Theorem 3.11. Let

$$B = E \oplus A = \{2n : n \in E\} \cup \{2n+1 : n \in A\},$$

where as usual E is the set of even numbers (though used here for an unusual purpose). Then B and $\overline{B}$ both lie in degree **a**, hence cannot be order-computable. However, it is simple to build a limitwise monotonic function with range B. First set $f(4x, s) = 4x$ for all x and s. Then, whenever an element n enters the computable approximation A_s, we choose a fresh x and set $f(x, t) = 2n + 1$ for $t = s, s + 1, \ldots$ as long as $n \in A_t$. If we find some $t > s$ with $n \notin A_t$, we set $f(x, t) = 4n$. If later n re-enters A, a new x is assigned to it. Then $\lim_s f(\cdot, s)$ is total with range B. A similar construction works for $\overline{B}$. ∎

Notice that in the proof of Corollary 3.18, the only property of A used in the construction of B was that $A \leq_T \emptyset'$. Thus this proof also establishes that every Δ^0_2 degree contains the range of some limitwise-monotonic function.

We now briefly consider randomness issues. For a finite binary string σ, let $K(\sigma)$ be the prefix-free Kolmogorov complexity of σ. (See [7] for more on Kolmogorov complexity.) Recall that a set A is 1-random iff there is a c such that $K(A\upharpoonright n) \geq n - c$ for all n. We will need the following well-known result (often known as the "Kraft-Chaitin Theorem"). (See [1] for a proof of this result and more on the interaction between algorithmic randomness and computability theory.)

Theorem 3.19. *Let $\{\langle n_i, \sigma_i\rangle : i \in \omega\}$ be a c.e. set of pairs (which we call* axioms*) consisting of a natural number and a binary string such that $\sum_i 2^{-n_i} \leq 1$. Then there is a d such that $K(\sigma_i) \leq n_i + d$ for all i.*

From this we derive the following proposition, which was also proven independently by André Nies (unpublished result).

Proposition 3.20. *No* 1*-random set is order-computable.*

Proof. Let $(\omega, \prec)$ be a computable ordering of type ω and let A be the set order-computed by $(\omega, \prec, E)$. Let $\prec_s$ be $\prec$ restricted to $0, \ldots, s$, and for $n \leq s$ let the binary string σ_s^n of length $n+1$ be defined by

$$\sigma_s^n(i) = \begin{cases} 0 & \text{if the } i\text{th element of } \prec_s \text{ is odd} \\ 1 & \text{if the } i\text{th element of } \prec_s \text{ is even.} \end{cases}$$

There are two cases to consider. The first is handled by the following lemma.

Lemma 3.21. *Suppose there is a* c *such that every* n *has at most* $n + c$ *predecessors in* $\prec$*. Then* A *is computable.*

Proof. For each n simultaneously, proceed as follows. Let a be the last element of $\prec_n$. Since $a \leq n$, we know that a has at most $n + c$ predecessors, so the c.e. set S_n of all σ_s^n for $s \geq n$ has size at most $c + 1$. Note that S_n contains $A\restriction(n+1)$.

Let k be the greatest number such that $|S_n| = k$ for infinitely many n and let m be such that $|S_n| \leq k$ for $n > m$. Build a computable tree T as follows. Start with $2^{<\omega}$. Whenever an $n > m$ is found with $|S_n| = k$, prune the tree to remove all paths extending $2^{n+1} - S_n$. Since T is a computable tree of finite width, all paths of T are computable. But $A\restriction(n+1)$ is in every S_n, so A is a path of T. ∎

Now suppose that for every c there is an n with more than $n + c$ many predecessors in $\prec$. Enumerate axioms as follows. For all $c > 0$ simultaneously, look for $n > c$ and s such that n has $k > n + c$ predecessors in $\prec_s$. Enumerate an axiom $\langle n, \sigma_s^k \rangle$. For every $t > s$, if n has $j - 1$ predecessors in $\prec_{t-1}$ but j predecessors in $\prec_t$ (in other words, if $t \prec n$), enumerate an axiom $\langle n + j - k, \sigma_s^j \rangle$. Note that if l is the number of predecessors of n in $\prec$ then we eventually enumerate an axiom $\langle n + l - k, A\restriction(l+1) \rangle$, and $n + l - k < l - c$.

The total measure corresponding to all axioms enumerated for c is less than $\sum_{i \geq n} 2^{-i} = 2^{-n+1} \leq 2^{-c}$ (since $n > c$). So the total measure corresponding to all enumerated axioms is less than $\sum_{c>0} 2^{-c} = 1$, and hence by

Theorem 3.19 there is a d such that for all enumerated axioms $\langle m, \sigma \rangle$ we have $K(\sigma) \leq m + d$. But, as noted above, for every c there is an l and an $i < l - c$ such that we eventually enumerate an axiom $\langle i, A\restriction(l+1)\rangle$, whence $K(A\restriction(l+1)) < l - c + d$. Thus A cannot be 1-random. ∎

We note that this lemma probably does not follow automatically from the fact that every order-computable set is the range of a limitwise-monotonic function. Indeed, we believe that there do exist 1-random sets that are themselves the ranges of such functions.

4 Other Results and Open Questions

In [2], Downey, Khoussainov, J. Miller, Nies, and Yu consider questions about order-computability from a somewhat different standpoint. Recall the definition of the degree spectrum of a relation R (of arbitrary arity) on a computable structure $\mathcal{A}$:

$$\mathrm{DgSp}_{\mathcal{A}}(R) = \{\deg(S) : (\exists \mathcal{B} \leq_T \emptyset)(\mathcal{B}, S) \cong (\mathcal{A}, R)\}.$$

(Normally R is not in the language of the structure $\mathcal{A}$; otherwise its spectrum contains only the degree $\mathbf{0}$.) Intuitively, this measures how complex the relation R can be made. We require that $\mathcal{B}$ be computable as a way of guaranteeing that $\mathrm{DgSp}_{\mathcal{A}}(R)$ measure the possible complexity of R uniquely; one is not allowed to make R more complicated by making the functions and relations of $\mathcal{A}$ noncomputable.

Thus a set A is order-computable iff the degree $\mathbf{0}$ lies in the degree spectrum of A as a unary relation on the structure $(\omega, <)$. The paper [2] asks more general questions about this degree spectrum, for arbitrary c.e. sets A: must it be upwards-closed under $\leq_T$ in the Δ^0_2-degrees, must it contain a low degree, etc. One pleasing result is a strong extension of our Theorem 2.6: the paper shows that a c.e. degree $\mathbf{a}$ is high iff $\mathbf{a}$ contains a non-order-computable c.e. set.

It would also be possible to ask about the spectrum of the *structure* $(\omega, <, A)$, for an arbitrary set A. By definition, the spectrum of (the isomorphism type of) a structure $\mathcal{M}$ is the set of all Turing degrees of isomorphic copies of $\mathcal{M}$:

$$\mathrm{Spec}(\mathcal{M}) = \{\deg(\mathcal{N}) : \mathcal{N} \cong \mathcal{M}\}.$$

(By our conventions, $\mathcal{N}$ ranges only over structures with domain ω.) Intuitively, this measures the intrinsic difficulty of computing a copy of $\mathcal{M}$: each degree $\mathbf{d}$ in $\mathrm{Spec}(\mathcal{M})$ is smart enough to build a structure isomorphic to $\mathcal{M}$. It is immediate that $\mathrm{DgSp}_{(\omega,<)}(A) \subseteq \mathrm{Spec}(\omega, <, A)$, and that the degree $\mathbf{0}$ lies in one iff it lies in the other, but the reverse inclusion need not hold. (For instance, the spectrum of any finite set F as a relation on $(\omega, <)$ would just be $\{\mathbf{0}\}$, whereas the spectrum of the structure $(\omega, <, F)$ would contain all Turing degrees, by a result of Knight in [6].)

Another collection of questions concerns the derived order, as defined on p. 5. The main questions to be asked about the derived order (for a Δ^0_2 set A that may or may not be order-computable) involve its order type. If it is of type ω, of course, then A is order-computable. Indeed, if $\mathcal{L}$ has an initial segment of order type $\omega + 1$, then A is order-computable, since if y is the rightmost element of this segment, then $(\{x \prec y\}, \prec, \{x \in E : x \prec y\})$ is a computable copy of $(\omega, <, A)$. In particular, if the derived order is well-ordered, then A is order-computable. Moreover, a set A is order-computable iff there exists a computable approximation to A for which the above process builds an order of type ω. (The forward direction follows by taking the computable approximation given by the predecessor approximation function for the order that order-computes A.)

We can still ask, for order-computable A, which other order types might be built using other computable approximations. Of more interest are sets A that are not order-computable. For such an A, we would like to find a "smallest" linear order $\mathcal{L}$ such that some computable approximation to A gives rise to an order $\prec$ of type $\mathcal{L}$. This would measure, in some sense, how close A is to being order-computable. As noted above, however, this $\mathcal{L}$ cannot be an ordinal, which makes it difficult to define "smallest" rigorously. What (countable) order types are possible? Is there an order type that would characterize any interesting properties of sets? (For instance, recall the characterization of the non-high c.e. degrees from [2], given above, as those containing an order-computable c.e. set. For other order types $\mathcal{L}$, what can we say about the collection of sets (or the collection of their degrees) with a computable approximation whose derived order is of type $\mathcal{L}$?)

We close with a few other specific questions arising from results in this paper.

- Can the non-order-computable degree in Theorem 3.11 be made low?
- Shore has asked whether we can use high permitting to build a noncom-

putable set A below an arbitrary high set C such that every $B \leq_T A$ is order-computable. (If we restrict our attention to c.e. sets, then the answer is positive, because $\deg(C)$ must have a low noncomputable c.e. degree below it.) Also, what if C is high but $\deg(C)$ is not c.e.?

- Can we do the same below a not-necessarily-high set C?
- Among well-known Δ^0_2 sets, which are order-computable? Simpson has asked about sets such as K and K_0 (as defined in [8]) specifically. Chaitin's set Ω is not order-computable, of course, by Proposition 3.20.

References

[1] R. Downey, D. R. Hirschfeldt, A. Nies, and S. A. Terwijn, Calibrating Randomness, *Bulletin of Symbolic Logic* **12** (2006), 411–491.

[2] R. Downey, B. Khoussainov, J. Miller, A. Nies, and L. Yu, Degree Spectra of Unary Relations on $(\omega, \leq)$, to appear.

[3] V. S. Harizanov, Pure Computable Model Theory, *Handbook of Recursive Mathematics*, vol. 1 (Amsterdam: Elsevier, 1998), 3–114.

[4] N. G. Khisamiev, Criterion for constructivizability of a direct sum of cyclic p-groups, *Izv. Akad. Nauk Kazakh. SSR Ser. Fiz.-Mat.* 1981, 51–55, 86.

[5] B. Khoussainov, A. Nies, and R.A. Shore, Computable Models of Theories with Few Models, *Notre Dame Journal of Formal Logic* **38** (1997), 165–178.

[6] J. F. Knight, Degrees Coded in Jumps of Orderings, *Journal of Symbolic Logic* **51** (1986), 1034–1042.

[7] M. Li and P. Vitányi, *An Introduction to Kolmogorov Complexity and its Applications*, 2nd ed., Graduate Texts in Computer Science, (Springer-Verlag, 1997).

[8] R. I. Soare, *Recursively Enumerable Sets and Degrees* (New York: Springer-Verlag, 1987).

Department of Mathematics
University of Chicago
5734 S. University Ave.
Chicago, IL 60637 U.S.A.
E-mail: drh@math.uchicago.edu

Department of Mathematics
Queens College – C.U.N.Y.
65-30 Kissena Blvd.
Flushing, New York 11367 U.S.A.
Ph.D. Program in Computer Science
The Graduate Center of C.U.N.Y.
365 Fifth Avenue
New York, New York 10016 U.S.A.
E-mail: Russell.Miller@qc.cuny.edu

Department of Mechanics and Mathematics
Novosibirsk State University
Pirogova st., 2
Novosibirsk 630090 Russia
E-mail: podz@math.nsc.ru